############################### Task1

regfit.full <- regsubsets(medv~., data = Boston, nbest = 1, nvmax = 12, method = "exhaustive") my_sum <- summary(regfit.full) graphics.off x11() par(mfrow = c(2,2)) plot(my_sum$cp, xlab = "No. of variables", ylab = "AIC", type = "l") plot(my_sum$bic, xlab = "No. of variables", ylab = "BIC", type = "l") plot(my_sum$rss, xlab = "No. of variables", ylab = "RSS", type = "l") plot(my_sum$adjr2, xlab = "No. of variables", ylab = "Adjusted Rsq", type = "l")

plot(my_sum$cp, xlab = "No. of variables", ylab = "AIC", type = "l",main="AIC PLOT") which(my_sum$cp == min(my_sum$cp)) plot(my_sum$bic, xlab = "No. of variables", ylab = "BIC", type = "l") which(my_sum$bic == min(my_sum$bic))

#

install.packages("bestglm") Xy<-cbind(as.data.frame(Boston), medv=y) Boston.bestglm<-within(Boston, { y <- medv medv<-NULL# Delete bwt }) regfit.full_BIC<-bestglm(Xy=Boston.bestglm,family = gaussian,IC = "BIC",nvmax =12,method = "exhaustive",CVArgs = "default", qLevel = 0.99, TopModels = 1) regfit.full_AIC<-bestglm(Xy=Boston.bestglm,family = gaussian,IC = "AIC",nvmax =12,method = "exhaustive",CVArgs = "default", qLevel = 0.99, TopModels = 1)

select = summary(regfit.full)$outmat calculate_AIC <-c() calculate_BIC <-c() regfit.full <- regsubsets(medv~., data = Boston, method = "exhaustive", nvmax = 12,nbest=1) Y <- Boston[,13]

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 } for (i in 1:12){ Y_hat= predict(regfit.full, newdata = Boston, id = i) error = (1/length(Y))*sum((Y-Y_hat)^2) AIC<-506 * log(error) +2*i calculate_AIC<- c(calculate_AIC,AIC) BIC = 506 * log(error) + 2*i * log(506) calculate_BIC<- c(calculate_BIC,BIC) }

beta.fit <- function(X,Y){ lsfit(X,Y)
}

beta.predict <- function(fit, X){ cbind(1,X)%*%fit$coef }

sq.error <- function(Y,Yhat){ (Y-Yhat)^2 } X<-Boston[,1:12] Bootstrap_.632_error <- c() for (i in 1:12){ #Pull out the model temp <- which(select[i,] == "*")

res <- bootpred(X[,temp], Y, nboot = 50, theta.fit = beta.fit, theta.predict = beta.predict, err.meas = sq.error) Bootstrap_.632_error <- c(Bootstrap_.632_error, res[[3]])

}

graphics.off() x11() ylim = c(lower -1, upper +1) plot(Bootstrap_.632_error, type = "o", lty = 1, col = "blue" , xlab = "k", ylab = "error", main = "Bootstrap_.632_error") which(Bootstrap_.632_error == min(Bootstrap_.632_error))

jk.errors <- matrix(NA, 506, 12) for (k in 1:506){ #uses regsubsets in the data with 1 observation removed best.model.cv <- regsubsets(medv~.,data = Boston[-k,],nvmax = 12,nbest=1,method="exhaustive") #removes the kth obsv

```
  #Models with 1, 2 ,...14 predictors
for (i in 1:12){
  #that was left out
  pred <- predict(best.model.cv,                 #prediction in the obsv
                  Boston[k,],
                  id=i)
  jk.errors[k,i] <- (Boston$medv[k]-pred)^2       #error in the obsv
}
}
```

LOOCV_error <- apply(jk.errors, 2, mean)

```
 k = 5        # number of folds
set.seed(1)   # set the random seed so we all get the same results

# Assign each observation to a single fold
folds = sample(1:k, nrow(Boston), replace = FALSE)

# Create a matrix to store the results of our upcoming calculations
five_fold_cv_errors = matrix(NA, k, 12, dimnames = list(NULL, paste(1:12)))



for(j in 1:k){

# The perform best subset selection on the full dataset, minus the jth fold
best_fit = regsubsets(medv~., data = Boston[folds!=j,], nvmax=12,nbest=1,method="exhaustive")

# Inner loop iterates over each size i
for(i in 1:12){

    # Predict the values of the current fold from the "best subset" model on i predictors
    pred = predict(best_fit, Boston[folds==j,], id=i)

    # Calculate the MSE, store it in the matrix we created above
    five_fold_cv_errors[j,i] = mean(Boston$medv[folds==j]-pred)^2
}
```

} mean_five_fold_cv_errors = apply(five_fold_cv_errors, 2, mean) plot(mean_five_fold_cv_errors, type = "o", lty = 1, col = "blue" , xlab = "k", ylab = "error", main = "Five Fold Cross Validation") which(mean_five_fold_cv_errors == min(mean_five_fold_cv_errors))

```
 k = 10        # number of folds
set.seed(1)   # set the random seed so we all get the same results

# Assign each observation to a single fold
folds = sample(1:k, nrow(Boston), replace = FALSE)

# Create a matrix to store the results of our upcoming calculations
ten_fold_cv_errors = matrix(NA, k, 12, dimnames = list(NULL, paste(1:12)))



for(j in 1:k){

# The perform best subset selection on the full dataset, minus the jth fold
best_fit = regsubsets(medv~., data = Boston[folds!=j,], nvmax=12,nbest=1,method="exhaustive")

# Inner loop iterates over each size i
for(i in 1:12){

    # Predict the values of the current fold from the "best subset" model on i predictors
    pred = predict(best_fit, Boston[folds==j,], id=i)

    # Calculate the MSE, store it in the matrix we created above
    ten_fold_cv_errors[j,i] = mean(Boston$medv[folds==j]-pred)^2
}
```

} mean_ten_fold_cv_errors = apply(ten_fold_cv_errors, 2, mean) plot(mean_ten_fold_cv_errors, type = "o", lty = 1, col = "blue" , xlab = "k", ylab = "error", main = "Ten Fold Cross Validation") which(mean_ten_fold_cv_errors == min(mean_ten_fold_cv_errors))

upper = max(my_sumcp,my_sumbic,Bootstrap_.632_error,mean_five_fold_cv_errors,mean_ten_fold_cv_errors) lower = min(my_sumcp,my_sumbic,Bootstrap_.632_error,mean_five_fold_cv_errors,mean_ten_fold_cv_errors)

graphics.off() x11() plot(my_sum$cp, xlab = "No. of variables",ylab = "error",ylim = c(lower -1, upper +1),type = "l",lty=2,main = "Model Selection") lines(my_sum$bic, type = "o", lty = 1, col = "blue") lines(Bootstrap_.632_error, type = "o", lty = 3, col = "black") lines(mean_five_fold_cv_errors, type = "o", lty = 4, col = "yellow") lines(mean_ten_fold_cv_errors, type = "o", lty = 5, col = "green") legend("topright", c("AIC_Error", "BIC_Error","Bootstrap_.632_error","five_fold_cross_val","ten_fold_cross_val"), lty = c(2,1,3,4,5), col = c("blue", "red","black","orange","yellow"))

########## Task2

Task2---------- library(dplyr) library(stringr) summary(Boston) str<-summary(Boston)[3] crim_median<-strsplit(str, " : ") crim_median<-as.numeric(data.frame(crim_median)[2,1])

Boston<-Boston %>% mutate(crime_factor = case_when(Boston$crim > crim_median ~ "High", Boston$crim < crim_median ~ "Low")) which(Boston$Census =="A")==which(Boston$crim > crim_median) Boston$crime_factor<-as.factor(Boston$crime_factor)

summary(Boston) sum(is.na(Boston)) cor(Boston) correlation_data <-cor(Boston) library(corrplot) scale_this <- function(x) { (x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE) }

table_1 <- Boston %>% dplyr::select(zn:crime_factor) %>% gather(Variable, value, -crime_factor, -chas) %>% group_by(Variable) %>% mutate(value = scale_this(value)) %>% group_by(Variable, crime_factor) %>% summarize(Q10 = quantile(value, .10), Q25 = quantile(value, .25), median = median(value), mean = mean(value), Q75 = quantile(value, .75), Q90 = quantile(value, .90))

kable(table_1, digits = 3, format = 'html') %>% kable_styling(bootstrap_options = c('striped', 'hover', 'condensed')) %>% column_spec(1:2, bold = T) %>% scroll_box(height = '400px') difference <- function(x) x[1] - x[2]

table_2 <- table_1 %>% group_by(Variable) %>% summarize(diff_Q10 = difference(Q10), diff_Q25 = difference(Q25), diff_med = difference(median), diff_mean = difference(mean), diff_Q75 = difference(Q75), diff_Q90 = difference(Q90))

kable(table_2, digits = 3, format = 'html') %>% kable_styling(bootstrap_options = c('striped', 'hover', 'condensed')) %>% column_spec(1, bold = T) %>% add_header_above(c(' ' = 1, 'Between-Groups Differences' = 6)) %>% scroll_box(height = '400px')

table_3 <- table_2 %>% gather(Measure, value, -Variable) %>% group_by(Variable) %>% summarize(`Absolute Mean Differences` = abs(mean(value))) %>% arrange(desc(`Absolute Mean Differences`))

kable(table_3, digits = 3, format = 'html') %>% kable_styling(bootstrap_options = c('striped', 'hover', 'condensed')) %>% column_spec(1, bold = T) %>% scroll_box(height = '400px')

Boston %>% dplyr::select(zn:crime_factor) %>% gather(value_type, value, -crime_factor, -chas) %>% ggplot(aes(value_type, value, fill = crime_factor)) + geom_boxplot(alpha = 0.5) + facet_wrap(~value_type, scales = 'free') + scale_fill_discrete(name = 'Crime Rate') + theme(legend.position = 'top')

Boston %>% dplyr::select(crim, crime_factor, rad, nox, tax, age, dis, indus) %>% gather(Variable, value, -crim, -crime_factor) %>% mutate(Variable = str_to_title(Variable)) %>% ggplot(aes(value, crim)) + geom_point(aes(col = crime_factor)) + facet_wrap(~ Variable, scales = 'free') + geom_smooth(method = 'lm', formula = y ~ poly(x, 3), se = FALSE) + guides(col = FALSE) + labs(title = 'Scatterplots for each strong predictor')

graphics.off() x11() corrplot(correlation_data, method = "number", type = "upper", diag = FALSE) xtabs(~Census,data=Boston) graphics.off() Boston_modified<-Boston[,-1]

######Fitting with Logistic Regression

set.seed(1) train_size <- nrow(Boston_modified) * 0.75 inTrain <- sample(1:nrow(Boston_modified), size = train_size) train <- Boston_modified[inTrain,] test <- Boston_modified[-inTrain,]

log_reg <- glm(crime_factor ~ rad + nox + tax + age + dis + indus, data = train, family = binomial) pred <- predict(log_reg, test, type = 'response') pred_value <- ifelse(pred >= 0.5, 'Low', 'High') acc <- mean(pred_value == test$crime_factor) table(pred_value, test$crime_factor) %>% kable(format = 'html') %>% kable_styling() %>% add_header_above(c('Predicted' = 1, 'Observed' = 2)) %>% column_spec(1, bold = T) %>% add_footnote(label = acc)

######################Fitting data with linear discriminant analysis

library(klaR) lda_model <- lda(crime_factor ~ rad + nox + tax + age + dis, data = train) lda_pred <- predict(lda_model, test) lda_acc <- mean(lda_pred$class == test$crime_factor) table(lda_pred$class, test$crime_factor) %>% kable(format = 'html') %>% kable_styling() %>% add_header_above(c('Predicted' = 1, 'Observed' = 2)) %>% column_spec(1, bold = T) %>% add_footnote(label = lda_acc)

##############################Fitting with KNN for different K values

require(class) variables <- c('rad', 'nox', 'tax', 'age', 'dis', 'zn', 'indus') x_train <- train[, variables] y_train <- train$crime_factor x_test <- test[, variables] knn_acc <- list() for (i in 1:20) { knn_pred <- knn(train = x_train, test = x_test, cl = y_train, k = i) knn_acc[as.character(i)] = mean(knn_pred == test$crime_factor) } knn_acc <- unlist(knn_acc) data_frame(knn_acc = knn_acc) %>% mutate(k = row_number()) %>% ggplot(aes(k, knn_acc)) + geom_col(aes(fill = k == which.max(knn_acc))) + labs(x = 'K', y = 'Accuracy', title = 'KNN Accuracy for different values of K') + scale_x_continuous(breaks = 1:20) + scale_y_continuous(breaks = round(c(seq(0.90, 0.94, 0.01), max(knn_acc)), digits = 3)) + geom_hline(yintercept = max(knn_acc), lty = 2) + coord_cartesian(ylim = c(min(knn_acc), max(knn_acc))) + guides(fill = FALSE)

##############Fitting with CART

> rm(list=ls())
>
> library(ISLR2) require(MASS); require(tidyverse); require(ggthemes) require(knitr); require(kableExtra); require(corrplot) require(broom) set.seed(1) theme_set(theme_tufte(base_size = 14)) data('Boston') Boston <- Boston %>% + mutate(chas = factor(chas), + crime_factor = factor(ifelse(crim > median(crim), + 'High', 'Low'), + levels = c('High', 'Low'))) Boston_Modified<-Boson[,-1] set.seed(1) train_size <- nrow(Boston_Modified) * 0.75 inTrain <- sample(1:nrow(Boston_Modified), size = train_size) train <- Boston_Modified[inTrain,] test <- Boston_Modified[-inTrain,] library("rpart") model.controls <- rpart.control(minbucket = 2, minsplit = 4, xval = 10, cp = 0) fit_boston <- rpart(crime_factor~., data = train, control = model.controls)
>
> fit_boston$cptable graphics.off() x11() plot(fit_boston$cptable[,4], type = "o", lty = 1, col = "blue",main = "Cp for model selection", ylab = "cv error") pruned_fit_boston <- prune(fit_boston, cp = fit_boston$cptable[min_cp, 1]) plot(fit_boston, main = "Tree") text(fit_boston, cex = .5) min_cp = which.min(fit_boston$cptable[,4]) pruned_fit_boston <- prune(fit_boston, cp = fit_boston$cptable[min_cp, 1]) predict.full_tree <- predict(fit_boston, test) predict.pruned_tree <- predict(pruned_fit_boston, test) predict.full_tree_frame<-as.data.frame(predict.full_tree) y_true_test <- test$crime_factor

predicted<-predict.full_tree_frame %>% mutate(crime_factor = case_when(predict.full_tree_frame$High > predict.full_tree_frame$Low ~ "High", predict.full_tree_frame$High < predict.full_tree_frame$Low ~ "Low")) x<- which(predicted[,3]!=y_true_test) length(x) error<-length(x)/length(predicted[,3]) summary(predict.full_tree)

predict.pruned_tree_frame<-as.data.frame(predict.pruned_tree) predicted_prune_tree<-predict.pruned_tree_frame %>% mutate(crime_factor = case_when(predict.pruned_tree_frame$High > predict.pruned_tree_frame$Low ~ "High", predict.pruned_tree_frame$High < predict.pruned_tree_frame$Low ~ "Low")) x<- which(predicted_prune_tree[,3]!=y_true_test) error<-length(x)/length(predicted_prune_tree[,3])

Task 3

library(dplyr) library(stringr) summary(Auto) str<-summary(Auto)[3] mpg_median<-strsplit(str, ":") mpg_median<-as.numeric(data.frame(mpg_median)[2,1])

Auto<-Auto %>% mutate(mpg01 = case_when(Auto$mpg > mpg_median ~ 1, Auto$mpg < mpg_median ~ 0)) Auto$mpg01 <-as.factor(Auto$mpg01) Auto_new <- Auto[,1:7]

correlation_data <-cor(Auto_new)

library(corrplot) Auto_new <- Auto[,-c(8,9)] graphics.off() x11() corrplot(correlation_data, type = "upper", order = "hclust", tl.col = "black", tl.srt = 45) xtabs(~mpg01,data=Auto)

col <- c("blue", "red")[Auto_new$mpg01]

pch <- c(15,16)[Auto_new$mpg01]

graphics.off() x11() par(mfrow = c(2,3)) plot(mpg ~ cylinders, data=Auto_new, pch=pch,col=col, cex.lab=1.25, xlab="miles per gallon", ylab="Number of cylinders between 4 and 8") legend("topright",c("<med_mpg",">med_mpg"),lty=c(1,1),lwd=c(2.5,2.5),col=c("blue","red")) plot(mpg ~ displacement, data=Auto_new, pch=pch,col=col, cex.lab=1.25, xlab="miles per gallon", ylab="displacement") legend("topright",c("<med_mpg",">med_mpg"),lty=c(1,1),lwd=c(2.5,2.5),col=c("blue","red")) plot(mpg ~ horsepower, data=Auto_new, pch=pch,col=col, cex.lab=1.25, xlab="miles per gallon", ylab="Engine horsepower") legend("topright",c("<med_mpg",">med_mpg"),lty=c(1,1),lwd=c(2.5,2.5),col=c("blue","red")) plot(mpg ~ weight, data=Auto_new, pch=pch,col=col, cex.lab=1.25, xlab="miles per gallon", ylab="Vehicle weight") legend("topright",c("<med_mpg",">med_mpg"),lty=c(1,1),lwd=c(2.5,2.5),col=c("blue","red")) plot(mpg ~ year, data=Auto_new, pch=pch,col=col, cex.lab=1.25, xlab="miles per gallon", ylab="Model y ear") legend("topright",c("<med_mpg",">med_mpg"),lty=c(1,1),lwd=c(2.5,2.5),col=c("blue","red"))

graphics.off() x11() par(mfrow = c(4,2)) ggplot(Auto_new, aes(x = displacement, y =cylinders)) + geom_point()

ggplot(Auto_new, aes(x = weight, y =cylinders)) + geom_point()

ggplot(Auto_new, aes(x = mpg, y = weight)) + geom_point(aes(color = cylinders))

ggplot(Auto_new, aes(x = mpg, y = weight)) + geom_point(aes(color = horsepower))

ggplot(Auto_new, aes(x = mpg, y = cylinders)) + geom_point()

ggplot(Auto_new, aes(x = mpg, y = displacement)) + geom_point()

ggplot(Auto_new, aes(x = mpg, y = acceleration)) + geom_point(aes(color = cylinders))

ggplot(Auto_new, aes(x = mpg, y = acceleration)) + geom_point(aes(color = displacement))

```
set.seed(1)
Auto_new<-Auto[,-c(8,9)]
Auto_new<-Auto_new[,2:8]
train = sample(1:nrow(Auto_new), .80*nrow(Auto_new))
```

linear_fit <-lm(mpg ~ cylinders + horsepower + weight + displacement + year + acceleration, data = Auto) linear_fit <-lm(mpg ~ cylinders + horsepower + weight + displacement + year + acceleration, data = Auto)

library(klaR) y_true_train <- Auto_new[train,]$mpg01 y_true_test <- Auto_new[-train,]$mpg01 lda.fit = lda(mpg01 ~ weight + year + acceleration+displacement, data = Auto_new,subset =train) train_pred <-predict(lda.fit, Auto_new[train,]) test_pred <-predict(lda.fit, Auto_new[-train,]) y_hat_train <- train_pred$class y_hat_test <- test_pred$class

train_err <- (1/length(y_true_train))*length(which(y_true_train != y_hat_train)) test_err <- (1/length(y_true_test))*length(which(y_true_test != y_hat_test))

> train_err [1] 0.09904153 test_err [1] 0.1139241

####################Fit via QDA

qda.fit <- qda(mpg01~ weight + year + acceleration+displacement, data = Auto_new,subset =train) train_pred <- predict(qda.fit, Auto_new[train,]) test_pred <- predict(qda.fit, Auto_new[-train,]) y_hat_train <- train_pred$class y_hat_test <- test_pred$class

qda_train_error <- (1/length(y_true_train))*length(which(y_true_train != y_hat_train)) qda_test_error <- (1/length(y_true_test))*length(which(y_true_test != y_hat_test))

> qda_train_error [1] 0.09584665 qda_test_error [1] 0.08860759

#########Fit via Logistic Regression

##################Fit Via KNN

library(caret) library(class) set.seed(1) train.X = data.frame(Auto_new[train,]$weight,Auto_new[train,]$year,Auto_new[train,]$acceleration,Auto_new[train,]$displacement) test.X = data.frame(Auto_new[-train,]$weight,Auto_new[-train,]$year,Auto_new[-train,]$acceleration,Auto_new[-train,]$displacement) train.mpg01 = Auto_new[train,]$mpg01 test.mpg01 = Auto_new[-train,]$mpg01 knn.pred = knn(train.X, test.X,train.mpg01, k = 1) mean(knn.pred == test.mpg01)

> mean(knn.pred != train.mpg01)
>
> mean(knn.pred != test.mpg01) [1] 0.1139241 mean(knn.pred != train.mpg01) [1] 0.4984026

library(caret) library(class) set.seed(1) train.X = data.frame(Auto_new[train,]$weight,Auto_new[train,]$year,Auto_new[train,]$acceleration,Auto_new[train,]$displacement) test.X = data.frame(Auto_new[-train,]$weight,Auto_new[-train,]$year,Auto_new[-train,]$acceleration,Auto_new[-train,]$displacement) train.mpg01 = Auto_new[train,]$mpg01 test.mpg01 = Auto_new[-train,]$mpg01 knn.pred = knn(train.X, test.X,train.mpg01, k = 9)

> mean(knn.pred != test.mpg01) [1] 0.1139241 mean(knn.pred != train.mpg01) [1] 0.5111821

require(ISLR); require(tidyverse); require(ggthemes); require(GGally) require(knitr); require(kableExtra); require(broom) theme_set(theme_tufte(base_size = 14)) set.seed(1)

data('Auto') Auto <- Auto %>% filter(!cylinders %in% c(3,5)) %>% mutate(mpg01 = factor(ifelse(mpg > median(mpg), 1, 0)), cylinders = factor(cylinders, levels = c(4,6,8), ordered = TRUE), origin = factor(origin, levels = c(1,2,3), labels = c('American', 'European', 'Asian'))) median(Auto$mpg)

Auto %>% dplyr::select(-name, -mpg) %>% ggpairs(aes(col = mpg01, fill = mpg01, alpha = 0.6), upper = list(combo = 'box'), diag = list(discrete = wrap('barDiag', position = 'fill')), lower = list(combo = 'dot_no_facet')) + theme(axis.text.x = element_text(angle = 90, hjust = 1))

Auto %>% dplyr::select(-name, -mpg, - origin, -cylinders) %>% gather(Variable, value, -mpg01) %>% mutate(Variable = str_to_title(Variable)) %>% ggplot(aes(mpg01, value, fill = mpg01)) + geom_boxplot(alpha = 0.6) + facet_wrap(~ Variable, scales = 'free', ncol = 1, switch = 'x') + coord_flip() + theme(legend.position = 'top') + labs(x = '', y = '', title = 'Variable Boxplots by mpg01')

set.seed(1) num_train <- nrow(Auto) * 0.75

inTrain <- sample(nrow(Auto), size = num_train)

training <- Auto[inTrain,] testing <- Auto[-inTrain,]

#############fit with LDA model

require(MASS) fmla <- as.formula('mpg01 ~ displacement + horsepower + weight + year + cylinders') lda_model <- lda(fmla, data = training)

pred <- predict(lda_model, testing) table(pred$class, testing$mpg01) mean(pred$class == testing$mpg01) mean(pred$class != testing$mpg01)

#############fit with QDA model

qda_model <- qda(fmla, data = training)

pred <- predict(qda_model, testing) table(pred$class, testing$mpg01) mean(pred$class != testing$mpg01)

#############fit with Logistic Regression model

log_reg <- glm(fmla, data = training, family = binomial)

pred <- predict(log_reg, testing, type = 'response') pred_values <- round(pred) table(pred_values, testing$mpg01)

mean(pred_values != testing$mpg01)

fit with KNN

require(class) set.seed(1) acc <- list()

x_train <- training[,c('cylinders', 'displacement', 'horsepower', 'weight', 'year')] y_train <- training$mpg0 x_test <- testing[,c('cylinders', 'displacement', 'horsepower', 'weight', 'year')]

for (i in 1:20) { knn_pred <- knn(train = x_train, test = x_test, cl = y_train, k = i) acc[as.character(i)] = mean(knn_pred == testing$mpg01) } acc <- unlist(acc)

data_frame(acc = acc) %>% mutate(k = row_number()) %>% ggplot(aes(k, acc)) + geom_col(aes(fill = k == which.max(acc))) + labs(x = 'K', y = 'Accuracy', title = 'KNN Accuracy for different values of K') + scale_x_continuous(breaks = 1:20) + scale_y_continuous(breaks = round(c(seq(0.90, 0.94, 0.01), max(acc)), digits = 3)) + geom_hline(yintercept = max(acc), lty = 2) + coord_cartesian(ylim = c(min(acc), max(acc))) + guides(fill = FALSE)

```
for (i in 1:20) {
knn_pred <- knn(train = x_train, test = x_test, cl = y_train, k = i)
acc[as.character(i)] = mean(knn_pred =!testing$mpg01)
```

} not_acc <- list() for (i in 1:20) { knn_pred <- knn(train = x_train, test = x_test, cl = y_train, k = i) not_acc[as.character(i)] = mean(knn_pred != testing$mpg01) } not_acc <- unlist(not_acc)

data_frame(not_acc = not_acc) %>% mutate(k = row_number()) %>% ggplot(aes(k, not_acc)) + geom_col(aes(fill = k == which.max(not_acc))) + labs(x = 'K', y = 'Error', title = 'KNN Error for different values of K') + scale_x_continuous(breaks = 1:20) + scale_y_continuous(breaks = round(c(seq(0.90, 0.94, 0.01), max(not_acc)), digits = 3)) + geom_hline(yintercept = max(not_acc), lty = 2) + coord_cartesian(ylim = c(min(not_acc), max(not_acc))) + guides(fill = FALSE)