Hoang

MATH158 Final Project

December 7th, 2023


Zing.vn

Predicting Used 🚙 Value

What's the maximum listing price my used car could be sold?

- 3,000,000 observations from (kaggle.com/us-used-cars-dataset) – 10GB.
- 66 features, half of them have no descriptions in metadata.
- Crawled on cargurus.com in September 2020 -> submissions are independent.
- Diverse data types: text, number, Nan
- Metric: custom Mean Absolute Error, where:
  - under-prediction will * 1.2
  - over-prediction will * 0.8

=> Motivation Questions:

1. What features should be used for prediction?
2. How well will my prediction be without knowing any feature descriptions?

Explore Data: getting started

- What's my response column name? "price".

=> X: DataFrame without column 'price'.

y: Column 'price' and nothing else.

- Do I want an intercept β_0?

<= Are there rows where all features' values being zero(s) ?

```
> sum(apply(X, 1, function(row) all(row == 0)))
[1] 0
```

=> YES

=> X: DataFrame without column 'price' and <u>has a column full of 1's</u>.

- Do I want duplicating rows / columns ?

<= Will duplicating rows / columns affect regression coefficients? No.

<= Will duplicating rows / columns invalidate regression assumptions? Yes (multi-collinearity).

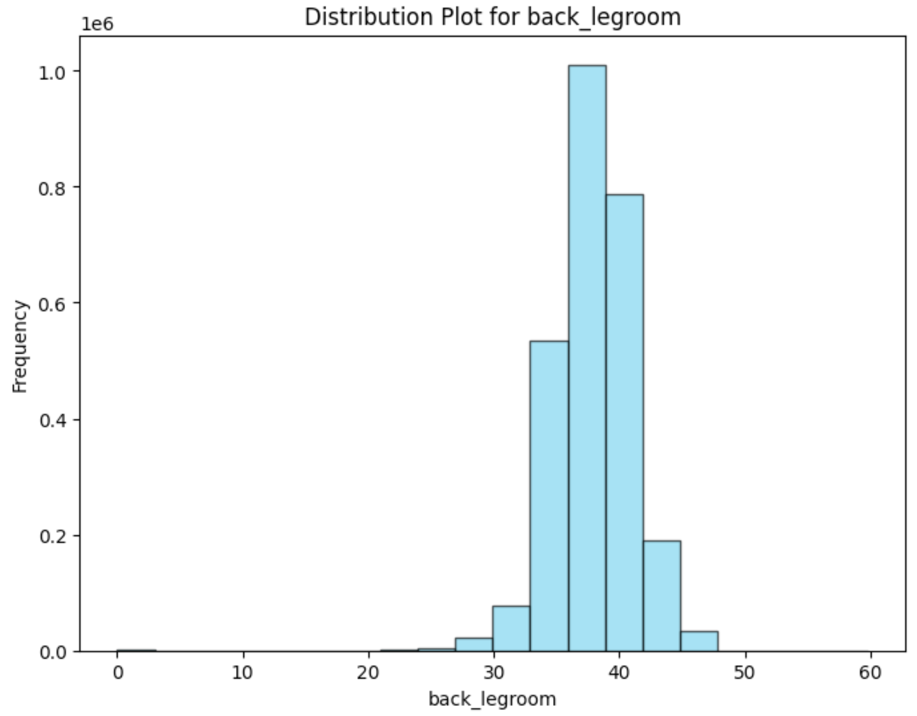
=> NO, remove them.

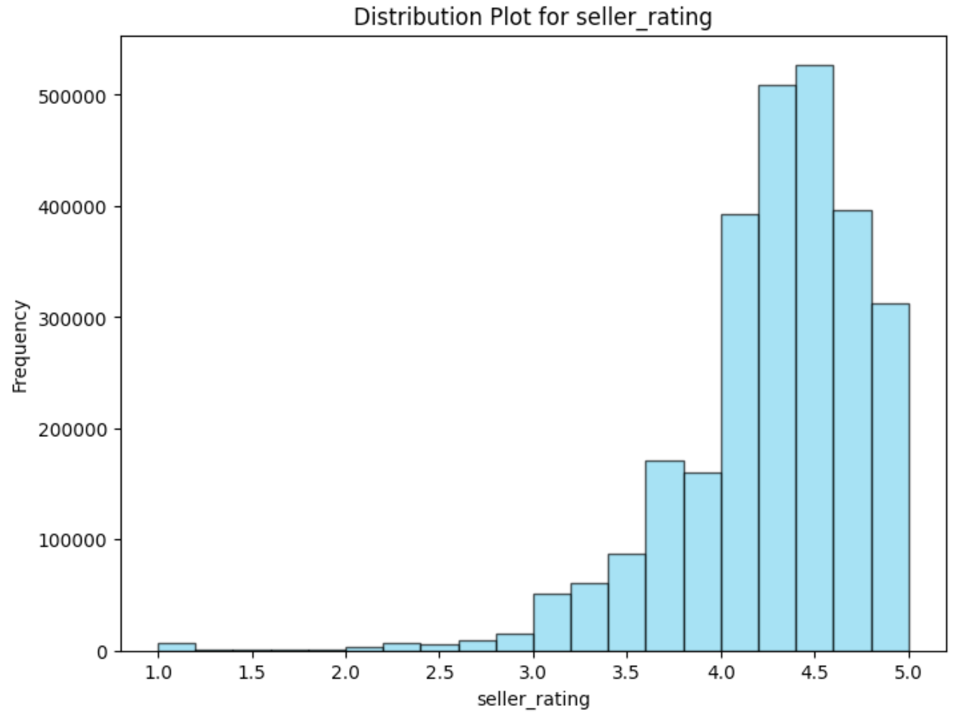
- How do I split the data?

=> Just split as usual (80% X_train, y_train; 20% X_test, y_test)

(because I forgot to split train-valid-test)

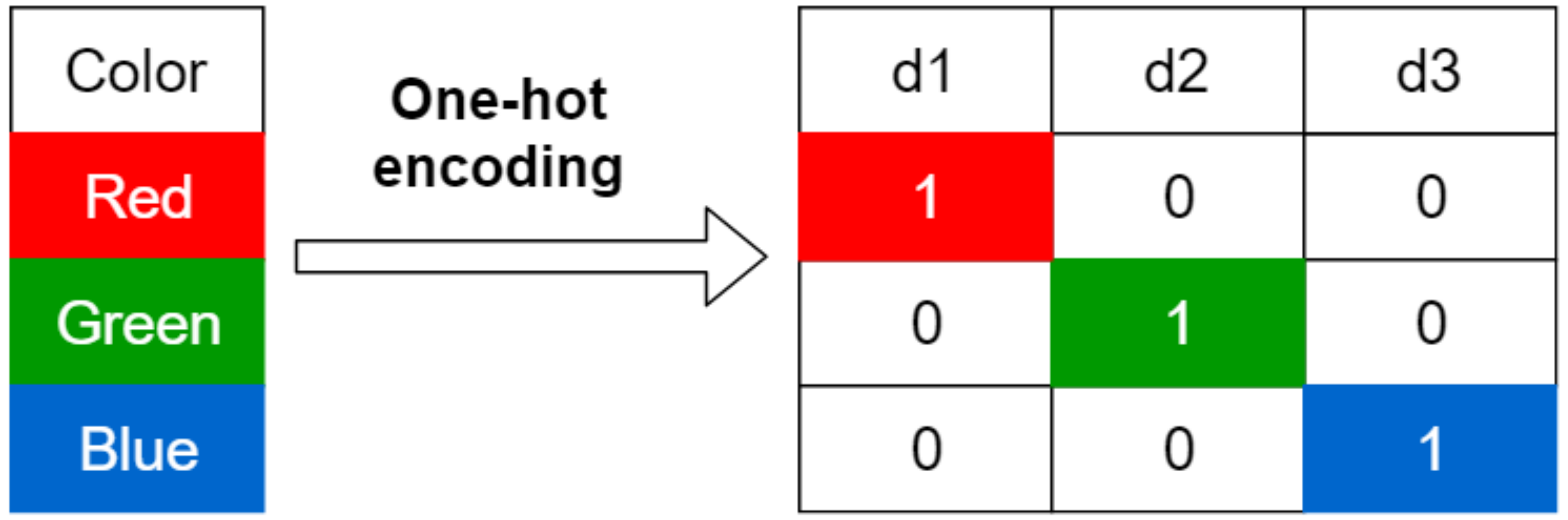
Explore Data: welcoming " the null"

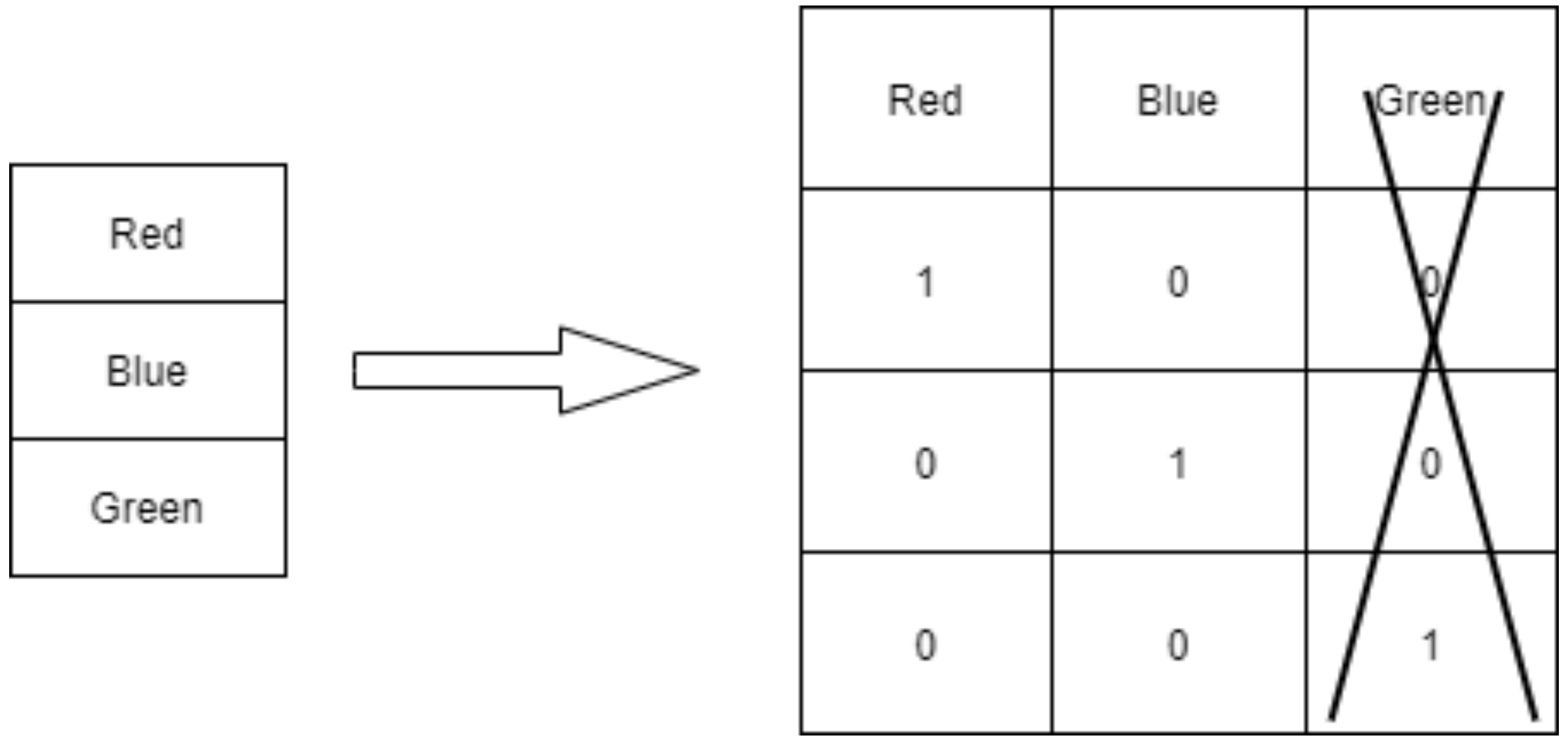
- Remove "massive null" columns having 20%+ (600,000+) missing observations.
- Impute "small null" columns: based on the distribution of a column
  - If this: 

=> Impute mean
  - If that: 

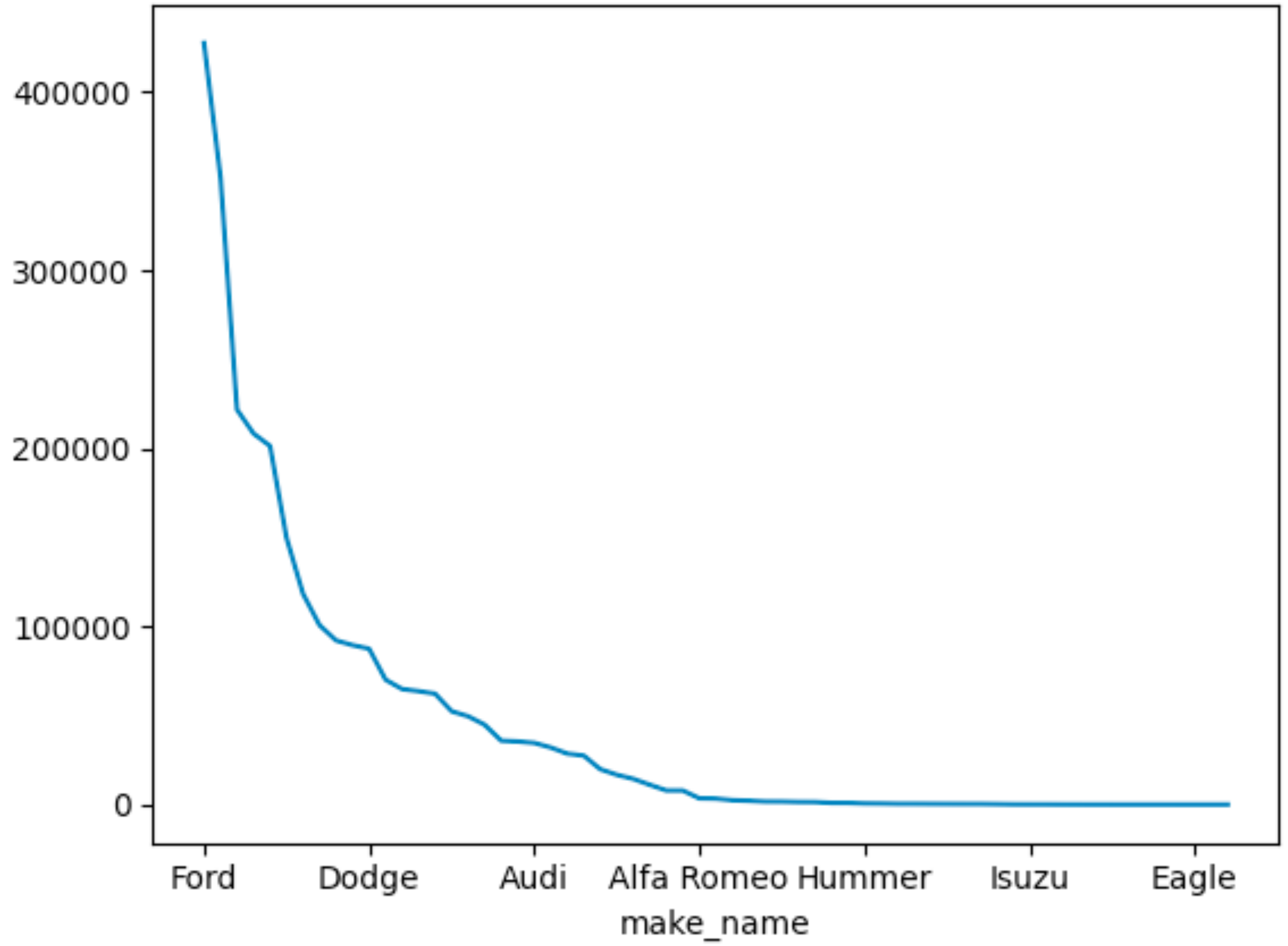
=> Impute median
  - Else: => Impute mode

Explore Data: translating "the text"





What if we're overcrowded ⁉️

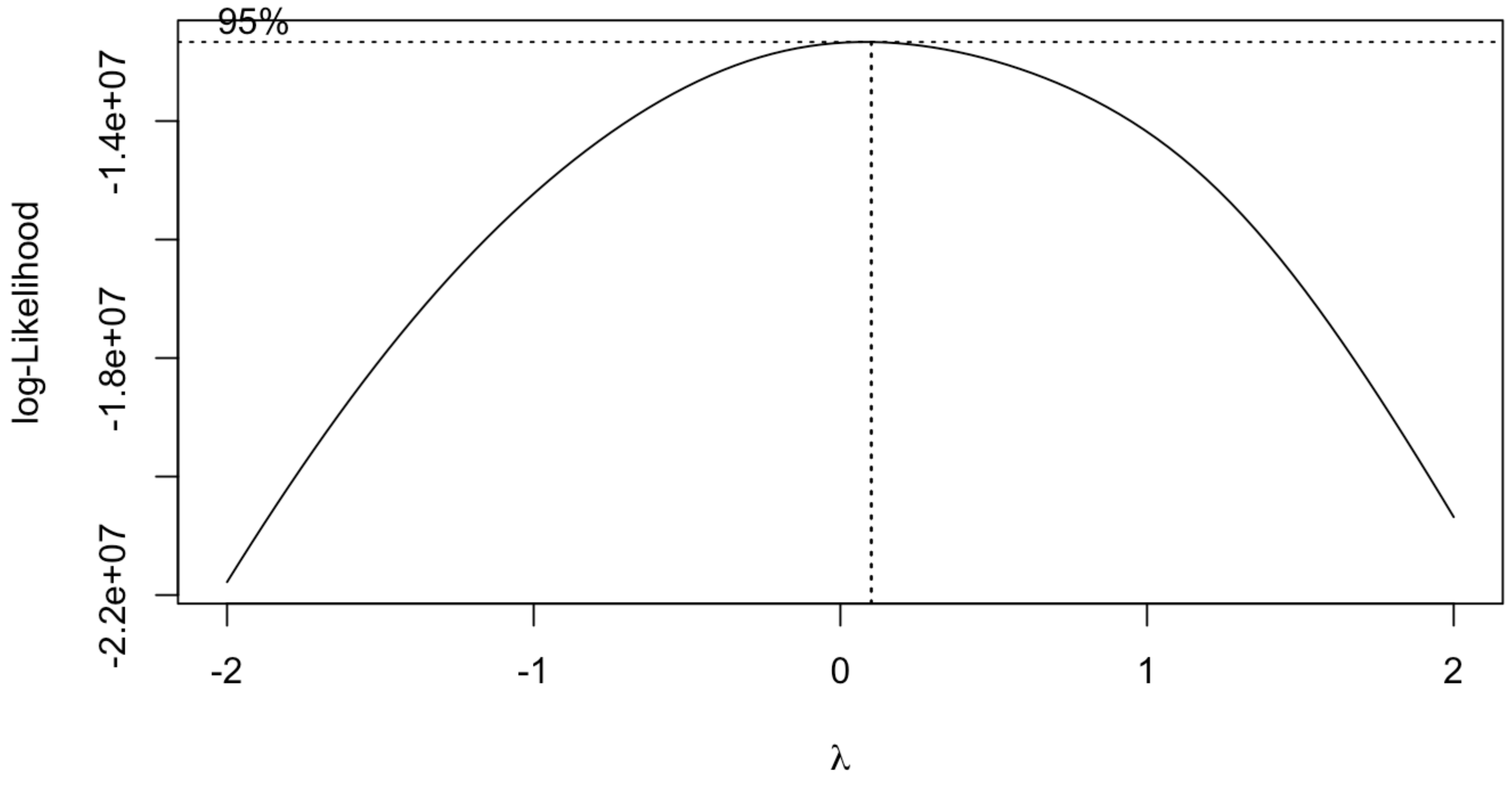


- What do I want from encoding? 1 to 2 same meaning as 2 to 3 <= IQR Frequency Encoding.

The Full Linear Model: preparing the ingredient

- Do I want to transform y?

=> YES



- Do I want to transform X? <= When do I need to transform X?
  - Non-Linearity $\xrightarrow{\text{magic}}$ Linearity
  - Something needs it (PCA, PLS, Ridge, ...)

=> JUST IN CASE

=>

```
get_standardized_df <- function(df) {
  # if (a column is non-encoded) {
  #   standardize(column)
  # }
}
```

=> std_X_train, std_y_train, std_...

The Full Linear Model: let it cook

- What's the R-Squared(s)?

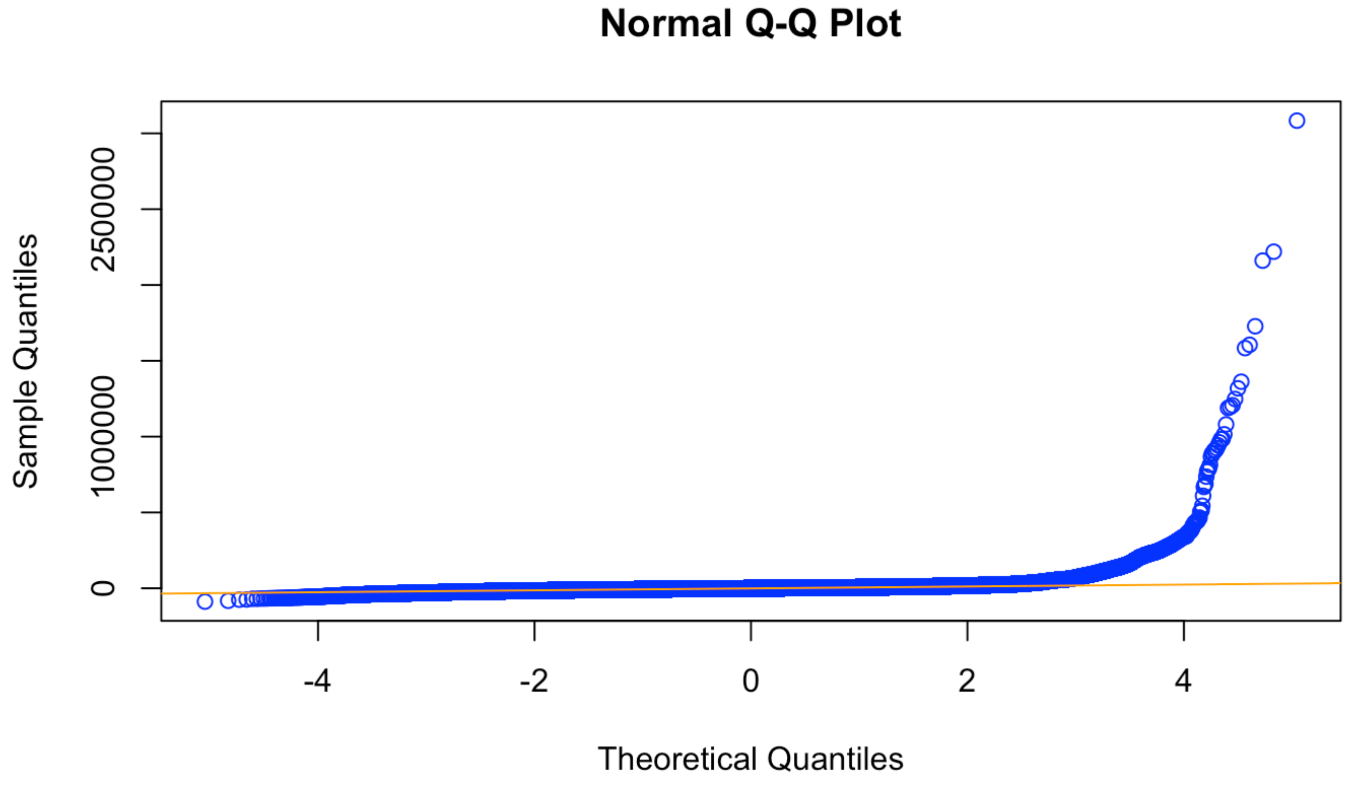
```
Residual standard error: 10610 on 2201001 degrees of freedom
Multiple R-squared:  0.674,	Adjusted R-squared:  0.674
F-statistic: 7.583e+04 on 60 and 2201001 DF,  p-value: < 2.2e-16
```

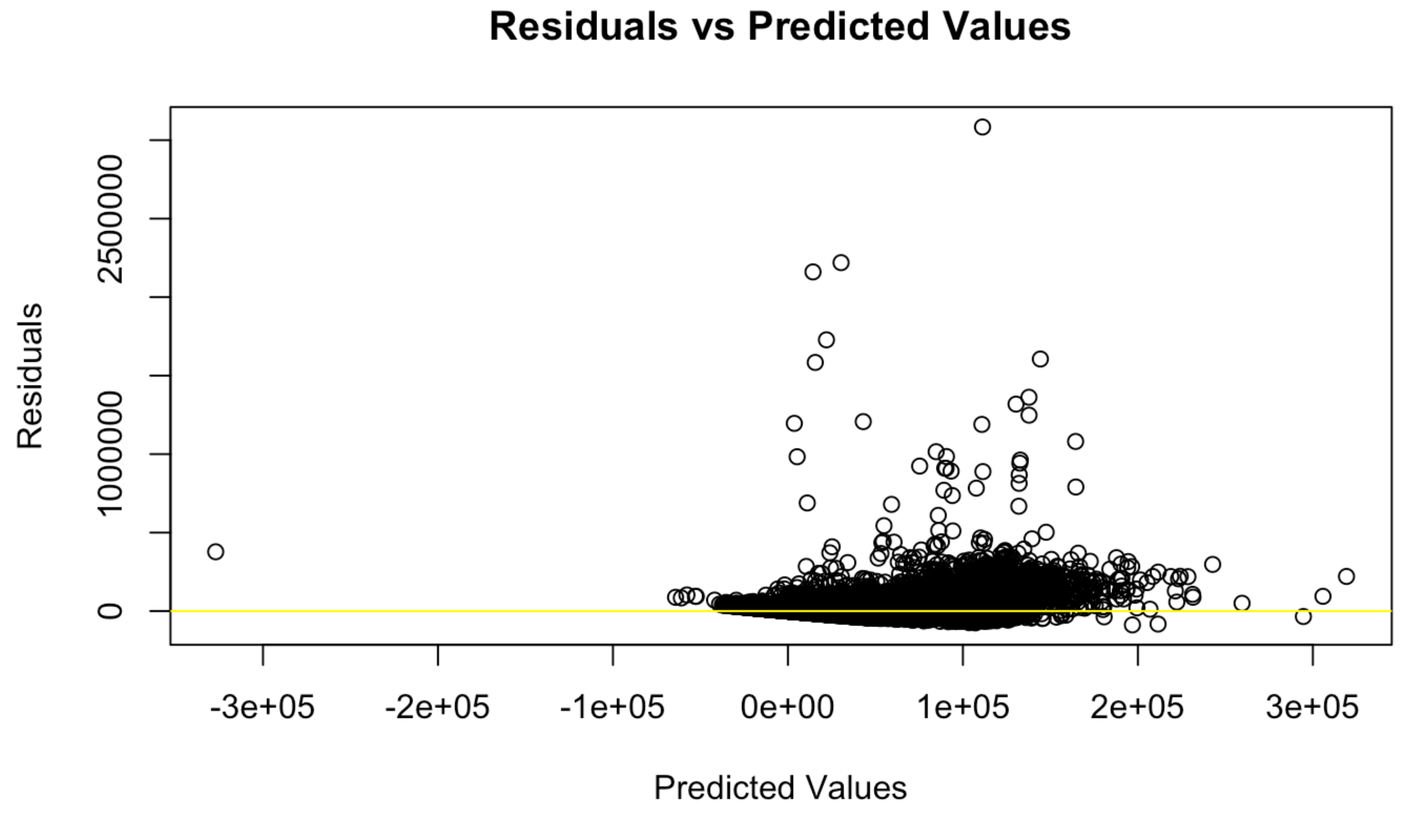
- How long did it cook?

```
  user
37.560
```

system.time({ *your_full_linear_model_here* **})**

- Does it violate normality and constant variance assumptions?





=> YES 😢

- Does multi-collinearity exist?

=> YES

```
> vif(full_linear_model)
Error in vif.default(full_linear_model) :
  there are aliased coefficients in the model
```

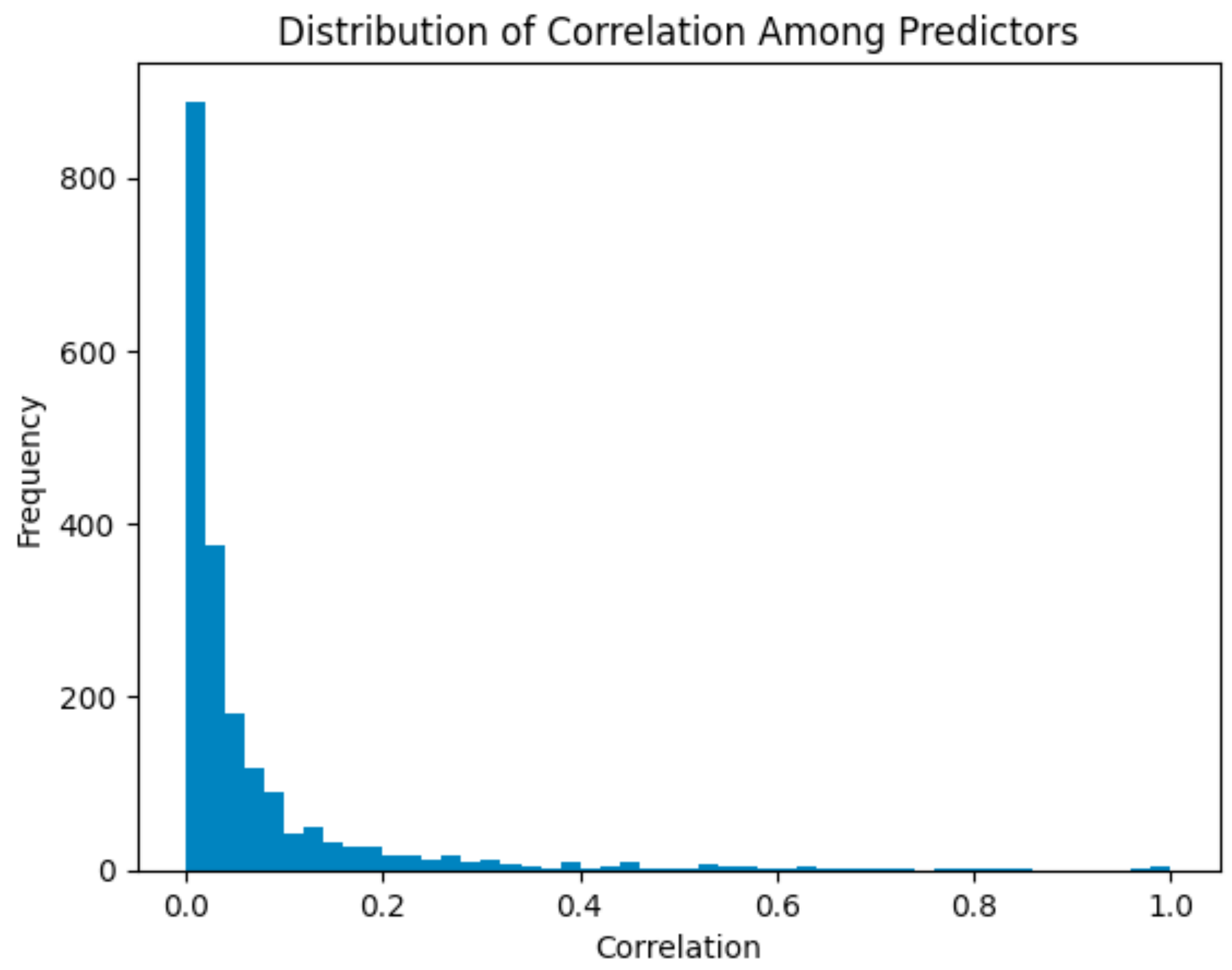
- Linearity
- Independence
- Normality
- Constant Variance

The First Red Flag: Multi-collinearity

I want: VIF(a linear model) < 5, ideally ~ 1 to 2

Principle Component Analysis

If majority of correlations among predictors are small,
PCA is not effective



Partial Least Squares

If majority of correlations with response are small,
PLS is not effective

````
```{r}
correlation_with_y <- cor(df)
correlation_without_itself <- correlation_with_y[,'price'][-which(names(correlation_with_y[,'price']) == 'price')]
mean(correlation_without_itself)
```

[1] 0.03499676
````

Ridge Regression

1.d(ridge function)= $-2y^TX + 2X^TX\hat{\beta} + 2\lambda\hat{\beta}$

2. Solution when d(ridge function) = 0:

$$\hat{\beta} = y^Tx/(x^Tx + \lambda)$$

always > 0 whatever the penalty rate

The LASSO

Solution when d(lasso function) = 0:

1. $\hat{\beta} = (y^Tx - \lambda)/(x^Tx)$ when $\lambda <= y^Tx$
2. $\hat{\beta} = (y^Tx + \lambda)/(x^Tx)$ when $\lambda >= y^Tx$

Whatever_Error(LASSO)
<
Whatever_Error(OLS)

✓

The LASSO
then OLS

The First Red Flag: Multi-collinearity

I want: VIF(a linear model) < 5, ideally ~ 1 to 2 , find how much should I penalize for the LASSO

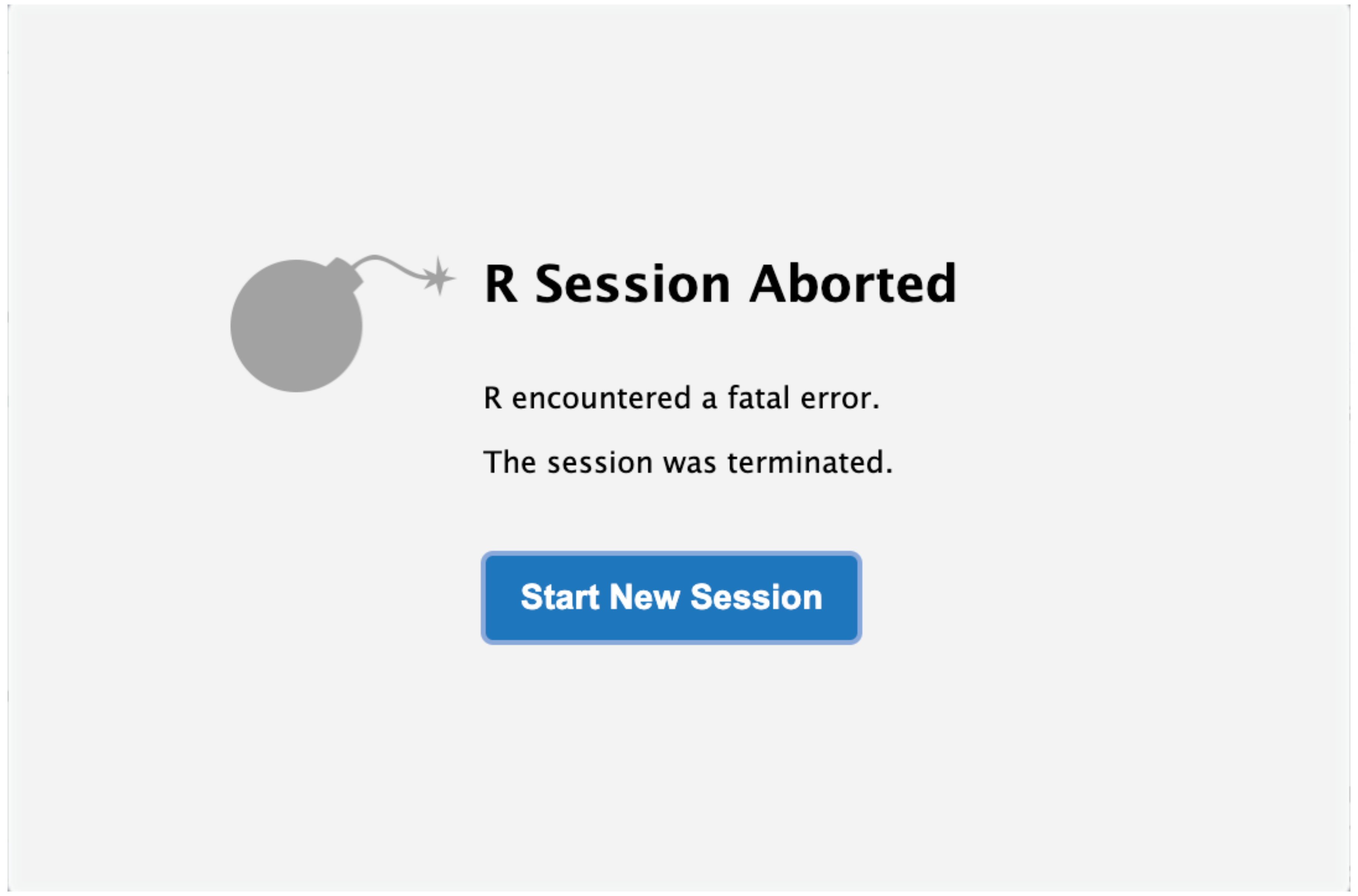
- Idea 1:
  - generate a list of penalties (lambdas), fit models, find the lambda with lowest max(VIF).

```
```{r}
find_optimal_lambda <- function(lambda_options, X_train, y_train) {
  # Standardize X_train, y_train
  for (lambda_option in lambda_options) {
    lasso_model = glmnet(std_X_train, std_y_train, alpha = 1, lambda = lambda_option)

    # Filter non-zero coefficient columns
    non_zero_rows = rownames(coef_values)[coef_values[,'s0'] != 0]
    lasso_stayed_columns <- non_zero_rows[2:length(non_zero_rows)] # Avoid (Intercept)
    # \Filter non-zero coefficient columns

    new_linear_model <- lm(y_train ~ ., data = X_train)
    cur_vif = vif(new_linear_model)

    # Update min_lambda if there is new min_vif
  }
}
lambda_options = seq(from = 0, to = 2, length.out = 100)
```
```

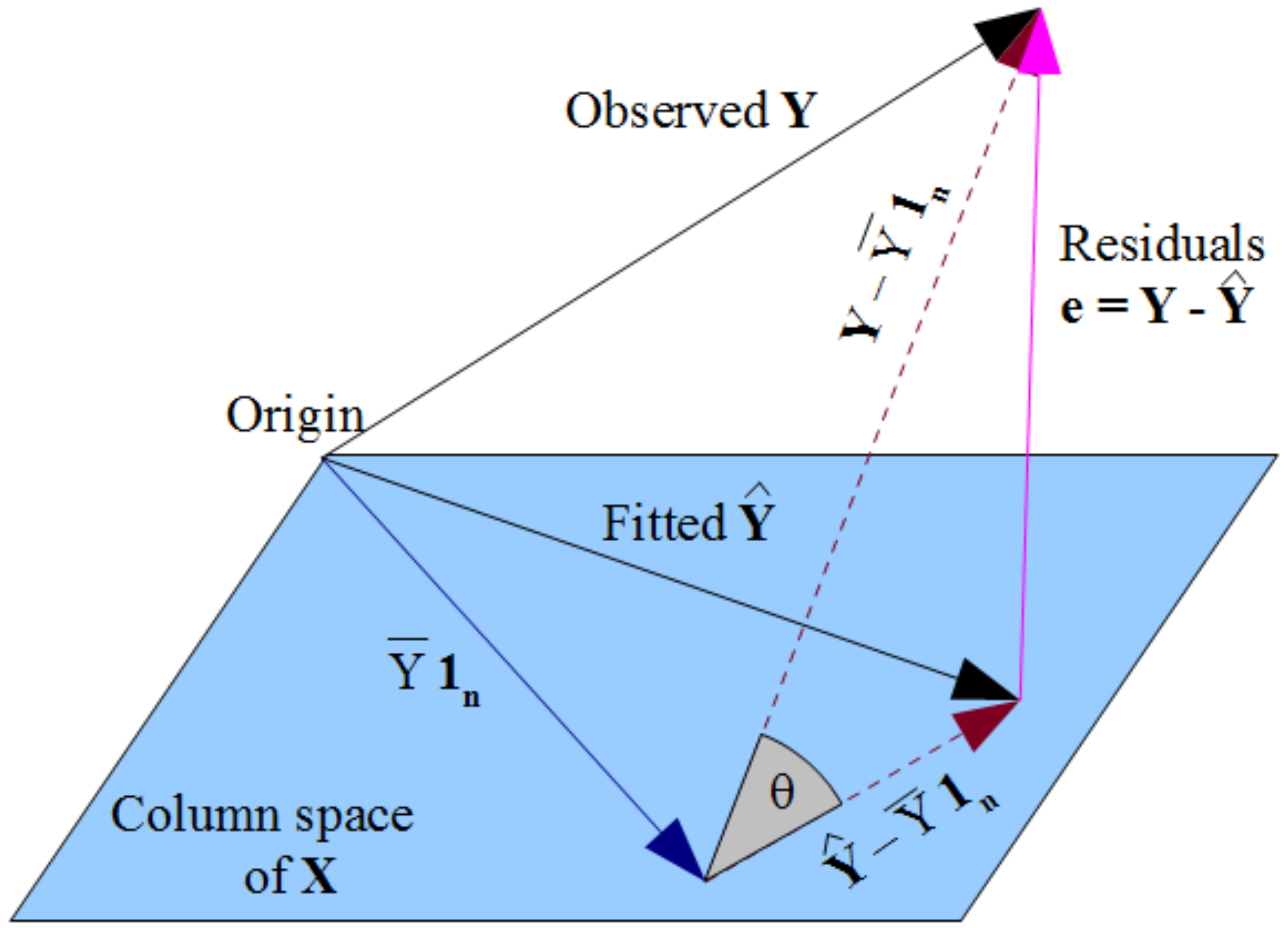


- There is a relationship between lambda and VIF => Can we compute VIF from lambda?

The First Red Flag: Multi-collinearity

I want: VIF(a linear model) < 5, ideally ~1 to 2 , find how much should I penalize for the LASSO

- Idea 2:
  - VIF = 1 / (1 – R^2)
  - R^2 is how much variance of actual_y (represented by SST) are from predicted_y (SSReg)
  - Geometrically:



=> **SSReg = cov_mat(y, X)^T @ cov_mat(X) @ cov_mat(y, X)**

=> **SST = cov_mat(y)**

=> R^2 = SSReg / SST = SSReg @ SST^-1

= cov_mat(y, X)^T @ cov_mat(X)^-1 @ cov_mat(y, X)

=> VIF = 1 / (1 – R^2) => cov_mat(X)^-1 = F(cov_mat(X))

=> VIF_regularization = F(cov_mat(X) + lambda * I)

The First Red Flag: Multi-collinearity

I want: VIF(a linear model) < 5, ideally ~ 1 to 2 , find how much should I penalize for the LASSO

- Idea 2: Proof

```python
def vif_statsmodels(X, intercept_colname):
    from statsmodels.stats.outliers_influence import variance_inflation_factor
    from statsmodels.tools.tools import add_constant

    X_copy = X.copy()
    if intercept_colname not in X_copy: X_copy = add_constant(X_copy)

    vif = pd.DataFrame()
    vif["variables"] = X_copy.columns
    vif["VIF"] = [(variance_inflation_factor(X_copy.values, i), 4) for i in range(X_copy.shape[1])]
    return vif

def vif_custom(X, lam):
    from numpy.linalg import inv

    n_cols = X.shape[1] - 1
    vif = np.zeros((len(lam), n_cols))

    rxx = X.iloc[:, :n_cols].corr().values
    rxy = X.corr().iloc[:n_cols, n_cols].values
    for i in range(len(lam)):
        tmp1 = inv(rxx + lam[i] * np.eye(n_cols))
        vif[i, :] = np.diag(np.dot(np.dot(tmp1, rxx), tmp1))

    vif_df = pd.DataFrame(vif, columns=X.columns[:-1], index=lam)

    return vif_df.T
```

```python
vif_df.T.head()
```

✓ 0.0s

	0
back_legroom	1.583718
city	1.085481
daysonmarket	1.163538
engine_displacement	6.345453
engine_type	1.990666

```python
vif_statsmodels_df.head()
```

✓ 0.0s

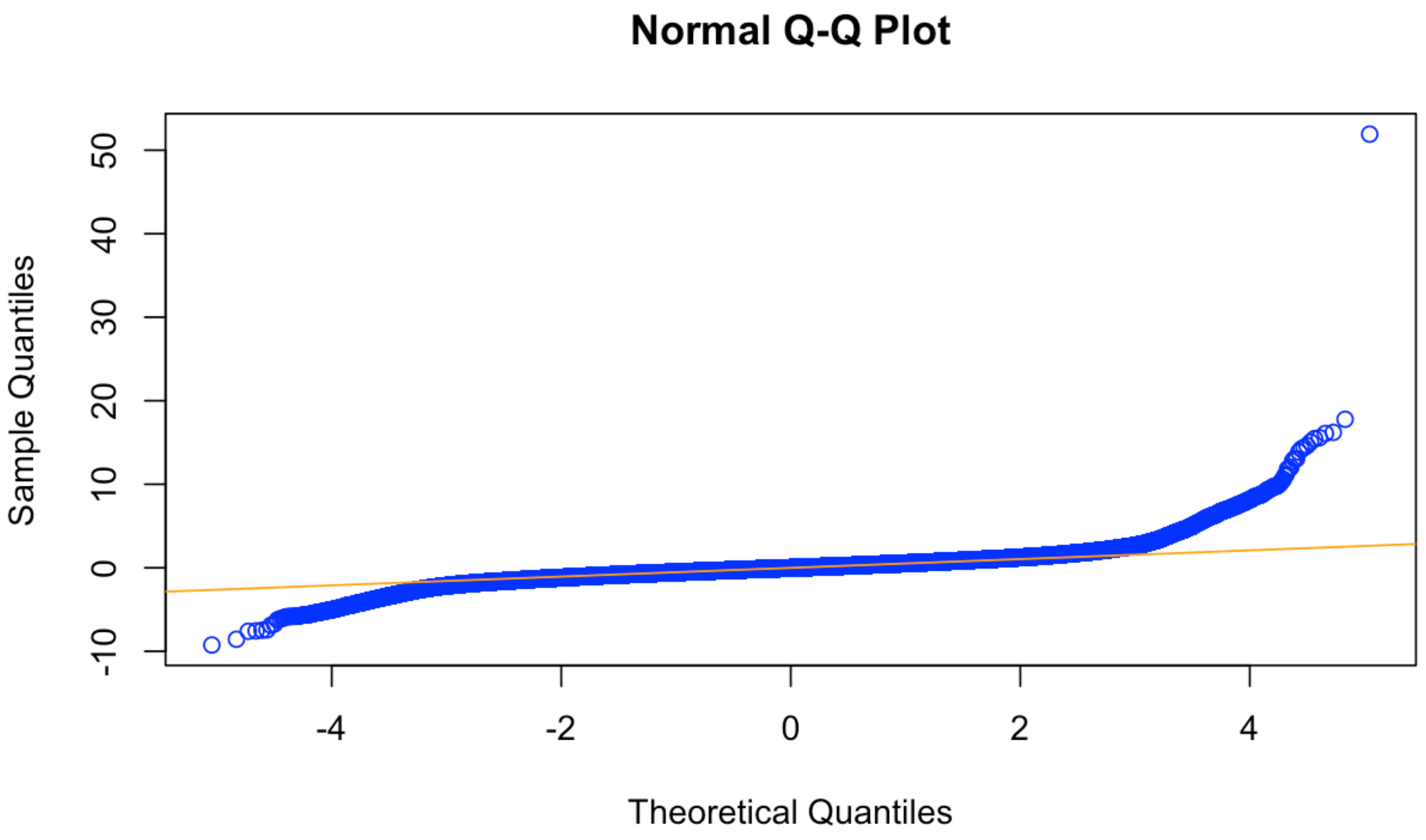
	variables	VIF
0	const	530.271953
1	back_legroom	1.583747
2	city	1.085473
3	daysonmarket	1.163727
4	engine_displacement	6.345435

The First Red Flag: Multi-collinearity

I want: VIF(a linear model) < 5, ideally ~ 1 to 2 , find how much should I penalize for the LASSO

- Idea 2:
  - generate a list of penalties (lambdas), find the smallest lambda that VIF table has > 0 rows, collect all the rows, fit each row with a linear model, then pick the model having the lowest AIC.

Result for optimal_lambda = 0.0440044




```
Coefficients:
                              Estimate Std. Error t value Pr(>|t|)    
(Intercept)                 -2.788e+02  4.449e-01 -626.73   <2e-16 ***
back_legroom                 1.225e-02  1.484e-04   82.54   <2e-16 ***
engine_type                 -2.556e-01  1.702e-03 -150.17   <2e-16 ***
fuel_tank_volume             2.041e-02  1.458e-04  139.95   <2e-16 ***
horsepower                   8.130e-03  8.112e-06 1002.18   <2e-16 ***
longitude                   -2.943e-03  2.983e-05  -98.65   <2e-16 ***
make_name                   -1.307e-01  1.809e-03  -72.24   <2e-16 ***
mileage                     -1.209e-05  1.816e-08 -665.75   <2e-16 ***
model_name                  -1.038e-01  1.690e-03  -61.45   <2e-16 ***
savings_amount               9.013e-05  4.573e-07  197.10   <2e-16 ***
seller_rating                1.320e-01  8.086e-04  163.19   <2e-16 ***
width                        1.897e-03  7.591e-05   25.00   <2e-16 ***
year                         1.462e-01  2.214e-04  660.61   <2e-16 ***
is_body_type_Sedan          -1.442e-01  1.054e-03 -136.77   <2e-16 ***
`is_body_type_Pickup Truck` -3.103e-01  1.548e-03 -200.46   <2e-16 ***
is_body_type_Convertible     4.778e-01  4.583e-03  104.24   <2e-16 ***
is_fuel_type_Diesel          5.379e-01  4.236e-03  126.97   <2e-16 ***
`is_fuel_type_Flex Fuel Vehicle` -3.865e-01  2.003e-03 -192.96   <2e-16 ***
is_fuel_type_Hybrid          4.225e-01  2.614e-03  161.62   <2e-16 ***
is_is_new_False             -2.910e-01  1.245e-03 -233.71   <2e-16 ***
is_franchise_dealer_False   -2.183e-01  1.262e-03 -172.91   <2e-16 ***
is_wheel_system_FWD         -4.654e-01  1.046e-03 -444.75   <2e-16 ***
is_wheel_system_4X2         -2.686e-01  2.190e-03 -122.64   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6067 on 2201039 degrees of freedom
Multiple R-squared:  0.861,	Adjusted R-squared:  0.861 
F-statistic: 6.199e+05 on 22 and 2201039 DF,  p-value: < 2.2e-16
```

The Second Red Flag: Normality

- Should I use Robust Regression?
- <= How much percentage of outliers are there and are most of them influential?

```r
abnormal_points = get_abnormal_points(lassoAIC_linear_model)
influentials <- abnormal_points$influential_points

outliers <- abnormal_points$outliers
nonInfluential_outliers = setdiff(influentials, outliers)
influential_outliers = intersect(influentials, outliers)

leverages <- abnormal_points$leverage_points
nonInfluential_leverages = setdiff(influentials, leverages)
influential_leverages = intersect(influentials, leverages)

all_abnormal_points <- union(outliers, leverages)

cat(length(all_abnormal_points) * 100 / dim(lassoAIC_X_train)[1], "\n")
cat(length(influentials) * 100 / dim(lassoAIC_X_train)[1], "\n")
cat(length(nonInfluential_outliers) * 100 / length(outliers), "\n")
cat(length(nonInfluential_leverages) * 100 / length(leverages), "\n")
```
```

```
12.62141
6.542206
74.95397
27.23772
```

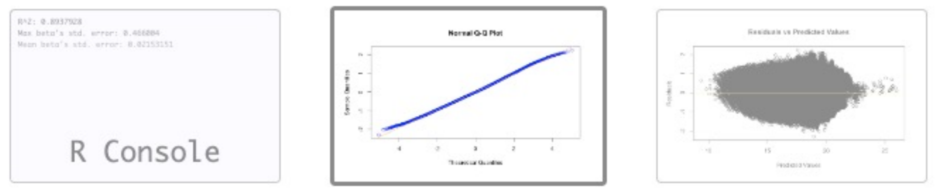
=>

````
```{r}
removed_lassoAIC_X_train <- lassoAIC_X_train[!(rownames(lassoAIC_X_train) %in% influentials), ]
removed_y_train <- y_train[!(index(y_train) %in% influentials)]
removed_boxcox_y_train <- boxcox_y_train[!(index(boxcox_y_train) %in% influentials)]

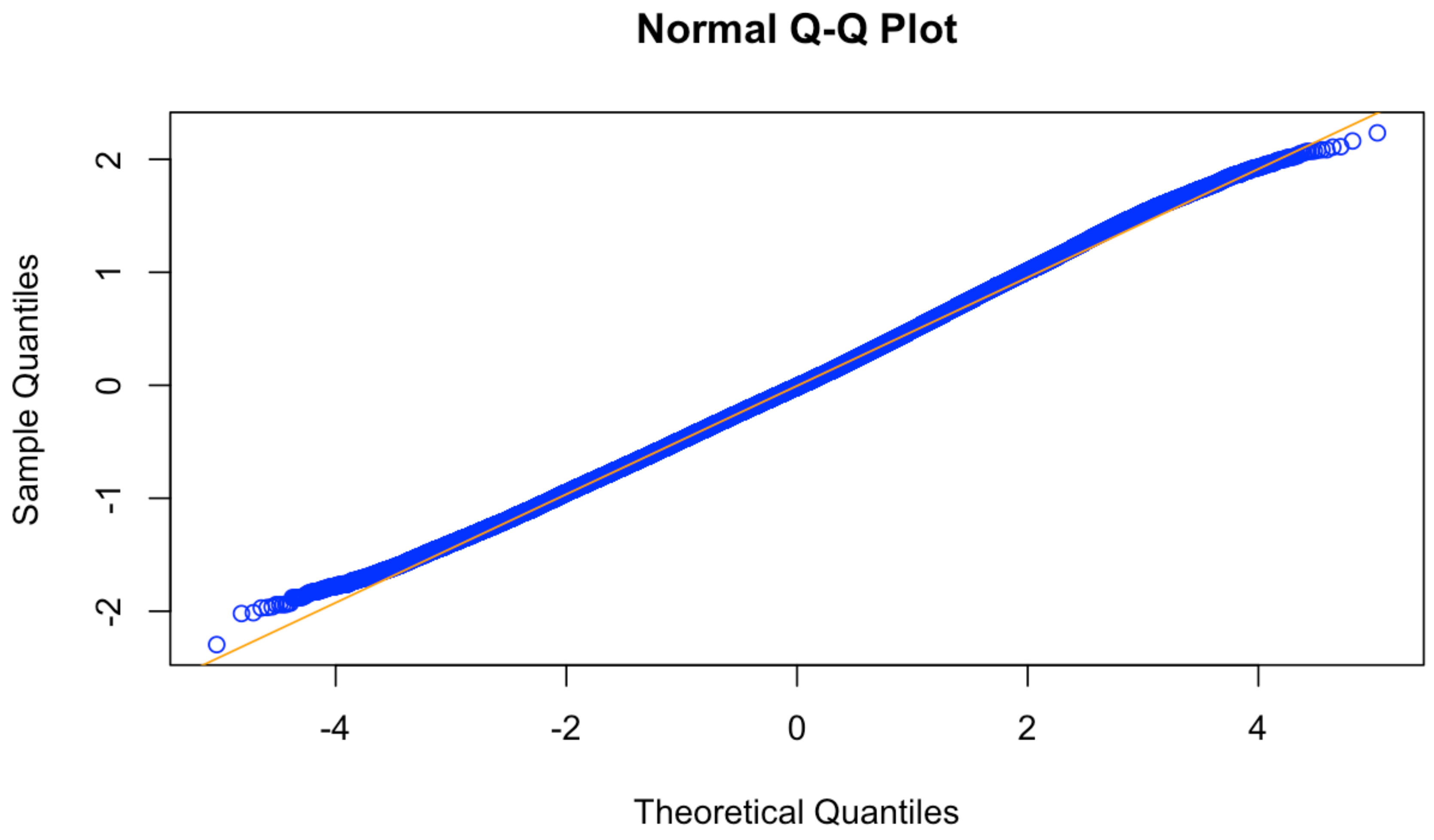
removed_lassoAIC_linear_model <- lm(formula = removed_boxcox_y_train ~ ., data = removed_lassoAIC_X_train)

results <- summary(removed_lassoAIC_linear_model)
cat("R^2:", results$r.squared, "\n")
cat("Max beta's std. error:", max(results$coefficients[,2]), "\n")
cat("Mean beta's std. error:", mean(results$coefficients[,2]), "\n")

check_assumptions(removed_lassoAIC_linear_model, fitted(removed_lassoAIC_linear_model))
```
````



=>


````

Improvements

- Can do a better job in encoding
- Huge datasets mean common assumptions tests like Shapiro-Wilk don't work in R

=> Assumptions validation is subjective

- Never forget to add training and testing result again! (4100~ training error, testing error ~ 4900)

Thank you for listening!