SMC Final Project

Varshini Yanamandra

2023-04-16

```
# required libraries
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-6
```

```
library(readxl)
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
##
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
# reading the data from an excel file
suppressWarnings({
  horses <- read_excel('HorseFavoriteDataset.xlsx')
})

head(horses)
```

```
## # A tibble: 6 x 23
##     won horse_age horse~1 actua~2 post_~3 Post_2 win_o~4 train~5 jocke~6 surface
##   <dbl>     <dbl>   <dbl>   <dbl>   <dbl>  <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1     0         3      30     123       2     25     2.9      97      64       0
```

```
## 2     1         4      34     127       7      0     3.4     164     140       0
## 3     0         3      39     132       3     16     3.2      80      64       1
## 4     1         5      28     120       5      4     3.3      54      63       0
## 5     1         4      27     120       5      4     3.6      97      64       0
## 6     0         4      32     125       1     36     2.6      97      64       0
## # ... with 13 more variables: distance <dbl>, going_factor <dbl>, prize <dbl>,
## #   race_class <dbl>, mean_difference <dbl>, is_weekday <dbl>,
## #   pastpPerformance <dbl>, Blin_Vis <dbl>, TongueTie <dbl>,
## #   horse_sex_value <dbl>, freshness <dbl>, jockey_change <dbl>,
## #   trainer_change <dbl>, and abbreviated variable names 1: horse_rating,
## #   2: actual_weight, 3: post_pos, 4: win_odds, 5: trainer_id, 6: jockey_id
```

```r
# creating a dataframe to hold all the accuracies
res = tibble(model = c("Ridge Regression", "LASSO Regression", "XGBoost"), accuracy = rep(0, 3))
```

```r
## ridge and lasso regression
set.seed(123)

# checking for null values
sum(is.na(horses)) # 0
```

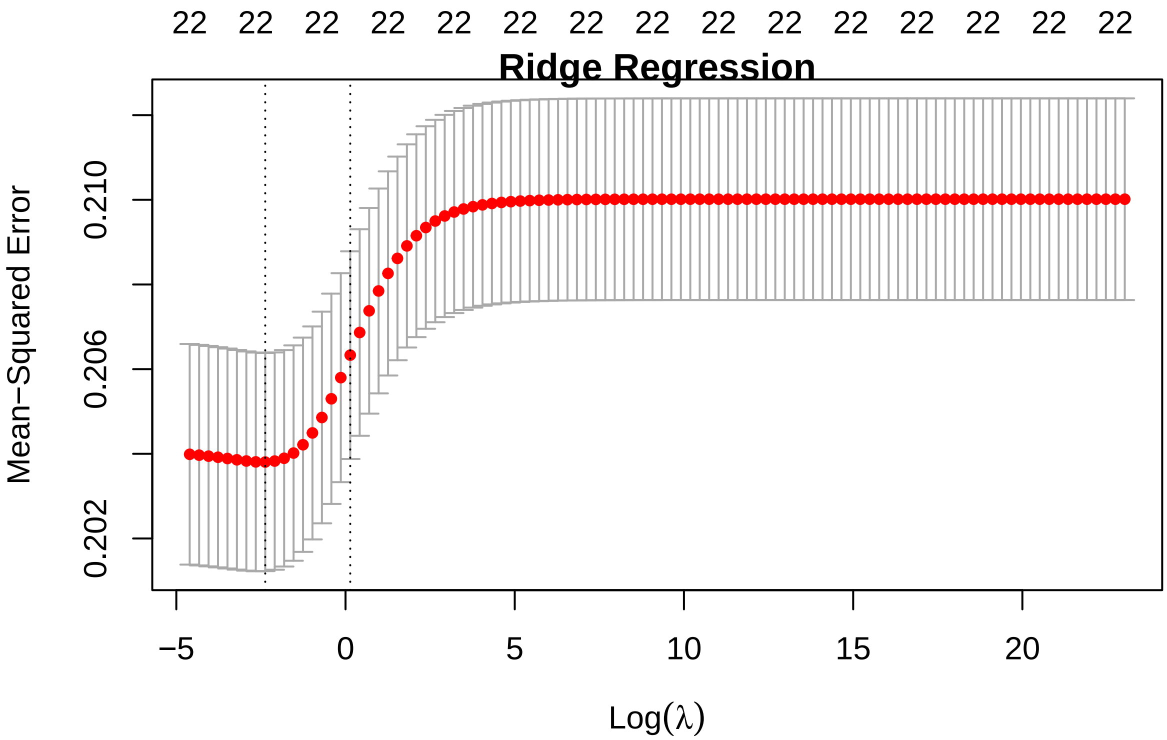
```
## [1] 0
```

```r
# splitting the data into training (80%) and test(20%) datasets
train = sample(nrow(horses), nrow(horses) * 0.8) # training indices
length(train)/nrow(horses) # checking the split percentage
```

```
## [1] 0.7999658
```

```r
# creating model matrices
x_original = model.matrix(won ~ ., horses)[, -1]
# scaling the data
x = scale(x_original, center = TRUE, scale = TRUE)
y.tr = horses[train, ]$won
y.test = horses[-train, ]$won
```

```r
## ridge regression

# training and cross-validation
grid=10^seq(10, -2, length=100) # grid for lambdas
horses.ridge <- cv.glmnet(x[train, ], y.tr, alpha = 0, lambda = grid)
# prediction using the best lambda
horses.ridge.preds <- predict(horses.ridge, s = horses.ridge$lambda.min, newx = x[-train, ])
horses.ridge.preds <- ifelse(horses.ridge.preds < 0.5, 0, 1)
res[1, 2] = mean(horses.ridge.preds == y.test) # accuracy
# plot to see how MSE varies with lambda
plot(horses.ridge)
title("Ridge Regression", line = -0.01)
```




```
print(horses.ridge)
```

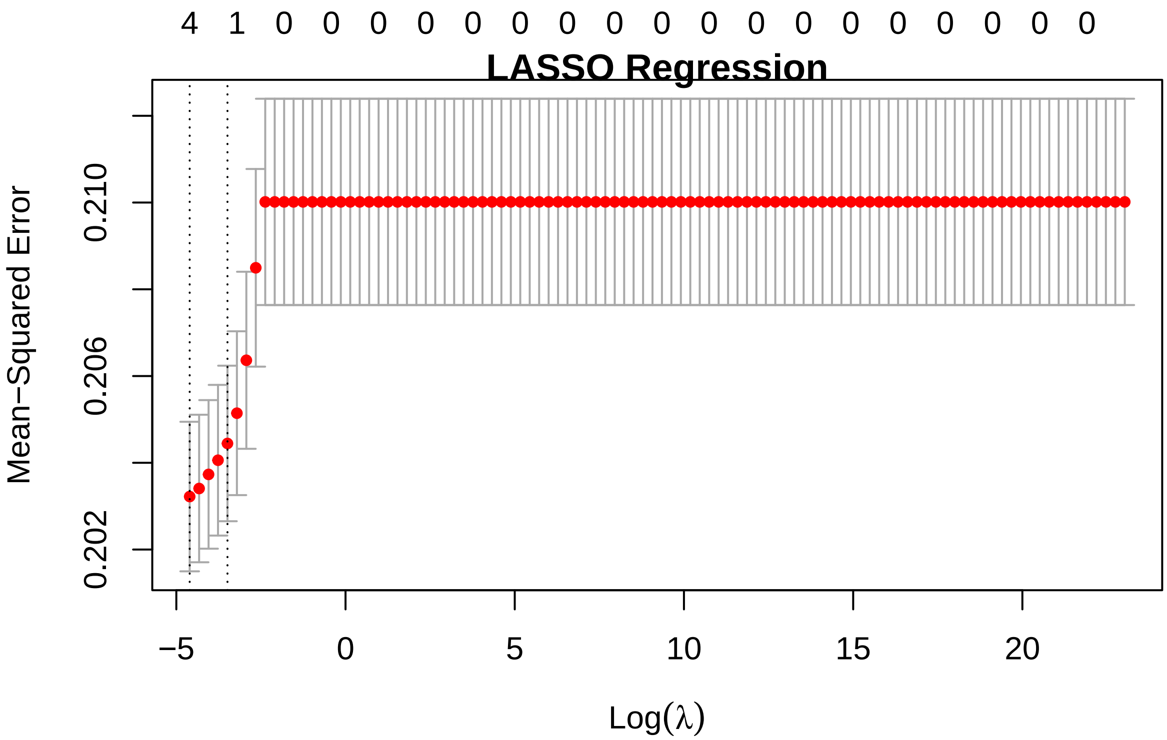
```
##
## Call:  cv.glmnet(x = x[train, ], y = y.tr, lambda = grid, alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure       SE Nonzero
## min 0.0933    92  0.2038 0.002576      22
## 1se 1.1498    83  0.2063 0.002453      22
```

```
coef(horses.ridge)
```

```
## 23 x 1 sparse Matrix of class "dgCMatrix"
##                             s1
## (Intercept)       0.2997433611
## horse_age         0.0016170455
## horse_rating     -0.0002055596
## actual_weight    -0.0009644911
## post_pos         -0.0019653521
## Post_2            0.0063159612
## win_odds         -0.0223811615
## trainer_id        0.0016758022
## jockey_id        -0.0011440643
## surface          -0.0028585944
## distance         -0.0047704645
## going_factor     -0.0006950653
## prize             0.0017454141
## race_class       -0.0013600486
## mean_difference  -0.0004293012
## is_weekday        0.0029436820
```

```
## pastpPerformance -0.0033683484
## Blin_Vis           0.0051352775
## TongueTie          0.0026773776
## horse_sex_value    0.0020001231
## freshness          0.0029697532
## jockey_change     -0.0020298934
## trainer_change     0.0044991800
```

```r
## lasso regression

# training and cross-validation
horses.lasso <- cv.glmnet(x[train, ], y.tr, alpha = 1, lambda = grid)
# prediction using the best lambda
horses.lasso.preds <- predict(horses.lasso, s = horses.lasso$lambda.min, newx = x[-train, ])
horses.lasso.preds <- ifelse(horses.lasso.preds < 0.5, 0, 1)
res[2, 2] = mean((horses.lasso.preds == y.test)) # accuracy
# plot to see how MSE varies with lambda
plot(horses.lasso)
title("LASSO Regression", line = -0.01)
```




```r
print(horses.lasso)
```

```
##
## Call:  cv.glmnet(x = x[train, ], y = y.tr, lambda = grid, alpha = 1)
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure       SE Nonzero
## min 0.01000   100  0.2032 0.001724       4
## 1se 0.03054    96  0.2044 0.001793       1
```

```r
coef(horses.lasso)
```

```
## 23 x 1 sparse Matrix of class "dgCMatrix"
##                           s1
## (Intercept)       0.29958065
## horse_age         .
## horse_rating      .
## actual_weight     .
## post_pos          .
## Post_2            .
## win_odds         -0.05126402
## trainer_id        .
## jockey_id         .
## surface           .
## distance          .
## going_factor      .
## prize             .
## race_class        .
## mean_difference   .
## is_weekday        .
## pastpPerformance  .
## Blin_Vis          .
## TongueTie         .
## horse_sex_value   .
## freshness         .
## jockey_change     .
## trainer_change    .
```

```r
## XGBoost
set.seed(123)

#define final training and testing sets
xgb_train <- xgb.DMatrix(data = x[train, ], label = y.tr)
xgb_test <- xgb.DMatrix(data = x[-train, ], label = y.test)

model <- xgboost(data = xgb_train, nround = 100, objective = "binary:logistic")
```

```
## [1]  train-logloss:0.631344
## [2]  train-logloss:0.595148
## [3]  train-logloss:0.570639
## [4]  train-logloss:0.555085
## [5]  train-logloss:0.545392
## [6]  train-logloss:0.531844
## [7]  train-logloss:0.525002
## [8]  train-logloss:0.514829
## [9]  train-logloss:0.507893
## [10] train-logloss:0.505329
## [11] train-logloss:0.500699
## [12] train-logloss:0.495764
## [13] train-logloss:0.492252
## [14] train-logloss:0.490635
## [15] train-logloss:0.485136
## [16] train-logloss:0.480834
## [17] train-logloss:0.477680
## [18] train-logloss:0.473131
```

```
## [19] train-logloss:0.467431
## [20] train-logloss:0.459386
## [21] train-logloss:0.454286
## [22] train-logloss:0.444884
## [23] train-logloss:0.442151
## [24] train-logloss:0.439636
## [25] train-logloss:0.434470
## [26] train-logloss:0.430308
## [27] train-logloss:0.427730
## [28] train-logloss:0.426717
## [29] train-logloss:0.426281
## [30] train-logloss:0.418761
## [31] train-logloss:0.411668
## [32] train-logloss:0.408828
## [33] train-logloss:0.405655
## [34] train-logloss:0.404849
## [35] train-logloss:0.400474
## [36] train-logloss:0.396764
## [37] train-logloss:0.391502
## [38] train-logloss:0.388363
## [39] train-logloss:0.385416
## [40] train-logloss:0.381053
## [41] train-logloss:0.377661
## [42] train-logloss:0.374081
## [43] train-logloss:0.367235
## [44] train-logloss:0.365222
## [45] train-logloss:0.363409
## [46] train-logloss:0.361518
## [47] train-logloss:0.360594
## [48] train-logloss:0.359192
## [49] train-logloss:0.353494
## [50] train-logloss:0.346210
## [51] train-logloss:0.344328
## [52] train-logloss:0.343067
## [53] train-logloss:0.340231
## [54] train-logloss:0.336309
## [55] train-logloss:0.333295
## [56] train-logloss:0.331866
## [57] train-logloss:0.326696
## [58] train-logloss:0.321346
## [59] train-logloss:0.320521
## [60] train-logloss:0.317002
## [61] train-logloss:0.314177
## [62] train-logloss:0.311942
## [63] train-logloss:0.309631
## [64] train-logloss:0.307717
## [65] train-logloss:0.305293
## [66] train-logloss:0.302340
## [67] train-logloss:0.301323
## [68] train-logloss:0.299223
## [69] train-logloss:0.296796
## [70] train-logloss:0.295668
## [71] train-logloss:0.293008
## [72] train-logloss:0.292782
```

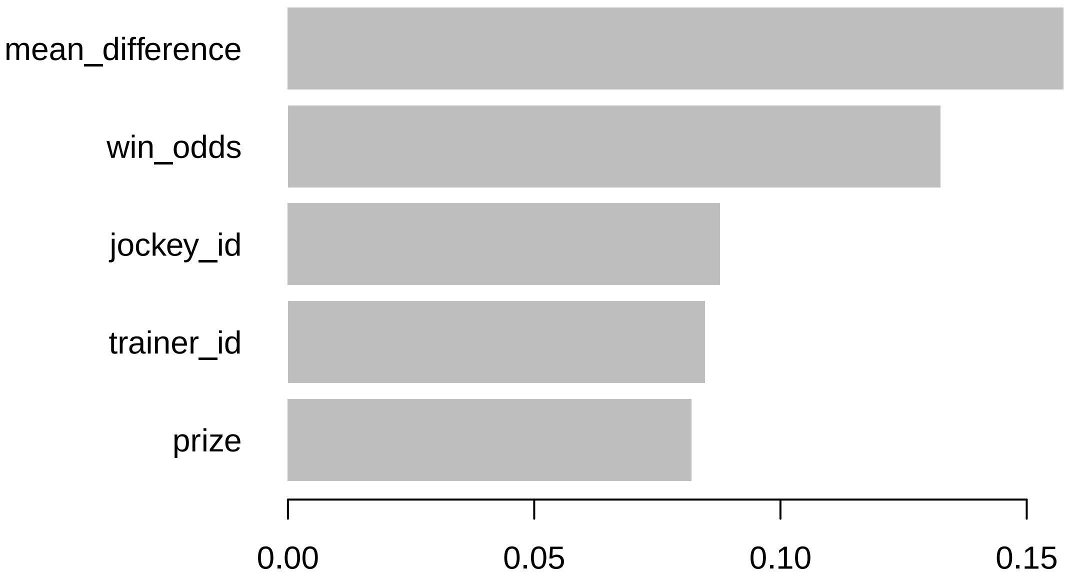
```
## [73] train-logloss:0.291425
## [74] train-logloss:0.290765
## [75] train-logloss:0.290595
## [76] train-logloss:0.287485
## [77] train-logloss:0.283348
## [78] train-logloss:0.278532
## [79] train-logloss:0.276181
## [80] train-logloss:0.271967
## [81] train-logloss:0.268222
## [82] train-logloss:0.265580
## [83] train-logloss:0.263361
## [84] train-logloss:0.260210
## [85] train-logloss:0.257808
## [86] train-logloss:0.255122
## [87] train-logloss:0.251632
## [88] train-logloss:0.249928
## [89] train-logloss:0.247716
## [90] train-logloss:0.246090
## [91] train-logloss:0.244505
## [92] train-logloss:0.244081
## [93] train-logloss:0.243433
## [94] train-logloss:0.242313
## [95] train-logloss:0.241876
## [96] train-logloss:0.240994
## [97] train-logloss:0.240734
## [98] train-logloss:0.240121
## [99] train-logloss:0.239202
## [100]    train-logloss:0.236442
```

```r
# generate predictions for our held-out testing data
pred <- predict(model, xgb_test)

# get the accuracy
res[3, 2] <- mean(as.numeric(pred > 0.5) == y.test)

# variable importance matrix
importance_matrix = xgb.importance(colnames(xgb_train), model = model)
importance_matrix
```

```
##              Feature        Gain       Cover   Frequency
##  1:  mean_difference 0.157477613 0.248019087 0.161586268
##  2:         win_odds 0.132428188 0.127132998 0.106836342
##  3:        jockey_id 0.087715877 0.050625409 0.084344481
##  4:       trainer_id 0.084656676 0.048439001 0.085232317
##  5:            prize 0.081940393 0.077578644 0.081976916
##  6:         post_pos 0.064996539 0.033873701 0.070730985
##  7:     horse_rating 0.064015726 0.158102966 0.079609352
##  8:    actual_weight 0.056680227 0.051019612 0.059189109
##  9: pastpPerformance 0.053529541 0.035297172 0.057709381
## 10:         distance 0.043672850 0.038215383 0.043503995
## 11:           Post_2 0.033351826 0.021681798 0.033737792
## 12:     going_factor 0.026195687 0.017833785 0.021899970
## 13:        freshness 0.024149224 0.017994897 0.023675644
## 14:        horse_age 0.017187743 0.016402474 0.017164842
## 15:    jockey_change 0.016224102 0.003066405 0.018644569
```

```
## 16:        race_class 0.015459006 0.023126889 0.014797277
## 17:        is_weekday 0.009800803 0.002861109 0.011245931
## 18:         TongueTie 0.007938965 0.004264650 0.007694584
## 19:           surface 0.007912235 0.003860537 0.007694584
## 20:          Blin_Vis 0.007220899 0.006381745 0.006214856
## 21:   horse_sex_value 0.004085396 0.005387437 0.003551347
## 22:    trainer_change 0.003360483 0.008834299 0.002959455
##               Feature        Gain       Cover   Frequency
```

```
# plot variable importance
xgb.plot.importance(importance_matrix[1:5,])
```




```
res
```

```
## # A tibble: 3 x 2
##   model            accuracy
##   <chr>               <dbl>
## 1 Ridge Regression    0.707
## 2 LASSO Regression    0.707
## 3 XGBoost             0.674
```