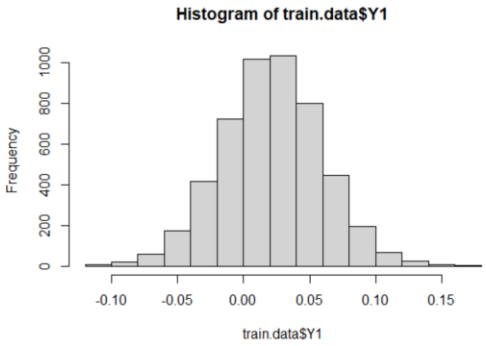
Task 1

Predictive model of Y1

(1)First, we can see below the distribution of Y1 is close to the gaussian distribution.

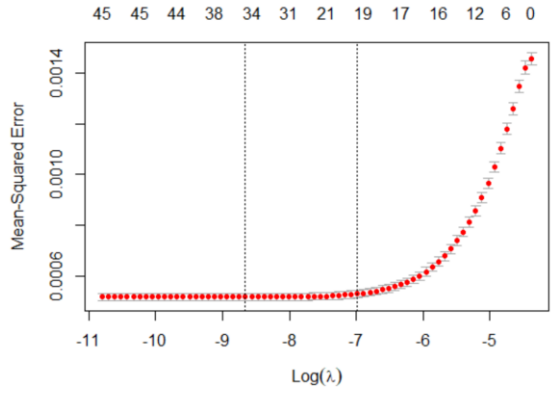
```
> hist(train.data$Y1)
```



(2)Hence, since 50 predictors seem to be too many, I used Lasso Regression to apply penalization for a better overall MSE. Also, by allowing coefficient to shrink to zero, Lasso Regression performed feature selection, in case of multicollinearity existing in the data, and to achieve a simpler model.

We set the lambda value that minimizes the MSE as the most optimal lambda value. As we can see in the plot of MSE against log(lambda) below, the optimal lambda value is 0.0001739218.

```
> library(glmnet)
> xvars = as.matrix(train.data[,4:53])
> y1 = train.data$Y1
> cv.model.1 <- cv.glmnet(xvars, y1, family = 'gaussian', alpha = 1)
> opt.lambda.1 <- cv.model.1$lambda.min
> opt.lambda.1
[1] 0.0001739218
> plot(cv.model.1)
```



(3) Then I used the optimal lambda value to fit into the final model. From the result below, we can see 34 out of 50 predictors are left after feature selection by lasso regression. 16 predictors were filtered out since they are not influential enough to Y1.

```
> opt.model.1 <- glmnet(xvars, y1, family = 'gaussian', alpha = 1, lambda = opt.l
ambda.1)
> opt.model.1

Call:  glmnet(x = xvars, y = y1, family = "gaussian", alpha = 1, lambda = opt.lam
bda.1)

  Df %Dev    Lambda
1 34 65.1 0.0001739
```

(4) I used the fitted lasso regression model to predict Y1, and calculate the R-squared value using the equation: $R^2 = 1 - \frac{SSE}{SST}$, where SSE (Sum Squared Regression Error) = $\sum_i (y_i - \hat{y}_i)^2$, SST (Sum Squared Total Error) = $\sum_i (y_i - \overline{y})^2$. And RMSE (Root Mean Square Error) = $\sqrt{\frac{SSE}{n}}$.

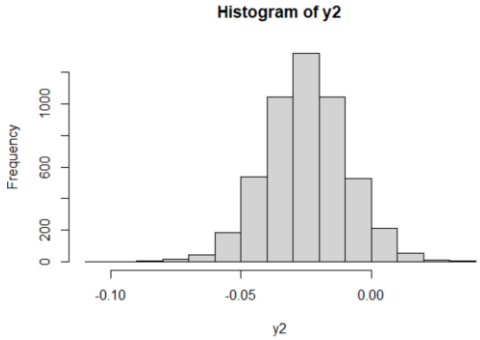
Here, R^2 is coefficient of determination and can measure how well the fitted model fits the observations. As shown below the R^2 is approximately 0.65, which means the fitted model can correctly fit 65% of the data in the training dataset. RMSE is another measurement of the model usefulness. Here, RMSE is low at 0.02255695. Both R^2 and RMSE indicate acceptable performance.

```
> ypredicted <- predict(opt.model.1, s = opt.lambda.1, newx = xvars)
> sst <- sum((y1 - mean(y1))^2)
> sse <- sum((ypredicted - y1)^2)
> #find R-Squared
> rsq <- 1 - sse/sst
> RMSE = sqrt(sse/5000)
> c(rsq,RMSE)
[1] 0.65097213 0.02255695
```

Predictive model of Y2

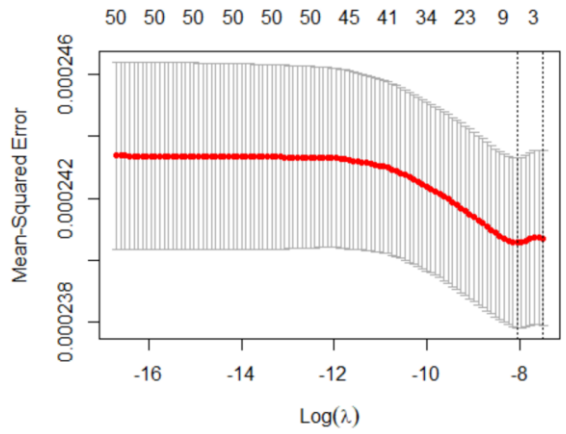
(1) I plotted Y2 as shown below and found it is also close to gaussian distribution.

```
> y2 <- train.data$Y2
> hist(y2)
```



(2) Hence, I went through the same workflow as for Y1 as shown below. We can see the predictors after filtering is only 5. The R^2 and RMSE are also not ideal at about 0.0028 and 0.0155, respectively. R^2 is too low, showing a poor prediction performance.

```
> cv.model.2 <- cv.glmnet(xvars, y2, family = 'gaussian', alpha = 1)
> opt.lambda.2 <- cv.model.2$lambda.min
> opt.lambda.2
[1] 0.0003187745
> plot(cv.model.2)
```

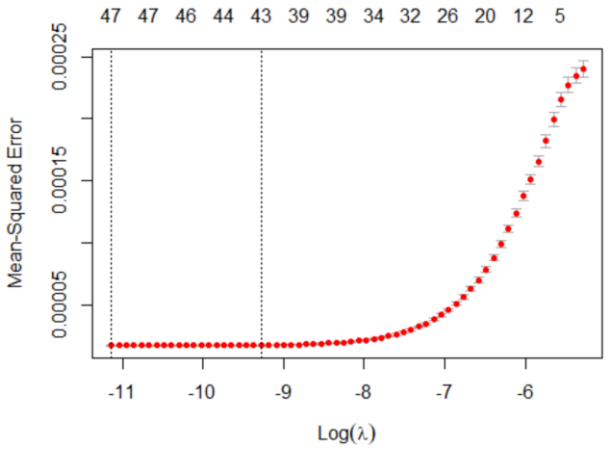


```
> opt.model.2 <- glmnet(xvars, y2, family = 'gaussian', alpha = 1, lambda = opt.l
ambda.2)
> opt.model.2

Call:  glmnet(x = xvars, y = y2, family = "gaussian", alpha = 1, lambda = opt.lam
bda.2)

  Df %Dev    Lambda
1  5 0.28 0.0003188

> ypredicted.2 <- predict(opt.model.2, s = opt.lambda.2, newx = xvars)
> sst.2 <- sum((y2 - mean(y2))^2)
> sse.2 <- sum((ypredicted.2 - y2)^2)
> rsq.2 <- 1 - sse.2 / sst.2
> RMSE.2 = sqrt(sse.2/5000)
> c(rsq.2,RMSE.2)
[1] 0.002822299 0.015487720
```

(3) I then reconsidered a better model to fit Y2. I tried using Lasso Regression model to fit the $|X_{i...50}|$ and Y2 as well as $(X_{i...50})^2$ and Y2, as shown below. We can see both the two model can achieve a good performance with R^2 and RMSE at about 0.93, 0.0041 for $|X_{i...50}|$ model, and 0.96, 0.0020 for $(X_{i...50})^2$ model. In comparison, $(X_{i...50})^2$ model has a better performance with higher usefulness.

 # Try $|X_{i...50}|$

```
> cv.model.2.abs <- cv.glmnet(abs(xvars), y2, family = 'gaussian', alpha = 1)
> opt.lambda.2.abs <- cv.model.2.abs$lambda.min
> opt.lambda.2.abs
[1] 1.453702e-05
> plot(cv.model.2.abs)
```

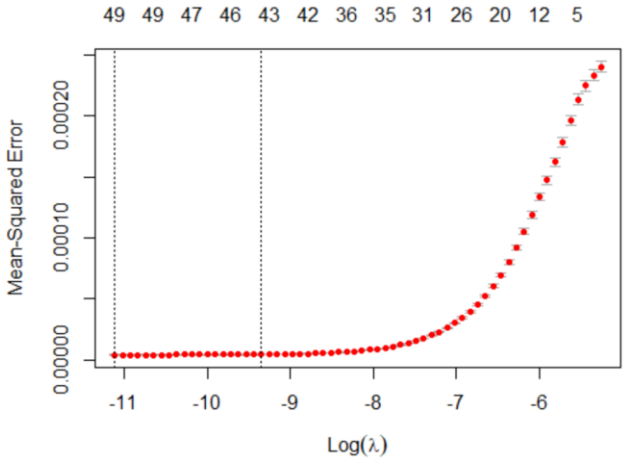



```
> opt.model.2.abs <- glmnet(abs(xvars), y2, family = 'gaussian', alpha = 1, lambda =
 opt.lambda.2.abs)
> opt.model.2.abs

Call:  glmnet(x = abs(xvars), y = y2, family = "gaussian", alpha = 1,      lambda =
 opt.lambda.2.abs)

  Df  %Dev    Lambda
1 47 93.01 1.454e-05
> ypredicted.2.abs <- predict(opt.model.2.abs, s = opt.lambda.2.abs, newx = abs(xvar
s))
> sst.2 <- sum((y2 - mean(y2))^2)
> sse.2 <- sum((ypredicted.2.abs - y2)^2)
> rsq.2 <- 1 - sse.2 / sst.2
> RMSE.2 = sqrt(sse.2/5000)
> c(rsq.2,RMSE.2)
[1] 0.930104573 0.004100394
```

 # Try $(X_{i...50})^2$

```
> cv.model.2.sq <- cv.glmnet(xvars^2, y2, family = 'gaussian', alpha = 1)
```

```
> opt.lambda.2.sq <- cv.model.2.sq$lambda.min
> opt.lambda.2.sq
[1] 1.490779e-05
> plot(cv.model.2.sq)
```

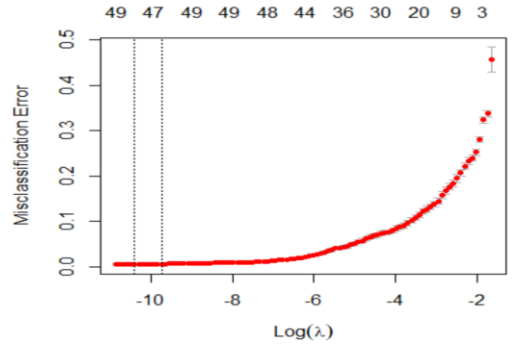



```
> opt.model.2.sq <- glmnet(xvars^2, y2, family = 'gaussian', alpha = 1, lambda = opt.
lambda.2.sq)
> opt.model.2.sq

Call:  glmnet(x = xvars^2, y = y2, family = "gaussian", alpha = 1, lambda = opt.lambd
a.2.sq)

  Df  %Dev    Lambda
1 49 98.29 1.491e-05
> ypredicted.2.sq <- predict(opt.model.2.sq, newx = xvars^2)
> sst.2 <- sum((y2 - mean(y2))^2)
> sse.2 <- sum((ypredicted.2.sq - y2)^2)
> rsq.2 <- 1 - sse.2 / sst.2
> RMSE.2 = sqrt(sse.2/5000)
> c(rsq.2,RMSE.2)
[1] 0.982928244 0.002026471
```

Predictive model of Y3

Y3 is a binary outcome, so I first tried using Lasso logistic regression model to fit it. The workflow is similar to the previous ones for Y1 and Y2, except for some arguments that were changed as shown below in R codes. By changing the 'type.measure' into 'class', the cross-validation provided the lambda value that corresponded to the minimum misclassification error. We can see below the usefulness evaluated by Lasso is 98.46%, which is high enough.

```
> y3 <- train.data$Y3
> cv.model.3 <- cv.glmnet(xvars,y3,alpha = 1, family = "binomial", type.measure = "class")
> opt.lambda.3 <- cv.model.3$lambda.min
> opt.lambda.3
[1] 3.065051e-05
> plot(cv.model.3)
```



```
> opt.model.3 <- glmnet(xvars, y3, alpha = 1, family = "binomial", lambda = opt.lambda.3)
> opt.model.3

Call:  glmnet(x = xvars, y = y3, family = "binomial", alpha = 1, lambda = opt.lambda.3)

  Df  %Dev    Lambda
1 49 98.46 3.065e-05
```

Y1, Y2, Y3 prediction for data in test.csv

I used R codes shown below to predict Y1, Y2, Y3 for the data in test.csv, and write them into a new csv file.

```
> y1.test.pred <- predict(opt.model.1, s = opt.lambda.1, newx = xvars.test)
> y2.test.pred <- predict(opt.model.2.sq, s = opt.lambda.2.sq, newx = xvars.test)
> y3.test.pred <- predict(opt.model.3, s = opt.lambda.3, family = "binomial",type = "class", newx
 = xvars.test)
> output <- data.frame(y1.test.pred, y2.test.pred, y3.test.pred)
> colnames(output) = c("Y1", "Y2", "Y3")
> head(output)
           Y1           Y2 Y3
1  0.029626679 -0.045907663  1
2 -0.022968418 -0.001880162  0
3 -0.005801067  0.001488479  0
4  0.021217641  0.014698013  0
5 -0.001050390 -0.028843679  1
6 -0.006162187 -0.023895133  0

> write.csv(output, file = "pred_1039580.csv", row.names = FALSE)
```