Feature and Target Engineering

Data Set

h2o

```
ames <- AmesHousing::make_ames()
ames.h2o <- as.h2o(ames)
```

stratified (*Sale_Price*) training sample

```
set.seed(123)

split <- initial_split(ames, prop = 0.7,
                       strata = "Sale_Price")

ames_train <- training(split)
ames_test <- testing(split)
```

log transformation (Sale_Price)

```
ames_recipe <- recipe(Sale_Price ~ ., data = ames_train) %>%
   step_log(all_outcomes())

ames_recipe
```

```
Data Recipe

Inputs:

      role #variables
   outcome          1
 predictor         80

Operations:

Log transformation on all_outcomes
```

Box-Cox transformation (example)

```
lambda <- 3

y <- forecast::BoxCox(10, lambda)

inv_box_cox <- function(x, lambda) {
   # for Box-Cox, lambda = 0 -> log transform
   if(lambda == 0) exp(x) else (lambda*x + 1)^(1/lambda)
```

```r
}

inv_box_cox(y, lambda)
```

```
[1] 10
attr(,"lambda")
[1] 3
```

```r
# Log transformation
train_log_y <- log(ames_train$Sale_Price)
test_log_y  <- log(ames_train$Sale_Price)

# Box Cox transformation
lambda  <- forecast::BoxCox.lambda(ames_train$Sale_Price)
train_bc_y <- forecast::BoxCox(ames_train$Sale_Price, lambda)
test_bc_y  <- forecast::BoxCox(ames_test$Sale_Price, lambda)

# Plot differences
levs <- c("Normal", "Log_Transform", "BoxCox_Transform")
data.frame(
  Normal = ames_train$Sale_Price,
  Log_Transform = train_log_y,
  BoxCox_Transform = train_bc_y
) %>%
  gather(Transform, Value) %>%
  mutate(Transform = factor(Transform, levels = levs)) %>%
  ggplot(aes(Value, fill = Transform)) +
    geom_histogram(show.legend = FALSE, bins = 40) +
    facet_wrap(~ Transform, scales = "free_x")
```

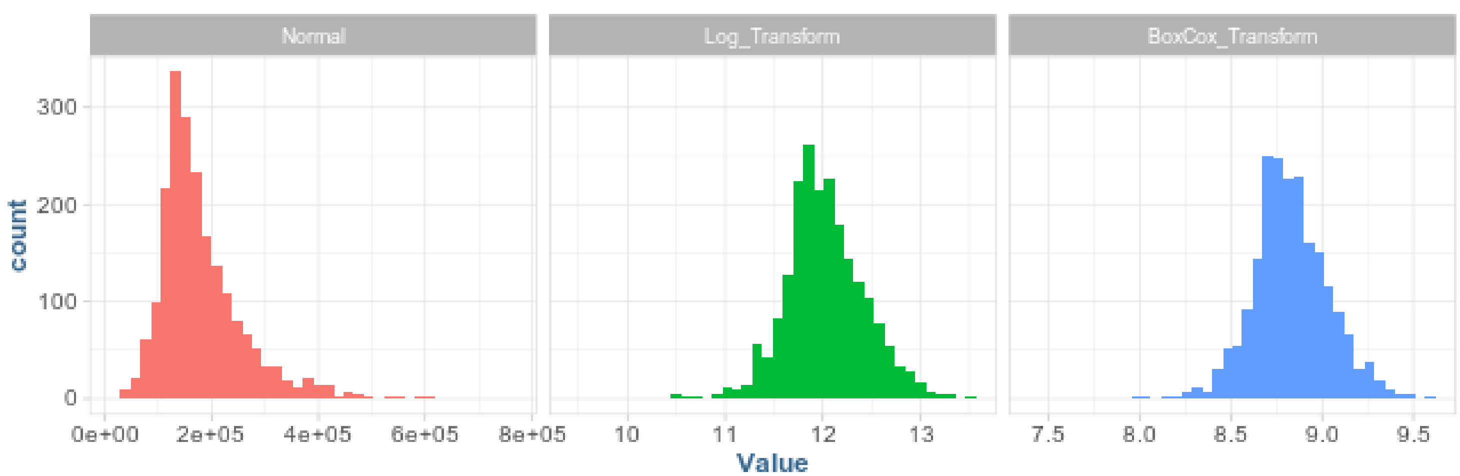


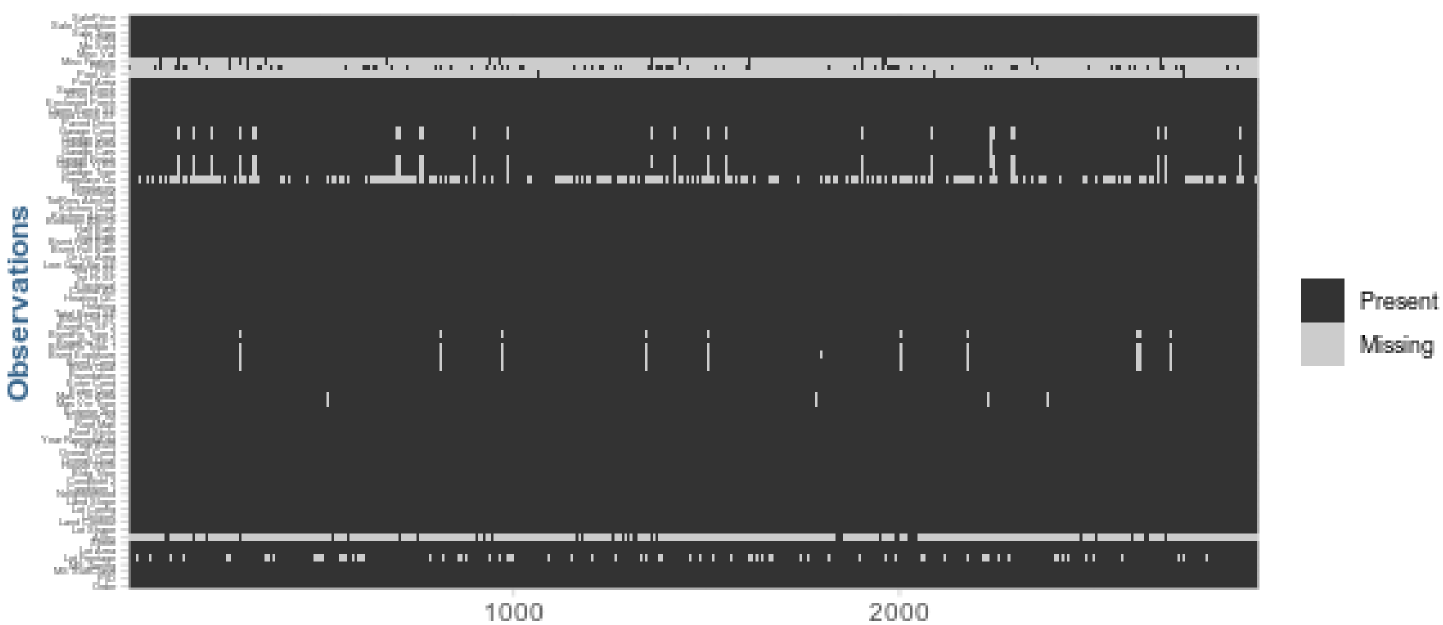
Figure 1: Response variable transformations.

Missing Values

```
sum(is.na(AmesHousing::ames_raw))
```

```
[1] 13997
```

```
AmesHousing::ames_raw %>%
   is.na() %>%
   reshape2::melt() %>%
   ggplot(aes(Var2, Var1, fill = value)) +
      geom_raster() +
      coord_flip() +
      scale_y_continuous(NULL, expand = c(0,0)) +
      scale_fill_grey(name = "",
                      labels = c("Present",
                                 "Missing")) +
   xlab("Observations") +
   theme(axis.text.y = element_text(size = 4))
```



Missing Garage?

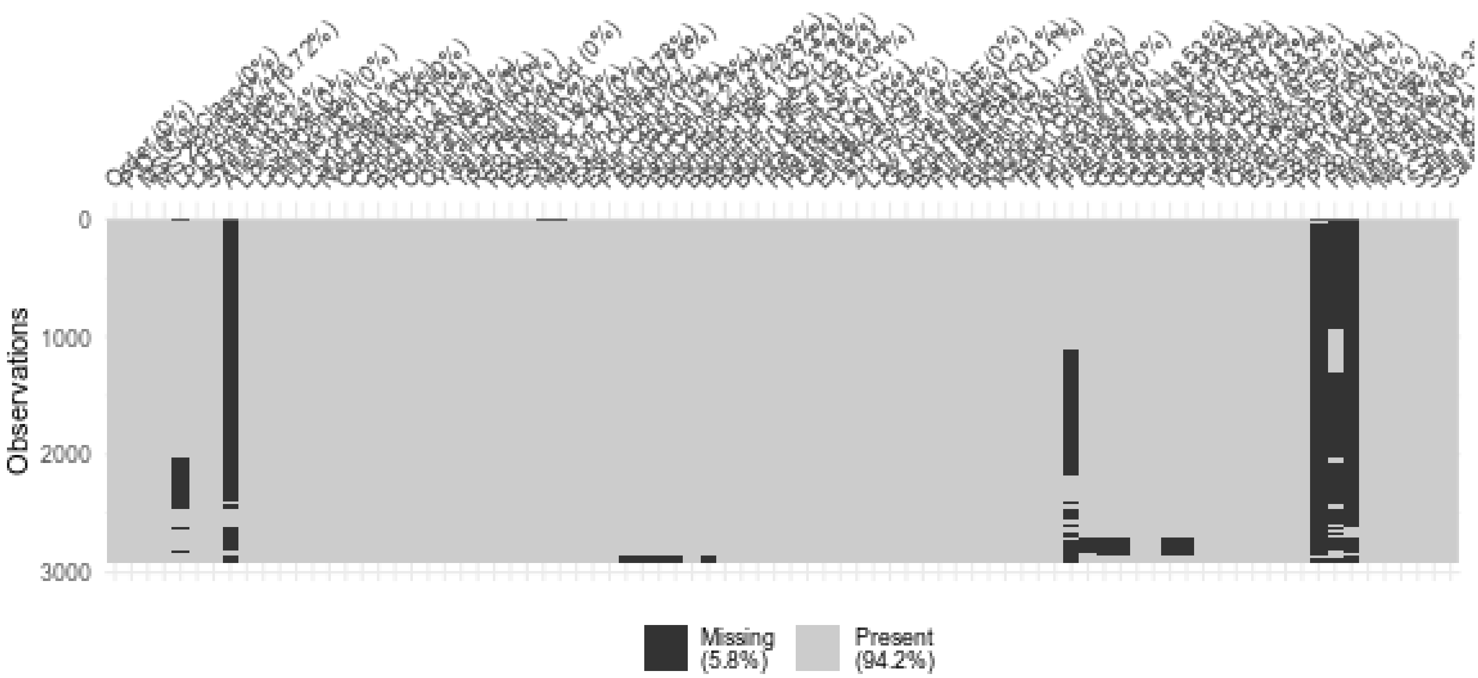
```
AmesHousing::ames_raw %>%
   filter(is.na(`Garage Type`)) %>%
   select(starts_with("Garage"))
```

```
# A tibble: 157 x 7
   `Garage Type` `Garage Yr Blt` `Garage Finish` `Garage Cars` `Garage Area`
   <chr>                   <int> <chr>                   <int>         <int>
```

```
 1 <NA>                     NA <NA>                      0            0
 2 <NA>                     NA <NA>                      0            0
 3 <NA>                     NA <NA>                      0            0
 4 <NA>                     NA <NA>                      0            0
 5 <NA>                     NA <NA>                      0            0
 6 <NA>                     NA <NA>                      0            0
 7 <NA>                     NA <NA>                      0            0
 8 <NA>                     NA <NA>                      0            0
 9 <NA>                     NA <NA>                      0            0
10 <NA>                     NA <NA>                      0            0
# … with 147 more rows, and 2 more variables: `Garage Qual` <chr>, `Garage
#   Cond` <chr>
```

Missing values w/cluster (*visdat*)

```
vis_miss(AmesHousing::ames_raw, cluster = T)
```



Missing Value Imputation

basic descriptive statistic

```
ames_recipe %>%
  step_medianimpute(Gr_Liv_Area)
```

```
Data Recipe
```

```
Inputs:

      role #variables
   outcome          1
 predictor         80

Operations:

Log transformation on all_outcomes
Median Imputation for Gr_Liv_Area
```

KNN approach (typical k = 5-10)

```
ames_recipe %>%
   step_knnimpute(all_predictors(), neighbors = 6)
```

```
Data Recipe

Inputs:

      role #variables
   outcome          1
 predictor         80

Operations:

Log transformation on all_outcomes
K-nearest neighbor imputation for all_predictors
```

```
impute_ames <- ames_train
set.seed(123)
index <- sample(seq_along(impute_ames$Gr_Liv_Area), 50)
actuals <- ames_train[index, ]
impute_ames$Gr_Liv_Area[index] <- NA

p1 <- ggplot() +
  geom_point(data = impute_ames, aes(Gr_Liv_Area, Sale_Price), alpha = .2) +
  geom_point(data = actuals, aes(Gr_Liv_Area, Sale_Price), color = "red") +
  scale_x_log10(limits = c(300, 5000)) +
  scale_y_log10(limits = c(10000, 500000)) +
  ggtitle("Actual values")

# Mean imputation
mean_juiced <- recipe(Sale_Price ~ ., data = impute_ames) %>%
  step_meanimpute(Gr_Liv_Area) %>%
  prep(training = impute_ames, retain = TRUE) %>%
```

```r
  juice()
mean_impute <- mean_juiced[index, ]

p2 <- ggplot() +
  geom_point(data = actuals, aes(Gr_Liv_Area, Sale_Price), color = "red") +
  geom_point(data = mean_impute, aes(Gr_Liv_Area, Sale_Price), color = "blue") +
  scale_x_log10(limits = c(300, 5000)) +
  scale_y_log10(limits = c(10000, 500000)) +
  ggtitle("Mean Imputation")

# KNN imputation
knn_juiced <- recipe(Sale_Price ~ ., data = impute_ames) %>%
  step_knnimpute(Gr_Liv_Area) %>%
  prep(training = impute_ames, retain = TRUE) %>%
  juice()
knn_impute <- knn_juiced[index, ]

p3 <- ggplot() +
  geom_point(data = actuals, aes(Gr_Liv_Area, Sale_Price), color = "red") +
  geom_point(data = knn_impute, aes(Gr_Liv_Area, Sale_Price), color = "blue") +
  scale_x_log10(limits = c(300, 5000)) +
  scale_y_log10(limits = c(10000, 500000)) +
  ggtitle("KNN Imputation")

# Bagged imputation
bagged_juiced <- recipe(Sale_Price ~ ., data = impute_ames) %>%
  step_bagimpute(Gr_Liv_Area) %>%
  prep(training = impute_ames, retain = TRUE) %>%
  juice()
bagged_impute <- bagged_juiced[index, ]

p4 <- ggplot() +
  geom_point(data = actuals, aes(Gr_Liv_Area, Sale_Price), color = "red") +
  geom_point(data = bagged_impute, aes(Gr_Liv_Area, Sale_Price), color = "blue") +
  scale_x_log10(limits = c(300, 5000)) +
  scale_y_log10(limits = c(10000, 500000)) +
  ggtitle("Bagged Trees Imputation")

gridExtra::grid.arrange(p1, p2, p3, p4, nrow = 2)
```

```
Warning: Removed 63 rows containing missing values (geom_point).
```

Increase in training time by model type:

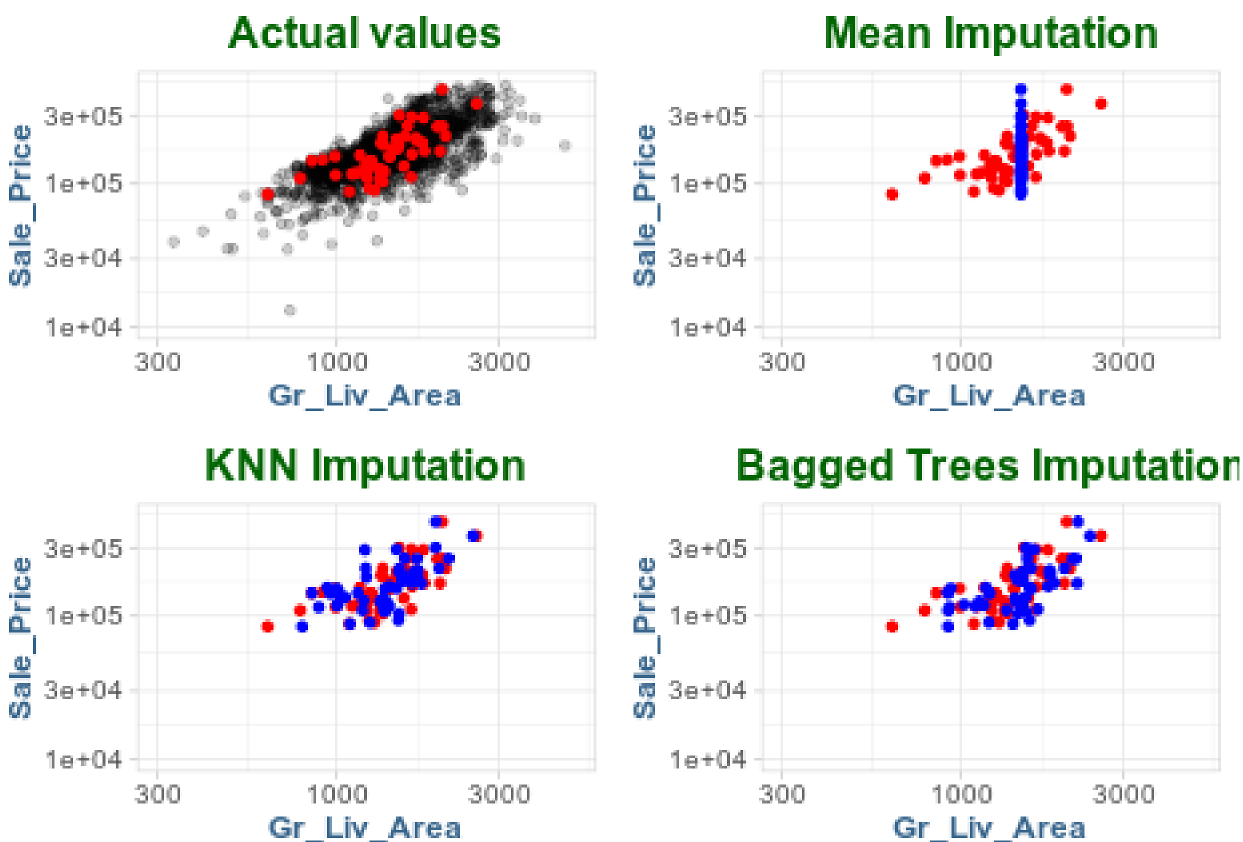
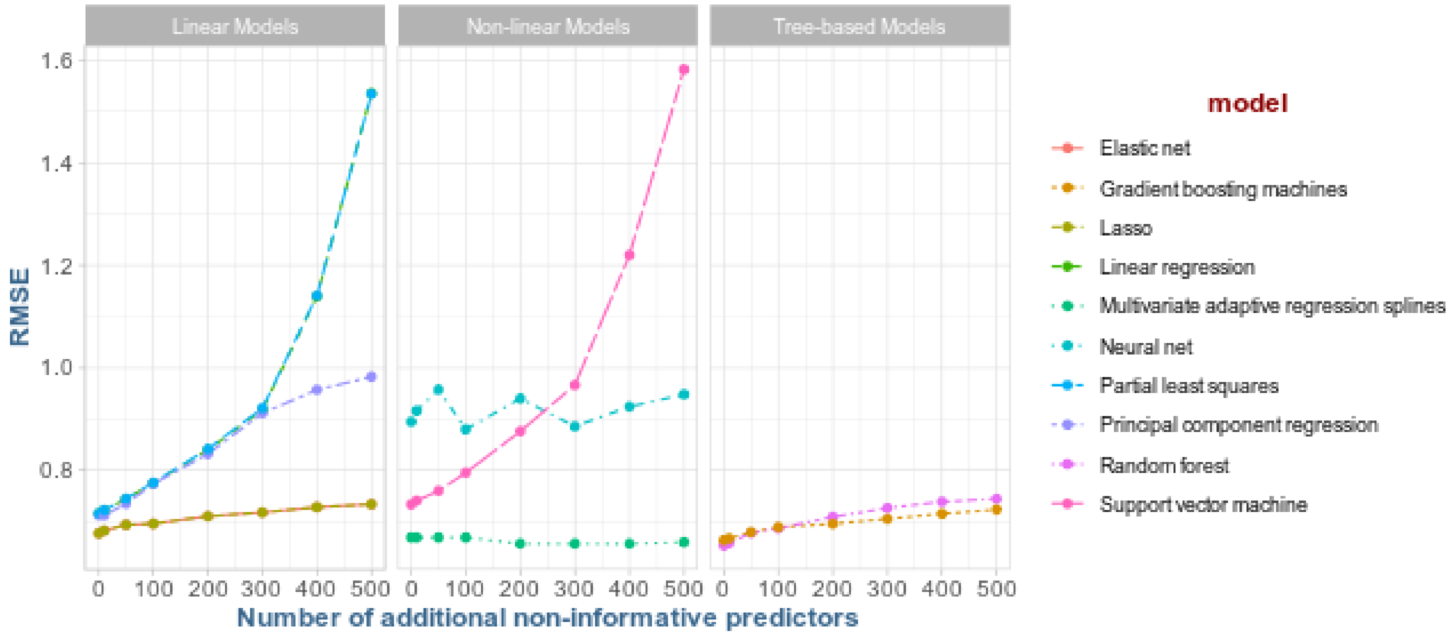


Figure 2: Comparison of three different imputation methods. The red points represent actual values which were removed and made missing and the blue points represent the imputed values. Estimated statistic imputation methods (i.e. mean, median) merely predict the same value for each observation and can reduce the signal between a feature and the response; whereas KNN and tree-based procedures tend to maintain the feature distribution and relationship.

```r
model_results <- read_csv(paste0(data.dir, "feature-selection-impacts-results.csv")) %>%
   mutate(type = case_when(
      model %in% c("lm", "pcr", "pls", "glmnet", "lasso") ~ "Linear Models",
      model %in% c("earth", "svmLinear", "nn") ~ "Non-linear Models",
      TRUE ~ "Tree-based Models"
   )) %>%
   mutate(model = case_when(
      model == "lm" ~ "Linear regression",
      model == "earth" ~ "Multivariate adaptive regression splines",
      model == "gbm" ~ "Gradient boosting machines",
      model == "glmnet" ~ "Elastic net",
      model == "lasso" ~ "Lasso",
      model == "nn" ~ "Neural net",
      model == "pcr" ~ "Principal component regression",
      model == "pls" ~ "Partial least squares",
      model == "ranger" ~ "Random forest",
      TRUE ~ "Support vector machine"
  ))
```

```
Parsed with column specification:
cols(
  model = col_character(),
  NIP = col_double(),
  RMSE = col_double(),
  time = col_double()
)
```

```r
ggplot(model_results, aes(NIP, RMSE, color = model, lty = model)) +
  geom_line() +
  geom_point() +
  facet_wrap(~ type, nrow = 1) +
  xlab("Number of additional non-informative predictors")
```




```
model_results %>%
  group_by(model) %>%
  mutate(
    time_impact = time / first(time),
    time_impact = time_impact - 1
  ) %>%
  ggplot(aes(NIP, time_impact, color = model, lty = model)) +
    geom_line() +
    geom_point() +
    facet_wrap(~ type, nrow = 1) +
    scale_y_continuous("Percent increase in training time",
                       labels = scales::percent) +
    xlab("Number of additional non-informative predictors")
```

Rules of thumb for zero variance features:

- The fraction of unique values over the sample size is low (say < 10%)
- The ratio of the frequency of the most prevalent value to the frequency of the second most prevalent value is large (say > 20%)

If both of these criteria are met, then it is often advantageous to remove them from the model.

```
caret::nearZeroVar(ames_train, saveMetrics = T) %>%
   rownames_to_column() %>%
   filter(nzv)
```

```
              rowname  freqRatio percentUnique zeroVar  nzv
1              Street  292.28571    0.09741841   FALSE TRUE
2               Alley   20.52688    0.14612762   FALSE TRUE
```

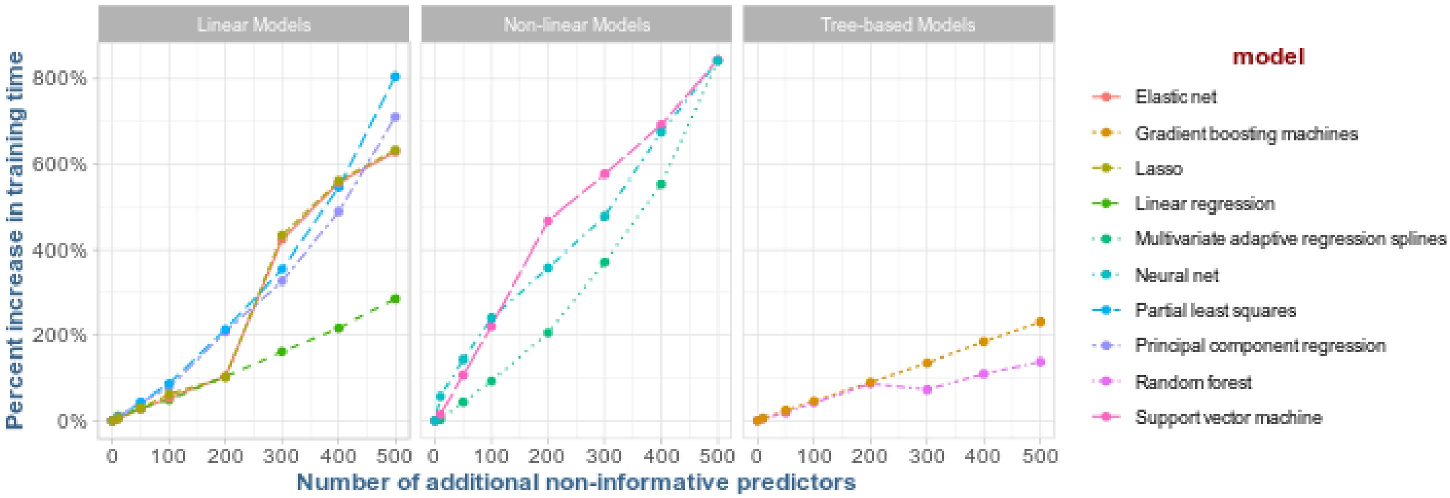


Figure 3: Impact in model training time as non-informative predictors are added.

```
3         Land_Contour   22.28916    0.19483682   FALSE TRUE
4            Utilities 1025.00000    0.14612762   FALSE TRUE
5           Land_Slope   22.76744    0.14612762   FALSE TRUE
6          Condition_2  203.10000    0.34096444   FALSE TRUE
7            Roof_Matl  126.50000    0.24354603   FALSE TRUE
8            Bsmt_Cond   19.93478    0.29225524   FALSE TRUE
9       BsmtFin_Type_2   21.50617    0.34096444   FALSE TRUE
10             Heating  101.05000    0.24354603   FALSE TRUE
11     Low_Qual_Fin_SF 1013.00000    1.31514856   FALSE TRUE
12       Kitchen_AbvGr   23.68675    0.19483682   FALSE TRUE
13          Functional   38.18000    0.34096444   FALSE TRUE
14      Enclosed_Porch  100.94118    7.40379932   FALSE TRUE
15 Three_season_porch  674.66667    1.16902094   FALSE TRUE
16        Screen_Porch  234.87500    4.52995616   FALSE TRUE
17           Pool_Area 2045.00000    0.43838285   FALSE TRUE
18             Pool_QC  681.66667    0.24354603   FALSE TRUE
19        Misc_Feature   30.49231    0.19483682   FALSE TRUE
20            Misc_Val  165.33333    1.41256698   FALSE TRUE
```

Numeric Feature Engineering

Skewness can have a drastic impact on the performance of GLMs & regularized models.

Non-parametric models are rarely affected by skewed features; however, normalizing features will not have a negative effect on these models' performance. For example, normalizing features will only shift the optimal split points in tree-based algoirthms. Consequently, when in doubt, normalize.

Skewness

```
Data Recipe

Inputs:

      role #variables
   outcome          1
 predictor         80

Operations:

Yeo-Johnson transformation on all_numeric
```

Standardization

```r
ames_recipe %>%
   step_center(all_numeric(), -all_outcomes()) %>%
   step_scale(all_numeric(), -all_outcomes())
```

```
Data Recipe

Inputs:

      role #variables
   outcome          1
 predictor         80

Operations:

Log transformation on all_outcomes
Centering for all_numeric, -, all_outcomes()
Scaling for all_numeric, -, all_outcomes()
```

```r
set.seed(123)
x1 <- tibble(
   variable = "x1",
   `Real Value` = runif(25, min = -30, max = 5),
   `Standardized Value` = scale(`Real Value`) %>% as.numeric()

)

set.seed(456)
x2 <- tibble(
```

```r
  variable = "x2",
  `Real value` = rlnorm(25, log(25)),
  `Standardized value` = scale(`Real value`) %>% as.numeric()
)

set.seed(789)
x3 <- tibble(
  variable = "x3",
  `Real value` = rnorm(25, 150, 15),
  `Standardized value` = scale(`Real value`) %>% as.numeric()
)

x1 %>%
  bind_rows(x2) %>%
  bind_rows(x3) %>%
  gather(key, value, -variable) %>%
  mutate(variable = factor(variable, levels = c("x3", "x2", "x1"))) %>%
  ggplot(aes(value, variable)) +
    geom_point(alpha = .6) +
    facet_wrap(~ key, scales = "free_x") +
    ylab("Feature") +
    xlab("Value")
```

```
Warning: Removed 150 rows containing missing values (geom_point).
```

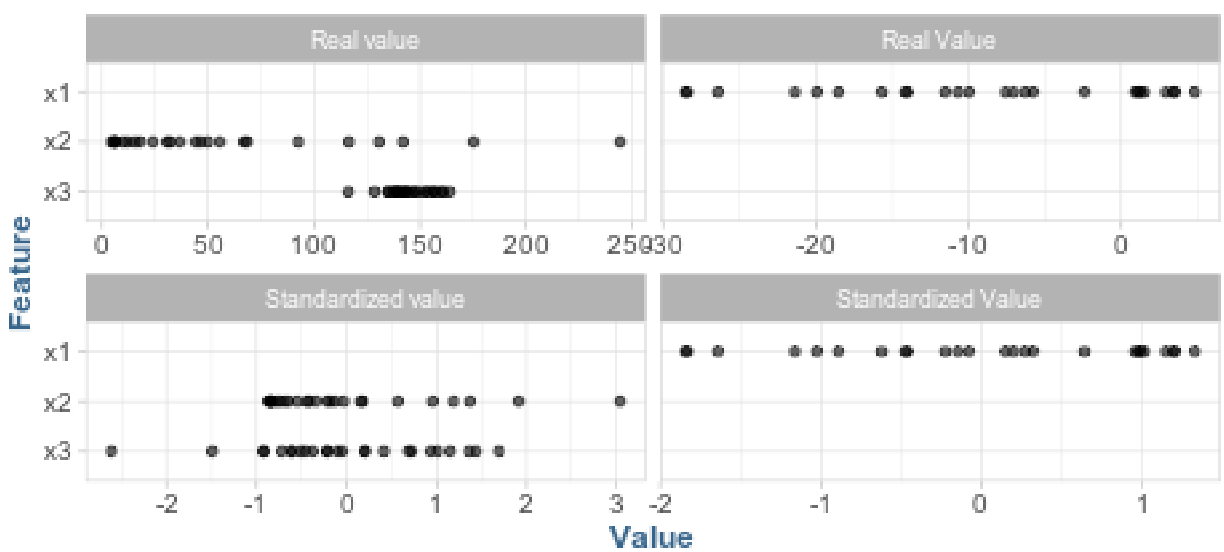


Figure 4: Standardizing features allows all features to be compared on a common value scale regardless of their real value differences.

Categorical Feature Engineering

Lumping

When a feature contains levels that have few observations.

For example:

```
count(ames_train, Neighborhood) %>% arrange(n)
```

```
# A tibble: 27 x 2
   Neighborhood                                  n
   <fct>                                     <int>
 1 Green_Hills                                   2
 2 Greens                                        7
 3 Blueste                                       8
 4 Northpark_Villa                              17
 5 Briardale                                    18
 6 Veenker                                      20
 7 Bloomington_Heights                          21
 8 South_and_West_of_Iowa_State_University      27
 9 Meadow_Village                               29
10 Clear_Creek                                  31
# … with 17 more rows
```

```
count(ames_train, Screen_Porch) %>% arrange(n)
```

```
# A tibble: 93 x 2
   Screen_Porch     n
          <int> <int>
 1           40     1
 2           63     1
 3           80     1
 4           92     1
 5           94     1
 6           99     1
 7          104     1
 8          109     1
 9          110     1
10          111     1
# … with 83 more rows
```

We can benefit from lumping these together into an “other” category when they contain less than 10% of the training sample.

Note: This can have an adverse effect on performance

```r
lumping <- recipe(Sale_Price ~., data = ames_train) %>%
   step_other(Neighborhood, threshold = 0.01,
              other = "other") %>%
   step_other(Screen_Porch, threshold = 0.1,
              other = ">0")

apply_2_training <- prep(lumping, training = ames_train) %>%
  bake(ames_train)

# New distribution of Neighborhood
count(apply_2_training, Neighborhood) %>% arrange(n)
```

```
# A tibble: 22 x 2
   Neighborhood                                   n
   <fct>                                      <int>
 1 Bloomington_Heights                           21
 2 South_and_West_of_Iowa_State_University       27
 3 Meadow_Village                                29
 4 Clear_Creek                                   31
 5 Stone_Brook                                   34
 6 Northridge                                    48
 7 Timberland                                    55
 8 Iowa_DOT_and_Rail_Road                        62
 9 Crawford                                      72
10 other                                         72
# … with 12 more rows
```

```r
# New distribution of Screen_Porch
count(apply_2_training, Screen_Porch) %>% arrange(n)
```

```
# A tibble: 2 x 2
  Screen_Porch     n
  <fct>        <int>
1 >0             174
2 0             1879
```

```r
dat <- data.table(id = 1:9, x = rep(c("a", "b" , "c"), 3))
dat
```

```
   id x
1:  1 a
2:  2 b
3:  3 c
4:  4 a
5:  5 b
6:  6 c
```

```
7:  7 a
8:  8 b
9:  9 c
```

```r
# full-rank
dat[, .(id,
        `X = a` = as.numeric(x == "a"),
        `X = b` = as.numeric(x == "b"),
        `X = c` = as.numeric(x == "c")) ]
```

```
   id X = a X = b X = c
1:  1     1     0     0
2:  2     0     1     0
3:  3     0     0     1
4:  4     1     0     0
5:  5     0     1     0
6:  6     0     0     1
7:  7     1     0     0
8:  8     0     1     0
9:  9     0     0     1
```

```r
# one-hot (leave one out)
dat[, .(id,
        `X = a` = as.numeric(x == "a"),
        `X = b` = as.numeric(x == "b")) ]
```

```
   id X = a X = b
1:  1     1     0
2:  2     0     1
3:  3     0     0
4:  4     1     0
5:  5     0     1
6:  6     0     0
7:  7     1     0
8:  8     0     1
9:  9     0     0
```