title: "Task1:Modelling of cereal data" output: pdf_document

**####################Using Linear Regression Model**

cereal <- read.delim("cereal.csv",sep=",") rownames(cereal) <- cereal$name cereal<- subset(cereal, select =-name) sum(is.na(cereal)) summary(cereal) is.null(cereal) cereal$mfr<- as.factor(cereal$mfr) cereal$mfr<- as.factor(cereal$type) cereal$type<- as.numeric(cereal$mfr) cereal$type<- as.numeric(cereal$type) set.seed(1) sample_size = round(nrow(cereal)*.80) index <- sample(seq_len(nrow(cereal)), size = sample_size) train <- cereal[index, ] test <- cereal[-index, ] model_linear_regression<-lm(rating~.,data=train) mean(model_linear_regression$residuals^2)

**####################MSE plot of Forward Subset Selection**

forward_Subset_selection<- regsubsets(rating~., data = train,method="forward,nvmax=20) new_test<-cbind(rep(1,length(test[,1]))) new_test<-cbind(rep(1,length(test[,1])),test) colnames(new_test[,1])<-"(Intercept)" colnames(new_train[,1])<-"(Intercept)" test.errors=rep(NA,14)#for collecting the test errors train.errors=rep(NA,14)#for collecting the train errors for (i in 1:14){ coefi =coef(forward_Subset_selection,id=i ) pred_test=as.matrix(new_test[,names(coefi)])%*%coefi pred_train=as.matrix(new_train[,names(coefi)])%*%coefi test.errors[i]=(1/length(new_test$rating))*sum((new_test$rating-pred_test)^2) train.errors[i]=(1/length(new_train$rating))*sum((new_train$rating-pred_train)^2) } x11() plot(test.errors,col="red",type="b",xlab="MSE of forward subset selection",ylab="MSE") lines(train.errors,col="blue",type="b") legend("topright",c("Test","Training"),col=c("red","blue"))

# Residual error plot of Forward Subset Selection

summary<-summary(forward_Subset_selection) cereal_model_size <- as.numeric(attr(summary$which,"dimnames")[[1]]) cereal_model_rss <- summary$rss cereal_model_min_rss <- tapply(cereal_model_rss,cereal_model_size,min) dummy <- lm(rating ~ 1,data=train) cereal_model_min_rss <- c(sum(resid(dummy)^2),cereal_model_min_rss) cereal_model_min_rss <- c(sum(resid(dummy)^2),cereal_model_min_rss) plot(0:14,cereal_model_min_rss,cereal_model_size,xlab="subset size",ylab="Residual Sum Square of forward subset selection",col="red2",type="b",) points(cereal_model_size,cereal_model_rss,pch=17,col="brown",cex=0.7)

**####################MSE plot of Exhaustive Subset Selection**

exhaustive_Subset_selection<- regsubsets(rating~., data = train,nbest=20,really.big=TRUE) new_test<-cbind(rep(1,length(test[,1]))) new_test<-cbind(rep(1,length(test[,1])),test) colnames(new_test[,1])<-"(Intercept)" colnames(new_train[,1])<-"(Intercept)" test.errors=rep(NA,20)#for collecting the test errors train.errors=rep(NA,20)#for collecting the train errors for (i in 1:20){ coefi =coef(exhaustive_Subset_selection,id=i ) pred_test=as.matrix(new_test[,names(coefi)])%*%coefi pred_train=as.matrix(new_train[,names(coefi)])%*%coefi test.errors[i]=(1/length(new_test$rating))*sum((new_test$rating-pred_test)^2) train.errors[i]=(1/length(new_train$rating))*sum((new_train$rating-pred_train)^2) } x11() plot(test.errors,col="red",type="b",xlab="MSE of exhaustive subset selection",ylab="MSE") lines(train.errors,col="blue",type="b") legend("topright",c("Test","Training"),col=c("red","blue"))

# Residual error plot of Exhaustive Subset Selection

summary<-summary(exhaustive_Subset_selection) cereal_model_size <- as.numeric(attr(summary$which,"dimnames")[[1]]) cereal_model_rss <- summary$rss cereal_model_min_rss <- tapply(cereal_model_rss,cereal_model_size,min) dummy <- lm(rating ~ 1,data=train) cereal_model_min_rss <- c(sum(resid(dummy)^2),cereal_model_min_rss) plot(0:8,cereal_model_min_rss,cereal_model_size,xlab="subset size",ylab="Residual Sum Square of exhaustive subset selection",col="red2",type="b",) points(cereal_model_size,cereal_model_rss,pch=17,col="brown",cex=0.7)

**##################################################—**

title: "Task2:Modelling of zip code data" output: pdf_document

**#**

**Checking and cleaning data of Zip Code**

**#**

zip.train <- data.frame(zip.train) sum(is.na(zip.train)) zip.train[,c(4,7)] is.null(zip.train) zip.test <- data.frame(zip.test)

sum(is.na(zip.test)) is.null(zip.test) zip.train<-subset(zip.train,X1==4|X1==7) sapply(zip.train[1, ], class) summary(zip.train) zip.test<-subset(zip.test,X1==4|X1==7) sapply(zip.test[1, ], class) summary(zip.test)

**#**

**###########Transformation. Transform Label as Factor (Categorical) and Change Column Names (TRAINING data set)**

**#**

zip.train[, 1] <- as.factor(zip.train[, 1]) # As Category colnames(zip.train) <- c("Y", paste("X.", 1:256, sep = "")) class(zip.train[, 1])

zip.test[, 1] <- as.factor(zip.test[, 1]) # As Category colnames(zip.test) <- c("Y", paste("X.", 1:256, sep = "")) class(zip.test[, 1]) pty = "s","m"

**#**

**################Display digits of training data set (An average of each digits)**

**#**

par(mfrow =c(1,2), pty = "s", xaxt = "n", yaxt = "n") images_digits_0_2 <- array(dim = c(2, 16 * 16)) colors<-c('white','black') cus_col<-colorRampPalette(colors=colors) x <- c(4,7) for (digit in 0:1) { print(digit) images_digits_0_2[digit + 1, ] <- apply(zip.train[zip.train[, 1] == x[digit +1], -1], 2, sum) images_digits_0_2[digit + 1, ] <- images_digits_0_2[digit + 1, ]/max(images_digits_0_2[digit + 1, ]) * 255 z <- array(images_digits_0_2[digit + 1, ], dim = c(16, 16)) z <- z[, 16:1] ##right side up image(1:16, 1:16, z, main = digit, col = cus_col(256)) } zip.train[zip.train[, 1] == x[1], -1] par(mfrow=c(1,2),pty='s',xaxt='n',yaxt='n') colors<-c('white','black') cus_col<-colorRampPalette(colors=colors) for(i in 1:2) { z<-array(as.vector(as.matrix(zip.train[i, -1])),dim=c(16,16)) z<-z[,16:1] ##right side up image(1:16,1:16,z,main=zip.train[i,1],col=cus_col(256))

}

resTable <- table(zip.train$Y) par(mfrow = c(1, 1)) par(mar = c(5, 4, 4, 2) + 0.1) # increase y-axis margin. plot <- plot(zip.train$Y, col = cus_col(2), main = "Total Number of Digits (Training Set)", ylim = c(0, 1500), ylab = "Examples Number") text(x = plot, y = resTable + 50, labels = resTable, cex = 0.75)

par(mfrow = c(1, 1)) percentage <- round(resTable/sum(resTable) * 100) labels <- paste0(row.names(resTable), " (", percentage, "%) ") # add percents to labels pie(resTable, labels = labels, col = cus_col(2), main = "Total Number of Digits (Training Set)")

#

**################Visualizing testing data**

#

## Histogram

resTable <- table(zip.test$Y) par(mfrow = c(1, 1)) par(mar = c(5, 4, 4, 2) + 0.1) # increase y-axis margin. plot <- plot(zip.test$Y, col = cus_col(2), main = "Total Number of Digits (Testing Set)", ylim = c(0, 400), ylab = "Examples Number") text(x = plot, y = resTable + 20, labels = resTable, cex = 0.75)

### Pie chart

par(mfrow = c(1, 1)) percentage <- round(resTable/sum(resTable) * 100) labels <- paste0(row.names(resTable), " (", percentage, "%) ") # add percents to labels pie(resTable, labels = labels, col = cus_col(2), main = "Total Number of Digits (Testing Set)")

#

**################ Modelling the data using linear regression**

#

zip_code_model <- lm(zip.train$Y ~.,data=zip.train)

zip_code_model <- lm(zip.train$Y ~.,data=zip.train) predict_linear_model <- predict(zip_code_model, newdata = zip.test) plot(zip_code_model$residuals) table(`Actual Class` = zip.test$Y, `Predicted Class` = predict_linear_model)

#

**################Visualizing the data and mOdelling the data using KNN**

#

library("ggplot2") library("magrittr")

**########library(readr)**

library("dplyr") library("tidyr")

#

1)It would be challenging to visualize this default data as images, but it becomes easier once we consider each pixel an “observation” hence the data is represented as one row per pixel per instance. 2)Also customizing features that have meaning within the problem For instance, the row values cannot be interpreted as 256 arbitrary features that is we have 28 rows and 28 columns. So rather than having labels like “X2” and “X3”, we’d like to keep track of variables for x and y (as coordinate positions of the images).

#

pixels_gathered <- zip.train %>% rename(label = X1) %>% mutate(instance = row_number()) %>% gather(pixel, value, -label, -instance) %>% tidyr::extract(pixel, "pixel", "(\d+)", convert = TRUE) %>% mutate(pixel = pixel - 2, x = pixel %% 16, y = 16 - pixel %/% 16)

#

**#####GGPLOT OF THE pixels:**

#

pixels_gathered %>% filter(instance <= 12) %>% ggplot(aes(x, y, fill = value)) + geom_tile() + + facet_wrap(~ instance + label)

## Histogram

ggplot(pixels_gathered, aes(value)) + geom_histogram()

#

## In order to check how much variability there is within each digit label. Do all 4s look like each other, and what is the “most typical” example of a 7? To answer this, we can find the mean value for each position within each label, using dplyr’s

group_by and summarize.

## some digits might fall widely outside the norm. It’s useful to explore atypical cases, since it could help us understand why the method fails and help us choose a method and engineer features.

## In this case, we could consider the Euclidean distance (square root of the sum of squares) of each image to its label’s centroid.

#

pixel_summary <- pixels_gathered %>% group_by(x, y, label) %>% summarize(mean_value = mean(value)) %>% ungroup() pixels_joined <- pixels_gathered %>% inner_join(pixel_summary, by = c("label", "x", "y"))

image_distances <- pixels_joined %>% group_by(label, instance) %>% summarize(euclidean_distance = sqrt(mean((value - mean_value) ^ 2)))

image_distances

#

ggplot(image_distances, aes(factor(label), euclidean_distance)) + geom_boxplot() + labs(x = "Digit", y = "Euclidean distance to the digit centroid")

#

model.knn11 <- knn(zip.train[, -1], zip.test[, -1], zip.train$Y, k = 11) error.rate.knn11 <- sum(zip.test$Y != model.knn11)/nrow(zip.test) error.rate.knn11 table(`Actual Class` = zip.test$Y, `Predicted Class` = model.knn11) summary(model.knn11)

#

title: "Task1:Modelling COllege Data"

#

output: pdf_document sum(is.na(College)) is.null(College) set.seed(1) sample_size = round(nrow(College)*.80) index <- sample(seq_len(nrow(College)), size = sample_size) train <- College[index, ] test <- College[-index, ] model_linear_regression<-lm(Apps~.,data=train)

#

## #

### Fit a Ridge Regression Model

#

new_train<-subset(train,select=-Apps) X<-as.matrix(new_train) Y <- train[,2] college_ridge_model = glmnet(X, Y, alpha=0) names(college_ridge_model) coef(college_ridge_model) dim(coef(college_ridge_model))

## Look at different lambdas

ridge.mod$lambda[10] coef(ridge.mod)[,10] l2_norm <- sqrt(sum(coef(ridge.mod)[2:9,10]^2)) l2_norm

#

predict the ridge regression model for a new value of lambda

#

predict(college_ridge_model, s = .0005, type = "coefficient") predict(college_ridge_model, s = .75, type = "coefficient")

#

#

## Model Selection

#

set.seed(12345) train <- sample(1:nrow(X), round(nrow(X)/2)) cv.out <- cv.glmnet(X[train,], Y[train], alpha = 0) plot(cv.out) bestlam <- cv.out$lambda.min ridge_pred_lamda_cross_val <- predict(college_ridge_model, s= bestlam, type = "coefficients") ridge_pred_lamda_cross_val_test_on_unpred_data<- predict(college_ridge_model, s = bestlam, newx = X[-train,], type = "response") y_hat <- ridge_pred_lamda_cross_val_test_on_unpred_data y_true <- Y[-train] test_error <- sum((y_hat - y_true)^2)

###### ##############Modelling with Lasso

lasso_mod <- glmnet(X[train,], Y[train], alpha = 1) x11() plot(lasso_mod)

# Best Lambda

cv.out = cv.glmnet(X[train,], Y[train], alpha = 1) bestlam = cv.out$lambda.min

collge_lasso_pred <- predict(lasso_mod, s = bestlam, type = "coefficients")

collge_lasso_pred1<- predict(lasso_mod, s = bestlam, newx = X[-train,], type = "response")

y_hat_lasso <- collge_lasso_pred1 y_true <- Y[-train]

test_error_lasso <- sum((y_hat_lasso-y_true)^2)