Title: Assignment #6: (*Subset selection for building a predicting model*)

Purpose: *This assignment is to study subset selection model in R. This assignment is to use the forward/backward subset selection, as well as Lasso and Ridge to try to create as accurate model as possible. This is to also use Cross-validation to determine the MSE.*

Dataset(s): *The dataset used is the Online News Popularity dataset.*

Link to the dataset: ***https://archive.ics.uci.edu/ml/datasets/Online+News+Popularity***

Approach:

- *First, the dataset is loaded, checked if there are any NA's and omitted if any. (This dataset had none)*
- *The dimension of the dataset is a total of 59 features (URL and timedelta aren't good for prediction) and a total of 39,644 records.*
- *Vectors such as : "**data_channel_is_lifestyle, data_channel_is_bus, data_channel_is_entertainment, data_channel_is_socmed, data_channel_is_tech, data_channel_is_world, weekday_is_Monday, weekday_is_Tuesday, weekday_is_Wednesday, weekday_is_Thursday, weekday_is_Friday, weekday_is_Saturday, weekday_is_Sunday and is_weekend"** are factors and are converted into one.*
- *Since the dataset has more than 50 vectors, we need to set the exhaustive search **reall.big to be True***
- *The number of "nvmax" can be determined after the exhaustive search is performed, in this case the **"nvmax" is 58 variables and 1subsets of each size upto 9.***
- *Once we do the forward model selection taking all the 58 variables, the best variables to be selected turns out to be 28 when we plot adjustedR^2 and also cp.*
- *After using the method regsubsets, we try to do the normal CV and k-fold cross-validation on the dataset.*
- *The predict function is not available for the regsubsets, so we write the predict function.*
- *The best predictor subset is **15** for the K-fold validation.*
- *The lasso and ridge is computed to create a accurate model as possible.*

Summary:

The subset selection first depends on converting the variables necessary as factors. The lasso and ridge coefficients give us some insights about the dataset and it is used in order to determine which subset provides us with the best co-efficient. It is also noticed that for very high record dataset the best method is not quite stable but the forward method provides us with the exhaustive results. The cross validation helps us to better determine the subset selection, since it provided lesser subset selection and the coefficients obtained where also more efficient.

Appendix :

This includes the R code that I have done to obtain the a subset selection for building a predicting model:

```
library(readr)
library(leaps)
library(ISLR)
library(glmnet)
OnlineNewsPopularity <- read_csv("~/Downloads/Education/Spring 17/R for Data Scientists/Assignments/OnlineNewsPopularity/OnlineNewsPopularity.csv")

myData = OnlineNewsPopularity

#remove url and timedelta vectors
myData <- myData[,-1]
myData <- myData[,-1]

#check to see the vectors that have to be a factor:


myData$data_channel_is_lifestyle = as.factor(myData$data_channel_is_lifestyle)
myData$data_channel_is_bus = as.factor(myData$data_channel_is_bus)
myData$data_channel_is_entertainment =
as.factor(myData$data_channel_is_entertainment)
myData$data_channel_is_socmed = as.factor(myData$data_channel_is_socmed)
myData$data_channel_is_tech = as.factor(myData$data_channel_is_tech)
myData$data_channel_is_world = as.factor(myData$data_channel_is_world)


myData$weekday_is_monday =as.factor(myData$weekday_is_monday)
myData$weekday_is_tuesday =as.factor(myData$weekday_is_tuesday)
myData$weekday_is_wednesday =as.factor(myData$weekday_is_wednesday)
myData$weekday_is_thursday =as.factor(myData$weekday_is_thursday)
myData$weekday_is_friday =as.factor(myData$weekday_is_friday)
myData$weekday_is_saturday =as.factor(myData$weekday_is_saturday)
myData$weekday_is_sunday =as.factor(myData$weekday_is_sunday)
myData$is_weekend = as.factor(myData$is_weekend)


#lets try forward model selection

news.model.fwd = regsubsets(shares~. , data=myData, method="forward") #1 subsets of each size up to 20
summary(news.model.fwd)
```

```
news.model.fwd = regsubsets(shares~., myData, really.big = T, nvmax= 58, method
="forward")

news.fwd.summary = summary(news.model.fwd)

news.fwd.summary$rsq
news.fwd.summary$adjr2
plot(news.fwd.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")
plot(news.fwd.summary$cp,xlab="Number of Variables",ylab="Cp",type="l")
plot(news.model.fwd,scale="adjr2")
news.fwd.summary$cp

coef(news.model.fwd,28)



#do a simple CV since regsubsets does not support CV
set.seed(1)
train=sample(c(TRUE,FALSE), nrow(myData),rep=TRUE)

test = (!train)
regfit.best=regsubsets(shares~.,data=myData[train,],nvmax=19,really.big =
T,method="forward")


# Problem - the predict function does not work with resubsets ...
# We need to create our own predict function
# Here is the model matrix - it contains the equation fof the (full) model

test.mat=model.matrix(shares~.,data=myData[test,])
val.errors=rep(NA,19)
for(i in 1:19){
  coefi=coef(regfit.best,id=i)
  pred=test.mat[,names(coefi)]%*%coefi
  val.errors[i]=mean((myData$shares[test]-pred)^2)
}
val.errors
# Now we can check which model gave the lowest MSE
which.min(val.errors)
coef(regfit.best,15)



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 - fold
regfit.best=regsubsets(shares~.,data=myData,nvmax=19, really.big = T, method = "forward")
k=10
set.seed(1)
folds=sample(1:k,nrow(myData),replace=TRUE)
cv.errors=matrix(NA,k,19, dimnames=list(NULL, paste(1:19)))
# This matrix will hold the MSE for each fold, for each model
for(j in 1:k){
  best.fit=regsubsets(shares~.,data=myData[folds!=j,],nvmax=19,method = "forward",
really.big = T)
  for(i in 1:19){
    pred=predict(best.fit,myData[folds==j,],id=i)
    cv.errors[j,i]=mean( (myData$shares[folds==j]-pred)^2)
  }
}
# Now we get the average over all the folds for each model size
mean.cv.errors=apply(cv.errors,2,mean)
mean.cv.errors
plot(mean.cv.errors,type='b') # looks like 15 variables is the best

reg.best=regsubsets(shares~.,data=myData, nvmax=19, really.big = T,method = "forward")
coef(reg.best,15)




x=model.matrix(shares~.,myData)[,-1]
y=myData$shares
# We create a sequence of lambdas to test
grid=10^seq(10,-2,length=100)
# We now do Ridge reqression for each lambda
ridge.mod=glmnet(x,y,alpha=0,lambda=grid)
dim(coef(ridge.mod))
# Notice we set alpha = 0; this is the option for Ridge
# Also, the glmnet scales the data automatically
# The coef matrix has a row for each predictor (plus intercept)
# and col for each lambda value
```

```
names(ridge.mod)
# We can access the lambda values, but the coeff we need to access
# separately
ridge.mod$lambda [50]
coef(ridge.mod)[,50]
# The predict function allows us to calculate the coefficients
# for lambdas that were not in our original grid
# Here are the coefficients for lambda = 50 ...
predict(ridge.mod,s=50,type="coefficients")[1:20,]
# We would like to choose an optimal lambda - we'll demonstrate
# a simple CV method here (K-fold can be used with more work).
set.seed(1)
train=sample(1:nrow(x), nrow(x)/2)
test=(-train)
y.test=y[test]
ridge.mod=glmnet(x[train,],y[train],alpha=0,lambda=grid, thresh=1e-12)
# We need to find the optimal lambda - we can use CV
set.seed(1)
cv.out=cv.glmnet(x[train,],y[train],alpha=0)
# The cv function does 10-fold CV by default
plot(cv.out)
bestlam=cv.out$lambda.min
bestlam
ridge.pred=predict(ridge.mod,s=bestlam,newx=x[test,])
mean((ridge.pred-y.test)^2)
# ONce we do CV find the best lambda, we can use all of the data
# to build our model using this lambda ...
ridge.model=glmnet(x,y,alpha=0)
predict(ridge.model,type="coefficients",s=bestlam)[1:20,]
# Note all coef are included, but some a weighted heavier than others
# Now we apply the Lasso - note all we do is change the alpha option
lasso.mod =glmnet (x[train ,],y[train],alpha =1, lambda =grid)
plot(lasso.mod)
# Note that Lasso takes certain coeff to zero for large enough lambda
# We'll do CV to find the best lambda again ...
set.seed (1)
cv.out =cv.glmnet (x[train ,],y[train],alpha =1)
plot(cv.out)
bestlam =cv.out$lambda.min
lasso.pred=predict (lasso.mod,s=bestlam,newx=x[test,])
mean(( lasso.pred -y.test)^2)
# The MSE is similar to what we saw for Ridge
# Let's find the coefficients ...
out=glmnet (x,y,alpha =1, lambda =grid)
```

```
lasso.coef=predict(out,type ="coefficients",s=bestlam )[1:20,]
lasso.coef
lasso.coef[lasso.coef!=0]
```