Assignment 2 Machine learning

jean aime Iraguha

2024-12-14

Introduction

This assignment focuses on the application of Statistical Machine Learning (SML) techniques using two distinct datasets: the prostate cancer dataset and the MNIST dataset. The primary goal is to apply various machine learning models, including k-Nearest Neighbors (1NN, 7NN, 9NN) and decision trees with varying complexity parameters, to classify and analyze the datasets.

In **Exercise 1**, the prostate cancer dataset, which contains DNA MicroArray gene expression data from cancer and non-cancer subjects, is explored. The task involves comparing the performance of different learning algorithms based on test error rates. Additionally, statistical techniques such as the Kruskal-Wallis test are applied to identify the most significant predictor variables, and decision trees are built to evaluate the most influential features.

Exercise 2 involves the use of the MNIST dataset for digit recognition, specifically focusing on the binary classification of digits '1' and '7'. Various kNN models (1NN, 5NN, 7NN, 9NN, and 13NN) will be applied, and model performance will be assessed using confusion matrices, ROC curves, and error analysis.

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

loading the necessary packages.

Loading the data

```
prostate <- read.csv("prostate-cancer-1.csv") # DNA MicroArray Gene Expression
```

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

```
## [1]  79 501
```

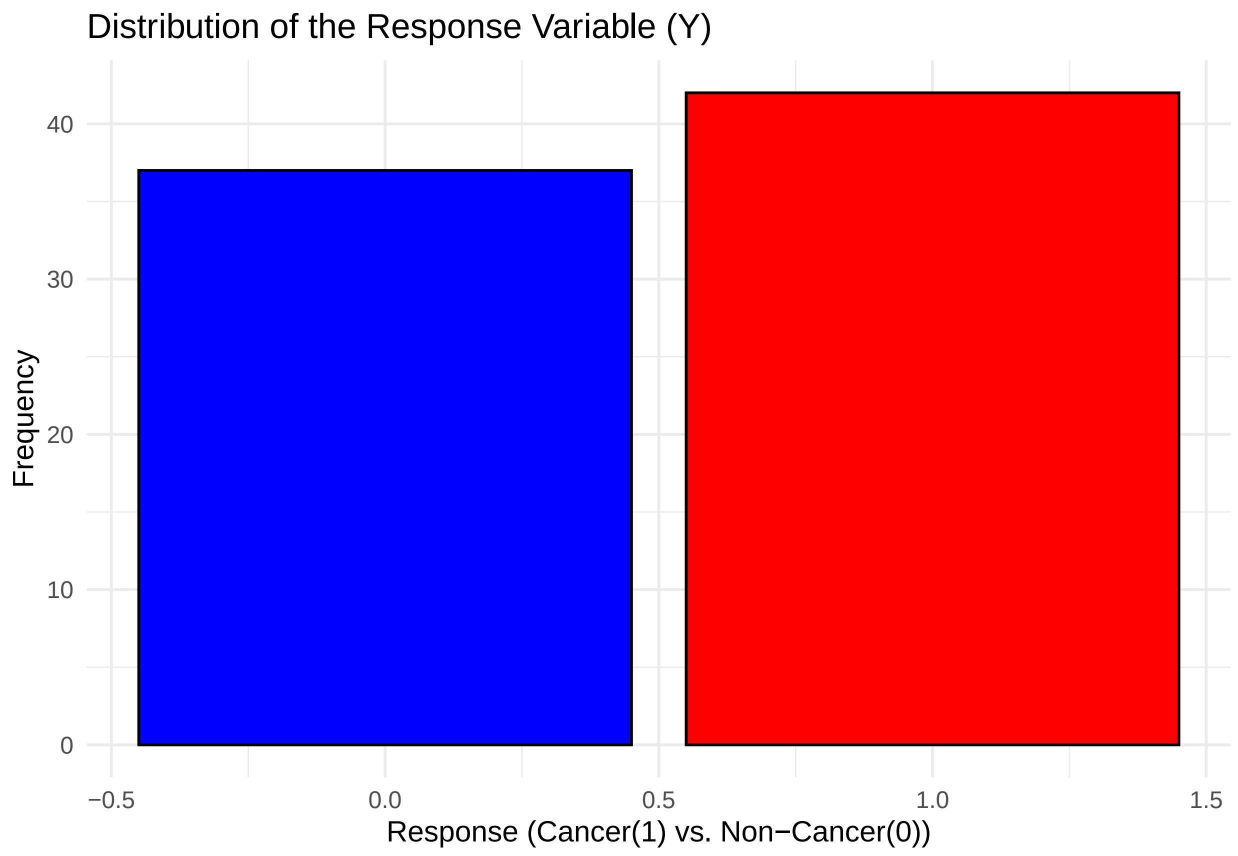
The dataset has 79 instances (rows), each representing a sample (e.g., a patient), and 500 features (columns), which correspond to gene expression levels. This means the dataset has high dimensionality, with each instance being described by 500 measurements. While the large number of features may provide rich information for classification, the small sample size (79 instances) can lead to challenges such as overfitting and poor generalization, as there may not be enough data to effectively capture the relationships between features and the target variable. This combination of high dimensionality and small sample size requires careful model selection and regularization to avoid overfitting.

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

From a **statistical perspective**, the input space in this dataset has the following characteristics:

1. **Continuous Data**:
   - The predictors (e.g., `X206212_at`, `X207075_at`, etc.) are **continuous numerical values**, representing gene expression levels.
   - These values are derived from DNA microarray experiments, typically measured as fluorescence intensities.
2. **High Dimensionality**:
   - The dataset consists of a large number of predictors (features) relative to the sample size.
   - This scenario, known as **high-dimensional data**, poses challenges such as overfitting and the "curse of dimensionality."
3. **Potential Correlations**:
   - Gene expressions are likely to exhibit **strong correlations** because genes often operate in pathways or networks.
4. **Variation Across Classes**:
   - The response variable `Y` is **categorical** (cancer vs. non-cancer), so the predictors are expected to show distinct patterns across these classes.
5. **Statistical Nature**:
   - Gene expression data may not follow a Gaussian distribution due to biological variability and noise in measurements.
   - Using non-parametric tests like the **Kruskal-Wallis test** is appropriate to assess relationships between predictors and the response variable.

Plot the distribution of the response



The response variable y is categorical with two categories which are cancer (1) and no cancer (0). From barplot it is clear that we have high proportion of no cancer compared to that of cancer in dataset.

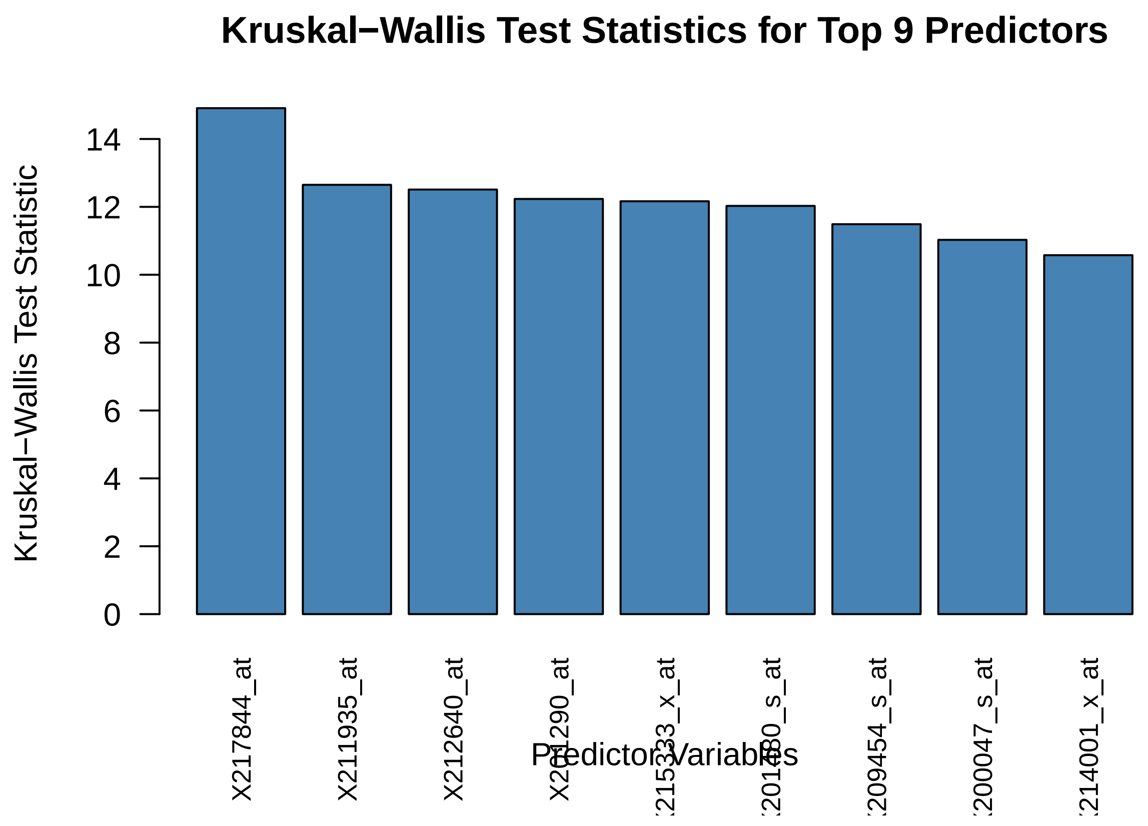
4. Identify the 9 individually most powerful predictor variables with respect to the response according the Kruskal-Wallis test statistic

```
## [1] "The top 9 variable are "
```

```
## X217844_at
## X211935_at
## X212640_at
## X201290_at
## X215333_x_at
## X201480_s_at
## X209454_s_at
## X200047_s_at
## X214001_x_at
```

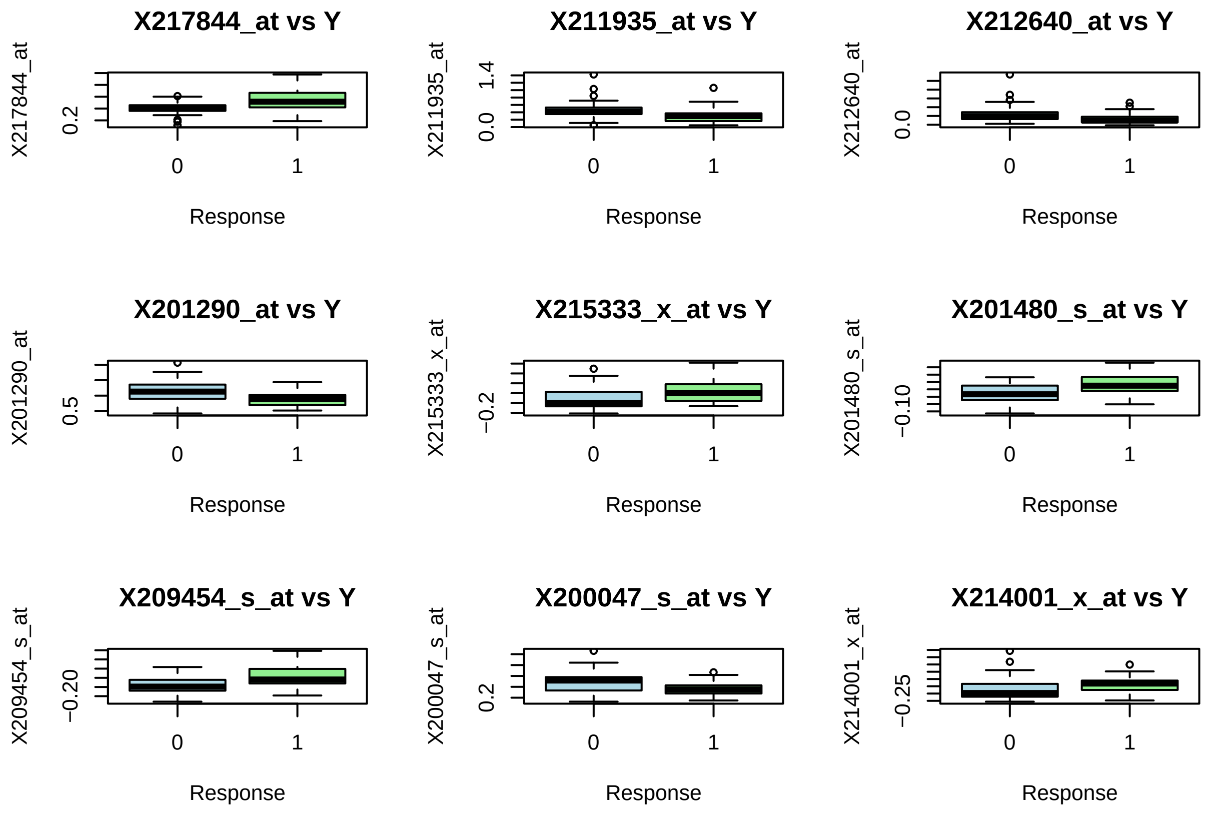
5.Generate a type='h' plot with the Kruskal-Wallis test statistic as the y-axis and the variable name as the x-axis

This is plot with the Kruskal-Wallis test statistic as the y-axis and the variable name as the x-axis for top 9 predictors.



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

In this part we aimed to plot most powerful variable with respect to response variable to gain an insight about the root of tree.

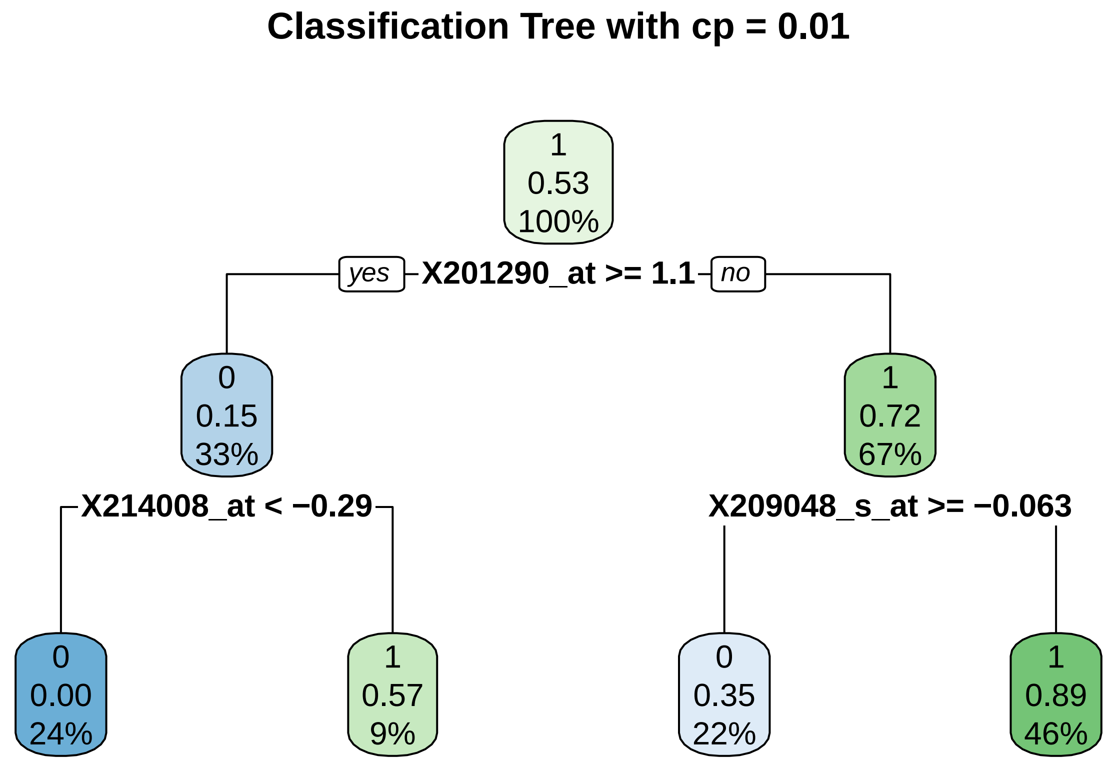


### *Comment*

This plot Doesn't not show well the variable that allocate well the data into categories of cancer but it shows that the variable \$ X201290_{at}\$, $X217844_{at}$ and $X211935_{at}$ is some how exceeding other is separating the group.

Building the classification tree with cp=0.01

Plot the tree you just built



### Determine the number of terminal nodes

```
## number of terminal nodes is:  4
```

Write down in mathematical form region 2 and Region 4.

Region 2: This is the formart of region 2

$$\text{Region 2} : \{\mathbf{x} \in \mathbb{R}^p : X_{201290\_at} \geq 1.1 \text{ and } X214008\_at \geq -0.29\}$$

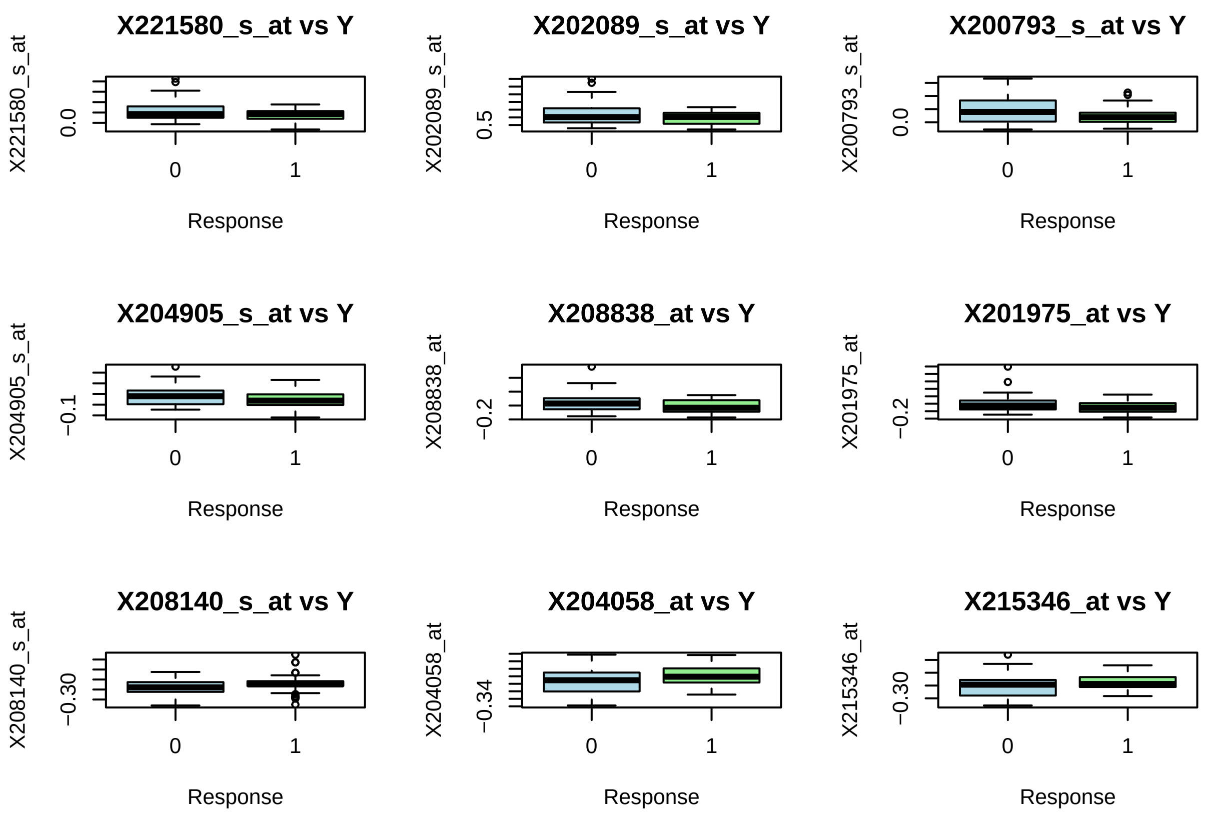
Region 4: This is the formart of region 4

$$\text{Region 4} : \{\mathbf{x} \in \mathbb{R}^p : X_{201290\_at} < 1.1 \text{ and } X209048\_s\_at < -0.063\}$$

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

The variable at the root is what we were expecting to be there, as it is among the best predictors for separating the classes, based on the Kruskal-Wallis statistic. The Kruskal-Wallis test measures the difference in distributions between classes, and the variable at the root likely has the highest test statistic, indicating a strong ability to discriminate between the groups.

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

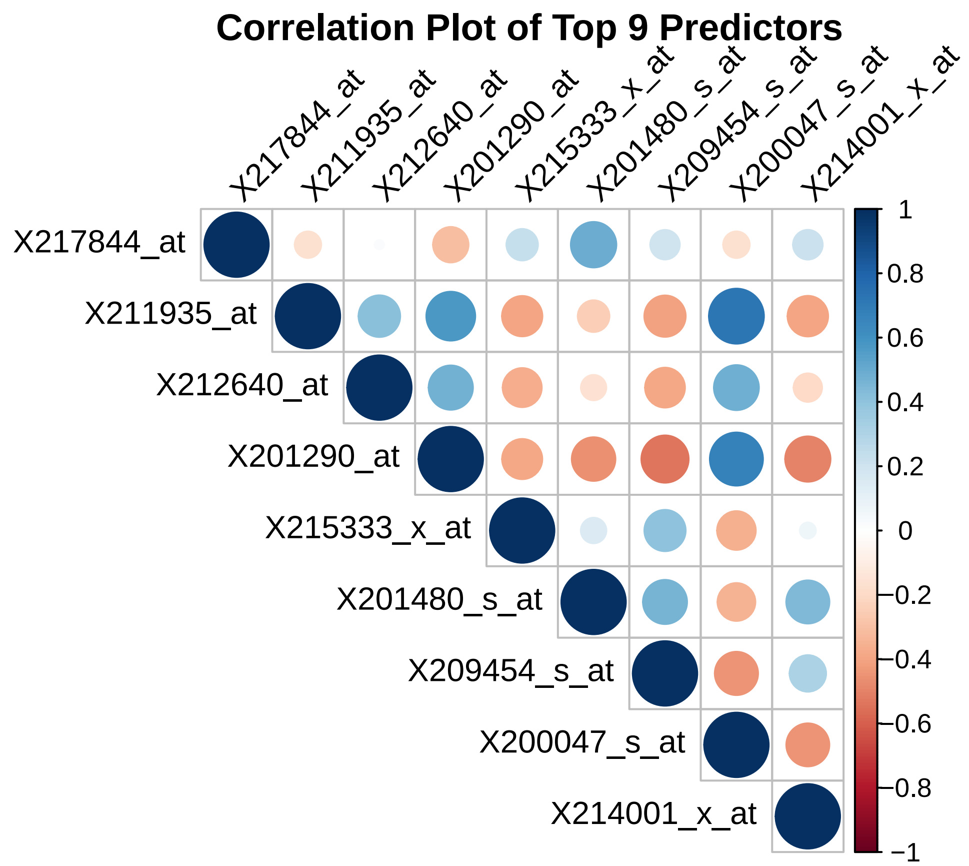


9. Generate the correlation plot of the predictor variables and comment extensively on what they reveal, if anything.

In this section we Generate the correlation plot of the predictor variables and comment extensively on what they reveal, if anything.

```
## Warning: package 'corrplot' was built under R version 4.4.2
```

```
## corrplot 0.95 loaded
```



The correlation plot presented above shows the relationships between the top 9 predictors in the dataset. The size of the circles represents the strength of the correlation, with larger circles indicating stronger correlations. Additionally, the color of the circles reflects the direction of the correlation: **blue** for positive correlations and **red** for negative correlations.

Specific Observations

- **Strong Positive Correlations:** `X217844_at` and `X211935_at` exhibit a **strong positive correlation**, indicated by a large **blue circle**. This suggests that as one predictor increases, the other tends to increase as well. Similarly, `X217844_at` shows a **strong positive correlation** with `X212640_at` and `X201290_at`, also represented by large blue circles.
- **Moderate Positive Correlations:** Several predictors show moderate positive correlations with `X217844_at`, such as `X215333_x_at`, `X201480_s_at`, and `X209454_s_at`, represented by medium-sized blue circles. These relationships suggest that while the variables are related, the strength of the association is not as high as the strong correlations mentioned earlier.
- **Weak or No Correlations:** Most other correlations in the plot are either weak or negligible, indicated by small circle sizes and light colors. This suggests that these predictors have little to no linear relationship with each other.

10. Compute the eigendecomposition of the correlation matrix and comment on the ratio.

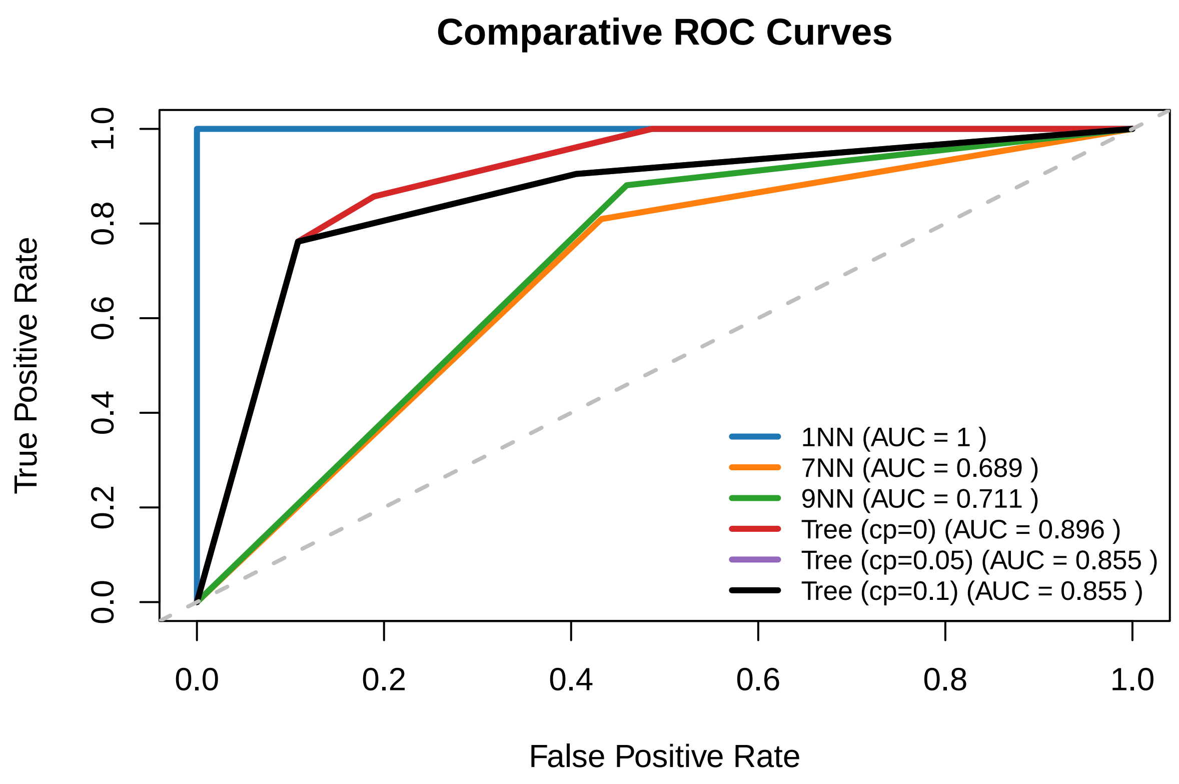
```
## The ratio = 17.02747
```

Interpretation of The ratio of Lambda

The ratio $\lambda_{\max}/\lambda_{\min} = 17.03$ indicates that the top 9 predictors capture most of the variance in a few principal components. This is advantageous for classification tasks, as it suggests that the top predictors are highly informative and collectively dominate in distinguishing between classes. The concentration of variance in fewer dimensions can simplify the model's learning process, potentially leading to more accurate and efficient classification. Since classification models like decision trees, kNN, or random forests are robust to multicollinearity, this ratio highlights the strength of these predictors in capturing essential patterns without compromising the model's performance.

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

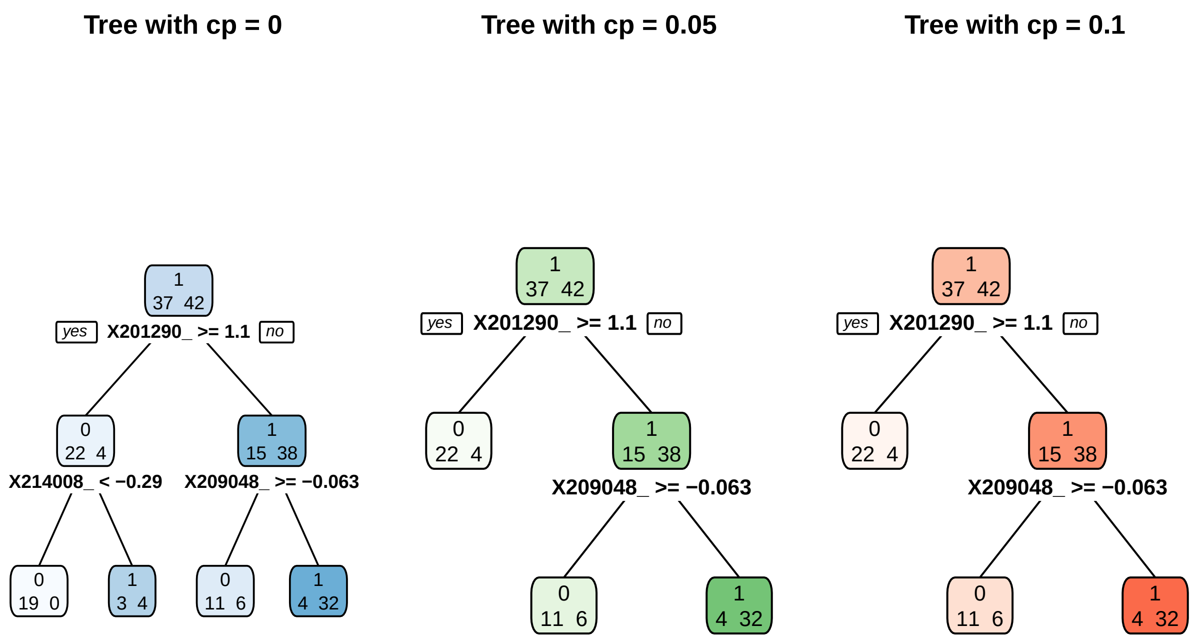
```
## Warning: package 'ROCR' was built under R version 4.4.2
```



12. 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.

In this classification task, decision trees are generally preferred over kNN due to their superior performance and interpretability. The decision trees, especially with a complexity parameter (cp = 0.05), consistently show higher AUC values (0.855 to 0.896), indicating effective class separation and better generalization. In contrast, the kNN models, particularly the 7NN and 9NN, exhibit lower AUC values (around 0.689 and 0.711), likely due to the curse of dimensionality, which degrades performance in high-dimensional spaces. Although the 1NN model shows perfect performance with an AUC of 1, this result is likely due to overfitting, as 1NN is highly sensitive to noise and small variations in the data. It also lacks generalization capability, making it unsuitable for real-world applications where the model needs to perform well on unseen data. Additionally, decision trees provide clear, interpretable classification rules, while kNN lacks transparency. Overall, decision trees offer a better balance of performance, generalization, and interpretability, making them the preferred model for this classification problem.

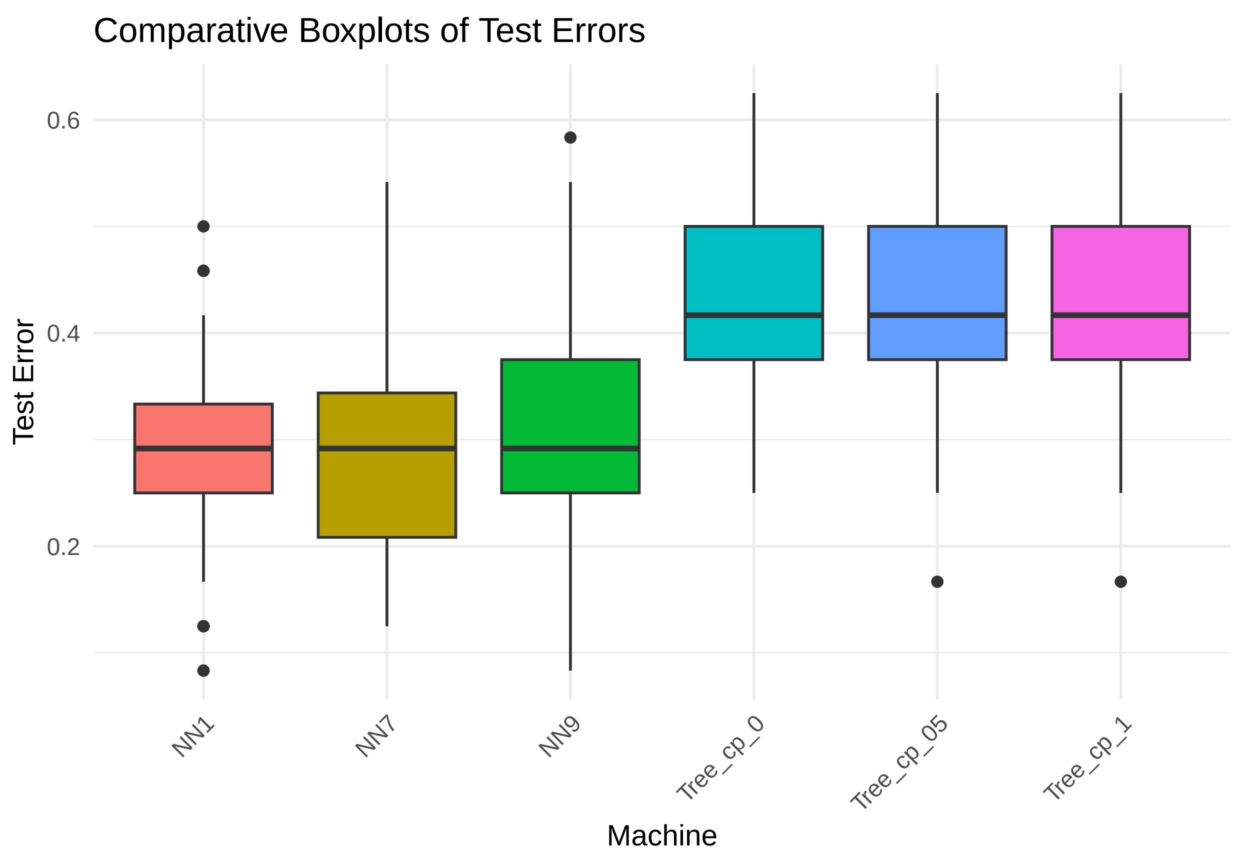
13. Plot all the three classification tree grown, using the prp function for the package rpart.plot.



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.

a) Plot the comparative boxplots (be sure to properly label the plots)

```
## Warning: package 'tidyr' was built under R version 4.4.2
```



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

According to this box plot it is absoltly clear that KNN model perform better than Trees. This negate information we was having on ROC curve . We have to relie on the information of the box plot because we use the test set to test our model while we use train set for testing data.

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

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## Model         5  2.560  0.5120   63.39 <2e-16 ***
## Residuals   594  4.797  0.0081
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Comment

The ANOVA results show that the five models (1NN, 7NN, 9NN, and decision trees with varying complexities) are significantly different in their performance, with a high F value of 63.39 and a p-value of <2e-16. This indicates that at least one model is performing better than the others. The low mean square residuals (0.0081) suggest that the models explain most of the variance in the data effectively. Overall, the models show strong explanatory power, and further analysis is needed to identify which models perform the best.

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

The general observation from thism exploatation is divided into this part:

- It is not good to test the model on the train set because as we haveseen it give wrong information on ROC curve . it is better to test your model ion the dataset which is not train set.
-

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

This exercise features the analysis of the USPS digit recognition dataset using kNN with various neighborhood sizes.

```
## Warning: package 'dslabs' was built under R version 4.4.2
```

Part 1: Multi-class classification on MNIST

Throughout this part of the exercise, you will perform multiclass classification in the MNIST data using the learning machines 1NN, 5NN, 7NN, 9NN, and 13NN.

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

1.Mathematical Expression

The prediction function for the k-Nearest Neighbors (kNN) classifier is of the form :

$$\hat{f}_{kNN}(x) = \arg \max_{g=1,2,\ldots,G} \left(\frac{1}{k} \sum_{i=1}^{n} \mathbb{I}(y_i = g)\mathbb{I}(x_i \in V_k(x)) \right)$$

Where:

- x is the test point for which we want to predict the class label.
- $V_k(x)$ is the set of the k nearest neighbors of x in the training data.
- $\mathbb{I}$ is the indicator function, which equals 1 if the condition inside is true, and 0 otherwise.
- G is the number of classes in the whole dataset.
- y_i is the label of the i-th training point.
- x_i is the feature vector of the i-th training point.

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.

Here's how you can choose a smaller subset of the MNIST dataset and sample it in a way that maintains its integrity for model training:

Sampling Strategy for MNIST Dataset

The MNIST dataset contains 60,000 training samples and 10,000 test samples. In this experiment, we use a sampling strategy to create a smaller training set of 200 samples and a test set of 1000 samples. According to **sampling theory**, random sampling allows us to select representative subsets from the large dataset while maintaining the underlying distribution of the full data. By sampling 200 training samples, we can quickly experiment with model configurations, although smaller training sets may lead to overfitting. The 1000 test samples provide a sufficiently large and reliable sample to evaluate model performance while ensuring the experiment runs efficiently. This approach strikes a balance between computational efficiency and reliable performance assessment, which is crucial for quick iterations in model development.

3. Let S = 50 be the number of random splits of the data into 70% training and 30% Test.

a) Random Splitting of the Data

We divide the dataset into **50 random splits**, with **70% of the data for training** and **30% for testing** in each split. This approach ensures that the model's performance is evaluated across multiple data partitions, reducing the impact of variability in a single split.

b) Code Implementation

The following R code demonstrates how to achieve this:

```
## Loading required package: lattice
```

```
## Warning: package 'lattice' was built under R version 4.4.2
```

1. Build overall the 5 models and compute the test errors for each split, storing the results into a matrix of test errors

Test Error Computation Across 50 Random Splits

We evaluate 5 kNN models ($k = 1, 5, 7, 9, 13$) across **50 random splits** of the dataset. For each split, we compute the test errors and store them in a matrix for further analysis.

Code Implementation

```
##        kNN_k=1   kNN_k=5   kNN_k=7   kNN_k=9  kNN_k=13
## [1,] 0.1461794 0.1495017 0.1694352 0.1926910 0.2325581
## [2,] 0.1262458 0.1594684 0.1528239 0.1893688 0.2093023
## [3,] 0.1162791 0.1561462 0.1627907 0.1694352 0.1860465
## [4,] 0.1428571 0.1262458 0.1328904 0.1561462 0.1495017
## [5,] 0.0897010 0.1328904 0.1229236 0.1395349 0.1528239
## [6,] 0.1129568 0.1129568 0.1262458 0.1461794 0.1694352
```

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

In this section We identify the kNN model with the **smallest median test error** across 50 splits. Then, we generate the **confusion matrix** for the last split to evaluate the classification performance in detail.

Code Implementation

```
## The best kNN model is with k = 1 having the smallest median test error.
```

```
##           Reference
## Prediction  0  1  2  3  4  5  6  7  8  9
##          0 28  0  0  0  0  0  0  0  0  0
##          1  0 33  1  2  1  0  0  1  0  0
##            2  0  0 24  1  0  0  0  0  0  0
##            3  0  0  2 24  0  1  0  0  1  0
##            4  0  0  0  0 24  0  1  0  0  1
##            5  1  0  0  2  0 25  1  0  2  0
##            6  0  0  0  0  0  1 28  0  3  0
##            7  0  0  2  0  0  0  0 29  0  2
##            8  1  0  1  0  0  0  0  0 19  0
##            9  0  1  0  2  4  0  0  1  4 27
```

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

The confusion observed between specific pairs of digits aligns well with prior expectations and highlights the challenges posed by handwritten variability and structural similarities. it meet the expecteed since digit like 4 and 9 look to be the same and was confused 5 times.

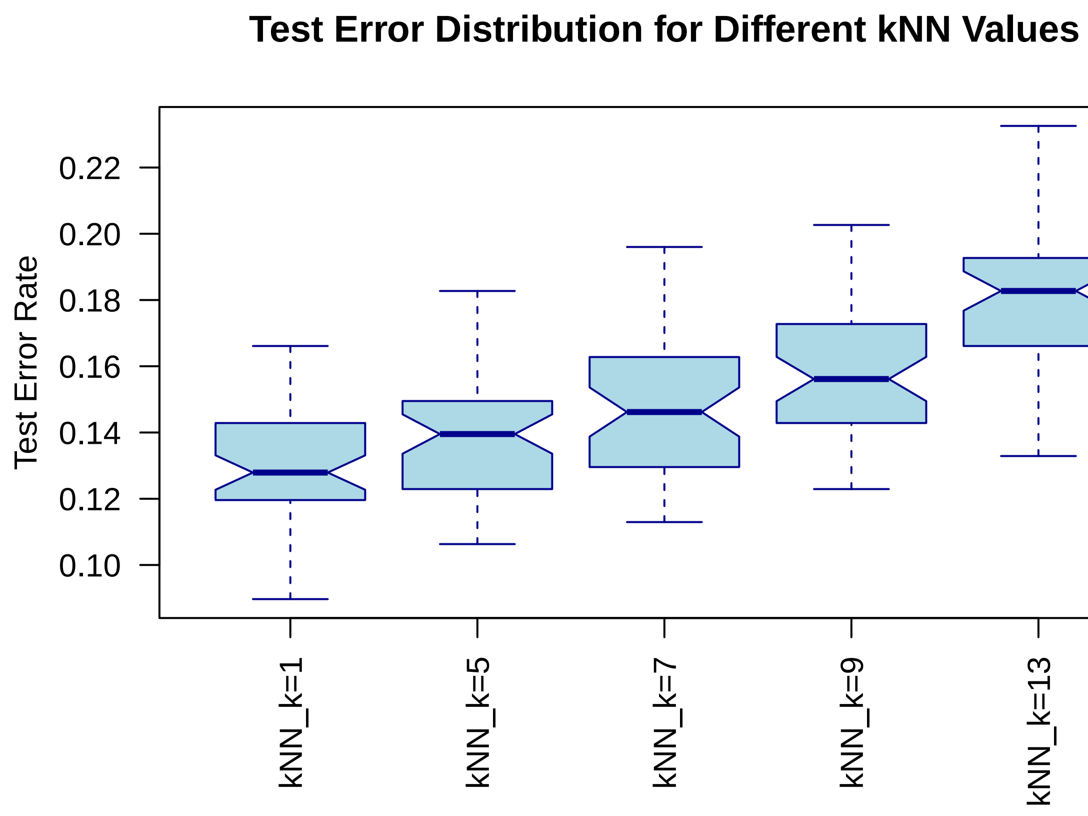
4. Perform an ANOVA of the test errors and comment on the patterns that emerge.

```
##              Df  Sum Sq  Mean Sq F value Pr(>F)
## kNN           4 0.07471 0.018678   49.44 <2e-16 ***
## Residuals   245 0.09255 0.000378
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From this Anova results we can obtain that p value is very small that why we have enough statisticall evidence that the error are not the same. so the model has differrent test error.

boxplot for checking the model error.

Here is the box plot to over see the performance of the models.



The box plot shows that the 1NN is performing well than other modeleven if it would be over fitting . 7NN is in the middle so it have median error.

Part 2: Binary classification on MNIST

Consider classifying digit '1' against digit '7', with '1' representing positive and '7' representing negative. You will be using just 1NN, 5NN, 7NN, 9NN, and 13NN.

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.

```
## Filtered Training Set Size: 2168 Filtered Test Set Size: 433
```

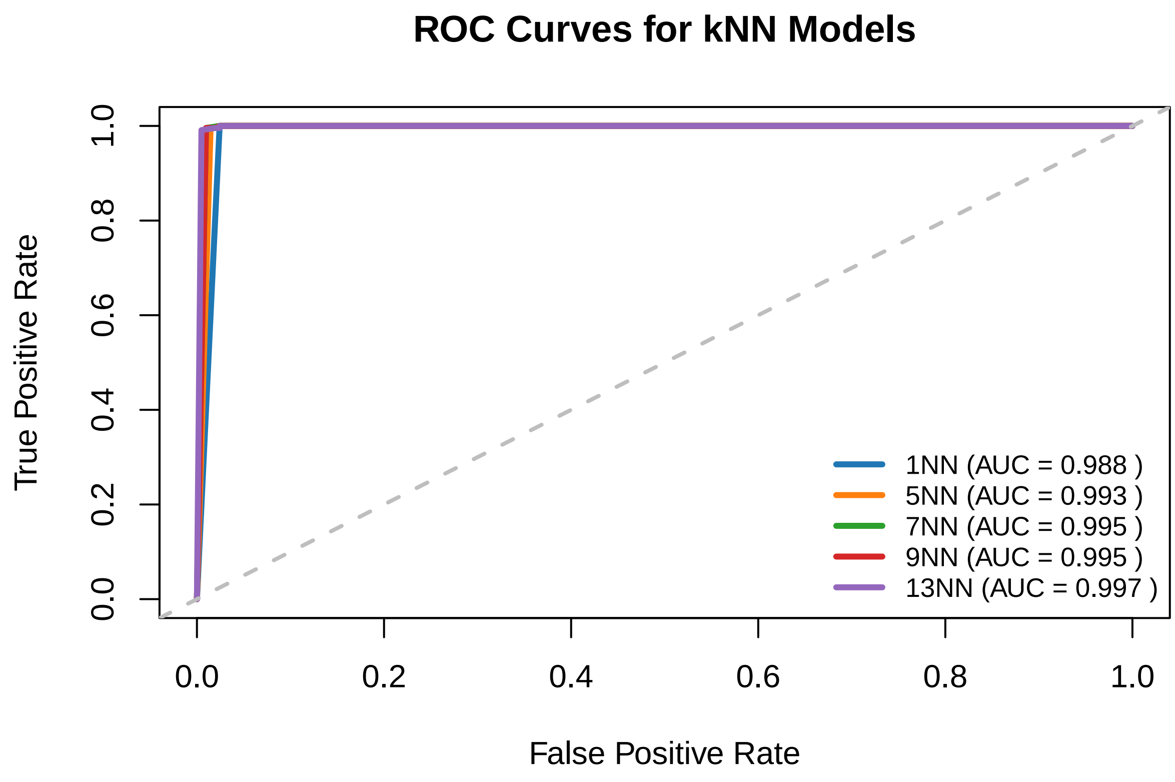
2. Display both your training confusion matrix and your test confusion matrix

```
##
## Evaluating confusion matrix at  k = 1
##
## Training Confusion Matrix for k = 1 :
##          Actual
## Predicted    1    7
##         1 1124    0
##         7    0 1044
##
## Test Confusion Matrix for k = 1 :
##          Actual
## Predicted   1   7
##         1 227   5
##         7   0 201
##
## Evaluating confusion matrix at  k = 5
##
## Training Confusion Matrix for k = 5 :
##          Actual
## Predicted    1    7
##         1 1123   19
##         7    1 1025
##
## Test Confusion Matrix for k = 5 :
##          Actual
## Predicted   1   7
##         1 227   5
##         7   0 201
##
## Evaluating confusion matrix at  k = 7
##
## Training Confusion Matrix for k = 7 :
##          Actual
## Predicted    1    7
##         1 1123   20
##         7    1 1024
##
## Test Confusion Matrix for k = 7 :
##          Actual
## Predicted   1   7
##         1 227   6
##         7   0 200
##
## Evaluating confusion matrix at  k = 9
##
## Training Confusion Matrix for k = 9 :
##          Actual
## Predicted    1    7
##         1 1123   25
##         7    1 1019
##
## Test Confusion Matrix for k = 9 :
##          Actual
## Predicted   1   7
##         1 227   6
##         7   0 200
##
## Evaluating confusion matrix at  k = 13
##
## Training Confusion Matrix for k = 13 :
##          Actual
## Predicted    1    7
##         1 1123   27
##         7    1 1017
##
## Test Confusion Matrix for k = 13 :
##          Actual
## Predicted   1   7
##         1 227   7
##         7   0 199
```

The obtained confution matrix provide the information that there was no more noise in general but most of time the digit 7 was cofused and predicted as 1.

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

This section we discuss the comparative ROC curves of the five learning machines to check



The ROC curve shows that the curve are likely to be the same. The model 1NN, 9NN, and 13NN perform equally which was the best in this case. The next was 5NN and 7NN which also appear to be the same.

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

This image show case where the machime mistakenly make noise and confuse digit 1 and 7. because they look alike in wrting style it is meaningful that machine can confuse the digit.

5. Comment on any pattern that might have emerged.

In this classification task, we observed that the model effectively identifies the digits '1' and '7' in most cases, as indicated by the true positives (TP) and true negatives (TN). This digit is more likely to be confused since it have the same written structure. However, false positives (FP) and false negatives (FN) highlight that the model struggles with certain variations, such as distorted shapes or ambiguous handwriting. These misclassifications suggest that the model's generalization ability is challenged by noisy or diverse data. The ROC curves reveal the model's ability to distinguish between the two digits, with the AUC values providing an overall measure of performance. The pattern of errors emphasizes the importance of clean, consistent data and the potential for improvement with better data quality and more sophisticated models. Enhancing the model's robustness to variations could lead to better performance on unseen data.

The link of video part https://youtu.be/r94V57DjZcs