title: "Task1:Generating Simulated Data and performing subset selection on it"

#

########Generating the simulated data set and response variable

#

set.seed(123) data_simu <- rnorm(1000 * 20) data_simu <- matrix(data_simu,1000,20) colnames(data_simu) <- paste("X",1:20,sep="")

setting beta

set.seed(123) beta <- runif(20)

setting some of the values of beta to 0 for further investigation

beta[c(3,9,13,20)]=0

add some noise

set.seed(123) noise <- 0.0000001*rnorm(1000) reponse_var <- data_simu%%*beta + noise data_simu_new <- cbind(data_simu,reponse_var) data_simu_new<- data.frame(data_simu_new) colnames(data_simu_new)[21]<- "response_var"

###########dividing into test and training set

set.seed(123) train_indis <- sample(1:nrow(data_simu_new), round(nrow(data_simu_new)*.90), replace = FALSE)

train <- data_simu_new[train_indis,] test <- data_simu_new[-train_indis,] y_true_train = train$response_var y_true_test = test$response_var

#

#####Look at subset selection using test/training data

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 }

#####################creating objects to store error

train_err_store <- matrix(rep(NA, 20)) test_err_store <- matrix(rep(NA, 20))

#

#######fitting the exhaustive subset selection model

regfit.full <- regsubsets(response_var~., data =train, nbest = 1, nvmax = 20, method = "exhaustive") regfit.full.summary <- summary(regfit.full) names(regfit.full.summary) graphics.off() x11() par(mfrow = c(2,2)) plot(regfit.full.summary$cp, xlab = "No. of variables", ylab = "Cp", type = "l") plot(regfit.full.summary$bic, xlab = "No. of variables", ylab = "BIC", type = "l") plot(regfit.full.summary$rss, xlab = "No. of variables", ylab = "RSS", type = "l") plot(regfit.full.summary$adjr2, xlab = "No. of variables", ylab = "Adjusted Rsq", type = "l")

##############identify the optimal models using model selection measures

which(regfit.full.summary$cp == min(regfit.full.summary$cp)) which(regfit.full.summary$bic == min(regfit.full.summary$bic)) which(regfit.full.summary$rss == min(regfit.full.summary$rss)) which(regfit.full.summary$adjr2 == max(regfit.full.summary$adjr2))

##############calculatingthe MSE for each of the p variable model

for (i in 1:20){ # make the predictions and compare with the tru values# y_hat_train = predict(regfit.full, newdata = train, id = i) y_hat_test = predict(regfit.full, newdata = test, id = i) train_err_store[i] = (1/length(y_true_train))*sum((y_true_train-y_hat_train)^2) test_err_store[i] = (1/length(y_true_test))*sum((y_true_test-y_hat_test)^2)

}

#############ploting the MSE vs p variable model

graphics.off() x11() plot(train_err_store, col = "blue", type = "b", xlab = "No. of variables", ylab = "MSE") lines(test_err_store, col = "red", type = "b") legend("topright",c("Training","Test"),lty=c(1,1),lwd=c(2.5,2.5),col=c("blue","red"))

#

checking for the p variable model that generates minimum test error and training error## which(test_err_store == min(test_err_store)) which(test_err_store == min(test_err_store))

#

Some of the Beta value were set to 0 while simulating the response variable which in this case were for variables x3,x9,x13,x20 Now since its already known that the betas of these variables are 0 then the best p variable model should not contain these variables

So from the output of this command which(test_err_store == min(test_err_store)) is 16

Hence these 16 variables should not have x3,x9,x13 and x20

coef(regfit.full, 16)

#

#####Comparing the model variables coefficients which gives the minimum error with the original coefficients which in this case is

beta that was set at the beginning to generate the model

beta[c(1,2,4,5,6,7,8,10,11,12,14,15,16,17,18,19)]

coef(regfit.full, 16)

###Visualizing the squared difference between original beta(simulated model) and and new p variable models

var_min_error <-which(test_err_store == min(test_err_store)) beta_matrix <- t(beta) colnames(beta_matrix) <- paste("X",1:20,sep="")

coef_diff_store <- c() for (i in 1:var_min_error){

coef_matrix <- t(coef(regfit.full, i))

#

"which(as.numeric(colnames(beta_matrix)%in%colnames(coef_matrix))==1)" — this command picks up only those columns from beta(original coeficients)

which are present in the p variable model that is being compared to the original model(simulated one)

#

######## beta_padded_0 — a new variable is created so that it wil hold 0 as the first coefcient that corresponds to the intercept term of p

####### variable model

#

beta_padded_0 <-c(0,beta_matrix[which(as.numeric(colnames(beta_matrix)%in%colnames(coef_matrix))==1)])

coef_diff <- sqrt(sum((beta_padded_0-coef_matrix)*(beta_padded_0-coef_matrix)))

coef_diff_store <-c(coef_diff_store,coef_diff)

}

coef_diff_store<-as.matrix(coef_diff_store)

graphics.off() x11() plot(coef_diff_store, col = "blue", type = "b", xlab = "No. of variables", ylab = "coef_root_squared_diff",xlim=c(1,16))

#

Title - Task3————Analysis of Diabetes data

#

load("Diabetes.RData") sum(is.na(Diabetes))
col <- c("blue", "red", "darkgreen")[Diabetes$group] col <- c("blue", "red", "darkgreen")[Diabetes$group] pch <- c(16,15,17)[Diabetes$group]

########Instest versus all variables

graphics.off() x11() par(mfrow = c(2,2)) plot(instest ~ glutest, data=Diabetes, pch=pch,col=col, cex.lab=1.25, xlab="Insulin response to oral glucose", ylab="Glucose intolerance level") plot(instest ~ relwt, data=Diabetes, pch=pch,col=col, cex.lab=1.25, xlab="Insulin response to oral glucose", ylab="Relative Weight") plot(instest ~ glufast, data=Diabetes, pch=pch,col=col, cex.lab=1.25, xlab="Insulin response to oral glucose", ylab="Fasting plasma glucose level,") plot(instest ~ sspg, data=Diabetes, pch=pch,col=col, cex.lab=1.25, xlab="Insulin response to oral glucose", ylab="Insulin resistance level,")

########Glutest versus all variables

graphics.off() x11() par(mfrow = c(2,2)) plot(glutest ~ instest, data=Diabetes, pch=pch,col=col, cex.lab=1.25, xlab="Glucose intolerance level", ylab="Insulin response to oral glucose") plot(glutest ~ relwt, data=Diabetes, pch=pch,col=col, cex.lab=1.25, xlab="Glucose intolerance level", ylab="Relative Weight") plot(glutest ~ glufast, data=Diabetes, pch=pch,col=col, cex.lab=1.25, xlab="Glucose intolerance level", ylab="Fasting plasma glucose level,") plot(glutest ~ sspg, data=Diabetes, pch=pch,col=col, cex.lab=1.25, xlab="Glucose intolerance level", ylab="Insulin resistance level,")

########Relative weight versus all variables

graphics.off() x11() par(mfrow = c(2,2)) plot(relwt ~ instest, data=Diabetes, pch=pch,col=col, cex.lab=1.25, xlab="Relative Weight", ylab="Insulin response to oral glucose") plot(relwt ~ glutest, data=Diabetes, pch=pch,col=col, cex.lab=1.25, xlab="Relative Weight", ylab="Glucose intolerance level") plot(relwt ~ glufast, data=Diabetes, pch=pch,col=col, cex.lab=1.25, xlab="Relative Weight", ylab="Fasting plasma glucose level,") plot(relwt ~ sspg, data=Diabetes, pch=pch,col=col, cex.lab=1.25, xlab="Relative Weight", ylab="Insulin resistance level,")

#########Insulin resistance level versus all variables

graphics.off() x11() par(mfrow = c(2,2)) plot(sspg ~ instest, data=Diabetes, pch=pch,col=col, cex.lab=1.25, xlab="Insulin resistance level", ylab="Insulin response to oral glucose") plot(sspg ~ glutest, data=Diabetes, pch=pch,col=col, cex.lab=1.25, xlab="Insulin resistance level", ylab="Glucose intolerance level") plot(sspg ~ glufast, data=Diabetes, pch=pch,col=col, cex.lab=1.25, xlab="Insulin resistance level", ylab="Fasting plasma glucose level,") plot(sspg ~ relwt, data=Diabetes, pch=pch,col=col, cex.lab=1.25, xlab="Insulin resistance level", ylab="Relative Weight,")

graphics.off() x11() covEllipses(Diabetes[,2:5], Diabetes$group, fill=TRUE, pooled=FALSE, col=col)

##Visulaing the density distribution of each of the variables

dens_instest<- density(Diabetes$instest) dens_glutest <- density(Diabetes$glutest) dens_relwt <- density(Diabetes$relwt) dens_glufast <- density(Diabetes$glufast) dens_sspg <- density(Diabetes$sspg) graphics.off() x11() par(mfrow= c(3,2)) hist(Diabetes$instest,probability=T,ylim=c(0,0.007),xlim=c(-50,800),xlab="Insulin response to oral glucose",main="Hist of Insulin response") lines(dens_instest) hist(Diabetes$glutest,probability=T,ylim=c(0,0.0035),xlim=c(0,1500),xlab="Glucose intolerance level",main="Hist of Glucose intolerance") lines(dens_glutest) hist(Diabetes$relwt,probability=T,xlim=c(0.5,1.6),xlab="Relative Weight",main="Hist of Relative Weight") lines(dens_relwt) hist(Diabetes$glufast,probability=T,ylim=c(0,0.0350),xlab="Fasting plasma glucose level",main="Hist of Plasma glucose") lines(dens_glufast) hist(Diabetes$sspg,probability=T,xlim=c(-100,600),xlab="Insulin resistance level",main="Hist of Insulin resistance") lines(dens_sspg)

dens <- density(Diabetes$group)

hist(Diabetes$group,probability=T,xlab="Group",main="Hist of group") lines(dens)

#############################ggplot

graphics.off()
x11() ggpairs(Diabetes, columns = 1:5, aes(color = group, alpha = 0.5), upper = list(continuous = wrap("cor", size = 2.5)))

########## Visualizing the corelation matrix

Diabetes_data <- Diabetes[,-6] cor(Diabetes_data)

##Splitting data into train and test

set.seed(123) indis <- sample(1:nrow(Diabetes), round(2/3*nrow(Diabetes)), replace = FALSE) Diabetes_train <- Diabetes[indis,] Diabetes_test <- Diabetes[-indis,] dim(Diabetes_train) dim(Diabetes_test)

##Fitting LDA and QDA

lda.fit <- lda(group~., data = Diabetes_train) train_pred <- predict(lda.fit, newdata = Diabetes_train) class(train_pred) data.frame(train_pred$class, train_pred$posterior, train_pred$x)[1:5,]

test_pred <- predict(lda.fit, newdata = Diabetes_test) class(test_pred) data.frame(test_pred$class, test_pred$posterior, test_pred$x)[1:5,]

lda_train_error <- (1/length(Diabetes_train$group))*length(which(Diabetes_train$group != train_pred$class)) lda_test_error <- (1/length(Diabetes_test$group))*length(which(Diabetes_test$group != test_pred$class)) lda_train_error lda_test_error

##QDA

qda.fit <- qda(group~., data = Diabetes_train) train_pred <- predict(qda.fit, newdata = Diabetes_train) class(train_pred)

test_pred <- predict(qda.fit, newdata = Diabetes_test) class(test_pred)

qda_train_error <- (1/length(Diabetes_train$group))*length(which(Diabetes_train$group != train_pred$class)) qda_test_error <- (1/length(Diabetes_test$group))*length(which(Diabetes_test$group != test_pred$class)) qda_train_error qda_test_error

new_row <- c(1.86, 184, 68,122,544) Diabetes_new <- rbind(Diabetes, new_row)

Diabetes_new$relwt <- as.numeric(Diabetes_new$relwt) Diabetes_new$glufast <- as.integer(Diabetes_new$glufast) Diabetes_new$glutest <- as.integer(Diabetes_new$glutest) Diabetes_new$instest <- as.integer(Diabetes_new$instest) Diabetes_new$sspg <- as.integer(Diabetes_new$sspg) Diabetes_new$group <- as.factor(Diabetes_new$group)

lda_pred_class <- predict(lda.fit, newdata = Diabetes_new[146,])$class

qda_pred_class <- predict(qda.fit, newdata = Diabetes_new[146,])$class

lda_pred_class qda_pred_class

#

############Task 2 Weekly data of ISLR package

#########Visualising the data

summary(Weekly) sum(is.na(Weekly)) Weekly_new <- Weekly[,-9] cor(Weekly_new) correlation_data <-cor(Weekly_new) library(corrplot) graphics.off() x11() corrplot(correlation_data, type = "upper", order = "hclust", tl.col = "black", tl.srt = 45) xtabs(~Direction,data=Weekly) graphics.off() favstats(Lag1~Direction,data=Weekly) favstats(Today~Direction,data=Weekly)

graphics.off() x11() par(mfrow = c(2,1)) boxplot(Today~Direction,data=Weekly, main="Stock indicator", xlab="Today", ylab="Direction") boxplot(Lag1~Direction,data=Weekly, main="Stock indicator", xlab="Percentage return for week1", ylab="Direction") graphics.off() x11() par(mfrow = c(2,1)) boxplot(Lag2~Direction,data=Weekly, main="Stock indicator", xlab="Percentage return for week2", ylab="Direction") boxplot(Lag3~Direction,data=Weekly, main="Stock indicator", xlab="Percentage return for week3", ylab="Direction") graphics.off() x11() par(mfrow = c(2,2)) boxplot(Lag4~Direction,data=Weekly, main="Stock indicator", xlab="Percentage return for week4", ylab="Direction") boxplot(Lag5~Direction,data=Weekly, main="Stock indicator", xlab="Percentage return for week5", ylab="Direction") boxplot(Year~Direction,data=Weekly, main="Stock indicator", xlab="Year", ylab="Direction")

Density distribution of each variable broken down by DIrection

library(caret) graphics.off() x11() x <- Weekly[,1:8] y <- Weekly[,9] scales <- list(x=list(relation="free"), y=list(relation="free")) featurePlot(x=x, y=y, plot="density", scales=scales)

library(dplyr) library(GGally)

graphics.off()
x11() ggpairs(Weekly, columns = 1:8, aes(color = Direction, alpha = 0.5), upper = list(continuous = wrap("cor", size = 2.5)))

#

Fitting the model with Logistic Regression

glm.fit <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Weekly, family = binomial) summary(glm.fit)

###############################predict() of glm.fit() to glm.probs, with type equals to response.

This will make predictions on the training data that we used to fit the model and give me a vector of fitted probabilities.

glm.probs <- predict(glm.fit,type = "response")

##############################prediction of whether the market will be up or down based on the lags and other predictors.

##In particular, the probabilities shall be turned into classifications by thresholding at 0.5.

glm.pred <- ifelse(glm.probs > 0.5, "Up", "Down")

glm.pred is a vector of trues and falses. If glm.probs is bigger than 0.5, glm.pred calls "Up"; otherwise, it calls "Down"

table(glm.pred,Direction)

######Creating Training and Test Samples as per the question

train <- (Weekly$Year<=2008)

glm.fit <- glm(Direction ~ Lag2, data = Weekly, subset = train, family = "binomial")

glm.probs = predict(glm.fit, newdata=Weekly[!train,], type = "response") glm.pred <- ifelse(glm.probs > 0.5, "Up", "Down")

Direction.2009_2010 <- Weekly$Direction[!train] table(glm.pred, Direction.2009_2010) mean(glm.pred == Direction.2009_2010)

##Fitting data with Linerar Dicriminant Analysis

library(klaR) lda.fit = lda(Direction ~ Lag2, data = Weekly, subset = train) lda.pred <-predict(lda.fit, Weekly[!train,]) table(lda.pred$class, Direction.2009_2010) mean(lda.pred$class == Direction.2009_2010)

#######################################Splitting data into train test with Lag 2 as only predictor

##and Fitting the data with K nearest neighbor model where k=1

set.seed(1) train.X = data.frame(Weekly[train,]$Lag2) test.X = data.frame(Weekly[!train,]$Lag2) train.Direction = Weekly[train,]$Direction library(class) knn.pred = knn(train.X, test.X, train.Direction, k = 1) table(knn.pred, Direction.2009_2010) mean(knn.pred == Direction.2009_2010)

#################### tring to fit the model with different combination of predictors

############Trying to fit and predict with logistic regression

weighted.lag.avg = 0.4*Weekly$Lag1 + 0.35*Weekly$Lag2 + 0.15*Weekly$Lag3 + 0.05*Weekly$Lag4 + 0.05*Weekly$Lag5 Weekly = data.frame(Weekly, weighted.lag.avg) head(Weekly) train <- (Weekly$Year<=2008)

glm.fit <- glm(Direction ~ weighted.lag.avg, data = Weekly, subset = train, family = "binomial")

glm.probs = predict(glm.fit, newdata=Weekly[!train,], type = "response") glm.pred <- ifelse(glm.probs > 0.5, "Up", "Down")

Direction.2009_2010 <- Weekly$Direction[!train] table(glm.pred, Direction.2009_2010) mean(glm.pred == Direction.2009_2010)

#############Fitting with Linear Dicriminant Analysis

lda.fit = lda(Direction ~ weighted.lag.avg, data = Weekly, subset = train) lda.pred <-predict(lda.fit, Weekly[!train,]) table(lda.pred$class, Direction.2009_2010) mean(lda.pred$class == Direction.2009_2010)

#############trying to fit and predict with QDA

qda.fit = qda(Direction ~ weighted.lag.avg, data = Weekly, subset = train) qda.pred <-predict(qda.fit, Weekly[!train,]) table(qda.pred$class, Direction.2009_2010) mean(qda.pred$class == Direction.2009_2010)

############trying to fit with Knn where k=1

set.seed(1) train.X = data.frame(Weekly[train,"weighted.lag.avg"]) test.X = data.frame(Weekly[!train,"weighted.lag.avg"]) train.Direction = Weekly[train,"Direction"] library(class) knn.pred = knn(train.X, test.X, train.Direction, k = 1) table(knn.pred, Direction.2009_2010) mean(knn.pred == Direction.2009_2010)

############trying to fit with Knn where k=3

set.seed(1) knn.pred = knn(train.X, test.X, train.Direction, k = 3) table(knn.pred, Weekly[!train,]$Direction) mean(knn.pred == Direction.2009_2010)

############trying to fit with Knn where k=7

set.seed(1) knn.pred = knn(train.X, test.X, train.Direction, k = 7) table(knn.pred, Weekly[!train,]$Direction) mean(knn.pred == Direction.2009_2010)