```
#define own knn

L2distance = function(v,u){
  if (length(v) != length(u)){
    stop('Vectors are be of the different length')
  }
  else{
    return(sum(abs(u-v)))
  }
}

#find k nearest points neighbours from inp among x[1],...,x[n]
nearest_nb = function(x, inp, k, d=L2distance){
  if(ncol(x) != length(inp)){
    stop('Data have different number of variables')
  }
  dist = 1:nrow(x)
  for(i in 1:nrow(x)){
    dist[i] = d(x[i,],inp)
  }
  dist.sort = sort(dist)
  closest_k = dist.sort[1:k]
  knearest_ind = which(dist %in% closest_k)
  #print(c(length(knearest_ind),dist.sort[1]))
  return(knearest_ind)
}

#self-defined knn function
knn_own = function(trainX, testX, trainY, k){
  all_pred = c()
  for(i in 1:nrow(testX)){
    #print(length(testX[i,]))
    nbs = nearest_nb(trainX, testX[i,], k)
    #print(length(nbs))
    prediction = mean(trainY[nbs])
    all_pred[i] = round(prediction,0)
  }
  return(all_pred)
}

X <- as.matrix(read.table(gzfile("/Users/nm/Documents/GitHub/Data
Mining/Assign1/zip.train.gz")))
y2or3 <- which(X[, 1] == 2 | X[, 1] == 3)
train.X <- X[y2or3, -1]
train.Y <- X[y2or3, 1] == 3

X <- as.matrix(read.table(gzfile("/Users/nm/Documents/GitHub/Data
Mining/Assign1/zip.train.gz")))
y2or3 <- which(X[, 1] == 2 | X[, 1] == 3)
test.X <- X[y2or3, -1]
test.Y <- X[y2or3, 1] == 3


#Linear regression
lr.fit <- lm(train.Y ~ train.X)
prediction_test <- round(cbind(1, test.X) %*% lr.fit$coef,0)
prediction_train <- round(cbind(1, train.X) %*% lr.fit$coef,0)
errortest.lr <- mean(prediction_test != test.Y)*100
errortrain.lr <- mean(prediction_train != train.Y)*100

#using the inbuilt knn function
k <- 1:20
library(class)
errortest.knn <- numeric(length(k))
errortrain.knn <- numeric(length(k))
for (i in 1:length(k)) {
  prediction_test <- knn(train.X, test.X, train.Y, k[i])
  prediction_train <- knn(train.X, train.X, train.Y, k[i])
  errortest.knn[i] <- mean(prediction_test != test.Y)*100
```

```
  errortrain.knn[i] <- mean(prediction_train != train.Y)*100
}
```

```
#using my own knn function
errortest.knn_own <- numeric(length(k))
errortrain.knn_own <- numeric(length(k))
for (i in 1:length(k)) {
  prediction_test <- knn_own(train.X, test.X, train.Y, k[i])
  prediction_train <- knn_own(train.X, train.X, train.Y, k[i])
  errortest.knn_own[i] <- mean(prediction_test != test.Y)*100
  errortrain.knn_own[i] <- mean(prediction_train != train.Y)*100
}
```

```
error <- matrix(c(errortest.lr, errortest.knn, errortest.knn_own, errortrain.lr,
errortrain.knn, errortrain.knn_own), ncol = 2)
colnames(error) <- c("Test Error(%)", "Train Error(%)")
rownames(error) <- c("Linear Regression", paste("inbuilt k-NN with k =", k), paste("own
k-NN with k =", k))
print(error)
```

```
y.limit = c(min(c(errortest.knn_own, errortest.knn, errortest.lr)),
            max(c(errortest.knn_own, errortest.knn,errortest.lr)))
plot(k,errortest.knn, type='o', lty = 2, ylim=y.limit, col="green", ylab="testing
error(%)")
points(k, errortest.knn_own, type="o", lty=2, pch = "*", col="blue")
abline(h = errortest.lr, col="red", lty = 3)
legend("bottomright", legend=c("Inbuilt knn", "My knn", "Linear regression"),
       col=c("green", "blue","red"), lty=c(2,2,3), cex=0.8)

y.limit = c(min(c(errortrain.knn_own, errortrain.knn,errortrain.lr)),
            max(c(errortrain.knn_own, errortrain.knn,errortrain.lr)))
plot(k,errortrain.knn, type='o', lty = 2, ylim=y.limit, col="green", ylab="training
error(%)")
points(k, errortrain.knn_own, type="o", lty=2, pch = "*", col="blue")
abline(h = errortrain.lr, col="red", lty = 3)
legend("bottomright", legend=c("Inbuilt knn", "My knn", "Linear regression"),
       col=c("green", "blue","red"), lty=c(2,2,3), cex=0.8)
```