Review Session 1

kNN and cross validation

1/6/2018

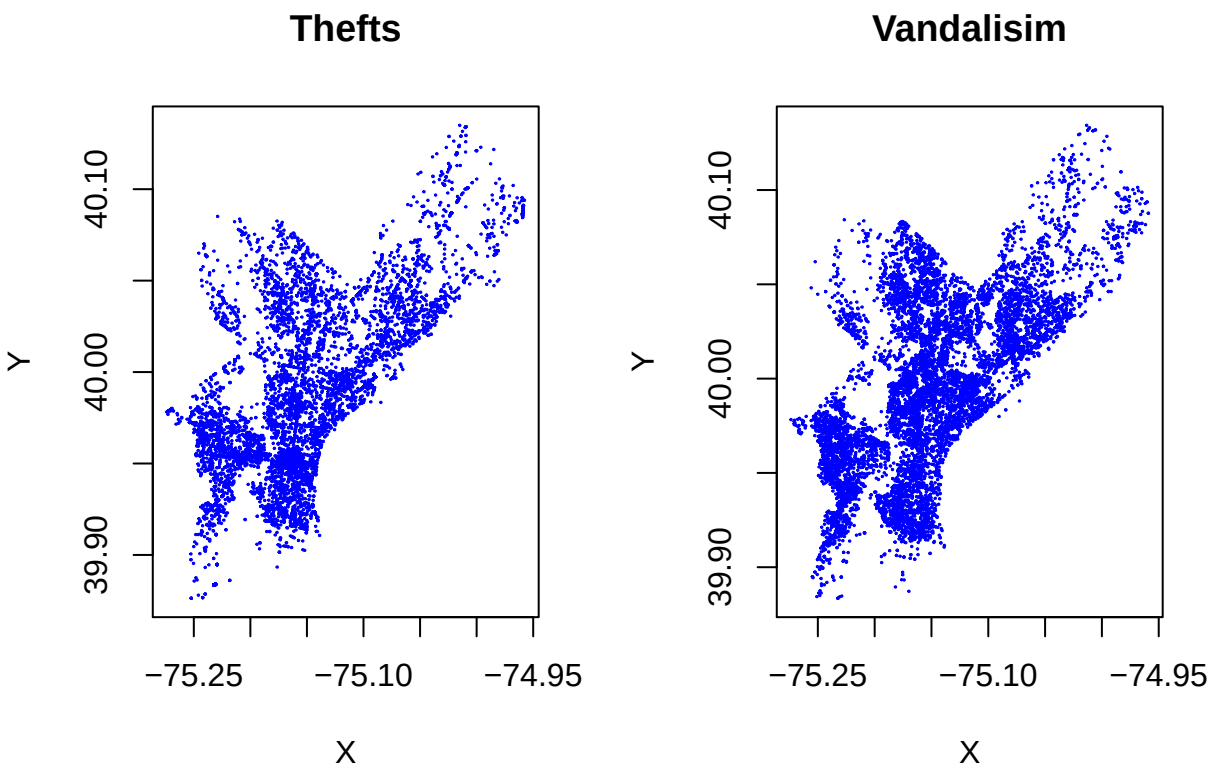
The dataset `PhillyCrime.csv` contains information of two types of crime incidents (`Vandalism` and `Thefts`) occurred in Philadelphia in 2016.[1] In this problem, we will use k nearest neighbor to predict the category of the crimes.

1 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.*)

```
set.seed(100)
download.file("https://raw.githubusercontent.com/ChicagoBoothML/HelpR/master/docv_classifi
              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 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 $k = 1, 2, \ldots, 100$ on the training set to predict the crime category using latitude (X) and longitude (Y) as predictors.

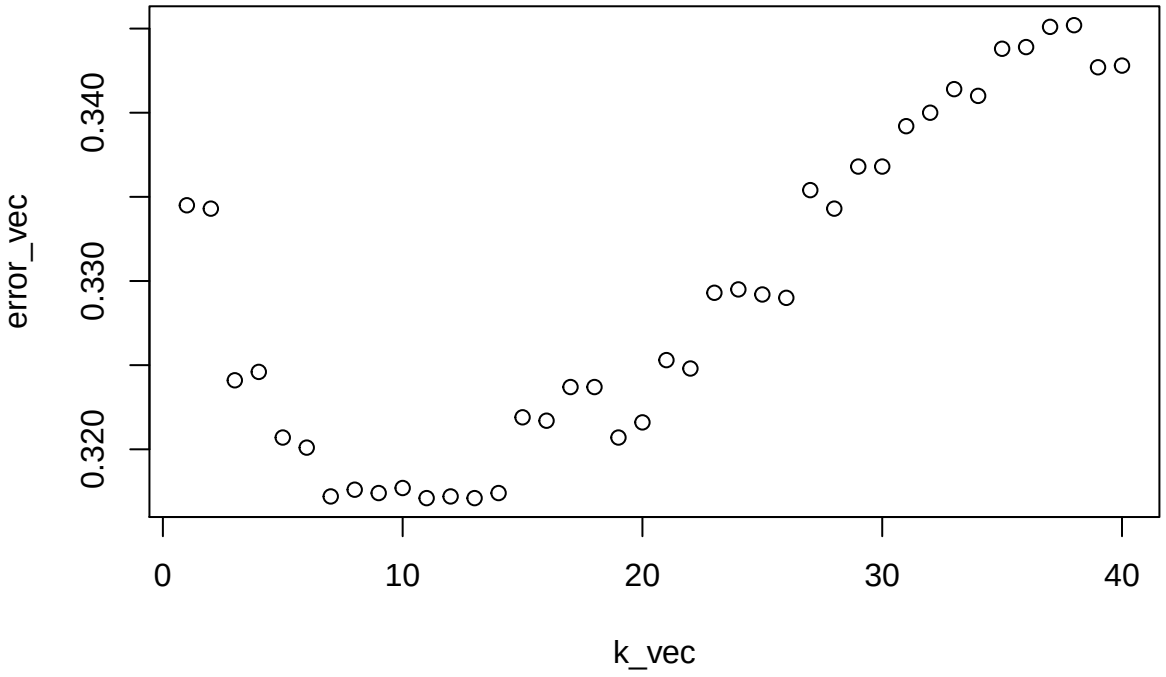
2.1 2.a

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)
    error_vec[i] = (results[1,2] + results[2,1]) / dim(test)[1]
  }
  return(error_vec)
}

N = dim(PhillyCrime)[1]
train_ratio = 0.5
train_index = sample(N,train_ratio*N)
train = PhillyCrime[train_index,]
test = PhillyCrime[-train_index,]
```

```
k_vec =  1:40
error_vec = compute_error(train, test, k_vec)
plot(k_vec, error_vec)
```



2.2 2.b

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

The optimal k is

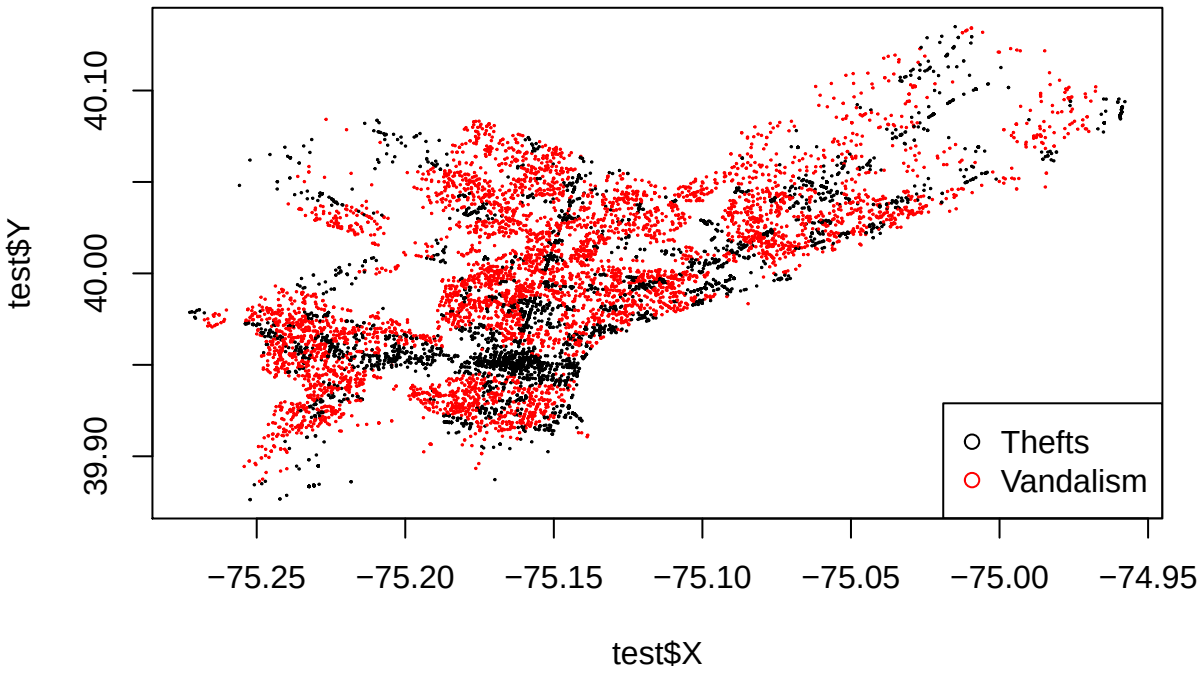
```
best_k = k_vec[which.min(error_vec)]
best_k
```

```
## [1] 11
```

2.3 2.c

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 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,])
}
```

3.1 3.a

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

The optimal k for all 20 CVs are

```
best_k_1
```

```
##  [1]  7  7 12  6 10  9 10  7  7  7 18 16 13  8  5 14  6 11 10  7
```

3.2 3.b

Report the average of the minimum out-of-sample misclassification rates as well as its standard error.

The minimal misclassification rate for each CV is

```
min_error_1
```

```
##  [1] 0.3167 0.3128 0.3179 0.3152 0.3152 0.3132 0.3141 0.3073 0.3158 0.3185
## [11] 0.3091 0.3187 0.3168 0.3101 0.3102 0.3123 0.3167 0.3149 0.3168 0.3197
```

The average and standard error of minimal misclassification rate are

```
mean(min_error_1)
```

```
## [1] 0.3146
```

```
sd(min_error_1)
```

```
## [1] 0.003430206
```

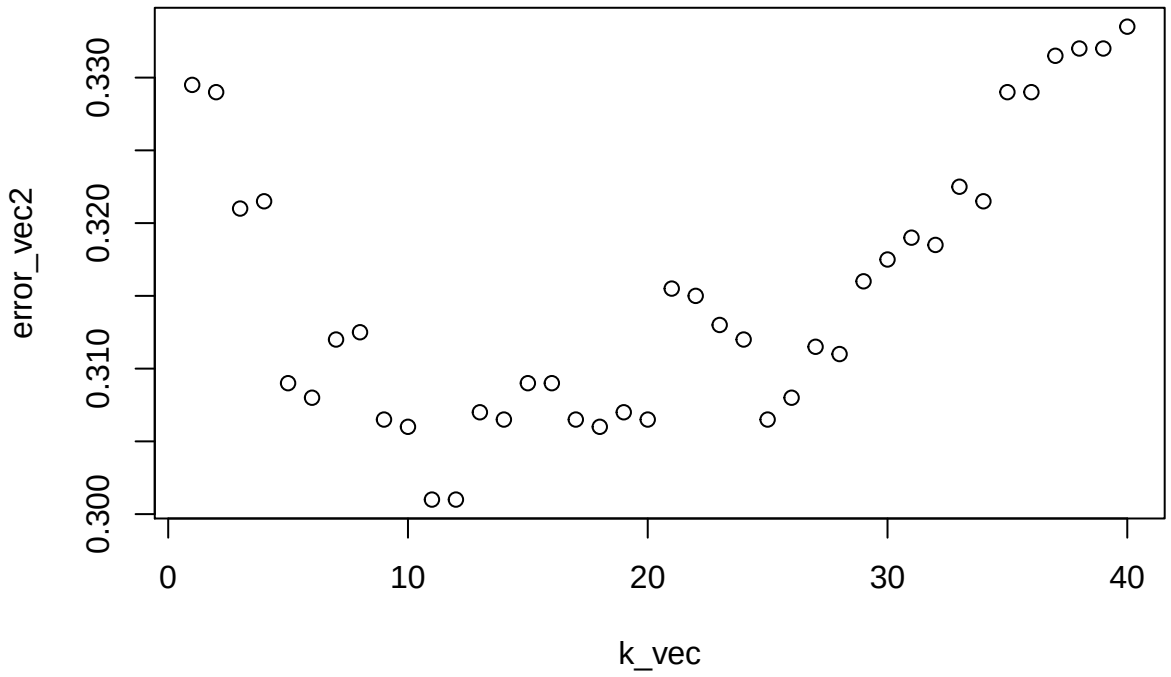
4 4

Repeat (2) and (3), this time with a 90/10 split of the data. Make sure you include in your solutions

4.1 4.a

(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)
```



4.2 4.b

(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.

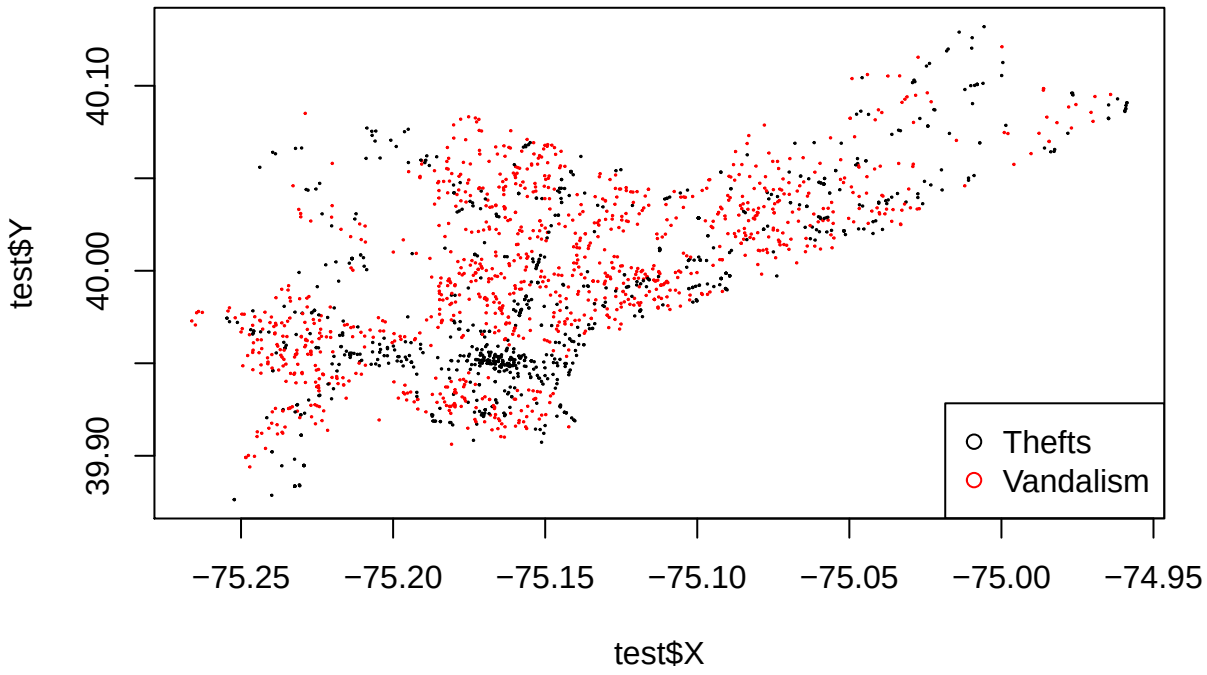
The optimal k is

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

```
## [1] 11
```

The plot of predictions by knn model with optimal k is

```
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))
```



4.3 4.c

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,])
}
```

The optimal k for all 20 CVs are

```
best_k_2
```

```
##  [1] 13 11  9 16 21 17  6 25 10 19 17 21  5 10  7 14 14 25 10 21
```

4.4 4.d

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

The minimal misclassification rate for each CV is

```
min_error_2
```

```
##  [1] 0.2990 0.2955 0.3010 0.2885 0.2960 0.3095 0.2955 0.2945 0.2920 0.2970
## [11] 0.2980 0.2920 0.2955 0.2905 0.3085 0.2895 0.2855 0.3135 0.3000 0.2880
```

The average and standard error of minimal misclassification rate are

```
mean(min_error_2)
```

```
## [1] 0.296475
```

```
sd(min_error_2)
```

```
## [1] 0.007374342
```

5 5

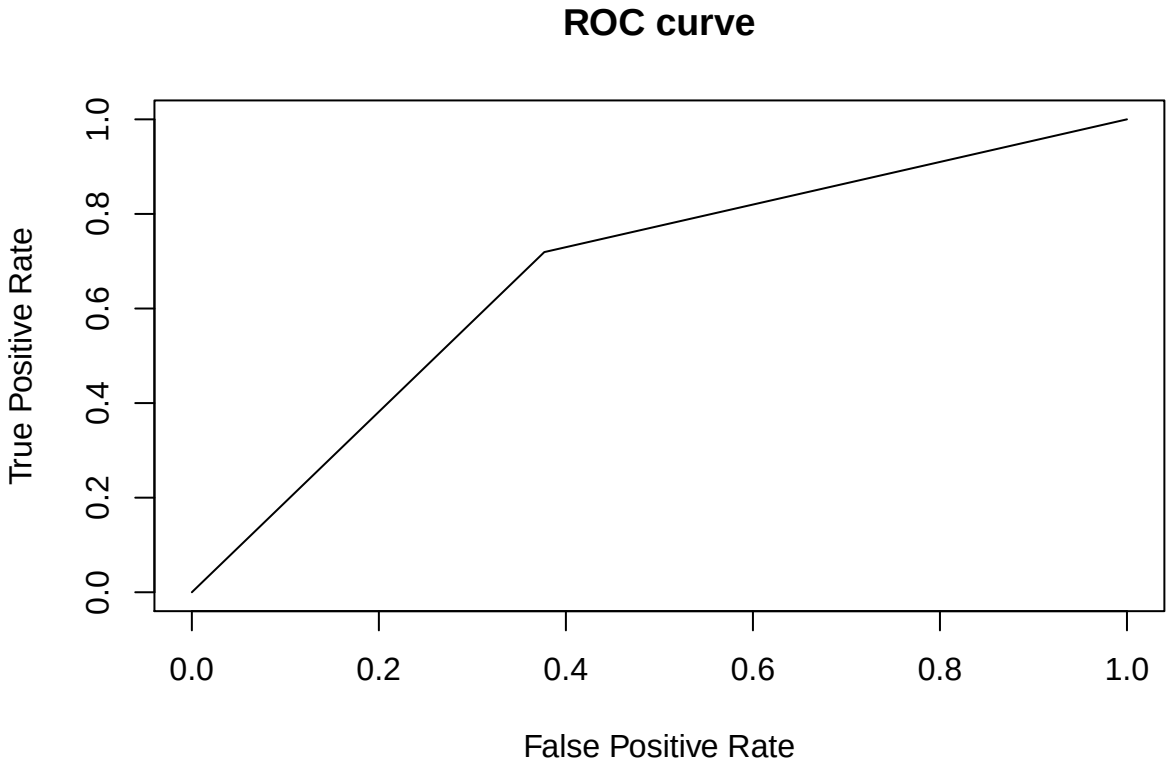
Comment on the difference between the results obtained in (2), (3) and (4).

1. We repeat CV multiple times in question 2.3. We can see that the optimal k varies across different CVs, which is due to random spliting testing sets.
2. 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 trainning set is bigger, there are more data points in the neighbours.
3. 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 neighbours of testing set, which might increase prediction accuracy. The bigger variance is due to smaller size of testing set.

6 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,type="n")
lines(perf@x.values[[1]], perf@y.values[[1]])
```


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