Predictive Modeling Report

Giodanno Limin

2025-11-27

Contents

```r
# get the training data
train <- read.csv("trainingdata.csv")

# direct access
attach(train)
# write a cv_rmse function
cv_rmse <- function(formula, data, K = 10) {
  set.seed(1)
  n <- nrow(data)
  folds <- sample(rep(1:K, length.out = n))

  rmse <- numeric(K)

  for (k in 1:K) {
    train_fold <- data[folds != k, ]
    test_fold  <- data[folds == k, ]

    fit <- lm(formula, data=train_fold)
    pred <- predict(fit, newdata=test_fold)

    rmse[k] <- sqrt(mean((pred - test_fold$y)^2))
  }

  return(list(
    mean_RMSE = mean(rmse),
    sd_RMSE   = sd(rmse),
    all_RMSE  = rmse
  ))
}
```

```r
lm1 = lm(y~.,data=train)
summary(lm1)
```

```
##
## Call:
## lm(formula = y ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.40482 -0.36592 -0.04287  0.27987  2.08131
```

```
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.836e+00  1.285e+00  -1.428 0.154835
## X1          -1.244e-01  1.741e-01  -0.714 0.475945
## X2          -3.155e-02  1.675e-01  -0.188 0.850772
## X3          -5.407e-03  1.359e-02  -0.398 0.691125
## X4           1.087e-01  1.669e-01   0.651 0.515867
## X5           5.496e-02  1.750e-01   0.314 0.753820
## X6          -1.084e-02  1.814e-01  -0.060 0.952411
## X7          -1.069e-01  4.619e-02  -2.314 0.021754 *
## X8           1.281e-02  4.768e-02   0.269 0.788477
## X9          -9.935e-02  4.636e-02  -2.143 0.033388 *
## X10          1.234e-02  4.846e-02   0.255 0.799310
## X11          7.327e-03  4.632e-02   0.158 0.874471
## X12          2.535e-01  1.700e-01   1.491 0.137711
## X13         -2.748e-01  1.676e-01  -1.640 0.102688
## X14          1.093e-01  1.584e-01   0.690 0.491207
## X15          6.753e-02  1.627e-01   0.415 0.678518
## X16          1.821e-01  1.702e-01   1.070 0.285863
## X17          8.294e-02  4.769e-02   1.739 0.083638 .
## X18         -2.416e-02  5.150e-02  -0.469 0.639496
## X19          8.760e-02  4.314e-02   2.030 0.043730 *
## X20         -3.196e-02  1.651e-01  -0.194 0.846704
## X21          4.557e-02  7.392e-03   6.165 4.24e-09 ***
## X22         -9.078e-03  4.641e-02  -0.196 0.845136
## X23          1.773e-02  4.684e-02   0.379 0.705401
## X24         -2.134e-02  4.716e-02  -0.452 0.651525
## X25         -1.274e-01  1.719e-01  -0.741 0.459598
## X26          1.752e-02  1.637e-01   0.107 0.914886
## X27         -1.039e-01  1.659e-01  -0.627 0.531665
## X28          1.733e-01  1.642e-01   1.055 0.292700
## X29          1.317e-02  4.700e-02   0.280 0.779661
## X30         -9.838e-02  3.017e-02  -3.260 0.001323 **
## X31          9.855e-02  1.668e-01   0.591 0.555418
## X32          1.385e-02  1.072e-01   0.129 0.897335
## X33         -2.752e-02  1.629e-01  -0.169 0.866020
## X34          1.455e-01  1.681e-01   0.866 0.387872
## X35         -2.209e-01  1.736e-01  -1.272 0.204831
## X36         -2.453e-03  3.882e-03  -0.632 0.528307
## X37         -1.959e-02  1.665e-01  -0.118 0.906462
## X38          1.836e-05  1.013e-04   0.181 0.856392
## X39         -1.197e-01  1.564e-01  -0.765 0.445066
## X40          4.388e-01  1.685e-01   2.605 0.009937 **
## X41          1.551e-01  1.147e-01   1.351 0.178233
## X42          1.318e-02  4.826e-02   0.273 0.785022
## X43          1.679e-01  1.708e-01   0.983 0.326747
## X44          3.610e-02  1.737e-01   0.208 0.835568
## X45         -3.232e-02  7.625e-02  -0.424 0.672183
## X46         -2.032e-01  1.713e-01  -1.187 0.236865
## X47          3.214e-02  1.660e-01   0.194 0.846703
## X48         -1.260e-02  2.972e-02  -0.424 0.672116
## X49         -5.978e-02  1.653e-01  -0.362 0.718031
## X50          1.771e-02  1.614e-01   0.110 0.912759
```

```
## X51          8.932e-02  1.806e-01   0.494 0.621544
## X52         -1.835e-01  1.766e-01  -1.039 0.300114
## X53          3.609e-02  1.730e-01   0.209 0.835017
## X54          1.805e-02  4.775e-02   0.378 0.705803
## X55         -1.776e-01  1.795e-01  -0.989 0.323782
## X56          1.639e-01  1.684e-01   0.973 0.331777
## X57         -9.789e-02  1.711e-01  -0.572 0.568028
## X58          2.930e-02  4.481e-02   0.654 0.514000
## X59         -7.999e-02  6.554e-02  -1.220 0.223814
## X60         -6.243e-02  4.555e-02  -1.370 0.172196
## X61         -4.652e-02  4.842e-02  -0.961 0.337910
## X62         -1.158e-02  1.774e-01  -0.065 0.948023
## X63          9.788e-02  1.741e-01   0.562 0.574619
## X64         -6.892e-02  4.710e-02  -1.463 0.145115
## X65          1.156e-02  2.238e-01   0.052 0.958841
## X66          5.121e-02  4.589e-02   1.116 0.265862
## X67          2.353e-01  1.613e-01   1.459 0.146250
## X68          1.476e-01  1.658e-01   0.891 0.374203
## X69         -1.332e-02  4.877e-02  -0.273 0.785088
## X70          4.670e-02  4.646e-02   1.005 0.316192
## X71          7.835e-02  4.736e-02   1.654 0.099722 .
## X72         -4.705e-02  1.755e-01  -0.268 0.788985
## X73          5.781e-02  4.611e-02   1.254 0.211556
## X74         -1.121e-01  1.801e-01  -0.622 0.534543
## X75         -4.404e-02  1.714e-01  -0.257 0.797470
## X76         -6.731e-03  1.619e-01  -0.042 0.966891
## X77         -4.527e-03  4.596e-02  -0.099 0.921640
## X78         -3.080e-01  1.635e-01  -1.884 0.061144 .
## X79         -5.365e-02  1.701e-01  -0.315 0.752839
## X80          5.361e-02  4.567e-02   1.174 0.241975
## X81         -1.099e-02  4.608e-02  -0.238 0.811805
## X82         -1.016e-01  1.603e-01  -0.634 0.527028
## X83          3.613e-02  4.447e-02   0.812 0.417566
## X84         -5.795e-02  1.678e-01  -0.345 0.730214
## X85          7.753e-02  1.779e-01   0.436 0.663545
## X86         -3.172e-01  1.649e-01  -1.924 0.055922 .
## X87         -3.311e-01  1.608e-01  -2.059 0.040895 *
## X88          1.053e-01  4.483e-02   2.350 0.019817 *
## X89          5.708e-02  4.830e-02   1.182 0.238821
## X90         -2.541e-02  4.021e-02  -0.632 0.528302
## X91          8.141e-03  4.964e-02   0.164 0.869909
## X92         -1.035e-02  1.637e-01  -0.063 0.949637
## X93          1.041e-02  4.654e-02   0.224 0.823168
## X94         -1.596e-02  1.785e-01  -0.089 0.928847
## X95          2.680e-01  1.637e-01   1.637 0.103335
## X96          2.732e-03  4.564e-02   0.060 0.952322
## X97          2.712e-01  1.783e-01   1.521 0.129900
## X98          2.004e-01  1.718e-01   1.166 0.245107
## X99         -1.650e-01  1.674e-01  -0.986 0.325498
## X100         4.192e-02  4.946e-02   0.848 0.397738
## X101         2.041e-01  1.671e-01   1.222 0.223433
## X102        -9.294e-03  5.394e-02  -0.172 0.863376
## X103         2.856e-02  4.616e-02   0.619 0.536885
## X104         1.412e-02  1.551e-01   0.091 0.927562
```

```
## X105        -6.734e-03  1.711e-01  -0.039 0.968653
## X106        -1.323e-02  4.811e-02  -0.275 0.783576
## X107        -1.684e-02  4.641e-02  -0.363 0.717120
## X108        -9.836e-05  7.310e-05  -1.346 0.180074
## X109        -1.411e-01  1.734e-01  -0.814 0.416648
## X110        -1.984e-01  1.583e-01  -1.253 0.211699
## X111         1.065e+00  2.877e-01   3.702 0.000281 ***
## X112        -4.978e-04  4.910e-02  -0.010 0.991921
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6459 on 187 degrees of freedom
## Multiple R-squared:  0.6955, Adjusted R-squared:  0.5131
## F-statistic: 3.814 on 112 and 187 DF,  p-value: 3.227e-16
```

```
cv_rmse(lm1,train)
```

```
## $mean_RMSE
## [1] 0.829424
##
## $sd_RMSE
## [1] 0.1684319
##
## $all_RMSE
##  [1] 0.9346653 0.9864522 0.9430970 0.8060121 0.7775150 0.3984574 0.7934964
##  [8] 0.9436986 0.8869845 0.8238614
```

```
sig_vars <- summary(lm1)$coefficients
sig_vars[sig_vars[, "Pr(>|t|)"] < 0.05, ]
```

```
##         Estimate  Std. Error   t value     Pr(>|t|)
## X7   -0.10687732 0.046186860 -2.314020 2.175352e-02
## X9   -0.09935425 0.046357655 -2.143211 3.338833e-02
## X19   0.08759752 0.043142547  2.030421 4.373009e-02
## X21   0.04557020 0.007391737  6.165019 4.243496e-09
## X30  -0.09837712 0.030174855 -3.260235 1.322716e-03
## X40   0.43883711 0.168482914  2.604639 9.936539e-03
## X87  -0.33108819 0.160812368 -2.058848 4.089515e-02
## X88   0.10534484 0.044828094  2.349974 1.981740e-02
## X111  1.06523359 0.287735588  3.702127 2.812017e-04
```

```
lm2 = lm(y~X7+X9+X19+X21+X30+X40+X87+X88+X111,data=train)
summary(lm2)
```

```
##
## Call:
## lm(formula = y ~ X7 + X9 + X19 + X21 + X30 + X40 + X87 + X88 +
##     X111, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.2782 -0.3967 -0.1091  0.3479  2.3933
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.524555   0.277289  -9.104  < 2e-16 ***
```

```
## X7          -0.056952   0.043327  -1.314   0.1897
## X9          -0.083958   0.041889  -2.004   0.0460 *
## X19          0.266808   0.027271   9.784  < 2e-16 ***
## X21          0.050279   0.006094   8.250 5.61e-15 ***
## X30         -0.053695   0.026520  -2.025   0.0438 *
## X40          0.279613   0.110431   2.532   0.0119 *
## X87         -0.271269   0.108995  -2.489   0.0134 *
## X88          0.048036   0.042130   1.140   0.2551
## X111         1.455985   0.264727   5.500 8.34e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7167 on 290 degrees of freedom
## Multiple R-squared:  0.4186, Adjusted R-squared:  0.4005
## F-statistic:  23.2 on 9 and 290 DF,  p-value: < 2.2e-16
```

```
cv_rmse(lm2,train)
```

```
## $mean_RMSE
## [1] 0.7277796
##
## $sd_RMSE
## [1] 0.1247808
##
## $all_RMSE
##  [1] 0.6820813 0.7446164 0.8649398 0.8521659 0.8207434 0.4932418 0.5423195
##  [8] 0.7996716 0.7368539 0.7411627
```

Let's try better way

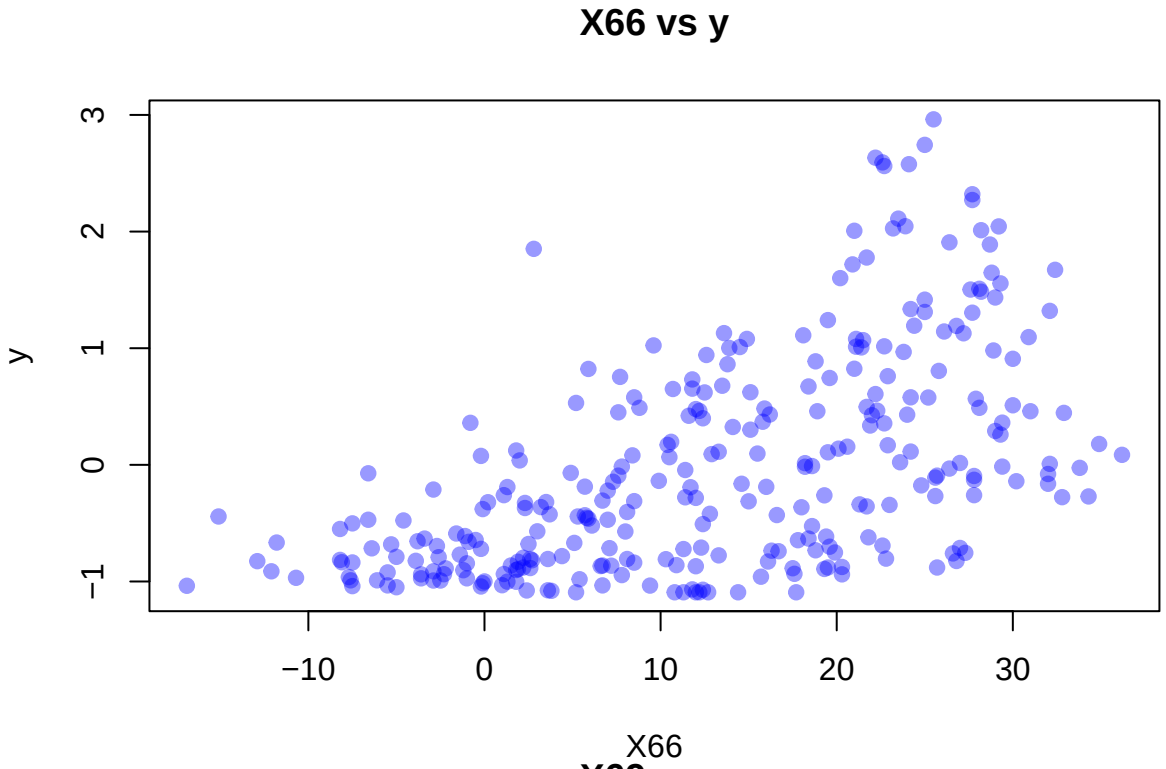
```
cor_with_y <- function(x) {
  cor(train$y, x)
}
cors <- sapply(train[, -1], cor_with_y)
head(sort(abs(cors), decreasing=TRUE), 10)
```

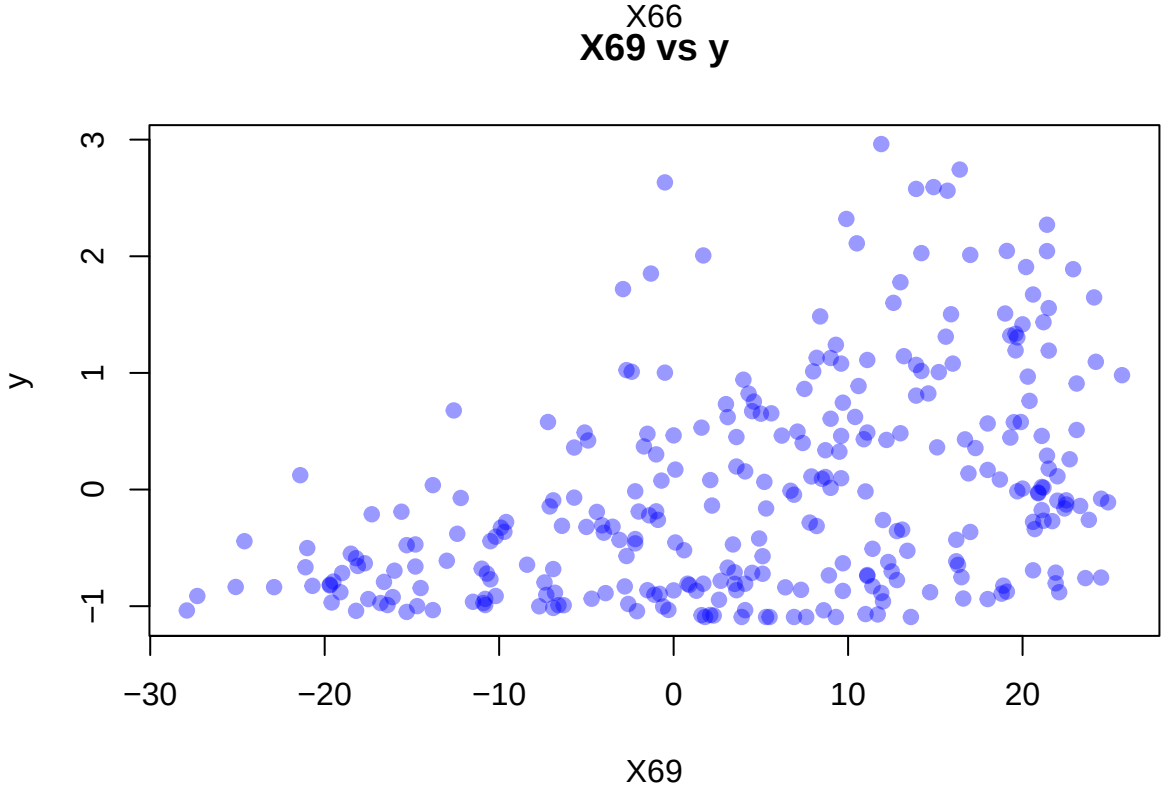
```
##       X66       X69       X21       X19       X45      X111       X41       X38
## 0.5700591 0.4227833 0.4184403 0.4107010 0.2733106 0.1926304 0.1697941 0.1653557
##        X3       X59
## 0.1652030 0.1417632
```

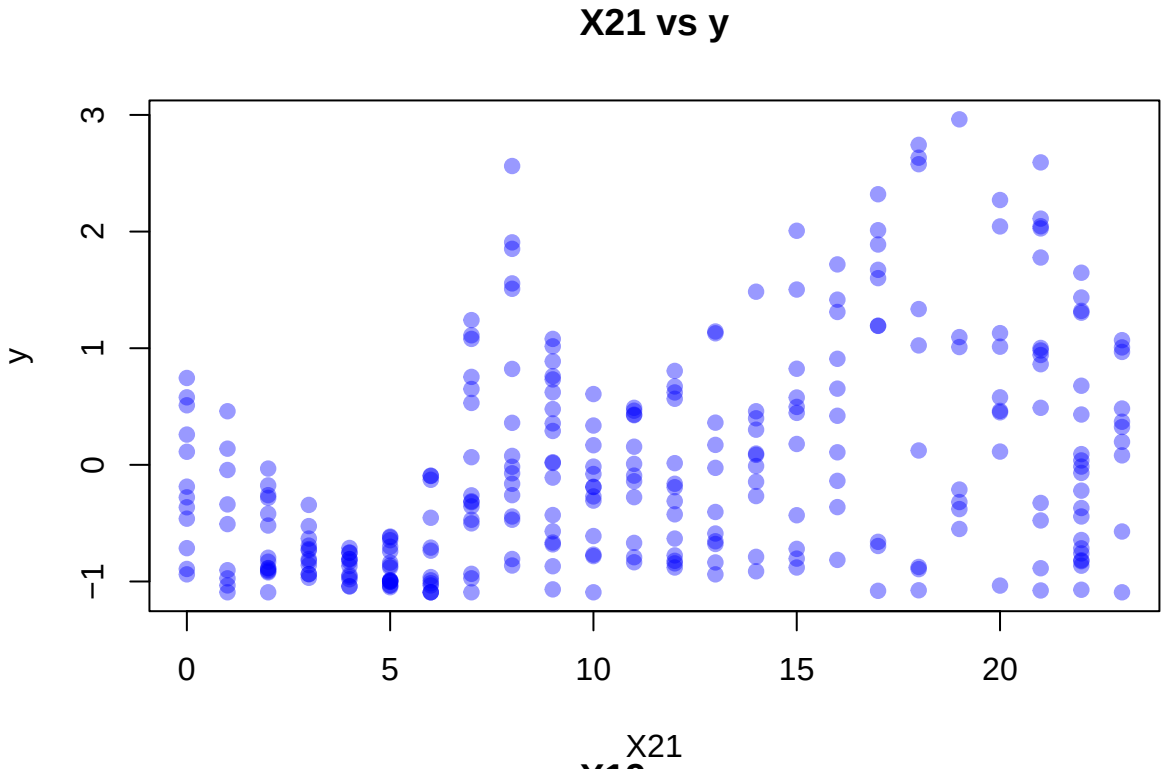
```
top10 <- names(sort(abs(cors), decreasing = TRUE))[1:10]
top10
```

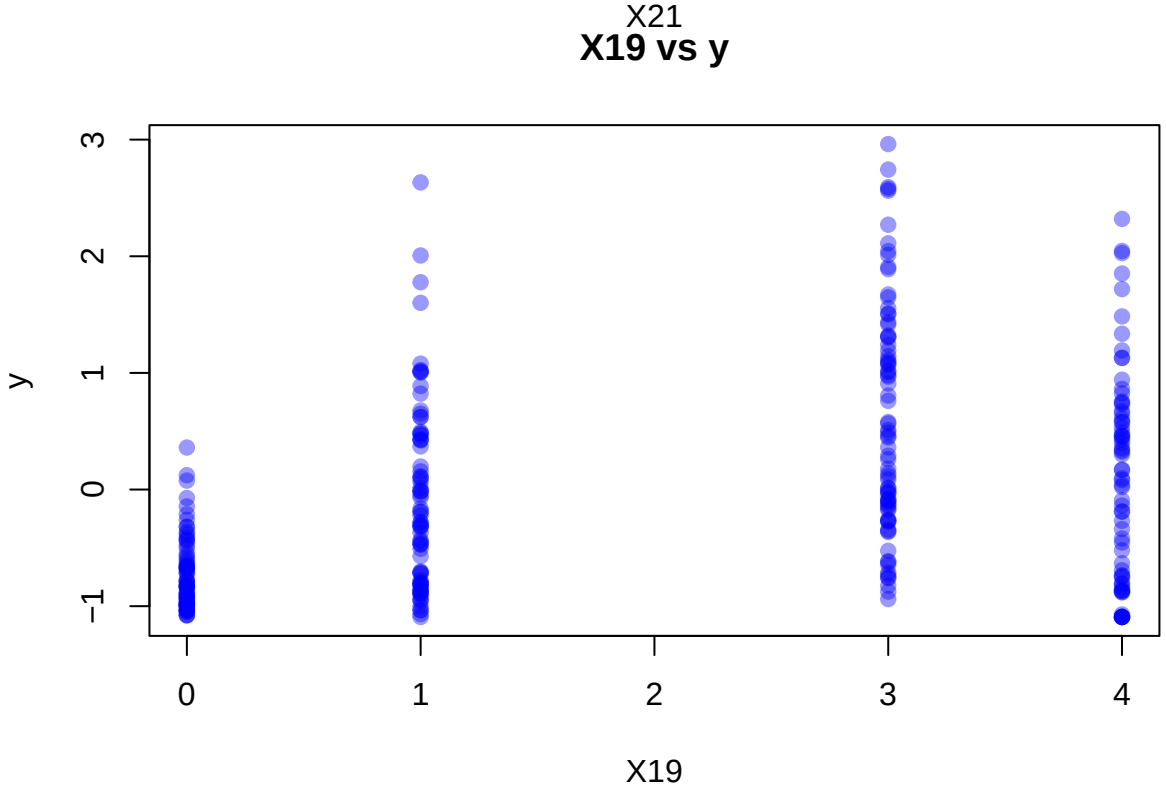
```
##  [1] "X66"  "X69"  "X21"  "X19"  "X45"  "X111" "X41"  "X38"  "X3"   "X59"
```

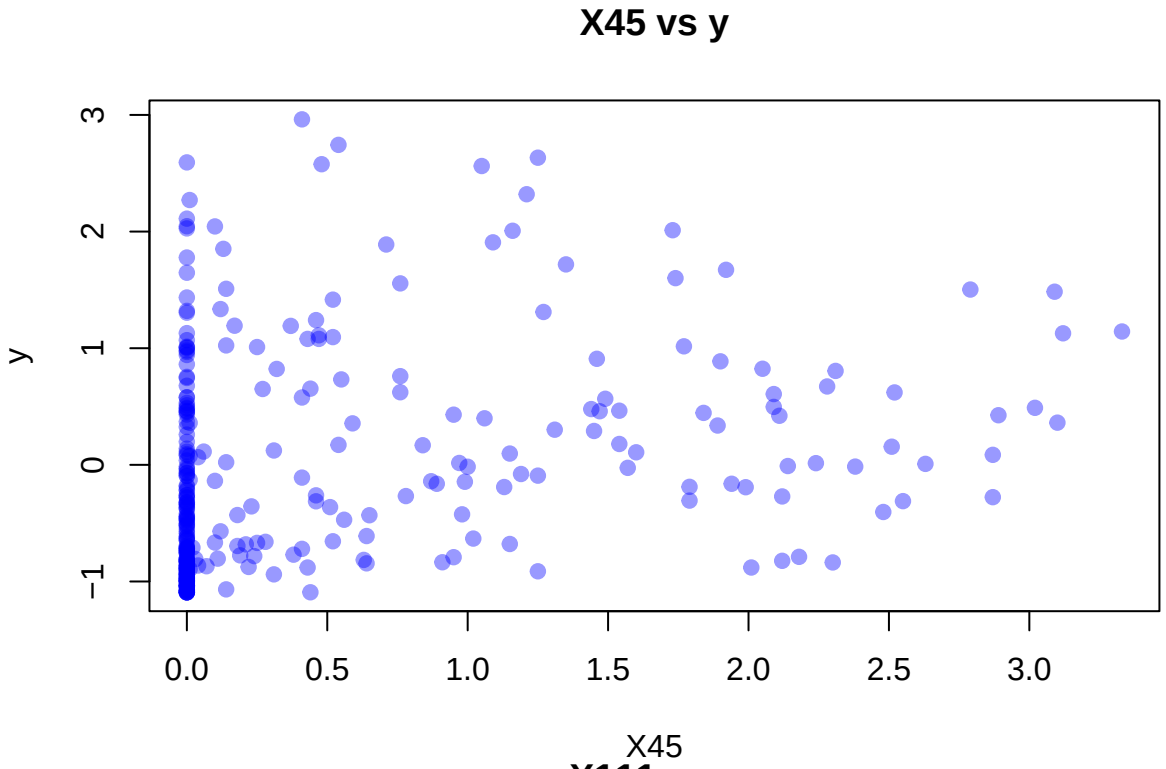
```
for (var in top10) {
  plot(
    train[[var]], train$y,
    main = paste(var, "vs y"),
    xlab = var,
    ylab = "y",
    pch = 19,
    col = rgb(0, 0, 1, 0.4)
  )
}
```

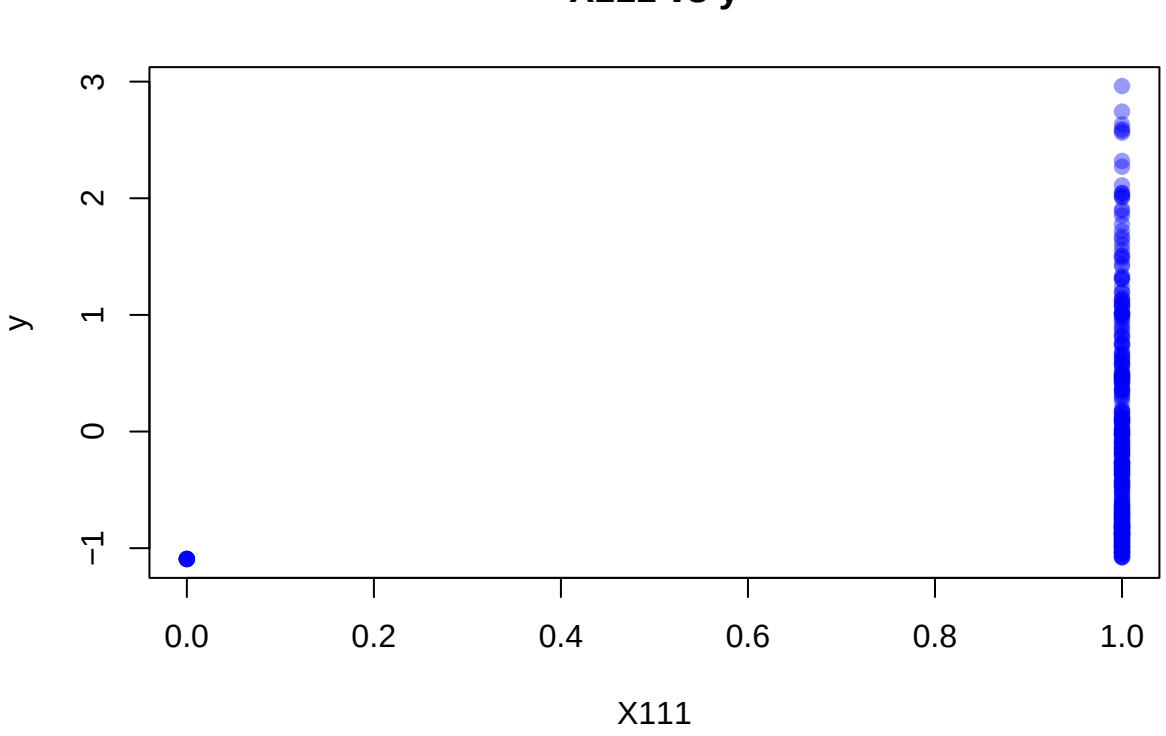

X66 vs y
y
X66
−10
0
10
20
30
−1
0
1
2
3

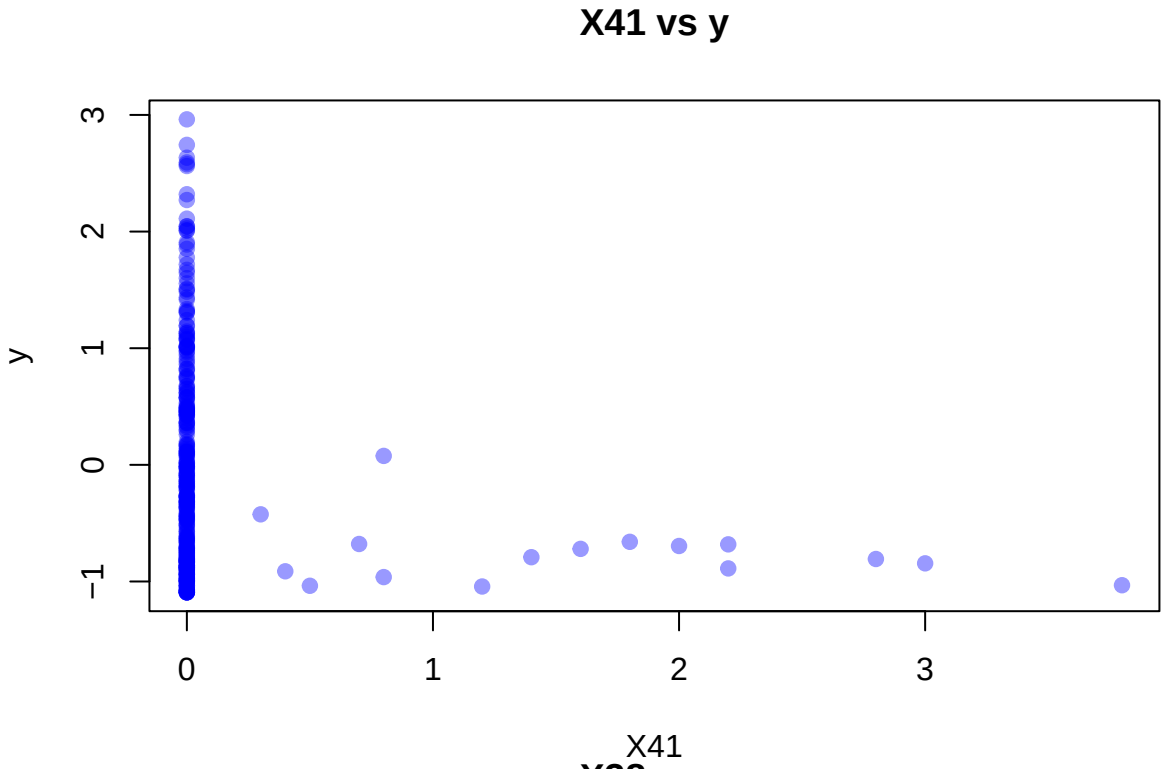

X69 vs y
y
X69
−30
−20
−10
0
10
20
−1
0
1
2
3

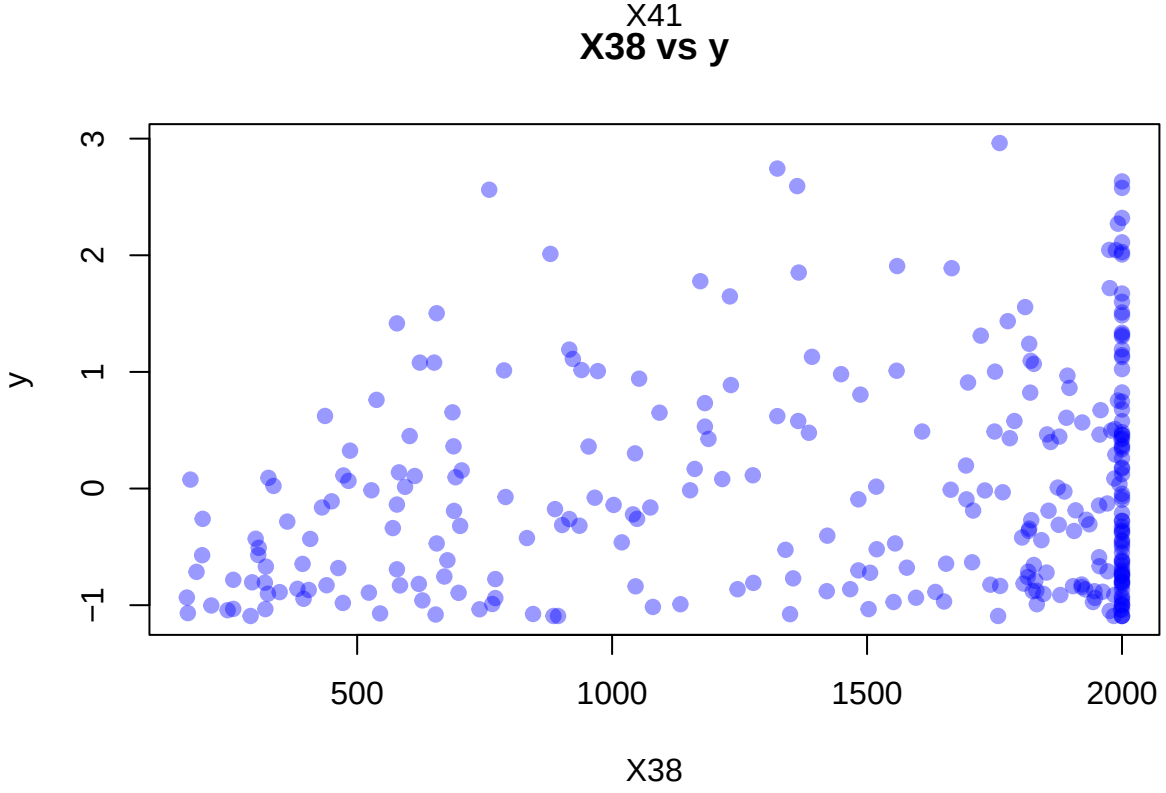

X21 vs y
y
X21
−1
0
1
2
3
0
5
10
15
20

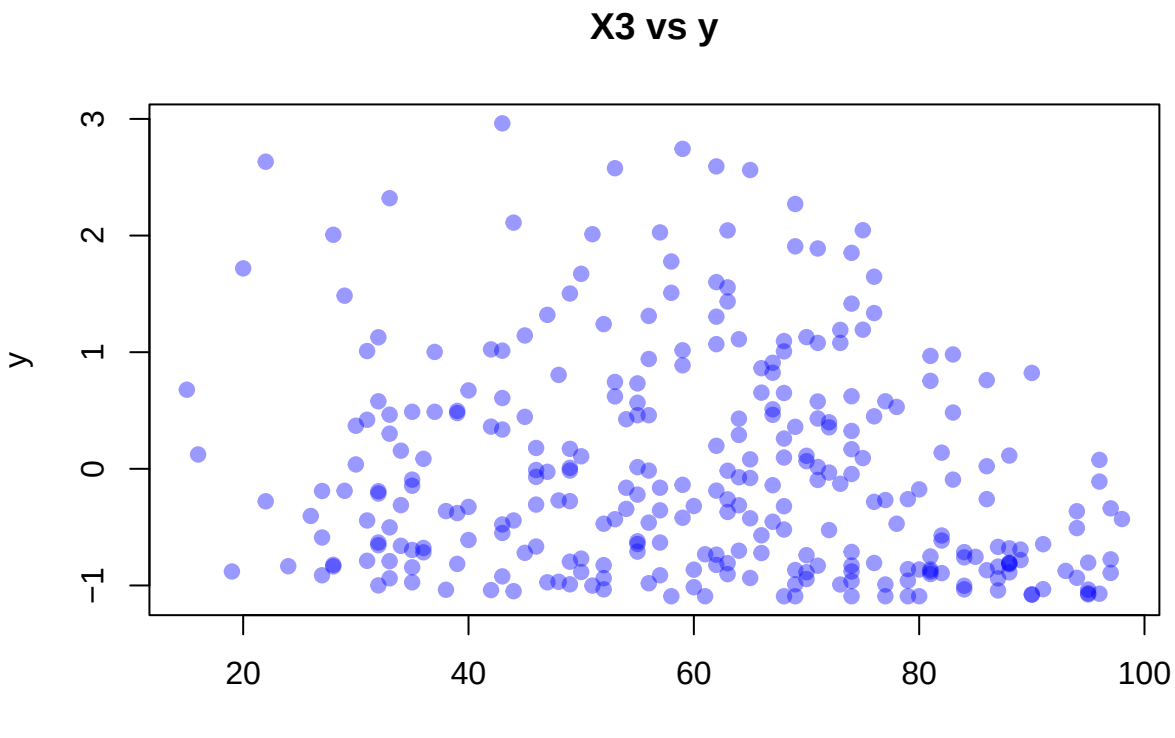

X19 vs y
y
X19
−1
0
1
2
3
0
1
2
3
4

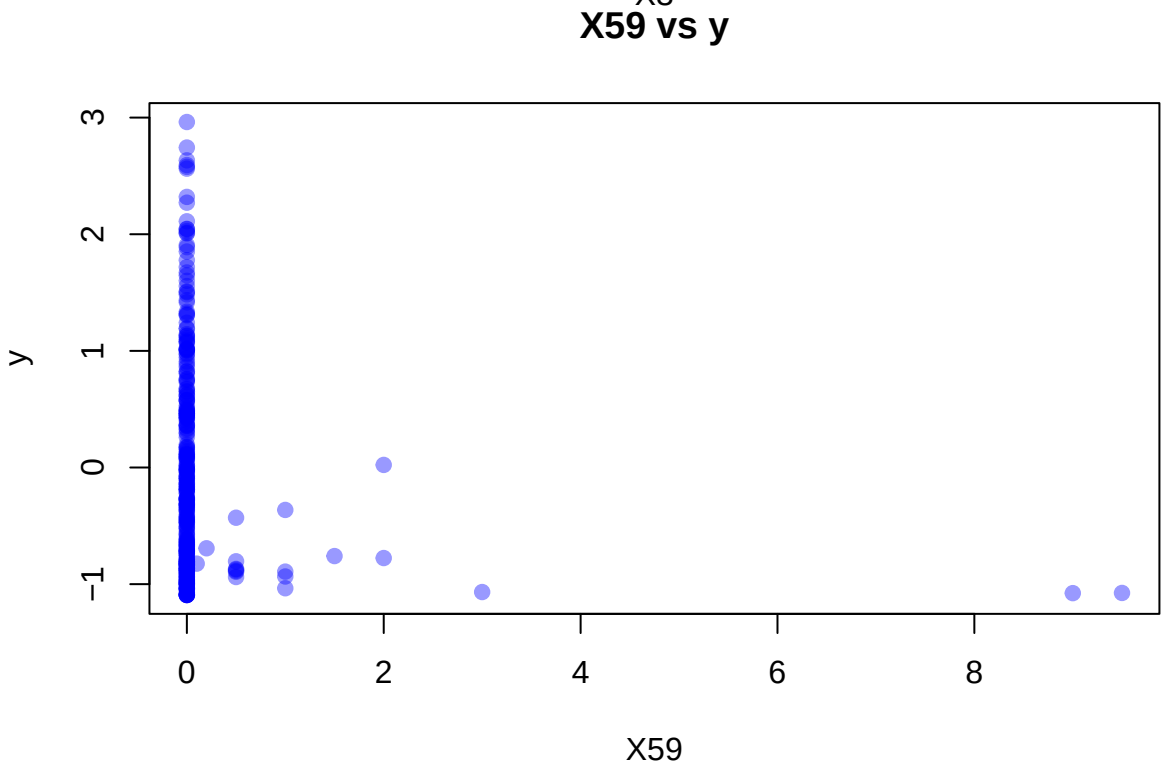

X45 vs y
y
3
2
1
0
−1
0.0
0.5
1.0
1.5
2.0
2.5
3.0
X45


X111 vs y
y
3
2
1
0
−1
0.0
0.2
0.4
0.6
0.8
1.0
X111


X41 vs y
y
3
2
1
0
−1
0
1
2
3
X41


X38 vs y
y
3
2
1
0
−1
500
1000
1500
2000
X38






```
lm4 = lm(y~X66+X69+X21,data=train)
summary(lm4)
```

```
##
## Call:
## lm(formula = y ~ X66 + X69 + X21, data = train)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.58451 -0.42403 -0.09015  0.35334  2.56191
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.294099   0.096381 -13.427  < 2e-16 ***
## X66          0.075799   0.008765   8.648 3.37e-16 ***
## X69         -0.033818   0.007909  -4.276 2.57e-05 ***
## X21          0.040972   0.005901   6.944 2.42e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6722 on 296 degrees of freedom
## Multiple R-squared:  0.4779, Adjusted R-squared:  0.4726
## F-statistic: 90.33 on 3 and 296 DF,  p-value: < 2.2e-16
```

```
cv_rmse(lm4,train)
```

```
## $mean_RMSE
## [1] 0.6701781
##
## $sd_RMSE
## [1] 0.09868585
##
## $all_RMSE
##  [1] 0.6039787 0.7512891 0.7767443 0.6238047 0.7737403 0.4714130 0.5797160
##  [8] 0.7229067 0.6865790 0.7116088
```

Now let's check if the adding other variable lower the mean_RMSE "X19" "X45" "X111" "X41" "X38" "X3" "X59"

```
vars_to_test <- c("X19", "X45", "X111", "X41", "X38", "X3", "X59")

results <- data.frame(
  variable = vars_to_test,
  mean_RMSE = NA
)

base_form <- "y ~ X66 + X69 + X21"
base_rmse <- cv_rmse(as.formula(base_form), train)$mean_RMSE

results <- data.frame(
  variable = vars_to_test,
  mean_RMSE = NA,
  improvement = NA
)

for (i in seq_along(vars_to_test)) {

  var <- vars_to_test[i]

  # construct formula dynamically
  form <- as.formula(
    paste(base_form, "+", var)
```

```
  )

  # compute rmse
  new_rmse <- cv_rmse(form, train)$mean_RMSE

  # fill results
  results$mean_RMSE[i] <- new_rmse
  results$improvement[i] <- new_rmse < base_rmse
}

results
```

```
##   variable mean_RMSE improvement
## 1      X19 0.6672321        TRUE
## 2      X45 0.6709039       FALSE
## 3     X111 0.6555636        TRUE
## 4      X41 0.6724600       FALSE
## 5      X38 0.6712700       FALSE
## 6       X3 0.6672124        TRUE
## 7      X59 0.6641478        TRUE
```

```
lm5 = lm(y~X66+X69+X21+X19+X111+X3+X59,data=train)
cv_rmse(lm5,train)
```

```
## $mean_RMSE
## [1] 0.6411805
##
## $sd_RMSE
## [1] 0.1118216
##
## $all_RMSE
##  [1] 0.6100652 0.6954469 0.7567000 0.6371614 0.7518331 0.4045187 0.4985694
##  [8] 0.7063156 0.6701002 0.6810947
```

Now let's try polynomial for X66, X69, X21, and X3.

```
search_poly_multivar <- function(max66 = 10, max69 = 10, max21 = 10) {
  results <- data.frame()

  for (d66 in 1:max66) {
    for (d69 in 1:max69) {
      for (d21 in 1:max21) {

        form <- as.formula(
          paste(
            "y ~ poly(X66,", d66, ",raw=TRUE) +",
            "poly(X69,", d69, ",raw=TRUE) +",
            "poly(X21,", d21, ",raw=TRUE) +",
            "+ X19 + X111 + X3 + X59",
            sep=" "
          )
        )

        cv <- cv_rmse(form, train)$mean_RMSE

        results <- rbind(
```

```
          results,
          data.frame(
            d66 = d66,
            d69 = d69,
            d21 = d21,
            RMSE = cv
          )
        )
      }
    }
  }

  return(results)
}

# run the search
poly_results <- search_poly_multivar()
best_row <- poly_results[which.min(poly_results$RMSE), ]
best_row
```

```
##     d66 d69 d21      RMSE
## 430   5   3  10 0.5068829
```

```
search_poly_X3 <- function(max3=10) {
  results <- data.frame()

  for (d3 in 1:max3){

    form <- as.formula(
      paste(
        "y ~",
        "poly(X66, 5, raw=TRUE) +",
        "poly(X69, 3, raw=TRUE) +",
        "poly(X21, 10, raw=TRUE) +",
        "poly(X3,", d3, ", raw=TRUE) +",
        "X19 + X111 + X59",
        sep=" "
      )
    )

    cv <- cv_rmse(form, train)$mean_RMSE

    results <- rbind(
      results,
      data.frame(
        d3 = d3,
        RMSE = cv
      )
    )
  }

  return(results)
}
```

```r
# run the search
poly_results <- search_poly_X3()

best_row <- poly_results[ which.min(poly_results$RMSE) , ]
best_row
```

```
##   d3      RMSE
## 3  3 0.4961006
```

```r
# let's check again for X21, since previously our max degree is 10,
# maybe if the degree greater than 10 is better

search_poly_X21 <- function(max21 = 20) {
  results <- data.frame()

  for (d21 in 10:max21) {

    form <- as.formula(
      paste(
        "y ~",
        "poly(X66, 5, raw=TRUE) +",
        "poly(X69, 3, raw=TRUE) +",
        "poly(X3,  3, raw=TRUE) +",
        "poly(X21,", d21, ", raw=TRUE) +",
        "X19 + X111 + X59",
        sep=" "
      )
    )

    cv <- cv_rmse(form, train)$mean_RMSE

    results <- rbind(results, data.frame(d21=d21, RMSE=cv))
  }

  return(results)
}
poly21_results <- search_poly_X21(max21 = 15)

# best degree
best_row <- poly21_results[which.min(poly21_results$RMSE), ]
best_row
```

```
##   d21      RMSE
## 5  14 0.4931188
```

```r
lm6 = lm(y~poly(X66, 5, raw=TRUE)+
           poly(X69, 3, raw=TRUE)+
           poly(X21, 14, raw=TRUE)+
           poly(X3, 3, raw=TRUE)+
           + X19 + X111 + X59)
cv_rmse(lm6,train)
```

```
## $mean_RMSE
## [1] 0.4931188
##
```

```
## $sd_RMSE
## [1] 0.07598946
##
## $all_RMSE
##  [1] 0.5195799 0.5533502 0.5138166 0.4268541 0.5644166 0.4426720 0.3197027
##  [8] 0.5090485 0.5285066 0.5532411
```

Now let's check the interactions for X66, X69, X21, X3

```
base_rmse <- cv_rmse(lm6, train)$mean_RMSE

base_string <- paste(
  "y ~",
  "poly(X66, 5, raw=TRUE) +",
  "poly(X69, 3, raw=TRUE) +",
  "poly(X21,14, raw=TRUE) +",
  "poly(X3,  3, raw=TRUE) +",
  "+ X19 + X111 + X59"
)

candidates <- c(
  "X66:X69",
  "X66:X21",
  "X66:X3",
  "X69:X21",
  "X69:X3",
  "X21:X3"
)

test_interactions <- function(base_string, candidates, train) {

  results <- data.frame()

  for (int in candidates) {
    full_form_string <- paste(base_string, "+", int)
    form <- as.formula(full_form_string)

    cv <- cv_rmse(form, train)$mean_RMSE

    results <- rbind(
      results,
      data.frame(
        interaction = int,
        RMSE = cv
      )
    )
  }

  return(results)
}
interaction_results <- test_interactions(base_string, candidates, train)
improved <- interaction_results$RMSE < base_rmse
interaction_results[ improved , ]
```

```
##   interaction      RMSE
```

```
## 2     X66:X21 0.4737751
## 4     X69:X21 0.4850535
## 6      X21:X3 0.4928832
```

```
lm7 = lm(y~poly(X66, 5, raw=TRUE)+
           poly(X69, 3, raw=TRUE)+
           poly(X21, 14, raw=TRUE)+
           poly(X3, 3, raw=TRUE)+
           +X66:X21+X69:X21+X21:X3+
           +X19+X111+X59)
cv_rmse(lm7,train)
```

```
## $mean_RMSE
## [1] 0.461731
##
## $sd_RMSE
## [1] 0.06842499
##
## $all_RMSE
##  [1] 0.4493378 0.5376595 0.5001442 0.4012809 0.4938008 0.4157965 0.3281635
##  [8] 0.4936212 0.4423954 0.5551105
```

Next let's try 2nd degree of interactions ()

```
generate_interactions2 <- function(var1, var2) {
  terms <- c()

  for (i in 1:2) {
    for (j in 1:2) {
      terms <- c(terms, paste0("I(", var1, "^", i, " * ", var2, "^", j, ")"))
    }
  }

  return(terms)
}

int_X66_X21 <- generate_interactions2("X66", "X21")
int_X69_X21 <- generate_interactions2("X69", "X21")
int_X21_X3  <- generate_interactions2("X21", "X3")

interaction_list <- c(int_X66_X21, int_X69_X21, int_X21_X3)
length(interaction_list)   # should be 12
```

```
## [1] 12
```

```
# This is lm6, not lm7 to check whether higher interactions are better.

base_string <- paste(
  "y ~",
  "poly(X66,5,raw=TRUE) +",
  "poly(X69,3,raw=TRUE) +",
  "poly(X21,14,raw=TRUE) +",
  "poly(X3,3,raw=TRUE) +",
  "X19 + X111 + X59 "
)
```

```r
test_high_interactions <- function(base_string, interaction_list, train) {

  results <- data.frame()

  for (int in interaction_list) {

    form_string <- paste(base_string, "+", int)
    form <- as.formula(form_string)

    cv <- cv_rmse(form, train)$mean_RMSE

    results <- rbind(
      results,
      data.frame(
        interaction = int,
        RMSE = cv
      )
    )
  }

  return(results)
}


results_deg <- test_high_interactions(base_string, interaction_list, train)

# Sort by RMSE
sorted_interactions <- results_deg[order(results_deg$RMSE), ]


rmse_lm6 <- cv_rmse(lm6, train)$mean_RMSE

better <- sorted_interactions[ sorted_interactions$RMSE < rmse_lm6 , ]

cat("\n=== Interactions BETTER than lm6 ===\n")
```

```
##
## === Interactions BETTER than lm6 ===
```

```r
print(better)
```

```
##         interaction      RMSE
## 4  I(X66^2 * X21^2) 0.4626677
## 3  I(X66^2 * X21^1) 0.4691350
## 2  I(X66^1 * X21^2) 0.4708740
## 1  I(X66^1 * X21^1) 0.4737751
## 6  I(X69^1 * X21^2) 0.4819969
## 5  I(X69^1 * X21^1) 0.4850535
## 10  I(X21^1 * X3^2) 0.4902256
## 12  I(X21^2 * X3^2) 0.4910736
## 9   I(X21^1 * X3^1) 0.4928832
```

```r
better_terms <- c(
  "I(X66^2 * X21^2)",
```

```
  "I(X66^2 * X21^1)",
  "I(X66^1 * X21^2)",
  "I(X66^1 * X21^1)",
  "I(X69^1 * X21^2)",
  "I(X69^1 * X21^1)",
  "I(X21^1 * X3^2)",
  "I(X21^2 * X3^2)",
  "I(X21^1 * X3^1)"
)

# best subset 2^9=512 subsets, it's still doable

best_subset <- function(base_string, terms, train) {

  best_rmse <- Inf
  best_model <- NULL
  best_terms <- NULL

  k <- length(terms)

  for (size in 0:k) {

    if (size == 0) {
      subsets <- list(character(0))
    } else {
      subsets <- combn(terms, size, simplify = FALSE)
    }

    for (subset in subsets) {

      # Build formula
      if (length(subset) == 0) {
        form_string <- base_string
      } else {
        form_string <- paste(base_string, "+", paste(subset, collapse = " + "))
      }

      rmse <- cv_rmse(as.formula(form_string), train)$mean_RMSE

      if (rmse < best_rmse) {
        best_rmse <- rmse
        best_model <- form_string
        best_terms <- subset
        message("New best RMSE: ", rmse,
                " | terms: ", paste(subset, collapse=", "))
      }
    }
  }

  list(
    best_rmse = best_rmse,
    best_terms = best_terms,
    best_model = best_model
```

```
  )
}


result_bs <- best_subset(base_string, better_terms, train)

result_bs$best_rmse
```

```
## [1] 0.4513251
```

```
result_bs$best_terms
```

```
## [1] "I(X66^2 * X21^2)" "I(X69^1 * X21^2)" "I(X21^1 * X3^2)"
```

```
result_bs$best_model
```

```
## [1] "y ~ poly(X66,5,raw=TRUE) + poly(X69,3,raw=TRUE) + poly(X21,14,raw=TRUE) + poly(X3,3,raw=TRUE) +
```

```
lm8 = lm(y~poly(X66, 5, raw=TRUE)+
           poly(X69, 3, raw=TRUE)+
           poly(X21, 14, raw=TRUE)+
           poly(X3, 3, raw=TRUE)+
           +I(X66^2 * X21^2) + I(X69^1 * X21^2) + I(X21^1 * X3^2)+
           +X19 + X111+X59)
summary(lm8)
```

```
##
## Call:
## lm(formula = y ~ poly(X66, 5, raw = TRUE) + poly(X69, 3, raw = TRUE) +
##     poly(X21, 14, raw = TRUE) + poly(X3, 3, raw = TRUE) + +I(X66^2 *
##     X21^2) + I(X69^1 * X21^2) + I(X21^1 * X3^2) + +X19 + X111 +
##     X59)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.74511 -0.21525 -0.01107  0.21523  1.60252
##
## Coefficients: (1 not defined because of singularities)
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   5.927e-01  3.158e+00   0.188 0.851298
## poly(X66, 5, raw = TRUE)1    -1.897e-02  8.432e-02  -0.225 0.822163
## poly(X66, 5, raw = TRUE)2     4.233e-04  1.267e-03   0.334 0.738643
## poly(X66, 5, raw = TRUE)3     4.981e-05  5.061e-05   0.984 0.325937
## poly(X66, 5, raw = TRUE)4     5.116e-06  3.505e-06   1.460 0.145521
## poly(X66, 5, raw = TRUE)5    -1.877e-07  6.859e-08  -2.737 0.006608 **
## poly(X69, 3, raw = TRUE)1     1.445e-02  8.379e-02   0.172 0.863243
## poly(X69, 3, raw = TRUE)2    -1.461e-03  4.890e-04  -2.989 0.003060 **
## poly(X69, 3, raw = TRUE)3    -2.572e-05  1.612e-05  -1.596 0.111764
## poly(X21, 14, raw = TRUE)1    2.911e+00  1.563e+00   1.862 0.063708 .
## poly(X21, 14, raw = TRUE)2   -7.117e+00  3.115e+00  -2.284 0.023124 *
## poly(X21, 14, raw = TRUE)3    6.487e+00  2.518e+00   2.576 0.010522 *
## poly(X21, 14, raw = TRUE)4   -3.137e+00  1.102e+00  -2.848 0.004742 **
## poly(X21, 14, raw = TRUE)5    9.107e-01  2.954e-01   3.082 0.002266 **
## poly(X21, 14, raw = TRUE)6   -1.696e-01  5.189e-02  -3.268 0.001222 **
## poly(X21, 14, raw = TRUE)7    2.107e-02  6.182e-03   3.407 0.000756 ***
## poly(X21, 14, raw = TRUE)8   -1.777e-03  5.067e-04  -3.507 0.000531 ***
```

```
## poly(X21, 14, raw = TRUE)9   1.016e-04  2.842e-05   3.576 0.000414 ***
## poly(X21, 14, raw = TRUE)10 -3.833e-06  1.058e-06  -3.621 0.000350 ***
## poly(X21, 14, raw = TRUE)11  8.848e-08  2.424e-08   3.650 0.000315 ***
## poly(X21, 14, raw = TRUE)12 -1.004e-09  2.740e-10  -3.665 0.000297 ***
## poly(X21, 14, raw = TRUE)13         NA         NA      NA       NA
## poly(X21, 14, raw = TRUE)14  8.074e-14  2.199e-14   3.672 0.000290 ***
## poly(X3, 3, raw = TRUE)1    -1.144e-01  8.234e-02  -1.389 0.165911
## poly(X3, 3, raw = TRUE)2     2.077e-03  9.398e-04   2.210 0.027941 *
## poly(X3, 3, raw = TRUE)3    -1.155e-05  4.374e-06  -2.641 0.008752 **
## I(X66^2 * X21^2)             1.500e-06  1.248e-06   1.202 0.230290
## I(X69^1 * X21^2)             8.286e-05  3.476e-05   2.384 0.017816 *
## I(X21^1 * X3^2)             -1.178e-05  2.706e-06  -4.353 1.91e-05 ***
## X19                          8.307e-02  2.346e-02   3.541 0.000469 ***
## X111                         1.012e+00  1.688e-01   5.997 6.44e-09 ***
## X59                         -5.360e-02  3.746e-02  -1.431 0.153591
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4324 on 269 degrees of freedom
## Multiple R-squared:  0.8037, Adjusted R-squared:  0.7818
## F-statistic: 36.72 on 30 and 269 DF,  p-value: < 2.2e-16
```

```
cv_rmse(lm8,train)
```

```
## $mean_RMSE
## [1] 0.4513251
##
## $sd_RMSE
## [1] 0.0767361
##
## $all_RMSE
##  [1] 0.4425578 0.5221690 0.5043998 0.4000440 0.4901194 0.3625572 0.3188291
##  [8] 0.5083940 0.4101056 0.5540755
```

Now our model is quite complex. Let's see if removing any of the term could reduce cv rmse.

```
blocks <- c(
  "poly(X66,5,raw=TRUE)",
  "poly(X69,3,raw=TRUE)",
  "poly(X21,14,raw=TRUE)",
  "poly(X3,3,raw=TRUE)",
  "I(X66^2 * X21^2)",
  "I(X69^1 * X21^2)",
  "I(X21^1 * X3^2)",
  "X19",
  "X111",
  "X59"
)

best_subset_blocks <- function(block_terms, train) {

  best_rmse  <- Inf
  best_model <- NULL
  best_terms <- NULL

  k <- length(block_terms)
```

```
  for (size in 1:k) {# or 0:k if you want intercept-only too
    subsets <- combn(block_terms, size, simplify = FALSE)

    for (subset in subsets) {
      rhs <- paste(subset, collapse = " + ")
      form_string <- paste("y ~", rhs)

      rmse <- cv_rmse(as.formula(form_string), train)$mean_RMSE

      if (rmse < best_rmse) {
        best_rmse  <- rmse
        best_model <- form_string
        best_terms <- subset
        message("New best RMSE: ", rmse,
                " | terms: ", paste(subset, collapse = ", "))
      }
    }
  }

  list(
    best_rmse  = best_rmse,
    best_terms = best_terms,
    best_model = best_model
  )
}

result_blocks <- best_subset_blocks(blocks, train)
result_blocks$best_rmse
```

```
## [1] 0.4496267
```

```
result_blocks$best_terms
```

```
## [1] "poly(X66,5,raw=TRUE)"  "poly(X69,3,raw=TRUE)"  "poly(X21,14,raw=TRUE)"
## [4] "poly(X3,3,raw=TRUE)"   "I(X66^2 * X21^2)"      "I(X69^1 * X21^2)"
## [7] "I(X21^1 * X3^2)"       "X19"                   "X111"
```

```
result_blocks$best_model
```

```
## [1] "y ~ poly(X66,5,raw=TRUE) + poly(X69,3,raw=TRUE) + poly(X21,14,raw=TRUE) + poly(X3,3,raw=TRUE) +
```

```
lm9 = lm(y ~ poly(X66,5,raw=TRUE) +
           poly(X69,3,raw=TRUE) +
           poly(X21,14,raw=TRUE) +
           poly(X3,3,raw=TRUE) +
           I(X66^2 * X21^2) +
           I(X69^1 * X21^2) +
           I(X21^1 * X3^2)+
           X19 + X111)
summary(lm9)
```

```
##
## Call:
## lm(formula = y ~ poly(X66, 5, raw = TRUE) + poly(X69, 3, raw = TRUE) +
##     poly(X21, 14, raw = TRUE) + poly(X3, 3, raw = TRUE) + I(X66^2 *
##     X21^2) + I(X69^1 * X21^2) + I(X21^1 * X3^2) + X19 + X111)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.74289 -0.21825 -0.00186  0.21783  1.59882
##
## Coefficients: (1 not defined because of singularities)
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  5.605e-01  3.165e+00   0.177 0.859558
## poly(X66, 5, raw = TRUE)1   -1.892e-02  8.448e-02  -0.224 0.823001
## poly(X66, 5, raw = TRUE)2    5.020e-04  1.269e-03   0.396 0.692613
## poly(X66, 5, raw = TRUE)3    5.051e-05  5.070e-05   0.996 0.320090
## poly(X66, 5, raw = TRUE)4    4.795e-06  3.504e-06   1.368 0.172375
## poly(X66, 5, raw = TRUE)5   -1.813e-07  6.857e-08  -2.643 0.008686 **
## poly(X69, 3, raw = TRUE)1    1.292e-02  8.394e-02   0.154 0.877749
## poly(X69, 3, raw = TRUE)2   -1.418e-03  4.889e-04  -2.899 0.004048 **
## poly(X69, 3, raw = TRUE)3   -2.432e-05  1.612e-05  -1.509 0.132549
## poly(X21, 14, raw = TRUE)1   2.842e+00  1.566e+00   1.815 0.070585 .
## poly(X21, 14, raw = TRUE)2  -6.912e+00  3.118e+00  -2.217 0.027474 *
## poly(X21, 14, raw = TRUE)3   6.286e+00  2.519e+00   2.495 0.013180 *
## poly(X21, 14, raw = TRUE)4  -3.035e+00  1.101e+00  -2.756 0.006252 **
## poly(X21, 14, raw = TRUE)5   8.801e-01  2.952e-01   2.981 0.003135 **
## poly(X21, 14, raw = TRUE)6  -1.637e-01  5.183e-02  -3.159 0.001765 **
## poly(X21, 14, raw = TRUE)7   2.030e-02  6.171e-03   3.290 0.001134 **
## poly(X21, 14, raw = TRUE)8  -1.710e-03  5.055e-04  -3.383 0.000823 ***
## poly(X21, 14, raw = TRUE)9   9.765e-05  2.834e-05   3.446 0.000660 ***
## poly(X21, 14, raw = TRUE)10 -3.677e-06  1.055e-06  -3.486 0.000572 ***
## poly(X21, 14, raw = TRUE)11  8.476e-08  2.415e-08   3.510 0.000526 ***
## poly(X21, 14, raw = TRUE)12 -9.606e-10  2.728e-10  -3.521 0.000505 ***
## poly(X21, 14, raw = TRUE)13         NA         NA      NA       NA
## poly(X21, 14, raw = TRUE)14  7.699e-14  2.187e-14   3.520 0.000506 ***
## poly(X3, 3, raw = TRUE)1    -1.176e-01  8.247e-02  -1.426 0.155079
## poly(X3, 3, raw = TRUE)2     2.172e-03  9.393e-04   2.312 0.021507 *
## poly(X3, 3, raw = TRUE)3    -1.219e-05  4.360e-06  -2.795 0.005559 **
## I(X66^2 * X21^2)             1.381e-06  1.247e-06   1.107 0.269180
## I(X69^1 * X21^2)             9.092e-05  3.436e-05   2.646 0.008626 **
## I(X21^1 * X3^2)             -1.297e-05  2.579e-06  -5.029 8.99e-07 ***
## X19                          8.607e-02  2.341e-02   3.677 0.000284 ***
## X111                         1.016e+00  1.691e-01   6.008 6.07e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4332 on 270 degrees of freedom
## Multiple R-squared:  0.8022, Adjusted R-squared:  0.781
## F-statistic: 37.77 on 29 and 270 DF,  p-value: < 2.2e-16
```

```
cv_rmse(lm9,train)
```

```
## $mean_RMSE
## [1] 0.4496267
##
## $sd_RMSE
## [1] 0.07273673
##
## $all_RMSE
##  [1] 0.4415505 0.4814945 0.5105339 0.4084901 0.4914399 0.3642895 0.3333033
```

```
##  [8] 0.5133146 0.3948952 0.5569552
```

```
# Now we have finished everything with lm. Let's try it on Kaggle.

test <- read.csv("test_predictors.csv")

pred <- predict(lm9, newdata = test)
sample <- read.csv("SampleSubmission.csv")

submission <- data.frame(
  id = sample$id,
  y  = pred
)

write.csv(submission, "LM9_Submission.csv", row.names=FALSE)

cat("Saved: LM9_Submission.csv\n")
```

```
## Saved: LM9_Submission.csv
```

Score on Kaggle: Private Score: 0.52147 Public Score: 0.52681

Now let's use GAM to get lower RMSE

s: smoothing splines ti: smooth interactions

```
library(mgcv)

# This is similar with LM9,
# but we change the one variable polynomials to smoothing splines
gam1_formula <- y ~
  s(X66, k = 12) +
  s(X69, k = 12) +
  s(X21, k = 18) +
  s(X3,  k = 10) +
  I(X66^2 * X21^2) +
  I(X69^1 * X21^2) +
  I(X21^1 * X3^2)+
  X19 + X111

# Fit GAM1
gam1 <- gam(gam1_formula, data=train, method="REML", select=TRUE)

summary(gam1)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ s(X66, k = 12) + s(X69, k = 12) + s(X21, k = 18) + s(X3,
##     k = 10) + I(X66^2 * X21^2) + I(X69^1 * X21^2) + I(X21^1 *
##     X3^2) + X19 + X111
##
## Parametric coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -7.287e-01  2.235e-01  -3.261 0.001254 **
```

```
## I(X66^2 * X21^2)  1.124e-06  1.156e-06   0.972 0.331684
## I(X69^1 * X21^2)  9.951e-05  3.153e-05   3.156 0.001783 **
## I(X21^1 * X3^2)  -1.369e-05  2.428e-06  -5.636 4.38e-08 ***
## X19               8.606e-02  2.313e-02   3.720 0.000242 ***
## X111              9.929e-01  1.649e-01   6.020 5.70e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##           edf Ref.df      F  p-value
## s(X66)  6.683     11  5.353  < 2e-16 ***
## s(X69)  3.583     11  2.948  < 2e-16 ***
## s(X21) 11.606     17 12.443  < 2e-16 ***
## s(X3)   3.049      9  1.694 0.000354 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.795   Deviance explained = 81.5%
## -REML = 247.57  Scale est. = 0.17576   n = 300
```

```r
# Create cv_gam function
cv_gam <- function(formula, data, K = 10, seed = 1) {
  set.seed(seed)

  n <- nrow(data)
  folds <- sample(rep(1:K, length.out = n))

  rmse <- numeric(K)

  for (k in 1:K) {
    train_fold <- data[folds != k, ]
    test_fold  <- data[folds == k, ]

    # Fit GAM
    fit <- gam(formula, data = train_fold, method = "REML")

    # Predict
    pred <- predict(fit, newdata = test_fold)

    # RMSE
    rmse[k] <- sqrt(mean((pred - test_fold$y)^2))
  }

  return(list(
    mean_RMSE = mean(rmse),
    sd_RMSE   = sd(rmse),
    all_RMSE  = rmse
  ))
}
cv_gam1 = cv_gam(gam1_formula,train)
cv_gam1
```

```
## $mean_RMSE
## [1] 0.4449057
##
```

```
## $sd_RMSE
## [1] 0.07555486
##
## $all_RMSE
##  [1] 0.4209355 0.4885687 0.5117365 0.4091546 0.5194580 0.3587830 0.3304465
##  [8] 0.4874085 0.3754565 0.5471092
```

Then, we change the interactions to smooth interactions.

```
gam2_formula <- y ~
  s(X66, k = 12) +
  s(X69, k = 12) +
  s(X21, k = 18) +
  s(X3,  k = 10) +
  ti(X66,X21, k=8) +
  ti(X69,X21, k=8) +
  ti(X21,X3, k=8)+
  X19 + X111

gam2 <- gam(gam2_formula, data=train, method="REML", select=TRUE)

summary(gam2)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ s(X66, k = 12) + s(X69, k = 12) + s(X21, k = 18) + s(X3,
##     k = 10) + ti(X66, X21, k = 8) + ti(X69, X21, k = 8) + ti(X21,
##     X3, k = 8) + X19 + X111
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.23535    0.16809  -7.349 2.62e-12 ***
## X19          0.08444    0.02219   3.805 0.000177 ***
## X111         1.01308    0.15593   6.497 4.19e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                edf Ref.df      F p-value
## s(X66)       6.3779     11  5.901  <2e-16 ***
## s(X69)       3.6854     11  2.475  <2e-16 ***
## s(X21)      11.7302     17 16.388  <2e-16 ***
## s(X3)        3.2428      9  3.696  <2e-16 ***
## ti(X66,X21)  5.6233     49  1.504  <2e-16 ***
## ti(X69,X21)  0.9208     49  0.022   0.216
## ti(X21,X3)   7.1478     49  0.847  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.818   Deviance explained = 84.3%
## -REML = 213.17  Scale est. = 0.15568   n = 300
```

```
cv_gam2 = cv_gam(gam2_formula,train)
cv_gam2
```

```
## $mean_RMSE
## [1] 0.4325536
##
## $sd_RMSE
## [1] 0.07460207
##
## $all_RMSE
##  [1] 0.4027576 0.4780151 0.4697028 0.4321348 0.4904316 0.3457749 0.2967451
##  [8] 0.5149694 0.3797129 0.5152916
```

```
cv_gam2$mean_RMSE < cv_gam1$mean_RMSE
```

```
## [1] TRUE
```

From the summary function, the ti(X69,X21) p-value is quite suspicious. Let's try removing it and check the RMSE.

```
gam3_formula <- y ~
  s(X66, k = 12) +
  s(X69, k = 12) +
  s(X21, k = 18) +
  s(X3,  k = 10) +
  ti(X66, X21, k=8) +
  ti(X21, X3,  k=8) +
  X111 + X19

gam3 <- gam(gam3_formula, data=train, method="REML", select=TRUE)

summary(gam3)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ s(X66, k = 12) + s(X69, k = 12) + s(X21, k = 18) + s(X3,
##     k = 10) + ti(X66, X21, k = 8) + ti(X21, X3, k = 8) + X111 +
##     X19
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.23700    0.16826  -7.352 2.57e-12 ***
## X111         1.01484    0.15607   6.502 4.06e-10 ***
## X19          0.08448    0.02222   3.802 0.000179 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                edf Ref.df      F p-value
## s(X66)       6.396     11  6.064  <2e-16 ***
## s(X69)       3.692     11  2.534  <2e-16 ***
## s(X21)      11.725     17 16.317  <2e-16 ***
## s(X3)        3.225      9  3.640  <2e-16 ***
```

```
## ti(X66,X21)  6.085     49  1.839  <2e-16 ***
## ti(X21,X3)   7.297     49  0.860  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.818   Deviance explained = 84.3%
## -REML =  213.2  Scale est. = 0.15601   n = 300
```

```
cv_gam3 = cv_gam(gam3_formula,train)
cv_gam3
```

```
## $mean_RMSE
## [1] 0.4308959
##
## $sd_RMSE
## [1] 0.07402865
##
## $all_RMSE
##  [1] 0.4033649 0.4767075 0.4680794 0.4332528 0.4886535 0.3405254 0.2961775
##  [8] 0.5076227 0.3802012 0.5143739
```

```
cv_gam3$mean_RMSE < cv_gam2$mean_RMSE
```

```
## [1] TRUE
```

X19 is a discrete variable, treating it as a categorical variable may improve the GAM's performance.

```
train$X19_f <- factor(train$X19)
test$X19_f  <- factor(test$X19, levels = levels(train$X19_f))

gam4_formula <- y ~
  s(X66, k = 12) +
  s(X69, k = 12) +
  s(X21, k = 18) +
  s(X3,  k = 10) +

  ti(X66, X21, k=8) +
  ti(X21, X3,  k=8) +

  X111 + X19_f

gam4 <- gam(gam4_formula, data=train, method="REML", select=TRUE)

summary(gam4)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ s(X66, k = 12) + s(X69, k = 12) + s(X21, k = 18) + s(X3,
##     k = 10) + ti(X66, X21, k = 8) + ti(X21, X3, k = 8) + X111 +
##     X19_f
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.25771    0.17926  -7.016 2.03e-11 ***
```

```
## X111         1.00641    0.15544   6.474 4.82e-10 ***
## X19_f1       0.03075    0.12121   0.254  0.79993
## X19_f3       0.43893    0.15439   2.843  0.00483 **
## X19_f4       0.29296    0.12571   2.330  0.02057 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                edf Ref.df      F p-value
## s(X66)       6.321     11  5.790  <2e-16 ***
## s(X69)       3.684     11  2.711  <2e-16 ***
## s(X21)      11.709     17 16.374  <2e-16 ***
## s(X3)        3.391      9  3.708  <2e-16 ***
## ti(X66,X21)  5.869     49  1.874  <2e-16 ***
## ti(X21,X3)   7.819     49  0.931  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.821   Deviance explained = 84.7%
## -REML = 211.63  Scale est. = 0.15312   n = 300
```

```
cv_gam4 = cv_gam(gam4_formula,train)
cv_gam4
```

```
## $mean_RMSE
## [1] 0.4270584
##
## $sd_RMSE
## [1] 0.07746553
##
## $all_RMSE
##  [1] 0.4060467 0.4646716 0.4546878 0.4237394 0.5037809 0.3228253 0.2858733
##  [8] 0.4970759 0.3922138 0.5196695
```

```
cv_gam4$mean_RMSE < cv_gam3$mean_RMSE
```

```
## [1] TRUE
```

```
pred <- predict(gam4, newdata = test)
sample <- read.csv("SampleSubmission.csv")

submission <- data.frame(
  id = sample$id,
  y  = pred
)

write.csv(submission, "GAM4_Submission.csv", row.names=FALSE)

cat("Saved: GAM4_Submission.csv\n")
```

```
## Saved: GAM4_Submission.csv
```

Score on Kaggle: Private Score: 0.48633 Public Score: 0.49299