Predicting Labour Wages using Ridge and Lasso Regression

Ben Roshan

- Ridge and Lasso Regression
    - R Markdown
- Read and Understand the data
- Data Pre-processing
    - Train-Test Split
    - Standardize the Data
    - Dummify the Data
    - Get the data into a compatible format
- Hyper-parameter Tuning
    - Choosing a lambda for Lasso Regression
    - Choosing a lambda for Ridge Regression
- Building The Final Model
    - Building the Final Lasso Regression Model
    - Building the Final Ridge Regression Model
- Model Performance Evaluation
    - Lasso Regression Model Metrics
    - Ridge Regression Model Metrics

Ridge and Lasso Regression

$$RSS(\beta) + \lambda \sum_{j=1}^{p} \beta_j^2$$

$$RSS(\beta) + \lambda \sum_{j=1}^{p} |\beta_j|$$

R Markdown

#Remove warnings

```
options(warn=-1)
```

#Reading libraries

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
#install.packages("glmnet", repos = "http://cran.us.r-project.org")
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.0-2
```

```
library(DMwR)
```

```
## Loading required package: grid
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

#Removing env variables

```
rm(list=ls(all=TRUE))
```

#Setting working directory

```
getwd()
```

```
## [1] "C:/Users/Ben Roshan/Documents"
```

```
setwd("C:/Users/Ben Roshan/Documents")
```

Read and Understand the data

```
labourincome=read.csv(file='labour_income.csv',header=T)
summary(labourincome)
```

```
##      wages         education          age            sex
##  Min.   : 2.30   Min.   : 0.00   Min.   :16.0   Length:3987
##  1st Qu.: 9.25   1st Qu.:12.00   1st Qu.:28.0   Class :character
##  Median :14.13   Median :13.00   Median :36.0   Mode  :character
##  Mean   :15.54   Mean   :13.34   Mean   :37.1
##  3rd Qu.:19.72   3rd Qu.:15.10   3rd Qu.:46.0
##  Max.   :49.92   Max.   :20.00   Max.   :69.0
##    language
##  Length:3987
##  Class :character
##  Mode  :character
##
##
##
```

```
str(labourincome)
```

```
## 'data.frame':    3987 obs. of  5 variables:
##  $ wages    : num  10.6 11 17.8 14 8.2 ...
##  $ education: num  15 13.2 14 16 15 13.5 12 14 18 11 ...
##  $ age      : int  40 19 46 50 31 30 61 46 43 17 ...
##  $ sex      : chr  "Male" "Male" "Male" "Female" ...
##  $ language : chr  "English" "English" "Other" "English" ...
```

Data Pre-processing

Train-Test Split

- Split the data into train and test

```
set.seed(007)
train_rows <- sample(x=seq(1,nrow(labourincome),1),size=0.7*nrow(labourincome))
train_data <- labourincome[train_rows,]
test_data <- labourincome[-train_rows,]
```

Standardize the Data

- Standardize the continuous independent variables

```
std_obj <- preProcess(x = train_data[, !colnames(train_data) %in% c("wages")],method = c("cen
ter", "scale"))


train_std_data <- predict(std_obj,train_data)

test_std_data <- predict(std_obj,test_data)
```

Dummify the Data

- Use the dummyVars() function from caret to convert sex and age into dummy variables

```
dummy_obj <- dummyVars(~.,train_std_data)

train_dummy_data <- as.data.frame(predict(dummy_obj,train_std_data))

test_dummy_data <- as.data.frame(predict(dummy_obj,test_std_data))
```

Get the data into a compatible format

- The functions we will be using today from the glmnet package expect a matrix as an input and not our familiar formula structure, so we need to convert our dataframes into a matrix

```
X_train <- as.matrix(train_dummy_data[,-1])
y_train <- as.matrix(train_dummy_data[,1])
X_test <- as.matrix(test_dummy_data[,-1])
y_test <- as.matrix(test_dummy_data[,1])
```

Hyper-parameter Tuning

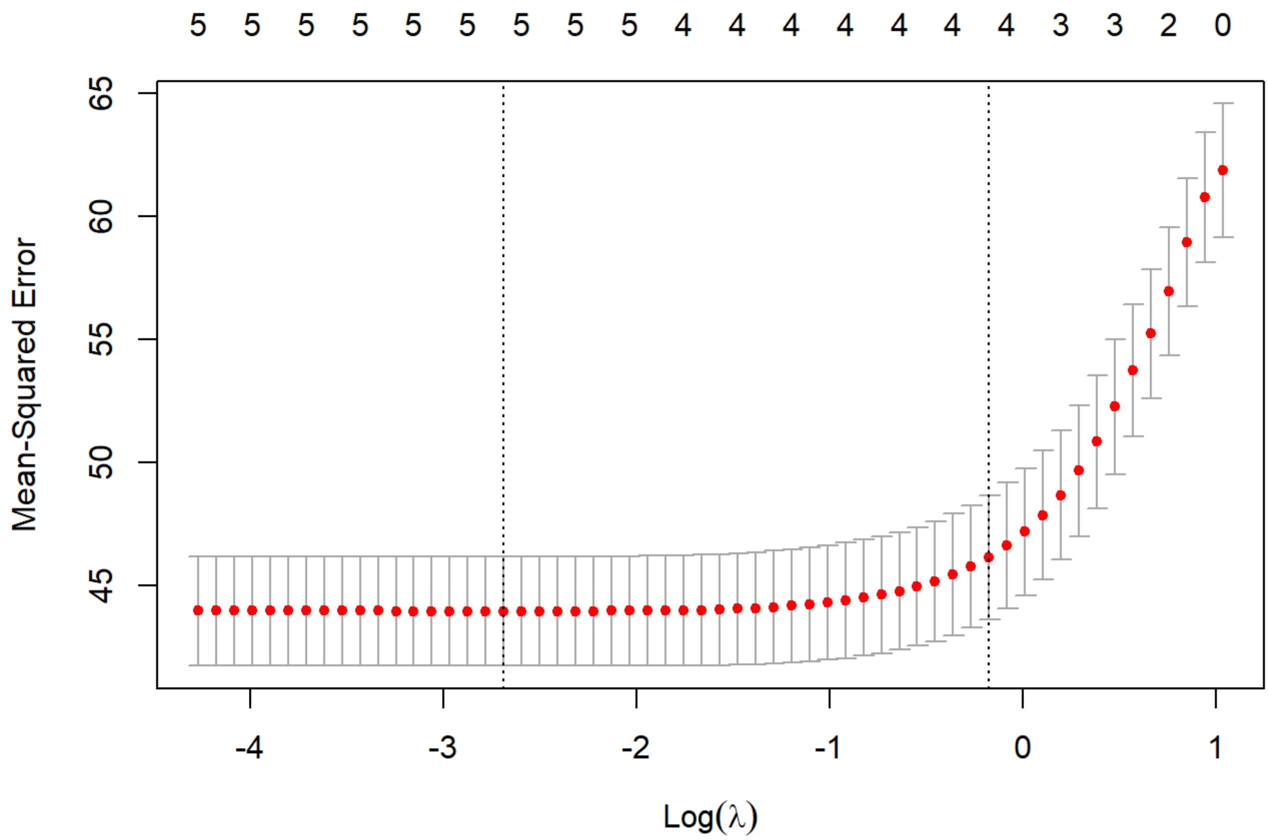
- Choose an optimal lambda value for the ridge and lasso regression models by using cross validation

Choosing a lambda for Lasso Regression

- The alpha value is 1 for lasso regression

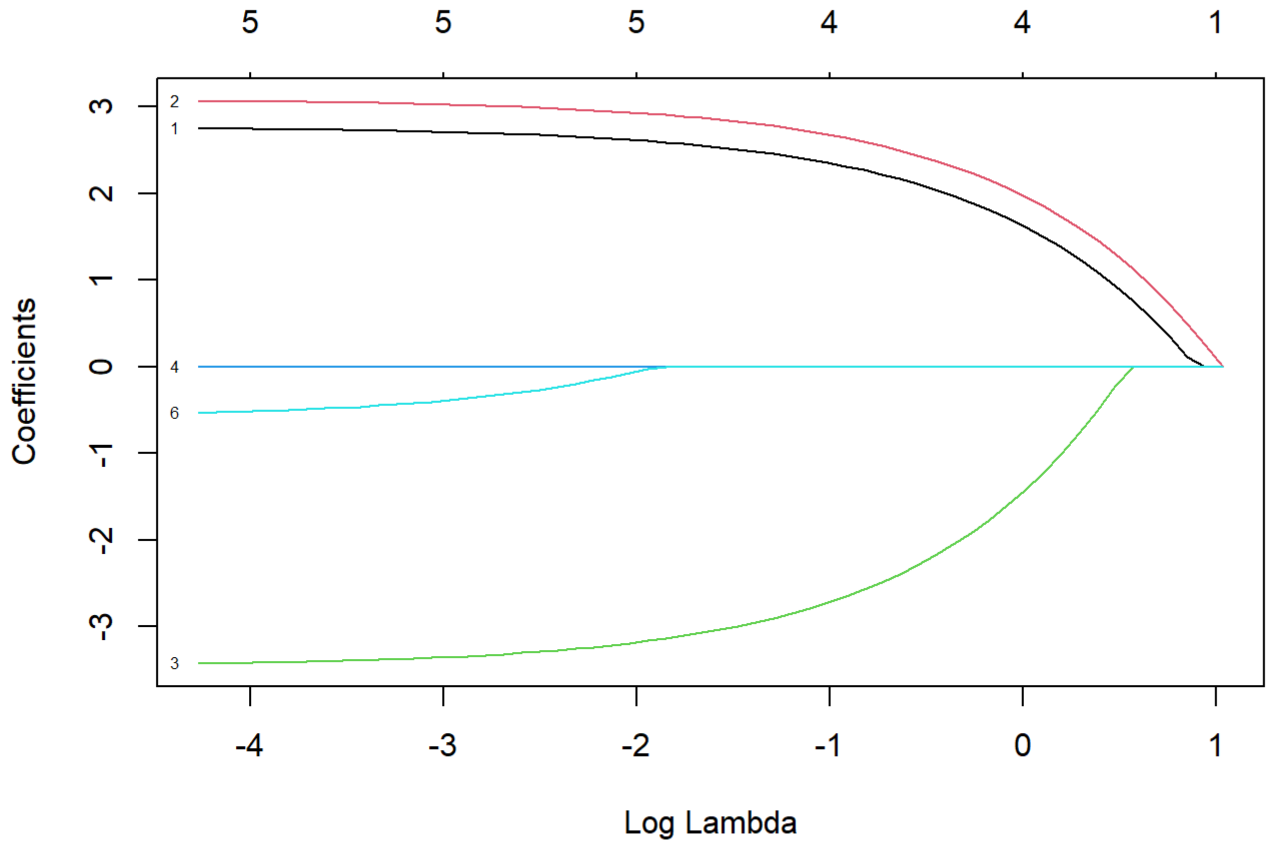
```
cv_lasso <- cv.glmnet(X_train,y_train,alpha=1,type.measure ="mse",nfolds=4 )

plot(cv_lasso)
```



- The object returned form the call to cv.glmnet() function, contains the lambda values of importance
- The coefficients are accessible calling the coef() function on the cv_lasso object

```
plot(cv_lasso$glmnet.fit,xvar="lambda",label=TRUE)
```




```
print(cv_lasso$lambda.min)
```

```
## [1] 0.06803175
```

```
coef(cv_lasso)
```

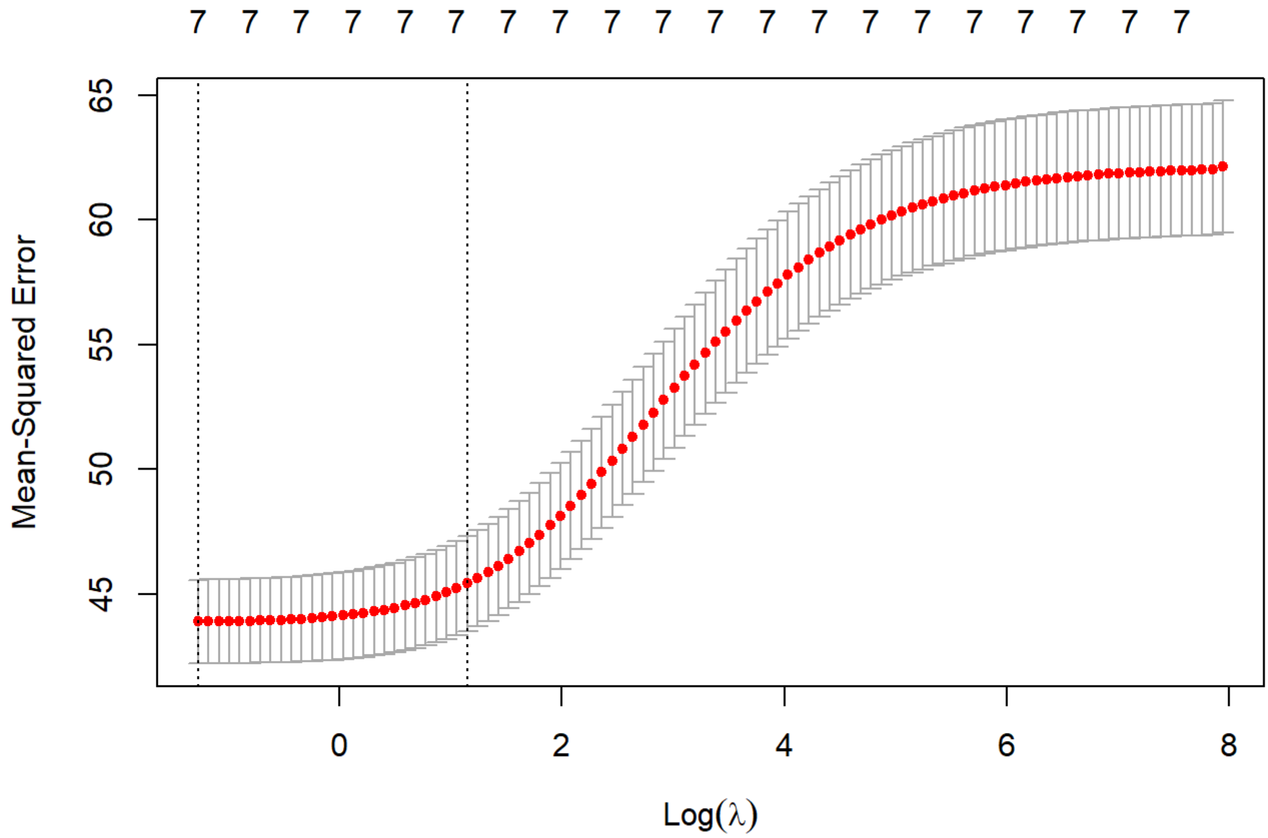
```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                             1
## (Intercept)      1.640412e+01
## education        1.812587e+00
## age              2.153717e+00
## sexFemale       -1.766271e+00
## sexMale          8.259089e-14
## languageEnglish  .
## languageFrench   .
## languageOther    .
```

Choosing a lambda for Ridge Regression

- The alpha value is 0 for ridge regression

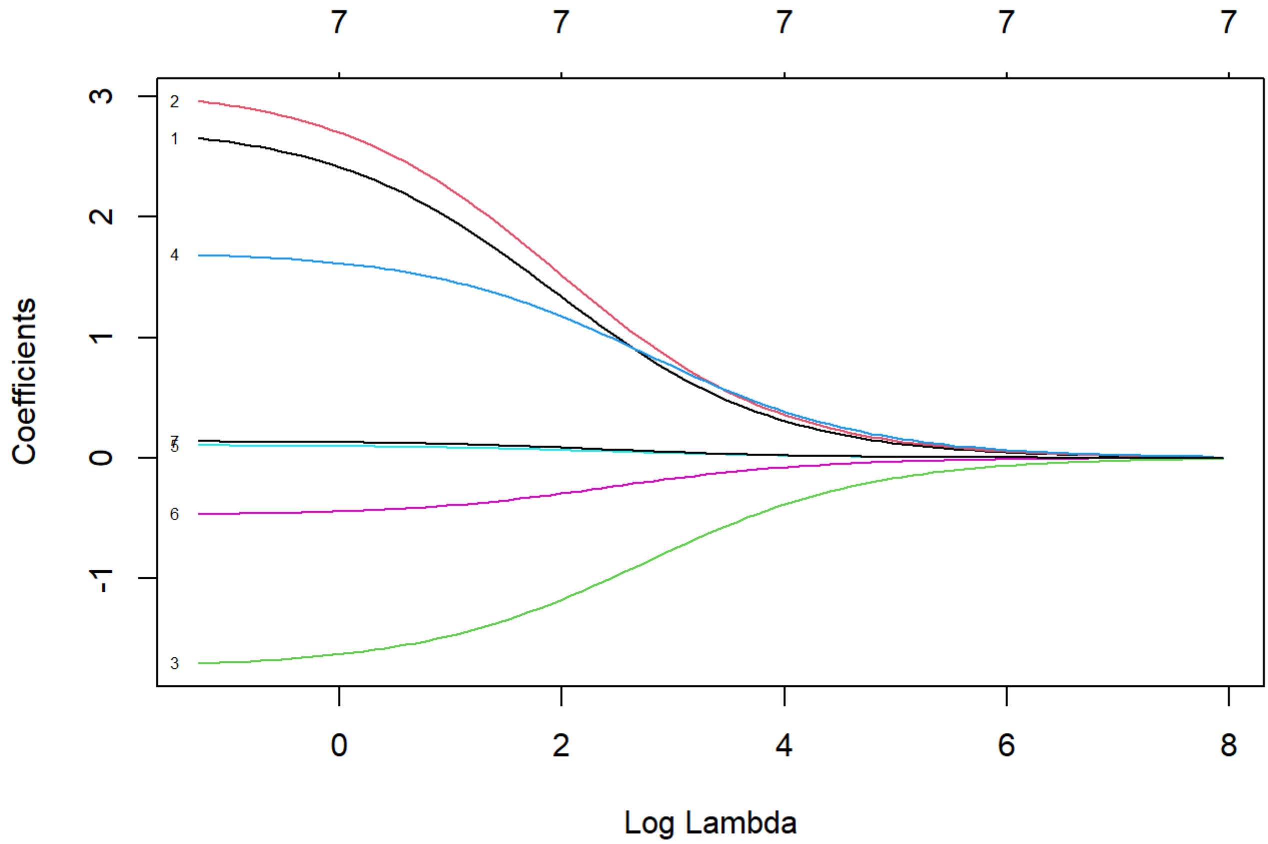
```
cv_ridge <- cv.glmnet(X_train,y_train,alpha=0,type.measure ="mse",nfolds=4 )

plot(cv_ridge)
```



- We can access the lambda and the coefficients as we did before

```
plot(cv_ridge$glmnet.fit,xvar="lambda",label=TRUE)
```




```
print(cv_ridge$lambda.min)
```

```
## [1] 0.281108
```

```
coef(cv_ridge)
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                             1
## (Intercept)      15.46625297
## education         1.89683861
## age               2.13566648
## sexFemale        -1.44034175
## sexMale           1.43636353
## languageEnglish   0.08263341
## languageFrench   -0.38150197
## languageOther     0.11172119
```

Building The Final Model

- By using the optimal lambda values obtained above, we can build our ridge and lasso models

Building the Final Lasso Regression Model

```
lasso_model <- glmnet(X_train,y_train,lambda=cv_lasso$lambda.min,alpha=1)

coef(lasso_model)
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                            s0
## (Intercept)      1.719687e+01
## education        2.689314e+00
## age              3.006946e+00
## sexFemale       -3.314947e+00
## sexMale          2.811204e-13
## languageEnglish  .
## languageFrench  -3.176708e-01
## languageOther    .
```

- Use the model to predict on test data

```
pred_lasso <- predict(lasso_model,X_test)
```

Building the Final Ridge Regression Model

```
ridge_model <- glmnet(X_train,y_train,lambda=cv_ridge$lambda.min,alpha=0)

coef(ridge_model)
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept)     15.5071089
## education        2.6575550
## age              2.9648092
## sexFemale       -1.7573840
## sexMale          1.6383518
## languageEnglish  0.1050676
## languageFrench  -0.4659695
## languageOther    0.1378338
```

- Use the model to predict on test data

```
pred_ridge <- predict(ridge_model,X_test)
```

Model Performance Evaluation

Lasso Regression Model Metrics

```
regr.eval(trues=y_test,preds=pred_lasso)
```

```
##       mae       mse      rmse      mape
##  4.920026 43.191036  6.571989  0.380545
```

Ridge Regression Model Metrics

```
regr.eval(trues=y_test,preds=pred_ridge)
```

```
##        mae        mse       rmse       mape
##  4.9280828 43.2711221  6.5780789  0.3814506
```