BUSN41204 Week 1 Review Session

Boxiang (Shawn) Lyu

1/7/2022

Introduction

Welcome to the first review session for BUSN41204! Today we will go over the Philly Crime dataset, which appeared in a previous midterm. We will stick with R for demonstration.

The write-up is available on Canvas in both `.pdf` *and* `.Rmd`*. In order to replicate the results yourself, please also download the* `PhillyCrime.csv` *dataset from Canvas.*

Free Resources for Learning R

Here are some free online resources for learning R:

- Codecademy
- R for Beginners by Emmanuel Paradis.
- Swirl, an interactive program for learning R.
- Additional Resources.

Ed

- Please post any question you may have on the Ed platform. You will probably get a faster response from the TAs, Professor Kolar, or one of your peers. Making your questions public can also benefit your classmates who have similar questions.

Getting Started

- What is R?
  - R is an open-source, free programming language popular in data science.
  - R has many packages, code written by other users that implement classic machine learning algorithms, such as kNN, Lasso.
  - R is user-friendly and easy to pickup.
- Why R instead of Python?
  - Python users may use the online Jupyter Notebook Server instead.
  - Python: swiss army knife. R: letter opener. Sometimes you don't need every bell and whistle.
- What is R Studio?

  - An interface for viewing R code, variables, and plots. All in the same place.
  - We *strongly recommend* using the R Studio Server instead of installing locally. Please login using your *CNetID*, rather than BoothID.
  - But if you want, you can also download RStudio here and download R here.
    * Setting up R on your local device can be time consuming (Windows users may have to deal with environmental variables).

- What is R Markdown?
  - A package `rmarkdown` + Rstudio + latex: converts your code and writeup into pdf, or html, or presentation.
  - You may use R Markdown to prepare your homework submissions. (For learning markdown, here is a great resource. R Markdown is basically markdown with chunks of R code.) A word of caution: R Markdown takes time to get used to and please use R Markdown *only if* you feel like spending the extra time to learn it.
  - Please feel free to use Word and paste the R outputs to the word doc. In most cases, preparing a document where you paste results from R will be sufficient.
  - R Markdown tutorials can be found here and here.
  - A template can be found here.
  - Again, you *do not* have to use R Markdown.
  - Setting up R Markdown on your local device can also be time consuming.

Installing and using packages in R

Use the function `install.packages()` and type the name of the package you want to install, *with quotation marks*, between the parenthesis.

```
install.packages('swirl')
```

DO NOT include the `install.packages()` function in your R markdown writeup. Otherwise your Rmarkdown file won't compile (cannot be knit into pdf).

For using packages, use the `library()` function.

```
library('swirl')
```

```
##
## | Hi! Type swirl() when you are ready to begin.
```

A more detailed guide for installing packages in R can be found here.

If you wish to install R locally, you may need to run the following script to install the required packages for your local machine. *The R Studio on the server comes with all these packages installed.*

```
packageNames = c("MASS", "ISLR", "animation", "ElemStatLearn", "glmnet",
                 "textir", "nnet", "methods", "statmod", "stats", "graphics",
                 "RCurl", "jsonlite", "tools", "utils", "data.table", "gbm",
                 "ggplot2", "randomForest", "tree", "class", "kknn", "e1071",
                 "data.table", "recommenderlab")

for (pkgName in packageNames) {
  if (!(pkgName %in% rownames(installed.packages()))) {
    install.packages(pkgName,
```

```
                      dependencies=TRUE,
                      repos='http://cran.rstudio.com')
  }
}

update.packages(ask=FALSE)
```

Setting Working Directory

A good practice is to keep all the files associated with a project in separate directories (folders). It is usually a good idea to make a separate folder for each homework assignment, final project, review session, etc. You can browse the files you have on the `Files` tab, bottom-right window.

You can use the following command to tell R that you are working in a specific directory.

```
setwd('~/w23 review session 1')
```

Alternatively, click the folder for the project you want and then click `More` (a small blue gear icon). Select `Set As Working Directory` from the drop down menu.

You can verify that your working directory is setup correctly by: 1. Running the following command

```
getwd()
```

```
## [1] "/home/blyu/w23 review session 1"
```

2. Go the the `Files` tab in the bottom-right window. Click `More` and select `Go To Working Directory` from the drop down menu.

Uploading Files to R Studio Server

1. Open up the R Studio webpage.
2. Go to the bottom right window and click on the `Files` tab.
3. Click `Upload`. It looks like a whiteboard with a tiny bronze "up" button on the top left corner.

Let's do that for the `PhillyCrime` dataset which we will later use!

Getting Help

- Again, please use Ed!
- You can also view the documentation using the command `help()`

```
help(swirl)
help(length)
```

- Alternatively, Google it! There are a lot of amazing free resources online that could probably answer your questions better than the TAs can.

Crimes in Philly

The dataset `PhillyCrime.csv` contains information of two types of crime incidents (`Vandalism` and `Thefts`) occurred in Philadelphia in 2016[1].

In this problem, we will use k nearest neighbor to predict the category of the crimes[2].

1. Make two separate scatterplots of the crime incidents using their latitudes (`X`) and longitudes (`Y`), one for `Vandalism` and one for `Thefts`. Describe any pattern or difference you find from the scatterplots. (You might want to decrease the size of the points so that they don't overlap too much.)

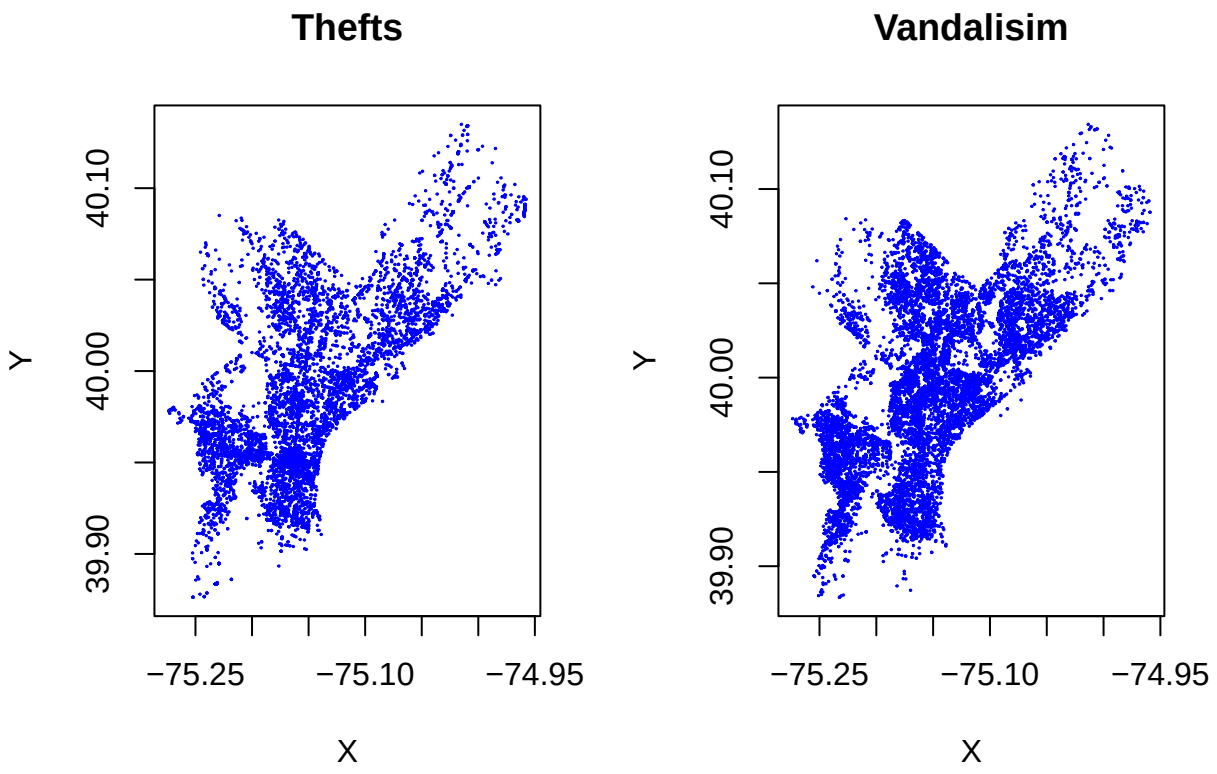
Downloading helper files and importing required libraries.

```
set.seed(100)
download.file("https://raw.githubusercontent.com/ChicagoBoothML/HelpR/master/docv_classification.R",
              destfile = "docv_classification.R")
source("docv_classification.R")
PhillyCrime <- read.csv("PhillyCrime.csv")
library(MASS)
library(kknn)
```

```
par(mfrow = c(1,2))
thefts = PhillyCrime[PhillyCrime$Category=="Thefts",]
plot(thefts$X, thefts$Y, cex = 0.1, xlab = "X", ylab = "Y",
     main = "Thefts", col = "blue")
vandalism = PhillyCrime[PhillyCrime$Category=="Vandalism",]
plot(vandalism$X, vandalism$Y, cex = 0.1, xlab = "X", ylab = "Y",
     main = "Vandalisim", col = "blue")
```

[1] The whole dataset is available at https://www.opendataphilly.org/dataset/crime-incidents

[2] Code for this problem is shamelessly borrowed from Jianeng Xu's writeup from a previous iteration of the course



2. Split the data, with 50% into a training set and the other 50% reserved for a validation set. Fit a set of k-nearest neighbors classifiers for $k = 1, 2, ..., 100$ on the training set to predict the crime category using latitude (`X`) and longitude (`Y`) as predictors.

    1. Make a scatterplot of the misclassification rate on the validation set against the parameter k.

```
compute_error = function(train, test, k_vec){
  error_vec = c()
  for(i in 1:length(k_vec)){
    fit = kknn(Category ~ X + Y, train, test, k = k_vec[i],
               kernel = "rectangular")
    results = table(fit$fitted.values, test$Category)
    print(results)
    error_vec[i] = (results[1,2] + results[2,1]) / dim(test)[1]
  }
  return(error_vec)
}
```

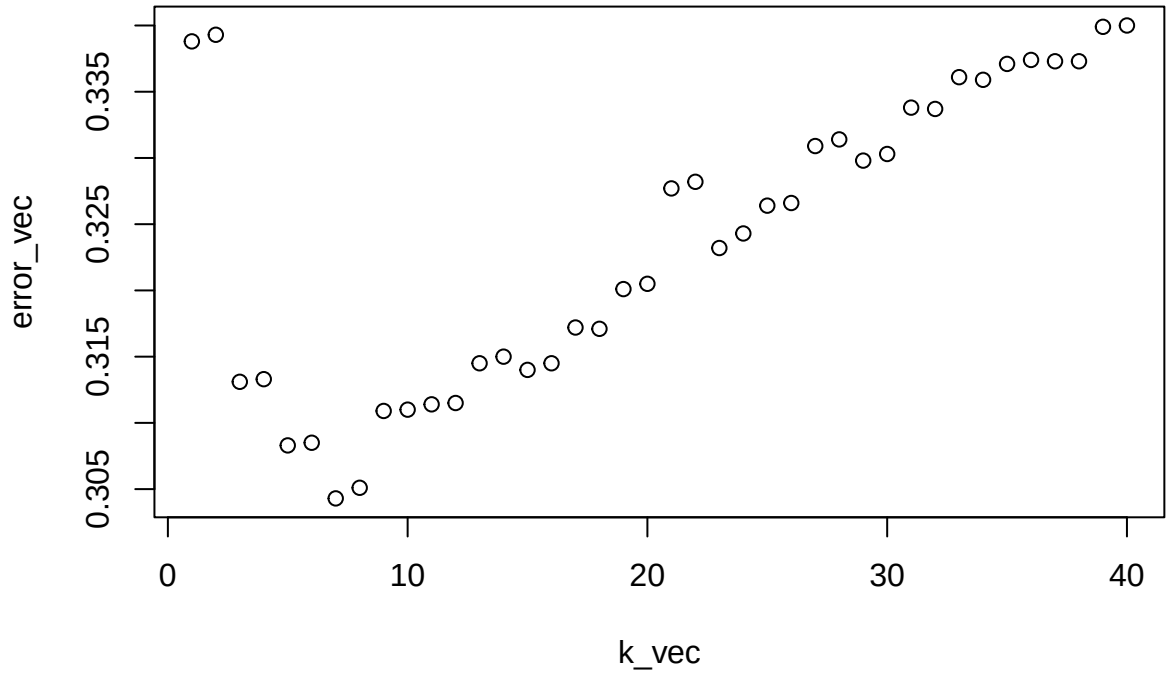
```
If you have problem running `kknn` and R spits out:
`Error in if (response != "continuous") { : argument is of length zero`,
please run the following code chunk
```

```
PhillyCrime$Category = as.factor(PhillyCrime$Category)
```

```
It tells R to treat the variable `Category` as a categorical variable, rather
than a string.
```

```
# Splitting the data into 50% training 50% testing.
N = dim(PhillyCrime)[1]  # Number of samples
```

```
train_ratio = 0.5
train_index = sample(N,train_ratio*N)
train = PhillyCrime[train_index,]
test = PhillyCrime[-train_index,]
# We only focus on k = 1 to 40 to save time
k_vec =  1:40
error_vec = compute_error(train, test, k_vec)
plot(k_vec, error_vec)
```



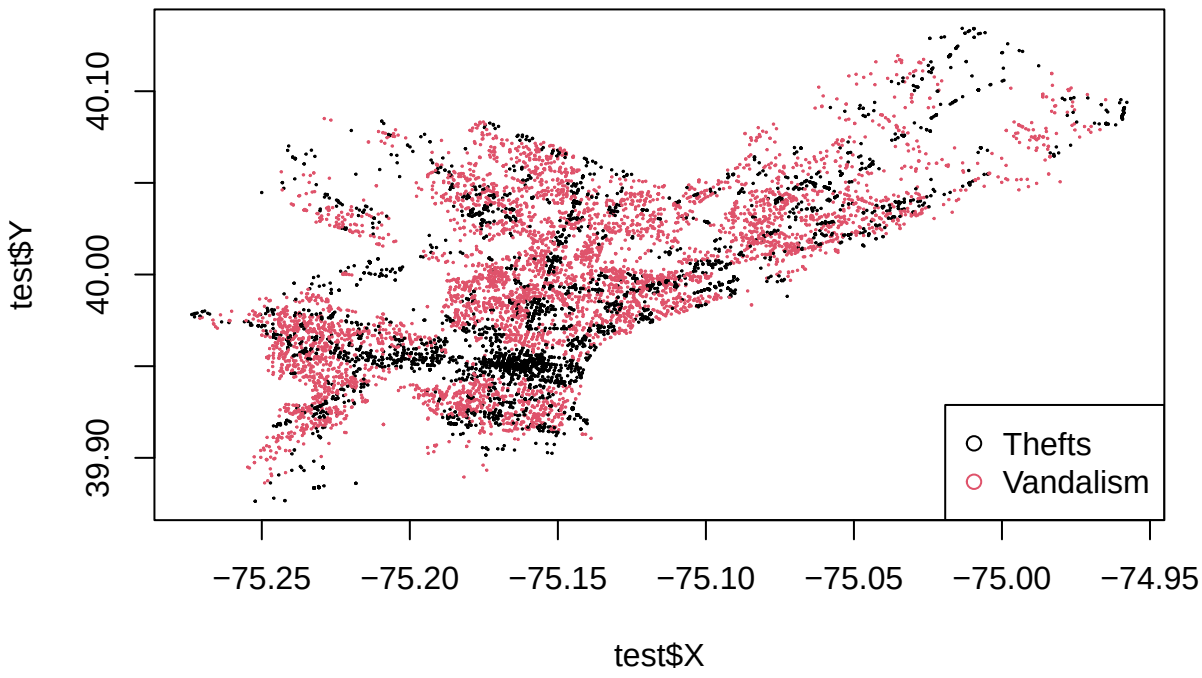
2. Report the optimal k selected by the validation-set approach and the minimum misclassification rate on the validation set.

```
best_k = k_vec[which.min(error_vec)]
best_k
```

```
## [1] 7
```

3. Make a single scatterplot of the crime incidents using their latitudes (`X`) and longitudes (`Y`). Color the points according to their predicted class given by the optimal k-nearest neighbors selected above.

```
knn = kknn(Category~X+Y,train,test, k=best_k, kernel = "rectangular")
plot(test$X, test$Y, col = as.numeric(knn$fitted.values), cex = 0.1)
legend("bottomright", c("Thefts","Vandalism"), col = c(1,2), pch = c(1,1))
```



3. Repeat (2) for 20 times (with 20 random splits of the dataset).

```
train_ratio = 0.5
error_mat = c()
for (j in 1:20) {
  train_index = sample(N,train_ratio*N)
  train = PhillyCrime[train_index,]
  test = PhillyCrime[-train_index,]
  k_vec = 1:40
  error_vec_temp = compute_error(train, test, k_vec)
  error_mat = rbind(error_mat, error_vec_temp)
}

best_k_1 = c()
min_error_1 = c()

for (i in 1:dim(error_mat)[1]) {
  best_k_1[i] = k_vec[which.min(error_mat[i,])]
  min_error_1[i] = min(error_mat[i,])
}
```

1. Report the 20 optimal k's selected by the validation sets.

```
best_k_1
```

```
##  [1] 15 11 12 10  7 10 10  9  9  5  7 10  8 10  9  6 10  9  7  9
```

2. Report the average of the minimum out-of-sample misclassification rates

as well as its standard error.

The minimal misclassification rate for each split is

```
min_error_1
```

```
##  [1] 0.3146 0.3149 0.3155 0.3174 0.3126 0.3173 0.3099 0.3153 0.3124 0.3170
## [11] 0.3194 0.3202 0.3123 0.3167 0.3140 0.3235 0.3168 0.3135 0.3107 0.3141
```

The average and standard error of minimal misclassification rate are

```
mean(min_error_1)
```

```
## [1] 0.315405
```

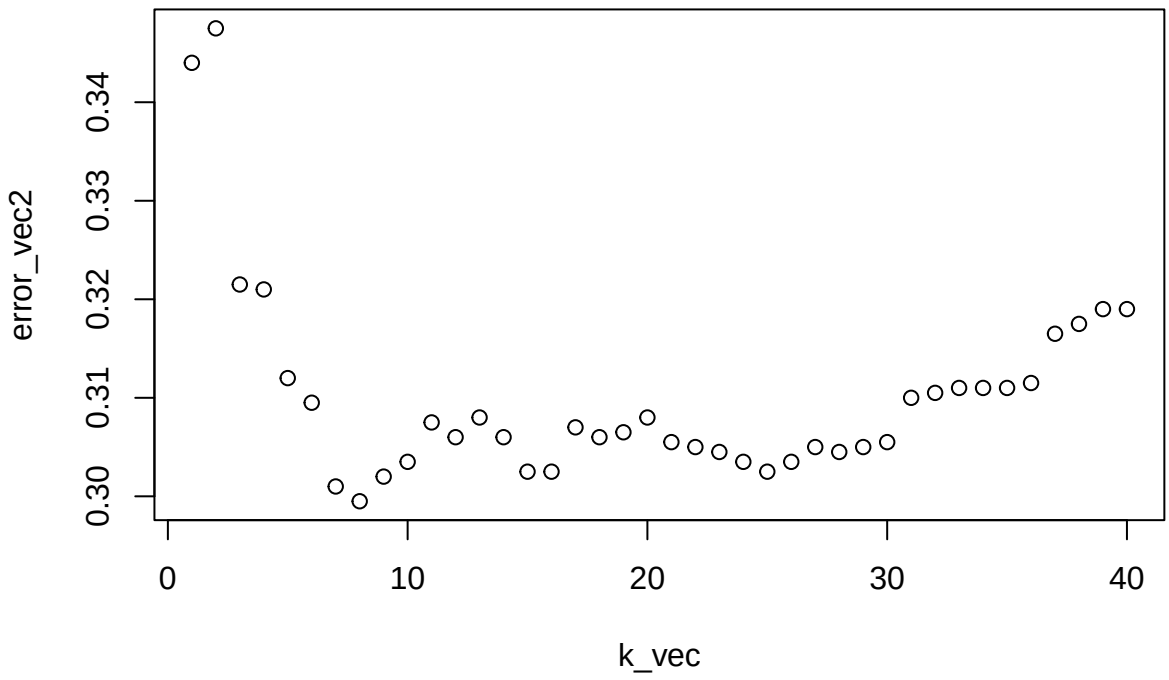
and

```
sd(min_error_1)
```

```
## [1] 0.003288173
```

4. Repeat (2) and (3), this time with a 90/10 split of the data. Make sure you include in your solutions
    1. (For a particular split,) a scatterplot of the misclassification rate on the validation set against the parameter k.

```
train_ratio = 0.9
N = dim(PhillyCrime)[1]
train_index = sample(N,train_ratio*N)
train = PhillyCrime[train_index,]
test = PhillyCrime[-train_index,]
k_vec = 1:40
error_vec2 = compute_error(train, test, k_vec)
plot(k_vec, error_vec2)
```



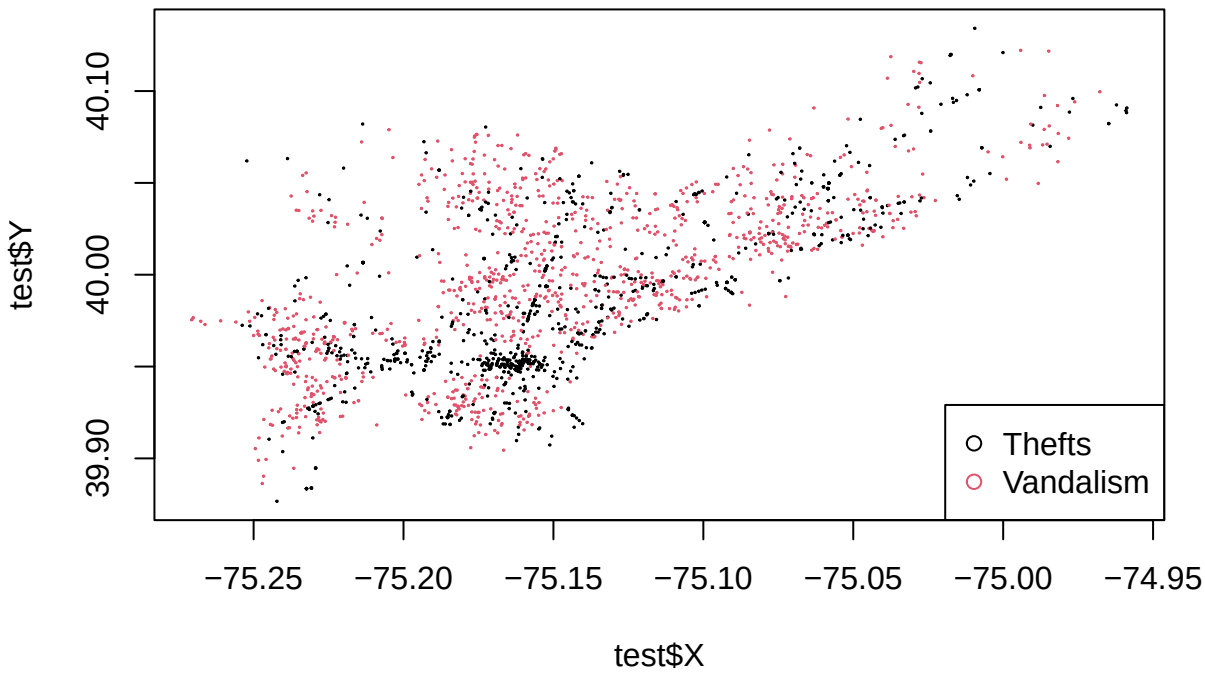
2. (For a particular split,) a scatterplot of the crime incidents (`Y` vs `X`) with points colored according to predicted class given by the optimal k.

```
best_k = k_vec[which.min(error_vec2)]
best_k
```

```
## [1] 8
```

The scatterplot can be seen below.

```
knn = kknn(Category~X+Y,train,test, k=best_k, kernel = "rectangular")
plot(test$X, test$Y, col = as.numeric(knn$fitted.values), cex = 0.1)
legend("bottomright", c("Thefts","Vandalism"), col = c(1,2), pch = c(1,1))
```



3. 20 optimal k's selected by the validation sets.

```
train_ratio = 0.9
error_mat_2 = c()
for(j in 1:20){
  train_index = sample(N,train_ratio * N)
  train = PhillyCrime[train_index,]
  test = PhillyCrime[-train_index,]
  k_vec = 1:40
  error_vec_temp = compute_error(train, test, k_vec)
  error_mat_2 = rbind(error_mat_2, error_vec_temp)
  }

best_k_2 = c()
min_error_2 = c()
for(i in 1:dim(error_mat_2)[1]){
  best_k_2[i] = k_vec[which.min(error_mat_2[i,])]
  min_error_2[i] = min(error_mat_2[i,])
  }
```

4. Average minimum out-of-sample misclassification rates and its standard error.

```
print(best_k_2)
```

```
##  [1] 17 20 17 25  9  7  9  9  5  7 10  9  9 24 13  4 13  9  7  7
```

```
print(min_error_2)
```

```
##  [1] 0.3075 0.2900 0.2895 0.3115 0.3065 0.2890 0.2885 0.2865 0.2960 0.2915
## [11] 0.2855 0.2955 0.2880 0.2920 0.3050 0.3110 0.2965 0.3075 0.2985 0.2905
```

```
# average minimum out-of-sample misclassification rates
print(mean(min_error_2))
```

```
## [1] 0.296325
```

```
# standard error of out-of-sample misclassification rates
print(sd(min_error_2))
```

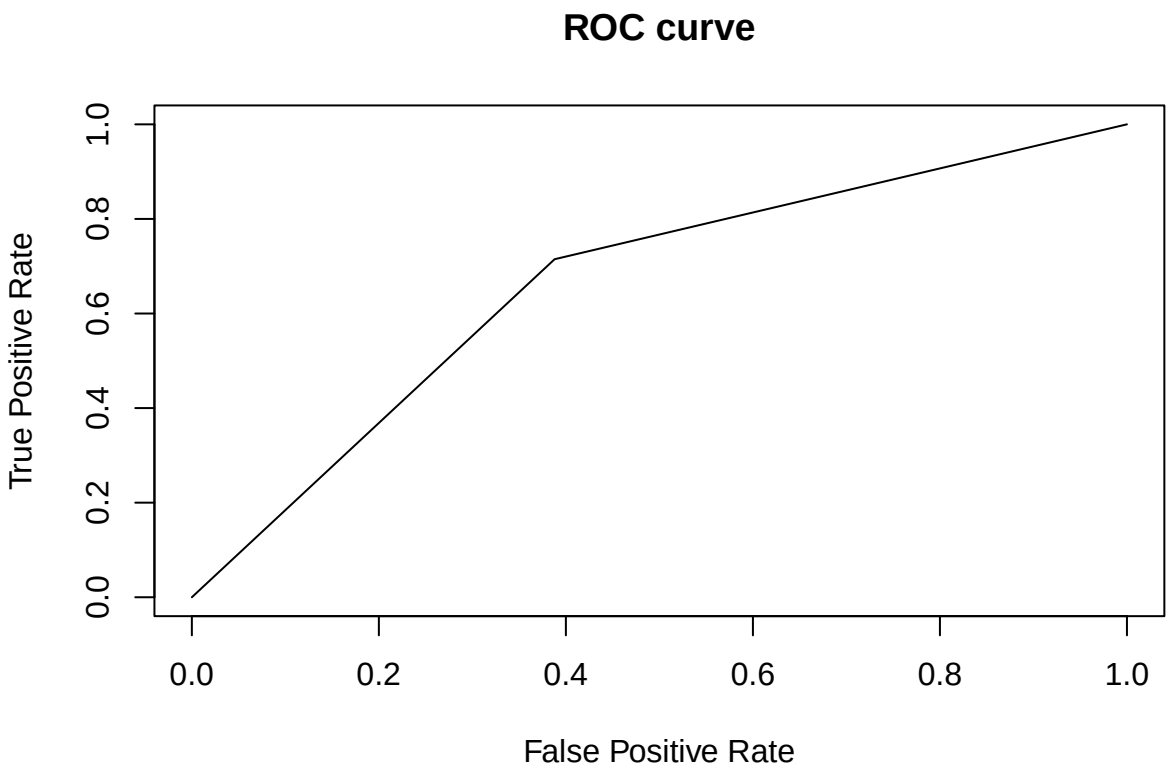
```
## [1] 0.008706131
```

5. Comment on the difference between the results obtained in (2), (3) and (4).

   - We repeat validation multiple times in question 2.3. We can see that the optimal k varies across different random splits, which is due to us randomly splitting the testing sets.
   - In question 2.4 (10 percent testing set), the average of optimal k is slightly higher than that of 50/50 case (question 2.3). Because in Q2.4 the training set is bigger, there are more data points in the neighbors.
   - Also in question 2.4 we get smaller average error but larger variance of errors across different runs. The lower mean is due to larger training set because there are more data in the neighbors of testing set, which might increase prediction accuracy. The bigger variance is due to smaller size of testing set.

6. For a 25-nearest neighbors classifier trained on 50% of the data in `PhillyCrime.csv`, plot the ROC curve calculated on a validation set with the other 50% of the data.

```
library(ROCR)
train_ratio = 0.5
train_index = sample(N,train_ratio * N)
train = PhillyCrime[train_index,]
test = PhillyCrime[-train_index,]
knn = kknn(Category~X+Y,train,test, k=25, kernel = "rectangular")
pred = prediction(as.numeric(knn$fitted.values), as.numeric(test$Category))
perf = performance(pred, measure = "tpr", x.measure = "fpr")
plot(c(0,1),c(0,1),xlab='False Positive Rate', ylab='True Positive Rate',main="ROC curve",cex.lab=1,typ
lines(perf@x.values[[1]], perf@y.values[[1]])
```


ROC curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0