

## Prepare the Data for glmnet()

- Let us split the sample dataset randomly into the training and test datasets with a ratio of 20-80.

```
set.seed(4991)
rate <- 0.2
train.size <- round(nrow(df.housing) * rate)
sample <- sample(nrow(df.housing), train.size)
training <- df.housing[sample, ]
test <- df.housing[-sample, ]
```

- As the glmnet() function only accepts a data matrix as X, we need to use the function model.matrix() to transform a data frame into a data matrix.

```
train.x <- model.matrix(Price ~., data = training)[,-1]
train.y <- training[,ncol(training)]
test.x <- model.matrix(Price ~., data = test)[,-1]
test.y <- test[,ncol(test)]
```

## Analyse: Model Building ( $\lambda = 0.1$ )

- Let us first try  $\lambda = 0.1$ .

```
model_ridge_trial1 <- glmnet(train.x,train.y, alpha = 0, lambda =  
  0.1)  
t(coef(model_ridge_trial1))
```

	(Intercept)	Crime_rate	Industry	Number_of_rooms	Access_to_highways	Tax_rate
s0	-3.514909	-0.1645028	-0.001327936	0.7245108	-0.02651105	-0.1642132

- The above list gives the (standardised) coefficients of the Ridge Regression model.
- “Industry”, “Access to highways” and “Tax rate” have smaller coefficients, compared with those of the MLR model.

## Analyse: Model Building ( $\lambda = 0.1$ )

```
summary(model_ridge_trial1)
```

	Length	Class	Mode
a0	1	-none-	numeric
beta	5	dgCMatrix	S4
df	1	-none-	numeric
dim	2	-none-	numeric
lambda	1	-none-	numeric
dev.ratio	1	-none-	numeric
nulldev	1	-none-	numeric
npasses	1	-none-	numeric
jerr	1	-none-	numeric
offset	1	-none-	logical
call	5	-none-	call
nobs	1	-none-	numeric

- For Ridge and LASSO Regression, the `summary()` function only gives a list of items that are included in the model.

## Analyse: Model Building ( $\lambda = 0.5, 1$ )

- Next, try  $\lambda = 0.5$ .

```
model_ridge_trial2 <- glmnet(train.x,train.y, alpha = 0, lambda =  
  0.5)  
t(coef(model_ridge_trial2))
```

	(Intercept)	Crime_rate	Industry	Number_of_rooms	Access_to_highways	Tax_rate
s0	-1.835258	-0.1446035	-0.05670596	0.5375144	-0.06324133	-0.113332

- Finally, try  $\lambda = 1$ .

```
model_ridge_trial3 <- glmnet(train.x,train.y, alpha = 0, lambda = 1)  
t(coef(model_ridge_trial3))
```

	(Intercept)	Crime_rate	Industry	Number_of_rooms	Access_to_highways	Tax_rate
s0	-0.711388	-0.1249319	-0.07160983	0.4111255	-0.06852507	-0.101091

## Compare the Coefficients of three Models ( $\lambda = 0.1, 0.5, 1$ )

	$\lambda = 0.1$	$\lambda = 0.5$	$\lambda = 1$
(Intercept)	-3.514909287	-1.83525756	-0.71138804
Crime_rate	-0.164502848	-0.14460354	-0.12493190
Industry	-0.001327936	-0.05670596	-0.07160983
Number_of_rooms	0.724510840	0.53751435	0.41112553
Access_to_highways	-0.026511049	-0.06324133	-0.06852507
Tax_rate	-0.164213158	-0.11333199	-0.10109101

- When  $\lambda$  value increases, the coefficients of most predictors tend to become smaller, and closer to zero.
- In general, a larger  $\lambda$  value means that a higher degree of regularisation is carried out, and in turn, the absolute values of the coefficients, for most predictors, tend to be smaller.
- We can visualise the changes of the coefficients of various models against different  $\lambda$  values, in a single plot.

# Train 100 Ridge Regression Models

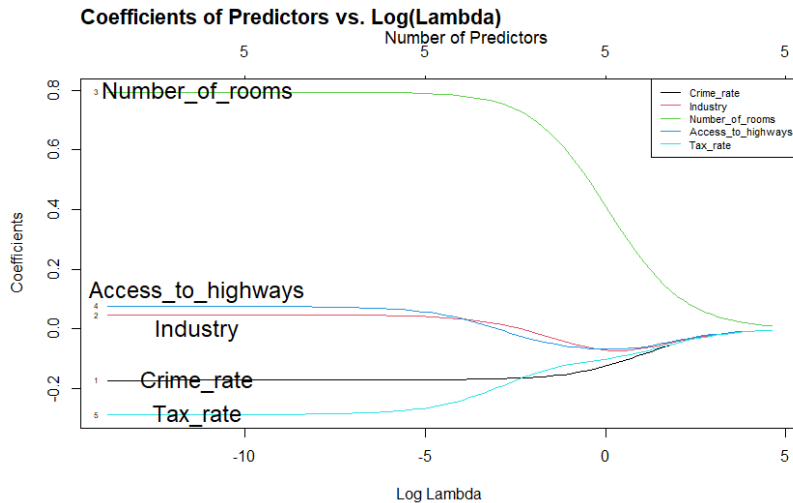
- Let us first train 100 Ridge Regression models using a sequence of lambda values, from  $10^{-6}$  to 100.

```
lambda <- 10^seq(-6, 2, length = 100)
ridge_model <- glmnet(train.x, train.y, alpha = 0, lambda = lambda)
```

- Then we use the following code chunk to generate the plot for the Coefficients vs. Log(Lambda):

```
add_lbs <- function(fit, offset_x=2.5) {
  L <- length(fit$lambda)
  x <- log(fit$lambda[L]) + offset_x
  y <- fit$beta[, L]
  labs <- names(y)
  text(x, y, labels=labs, cex = 1.1)}
plot(ridge_model, xvar = "lambda", label = TRUE)
add_lbs(ridge_model)
legend("topright", lwd = 1, col = 1:6, legend = colnames(train.x),
      cex = .7)
```

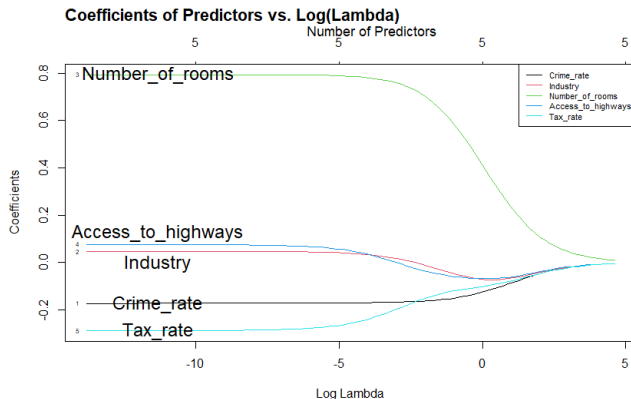
# Coefficients vs. $\text{Log}(\lambda)$



# Coefficients vs. $\text{Log}(\lambda)$

The plot shows:

- X axis is the Logarithm of the regularisation parameter  $\lambda$ .
- Y axis is the standardised coefficients for each predictor.
- In general, larger  $\lambda$  values result in more regularisation, and consequently, the coefficients will approach zero eventually.

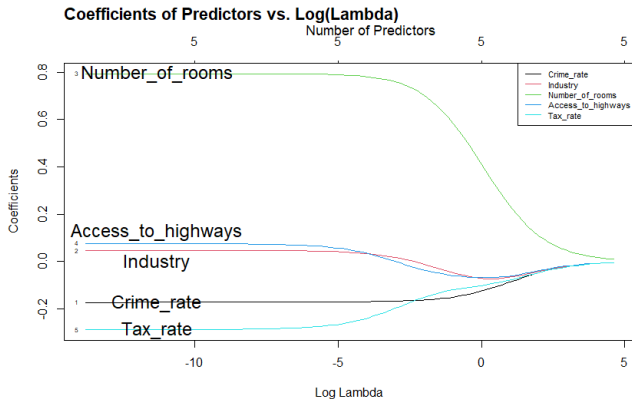




# Coefficients vs. $\text{Log}(\lambda)$

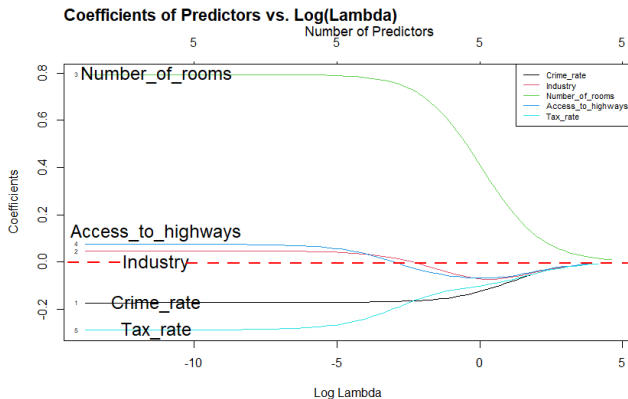
The plot shows:

- The numbers on the top axis indicate how many coefficients are non-zero.
- Ridge Regression does not perform variable selection directly.
- “Number of rooms” seem to be the most influential predictor, regardless of the choice of  $\lambda$ .



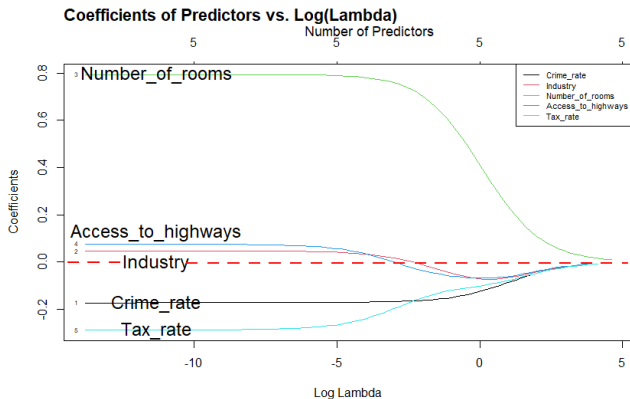
# Coefficients vs. $\text{Log}(\lambda)$

- Recall that without any regularisation, the coefficient of the "Access to highways", is positive, which may be problematic.
- For the coefficient of "Access to highways":
  - When  $\text{Log}(\lambda)$  increases from  $-12$  to  $-3$ , it gradually decreases to 0.
  - When  $\text{Log}(\lambda)$  increases from  $-3$  to 0, it becomes negative, and shifts closer to that of "Tax rate".
  - When  $\text{Log}(\lambda)$  increases from 0 to 5, the coefficients of both predictors remain similar, and they eventually approach 0.



# Coefficients vs. $\text{Log}(\lambda)$

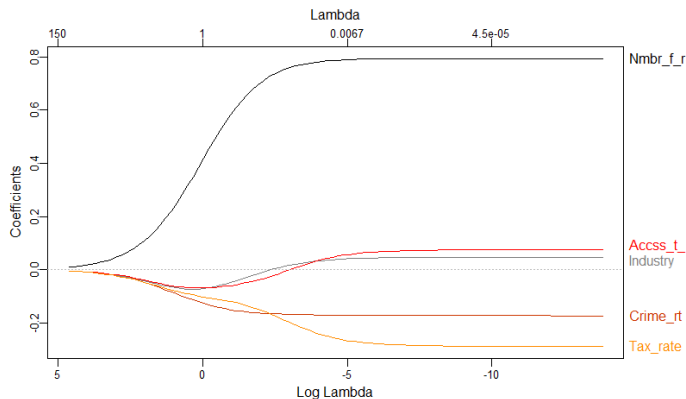
- In general, as  $\lambda$  increases, the coefficients of the correlated predictors tend to shift towards each other, until they become similar.
- This may explain a bit on how Ridge Regression deals with Multicollinearity.



## Coefficients vs. $\text{Log}(\lambda)$

- We can also use the “`plot_glmnet()`” function from the “`plotmo`” package, to generate a similar plot.

```
plot_glmnet(ridge_model)
```



# Introduction to the `cv.glmnet()` Function

## Cross Validation: `cv.glmnet()`

```
set.seed(123)
cv_ridge <- cv.glmnet(x, y, alpha = 0, type.measure = "mse")
```

The inputs of “`cv.glmnet`” include:

- `x` is a data matrix of predictor variables.
- `y` is the dependent variable.
- “`type.measure`” indicates the type of error metric that you wish to compute. It can be “MSE”, “MAE” or some other metrics.
- Alpha has the same meaning as that in “`glmnet()`”.
- `Alpha = 0`, means the models are trained for Ridge Regression.
- The default setting of “`cv.glmnet`” is to run the 10-fold Cross Validation. We can change it to any other  $K$ -fold Cross Validation, by setting “`nfolds = K`” inside the “`cv.glmnet`” function.

## Cross Validation: `cv.glmnet()`

```
cv_ridge
```

```
Call: cv.glmnet(x = train.x, y = train.y, type.measure = "mse", alpha = 0)
```

```
Measure: Mean-Squared Error
```

	Lambda	Index	Measure	SE	Nonzero
min	0.0844	100	0.4327	0.1176	5
1se	0.9478	74	0.5458	0.1316	5

```
cv_ridge$lambda.min
```

```
[1] 0.0843774
```

- "lambda.min" is the lambda at which the smallest MSE is achieved.

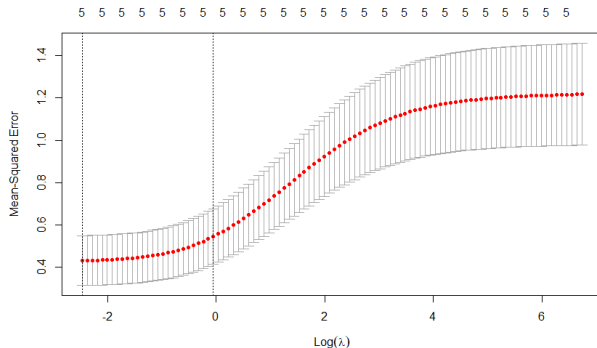
```
cv_ridge$lambda.1se
```

```
[1] 0.9478316
```

- "lambda.1se" is the largest lambda at which the MSE is within one standard error of the smallest MSE.

# Cross Validation: $\lambda_{\min}$ and $\lambda_{1se}$

- `plot(cv_ride)`

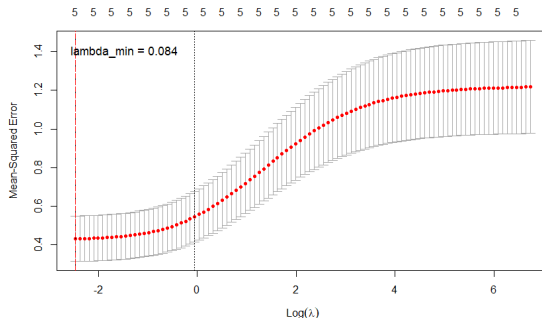


- The red dots represent the mean Cross Validation error.
- The error bar, extending from the red dot, represents the standard error of the MSE estimate.



# Cross Validation: $\lambda_{\min}$ and $\lambda_{1se}$

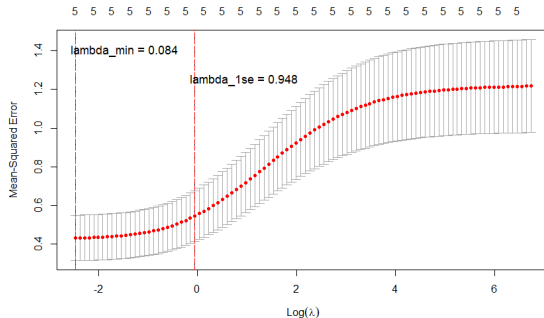
- `plot(cv_ride)`



- The first vertical line indicates where  $\text{Log}(\lambda_{\min})$  lies, and we achieve the least Cross Validation MSE at such  $\lambda$ .

# Cross Validation: $\lambda_{\min}$ and $\lambda_{1se}$

- `plot(cv_ride)`



- The first vertical line indicates where  $\text{Log}(\lambda_{\min})$  lies, and we achieve the least Cross Validation MSE at such  $\lambda$ .
- The other vertical line indicates where  $\text{Log}(\lambda_{1se})$  lies. The corresponding model is the most regularised model whose Cross Validation MSE is within one standard error of the least MSE.

## Analyse: Build the Final Ridge Regression Model

- Let us train the final Ridge Regression model, using the lambda.min.

```
glm_Ridge <- glmnet(train.x, train.y, alpha = 0, lambda =  
  cv_ridge$lambda.min)  
t(coef(glm_Ridge))
```

	(Intercept)	Crime_rate	Industry	Number_of_rooms	Access_to_highways	Tax_rate
s0	-3.602963	-0.1653398	0.00360547	0.7347811	-0.0207086	-0.1718418

- "Number of rooms" is the most important predictor, since its standardised coefficient, 0.735, has the highest absolute value among all the coefficients.
- "Industry" and "Access to highways" are the least important predictors, with the standardised coefficients close to zero.
- This reveals how Ridge Regression deals with Multicollinearity, by shrinking the coefficients of some predictors towards zero.
- "Crime rate" and "Tax rate" are negatively associated with "Price".

## Analyse: Evaluating the Final Ridge Regression Model

- We have designed the function, “eval\_results”, to evaluate MSE, MAE, RMSE and MAPE.

```
eval_results <- function(fit, true) {  
  data.frame(  
    MSE = MSE(fit, true),  
    MAE = MAE(fit, true),  
    RMSE = RMSE(fit, true),  
    MAPE = MAPE(fit, true),  
  )  
}
```

```
fit <- predict(glm_Ridge, train.x)  
true <- train.y  
summary_Ridge_train <- eval_results(fit, true)
```

```
fit <- predict(glm_Ridge, test.x)  
true <- test.y  
summary_Ridge_test <- eval_results(fit, true)
```

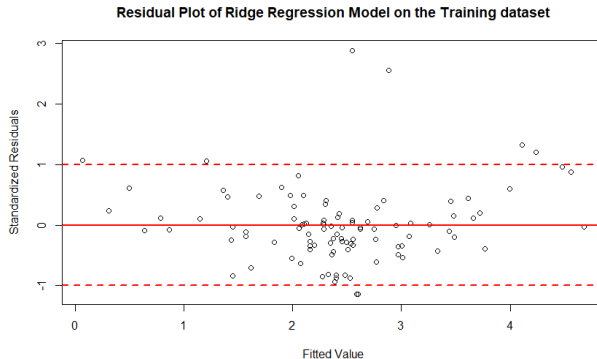
## Analyse: Evaluating the Final Ridge Regression Model

	MSE	MAE	RMSE	MAPE
Ridge Train	0.391	0.423	0.625	0.208
Ridge Test	0.461	0.439	0.679	0.222

From the summary table, we can see:

- The error metrics are consistently higher on the test dataset, compared with those on the training dataset.
- In general, it is expected that the model will perform better on the small training dataset, and have a higher error rate when being applied to the larger unseen test dataset.
- We will show in another video that the Ridge Regression model is performing better than the MLR model.

# Analyse: Residual Plots



- For the training dataset, the standardised residuals appear evenly scattered around the center line of zero.
- Most of the standardised residuals are between -1 and 1.

# Apply: Make Predictions

- Suppose the target house is located at some area, under the following conditions.

```
new_data
```

	Crime_rate	Industry	Number_of_rooms	Access_to_highways	Tax_rate
1	0.00632	2.31	6.575	1	296

- We first need to standardise the data using the scaler vector, and then transform the data into a data matrix.

```
new_x <- data.matrix(new_data/  
scaler)
```



Source: <https://www.qlik.com/blog/essential-steps-to-making-better-data-informed-decisions>

# Apply: Make Predictions

- Next, we use the final Ridge Regression model to predict the standardised median price, and multiply it by the standard deviation of Price.

```
predict(glm_Ridge, new_x) *  
  scaler[6]
```

s0

1 27.31479



Source: <https://www.qlik.com/blog/essential-steps-to-making-better-data-informed-decisions>



# Announce: Decide and Communicate

- Mr. Tan predicts that the median price of the new target house is 273 thousand dollars.
- The standardised coefficients suggest:
  - ▶ The top 3 most important price-predictors are “Number of rooms”, “Crime rate” and “Tax rate”.
  - ▶ The other two predictors, “Industry” and “Access to highways”, are not important for predicting the housing price.



Source: <https://www.qlik.com/blog/essential-steps-to-making-better-data-informed-decisions>

# Assess: Further Improvement

- We notice that there are some outliers spotted in the residual plots.
- We can study these outlier observations, and discuss whether they should be excluded.
- It will help us gain insights into the topic of predicting the housing price.



Source: <https://www.qlik.com/blog/essential-steps-to-making-better-data-informed-decisions>

# Features of the Ridge Regression Models

# Features of the Ridge Regression Models

Let us summarise some features of the Ridge Regression models:

- ① By introducing a penalty term to the cost function, the Ridge Regression model has the effect of shrinking the coefficients of predictors towards zero.
- ② By reducing the magnitudes of the coefficients, Ridge Regression helps solve Multicollinearity.
- ③ The regularisation parameter,  $\lambda$ , controls the amount of regularisation, and regularisation controls the amount of bias and variance.
- ④ With the bias-variance trade off, the optimal Ridge Regression model will have slightly higher bias on the training dataset, but in return, it will have smaller variance, and better performance, when being generalised to any unseen dataset.

# Summary

# Summary





## We have learnt to:

- ▶ Understand how a Ridge Regression model works.
- ▶ Understand how the regularisation parameter,  $\lambda$ , affects the Ridge Regression model coefficients.
- ▶ Learn to apply the “`cv.glmnet()`” function to select the optimal  $\lambda$  value.
- ▶ Learn to apply the “`glmnet()`” function to train a Ridge Regression model, and evaluate its performance.

## In the next video,

We will introduce the LASSO Regression, and learn how to implement it in R.

# References

-  Wessel N. van Wieringen (2021), *Lecture notes on ridge regression*
-  NCSS Documentation, *Ridge Regression*
-  Hastie, Qian, and Tay (2021), *An Introduction to glmnet*  
<https://glmnet.stanford.edu/articles/glmnet.html>
-  Dataset: the Boston Housing Dataset  
<https://www.cs.toronto.edu/delve/data/boston/bostonDetail.html>