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)
```

```
}

inv_box_cox(y, lambda)
```

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

```
# 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")
```

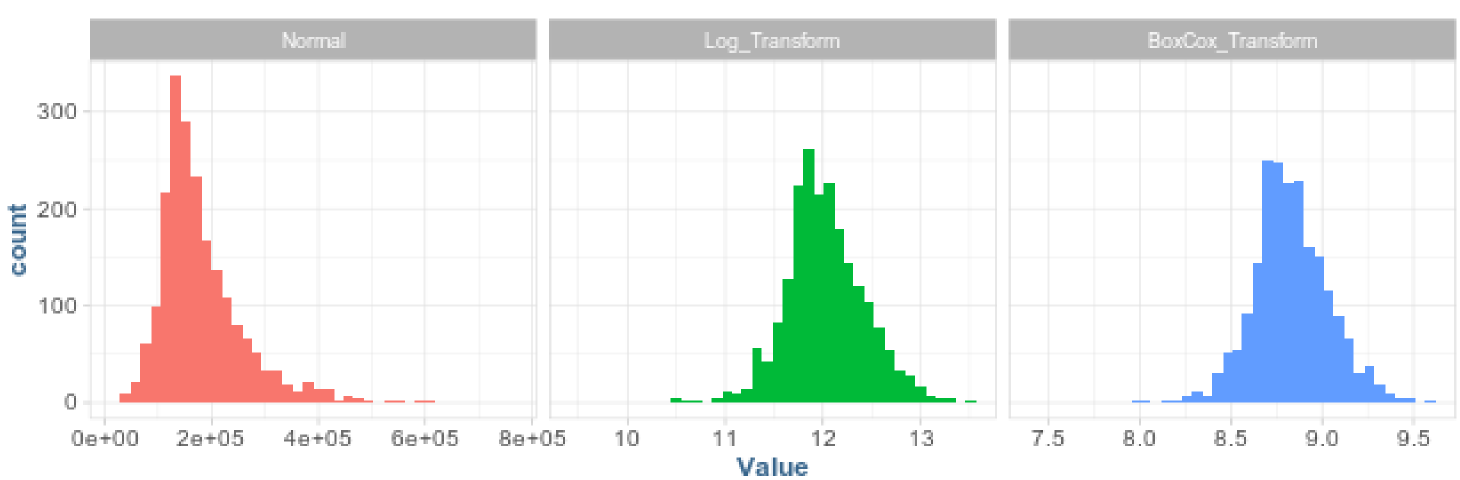


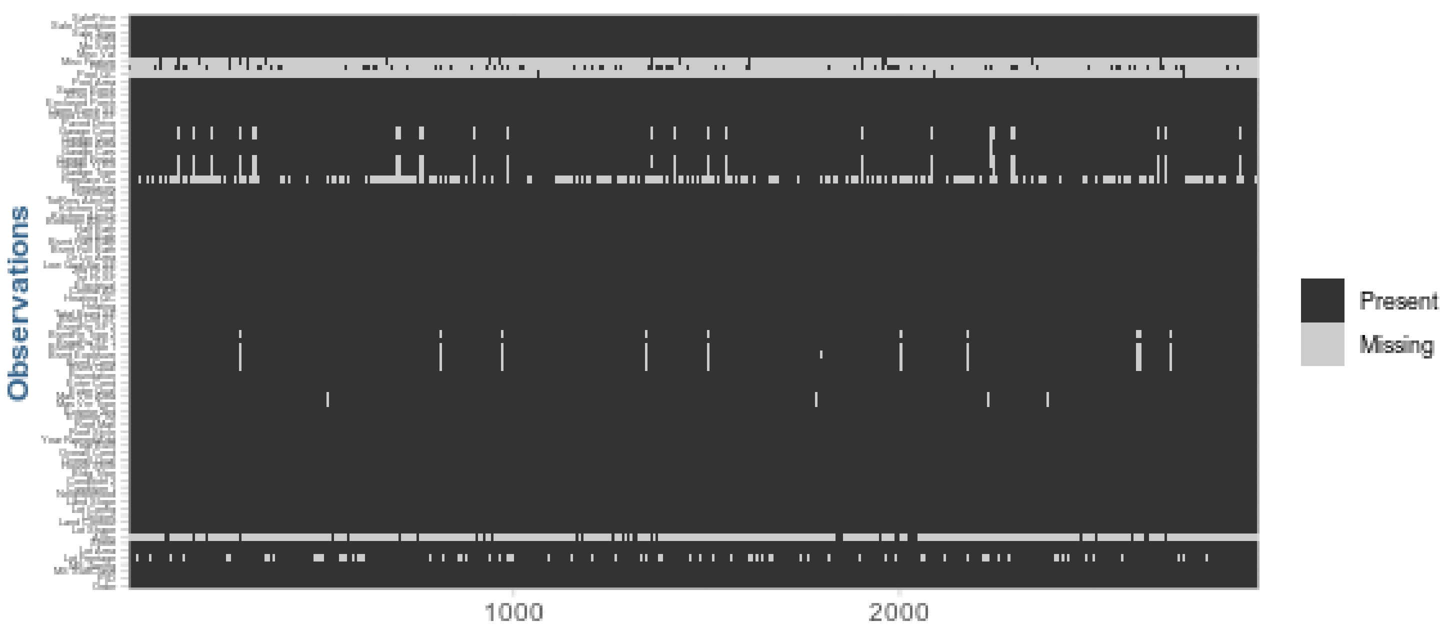
Figure 1: Response variable transformations.

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?

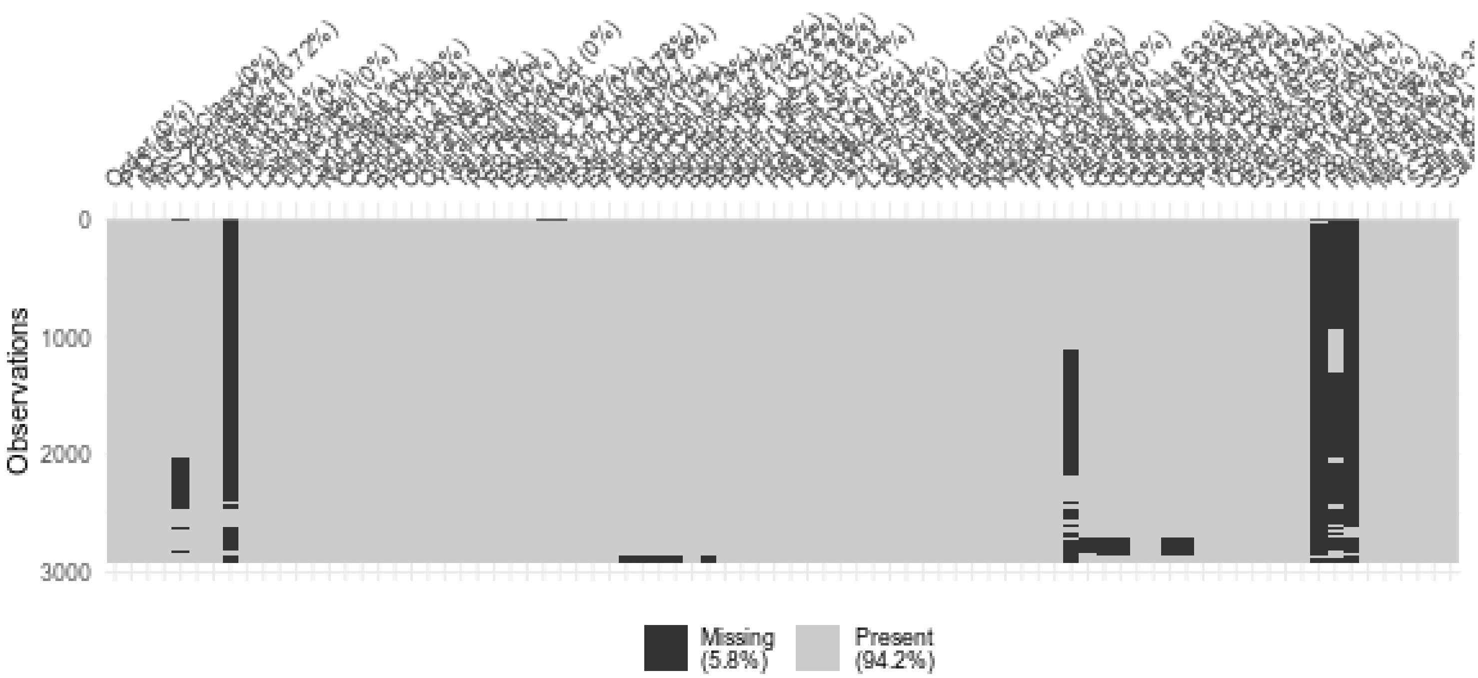
```
AmesHousing::ames_raw %>%
   filter(is.na(`Garage Type`)) %>%
   select(starts_with("Garage"))
```

```
# 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*)

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



Missing Value Imputation

basic descriptive statistic

```r
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)

```r
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
```

```r
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) %>%
```

```
  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:

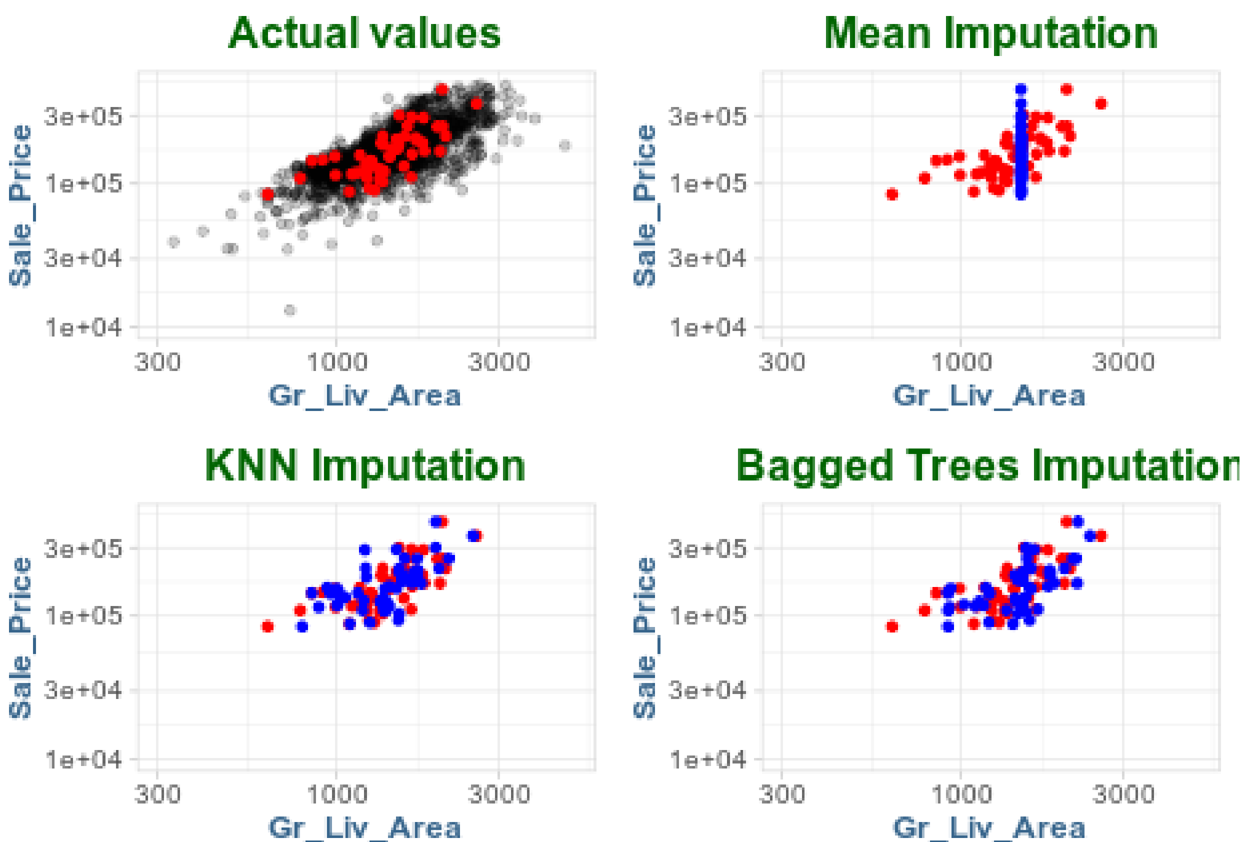
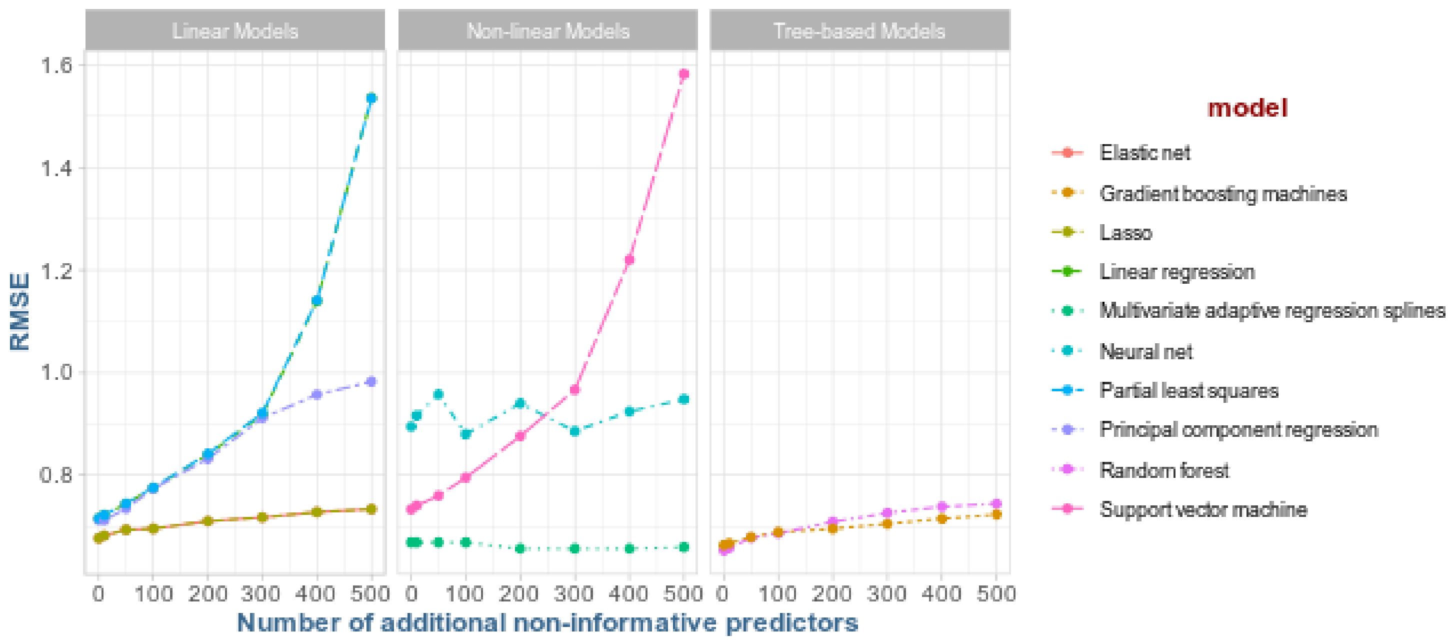


Figure 2: Comparison of three different imputation methods. The red points represent actual values which were removed and made missing and the blue points represent the imputed values. Estimated statistic imputation methods (i.e. mean, median) merely predict the same value for each observation and can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

```
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()
)
```

```
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
```

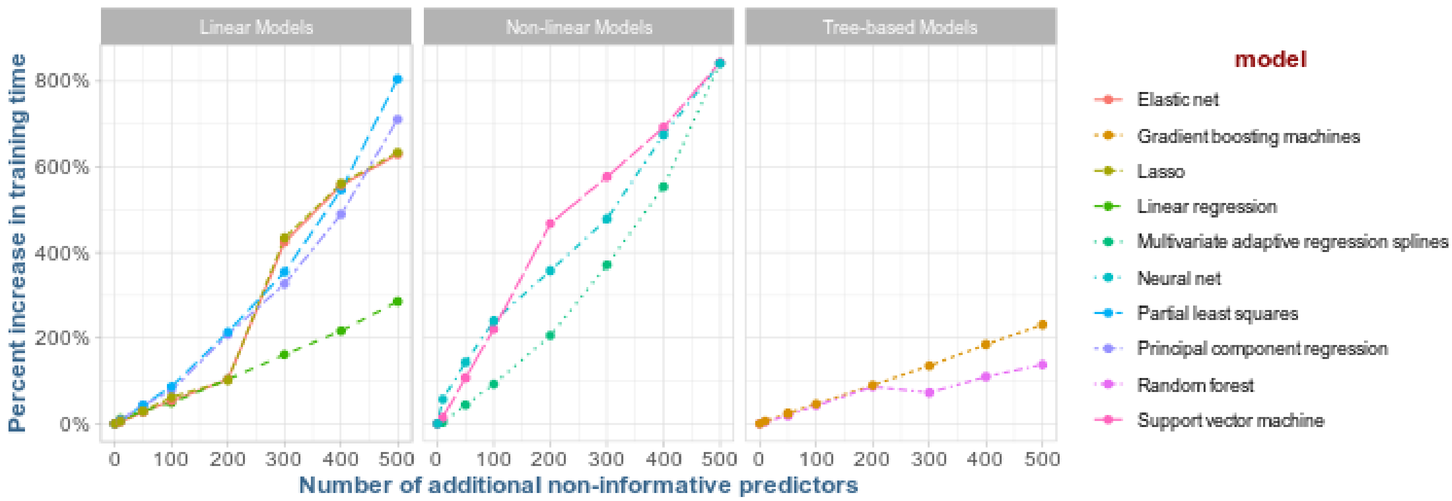


Figure 3: Impact in model training time as non-informative predictors are added.

```
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).
```

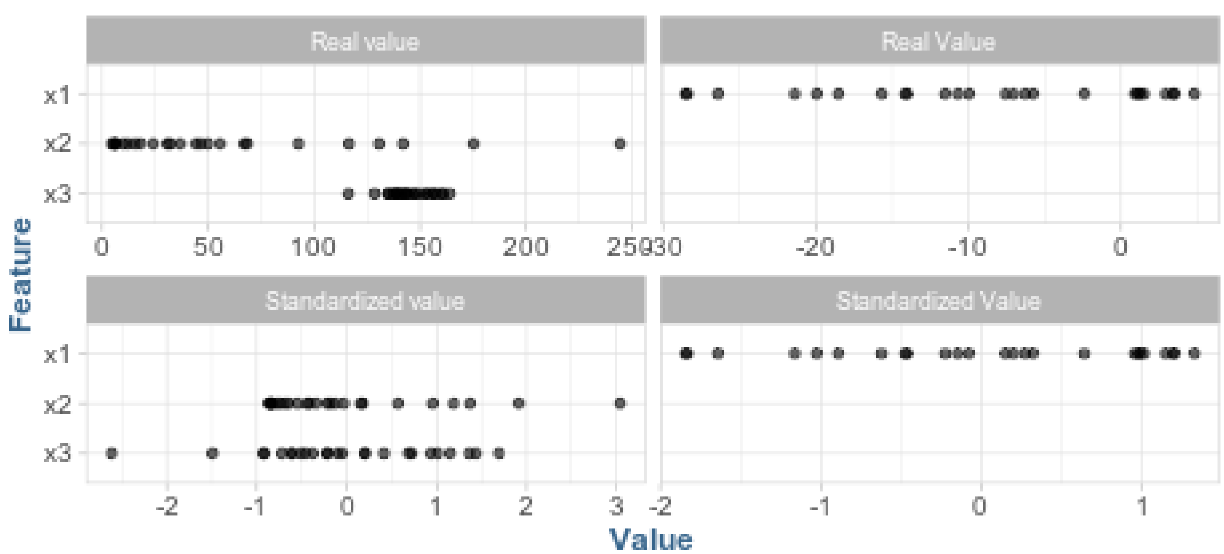


Figure 4: Standardizing features allows all features to be compared on a common value scale regardless of their real value differences.

Categorical Feature Engineering

Lumping

When a feature contains levels that have few observations.

For example:

```r
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
```

```r
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
```

```r
# Lump levels for two features
recipe(Sale_Price ~., data = ames_train) %>%
   step_dummy(all_nominal(), one_hot = T)
```

```
Data Recipe

Inputs:

      role #variables
   outcome          1
```

```
 predictor         80

Operations:

Dummy variables from all_nominal
```

Label Encoding

Label encoding is a pure numeric conversion of the levels of a categorical variable.

For example, the *MS_SubClass* variable has 16 levels, which we can reencode numerically.

Important: The features will be treated as ordered (ordnal encoding), so if the feature is not natually ordered, this will have a poor impact on the model.

```r
count(ames_train, MS_SubClass)
```

```
# A tibble: 16 x 2
   MS_SubClass                                   n
   <fct>                                     <int>
 1 One_Story_1946_and_Newer_All_Styles         753
 2 One_Story_1945_and_Older                     91
 3 One_Story_with_Finished_Attic_All_Ages        5
 4 One_and_Half_Story_Unfinished_All_Ages       11
 5 One_and_Half_Story_Finished_All_Ages         211
 6 Two_Story_1946_and_Newer                    395
 7 Two_Story_1945_and_Older                     98
 8 Two_and_Half_Story_All_Ages                  17
 9 Split_or_Multilevel                          75
10 Split_Foyer                                  32
11 Duplex_All_Styles_and_Ages                   66
12 One_Story_PUD_1946_and_Newer                145
13 One_and_Half_Story_PUD_All_Ages               1
14 Two_Story_PUD_1946_and_Newer                 96
15 PUD_Multilevel_Split_Level_Foyer             14
16 Two_Family_conversion_All_Styles_and_Ages    43
```

```r
# Label encoded
recipe(Sale_Price ~ ., data = ames_train) %>%
   step_integer(MS_SubClass) %>%
   prep(ames_train) %>%
   bake(ames_train) %>%
   count(MS_SubClass)
```

```
# A tibble: 16 x 2
   MS_SubClass     n
```

```
         <dbl> <int>
 1           1   753
 2           2    91
 3           3     5
 4           4    11
 5           5   211
 6           6   395
 7           7    98
 8           8    17
 9           9    75
10          10    32
11          11    66
12          12   145
13          13     1
14          14    96
15          15    14
16          16    43
```

Examples of ordnal features:

```
ames_train %>% select(contains("Qual"))
```

```
# A tibble: 2,053 x 6
   Overall_Qual  Exter_Qual Bsmt_Qual Low_Qual_Fin_SF Kitchen_Qual Garage_Qual
   <fct>         <fct>      <fct>               <int> <fct>        <fct>
 1 Above_Average Typical    Typical                 0 Typical      Typical
 2 Average       Typical    Typical                 0 Typical      Typical
 3 Above_Average Typical    Typical                 0 Good         Typical
 4 Above_Average Typical    Typical                 0 Good         Typical
 5 Very_Good     Good       Good                    0 Good         Typical
 6 Very_Good     Good       Good                    0 Good         Typical
 7 Good          Typical    Typical                 0 Good         Typical
 8 Above_Average Typical    Good                    0 Typical      Typical
 9 Above_Average Typical    Good                    0 Typical      Typical
10 Good          Typical    Good                    0 Good         Typical
# … with 2,043 more rows
```

```
count(ames_train, Overall_Qual)
```

```
# A tibble: 10 x 2
   Overall_Qual      n
   <fct>         <int>
 1 Very_Poor         4
 2 Poor              8
 3 Fair             26
 4 Below_Average   170
 5 Average         564
```

```
 6 Above_Average    511
 7 Good             439
 8 Very_Good        231
 9 Excellent         77
10 Very_Excellent    23
```

```r
# Label encoded
recipe(Sale_Price ~., data = ames_train) %>%
   step_integer(Overall_Qual) %>%
   prep(ames_train) %>%
   bake(ames_train) %>%
   count(Overall_Qual)
```

```
# A tibble: 10 x 2
   Overall_Qual     n
          <dbl> <int>
 1            1     4
 2            2     8
 3            3    26
 4            4   170
 5            5   564
 6            6   511
 7            7   439
 8            8   231
 9            9    77
10           10    23
```

Alternatives

Target encoding:

```r
ames_train %>%
  group_by(Neighborhood) %>%
  summarize(`Avg Sale_Price` = mean(Sale_Price, na.rm = TRUE)) %>%
  head(10) %>%
  kable(caption = "Example of target encoding the Neighborhood feature of the Ames housing data
  kable_styling(bootstrap_options = "striped", full_width = TRUE)
```

```r
ames_train %>%
  count(Neighborhood) %>%
  mutate(Proportion = n / sum(n)) %>%
  select(-n) %>%
  head(10) %>%
  kable(caption = 'Example of categorical proportion encoding the Neighborhood feature of the A
  kable_styling(bootstrap_options = "striped", full_width = TRUE)
```

Table 1: Example of target encoding the Neighborhood feature of the Ames housing data set.

Neighborhood	Avg Sale_Price
North_Ames	144562.7
College_Creek	199831.7
Old_Town	122736.7
Edwards	130652.2
Somerset	227379.6
Northridge_Heights	323289.5
Gilbert	192162.9
Sawyer	136460.6
Northwest_Ames	187328.2
Sawyer_West	188644.6

Table 2: Example of categorical proportion encoding the Neighborhood feature of the Ames housing data set.

Neighborhood	Proportion
North_Ames	0.1451534
College_Creek	0.0910862
Old_Town	0.0832927
Edwards	0.0711154
Somerset	0.0623478
Northridge_Heights	0.0560156
Gilbert	0.0565027
Sawyer	0.0496834
Northwest_Ames	0.0467608
Sawyer_West	0.0414028

Dimension Reduction

Example PCA using *resample* package.

```
Data Recipe

Inputs:

      role #variables
   outcome          1
 predictor         80

Operations:

Centering for all_numeric
Scaling for all_numeric
No PCA components were extracted.
```

Full Recipe

Full blueprint recipe applied to training and test data.

```r
blueprint <- recipe(Sale_Price ~ ., data = ames_train) %>%
   step_nzv(all_nominal()) %>%
   step_integer(matches("Qual|Cond|QC|Qu")) %>%
   step_center(all_numeric(), -all_outcomes()) %>%
   step_scale(all_numeric(), -all_outcomes()) %>%
   step_pca(all_numeric(), -all_outcomes())

blueprint
```

```
Data Recipe

Inputs:

      role #variables
   outcome          1
 predictor         80

Operations:

Sparse, unbalanced variable filter on all_nominal
Integer encoding for matches, Qual|Cond|QC|Qu
Centering for all_numeric, -, all_outcomes()
```

```
Scaling for all_numeric, -, all_outcomes()
No PCA components were extracted.
```

```r
prepare <- prep(blueprint, training = ames_train)
prepare
```

```
Data Recipe

Inputs:

      role #variables
   outcome          1
 predictor         80

Training data contained 2053 data points and no missing data.

Operations:

Sparse, unbalanced variable filter removed Street, Alley, Land_Contour, ... [trained]
Integer encoding for Condition_1, Overall_Qual, Overall_Cond, ... [trained]
Centering for Lot_Frontage, Lot_Area, ... [trained]
Scaling for Lot_Frontage, Lot_Area, ... [trained]
PCA extraction with Lot_Frontage, Lot_Area, ... [trained]
```

```r
baked_train <- bake(prepare, new_data = ames_train)
baked_test <- bake(prepare, new_data = ames_test)

baked_train
```

```
# A tibble: 2,053 x 27
   MS_SubClass MS_Zoning Lot_Shape Lot_Config Neighborhood Bldg_Type House_Style
   <fct>       <fct>     <fct>     <fct>      <fct>        <fct>     <fct>
 1 One_Story_… Resident… Slightly… Corner     North_Ames   OneFam    One_Story
 2 One_Story_… Resident… Regular   Inside     North_Ames   OneFam    One_Story
 3 One_Story_… Resident… Slightly… Corner     North_Ames   OneFam    One_Story
 4 Two_Story_… Resident… Slightly… Inside     Gilbert      OneFam    Two_Story
 5 One_Story_… Resident… Regular   Inside     Stone_Brook  TwnhsE    One_Story
 6 One_Story_… Resident… Slightly… Inside     Stone_Brook  TwnhsE    One_Story
 7 Two_Story_… Resident… Regular   Inside     Gilbert      OneFam    Two_Story
 8 Two_Story_… Resident… Slightly… Corner     Gilbert      OneFam    Two_Story
 9 Two_Story_… Resident… Slightly… Inside     Gilbert      OneFam    Two_Story
10 One_Story_… Resident… Regular   Inside     Gilbert      OneFam    One_Story
# … with 2,043 more rows, and 20 more variables: Roof_Style <fct>,
#   Exterior_1st <fct>, Exterior_2nd <fct>, Mas_Vnr_Type <fct>,
#   Foundation <fct>, Bsmt_Exposure <fct>, BsmtFin_Type_1 <fct>,
#   Central_Air <fct>, Electrical <fct>, Garage_Type <fct>,
```

```
#   Garage_Finish <fct>, Paved_Drive <fct>, Fence <fct>, Sale_Type <fct>,
#   Sale_Price <int>, PC1 <dbl>, PC2 <dbl>, PC3 <dbl>, PC4 <dbl>, PC5 <dbl>
```

Full recipe with cross-validation & grid search using carat.

```
k-Nearest Neighbors

2053 samples
  80 predictor

Recipe steps: nzv, integer, center, scale, dummy
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 1848, 1846, 1848, 1847, 1848, 1849, ...
Resampling results across tuning parameters:

  k   RMSE      Rsquared   MAE
   2  36116.20  0.8024564  22617.83
   3  35077.80  0.8149018  21698.15
   4  34670.83  0.8205474  21274.44
   5  34240.24  0.8280936  21067.36
   6  33813.06  0.8346543  20889.18
   7  33517.75  0.8404839  20777.87
   8  33324.82  0.8440427  20647.95
   9  33148.42  0.8468769  20598.96
  10  33059.04  0.8488142  20610.92
  11  33048.49  0.8500846  20657.02
  12  32951.81  0.8521235  20680.45
  13  33017.22  0.8529401  20736.74
  14  32959.37  0.8548174  20753.98
  15  32976.91  0.8558820  20797.87
  16  33013.07  0.8565768  20849.17
  17  33028.05  0.8571048  20907.07
  18  33119.22  0.8568609  21015.28
  19  33124.87  0.8572290  21065.00
  20  33199.35  0.8571015  21147.53
  21  33274.67  0.8566490  21204.72
  22  33291.52  0.8568531  21250.21
  23  33323.99  0.8571500  21295.17
  24  33399.65  0.8570289  21366.10
  25  33470.78  0.8569297  21439.81

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was k = 12.
```

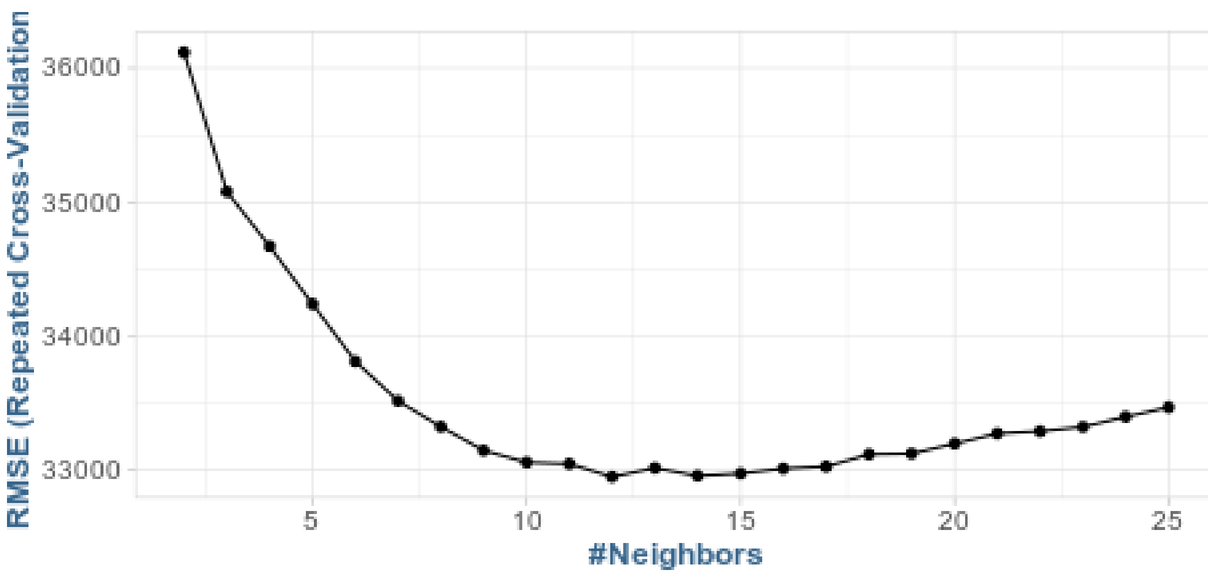


Figure 5: Results from the same grid search performed in Section 2.7 but with feature engineering performed within each resample.