```
library(ElemStatLearn)
```

```
## Warning: package 'ElemStatLearn' was built under R version 3.2.5
```

```
zip.train.23 = zip.train[zip.train[, 1] %in% c(2,3), 1:257]
dim(zip.train.23)
```

```
## [1] 1389  257
```

```
zip.test.23 = zip.test[zip.test[, 1] %in% c(2,3), 1:257]
dim(zip.test.23)
```

```
## [1] 364 257
```

```
library(kknn)
```

```
## Warning: package 'kknn' was built under R version 3.2.5
```

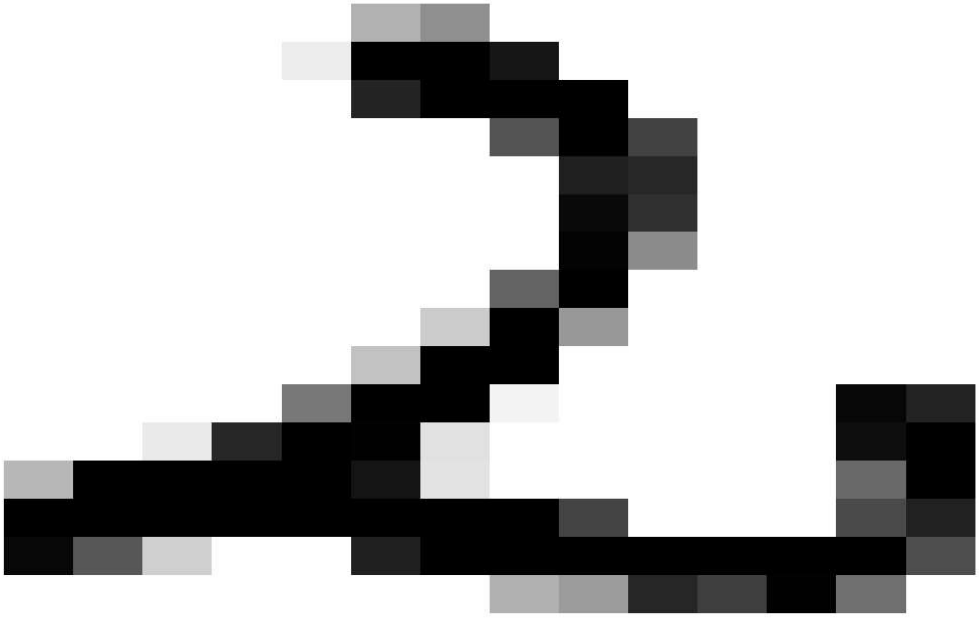
```
library(class)
set.seed(1)
```

Statement:- In this problem we are working on **Handwritten Digit Recognition Data** but we are restricting our analysis to digit 2 and 3. Training data is retrieved from **zip.train** and test data is retrieved from **zip.test** dataset from the **ElemStatLearn** library.

Example: As mentioned in the problem statement we have restricted our analysis to digits 2 and 3. To illustrate the kind of training data that we are dealing with, here is the image of one of the handwritten digit:

```
image(zip2image(zip.train.23, 6), col=gray(256:0/256), zlim=c(0,1), xlab="", ylab="", axes =
FALSE)
```

```
## [1] "digit  2  taken"
```



Methodology

1. knn: k-nearest neighbour classification for test set from training set. For each row of the test set, the k nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the kth nearest vector, all candidates are included in the vote.

2. 10 Fold cross validation: A 10-fold CV is carried out as follows: *Randomly split the data into 10 equal sized subsamples* Fit the model using 9 out of 10 subsamples as training data and calculate the testing error using the remaining one. *Alternate the testing sample, and average the total of 10 experiments

Part a

For the part(a), we perform 10 fold cross validation to find the best k, we use knn function as this problem requires classification algorithm.

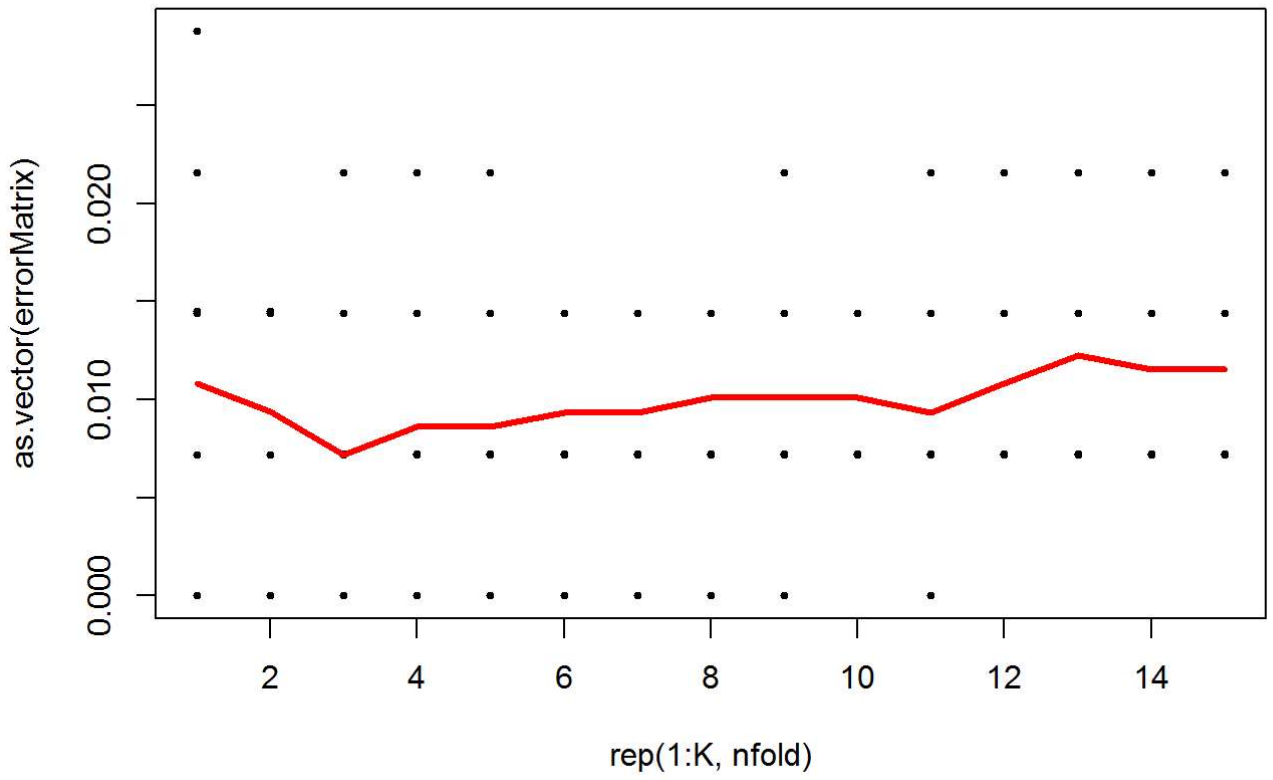
```
nfold = 10
infold = sample(rep(1:nfold, length.out=length(zip.train.23[,1])))



K = 15 # maximum number of k that I am considering
errorMatrix = matrix(NA, K, nfold) # save the prediction error of each fold

for (l in 1:nfold)
{
    for (k in 1:K)
    {
        knn.fit.e1.23<- knn(zip.train.23[infold !=l, 2:257], zip.train.23[infold==l, 2:257],
zip.train.23[infold !=l, 1], k=k)
    errorMatrix[k,l]=1-mean(knn.fit.e1.23==zip.train.23[infold==l,1])

    }
}
# plot the results
k=15
plot(rep(1:K, nfold), as.vector(errorMatrix), pch = 19, cex = 0.5)
points(1:K, apply(errorMatrix, 1, mean), col = "red", pch = 19, type = "l", lwd = 3)
```



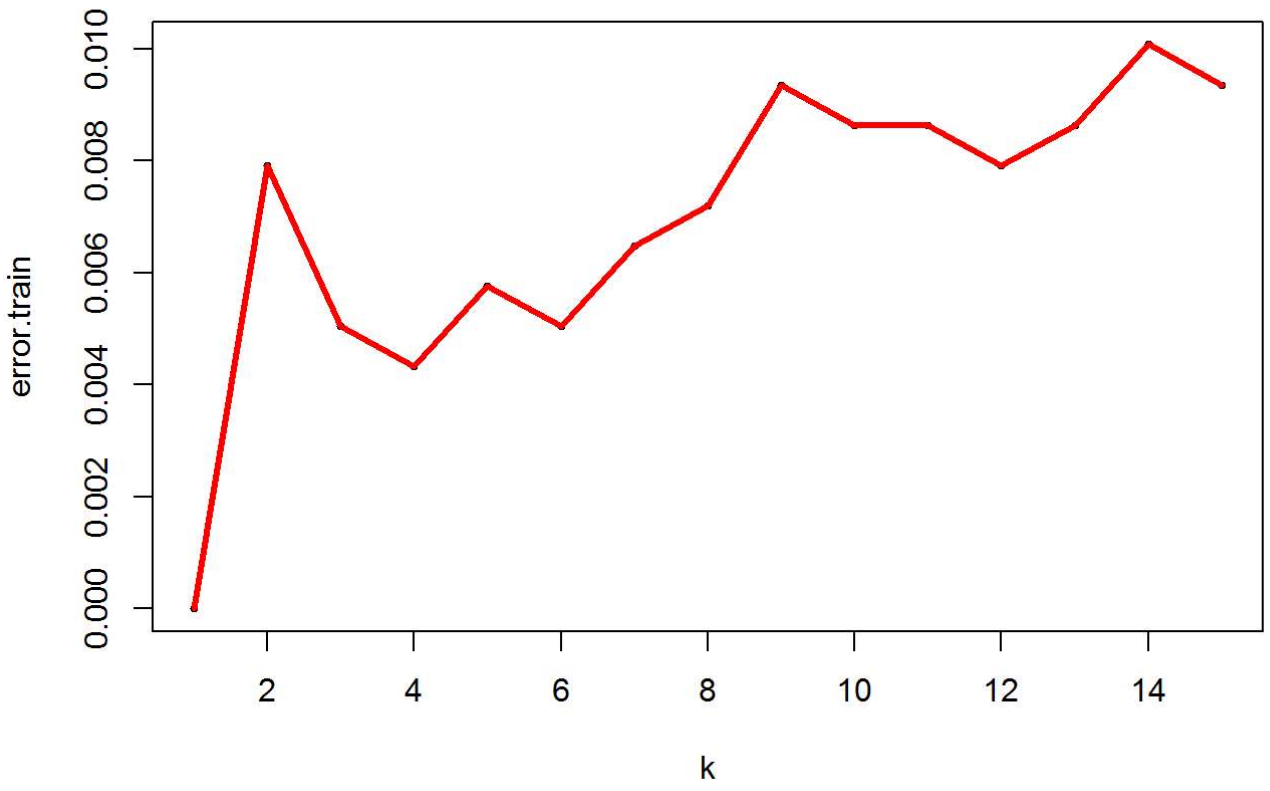
Results: The plot shows the cross validation error with respect to k values. We find that cross validation error is really small and is more or less same of any value of k. Hence we conculde that cross validation error is independent of k. We can choose any k.

Part b

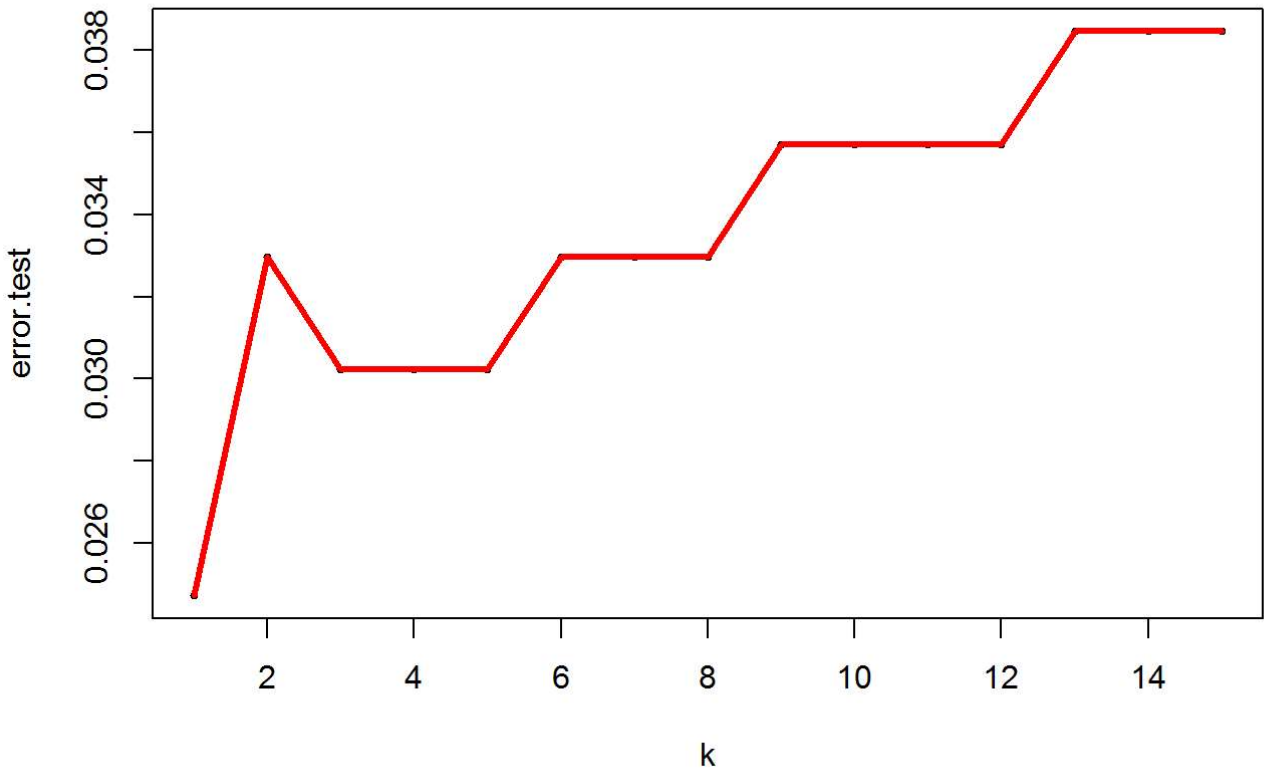
For this section, we will use kNN for the whole training dataset and test our results over testing dataset. We iterate k from 1 to 15 and find the test and training error for each iteration.

```
set.seed(2)

error.test<- vector(length=15)
error.train<-vector(length=15)
for (k in 1:15)
{
  knn.fit.23<- knn(zip.train.23[, 2:257], zip.test.23[, 2:257], zip.train.23[, 1], k=k)
  knn.fit.train.23<-knn(zip.train.23[, 2:257], zip.train.23[, 2:257], zip.train.23[, 1], k=k)
  error.test[k]=1-mean(knn.fit.23==zip.test.23[,1])
  error.train[k]=1-mean(knn.fit.train.23==zip.train.23[,1])
}
k=1:15
plot(k,error.train,pch = 19, cex = 0.5)
points(k, error.train, col = "red", pch = 19, type = "l", lwd = 3)
```



Result Above is the plot for training error with respect to different values of k. As expected training error is 0 for k=1 and increases as the k increases.



Result Given above is the plot for testing error with respect to different values of k. We select k based on least testing error.