DSA4211 Project Report

I will describe my project by the different phases:

- Data preprocessing
- Train Validation Test split
- Models considered / Findings
- Reflection

Data Pre-processing:

- Column X55 had numerous null values, hence I removed it as it can disrupt the learning of the target function
- No columns seemed to be collinear based on a scatterplot. However, 2 columns, X30 and X40 were found to be multicollinear. As such, I removed X30.

Train Validation Test split:

- For model training, I decided to split my data in a 7:3 ratio (7/10 parts for training, 3/10 parts for testing. I felt this was a good balance between having enough data to train my models with and having enough data to estimate the actual performance of the model on unseen data.
- An explicit validation set was not considered as I wanted to use as much data as I could to train my models. Hence, for all models that required hyperparameter selection, I used 10-fold cross validation using the training set.

Models considered / Findings:

Model	Test MSE
Boosting	23.28
Forward Selection	24.67
ISIS-Lasso	25.03
Lasso	25.07
Bagging	25.19
Random Forest	26.76
Ridge	27.00
Partial Least Squares	28.20
Principal Component Regression	28.28
Decision Tree	30.64

The table on the left summarises all test performances of the models that I have considered. After hyperparameter tuning, it is found that boosting gave the best test MSE on this dataset. I used boosting to predict the test data given for this project.

Reflection: Overall, this project felt like a good exercise that allowed me to put into practice the techniques that we were taught in class. As I had to re-iterate training several times, I got more familiar with coding the models out. One thing that this project forced me to do was to think hard about the methodology of model building. I realised that it was not so simple as train and test. I had to constantly ask myself if I was doing the statistically correct thing when I was training / validating my models, and if I was carrying out a proper method in comparing my models against each other.

Data Pre-processing, Splitting

```
set.seed(4211)
full_data <- read.csv('../Project/train-xy.csv')
colnames(full_data)[colSums(is.na(full_data)) > 0] # seems like X55 has a lot of NA values
```

```
## [1] "X55"
```

```
which(diag(solve(cor(full_data[,-1]))) > 10) # seems like x30 and x40 have multicolinearity
```

```
## named integer(0)
```

```
full_data <- subset(full_data, select = -c(X55, X30)) # drop column, 100 left

# train val split
train <- sample(1:nrow(full_data), 0.7*nrow(full_data))
x_test <- full_data[-train, 2:ncol(full_data)]
x_train <- full_data[train, 2:ncol(full_data)]
y_test <- full_data[-train, 1]
y_train <- full_data[train, 1]
```

Lasso

```
## Lasso
set.seed(4211)
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.0.5
```

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

```
## Loaded glmnet 4.1-2
```

```
# select lambda via CV
grid <- 10^seq(10,-2,length=100)
lasso_mod <- glmnet(as.matrix(x_train), y_train,alpha=1,lambda=grid)
lasso_cv_out <- cv.glmnet(as.matrix(x_train), y_train, alpha=1)
bestlam <- lasso_cv_out$lambda.min
# predict using best lambda
lasso_pred <- predict(lasso_mod, s = bestlam, newx = as.matrix(x_test))
# test MSE around 25
mean((lasso_pred- y_test)^2)
```

```
## [1] 25.07412
```

ISIS-Lasso

```
## ISIS-Lasso
set.seed(4211)
library(SIS)
```

```
## Warning: package 'SIS' was built under R version 4.0.5
```

```r
lasso_sis <- cv.glmnet(as.matrix(x_train), y_train, family = "gaussian")
bestlam_lasso_sis <- lasso_sis$lambda.min
# predict
lasso_sis_pred <- predict(lasso_sis, s = bestlam_lasso_sis, newx = as.matr
ix(x_test))
# test MSE around 25
mean((lasso_sis_pred- y_test)^2)
```

```
## [1] 25.0739
```

Ridge

```r
## Ridge
# select lambda via CV
set.seed(4211)
ridge_mod <- glmnet(as.matrix(x_train), y_train,alpha=0,lambda=grid)
ridge_cv_out <- cv.glmnet(as.matrix(x_train), y_train, alpha=0)
bestlam_ridge <- ridge_cv_out$lambda.min
# predict
ridge_pred <- predict(ridge_mod, s = bestlam_ridge, newx = as.matrix(x_tes
t))
# test MSE around 27
mean((ridge_pred - y_test)^2)
```

```
## [1] 26.99917
```

Decision Tree

```r
## Decision Tree
set.seed(4211)
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.0.5
```

```r
dectree <- tree(Y~., full_data, subset = train)
# prune tree
dectree_cv_out <- cv.tree(dectree)
# best size is 12
dectree_prune <- prune.tree(dectree,best=12)
# predict
dectree_pred <- predict(dectree_prune, newdata = full_data[-train,])
# test MSE around 30
mean((dectree_pred-y_test)^2)
```

```
## [1] 30.64354
```

Bagging

```r
## Bagging
set.seed(4211)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.5

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.
```

```r
# use default m = 98
bag_mod <- randomForest(Y~., data=full_data, subset = train, mtry = 98, im
portance=TRUE)
# predict
bag_pred <- predict(bag_mod, newdata = full_data[-train,])
# test MSE around 25
mean((bag_pred-y_test)^2)
```

```
## [1] 25.18971
```

Random Forest

```r
## Random Forest
# use default m = 33
set.seed(4211)
rf_mod <- randomForest(Y~., data=full_data, subset = train, mtry = 33, imp
ortance=TRUE)
# predict
rf_pred <- predict(rf_mod, newdata = full_data[-train,])
# test MSE around 27
mean((rf_pred-y_test)^2)
```

```
## [1] 26.76351
```

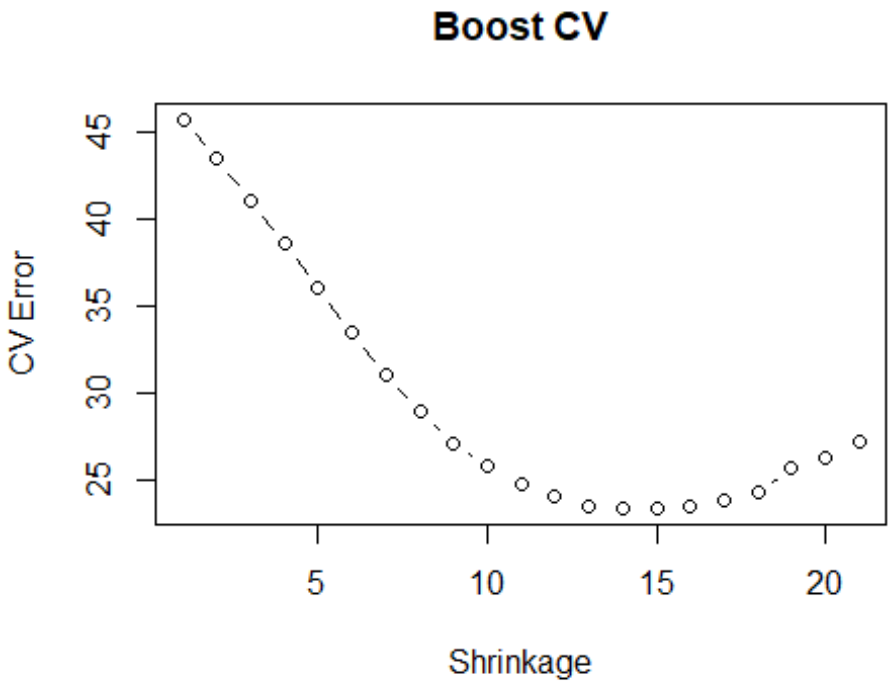
Boosting

```r
## Boosting
set.seed(4211)
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.0.5
```

```
## Loaded gbm 2.1.8
```

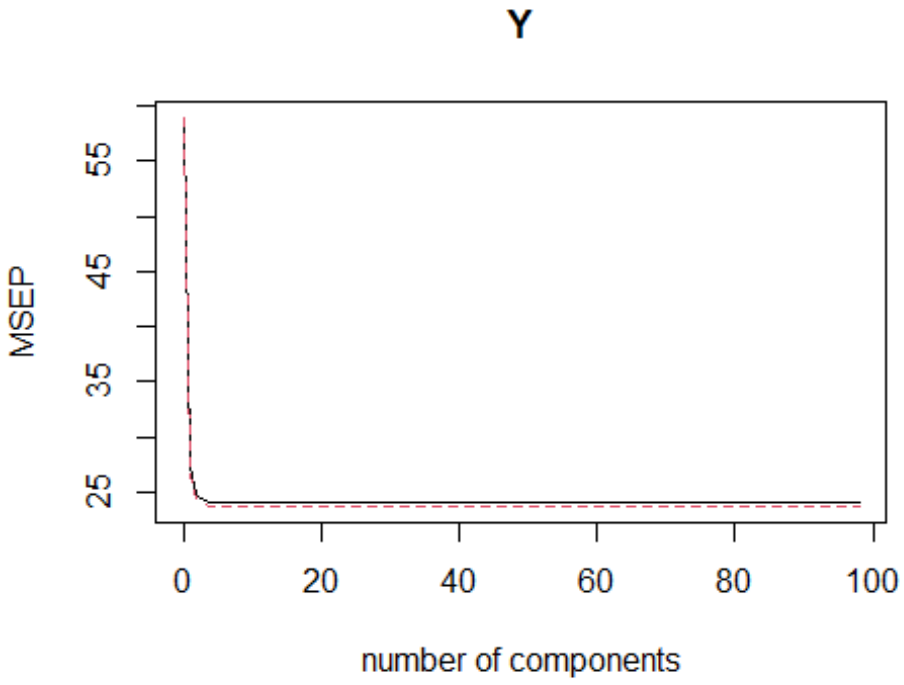
```r
k=10
# partition data into k folds
folds <- sample(1:k, nrow(full_data[train,]), replace=TRUE)
shrinkages <- 10^seq(-3, -1, 0.1)
# compute the CV error
boost_cv_errors <- matrix(NA, k, length(shrinkages), dimnames=list(NULL, p
aste(1:length(shrinkages))))
for(j in 1:length(shrinkages)) {
  for(i in 1:k){
    boost_temp <- gbm(Y~., data=full_data[train,][folds!=i,], distribution
="gaussian", n.trees=1000, shrinkage=shrinkages[j], verbose=F)
    boost_temp_pred <- predict(boost_temp, full_data[train,][folds==i,],
n.trees = 1000)
    boost_cv_errors[i,j] <- mean((full_data[train,]$Y[folds==i]-boost_temp
_pred)^2)
  }
}
# overall CV errors
mean_boost_cv_errors <- apply(boost_cv_errors, 2, mean)

plot(mean_boost_cv_errors, type='b', xlab = "Shrinkage", ylab = "CV Error
", main = "Boost CV")
```




```
# fit the final model
boost_full <- gbm(Y~., data=full_data[train,], distribution="gaussian", n.
trees=1000, shrinkage=shrinkages[13], verbose=F)

boost_full_pred <- predict(boost_full, full_data[-train,], n.trees = 1000)
# test MSE around 23
mean((boost_full_pred - y_test)^2)

## [1] 23.27814
```

PLS

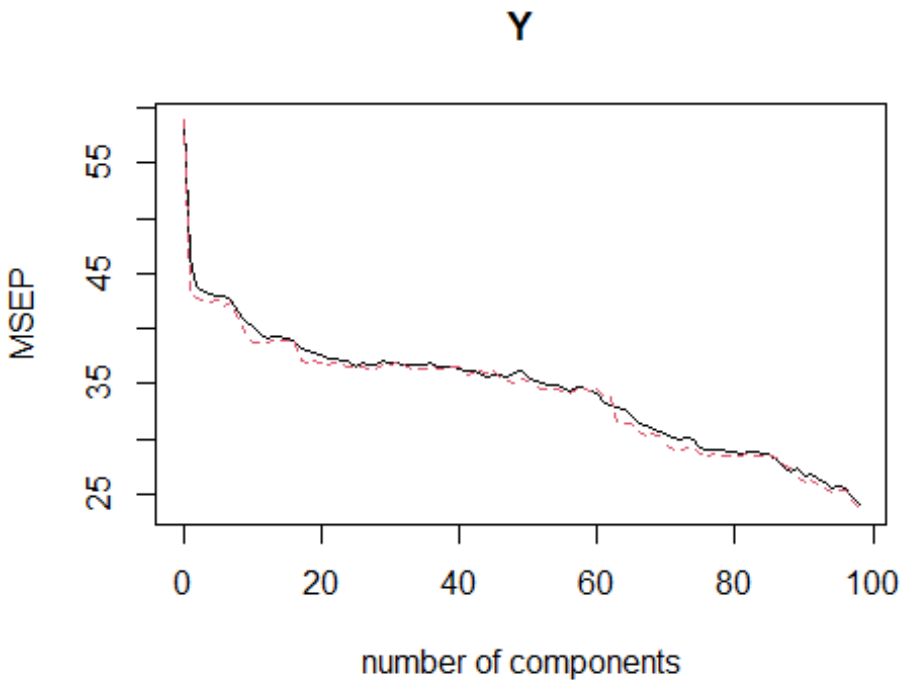
```
## PLS
set.seed(4211)
library(pls)

## Warning: package 'pls' was built under R version 4.0.5

##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings

# CV to select #PCs
pls_mod <- plsr(Y~., data = full_data, subset=train, scale=TRUE, validatio
n = "CV")
validationplot(pls_mod,val.type="MSEP")
```




```
# PLS with 4 PCs
pls_pred <- predict(pls_mod, x_test, ncomp=4)
## test MSE around 28
mean((pls_pred - y_test)^2)

## [1] 28.20802
```

PCR

```
## PCR
set.seed(4211)
pcr_mod <- pcr(Y~., data = full_data, subset=train, scale=TRUE, validation
 = "CV")
validationplot(pcr_mod,val.type="MSEP")
```

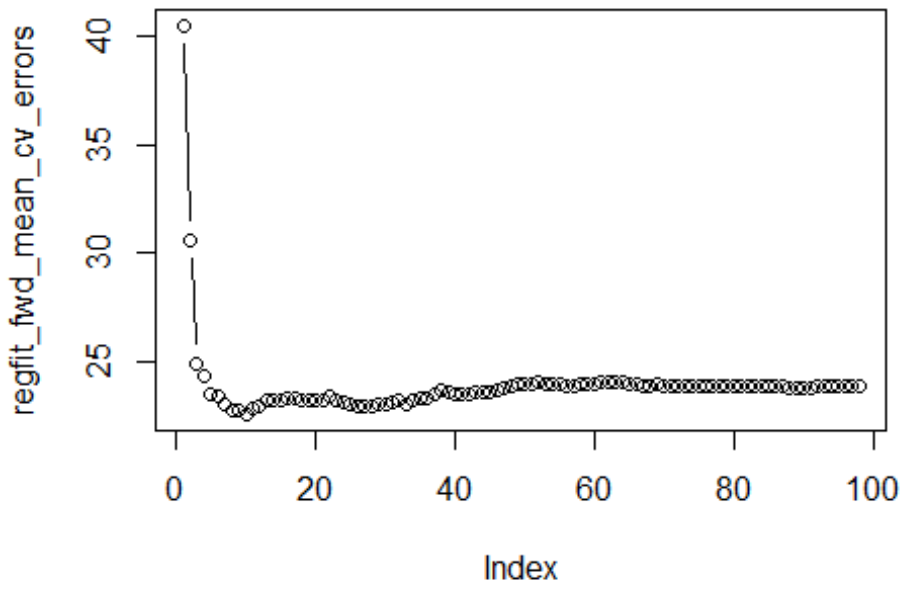



```
# PLS with 98 PCs
pcr_pred <- predict(pcr_mod, x_test, ncomp=98)
## test MSE around 28
mean((pcr_pred - y_test)^2)

## [1] 28.27716
```

Forward selection

```
## Forward selection
set.seed(4211)
library(leaps)

## Warning: package 'leaps' was built under R version 4.0.5

# define predict function
predict.regsubsets=function(object,newdata,id,...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
}
k <- 10
# partition data into k folds
folds <- sample(1:k,nrow(full_data[train,]),replace=TRUE)
# compute the CV error
regfit_fwd_cv_errors <- matrix(NA, k, 98, dimnames=list(NULL, paste(1:9
8)))
for(j in 1:k){
  best_fit <- regsubsets(Y~., data = full_data[train,][folds!=j,], nvmax=9
8, method = "forward")
```

```
  for(i in 1:98){
    pred <- predict(best_fit,full_data[train,][folds==j,],id=i)
    regfit_fwd_cv_errors[j,i] <- mean((full_data[train,]$Y[folds==j]-pred)
^2)
  }
}
# overall CV errors
regfit_fwd_mean_cv_errors <- apply(regfit_fwd_cv_errors,2,mean)
which(regfit_fwd_mean_cv_errors == min(regfit_fwd_mean_cv_errors))

## 10
## 10

# plot CV errors
# smallest test MSE at around 22 for 10 variables
plot(regfit_fwd_mean_cv_errors,type='b')
```




```
# test MSE around 24, but lower end
regfit_fwd <- regsubsets(Y~., data = full_data[train,], nvmax = 98, method
 = "forward")
regfit_fwd_coefi <- coef(regfit_fwd, 10)
regfit_fwd_pred <- predict(regfit_fwd, full_data[-train,], id = 10)
mean((regfit_fwd_pred - y_test)^2)

## [1] 24.67428
```