STAT443 Project

Chan Shing Yee

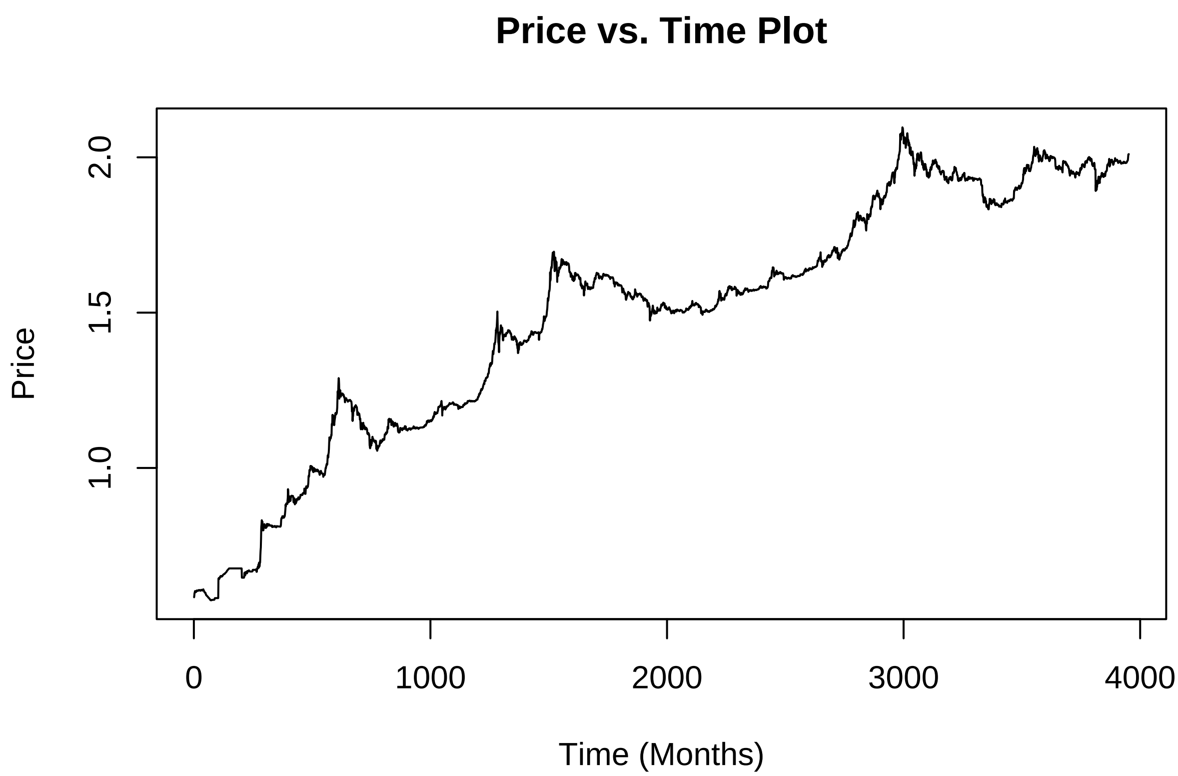
2024-04-04

Importing dataset

```
# data was taken everyday of the year
train <- read.csv("train.csv", header=TRUE)
train$date <- as.Date(train$date)
train$month <- as.factor(format(train$date, "%m"))

# start on the 285th day of 2009
Y <- train$price # raw values of Y
Y.training <- Y^0.075 # power values of Y (used for training)
Y.ts <- ts(Y.training, frequency=365, start=c(2009, 285))
tim <- 1:length(Y.training)
month.dummies <- model.matrix(~ month-1, data=train)
month.dummies <- month.dummies[,-ncol(month.dummies)]
```

Study trend

```
plot(x=tim, y=Y.ts, type="l",
     xlab="Time (Months)", ylab="Price",
     main="Price vs. Time Plot")
```



From the plot, it can be seen that there is an increasing trend over the years with some fluctuations. There is seasonal patterns of the price increasing and decreasing at some seasons.

Using regression with different degrees to fit the model

```
library(glmnet)
```

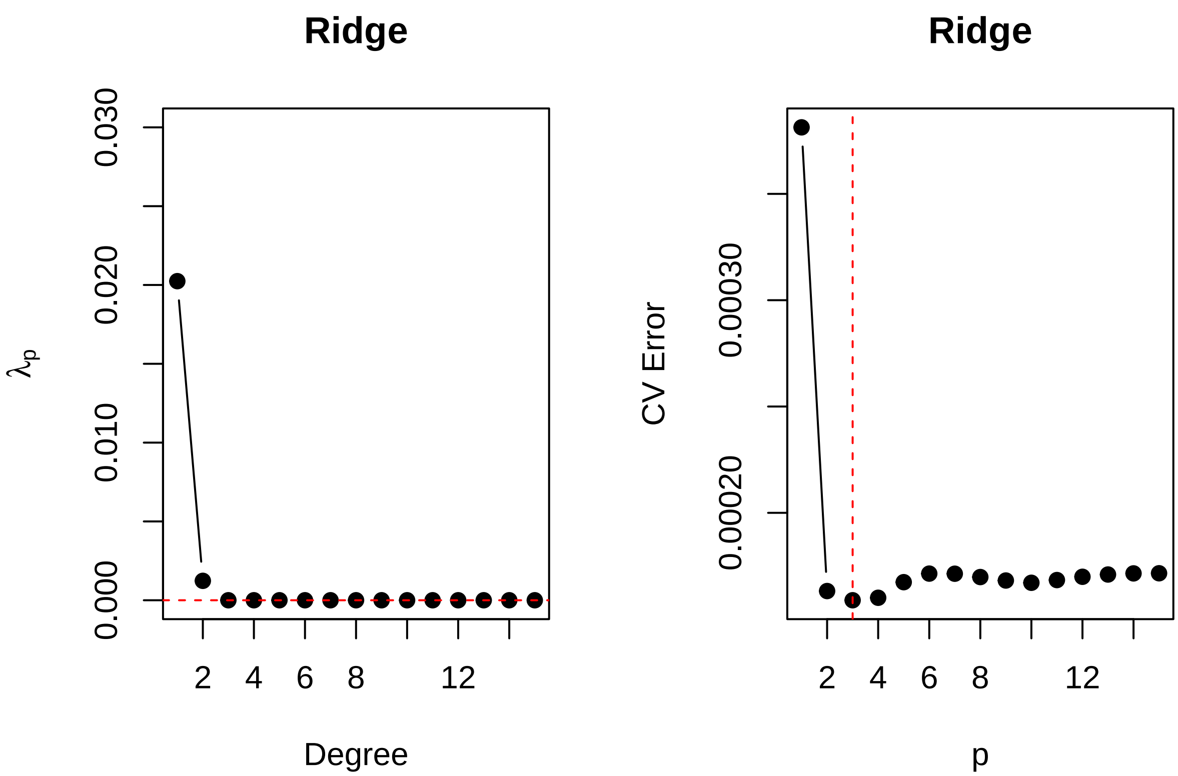
```
## Loading required package: Matrix

## Warning: package 'Matrix' was built under R version 4.2.3

## Loaded glmnet 4.1-7
```

```
# introduce month seasonality
max.p <- 15
X.training.max <- poly(tim, degree=max.p, raw=TRUE)[, 1:max.p]
Log.Lambda.Seq <- seq(-7, 3, by = 0.1)
Lambda.Seq <- c(0, exp(Log.Lambda.Seq))
p.sequence <- 1:max.p
CV.values.Ridge = OptimumLambda.Ridge = c()
CV.values.LASSO = OptimumLambda.LASSO = c()
CV.values.EN = OptimumLambda.EN = c()

for (p in p.sequence) {
  X.training <- cbind(X.training.max[, 1:p], month.dummies)
  if (p==1) { # need to add column of intercept
```

```
    X.training <- cbind(0, X.training)
  }
  set.seed(443)

  # Ridge Regression (alpha=0)
  CV.Ridge <- cv.glmnet(X.training, Y.training, lambda=Lambda.Seq,
                        alpha=0, nfolds=10)
  indx.lambda.1SE.Ridge <- which(round(CV.Ridge$lambda, 6) == round(CV.Ridge$lambda.1se, 6))
  CV.values.Ridge[p] <- CV.Ridge$cvsd[indx.lambda.1SE.Ridge]
  OptimumLambda.Ridge[p] <- CV.Ridge$lambda.1se

  # LASSO (alpha=1)
  CV.LASSO <- cv.glmnet(X.training, Y.training, lambda=Lambda.Seq,
                        alpha=1, nfolds=10)
  indx.lambda.1SE.LASSO <- which(round(CV.LASSO$lambda, 6) == round(CV.LASSO$lambda.1se, 6))
  CV.values.LASSO[p] <- CV.LASSO$cvsd[indx.lambda.1SE.LASSO]
  OptimumLambda.LASSO[p] <- CV.LASSO$lambda.1se

  # Elastic Net (alpha=0.5)
  CV.EN <- cv.glmnet(X.training, Y.training, lambda=Lambda.Seq,
                     alpha=0.5, nfolds=10)
  indx.lambda.1SE.EN <- which(round(CV.EN$lambda, 6) == round(CV.EN$lambda.1se, 6))
  CV.values.EN[p] <- CV.EN$cvsd[indx.lambda.1SE.EN]
  OptimumLambda.EN[p] <- CV.EN$lambda.1se
}
```

Plots for Ridge Regression

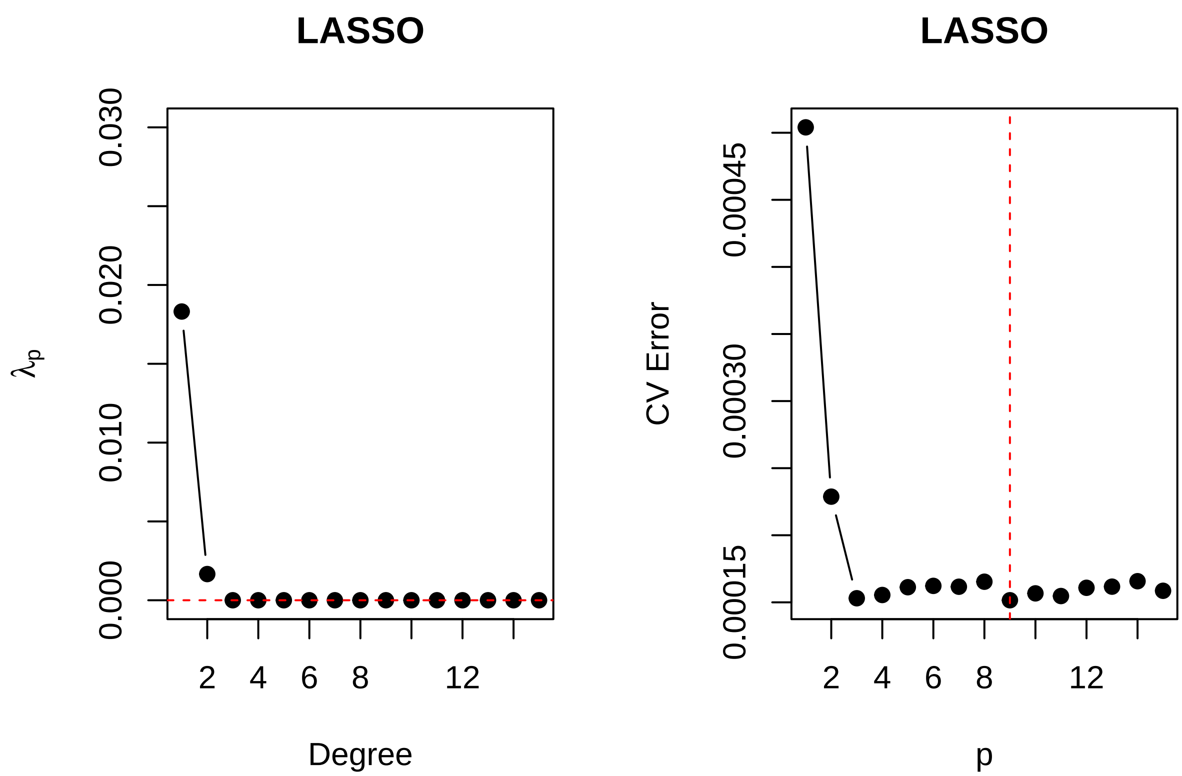
```
par(mfrow=c(1, 2))
plot(p.sequence, OptimumLambda.Ridge, type="b", pch=19,
     xlab="Degree", ylab=expression(lambda[p]), main="Ridge",
     ylim=c(0, 0.03))
abline(h=0, lty=2, col="red")
plot(p.sequence, CV.values.Ridge, type="b", pch=19,
     xlab="p", ylab="CV Error", main="Ridge")
optimum.p.Ridge <- which.min(CV.values.Ridge)
abline(v=optimum.p.Ridge, col="red", lty=2)
```



From the plot, optimum degree for ridge regression is 3.

Plots for LASSO

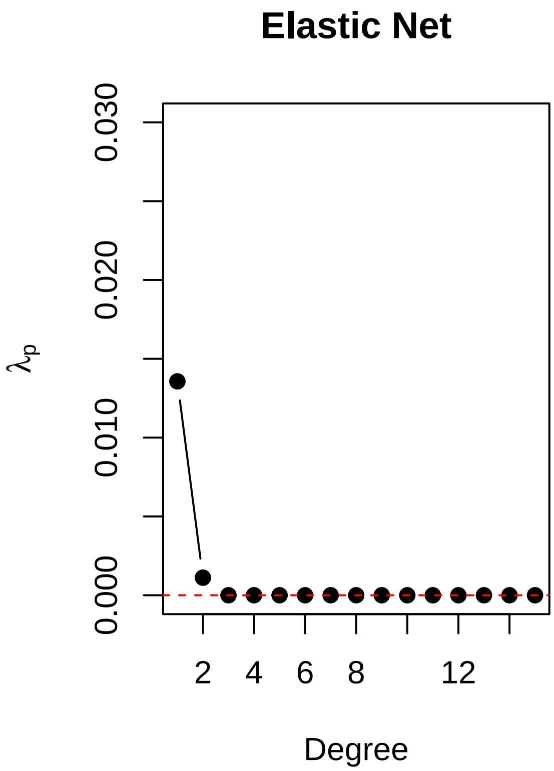
```
par(mfrow=c(1, 2))
plot(p.sequence, OptimumLambda.LASSO, type="b", pch=19,
     xlab="Degree", ylab=expression(lambda[p]), main="LASSO",
     ylim=c(0, 0.03))
abline(h=0, lty=2, col="red")
plot(p.sequence, CV.values.LASSO, type="b", pch=19,
     xlab="p", ylab="CV Error", main="LASSO")
optimum.p.LASSO <- which.min(CV.values.LASSO)
abline(v=optimum.p.LASSO, col="red", lty=2)
```

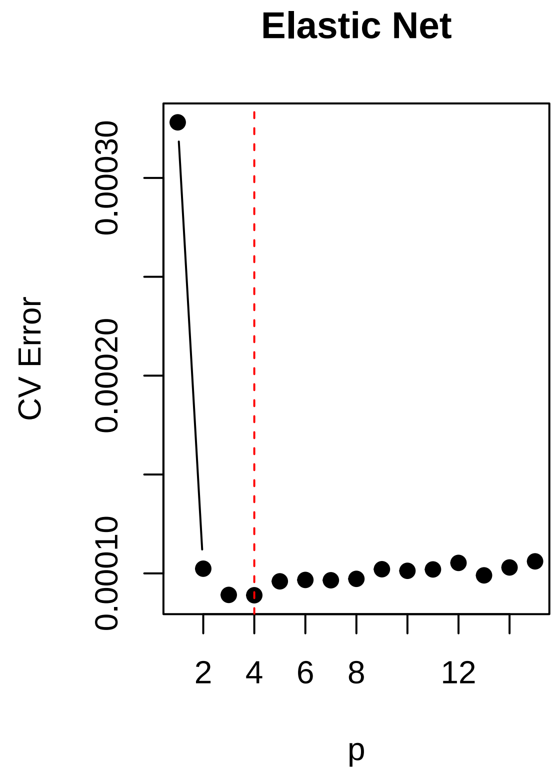


From the plot, optimum degree for LASSO regression is 9.

Plots for Elastic Net

```
par(mfrow=c(1, 2))
plot(p.sequence, OptimumLambda.EN, type="b", pch=19,
     xlab="Degree", ylab=expression(lambda[p]), main="Elastic Net",
     ylim=c(0, 0.03))
abline(h=0, lty=2, col="red")
plot(p.sequence, CV.values.EN, type="b", pch=19,
     xlab="p", ylab="CV Error", main="Elastic Net")
optimum.p.EN <- which.min(CV.values.EN)
abline(v=optimum.p.EN, col="red", lty=2)
```





From the plot, optimum degree for Elastic Net is 4.

Test set

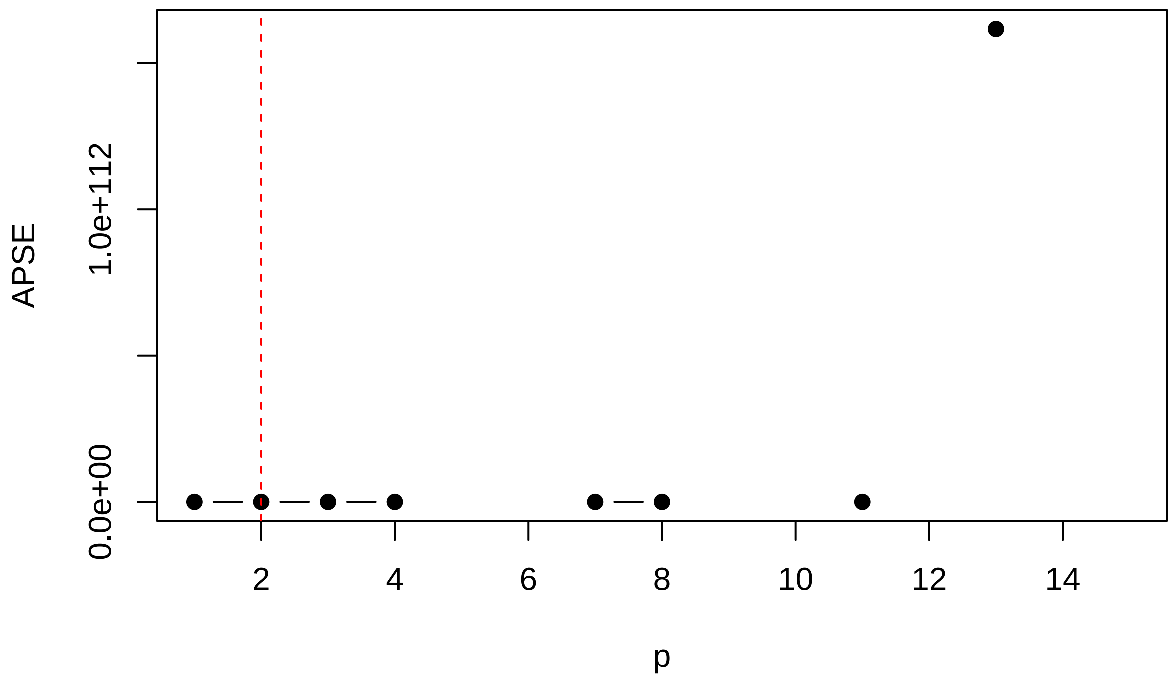
```
test <- read.csv("test.csv", header=TRUE)
test$date <- as.Date(test$date)
test$month <- as.factor(format(test$date, "%m"))

test.price <- test$price
Y.test <- test.price^0.075
tim.test <- (nrow(train)+1):(nrow(train)+length(test.price))
month.test <- test$month
```

Fitting classical non-regularized linear regression

Fit the model on training set and compute APSE on test set

```
p.sequence <- 1:max.p
APSE.LS <- c()
X.test.max <- poly(tim.test, degree=max.p, raw=TRUE)[, 1:max.p]

for (p in p.sequence) {
  X.training <- cbind(X.training.max[, 1:p], month.dummies)
  model <- lm(Y.training ~ X.training)

  X.test <- X.test.max[,1:p]
```

```r
  month.dummies.test <- model.matrix(~ month.test -1)
  month.dummies.test <- month.dummies.test[,-ncol(month.dummies.test)]
  new.data <- cbind(1, X.test, month.dummies.test)
  predict <- new.data %*% coef(model)
  APSE.LS[p] <- mean((test.price - predict^(1/0.075))^2)
}

plot(p.sequence, APSE.LS, pch=19, type="b",
     xlab="p", ylab="APSE")
optimum.p.LS <- which.min(APSE.LS)
abline(v=optimum.p.LS, col="red", lty=2)
```



From the plot, optimum degree for classical linear regression is 2.

Comparing all models

```r
APSES <- list()
month.dummies.test <- model.matrix(~ month -1, data=test)
month.dummies.test <- month.dummies.test[,-ncol(month.dummies.test)]

X.training.lm <- cbind(X.training.max[,1:11], month.dummies)
X.test.lm <- cbind(X.test.max[,1:11], month.dummies.test)
Classic.lm <- lm(Y.training ~ X.training.lm) # train the model
newdata.train.lm <- cbind(1, X.training.lm)
newdata.test.lm <- cbind(1, X.test.lm)
lm.predict.train <- newdata.train.lm %*% coef(Classic.lm)
lm.predict.test <- newdata.test.lm %*% coef(Classic.lm)
APSES["Classical LM"] <- mean((test.price - lm.predict.test^(1/0.075))^2)
```

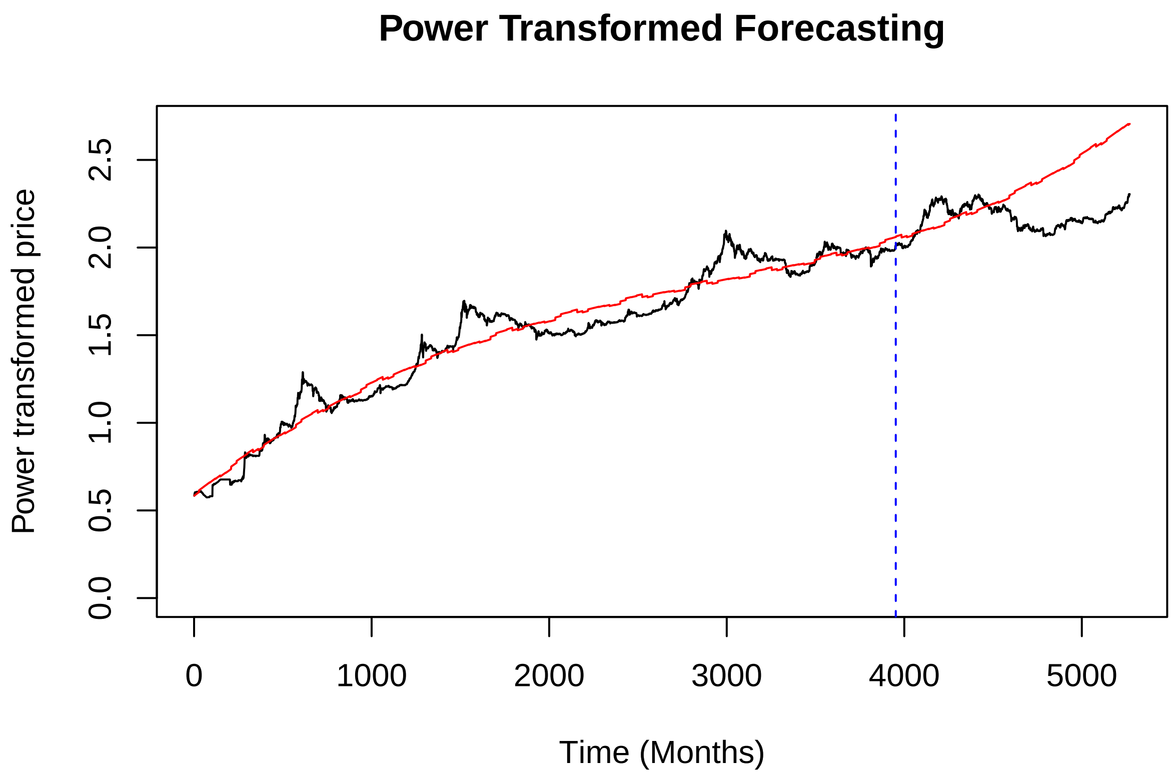
```r
set.seed(443)
X.training.Ridge <- cbind(X.training.max[, 1:3], month.dummies)
Ridge.model <- glmnet(X.training.Ridge, Y.training, alpha=0)
X.test.Ridge <- X.test.max[,1:3]
Ridge.predict <- predict(Ridge.model,
                         newx=cbind(X.test.Ridge,
                                    month.dummies.test))
APSES["Ridge"] <- mean((test.price - Ridge.predict^(1/0.075))^2)

X.training.LASSO <- cbind(X.training.max[, 1:9], month.dummies)
LASSO.model <- glmnet(X.training.LASSO, Y.training,
                      alpha=1)
X.test.LASSO <- X.test.max[,1:9]
LASSO.predict <- predict(LASSO.model,
                         newx=cbind(X.test.LASSO,
                                    month.dummies.test))
APSES["LASSO"] <- mean((test.price - LASSO.predict^(1/0.075))^2)

X.training.EN <- cbind(X.training.max[, 1:4], month.dummies)
EN.model <- glmnet(X.training.EN, Y.training, alpha=0.5)
X.test.EN <- X.test.max[,1:4]
EN.predict <- predict(EN.model,
                      newx=cbind(X.test.EN,
                                 month.dummies.test))
APSES["Elastic Net"] <- mean((test.price - EN.predict^(1/0.075))^2)

APSES
```

```
## $`Classical LM`
## [1] 1.660787e+87
##
## $Ridge
## [1] 17262067212
##
## $LASSO
## [1] NaN
##
## $`Elastic Net`
## [1] 4791823346
```

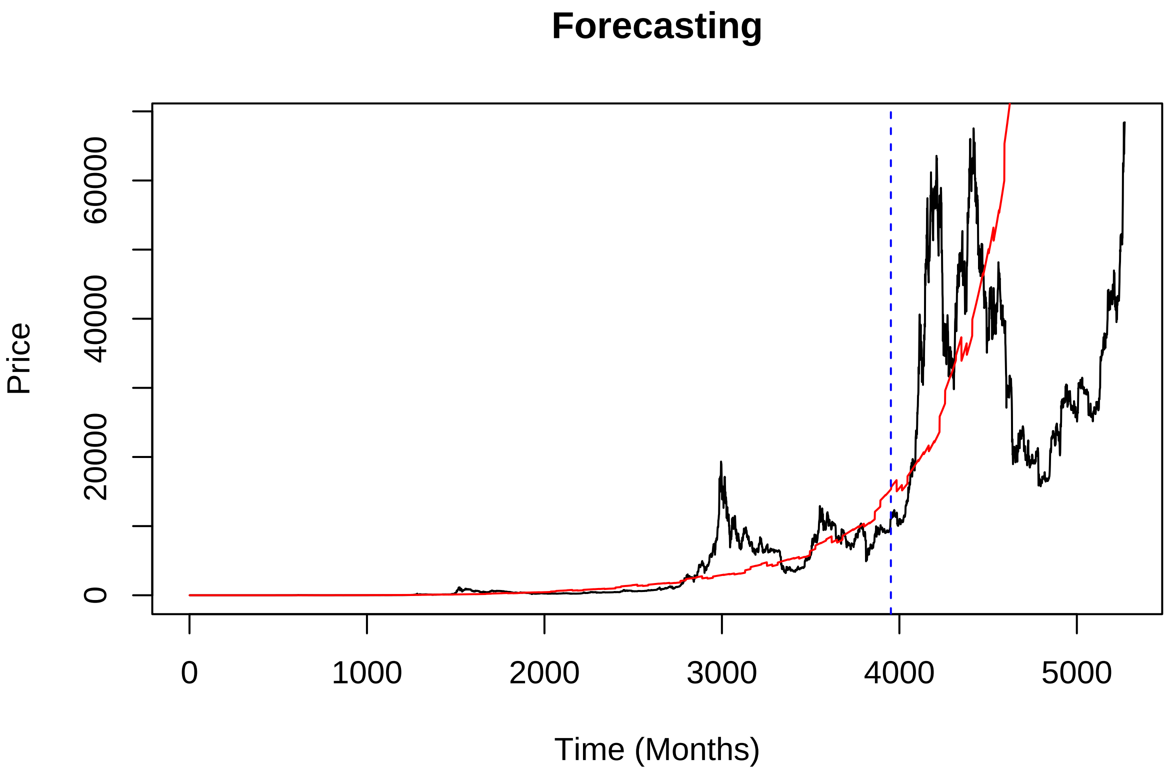
Plot Forecasting Model using Elastic Net Model

```r
full_dataset <- read.csv("/Users/pivaldhingra/Desktop/University courses/STAT 443 project /Data_Group24
full_dataset$date <- as.Date(full_dataset$date)
full_dataset$month <- as.factor(format(full_dataset$date, "%m"))
month.dummies.full <- model.matrix(~ month -1, data=full_dataset)
month.dummies.full <- month.dummies.full[,-ncol(month.dummies.full)]

Y.full <- full_dataset$price
tim.full <- 1:nrow(full_dataset)
```

```r
X.full <- cbind(poly(tim.full, degree=4, raw=TRUE), month.dummies.full)
EN.predict.full <- predict(EN.model, newx=X.full, s=0)

plot(x=tim.full, y=Y.full^0.075, type="l",
     xlab="Time (Months)", ylab="Power transformed price",
     main="Power Transformed Forecasting", ylim=c(0, 2.7))
lines(x=tim.full, y=EN.predict.full, col="red")
abline(v=3952, lty=2, col="blue")
```




```r
plot(x=tim.full, y=Y.full, type="l",
     xlab="Time (Months)", ylab="Price",
     main="Forecasting")
lines(x=tim.full, y=EN.predict.full^(1/0.075), col="red")
abline(v=3952, lty=2, col="blue")
```



Forecasting
Price
0
20000
40000
60000
0
1000
2000
3000
4000
5000
Time (Months)