Regularization in Linear Regression

Applied Machine Learning for Educational Data Science

true

10/20/2021

Contents

Regularization **1**

Ridge Regression 1

Ridge Penalty 1

Model Estimation 3

Using Ridge Regression to Predict Readability Scores 10

Lasso Regression 20

Lasso Penalty 20

Elastic Net 20

[Updated: Fri, Oct 22, 2021 - 17:00:44]

Regularization

Regularization is a general strategy to incorporate additional penalty terms into the model fitting process and used not just for regression but a variety of other types of models. The idea behind the regularization is to constrain the size of regression coefficients with the purpose of reducing their sampling variation and, hence, reducing the variance of model predictions. These constrains are typically incorporated into the loss function to be optimized. There are two commonly used regularization strategy: **ridge penalty** and **lasso penalty**. In addition, there is also **elastic net**, a mixture of these two strategies.

Ridge Regression

Ridge Penalty

Remember that we formulated the loss function for the linear regression as the sum of squared residuals across all observations. For ridge regression, we add a penalty term to this loss function and this penalty term is a function of all the regression coefficients in the model. Assuming that there are P regression coefficients in the model, the penalty term for the ridge regression would be

$$\lambda \sum_{i=1}^{P} \beta_p^2,$$

where λ is a parameter that penalizes the regression coefficients when they get larger. Therefore, when we fit a regression model with ridge penalty, the loss function to minimize becomes

$$Loss = \sum_{i=1}^{N} \epsilon_{(i)}^2 + \lambda \sum_{i=1}^{P} \beta_p^2,$$

$$Loss = SSR + \lambda \sum_{i=1}^{P} \beta_p^2.$$

Let's consider the same example from the previous class. Suppose we fit a simple linear regression model such that the readability score is the outcome (Y) and average word length is the predictor(X). Our regression model is

$$Y = \beta_0 + \beta_1 X + \epsilon,$$

and let's assume the set of coefficients are $\{\beta_0, \beta_1\}$ = {7.5,-2}, so my model is

$$Y = 7.5 - 2X + \epsilon.$$

Then, the value of the loss function when $\lambda = 0.2$ would be equal to 27.433.

```
readability_sub <- read.csv('https://raw.githubusercontent.com/uo-datasci-specialization/c4-ml-fall-202

d <-  readability_sub[,c('mean.wl','target')]

b0 = 7.5
b1 = -2

d$predicted <- b0 + b1*d$mean.wl
d$error     <- d$target - d$predicted

d
```

```
    mean.wl      target  predicted       error
1  4.603659 -2.58590836 -1.7073171 -0.87859129
2  3.830688  0.45993224 -0.1613757  0.62130790
3  4.180851 -1.07470758 -0.8617021 -0.21300545
4  4.015544 -1.81700402 -0.5310881 -1.28591594
5  4.686047 -1.81491744 -1.8720930  0.05717559
6  4.211340 -0.94968236 -0.9226804 -0.02700194
7  4.025000 -0.12103065 -0.5500000  0.42896935
8  4.443182 -2.82200582 -1.3863636 -1.43564218
9  4.089385 -0.74845172 -0.6787709 -0.06968077
10 4.156757  0.73948755 -0.8135135  1.55300107
11 4.463277 -0.96218937 -1.4265537  0.46436430
12 5.478261 -2.21514888 -3.4565217  1.24137286
13 4.770492 -1.21845136 -2.0409836  0.82253224
14 4.568966 -1.89544351 -1.6379310 -0.25751247
15 4.735751 -0.04101056 -1.9715026  1.93049203
16 4.372340 -1.83716516 -1.2446809 -0.59248431
17 4.103448 -0.18818586 -0.7068966  0.51871069
18 4.042857 -0.81739314 -0.5857143 -0.23167886
19 4.202703 -1.86307557 -0.9054054 -0.95767016
20 3.853535 -0.41630158 -0.2070707 -0.20923088
```

```r
lambda = 0.2

loss <- sum((d$error)^2) + lambda*(b0^2 + b1^2)

loss
```

```
[1] 27.43364
```

Notice that when λ is equal to 0, the loss function is identical to SSR; therefore, it becomes a linear regression with no regularization. As the value of λ increases, the degree of penalty linearly increases. Technically, the λ can take any positive value between 0 and ∞.

As we did in the previous lecture, imagine that we computed the loss function with the ridge penalty term for every possible combination of the intercept (β_0) and the slope (β_1). Let's say the plausible range for the intercept is from -10 to 10 and the plausible range for the slope is from -2 to 2. Now, we also have to think different values of λ because the surface we try to minimize is dependent on the value λ and different values of λ yield different estimates of β_0 and and β_1.

You can try a number of different values for λ using the shiny app at this link and explore how the loss function value and coefficient estimates change for different values of λ. Note that when λ is equal to zero, it should be equivalent of what we have seen in the earlier lecture. Try values of 1, 5, 10, 50, and 100.

Below is also a demonstration of what happens to loss function and the regression coefficients for increasing levels of ridge penalty (λ).

Model Estimation

Matrix Solution The matrix solution we learned before for regression without regularization can also be applied to estimate the coeffients from ridge regression given the λ value. Given that

- $\mathbf{Y}$ is an N x 1 column vector of observed values for the outcome variable,
- $\mathbf{X}$ is an N x (P+1) **design matrix* for the set of predictor variables including an intercept term,
- $\boldsymbol{\beta}$ is an (P+1) x 1 column vector of regression coefficients,
- $\mathbf{I}$ is a (P+1) x (P+1) identity matrix,
- and λ is positive real-valued number,

the set of ridge regression coefficients can be estimated using the following matrix operation.

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\mathbf{T}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\mathbf{T}\mathbf{Y}$$

Now, suppose we want to predict the readability score by using the two predictors, the average word length (X_1) and number of sentences (X_2). Our model will be

$$Y_{(i)} = \beta_0 + \beta_1 X_{1(i)} + \beta_2 X_{2(i)} + \epsilon_{(i)}.$$

If we estimate the ridge regression coefficients by using $\lambda = .5$, the estimates would be $\{\beta_0, \beta_1, \beta_2\} = \{0.277,-.593,0.097\}$.

```r
Y <-  as.matrix(readability_sub$target)
X <-  as.matrix(cbind(1,readability_sub$mean.wl,readability_sub$sents))

lambda <- 0.5
```

```r
beta <- solve(t(X)%*%X + lambda*diag(ncol(X)))%*%t(X)%*%Y

beta
```

```
            [,1]
[1,]  0.27693153
[2,] -0.59327091
[3,]  0.09692781
```

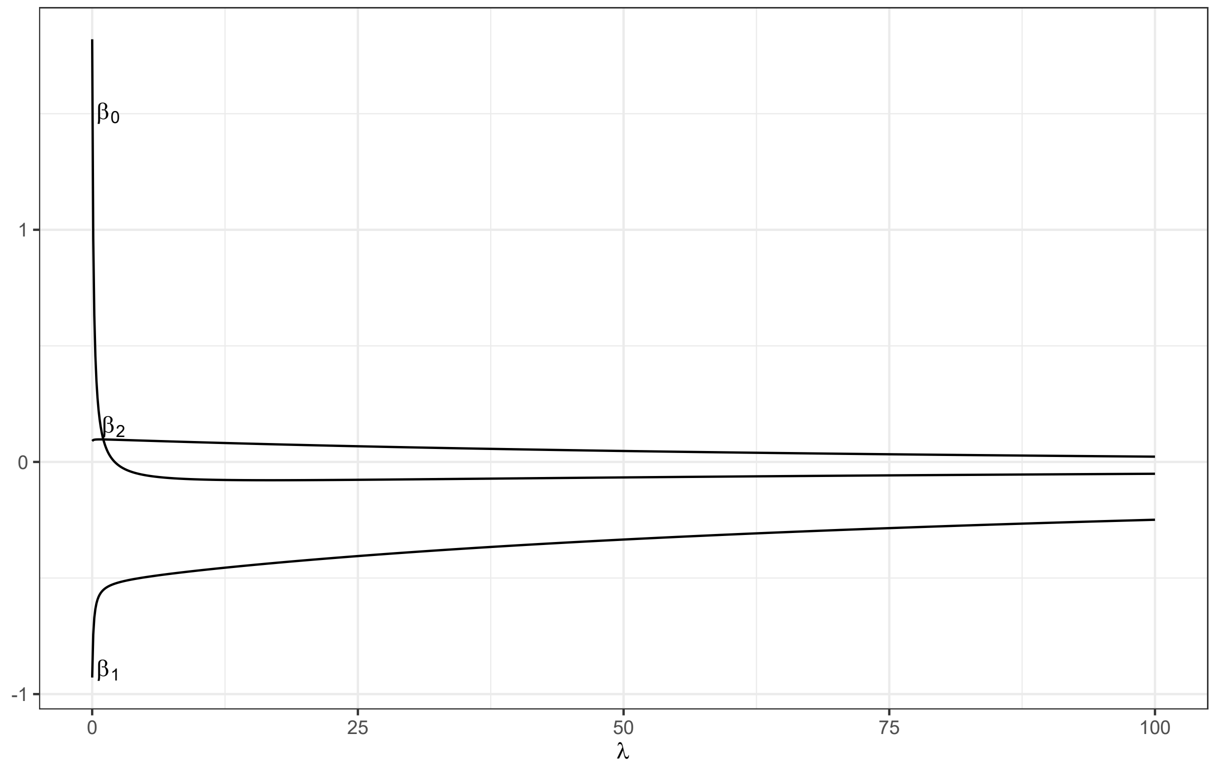
If we change the value of λ to 2, then we will get a different set of estimates for the regression coefficients.

```r
Y <-  as.matrix(readability_sub$target)
X <-  as.matrix(cbind(1,readability_sub$mean.wl,readability_sub$sents))

lambda <- 2

beta <- solve(t(X)%*%X + lambda*diag(ncol(X)))%*%t(X)%*%Y

beta
```

```
             [,1]
[1,]  0.006012867
[2,] -0.526374942
[3,]  0.095845692
```

We can manipulate the value of λ from 0 to 100 with increments of .1 and calculate the regression coefficients for every possible value of λ. Note the regression coefficients will shrink towards zero, but will never be exactly equal to zero in ridge regression.

```r
Y <-  as.matrix(readability_sub$target)
X <-  as.matrix(cbind(1,readability_sub$mean.wl,readability_sub$sents))

lambda <- seq(0,100,.1)

beta     <- data.frame(matrix(nrow=length(lambda),ncol=4))
beta[,1] <- lambda

for(i in 1:length(lambda)){
  beta[i,2:4] <- t(solve(t(X)%*%X + lambda[i]*diag(ncol(X)))%*%t(X)%*%Y)
}

ggplot(data = beta)+
  geom_line(aes(x=X1,y=X2))+
  geom_line(aes(x=X1,y=X3))+
  geom_line(aes(x=X1,y=X4))+
  xlab(expression(lambda))+
  ylab('')+
  theme_bw()+
  annotate(geom='text',x=1.5,y=1.5,label=expression(beta[0]))+
  annotate(geom='text',x=2,y=.15,label=expression(beta[2]))+
  annotate(geom='text',x=1.5,y=-.9,label=expression(beta[1]))
```



Standardized Variables We haven't considered a very important issue for the model estimation. This issue is not necessarily important if you have only one predictor. However, it is critical whenever you have more than one predictor. Different variables have different scales and therefore the magnitude of the regression coefficients for different variables will be dependent on the scales of the variables. A regression coefficient for a predictor with a range from 0 to 100 will be very different than a regression coefficient for a predictor with a range from 0 to 1. Therefore, if we work with the unstandardized variables, ridge penalty will be amplified for the coefficients of those variables with a larger range of values.

Therefore, it is critical that we standardize variables before we use ridge regression. Let's do the example in the previous section, but we now first standardize the variables in our model.

```
Y <-  as.matrix(readability_sub$target)

X <-  as.matrix(cbind(readability_sub$mean.wl,readability_sub$sents))

# Standardize Y

  Y <- scale(Y)

  Y
```

```
            [,1]
 [1,] -1.49010043
 [2,]  1.58384679
 [3,]  0.03504552
 [4,] -0.71410074
 [5,] -0.71199490
 [6,]  0.16122446
```

```
 [7,]  0.99752285
 [8,] -1.72837656
 [9,]  0.36431202
[10,]  1.86598181
[11,]  0.14860203
[12,] -1.11591963
[13,] -0.11002472
[14,] -0.79326406
[15,]  1.07828135
[16,] -0.73444792
[17,]  0.92974794
[18,]  0.29473442
[19,] -0.76059743
[20,]  0.69952720
attr(,"scaled:center")
[1] -1.109433
attr(,"scaled:scale")
[1] 0.9908565
```

```
# Standardized X

  X <- scale(X)
  X
```

```
            [,1]       [,2]
 [1,]  0.6695829 -0.7833675
 [2,] -1.3062112  1.6269940
 [3,] -0.4111573  0.7231084
 [4,] -0.8336993 -0.7833675
 [5,]  0.8801752 -0.9340151
 [6,] -0.3332238  0.8737560
 [7,] -0.8095289 -0.3314247
 [8,]  0.2593876 -1.2353102
 [9,] -0.6449529 -0.4820723
[10,] -0.4727448  2.3802319
[11,]  0.3107526  0.4218133
[12,]  2.9051581 -0.3314247
[13,]  1.0960262 -0.3314247
[14,]  0.5809039 -0.6327199
[15,]  1.0072258  1.0244036
[16,]  0.0783096  0.4218133
[17,] -0.6090069 -0.9340151
[18,] -0.7638842 -0.9340151
[19,] -0.3553022 -0.7833675
[20,] -1.2478105  1.0244036
attr(,"scaled:center")
[1]  4.341704 12.200000
attr(,"scaled:scale")
[1] 0.3912203 6.6380086
```

When we standardize the variables, the mean all variables become zero. So, the intercept estimate for any regression model with standardized variables is guaranteed to be zero. Note that our design matrix doesn't have a column of ones anymore because it is unnecessary (it would be a column of zeros if we had).

First, let's check the coefficients of the regression model with standardized variables when there is no ridge penalty.

```
lambda <- 0

beta.s <- solve(t(X)%*%X + lambda*diag(ncol(X)))%*%t(X)%*%Y

beta.s
```

```
           [,1]
[1,] -0.3666326
[2,]  0.6049359
```

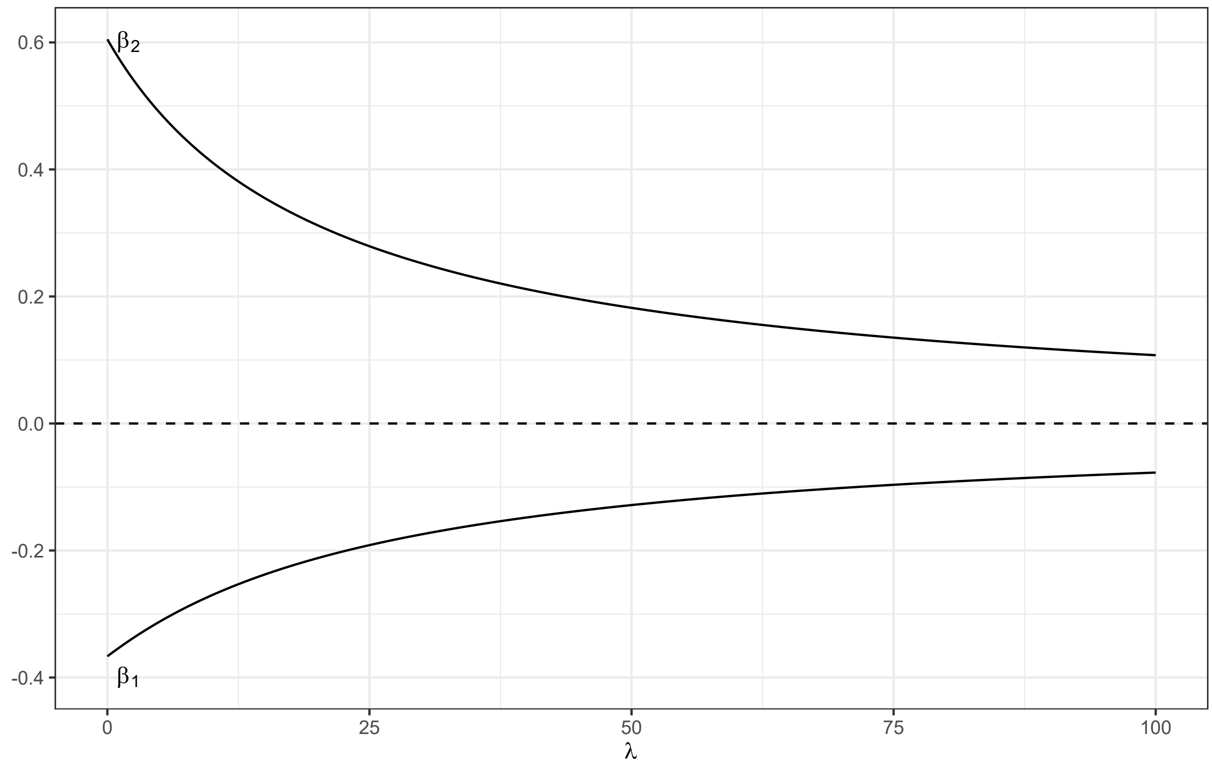
Now, let's increase the ridge penalty to 0.5.

```
lambda <- 0.5

beta.s <- solve(t(X)%*%X + lambda*diag(ncol(X)))%*%t(X)%*%Y

beta.s
```

```
           [,1]
[1,] -0.3604763
[2,]  0.5908420
```

Below, we can manipulate the value of λ from 0 to 100 with increments of .1 as we did before and calculate the standardized regression coefficients for every possible value of λ.

```
Y <-  as.matrix(readability_sub$target)
X <-  as.matrix(cbind(readability_sub$mean.wl,readability_sub$sents))

Y <- scale(Y)
X <- scale(X)

lambda <- seq(0,100,.1)

beta     <- data.frame(matrix(nrow=length(lambda),ncol=3))
beta[,1] <- lambda

for(i in 1:length(lambda)){
  beta[i,2:3] <- t(solve(t(X)%*%X + lambda[i]*diag(ncol(X)))%*%t(X)%*%Y)
}

ggplot(data = beta)+
  geom_line(aes(x=X1,y=X2))+
  geom_line(aes(x=X1,y=X3))+
  xlab(expression(lambda))+
  ylab('')+
  theme_bw()+
  geom_hline(yintercept=0,lty=2) +
  annotate(geom='text',x=2,y=.60,label=expression(beta[2]))+
  annotate(geom='text',x=2,y=-.4,label=expression(beta[1]))
```



`glmnet()` function Similar to `lm` function, we can use `glmnet()` function from the `glmnet` package to run a regression model with ridge penalty. There are many arguments of the `glmnet()` function. For now, the arguments we need to know are

- `x`: an N x P input matrix, where N is the number of observations and P is the number of predictor
- `y`: an N x 1 input matrix for the outcome variable
- `alpha`: a mixing constant for lasso and ridge penalty. When it is zero, the ridge regression is conducted
- `lambda`: penalty term
- `intercept`: set FALSE to avoid intercept for standardized variables

If you want to fit the linear regression without any regularization, you can specify `alpha = 0` and `lambda = 0`.

```
#install.packages('glmnet')

require(glmnet)

Y <-  as.matrix(readability_sub$target)
X <-  as.matrix(cbind(readability_sub$mean.wl,readability_sub$sents))
Y <- scale(Y)
X <- scale(X)

mod <- glmnet(x = X,
              y = Y,
              family = 'gaussian',
              alpha = 0,
              lambda = 0,
```

```
             intercept=FALSE)


coef(mod)
```

```
3 x 1 sparse Matrix of class "dgCMatrix"
                    s0
(Intercept)  .
V1          -0.3666327
V2           0.6049359
```

We can also increase the penalty term (λ).

```
#install.packages('glmnet')

require(glmnet)

Y <-  as.matrix(readability_sub$target)
X <-  as.matrix(cbind(readability_sub$mean.wl,readability_sub$sents))
Y <- scale(Y)
X <- scale(X)

mod <- glmnet(x = X,
              y = Y,
              family = 'gaussian',
              alpha = 0,
              lambda = 0.5,
              intercept=FALSE)


coef(mod)
```

```
3 x 1 sparse Matrix of class "dgCMatrix"
                    s0
(Intercept)  .
V1          -0.2720458
V2           0.4145987
```

NOTE

A careful eye should catch the fact that the coefficient estimates we obtained from `glmnet()` function for the the two standardized variables (average word length and number of sentences) are different when the penalty term (λ) is 0.5. When we apply the matrix solution above for the ridge regression, we obtained the estimates of -0.360 and 0.591 for the two predictors, respectively, at $\lambda = 0.5$. When we enter the same value in `glmnet()`, we obtained the estimates of -0.27 and 0.414. So, what is wrong? Where does this discrepancy come from?

In fact, there is nothing wrong. It appears that what `lambda` argument in `glmnet` indicates is $\frac{\lambda}{N}$. In most statistics textbook, the penalty term for the ridge regression is specified as

$$\lambda \sum_{i=1}^{P} \beta_p^2.$$

On the other hand, if we examine Equation 1-3 in this paper written by the developers of the `glmnet` package, we can see that the penalty term applied is equivalent of

$$\lambda N \sum_{i=1}^{P} \beta_p^2.$$

Therefore, if we want to get the identical results, then we should use $\lambda = 0.5/20$.

```r
N = 20

mod <- glmnet(x = X,
              y = Y,
              family = 'gaussian',
              alpha = 0,
              lambda = 0.5/N,
              intercept=FALSE)


coef(mod)
```

```
3 x 1 sparse Matrix of class "dgCMatrix"
                   s0
(Intercept)  .
V1          -0.3606303
V2           0.5911903
```

Note that these numbers are still slightly different. We can attribute this difference to numerical approximation `glmnet` is using when optimizing the loss function. `glmnet` doesn't use the closed form matrix solution for ridge regression. This is a good thing because there is not always a closed form solution for different types of regularization approaches (e.g., lasso). Therefore, the computational approximation in `glmnet` is very needed moving forward.

Tuning the Hyperparameter λ The λ parameter in ridge regression is called a **hyperparameter**. In the context of machine learning, the parameters in a model can be classified into two types: parameters and hyperparameters. The parameters of the model are typically estimated from data and not set by users. In the context of ridge regression, regression coefficients, $\{\beta_0, \beta_1, ..., \beta_P\}$, are parameters to be estimated from data. On the other hand, the hyperparameters are not estimable, most of the time due to the fact that there is no first order or second order derivatives for these hyperparameters. Therefore, they must be set by the users. In the context of ridge regression, penalty term, $\{\lambda\}$, is a hyperparameter.

The process of deciding what value to use for a hyperparameter is called **tuning**, and it is most of the time a trial-error process. The idea is simple. We try many different values of a hyperparameter and check how well the model performs based on a certain criteria (e.g., MAE, MSE, RMSE) using a k-fold cross validation. Then, we pick the value of a hyperparameter that provides the best performance.

Using Ridge Regression to Predict Readability Scores

In this section, we will apply ridge regression to predict the readability scores from all predictors in the dataset. We will use the `caret` package and use 10-fold cross validation to evaluate the model performance for different levels of penalty term (λ).

```r
# Load the packages

  require(caret)
  require(recipes)
  require(finalfit)
  require(glmnet)

# Import the dataset

  readability <- read.csv('https://raw.githubusercontent.com/uo-datasci-specialization/c4-ml-fall-2021/m

# Initial preparation (remove variables with large amount of missingness)

  require(finalfit)

  missing_ <- ff_glimpse(readability)$Continuous
  flag_na <- which(as.numeric(missing_$missing_percent) > 80)
  readability <- readability[,-flag_na]

# Set the random seed for reproducibility

  set.seed(10152021)

# Train/Test Split

  loc      <- sample(1:nrow(readability), round(nrow(readability) * 0.9))
  read_tr  <- readability[loc, ]
  read_te  <- readability[-loc, ]

# Blueprint

  blueprint <- recipe(x     = readability,
                      vars  = colnames(readability),
                      roles = c(rep('predictor',990),'outcome')) %>%
    step_zv(all_numeric()) %>%
    step_nzv(all_numeric()) %>%
    step_impute_mean(all_numeric()) %>%
    step_normalize(all_numeric_predictors()) %>%
    step_corr(all_numeric(),threshold=0.9)

# Cross validation settings

  cv <- trainControl(method = "cv",
                     p      = 10)

# Tune Grid

  # Here, we have to specify different values of lambda we want to try
  # This should be a dataframe with columns named are the same as
  # the tuning parameters available for the engine we are using

  # In order to get which parameters are available to tune for glmnet
  # run the following code
```

```r
  caret::getModelInfo()$glmnet$parameters

  # This indicates there are two hyperparameters available to tune for the glmnet
  # For ridge regression, we know that we will fix the value of alpha to 0
  # Let's assume that the lambda values we want to try are 1, 5, 10, and 100.


    # Remember how glmnet multiplies the lambda by sample size (N)
    # In this case, the sample size is 2834
    # So, for instance a lambda value of 1 would be 2834
    # You can try larger values and explore, but in this case a max value of 3
    # for lambda would be more than enough. I don't think it will improve performance
    # beyond this value

    # Also, note that there are 100 values, and for every lambda value we will do
    # 10-fold cross validation, so it can take a very long time to search this
    # grid

  grid <- data.frame(alpha = 0, lambda = seq(0.01,3,.01))
  grid


# Train the model

  ridge <- caret::train(blueprint,
                        data      = read_tr,
                        method    = "glmnet",
                        trControl = cv,
                        tuneGrid  = grid)

    # This training took about 3 minutes in my computer

  ridge$results

  ridge$bestTune

  plot(ridge)
```

```
   alpha lambda      RMSE  Rsquared       MAE     RMSESD RsquaredSD      MAESD
1      0   0.01 0.5750266 0.6963298 0.4593452 0.02221126 0.02504427 0.01591786
2      0   0.02 0.5750266 0.6963298 0.4593452 0.02221126 0.02504427 0.01591786
3      0   0.03 0.5750266 0.6963298 0.4593452 0.02221126 0.02504427 0.01591786
4      0   0.04 0.5750266 0.6963298 0.4593452 0.02221126 0.02504427 0.01591786
5      0   0.05 0.5750266 0.6963298 0.4593452 0.02221126 0.02504427 0.01591786
6      0   0.06 0.5745666 0.6967386 0.4590050 0.02225662 0.02504704 0.01598661
7      0   0.07 0.5712485 0.6996988 0.4565815 0.02230691 0.02459360 0.01604447
8      0   0.08 0.5685228 0.7021324 0.4546089 0.02237512 0.02421407 0.01606915
9      0   0.09 0.5662201 0.7041939 0.4529484 0.02244372 0.02389469 0.01613219
10     0   0.10 0.5642541 0.7059545 0.4515073 0.02251416 0.02361760 0.01621680
11     0   0.11 0.5625415 0.7074888 0.4502566 0.02260880 0.02338194 0.01633059
12     0   0.12 0.5610403 0.7088395 0.4491550 0.02266965 0.02315986 0.01642020
13     0   0.13 0.5597290 0.7100172 0.4481867 0.02276865 0.02298978 0.01656136
14     0   0.14 0.5585670 0.7110617 0.4473259 0.02285591 0.02283001 0.01671961
```

15	0	0.15	0.5575110	0.7120154	0.4465375	0.02289951	0.02265871	0.01685539
16	0	0.16	0.5565852	0.7128485	0.4458319	0.02300219	0.02255260	0.01703982
17	0	0.17	0.5557408	0.7136111	0.4451867	0.02307682	0.02243425	0.01718752
18	0	0.18	0.5549560	0.7143224	0.4445800	0.02314049	0.02231704	0.01731194
19	0	0.19	0.5542713	0.7149406	0.4440584	0.02322135	0.02223021	0.01743259
20	0	0.20	0.5536309	0.7155207	0.4435804	0.02329190	0.02213983	0.01755151
21	0	0.21	0.5530515	0.7160482	0.4431387	0.02334379	0.02204746	0.01765745
22	0	0.22	0.5525135	0.7165381	0.4427113	0.02340350	0.02196900	0.01776944
23	0	0.23	0.5520265	0.7169810	0.4423191	0.02347386	0.02191345	0.01788628
24	0	0.24	0.5515700	0.7173971	0.4419665	0.02353869	0.02185562	0.01798902
25	0	0.25	0.5511610	0.7177708	0.4416443	0.02359267	0.02179608	0.01808423
26	0	0.26	0.5507690	0.7181299	0.4413370	0.02364697	0.02173856	0.01818548
27	0	0.27	0.5504161	0.7184534	0.4410616	0.02370846	0.02169912	0.01828827
28	0	0.28	0.5500793	0.7187632	0.4408025	0.02377001	0.02166221	0.01838929
29	0	0.29	0.5497709	0.7190474	0.4405627	0.02382146	0.02162040	0.01847420
30	0	0.30	0.5494945	0.7193030	0.4403503	0.02386048	0.02157649	0.01854237
31	0	0.31	0.5492296	0.7195488	0.4401459	0.02389958	0.02153393	0.01861073
32	0	0.32	0.5489850	0.7197761	0.4399596	0.02394483	0.02150092	0.01868874
33	0	0.33	0.5487566	0.7199890	0.4397816	0.02399354	0.02147442	0.01877180
34	0	0.34	0.5485379	0.7201937	0.4396066	0.02404177	0.02144866	0.01885293
35	0	0.35	0.5483402	0.7203793	0.4394449	0.02408424	0.02142247	0.01892259
36	0	0.36	0.5481604	0.7205489	0.4392937	0.02412281	0.02139804	0.01898807
37	0	0.37	0.5479885	0.7207118	0.4391463	0.02416112	0.02137427	0.01905715
38	0	0.38	0.5478269	0.7208656	0.4390013	0.02419849	0.02135132	0.01912452
39	0	0.39	0.5476842	0.7210021	0.4388713	0.02423388	0.02133245	0.01918348
40	0	0.40	0.5475483	0.7211330	0.4387460	0.02426895	0.02131414	0.01923465
41	0	0.41	0.5474190	0.7212582	0.4386263	0.02430380	0.02129643	0.01927781
42	0	0.42	0.5473015	0.7213725	0.4385193	0.02433732	0.02128058	0.01931250
43	0	0.43	0.5471959	0.7214761	0.4384224	0.02436963	0.02126723	0.01934902
44	0	0.44	0.5470959	0.7215751	0.4383307	0.02440172	0.02125432	0.01938197
45	0	0.45	0.5470014	0.7216693	0.4382440	0.02443359	0.02124189	0.01941110
46	0	0.46	0.5469159	0.7217554	0.4381635	0.02446436	0.02123101	0.01943775
47	0	0.47	0.5468405	0.7218322	0.4380918	0.02449392	0.02122228	0.01946489
48	0	0.48	0.5467698	0.7219051	0.4380249	0.02452326	0.02121390	0.01949545
49	0	0.49	0.5467037	0.7219741	0.4379615	0.02455240	0.02120589	0.01952475
50	0	0.50	0.5466422	0.7220393	0.4378997	0.02458093	0.02119835	0.01955401
51	0	0.51	0.5465912	0.7220944	0.4378437	0.02460843	0.02119350	0.01958076
52	0	0.52	0.5465444	0.7221460	0.4377943	0.02463531	0.02118898	0.01960664
53	0	0.53	0.5465017	0.7221942	0.4377478	0.02466199	0.02118474	0.01963245
54	0	0.54	0.5464628	0.7222392	0.4377061	0.02468851	0.02118084	0.01965707
55	0	0.55	0.5464279	0.7222807	0.4376686	0.02471444	0.02117767	0.01968430
56	0	0.56	0.5464007	0.7223148	0.4376363	0.02473938	0.02117649	0.01971060
57	0	0.57	0.5463770	0.7223461	0.4376065	0.02476387	0.02117550	0.01973780
58	0	0.58	0.5463567	0.7223745	0.4375781	0.02478819	0.02117474	0.01976607
59	0	0.59	0.5463396	0.7224002	0.4375508	0.02481235	0.02117424	0.01979511
60	0	0.60	0.5463254	0.7224235	0.4375254	0.02483629	0.02117429	0.01982503
61	0	0.61	0.5463164	0.7224417	0.4375036	0.02485931	0.02117582	0.01985574
62	0	0.62	0.5463113	0.7224566	0.4374863	0.02488170	0.02117781	0.01988783
63	0	0.63	0.5463089	0.7224692	0.4374708	0.02490393	0.02117999	0.01992076
64	0	0.64	0.5463093	0.7224794	0.4374574	0.02492600	0.02118236	0.01995520
65	0	0.65	0.5463124	0.7224874	0.4374452	0.02494797	0.02118499	0.01998948
66	0	0.66	0.5463177	0.7224935	0.4374327	0.02496970	0.02118833	0.02002403
67	0	0.67	0.5463272	0.7224955	0.4374223	0.02499059	0.02119275	0.02005693
68	0	0.68	0.5463392	0.7224954	0.4374166	0.02501098	0.02119742	0.02008545

69	0	0.69	0.5463537	0.7224932	0.4374115	0.02503123	0.02120224	0.02011317
70	0	0.70	0.5463705	0.7224891	0.4374086	0.02505135	0.02120722	0.02014139
71	0	0.71	0.5463897	0.7224830	0.4374075	0.02507135	0.02121239	0.02017062
72	0	0.72	0.5464108	0.7224752	0.4374074	0.02509129	0.02121794	0.02019926
73	0	0.73	0.5464338	0.7224655	0.4374079	0.02511081	0.02122467	0.02022799
74	0	0.74	0.5464595	0.7224535	0.4374117	0.02512947	0.02123157	0.02025536
75	0	0.75	0.5464872	0.7224398	0.4374159	0.02514800	0.02123860	0.02028186
76	0	0.76	0.5465169	0.7224243	0.4374202	0.02516642	0.02124577	0.02030846
77	0	0.77	0.5465486	0.7224072	0.4374246	0.02518472	0.02125308	0.02033529
78	0	0.78	0.5465822	0.7223885	0.4374314	0.02520294	0.02126057	0.02035878
79	0	0.79	0.5466175	0.7223683	0.4374383	0.02522112	0.02126840	0.02038218
80	0	0.80	0.5466538	0.7223471	0.4374447	0.02523903	0.02127739	0.02040596
81	0	0.81	0.5466919	0.7223244	0.4374523	0.02525609	0.02128626	0.02042717
82	0	0.82	0.5467316	0.7223002	0.4374606	0.02527303	0.02129526	0.02044750
83	0	0.83	0.5467731	0.7222745	0.4374691	0.02528987	0.02130437	0.02046765
84	0	0.84	0.5468163	0.7222475	0.4374787	0.02530661	0.02131360	0.02048681
85	0	0.85	0.5468612	0.7222189	0.4374892	0.02532324	0.02132296	0.02050505
86	0	0.86	0.5469075	0.7221892	0.4375022	0.02533989	0.02133257	0.02051936
87	0	0.87	0.5469551	0.7221584	0.4375166	0.02535632	0.02134262	0.02053334
88	0	0.88	0.5470030	0.7221271	0.4375321	0.02537267	0.02135362	0.02054684
89	0	0.89	0.5470520	0.7220949	0.4375480	0.02538823	0.02136423	0.02055948
90	0	0.90	0.5471025	0.7220615	0.4375640	0.02540371	0.02137495	0.02057229
91	0	0.91	0.5471544	0.7220269	0.4375821	0.02541910	0.02138576	0.02058377
92	0	0.92	0.5472078	0.7219910	0.4376012	0.02543441	0.02139669	0.02059429
93	0	0.93	0.5472627	0.7219539	0.4376203	0.02544964	0.02140772	0.02060492
94	0	0.94	0.5473188	0.7219157	0.4376400	0.02546483	0.02141893	0.02061546
95	0	0.95	0.5473761	0.7218765	0.4376610	0.02548007	0.02143042	0.02062481
96	0	0.96	0.5474337	0.7218371	0.4376821	0.02549525	0.02144304	0.02063421
97	0	0.97	0.5474911	0.7217976	0.4377028	0.02550967	0.02145517	0.02064380
98	0	0.98	0.5475497	0.7217574	0.4377257	0.02552391	0.02146728	0.02065547
99	0	0.99	0.5476094	0.7217160	0.4377494	0.02553807	0.02147949	0.02066592
100	0	1.00	0.5476704	0.7216737	0.4377733	0.02555216	0.02149179	0.02067624
101	0	1.01	0.5477327	0.7216303	0.4377973	0.02556617	0.02150419	0.02068648
102	0	1.02	0.5477961	0.7215858	0.4378215	0.02558012	0.02151668	0.02069701
103	0	1.03	0.5478608	0.7215403	0.4378465	0.02559403	0.02152931	0.02070853
104	0	1.04	0.5479262	0.7214941	0.4378715	0.02560803	0.02154224	0.02072018
105	0	1.05	0.5479922	0.7214475	0.4378961	0.02562190	0.02155594	0.02073219
106	0	1.06	0.5480575	0.7214014	0.4379198	0.02563555	0.02156975	0.02074486
107	0	1.07	0.5481230	0.7213553	0.4379432	0.02564865	0.02158306	0.02075685
108	0	1.08	0.5481895	0.7213083	0.4379677	0.02566169	0.02159646	0.02076824
109	0	1.09	0.5482571	0.7212603	0.4379932	0.02567467	0.02160995	0.02077948
110	0	1.10	0.5483258	0.7212115	0.4380193	0.02568759	0.02162352	0.02079016
111	0	1.11	0.5483955	0.7211618	0.4380467	0.02570045	0.02163718	0.02079963
112	0	1.12	0.5484663	0.7211111	0.4380748	0.02571325	0.02165093	0.02080889
113	0	1.13	0.5485380	0.7210597	0.4381036	0.02572602	0.02166480	0.02081802
114	0	1.14	0.5486104	0.7210076	0.4381330	0.02573891	0.02167898	0.02082746
115	0	1.15	0.5486835	0.7209550	0.4381635	0.02575153	0.02169353	0.02083789
116	0	1.16	0.5487556	0.7209032	0.4381932	0.02576453	0.02170899	0.02084942
117	0	1.17	0.5488269	0.7208521	0.4382226	0.02577657	0.02172331	0.02086046
118	0	1.18	0.5488992	0.7208003	0.4382522	0.02578860	0.02173772	0.02087161
119	0	1.19	0.5489723	0.7207478	0.4382818	0.02580058	0.02175220	0.02088292
120	0	1.20	0.5490464	0.7206945	0.4383129	0.02581251	0.02176677	0.02089559
121	0	1.21	0.5491214	0.7206404	0.4383453	0.02582439	0.02178141	0.02090894
122	0	1.22	0.5491972	0.7205855	0.4383776	0.02583623	0.02179613	0.02092239

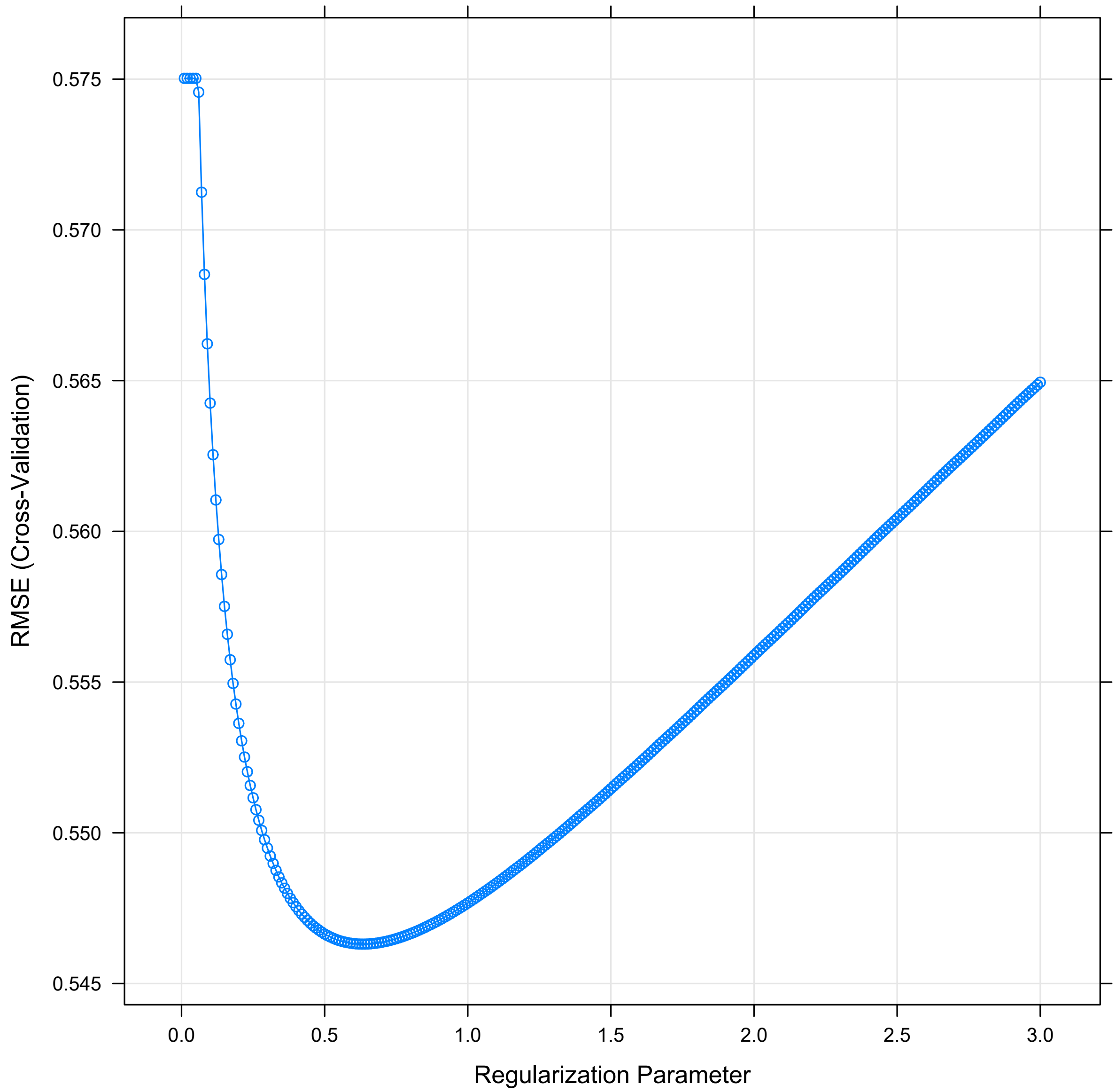
123	0	1.23	0.5492740	0.7205299	0.4384109	0.02584801	0.02181093	0.02093520
124	0	1.24	0.5493516	0.7204735	0.4384459	0.02585977	0.02182585	0.02094879
125	0	1.25	0.5494297	0.7204167	0.4384815	0.02587167	0.02184108	0.02096255
126	0	1.26	0.5495084	0.7203594	0.4385173	0.02588339	0.02185658	0.02097621
127	0	1.27	0.5495864	0.7203027	0.4385521	0.02589556	0.02187321	0.02099090
128	0	1.28	0.5496628	0.7202474	0.4385860	0.02590667	0.02188848	0.02100500
129	0	1.29	0.5497397	0.7201918	0.4386199	0.02591780	0.02190371	0.02101916
130	0	1.30	0.5498174	0.7201356	0.4386548	0.02592889	0.02191902	0.02103444
131	0	1.31	0.5498958	0.7200787	0.4386910	0.02593994	0.02193439	0.02105117
132	0	1.32	0.5499750	0.7200212	0.4387277	0.02595094	0.02194984	0.02106852
133	0	1.33	0.5500550	0.7199630	0.4387656	0.02596190	0.02196536	0.02108452
134	0	1.34	0.5501358	0.7199041	0.4388044	0.02597282	0.02198096	0.02109925
135	0	1.35	0.5502174	0.7198446	0.4388439	0.02598369	0.02199663	0.02111361
136	0	1.36	0.5502996	0.7197844	0.4388836	0.02599454	0.02201239	0.02112788
137	0	1.37	0.5503822	0.7197239	0.4389232	0.02600557	0.02202848	0.02114236
138	0	1.38	0.5504655	0.7196628	0.4389634	0.02601653	0.02204468	0.02115672
139	0	1.39	0.5505484	0.7196020	0.4390039	0.02602774	0.02206214	0.02117150
140	0	1.40	0.5506294	0.7195429	0.4390433	0.02603825	0.02207855	0.02118595
141	0	1.41	0.5507099	0.7194845	0.4390829	0.02604858	0.02209444	0.02119941
142	0	1.42	0.5507911	0.7194255	0.4391234	0.02605886	0.02211039	0.02121355
143	0	1.43	0.5508729	0.7193659	0.4391643	0.02606911	0.02212641	0.02122807
144	0	1.44	0.5509554	0.7193058	0.4392077	0.02607933	0.02214249	0.02124214
145	0	1.45	0.5510386	0.7192451	0.4392519	0.02608951	0.02215864	0.02125699
146	0	1.46	0.5511225	0.7191838	0.4392963	0.02609965	0.02217486	0.02127186
147	0	1.47	0.5512070	0.7191219	0.4393412	0.02610976	0.02219114	0.02128621
148	0	1.48	0.5512922	0.7190594	0.4393862	0.02611983	0.02220749	0.02130042
149	0	1.49	0.5513780	0.7189964	0.4394313	0.02612987	0.02222391	0.02131458
150	0	1.50	0.5514641	0.7189331	0.4394770	0.02614007	0.02224063	0.02132887
151	0	1.51	0.5515507	0.7188694	0.4395239	0.02615028	0.02225747	0.02134226
152	0	1.52	0.5516375	0.7188054	0.4395712	0.02616022	0.02227473	0.02135665
153	0	1.53	0.5517230	0.7187426	0.4396175	0.02617097	0.02229298	0.02137225
154	0	1.54	0.5518067	0.7186816	0.4396632	0.02618040	0.02230948	0.02138733
155	0	1.55	0.5518905	0.7186207	0.4397096	0.02618996	0.02232592	0.02140266
156	0	1.56	0.5519748	0.7185592	0.4397561	0.02619950	0.02234241	0.02141814
157	0	1.57	0.5520598	0.7184973	0.4398027	0.02620901	0.02235896	0.02143373
158	0	1.58	0.5521453	0.7184348	0.4398500	0.02621849	0.02237558	0.02144891
159	0	1.59	0.5522313	0.7183719	0.4398978	0.02622794	0.02239225	0.02146449
160	0	1.60	0.5523180	0.7183084	0.4399463	0.02623736	0.02240899	0.02148101
161	0	1.61	0.5524052	0.7182445	0.4399954	0.02624674	0.02242579	0.02149796
162	0	1.62	0.5524930	0.7181801	0.4400450	0.02625610	0.02244264	0.02151487
163	0	1.63	0.5525813	0.7181151	0.4400947	0.02626543	0.02245956	0.02153185
164	0	1.64	0.5526701	0.7180498	0.4401442	0.02627480	0.02247663	0.02154892
165	0	1.65	0.5527590	0.7179843	0.4401946	0.02628432	0.02249396	0.02156713
166	0	1.66	0.5528485	0.7179183	0.4402459	0.02629382	0.02251136	0.02158487
167	0	1.67	0.5529378	0.7178524	0.4402976	0.02630314	0.02252945	0.02160189
168	0	1.68	0.5530256	0.7177880	0.4403483	0.02631311	0.02254804	0.02161984
169	0	1.69	0.5531115	0.7177253	0.4403985	0.02632181	0.02256490	0.02163647
170	0	1.70	0.5531975	0.7176627	0.4404485	0.02633070	0.02258170	0.02165295
171	0	1.71	0.5532840	0.7175997	0.4404988	0.02633958	0.02259857	0.02166984
172	0	1.72	0.5533710	0.7175363	0.4405494	0.02634842	0.02261549	0.02168683
173	0	1.73	0.5534585	0.7174724	0.4406004	0.02635725	0.02263246	0.02170374
174	0	1.74	0.5535465	0.7174081	0.4406514	0.02636604	0.02264949	0.02172070
175	0	1.75	0.5536349	0.7173434	0.4407024	0.02637481	0.02266657	0.02173770
176	0	1.76	0.5537239	0.7172782	0.4407543	0.02638355	0.02268371	0.02175406

177	0	1.77	0.5538133	0.7172126	0.4408062	0.02639227	0.02270090	0.02177041
178	0	1.78	0.5539033	0.7171466	0.4408584	0.02640096	0.02271815	0.02178708
179	0	1.79	0.5539937	0.7170802	0.4409109	0.02640963	0.02273546	0.02180413
180	0	1.80	0.5540844	0.7170134	0.4409633	0.02641836	0.02275292	0.02182128
181	0	1.81	0.5541752	0.7169465	0.4410155	0.02642724	0.02277063	0.02183858
182	0	1.82	0.5542665	0.7168792	0.4410684	0.02643610	0.02278841	0.02185662
183	0	1.83	0.5543579	0.7168118	0.4411226	0.02644471	0.02280661	0.02187356
184	0	1.84	0.5544484	0.7167453	0.4411768	0.02645445	0.02282609	0.02189022
185	0	1.85	0.5545363	0.7166812	0.4412299	0.02646239	0.02284336	0.02190533
186	0	1.86	0.5546236	0.7166177	0.4412834	0.02647069	0.02286038	0.02192064
187	0	1.87	0.5547114	0.7165539	0.4413381	0.02647896	0.02287745	0.02193657
188	0	1.88	0.5547995	0.7164898	0.4413937	0.02648722	0.02289456	0.02195301
189	0	1.89	0.5548881	0.7164252	0.4414496	0.02649545	0.02291173	0.02196964
190	0	1.90	0.5549772	0.7163604	0.4415064	0.02650366	0.02292895	0.02198554
191	0	1.91	0.5550666	0.7162951	0.4415635	0.02651184	0.02294621	0.02200099
192	0	1.92	0.5551565	0.7162295	0.4416211	0.02652001	0.02296353	0.02201603
193	0	1.93	0.5552467	0.7161635	0.4416790	0.02652815	0.02298090	0.02203100
194	0	1.94	0.5553374	0.7160971	0.4417371	0.02653627	0.02299831	0.02204573
195	0	1.95	0.5554286	0.7160304	0.4417961	0.02654436	0.02301578	0.02206131
196	0	1.96	0.5555201	0.7159632	0.4418555	0.02655244	0.02303329	0.02207703
197	0	1.97	0.5556121	0.7158958	0.4419157	0.02656049	0.02305086	0.02209290
198	0	1.98	0.5557040	0.7158282	0.4419759	0.02656874	0.02306872	0.02210891
199	0	1.99	0.5557962	0.7157604	0.4420363	0.02657703	0.02308671	0.02212482
200	0	2.00	0.5558888	0.7156922	0.4420983	0.02658530	0.02310474	0.02213923
201	0	2.01	0.5559813	0.7156241	0.4421615	0.02659326	0.02312330	0.02215343
202	0	2.02	0.5560726	0.7155572	0.4422246	0.02660256	0.02314304	0.02216770
203	0	2.03	0.5561615	0.7154925	0.4422863	0.02660982	0.02316039	0.02217977
204	0	2.04	0.5562498	0.7154286	0.4423482	0.02661755	0.02317750	0.02219194
205	0	2.05	0.5563383	0.7153645	0.4424106	0.02662527	0.02319466	0.02220484
206	0	2.06	0.5564273	0.7153000	0.4424740	0.02663296	0.02321187	0.02221811
207	0	2.07	0.5565166	0.7152351	0.4425382	0.02664064	0.02322911	0.02223056
208	0	2.08	0.5566063	0.7151700	0.4426025	0.02664829	0.02324641	0.02224304
209	0	2.09	0.5566963	0.7151045	0.4426667	0.02665593	0.02326375	0.02225554
210	0	2.10	0.5567867	0.7150388	0.4427309	0.02666355	0.02328113	0.02226808
211	0	2.11	0.5568774	0.7149727	0.4427955	0.02667115	0.02329855	0.02228099
212	0	2.12	0.5569685	0.7149063	0.4428601	0.02667872	0.02331602	0.02229392
213	0	2.13	0.5570600	0.7148395	0.4429247	0.02668628	0.02333354	0.02230689
214	0	2.14	0.5571519	0.7147725	0.4429894	0.02669382	0.02335110	0.02232006
215	0	2.15	0.5572441	0.7147051	0.4430545	0.02670134	0.02336870	0.02233370
216	0	2.16	0.5573366	0.7146374	0.4431197	0.02670883	0.02338635	0.02234738
217	0	2.17	0.5574292	0.7145696	0.4431853	0.02671648	0.02340423	0.02236030
218	0	2.18	0.5575219	0.7145017	0.4432515	0.02672423	0.02342229	0.02237253
219	0	2.19	0.5576149	0.7144335	0.4433176	0.02673197	0.02344040	0.02238484
220	0	2.20	0.5577081	0.7143651	0.4433850	0.02673957	0.02345876	0.02239637
221	0	2.21	0.5578009	0.7142971	0.4434520	0.02674750	0.02347795	0.02240835
222	0	2.22	0.5578918	0.7142309	0.4435184	0.02675568	0.02349699	0.02242035
223	0	2.23	0.5579809	0.7141664	0.4435836	0.02676242	0.02351420	0.02243109
224	0	2.24	0.5580695	0.7141025	0.4436485	0.02676963	0.02353130	0.02244189
225	0	2.25	0.5581585	0.7140384	0.4437137	0.02677682	0.02354843	0.02245283
226	0	2.26	0.5582477	0.7139739	0.4437788	0.02678399	0.02356560	0.02246379
227	0	2.27	0.5583373	0.7139092	0.4438442	0.02679114	0.02358281	0.02247481
228	0	2.28	0.5584272	0.7138443	0.4439101	0.02679828	0.02360006	0.02248609
229	0	2.29	0.5585173	0.7137790	0.4439764	0.02680540	0.02361735	0.02249726
230	0	2.30	0.5586078	0.7137135	0.4440426	0.02681250	0.02363468	0.02250847

231	0	2.31	0.5586987	0.7136477	0.4441089	0.02681958	0.02365205	0.02251970
232	0	2.32	0.5587898	0.7135816	0.4441751	0.02682665	0.02366946	0.02253096
233	0	2.33	0.5588812	0.7135152	0.4442414	0.02683369	0.02368691	0.02254224
234	0	2.34	0.5589729	0.7134486	0.4443076	0.02684072	0.02370440	0.02255356
235	0	2.35	0.5590650	0.7133817	0.4443742	0.02684774	0.02372192	0.02256510
236	0	2.36	0.5591574	0.7133145	0.4444408	0.02685473	0.02373949	0.02257674
237	0	2.37	0.5592500	0.7132470	0.4445075	0.02686171	0.02375709	0.02258841
238	0	2.38	0.5593427	0.7131795	0.4445739	0.02686882	0.02377489	0.02260032
239	0	2.39	0.5594354	0.7131119	0.4446399	0.02687606	0.02379291	0.02261250
240	0	2.40	0.5595284	0.7130440	0.4447062	0.02688329	0.02381097	0.02262483
241	0	2.41	0.5596217	0.7129758	0.4447732	0.02689051	0.02382908	0.02263764
242	0	2.42	0.5597148	0.7129078	0.4448400	0.02689743	0.02384772	0.02265051
243	0	2.43	0.5598071	0.7128406	0.4449065	0.02690568	0.02386758	0.02266389
244	0	2.44	0.5598969	0.7127757	0.4449716	0.02691240	0.02388535	0.02267554
245	0	2.45	0.5599856	0.7127121	0.4450361	0.02691873	0.02390230	0.02268694
246	0	2.46	0.5600742	0.7126486	0.4451010	0.02692543	0.02391924	0.02269858
247	0	2.47	0.5601630	0.7125849	0.4451660	0.02693212	0.02393623	0.02271015
248	0	2.48	0.5602521	0.7125209	0.4452309	0.02693880	0.02395325	0.02272175
249	0	2.49	0.5603414	0.7124568	0.4452959	0.02694546	0.02397030	0.02273338
250	0	2.50	0.5604310	0.7123924	0.4453609	0.02695210	0.02398739	0.02274509
251	0	2.51	0.5605209	0.7123277	0.4454261	0.02695873	0.02400451	0.02275701
252	0	2.52	0.5606111	0.7122628	0.4454913	0.02696534	0.02402166	0.02276885
253	0	2.53	0.5607015	0.7121977	0.4455570	0.02697193	0.02403885	0.02278027
254	0	2.54	0.5607922	0.7121323	0.4456226	0.02697851	0.02405608	0.02279171
255	0	2.55	0.5608832	0.7120667	0.4456883	0.02698508	0.02407334	0.02280321
256	0	2.56	0.5609744	0.7120009	0.4457540	0.02699163	0.02409063	0.02281480
257	0	2.57	0.5610659	0.7119348	0.4458198	0.02699816	0.02410796	0.02282642
258	0	2.58	0.5611576	0.7118685	0.4458862	0.02700468	0.02412533	0.02283737
259	0	2.59	0.5612497	0.7118019	0.4459527	0.02701118	0.02414273	0.02284814
260	0	2.60	0.5613420	0.7117351	0.4460192	0.02701766	0.02416016	0.02285893
261	0	2.61	0.5614343	0.7116682	0.4460857	0.02702423	0.02417773	0.02286974
262	0	2.62	0.5615266	0.7116013	0.4461520	0.02703100	0.02419558	0.02288076
263	0	2.63	0.5616191	0.7115342	0.4462182	0.02703775	0.02421346	0.02289180
264	0	2.64	0.5617118	0.7114668	0.4462845	0.02704449	0.02423138	0.02290287
265	0	2.65	0.5618046	0.7113994	0.4463508	0.02705109	0.02424957	0.02291398
266	0	2.66	0.5618970	0.7113323	0.4464168	0.02705788	0.02426837	0.02292538
267	0	2.67	0.5619883	0.7112663	0.4464818	0.02706571	0.02428789	0.02293755
268	0	2.68	0.5620772	0.7112027	0.4465457	0.02707144	0.02430493	0.02294895
269	0	2.69	0.5621652	0.7111401	0.4466093	0.02707735	0.02432153	0.02296020
270	0	2.70	0.5622532	0.7110775	0.4466728	0.02708358	0.02433821	0.02297155
271	0	2.71	0.5623414	0.7110146	0.4467365	0.02708979	0.02435493	0.02298303
272	0	2.72	0.5624298	0.7109516	0.4468001	0.02709600	0.02437167	0.02299457
273	0	2.73	0.5625185	0.7108883	0.4468645	0.02710218	0.02438844	0.02300529
274	0	2.74	0.5626074	0.7108249	0.4469291	0.02710836	0.02440525	0.02301576
275	0	2.75	0.5626966	0.7107612	0.4469937	0.02711452	0.02442208	0.02302618
276	0	2.76	0.5627860	0.7106974	0.4470584	0.02712066	0.02443894	0.02303662
277	0	2.77	0.5628756	0.7106333	0.4471230	0.02712679	0.02445583	0.02304707
278	0	2.78	0.5629654	0.7105691	0.4471879	0.02713291	0.02447276	0.02305739
279	0	2.79	0.5630555	0.7105046	0.4472529	0.02713901	0.02448971	0.02306764
280	0	2.80	0.5631458	0.7104399	0.4473179	0.02714510	0.02450669	0.02307791
281	0	2.81	0.5632363	0.7103750	0.4473830	0.02715117	0.02452371	0.02308797
282	0	2.82	0.5633271	0.7103099	0.4474492	0.02715723	0.02454075	0.02309778
283	0	2.83	0.5634180	0.7102446	0.4475157	0.02716328	0.02455783	0.02310708
284	0	2.84	0.5635093	0.7101791	0.4475823	0.02716931	0.02457493	0.02311641

285	0	2.85	0.5636007	0.7101134	0.4476488	0.02717532	0.02459206	0.02312576
286	0	2.86	0.5636924	0.7100474	0.4477154	0.02718133	0.02460923	0.02313512
287	0	2.87	0.5637838	0.7099817	0.4477815	0.02718760	0.02462674	0.02314483
288	0	2.88	0.5638753	0.7099157	0.4478479	0.02719390	0.02464431	0.02315443
289	0	2.89	0.5639670	0.7098496	0.4479143	0.02720018	0.02466192	0.02316405
290	0	2.90	0.5640590	0.7097832	0.4479816	0.02720645	0.02467957	0.02317317
291	0	2.91	0.5641509	0.7097170	0.4480487	0.02721254	0.02469758	0.02318236
292	0	2.92	0.5642424	0.7096510	0.4481154	0.02721898	0.02471617	0.02319195
293	0	2.93	0.5643329	0.7095861	0.4481812	0.02722639	0.02473539	0.02320220
294	0	2.94	0.5644208	0.7095236	0.4482455	0.02723165	0.02475207	0.02321130
295	0	2.95	0.5645080	0.7094620	0.4483093	0.02723697	0.02476834	0.02322017
296	0	2.96	0.5645950	0.7094006	0.4483729	0.02724275	0.02478464	0.02322893
297	0	2.97	0.5646822	0.7093390	0.4484370	0.02724851	0.02480096	0.02323709
298	0	2.98	0.5647696	0.7092773	0.4485012	0.02725426	0.02481731	0.02324532
299	0	2.99	0.5648572	0.7092154	0.4485654	0.02726000	0.02483369	0.02325356
300	0	3.00	0.5649450	0.7091533	0.4486296	0.02726572	0.02485009	0.02326182

```
   alpha lambda
63     0   0.63
```



The 10-fold cross-validation results on the training dataset indicate that a λ value of 0.63 provides the best performance (minimum RMSE). Let's use this model to predict the outcome in the hold-out test dataset.

```
predict_te_ridge <- predict(ridge, read_te)

rsq_te <- cor(read_te$target,predict_te_ridge)^2
rsq_te
```

```
[1] 0.7271192
```

```
mae_te <- mean(abs(read_te$target - predict_te_ridge))
mae_te
```

```
[1] 0.4345475
```

```
rmse_te <- sqrt(mean((read_te$target - predict_te_ridge)^2))
rmse_te
```

```
[1] 0.5357382
```

Below is a table to compare the performance of ridge regression and linear regression (from earlier lecture) on the test dataset.

	R-square	MAE	RMSE
Linear Regression	0.644	0.522	0.644
Ridge Regression	0.727	0.435	0.536

Lasso Regression

Lasso Penalty

Elastic Net