# Statistical Machine Leaning: Assigement 2

Darix SAMANI SIEWE

14/12/2024

##Exercise 1: Practical SML on DNA Microarrays (60 points)

**Load the dataset**

1. Comment on the shape of this dataset in terms of the sample size and the dimensionality of the input space

```
## [1] 79
```

```
## [1] 500
```

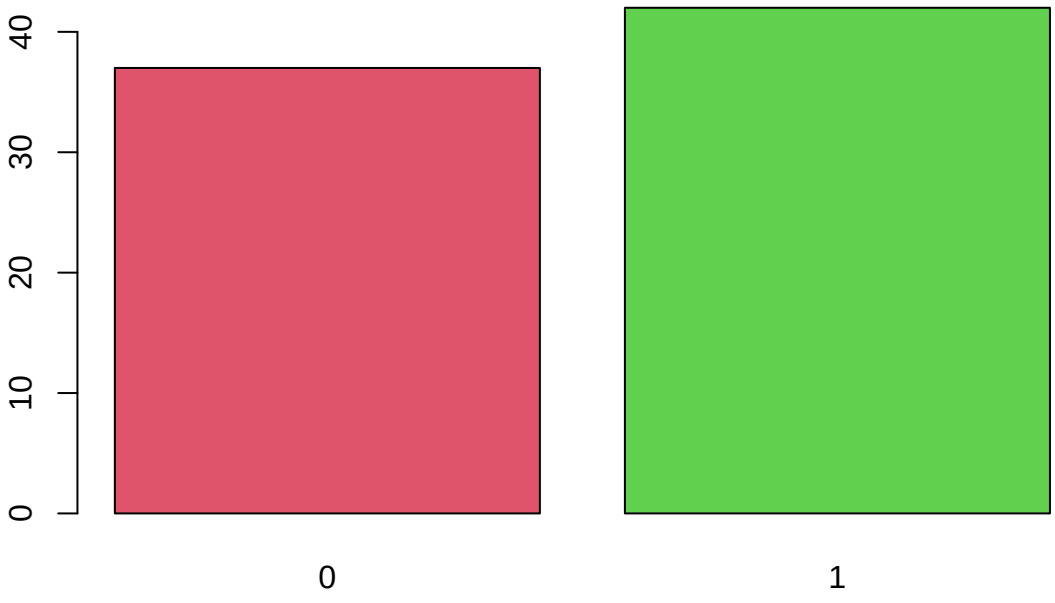
```
## Kapper (size divided by input space) is: 0.158
```

In this case the the number of features is greater than the number of the sample> the Kapper of this dataset is less that 1 which means we are in the case of ultra hight dimension. Since the number of features is very big, that means we need to reduce the dimension of the input space by selected the relevant input that capture more response variable using method like PCA or LASSO.

2. Comment succinctly from the statistical perspective on the type of data in the input space

- The values range from negative to positive, indicating a normalization or transformation step, such as standardization or log transformation.
- Min/Max: The minimum and maximum values of most variables range from roughly -0.4 to +0.5, suggesting relatively small deviations from the mean.
- most of the variable has the 1 quantile near to zero which means there the symmetric distribution in the central tendency.
- Looking at the distribution of almost the variable we can see that variable seems to follow normal distribution or log-normal distribution
- All the type of the input variable are continuous

2. Plot the distribution of the response for this dataset and comment.

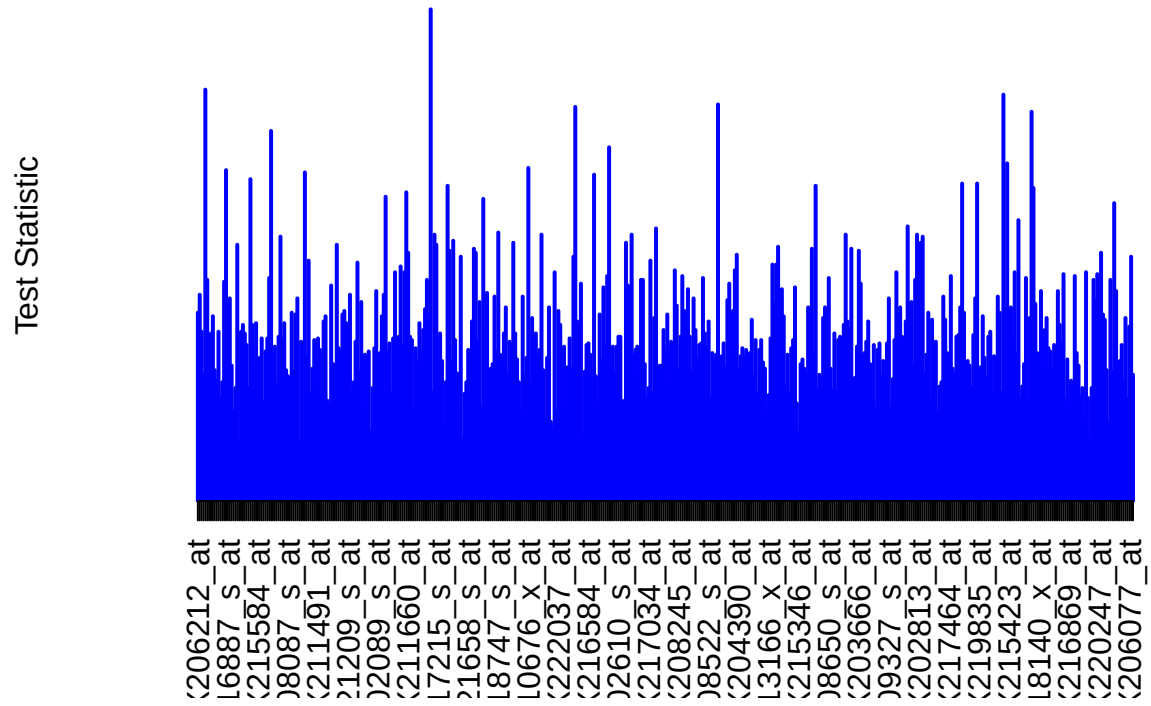



Based on the distribution above we can see that our dataset is balanced which means we don't need method like stratified holdout.

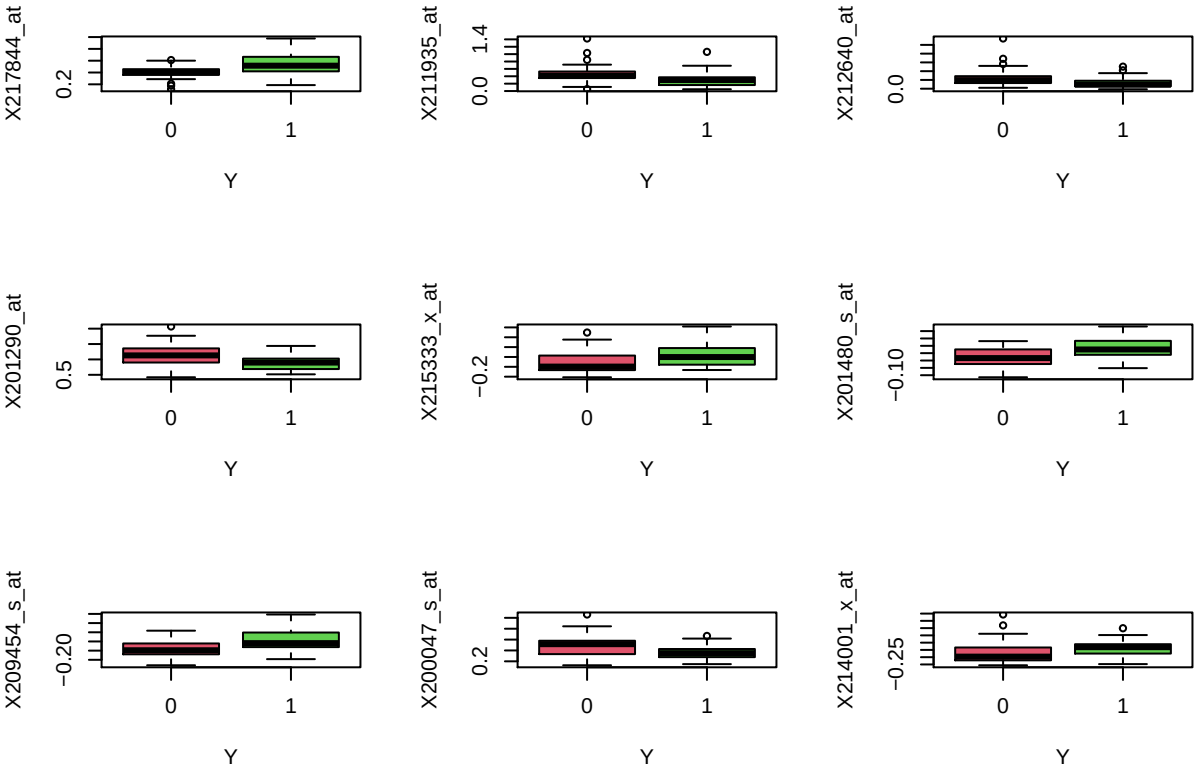
3. Identify the 9 individually most powerful predictor variables with respect to the response according the Kruskal-Wallis test statistic

```
##                 predictor      p_value
## X217844_at     X217844_at 0.0001128710
## X211935_at     X211935_at 0.0003757596
## X212640_at     X212640_at 0.0004048549
## X201290_at     X201290_at 0.0004694650
## X215333_x_at X215333_x_at 0.0004870575
## X201480_s_at X201480_s_at 0.0005241021
## X209454_s_at X209454_s_at 0.0007001308
## X200047_s_at X200047_s_at 0.0008977434
## X214001_x_at X214001_x_at 0.0011460333
```

4. Generate a type='h' plot with the Kruskal-Wallis test statistic as the y-axis and the variable name as the x-axis



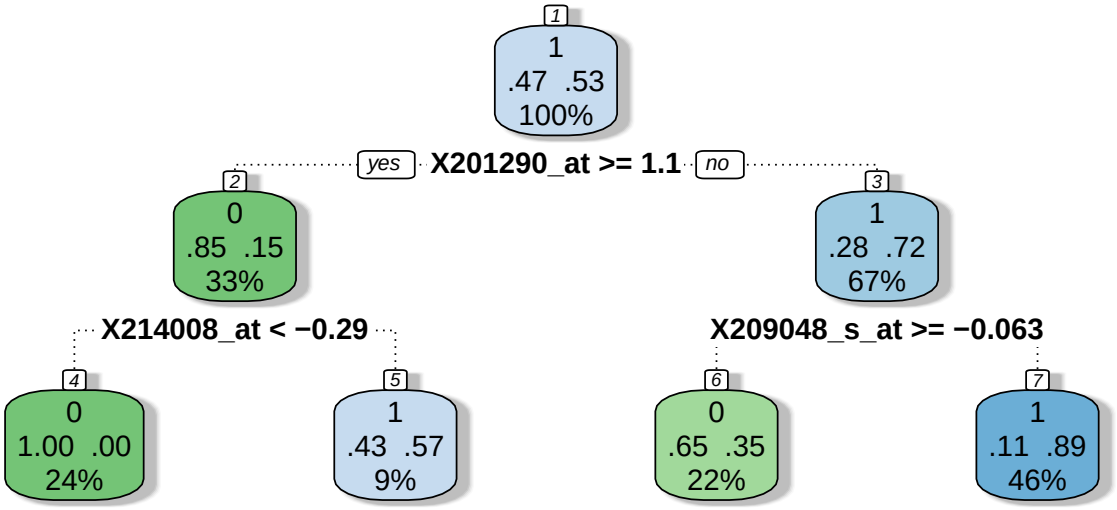

5. Generate the comparative boxplots of the 9 most powerful variable with respect to the response and comment on what you observe.




The plot above show us that our variable "X201290_at" in the input space split mostly data, which means this variable is the variable that certainly can be in the root of the tree.

7. Build the classification tree with cp=0.01

```
## Loading required package: tibble
## Loading required package: bitops
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```




Tree with cp 0.01

- Determine the number of terminal nodes

  Based on the plot above the number of terminal nodes is 4

- Write down in mathematical form region 2 and Region 4.
  - The expression of Region 2 is :

$$R_2 = \{x \quad in \quad \mathcal{X} \quad X2021290\_at \geq 1.097 \quad and \quad X214008 \geq -0.2916\}$$

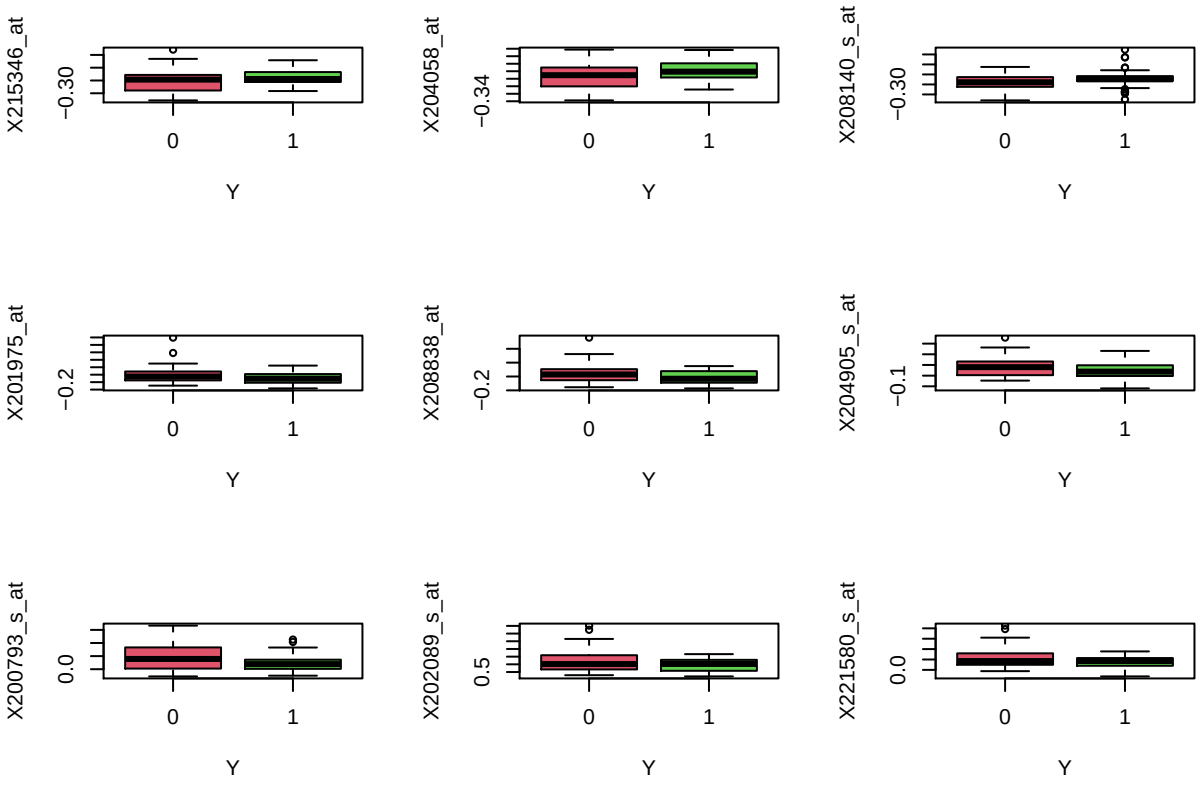
  - the expression of region 4 is:

$$R_4 = \{x \quad in \quad \mathcal{X} \quad X2021290\_at < 1.097 \quad and \quad X209048\_s\_at < 0.06313\}$$

- Comment on the variable at the root of the tree in light of the Kruskal-Wallis statistic

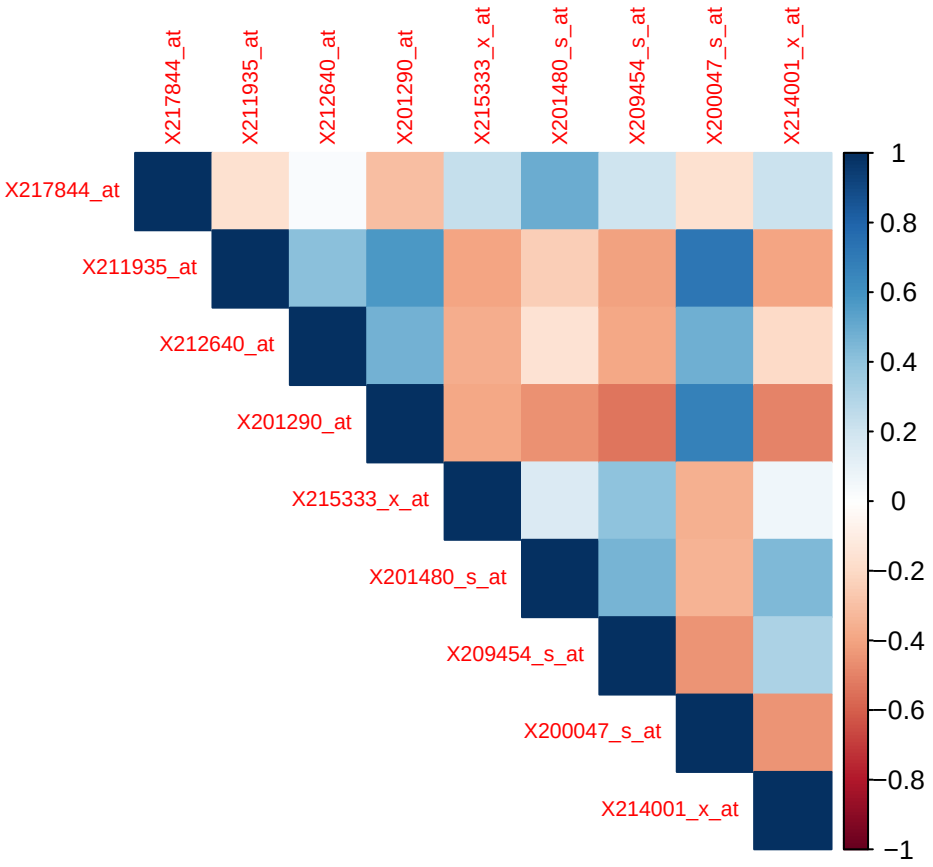
  we see that the variable in the root meet our expectation because the test statistic Kruskal-Wallis revel that the variable in the root well split our dataset into two classes.

8. Generate the comparative boxplots of the 9 weakest variable with respect to the response and comment on what you observe.




9. Generate the correlation plot of the predictor variables and comment extensively on what they reveal, if anything.

```
## corrplot 0.94 loaded
```



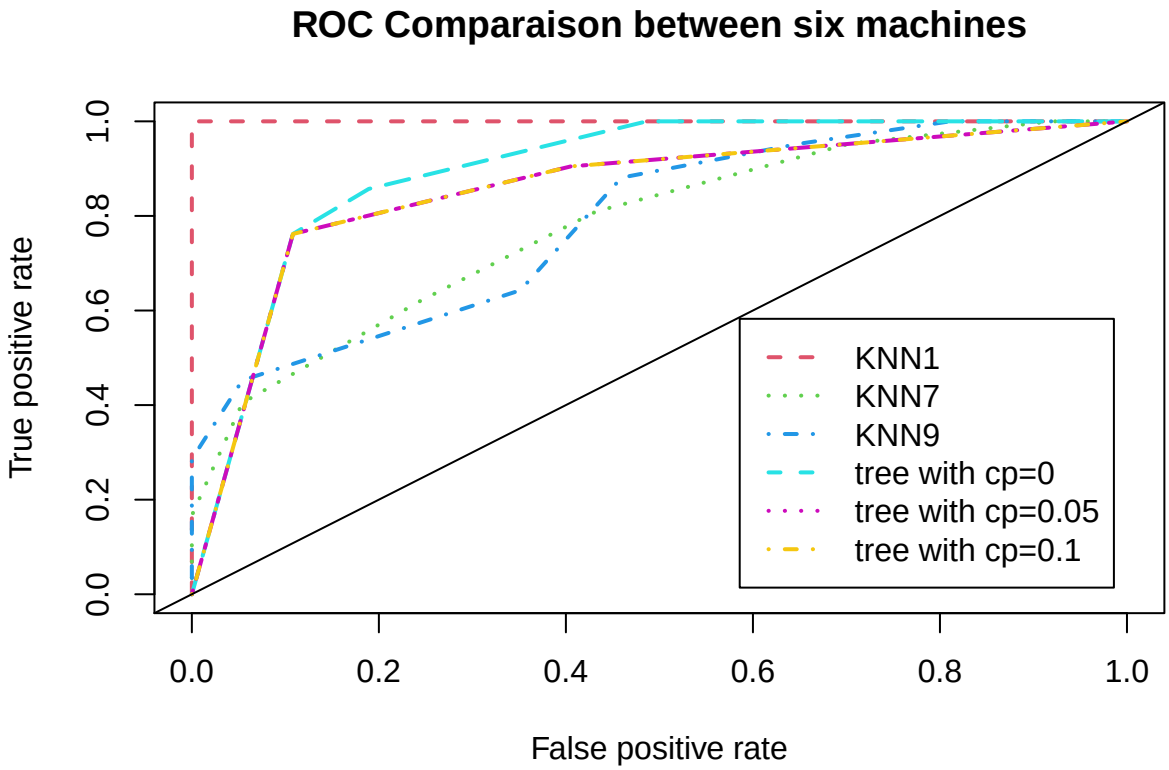

The Matrix Correlation plot above show us that must of the variable are strongly correlated.

10. Compute the eigendecomposition of the correlation matrix and comment on the ratio $\lambda_{max}/\lambda_{min}$.

```
## The Ration Lambda_max/Lambda_min is : -6.810747e+15
```

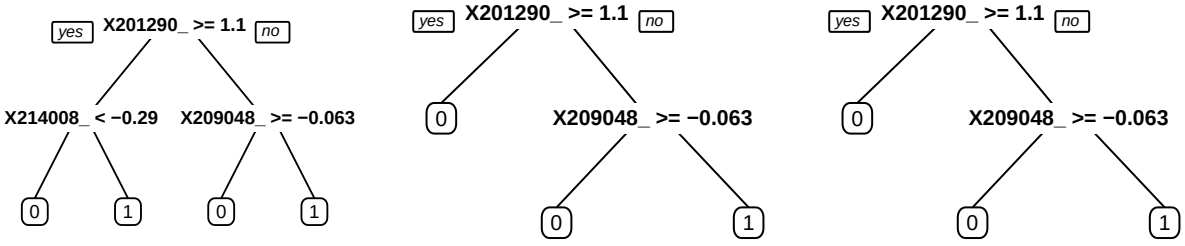
Since the ratio $\frac{\lambda_{max}}{\lambda_{min}}$ is very small, it means our correlation matrix is well-conditioned, so it is numerically stable.

11. Using the whole data for training and the whole data for test, build the above six learning machines, then plot the comparative ROC curves on the same grid




12. Plot all the three classification tree grown, using the prp function for the package rpart.plot

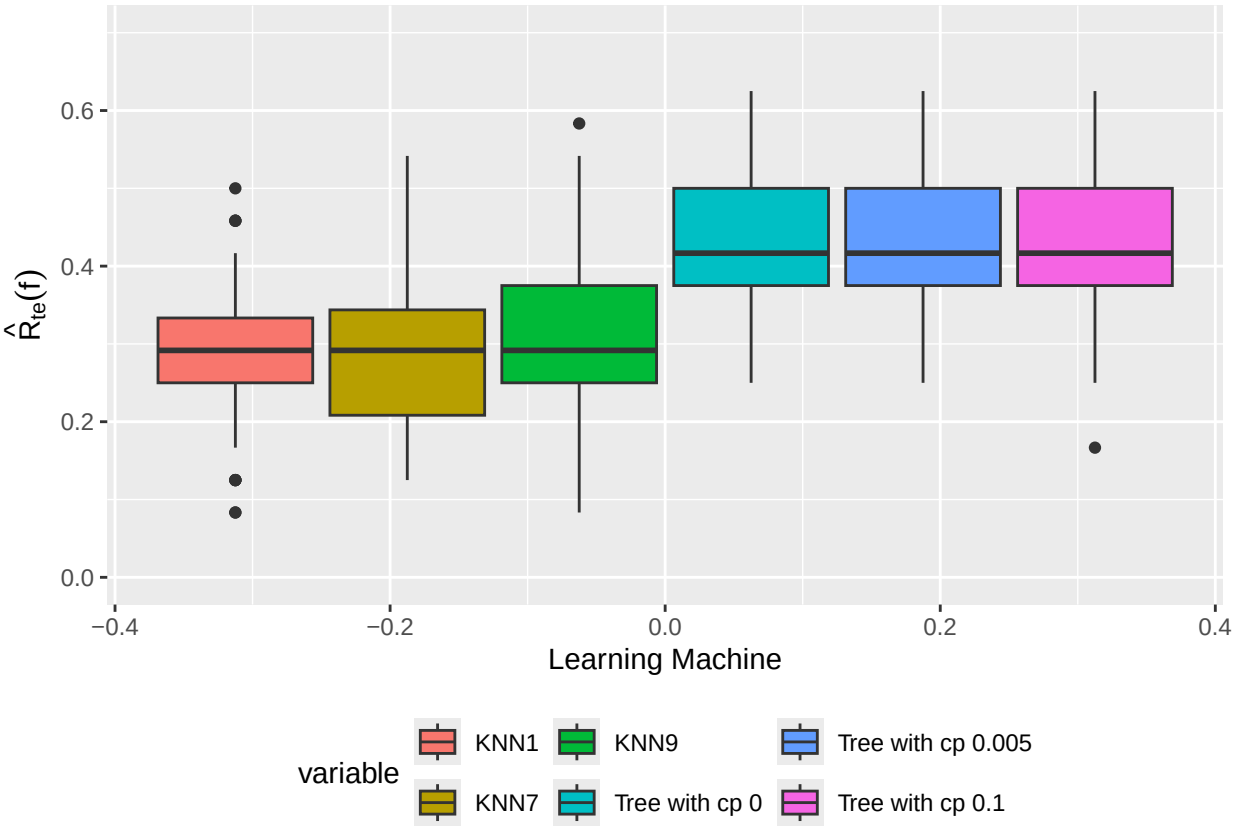
**Tree with cp =0** **Tree with cp =0.005** **Tree with cp =0.01**




13. Comment succinctly on what the ROC curves reveal for this data and argue in light of theory whether or not that was to be expected.

Based on the ROC plot, we can say that the machine KNN1 has an empirical risk equal to zero, which means it may be overfitting to the training data. Hence, this machine may not perform well on unknown data in a real-world scenario. Additionally, from the plot above, we can observe that the worst-performing machines are KNN7 and KNN9. We can also see that the best machine on the full dataset is one of the three decision tree models. However, it is not always guaranteed that the model that performs best on the full dataset will continue to be the best, because all the machines we trained above were learned from the full data. One of our recommendations is to use methods like cross-validation or stochastic holdout splits to validate our assumptions.

14. Using set.seed(19671210) along with a 7/10 training 3/10 test basic stochastic holdout split of the data, compute S = 100 replicated random splits of the test error for all the above learning machines.




- Comment on the distribution of the test error in light of (implicit) model complexity.

So, based on the plot above, we can see that when we use a method like stochastic holdout split of the data, with $S = 100$ and a 7/10 training and 3/10 testing split, we are surprised by the results. Without using this method, the universally best machine is one of the three decision trees. However, when we use this method, the best machines are KNN7 and KNN9. Hence, KNN7 and KNN9 have the capacity to better capture the unknown data.

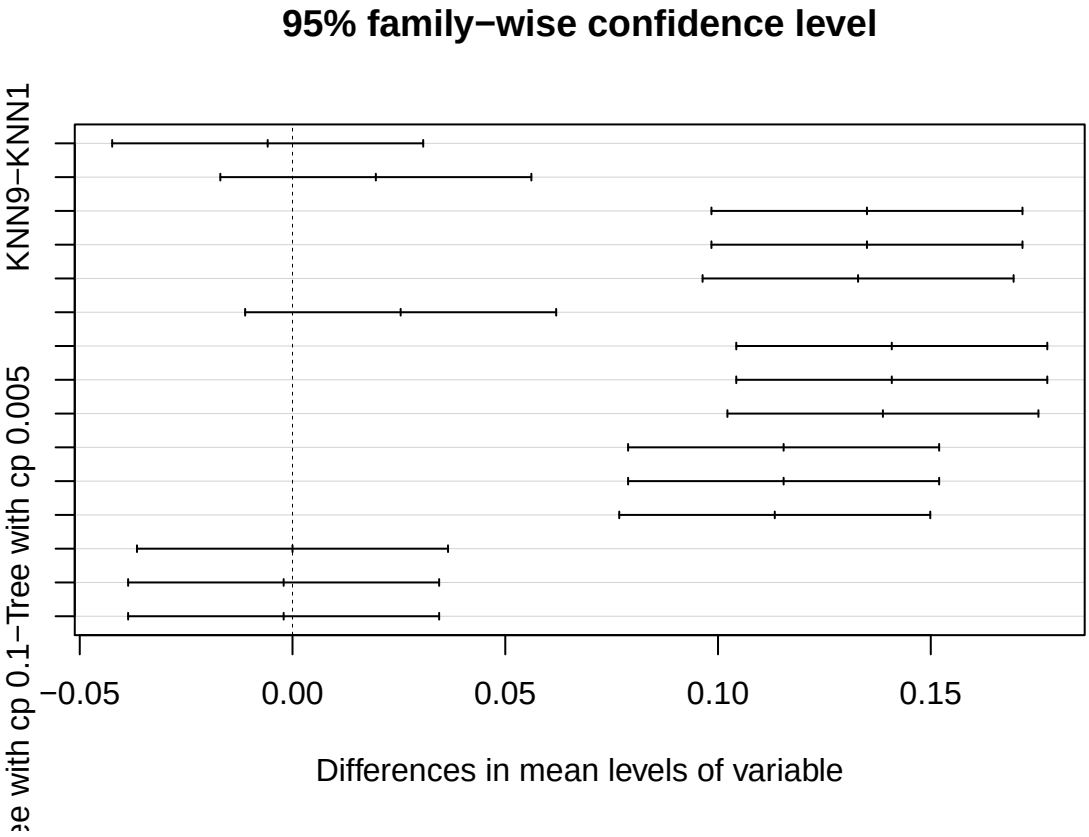
- Perform a basic analysis of variance (ANOVA) on those test errors and comment!

```
## Analysis of Variance Table
##
## Response: value
##            Df Sum Sq Mean Sq F value    Pr(>F)
## variable    5 2.5599 0.51198  62.666 < 2.2e-16 ***
## Residuals 594 4.8530 0.00817
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## variable      5  2.560  0.5120   62.67 <2e-16 ***
## Residuals   594  4.853  0.0082
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
```

```
##     factor levels have been ordered
##
## Fit: aov(formula = value ~ variable, data = melt(test_split, id.vars = NULL))
##
## $variable
##                                          diff         lwr        upr
## KNN1-KNN7                           0.005833333 -0.03071331 0.04237998
## KNN9-KNN7                           0.025416667 -0.01112998 0.06196331
## Tree with cp 0.1-KNN7               0.138750000  0.10220336 0.17529664
## Tree with cp 0-KNN7                 0.140833333  0.10428669 0.17737998
## Tree with cp 0.005-KNN7             0.140833333  0.10428669 0.17737998
## KNN9-KNN1                           0.019583333 -0.01696331 0.05612998
## Tree with cp 0.1-KNN1               0.132916667  0.09637002 0.16946331
## Tree with cp 0-KNN1                 0.135000000  0.09845336 0.17154664
## Tree with cp 0.005-KNN1             0.135000000  0.09845336 0.17154664
## Tree with cp 0.1-KNN9               0.113333333  0.07678669 0.14987998
## Tree with cp 0-KNN9                 0.115416667  0.07887002 0.15196331
## Tree with cp 0.005-KNN9             0.115416667  0.07887002 0.15196331
## Tree with cp 0-Tree with cp 0.1     0.002083333 -0.03446331 0.03862998
## Tree with cp 0.005-Tree with cp 0.1 0.002083333 -0.03446331 0.03862998
## Tree with cp 0.005-Tree with cp 0   0.000000000 -0.03654664 0.03654664
##                                         p adj
## KNN1-KNN7                           0.9975163
## KNN9-KNN7                           0.3501937
## Tree with cp 0.1-KNN7               0.0000000
## Tree with cp 0-KNN7                 0.0000000
## Tree with cp 0.005-KNN7             0.0000000
## KNN9-KNN1                           0.6436726
## Tree with cp 0.1-KNN1               0.0000000
## Tree with cp 0-KNN1                 0.0000000
## Tree with cp 0.005-KNN1             0.0000000
## Tree with cp 0.1-KNN9               0.0000000
## Tree with cp 0-KNN9                 0.0000000
## Tree with cp 0.005-KNN9             0.0000000
## Tree with cp 0-Tree with cp 0.1     0.9999840
## Tree with cp 0.005-Tree with cp 0.1 0.9999840
## Tree with cp 0.005-Tree with cp 0   1.0000000
```




15. Comment extensively on the most general observation and lesson you gleaned from this exploration.

So, the general observation that comes to our attention is that, in order to find the best model for our data, it is very important to use methods like cross-validation or stochastic holdout split to better capture realism. This is because learning from all the data increases the risk of overfitting our model.

## Exercise 2: Nearest Neighbors Method for Digit Recognition (30points)

### Part 1: Multi-class classification on MNIST

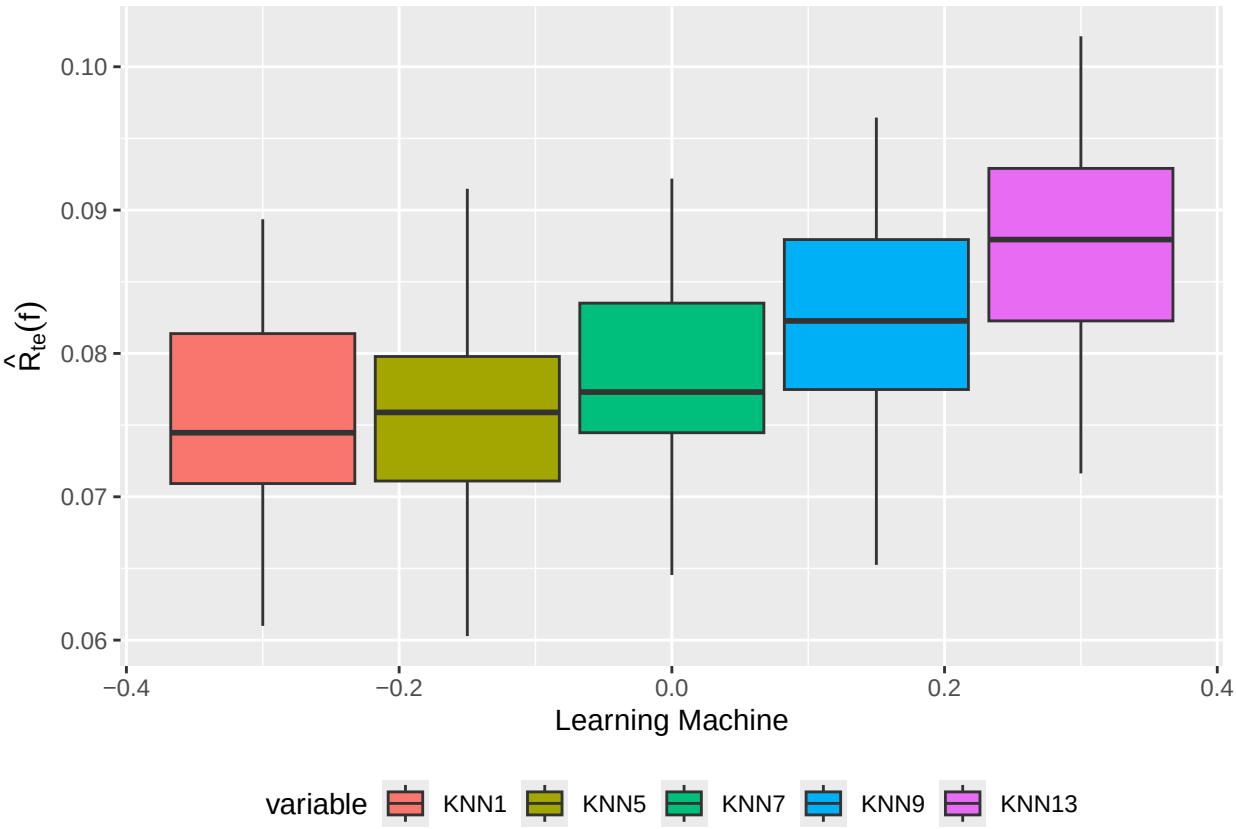
1. Write down in mathematical form the expression of bfkNN(x), the prediction function of the kNN learning machine.

the predicted class $\widehat{f}_{\texttt{kNN}}(x)$ of $x \in \mathcal{X}$ is given by

$$\widehat{f}_{\texttt{kNN}}(x) = \underset{c \in \mathcal{Y}}{\texttt{argmax}} \left\{ \frac{1}{k} \sum_{i=1}^{n} \mathbf{1}(x_i \in \mathcal{V}_k(x)) \mathbf{1}(y_i = c) \right\}$$

where $\mathcal{Y}$ represents our output space and $\mathcal{X}$ represent our input space, and $\mathcal{V}_k(x)$ represents the set of the $k$ nearest neighbors of $x \in \mathcal{X}$ in the provided dataset $\mathcal{D}_n$.

2. Choose n a training set size and m a test set size, and write a piece of code for sampling a fragment from the large dataset. Explain why you choose the numbers you chose.
3. Let S = 50 be the number of random splits of the data into 70% training and 30% Test.
4. Build over all the 5 models and compute the test errors for each split, storing the results into a matrix of test errors




2. Identify the machine with the smallest median test error and generate the test confusion matrix from the last split

based on the boxplot above we can see that the machine with small median test error is the KNN with k = 5.

```
##    y.te.hat_1
## yte   0   1   2   3   4   5   6   7   8   9
##   0 137   0   0   0   0   3   0   1   0   0
##   1   0 149   0   0   0   0   0   0   0   0
##   2   2   4 116   0   1   0   0   2   1   0
##   3   0   0   0 120   0   5   0   0   4   0
##   4   0   0   0   0 123   0   3   1   0  10
##   5   0   1   0   8   0 131   6   0   2   5
##   6   3   0   0   0   0   0 137   0   0   0
##   7   0   2   2   1   0   1   0 162   0   1
##   8   1   1   3   5   1   6   0   0 114   4
##   9   1   1   0   2   2   2   0   4   1 118
```

```
##    y.te.hat_5
## yte   0   1   2   3   4   5   6   7   8   9
##   0 135   0   0   0   1   2   3   0   0   0
##   1   0 149   0   0   0   0   0   0   0   0
##   2   2   5 117   0   1   0   0   1   0   0
##   3   0   1   1 117   0   4   0   2   4   0
##   4   0   3   0   0 120   0   3   0   0  11
##   5   0   4   0   8   0 127   6   2   1   5
##   6   2   1   0   0   1   0 136   0   0   0
##   7   0   6   0   0   0   1   0 160   0   2
##   8   2   1   1   3   2   3   0   1 119   3
```

```
##   9   2   2   0   2   1   1   0   3   0 120
```

3. Comment on the digits for which there is a lot more confusion. Does that agree with your own prior intuition about digits?

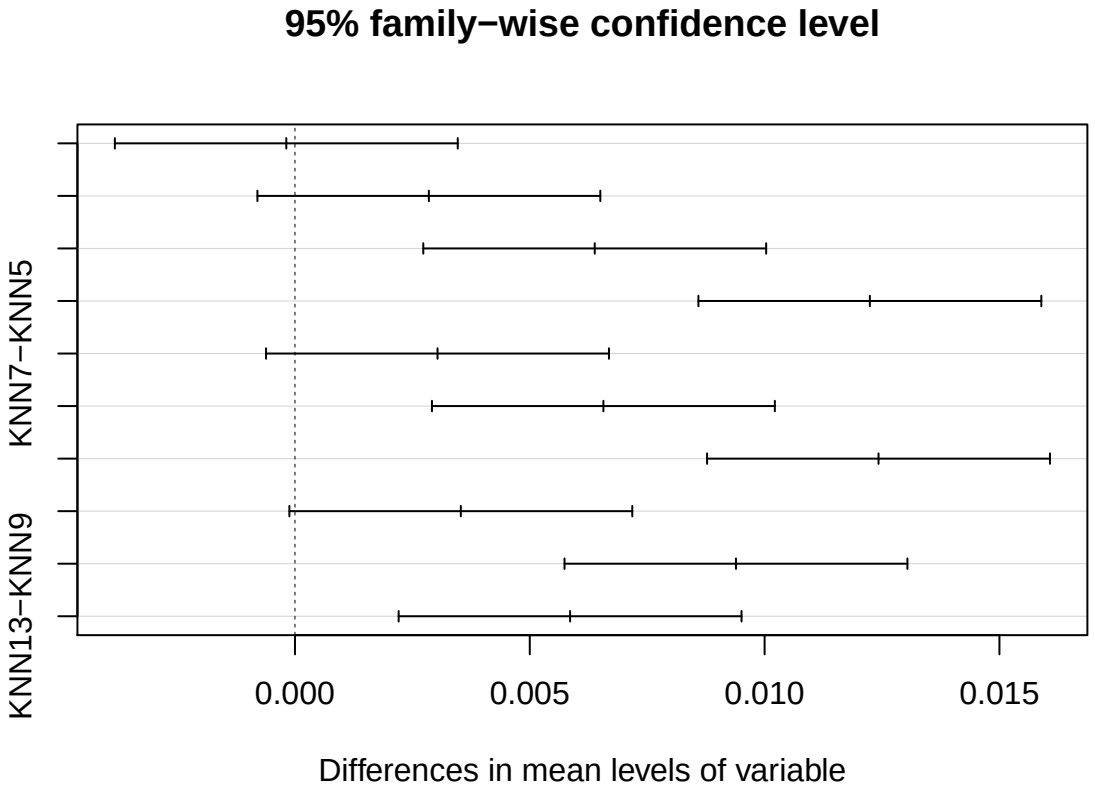
Based on the table of confusion matrix we can see that the digit with the lot more confusion the the digit 1 and 7, it meet our intuition because sometimes people write the digit 1 and 7 in different way, and Machine can make a confusion with other digit.

4. Perform an ANOVA of the test errors and comment on the patterns that emerge.

```
## Analysis of Variance Table
##
## Response: value
##            Df    Sum Sq    Mean Sq F value    Pr(>F)
## variable    4 0.0054045 0.00135114  30.631 < 2.2e-16 ***
## Residuals 245 0.0108069 0.00004411
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##              Df   Sum Sq   Mean Sq F value Pr(>F)
## variable      4 0.005405 0.0013511   30.63 <2e-16 ***
## Residuals   245 0.010807 0.0000441
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##     factor levels have been ordered
##
## Fit: aov(formula = value ~ variable, data = melt(test, id.vars = NULL))
##
## $variable
##                     diff           lwr         upr     p adj
## KNN1-KNN5  0.0001843972 -0.0034660467 0.003834841 0.9999159
## KNN7-KNN5  0.0030354610 -0.0006149829 0.006685905 0.1531731
## KNN9-KNN5  0.0065673759  0.0029169320 0.010217820 0.0000140
## KNN13-KNN5 0.0124255319  0.0087750880 0.016075976 0.0000000
## KNN7-KNN1  0.0028510638 -0.0007993801 0.006501508 0.2040291
## KNN9-KNN1  0.0063829787  0.0027325348 0.010033423 0.0000265
## KNN13-KNN1 0.0122411348  0.0085906909 0.015891579 0.0000000
## KNN9-KNN7  0.0035319149 -0.0001185290 0.007182359 0.0632259
## KNN13-KNN7 0.0093900709  0.0057396270 0.013040515 0.0000000
## KNN13-KNN9 0.0058581560  0.0022077121 0.009508600 0.0001498
```




### Part 2: Binary classification on MNIST

1. Store in memory your training set and your test set. Of course you must show the command that extracts only '1' and '7' from both the training and the test sets.
2. Display both your training confusion matrix and your test confusion matrix

```
##              y.te_7.hat_1
## y_test_custom  1  7
##             1 86  2
##             7  2 75
```

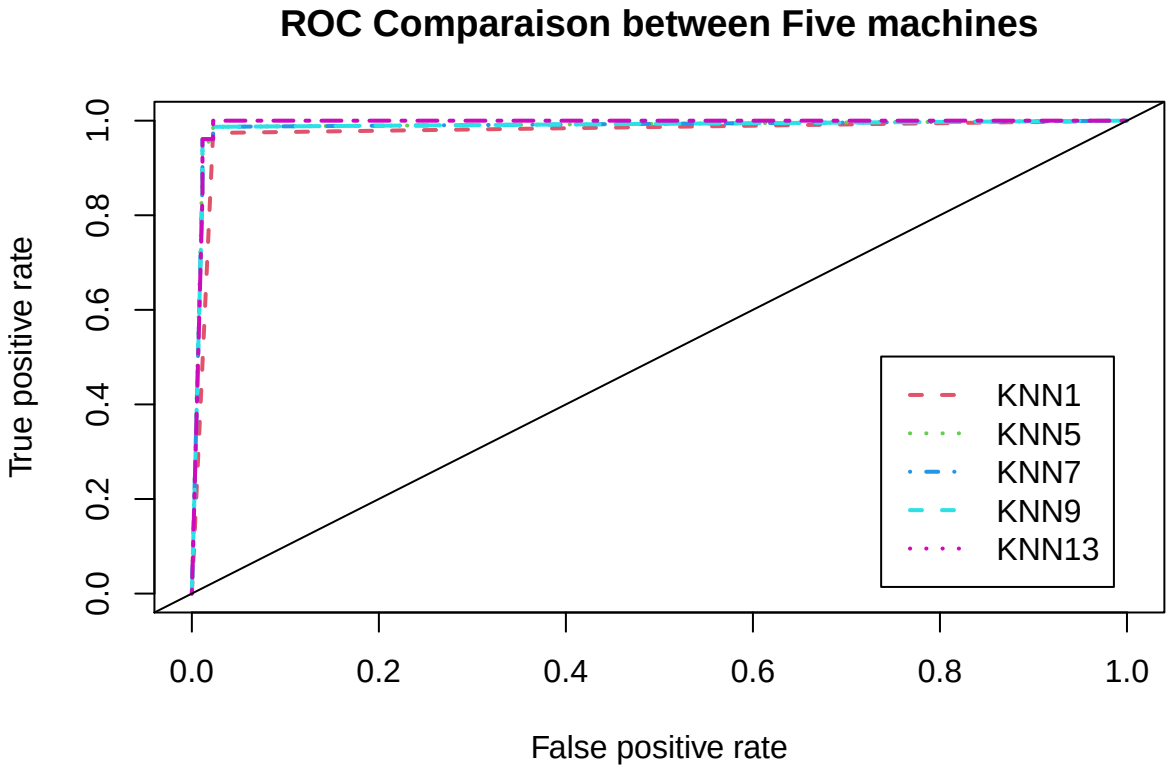
```
##              y.te_7.hat_5
## y_test_custom  1  7
##             1 87  1
##             7  4 73
```

```
##              y.te_7.hat_7
## y_test_custom  1  7
##             1 87  1
##             7  3 74
```

```
##              y.te_7.hat_9
## y_test_custom  1  7
##             1 87  1
##             7  3 74
```

```
##              y.te_7.hat_13
## y_test_custom  1  7
##             1 87  1
##             7  5 72
```

3. Display the comparative ROC curves of the five learning machines




4. Identify two false positives and two false negatives at the test phase, and in each case, plot the true image against its falsely predicted counterpart.

```
## [1] "Index postion of false positives and two false negatives for KNN1"

## $false_positives
## [1] 34 55
##
## $false_negatives
## [1]  8 52

## [1] "Index postion of false positives and two false negatives for KNN5"

## $false_positives
## [1] 55 NA
##
## $false_negatives
## [1]  8 52

## [1] "Index postion of false positives and two false negatives for KNN7"

## $false_positives
## [1] 55 NA
##
## $false_negatives
## [1]  8 52

## [1] "Index postion of false positives and two false negatives for KNN9"

## $false_positives
## [1] 55 NA
##
```

```
## $false_negatives
## [1] 52 85
```

```
## [1] "Index postion of false positives and two false negatives for KNN13"
```

```
## $false_positives
## [1] 55 NA
##
## $false_negatives
## [1]  8 52
```

**KNN1 − FP 1**

**KNN1 − FP 2**

**KNN5 − FP 1**

**KNN1 − FN 1**

**KNN1 − FN 2**

**KNN5 − FN 1**

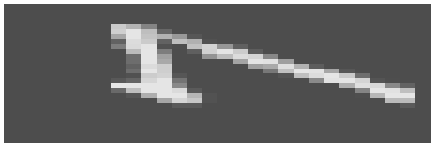
**KNN7 − FP 1**

**KNN7 − FP 2**

**KNN9 − FP 1**

**KNN7 − FN 1**

**KNN7 − FN 2**



**KNN9 − FN 1**

**KNN13 − FP 1**

**KNN13 − FP 2**

**KNN13 − FN 1**

**KNN13 − FN 2**

5. Comment on any pattern that might have emerged.