# Houses Prices Predictions with Regression-Regularization

Code

This is a R Notebook that utilizes Multiple Regression Models to predict House prices.

Hide

```
library(doSNOW) # Parallel
```

```
Loading required package: iterators
package 愀牶iterators愀牷 was built under R version 3.4.4Loading required package: snow
```

Hide

```
RFRun<-TRUE
# Delete all graphs
#
#if(!is.null(dev.list())) dev.off()
node_name<-Sys.info()["nodename"]
# Get Current time
sTime<-Sys.time()
print(sTime)
```

```
[1] "2018-07-30 20:18:54 CDT"
```

Hide

```
par(mfrow=c(2,2))
cl <- makeCluster(10, type = "SOCK") #10 parallel processes-RStudio running at the sam
e time
# Register cluster so that caret will know to train in parallel.
registerDoSNOW(cl)
#
# Lasso/Ridge/Elnet with caret Package
#
#MODELS
# set up caret model training parameters
# model specific training parameter
CARET.TRAIN.CTRL <- trainControl(method="repeatedcv",
                                 number=10,
                                 repeats=5,
                                 savePredictions='final',
                                 classProbs = FALSE,
                                 verboseIter=FALSE)
# test out Ridge regression model
lambdas <- seq(1,0,-0.001)
# train model
set.seed(12345)  # for reproducibility
model_ridge <- train(x=X_train,y=y,
                     method="glmnet",
                     metric="RMSE",
                     maximize=FALSE,
                     trControl=CARET.TRAIN.CTRL,
                     tuneGrid=expand.grid(alpha=0, # Ridge regression
                                          lambda=lambdas))
rid.rmse<-mean(model_ridge$resample$RMSE)
rid.mse<-mean(model_ridge$resample$MAE)
model_elnet <- train(x=X_train,y=y,
                     method="glmnet",
                     metric="RMSE",
                     maximize=FALSE,
                     trControl=CARET.TRAIN.CTRL,
                     tuneGrid=expand.grid(alpha=0.5, # Elastic Net regression
                                          lambda=lambdas))
```

```
There were missing values in resampled performance measures.
```

Hide

```
eln.rmse<-mean(model_elnet$resample$RMSE)
eln.mse<-mean(model_elnet$resample$MAE)
# test out Lasso regression model
# train model
model_lasso <- train(x=X_train,y=y,
                     method="glmnet",
                     metric="RMSE",
                     maximize=FALSE,
                     trControl=CARET.TRAIN.CTRL,
                     tuneGrid=expand.grid(alpha=1,  # Lasso regression
                                          lambda=c(1,0.1,0.05,0.01,seq(0.009,0.001,-0.
001),
                                                   0.00075,0.0005,0.0001)))
```

```
There were missing values in resampled performance measures.
```

Hide

```
model_lasso
```

```
glmnet

1408 samples
 274 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 1268, 1267, 1268, 1267, 1267, 1268, ...
Resampling results across tuning parameters:

  lambda   RMSE       Rsquared   MAE
  0.00010  0.1139896  0.9113574  0.07780207
  0.00050  0.1116611  0.9146520  0.07588912
  0.00075  0.1114456  0.9148937  0.07539103
  0.00100  0.1114419  0.9148354  0.07512915
  0.00200  0.1100124  0.9169273  0.07394089
  0.00300  0.1094834  0.9177762  0.07361829
  0.00400  0.1098758  0.9173445  0.07398758
  0.00500  0.1107748  0.9161997  0.07477825
  0.00600  0.1118068  0.9149130  0.07577305
  0.00700  0.1128336  0.9136488  0.07671154
  0.00800  0.1139706  0.9122169  0.07776817
  0.00900  0.1152171  0.9106242  0.07892971
  0.01000  0.1164318  0.9090802  0.08003120
  0.05000  0.1578359  0.8594955  0.11102653
  0.10000  0.2032363  0.8155816  0.14816451
  1.00000  0.3809305        NaN  0.29864528

Tuning parameter 'alpha' was held constant at a value of 1
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 1 and lambda = 0.003.
```

Hide

```
las.rmse<-mean(model_lasso$resample$RMSE, rm.na=TRUE)
las.mse<-mean(model_lasso$resample$MSE, rm.na=TRUE)
```

```
argument is not numeric or logical: returning NA
```

Hide

```
if (RFRun==TRUE) {
# train model
  model_rf <- train(x=X_train,y=y,
               #      method="rf",
                    method="rf",
                     metric="RMSE",
                     maximize=FALSE,
                     trControl=CARET.TRAIN.CTRL)
  model_rf
  rf.rmse<-mean(model_rf$resample$RMSE, rm.na=TRUE)
  rf.mse<-mean(model_rf$resample$MSE, rm.na=TRUE)
}
```

```
argument is not numeric or logical: returning NA
```

Hide

```
stopCluster(cl)
print (paste("Lasso RMSE : ",las.rmse))
```

```
[1] "Lasso RMSE :  0.109483403599545"
```

Hide

```
print (paste("Ridge RMSE : ",rid.rmse))
```

```
[1] "Ridge RMSE :  0.113019403661244"
```

Hide

```
print (paste("Elnet RMSE : ",eln.rmse))
```

```
[1] "Elnet RMSE :  0.109237328976038"
```

Hide

```
if (RFRun==TRUE) {
  print (paste("RF RMSE : ",rf.rmse))
}
```

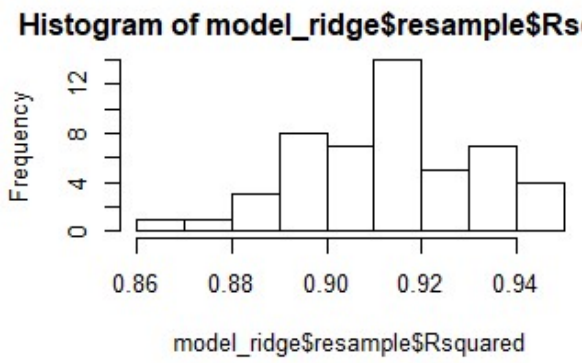
```
[1] "RF RMSE :  0.127804569408016"
```

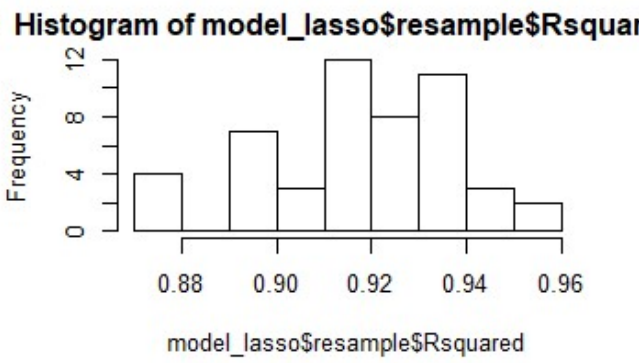
Hide

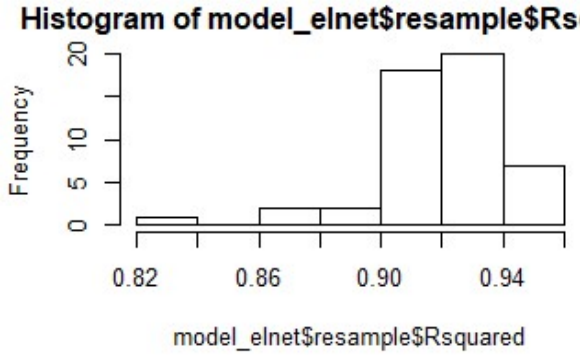
```
# transfor train to log+1
# LOG, SQRT, NONE
if (xrf=="LOG") {
    las.preds<-exp(predict(model_lasso,newdata=X_test)) - logC
    rid.preds<-exp(predict(model_ridge,newdata=X_test)) - logC
    eln.preds<-exp(predict(model_elnet,newdata=X_test)) - logC
    if (RFRun==TRUE) {
      rf.preds<-exp(predict(model_rf,newdata=X_test)) - logC
    }
} else if (xrf=="SQRT") {
    las.pred<-predict(model_lasso,newdata=X_test)
    las.preds<-las.pred^2
    rid.pred<-predict(model_ridge,newdata=X_test)
    rid.preds<-rid.pred^2
    eln.pred<-predict(model_elnet,newdata=X_test)
    eln.preds<-eln.pred^2
    if (RFRun==TRUE) {
      rf.pred<-predict(model_rf,newdata=X_test)
      rf.preds<-rf.pred^2
    }
} else {
    las.preds<-predict(model_lasso,newdata=X_test)
    rid.preds<-predict(model_ridge,newdata=X_test)
    eln.preds<-predict(model_elnet,newdata=X_test)
    if (RFRun==TRUE) {
      rf.preds<-predict(model_rf,newdata=X_test)
    }

}
hist(model_ridge$resample$Rsquared)
hist(model_lasso$resample$Rsquared)
```

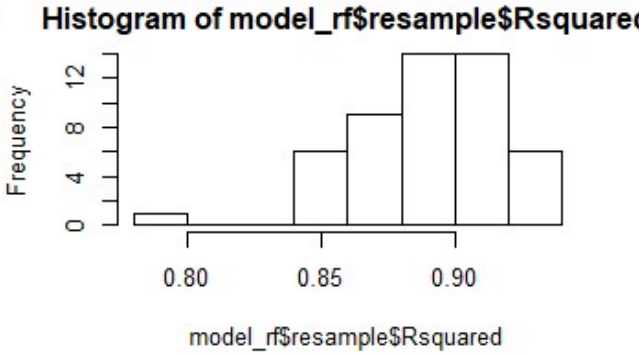
Hide

```
hist(model_elnet$resample$Rsquared)
hist(model_rf$resample$Rsquared)
```




Hide

```
eTime<-Sys.time()
print(paste0("Start Time: ",sTime))
```

```
[1] "Start Time: 2018-07-30 20:18:54"
```

Hide

```
print(paste0("End  Time: ",eTime))
```

```
[1] "End  Time: 2018-07-30 20:57:09"
```

Hide

```
print(paste0("Elapsed Time: ",eTime-sTime))
```

```
[1] "Elapsed Time: 38.251251967748"
```