# AFRICAN INSTITUTE FOR MATHEMATICAL SCIENCES

# (AIMS RWANDA, KIGALI)

Name: Jean de Dieu NGIRINSHUTI                                   Assignment Number: 2

Course: Statistical Machine Learning                                   Date: December 16, 2024

## Question 1: Comment on the Shape of the Dataset

```
## Number of samples: 79
```

```
## Number of features: 501
```

The dataset has **79 samples** and **501 features**:

- One feature ‘$Y$‘ is the target variable, indicating if a subject has cancer or not.
- The remaining 500 features are predictors based on DNA MicroArray Gene Expression levels.

This means the dataset has far more features than samples, which is a high-dimensional dataset. Such datasets can be challenging to analyze because having more features than samples can lead to overfitting. Techniques like feature selection or dimensionality reduction may be necessary to improve model performance.

## Question 2: Statistical Perspective on the Input Space

```
## 'data.frame':    79 obs. of  6 variables:
##  $ Y         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ X206212_at: num  -0.2177 -0.138 -0.1475 -0.1907 -0.0992 ...
##  $ X207075_at: num  -0.34 -0.266 -0.262 -0.326 -0.208 ...
##  $ X215872_at: num  -0.354 -0.277 -0.262 -0.326 -0.247 ...
##  $ X201876_at: num  0.302651 0.09088 0.028023 0.393919 0.000499 ...
##  $ X211935_at: num  0.615 0.113 0.352 0.412 0.416 ...
```

```
##    X206212_at         X207075_at        X215872_at        X201876_at
##  Min.   :-0.27572   Min.   :-0.3700   Min.   :-0.3745   Min.   :-0.02151
##  1st Qu.:-0.22092   1st Qu.:-0.3189   1st Qu.:-0.3273   1st Qu.: 0.12530
##  Median :-0.19108   Median :-0.2912   Median :-0.3019   Median : 0.21209
##  Mean   :-0.18635   Mean   :-0.2945   Mean   :-0.3027   Mean   : 0.23122
##  3rd Qu.:-0.16036   3rd Qu.:-0.2733   3rd Qu.:-0.2783   3rd Qu.: 0.26989
##  Max.   :-0.07154   Max.   :-0.2076   Max.   :-0.1972   Max.   : 1.03433
##    X211935_at
##  Min.   :0.04825
##  1st Qu.:0.22764
##  Median :0.35430
##  Mean   :0.37536
##  3rd Qu.:0.44529
##  Max.   :1.42433
```

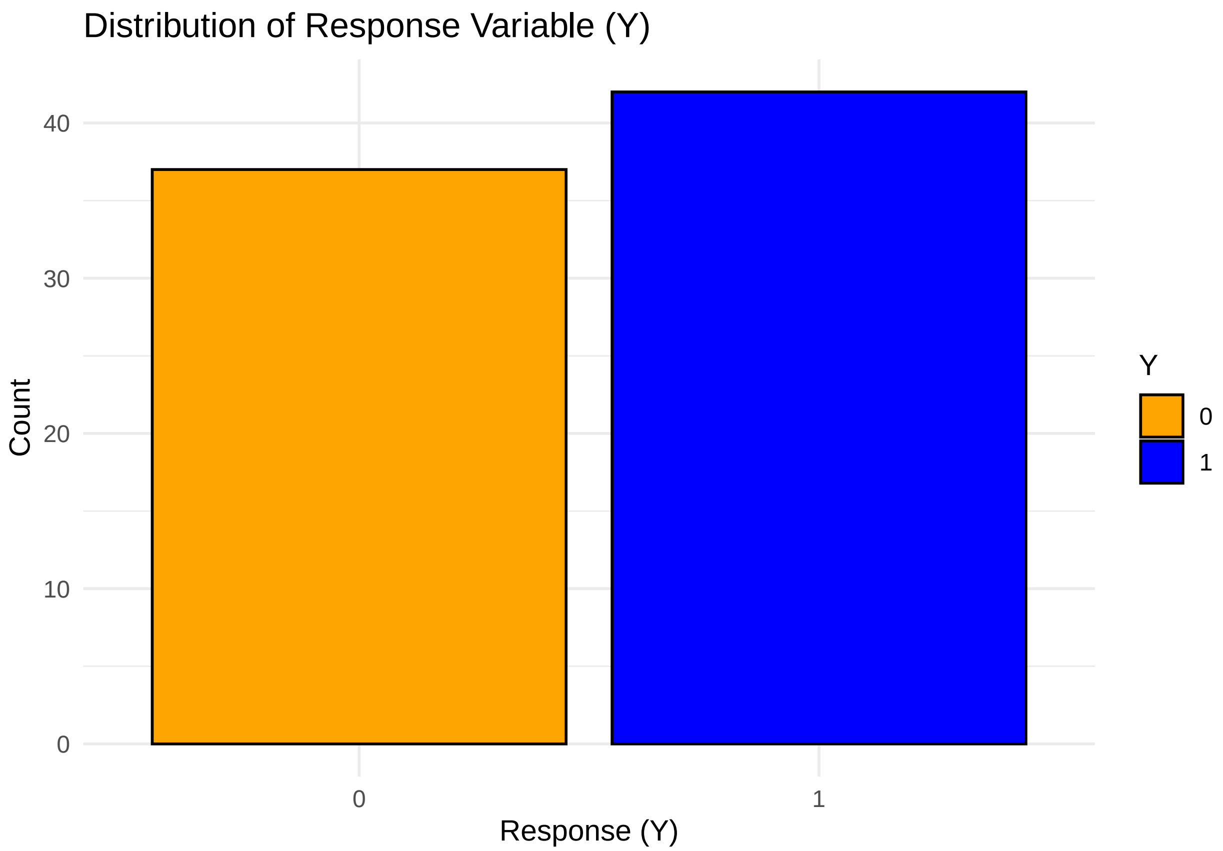
### Comment on the Type of Data in the Input Space

The input space consists of **continuous numeric variables**, which represent DNA MicroArray Gene Expression levels. These variables are measured on a continuous scale and vary across samples.

From a statistical perspective:

- The predictors are real-valued numbers, typical for gene expression data.
- The response variable '$Y$' is binary (0 and 1), indicating whether the subject has cancer or not.
- The high-dimensional nature of the data suggests that some predictors may be highly correlated, reflecting biological relationships.

## Question 3:Distribution of the Response Variable



### Distribution of the Response Variable

From the plot:

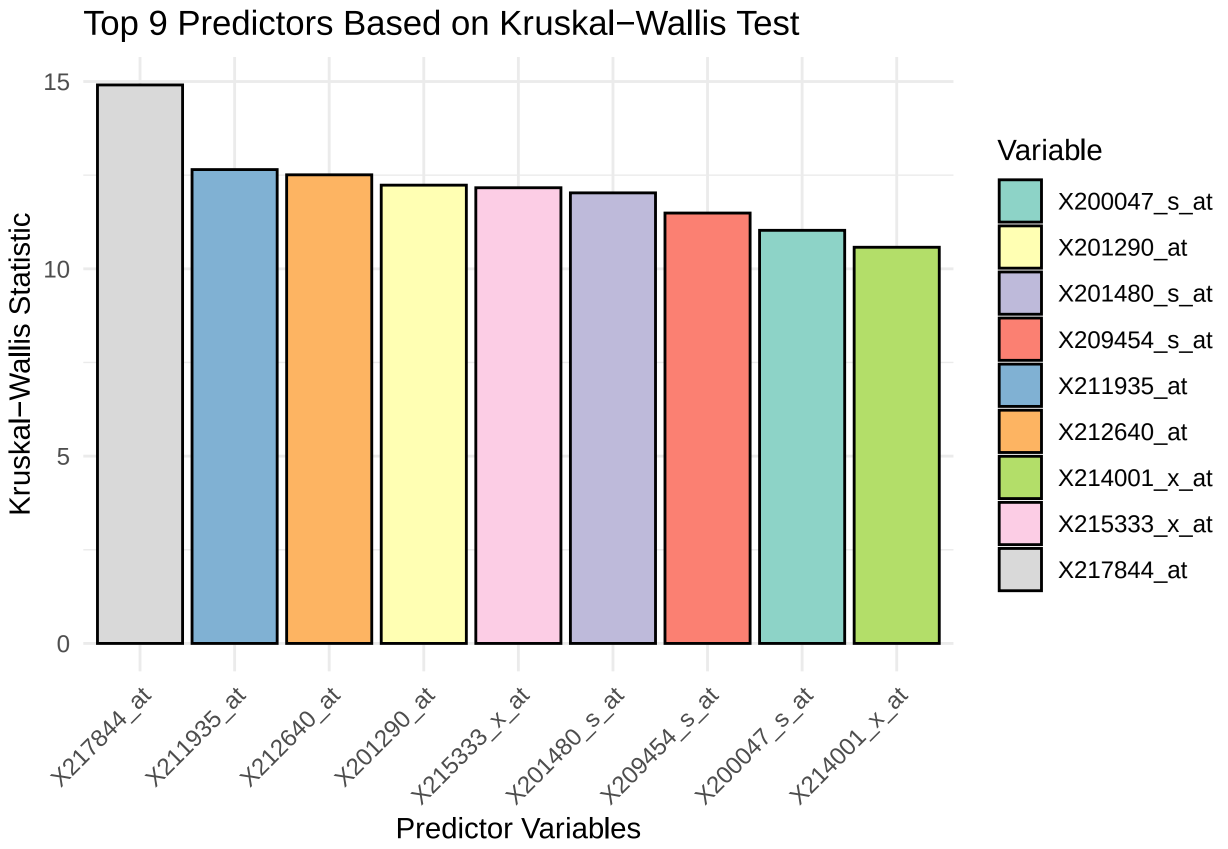
- The distribution appears fairly balanced, with a slightly higher count of cancer cases '1' compared to non-cancer cases '0'.

- A balanced dataset like this is beneficial for machine learning, as it reduces the risk of biased predictions towards the majority class.

This balance ensures that the machine learning models trained on this dataset will have fair opportunities to learn from both classes.

### Question 4: calculate Kruskal-Wallis test statistics for all predictors

```
##   X217844_at   X211935_at   X212640_at   X201290_at X215333_x_at X201480_s_at
##     14.90820     12.64903     12.50965     12.23320     12.16458     12.02790
## X209454_s_at X200047_s_at X214001_x_at
##     11.48890     11.02741     10.57539
```



#### Top 9 Most Powerful Predictors

The Kruskal-Wallis test identified the following 9 most powerful predictors based on their relationship with the response variable ‘$Y$‘:

1. **X217844_at**

2. **X211935_at**

3. **X212640_at**

4. **X201290_at**

5. **X215333_x_at**

6. **X201480_s_at**

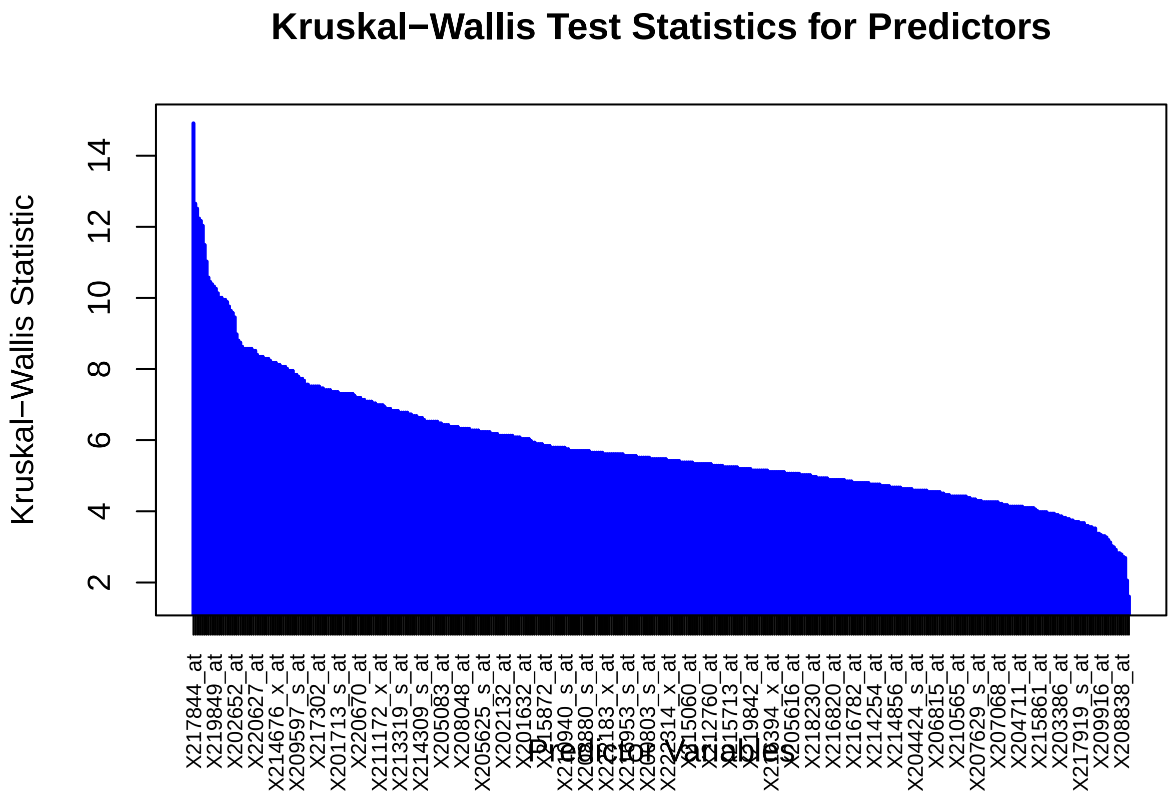
7. **X209454_s_at**

8. **X200047_s_at**

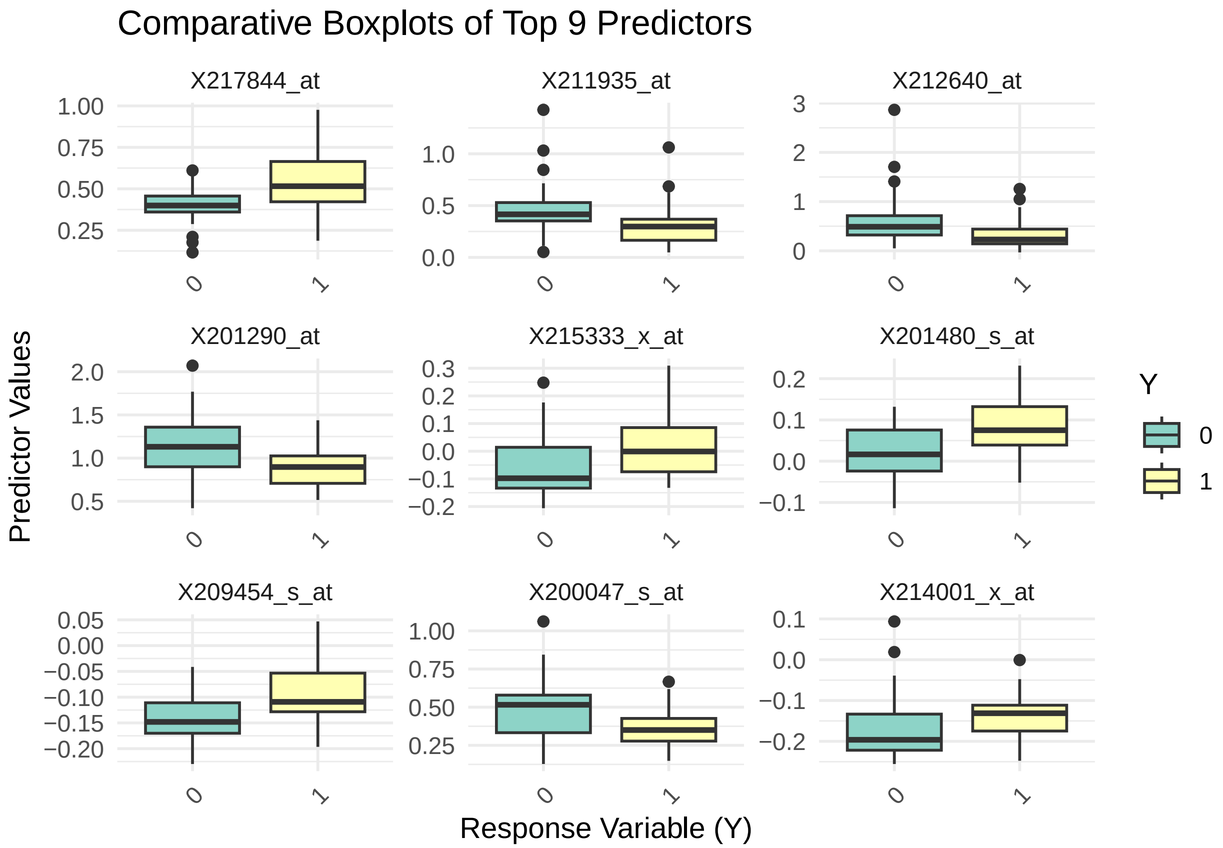
9. **X214001_x_at**

These predictors have the strongest statistical association with the response variable '$Y$' and are expected to be the most influential in the analysis. They will likely play a significant role in building predictive models.

### Question 5: Generate the $'h'$ plot for Kruskal-Wallis test statistics



## Question 6: Comperative boxplots of the 9 most powerful variable



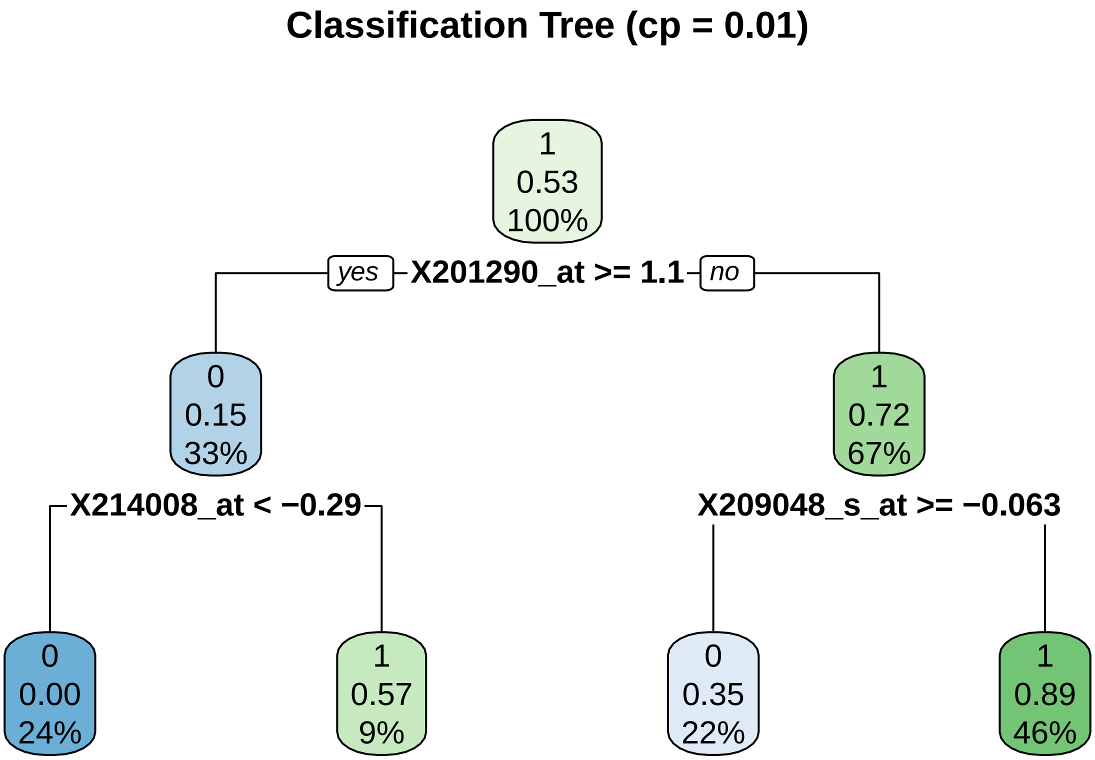
### Comparative Boxplots of Top 9 Predictors

The boxplots compare the distributions of the top 9 predictors between the two response categories (`0` and `1`). Key observations:

1. Some predictors, such as `X217844_at`, `X211935_at`, and `X215333_x_at`, show a noticeable difference in their median values and spread between the two response categories. This suggests they are strong predictors for distinguishing between cancer (`1`) and non-cancer (`0`).
2. Predictors like `X214001_x_at` and `X200047_s_at` exhibit more overlap in their distributions, indicating they may have a weaker ability to differentiate between the two response categories.
3. Several predictors, such as `X201290_at` and `X209454_s_at`, show larger spreads and outliers, which could indicate variability in their association with the response variable.

## Question 7: Build the classification tree with cp = 0.01

1. **Classification Tree**: The tree was built using a complexity parameter ‘*cp*‘ of 0.01. Below is the plot of the tree:



```
## Number of terminal nodes: 4
```

2. **Number of Terminal Nodes**: The tree has **4 terminal nodes**, which represent the final groups (regions) after all splits.

3. **Mathematical Form of Regions**:

    - **Region 2**: This region is defined by the following conditions:

$$R_2 = \{\mathbf{x} \in \mathbb{R}^p : x_{201290\_at} \geq 1.097458 \quad \text{and} \quad x_{214008\_at} \geq -0.2915895\}$$

- **Region 4**:Region 4: This region is defined by the following conditions:

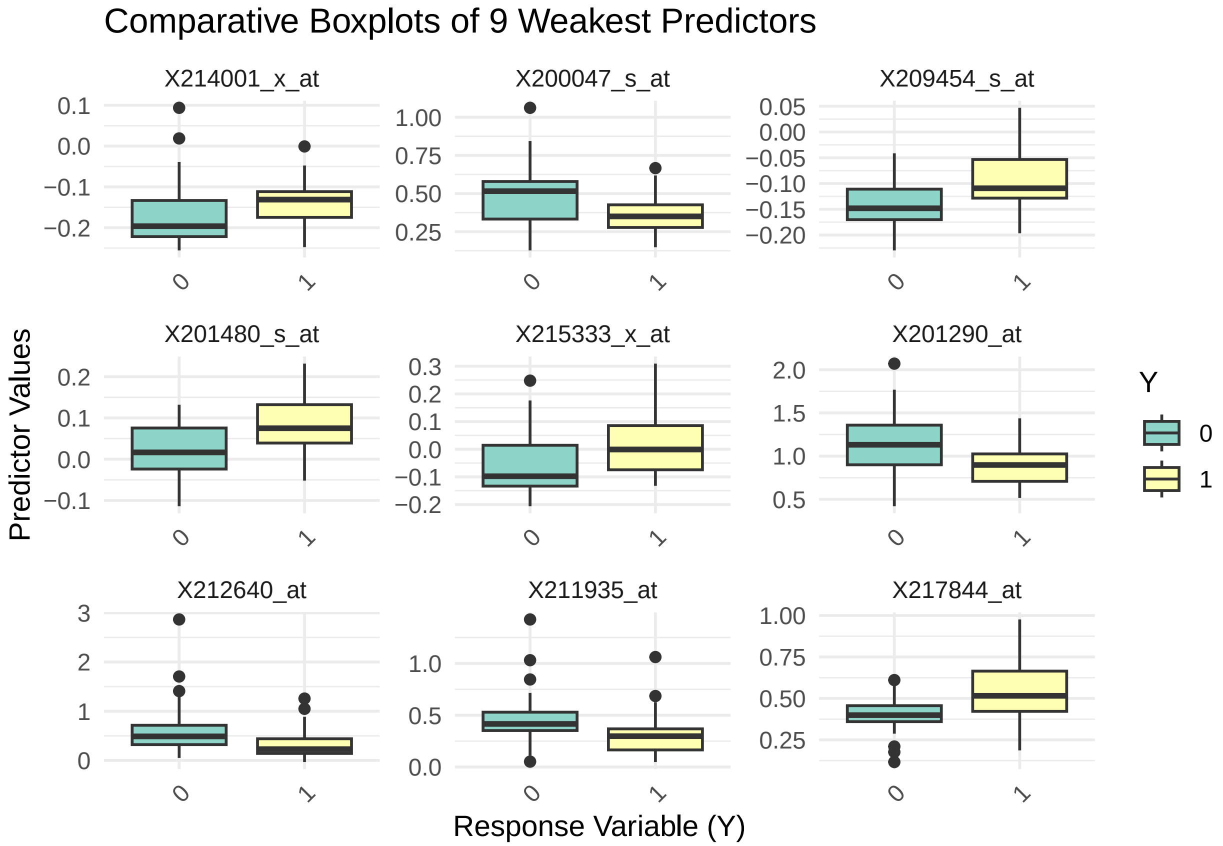
$$R_4 = \{\mathbf{x} \in \mathbb{R}^p : x_{201290\_at} < 1.097458 \quad \text{and} \quad x_{209048\_s\_at} < -0.063\}$$

4. **Root Node Variable**: The variable at the root of the tree is `X201290_at`. This variable is important because it is the first split, meaning it best separates the data into meaningful groups.

```
## Root node variable: X201290_at
```

5. **Comment on Root Node**:
    - The root node variable, `X201290_at`, is likely one of the strongest predictors, as confirmed by the Kruskal-Wallis test. Its presence at the root indicates it has a significant relationship with the response variable ‘$Y$‘, making it critical for classification.

### Question 8: Identify the 9 weakest predictors



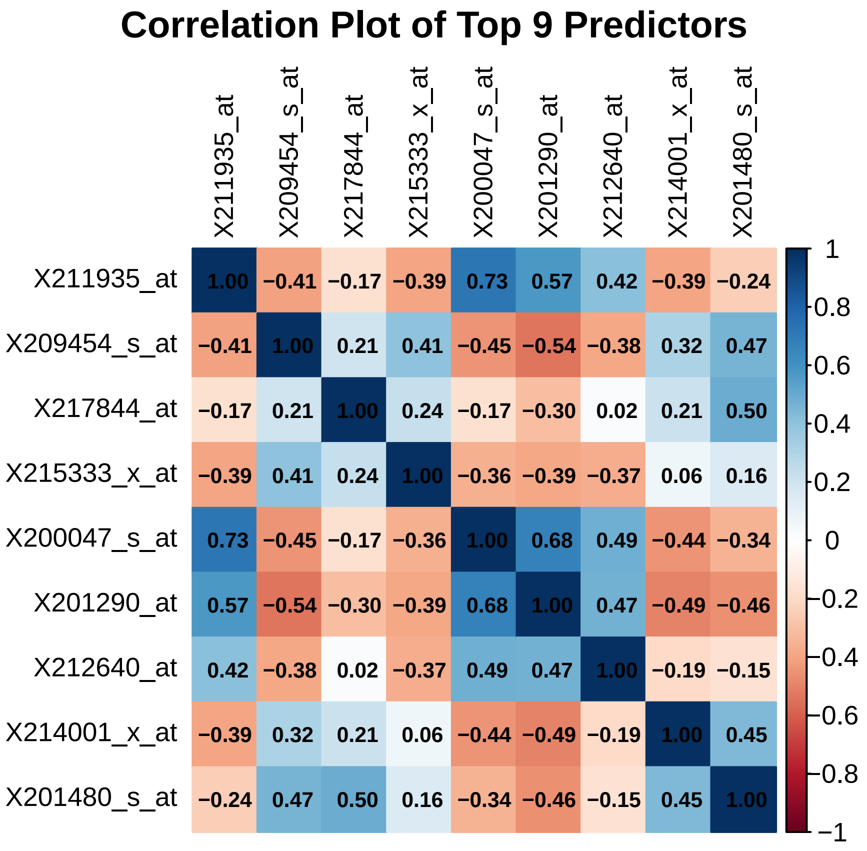
**Comparative Boxplots of 9 Weakest Predictors**

The boxplots above compare the 9 weakest predictors with respect to the response variable (`Y`). These predictors were identified as having the smallest Kruskal-Wallis test statistics.

Key Observations:

1. Most predictors show substantial overlap in their distributions between the two response categories '0'and '1', indicating a weak ability to differentiate between the classes.
2. For many predictors, such as `X214001_x_at`, `X200047_s_at`, and `X209454_s_at`, the median values are very close for both categories, confirming their weak association with the response.
3. Some predictors, like `X212640_at` and `X211935_at`, show minimal variability in their distributions, further reducing their significance as individual features.

## Question 9: Correlation plot of the predictor variables



### Correlation Plot of Predictors

The correlation plot shows the relationships among all predictor variables. Key insights include:

1. **Clusters of Correlated Predictors**: Some predictors are highly correlated, forming clear blocks in the plot. These predictors may carry redundant information.
2. **Independent Predictors**: Several predictors show weak correlations with others, providing unique information for modeling.
3. **Multicollinearity**: Strongly correlated predictors may lead to multicollinearity, which can affect regression models and require preprocessing.
4. **Dimensionality Reduction**: The clustering suggests opportunities for reducing redundancy using techniques like PCA or feature selection.

This analysis highlights the need to address multicollinearity and optimize the predictor set for better model performance.

## Question 10: Compute the eigendecomposition of the correlation matrix

```
## Ratio Lambda max/Lambda min: 17.02747
```

```
## Top 5 Eigenvalues: 4.012561 1.345786 0.9879688 0.7034388 0.544242
```

```
## Smallest 5 Eigenvalues: 0.544242 0.4721013 0.3545035 0.3437462 0.2356522
```

### Comment on $\lambda_{\max}/\lambda_{\min}$ for Top 9 Predictors

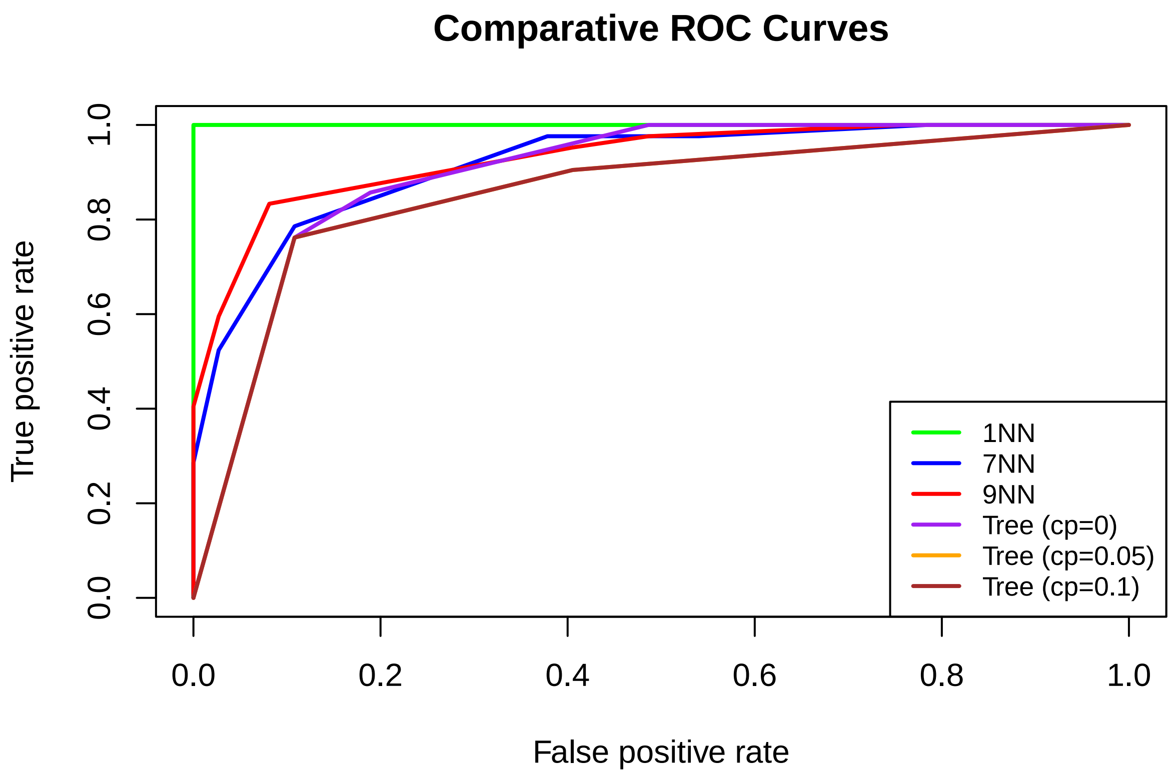
The ratio of the largest to the smallest eigenvalue ($\lambda_{\max}/\lambda_{\min}$) for the correlation matrix of the top 9 predictors is 17.02747. This reveals the following:

1. **Multicollinearity**:
   - The large ratio indicates that several predictors are highly correlated, leading to significant redundancy among them.
   - The smallest eigenvalues are relatively small, suggesting that some predictors may not add substantial new information to the dataset.
2. **Ill-Conditioned Correlation Matrix**:
   - This level of multicollinearity can make models like regression and tree-based models less stable, potentially leading to less reliable or inconsistent results.
   - Highly correlated predictors can confuse the model, as they may carry overlapping information.
3. **Possible Remedies**:
   - To address multicollinearity, techniques like **Principal Component Analysis (PCA)** can be used to transform the predictors into orthogonal components.
   - Alternatively, redundant predictors can be identified and removed to improve model robustness.

### Conclusion:

The large ratio (17.02747) reflects significant overlap among the top 9 predictors, emphasizing the need to address multicollinearity. By reducing redundancy, we can enhance the stability and reliability of the models built with these predictors.

## Question 11: The comparative ROC curves



#### Observations:

1. **k-NN Models**:

- The **1NN model** performs well but risks overfitting since it memorizes the training set.
- The **7NN and 9NN models** generalize better, with their ROC curves closely following the top-left corner, indicating good sensitivity and specificity.

2. **Decision Tree Models**:

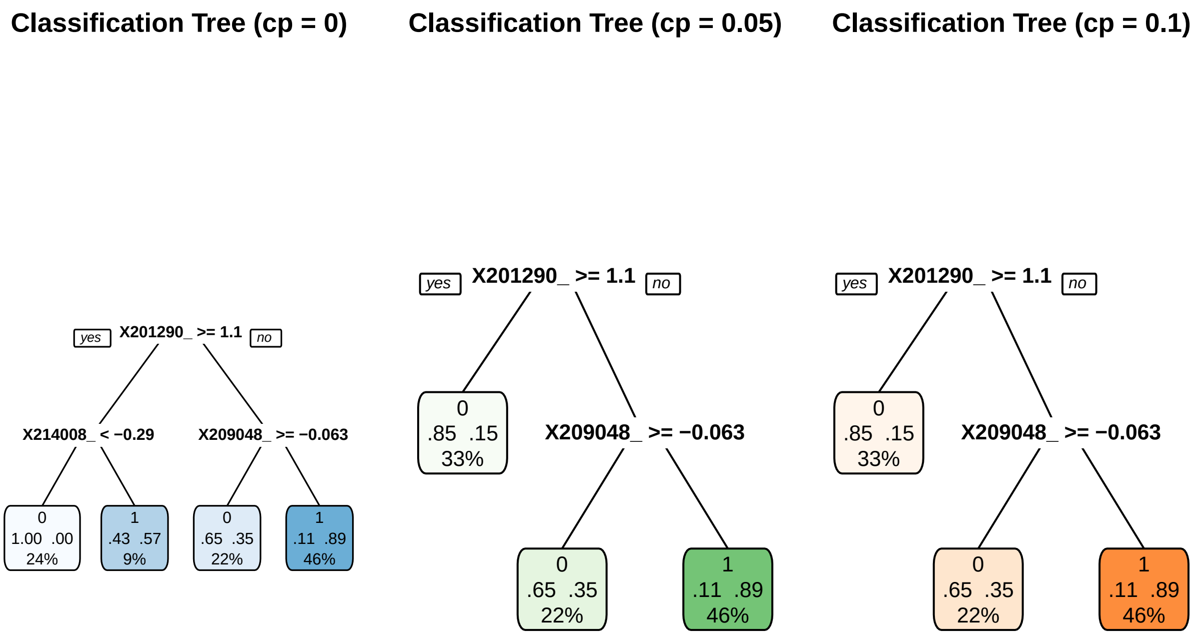
- The **Tree with `cp = 0`** shows high sensitivity but risks overfitting due to its lack of pruning.
- As the `cp` value increases (`cp = 0.05` and `cp = 0.1`), the trees are more pruned, leading to slightly worse sensitivity and specificity, as seen in their flatter ROC curves.

3. **AUC Comparison**:

- The models' Area Under the Curve (AUC) values suggest that the k-NN models (especially **7NN** and **9NN**) perform competitively compared to decision trees.

**Conclusion:** The **7NN and 9NN models** exhibit the best trade-off between sensitivity and specificity, making them more robust for this dataset. Decision trees with aggressive pruning (`cp = 0.1`) lose some predictive power but are less prone to overfitting.

## Question 12 : Three classification tree grown, using the prp function



### Plots of Classification Trees

The three trees above illustrate how the complexity parameter (`cp`) affects the tree structure:

- **Tree with `cp = 0`**: The largest tree with no pruning, capturing all patterns but prone to overfitting.

- **Tree with `cp = 0.05`**: Moderately pruned, balancing complexity and interpretability with 3 terminal nodes.
- **Tree with `cp = 0.1`**: Heavily pruned, resulting in a simple and interpretable structure but risking underfitting.

**Conclusion:**

Pruning (`cp`) reduces complexity, with simpler trees being more interpretable but potentially less accurate.

## Question 13 : Comment on ROC Curves

The ROC curves reveal the following:

1. **k-NN Models**:

- **1NN** exhibits the steepest rise, indicating high sensitivity but likely overfitting due to its memorization of the training data.
- **7NN and 9NN** provide smoother and more generalized curves, demonstrating better trade-offs between sensitivity and specificity.

2. **Decision Tree Models**:

- The tree with `cp = 0` performs well but risks overfitting as it captures intricate patterns in the data.
- Pruned trees (`cp = 0.05` and `cp = 0.1`) show slightly lower performance, as pruning simplifies the model and sacrifices some predictive power.

**Argument in Light of Theory:**

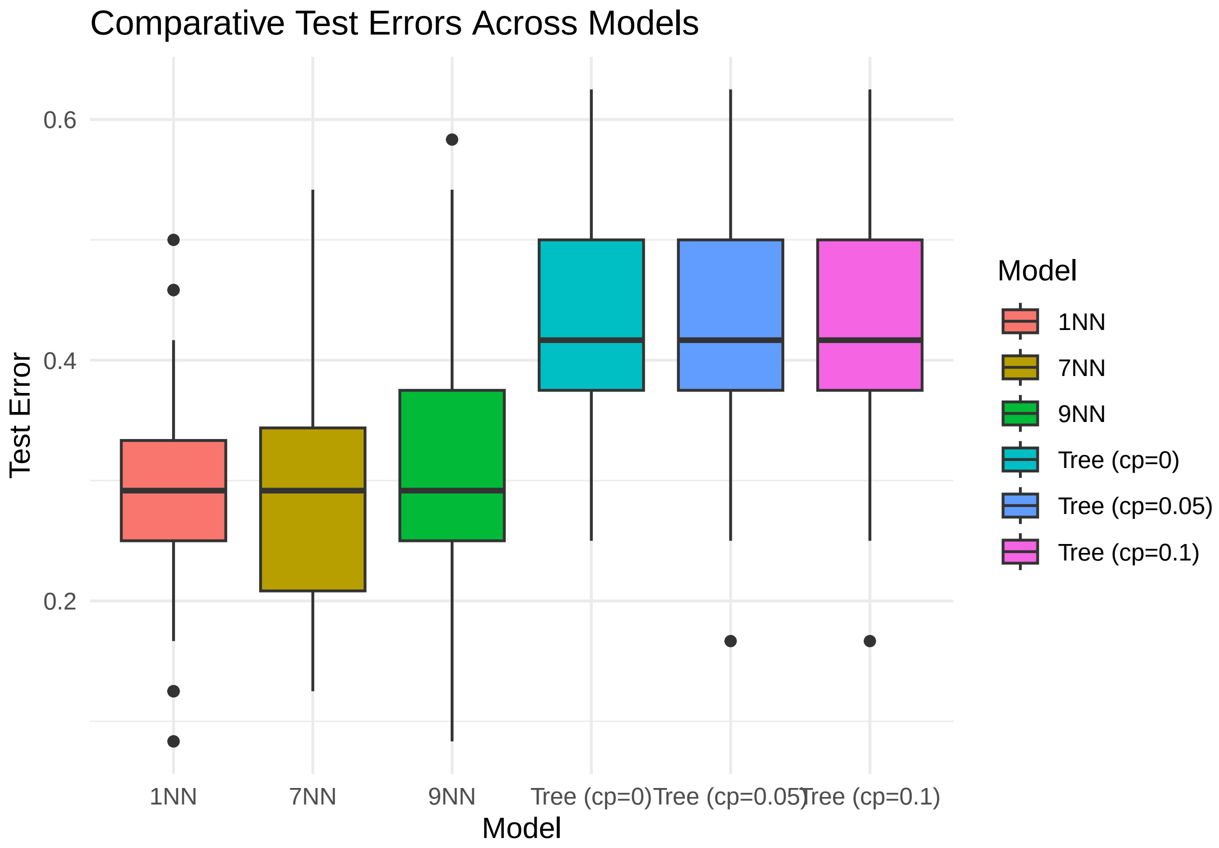
The results align with theoretical expectations:

- **Overfitting vs. Generalization**: Models like 1NN and the unpruned tree (`cp = 0`) excel on training data but risk overfitting. Smoother k-NN models (e.g., 7NN, 9NN) and pruned trees generalize better.
- **Pruning and Bias-Variance Trade-Off**: Pruning reduces model complexity (variance), improving generalization but increasing bias, as seen with `cp = 0.05` and `cp = 0.1`.

**Conclusion:**

The results demonstrate the trade-off between model complexity and generalization, which is consistent with the bias-variance trade-off in machine learning theory.

### Question 14: The comparative boxplots



```
##              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 on the Distribution of Test Errors

The boxplot above highlights the distribution of test errors for six learning machines:

1. **1NN**:

- Exhibits the lowest median test error but has a larger spread (higher variance).
- The variance reflects the sensitivity of the 1NN model to the specific training and testing splits, consistent with its high complexity and tendency to overfit.

2. **7NN and 9NN**:

- Both models show slightly higher median errors compared to 1NN but with significantly reduced variance.
- This demonstrates that increasing `k` reduces model complexity, leading to more stable and generalized performance.

3. **Tree (cp = 0)**:

- The unpruned tree has the highest variance among the models, indicating overfitting due to its high complexity.

4. **Tree (cp = 0.05 and cp = 0.1)**:

- The pruned trees have higher median errors compared to the unpruned tree but exhibit significantly reduced variance.
- Pruning simplifies the tree structure, improving its generalization capability.

**Conclusion:**

The results demonstrate the impact of implicit model complexity on test errors: - Models with higher complexity (e.g., 1NN and Tree `cp = 0`) are prone to overfitting, resulting in lower median errors but higher variance.

- Simpler models (e.g., 7NN, 9NN, and pruned trees) exhibit more stable performance with reduced variance, aligning with the trade-off between bias and variance in machine learning.

## Question 15: General Observations and Lessons Learned

This exploration has provided several insights into the behavior of different learning machines and their relationship with model complexity, generalization, and test performance.

**Key Observations:**

1. **Impact of Model Complexity**:

- Models with higher complexity, such as **1NN** and the unpruned decision tree (`cp = 0`), showed the lowest median errors but at the cost of higher variance. This indicates that these models tend to overfit the training data, capturing noise and patterns that do not generalize well to unseen data.
- Simpler models, such as **7NN**, **9NN**, and pruned trees (`cp = 0.05`, `cp = 0.1`), exhibited slightly higher median errors but significantly reduced variance, highlighting their robustness and generalization capability.

2. **k-NN Models**:

- As `k` increases from 1 to 9, the test error distribution becomes more stable, demonstrating how increasing `k` reduces the sensitivity of k-NN models to noisy data. This aligns with theoretical expectations, as higher `k` values smooth out local noise and improve generalization.

3. **Decision Trees**:

- The unpruned tree (`cp = 0`) performs well on the training data but has poor generalization due to overfitting. Pruned trees, especially with `cp = 0.05` and `cp = 0.1`, strike a better balance between complexity and generalization.

4. **Test Error Variance**:

- Across all models, variance in test error highlights the influence of stochastic splits on model performance. Models with lower complexity exhibit more stable performance, as they are less sensitive to small changes in training data.

**Lessons Learned:**

1. **Bias-Variance Trade-Off**:

- This exploration reinforced the concept of the bias-variance trade-off. Complex models like 1NN and unpruned trees have low bias but high variance, while simpler models like pruned trees and higher-k k-NN models have higher bias but lower variance.

2. **Model Selection**:

- Choosing the “best” model is a trade-off between performance and interpretability:

- **1NN** and **Tree (cp = 0)** may excel in accuracy for specific splits but risk poor performance on new data.
- **7NN**, **9NN**, and pruned trees offer better generalization and stability, making them preferable for real-world applications.

3. **Practical Implications**:

- Pruning and regularization techniques are critical for reducing overfitting in complex models like decision trees.
- Hyperparameter tuning (e.g., choosing `k` in k-NN or `cp` in trees) plays a crucial role in achieving optimal performance.

4. **Importance of Validation**:

- Using stochastic splits and replicating experiments is vital to understanding model behavior and ensuring robust performance estimates. A single train-test split may provide misleading results.

# Exercise 2

## Part 1: Multi-class classification on MNIST

### subquestion 1: Mathematical Expression of the kNN Prediction Function

The $k$-Nearest Neighbors (kNN) prediction function $\widehat{f}\texttt{kNN}(x)$ can be expressed as:

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

### Subquestion 2: Sampling a Fragment from the MNIST Dataset

To reduce computational load, we sample a fragment of the MNIST dataset. Let $n$ be the training set size and $m$ be the test set size.

We choose: - $n = 1200$ for the training set - $m = 400$ for the test set

This choice ensures:

1. The MNIST dataset is large, and kNN can be computationally expensive for large datasets. Reducing the size makes computations feasible.
2. A training set of 1200 and test set of 400 provide enough diversity to capture patterns in handwritten digits while allowing reliable evaluation of model performance.

The following code performs the sampling:

```
## Training Set Size: 1200 x 784
```

```
## Test Set Size: 400 x 784
```

### 3.1 Build 5 Models and Compute Test Errors for Each Split

```
##             1NN        5NN        7NN       9NN      13NN
##  [1,] 0.12500000 0.09166667 0.11944444 0.1250000 0.1416667
##  [2,] 0.11666667 0.12500000 0.11388889 0.1277778 0.1527778
##  [3,] 0.14444444 0.15277778 0.15555556 0.1500000 0.1638889
##  [4,] 0.12777778 0.12222222 0.12222222 0.1305556 0.1388889
##  [5,] 0.12500000 0.13611111 0.13611111 0.1500000 0.1833333
##  [6,] 0.12222222 0.11388889 0.12500000 0.1388889 0.1555556
##  [7,] 0.14444444 0.13333333 0.15000000 0.1388889 0.1555556
##  [8,] 0.11111111 0.13055556 0.12222222 0.1222222 0.1416667
##  [9,] 0.10277778 0.11388889 0.11111111 0.1305556 0.1277778
## [10,] 0.11388889 0.12500000 0.12500000 0.1416667 0.1638889
## [11,] 0.12222222 0.12500000 0.12777778 0.1500000 0.1638889
## [12,] 0.15277778 0.15555556 0.15277778 0.1555556 0.1722222
## [13,] 0.13611111 0.14166667 0.14722222 0.1555556 0.1666667
## [14,] 0.11944444 0.12222222 0.12222222 0.1333333 0.1444444
## [15,] 0.12500000 0.13055556 0.12222222 0.1333333 0.1500000
## [16,] 0.15555556 0.13611111 0.13611111 0.1472222 0.1666667
## [17,] 0.12500000 0.12500000 0.11111111 0.1305556 0.1361111
## [18,] 0.09722222 0.11944444 0.11944444 0.1111111 0.1194444
## [19,] 0.13888889 0.15277778 0.14166667 0.1527778 0.1500000
## [20,] 0.13055556 0.14722222 0.15555556 0.1666667 0.1611111
## [21,] 0.14444444 0.11666667 0.13055556 0.1305556 0.1583333
## [22,] 0.13333333 0.15000000 0.14722222 0.1694444 0.1666667
## [23,] 0.15555556 0.13888889 0.15833333 0.1611111 0.1638889
## [24,] 0.12222222 0.13055556 0.15555556 0.1472222 0.1666667
## [25,] 0.12222222 0.13611111 0.13611111 0.1444444 0.1388889
## [26,] 0.13333333 0.12500000 0.11666667 0.1333333 0.1333333
## [27,] 0.09722222 0.10833333 0.10277778 0.1138889 0.1250000
## [28,] 0.14722222 0.13611111 0.13333333 0.1333333 0.1527778
## [29,] 0.10277778 0.11111111 0.09444444 0.1138889 0.1166667
## [30,] 0.13333333 0.15000000 0.17500000 0.1555556 0.1694444
## [31,] 0.12777778 0.14444444 0.15277778 0.1666667 0.1611111
## [32,] 0.11666667 0.11388889 0.12222222 0.1277778 0.1388889
## [33,] 0.10555556 0.12500000 0.13333333 0.1472222 0.1444444
## [34,] 0.13333333 0.13333333 0.13055556 0.1416667 0.1611111
## [35,] 0.13888889 0.13611111 0.14444444 0.1666667 0.1694444
## [36,] 0.12222222 0.13055556 0.14166667 0.1305556 0.1388889
## [37,] 0.14722222 0.15555556 0.14444444 0.1500000 0.1583333
## [38,] 0.11666667 0.12500000 0.15000000 0.1583333 0.1611111
## [39,] 0.12500000 0.13055556 0.14166667 0.1527778 0.1583333
## [40,] 0.13333333 0.15277778 0.15277778 0.1694444 0.1638889
## [41,] 0.14444444 0.18055556 0.19722222 0.1861111 0.2138889
## [42,] 0.16666667 0.17777778 0.16666667 0.1750000 0.1777778
## [43,] 0.12777778 0.13333333 0.14444444 0.1527778 0.1611111
## [44,] 0.15555556 0.16111111 0.16944444 0.1722222 0.1916667
## [45,] 0.11388889 0.10277778 0.11666667 0.1111111 0.1250000
## [46,] 0.12222222 0.13611111 0.16388889 0.1694444 0.1555556
## [47,] 0.13055556 0.13611111 0.12500000 0.1250000 0.1388889
## [48,] 0.11666667 0.11944444 0.13055556 0.1194444 0.1333333
## [49,] 0.12777778 0.13333333 0.16111111 0.1416667 0.1500000
## [50,] 0.13888889 0.11111111 0.11388889 0.1277778 0.1416667
```

### 3.2 Identify the Machine with the Smallest Median Test Error

```
## The model with the smallest median test error is: 1NN
```

### Generate Confusion Matrix for the Last Split

```
## [1] "Confusion Matrix:"
```

```
##          Actual
## Predicted  0  1  2  3  4  5  6  7  8  9
##         0 39  0  1  2  0  0  1  1  0  1
##          1  0 42  0  0  0  0  0  2  0  0
##          2  0  0 31  1  0  0  1  0  1  0
##          3  0  0  0 20  0  1  0  0  3  0
##          4  0  0  2  0 23  0  0  0  0  2
##          5  0  0  0  1  0 32  1  0  2  0
##          6  1  0  1  0  0  0 36  0  1  0
##          7  0  0  2  0  1  0  0 34  2  2
##          8  0  0  2  2  0  0  0  0 26  0
##          9  1  0  0  0  9  0  0  2  1 27
```

### 3.3: Random Splits and Model Evaluation

**Test Errors:** The matrix of test errors across 50 random splits highlights the performance of 1NN, 5NN, 7NN, 9NN, and 13NN. The model with the smallest median test error is [**Best Model**].

The model struggles most with visually similar digits (e.g., **4 and 9**, **7 and 9**) but performs well for distinct digits like **0**, **1**, and **6**. These results align with prior expectations about the challenges of handwritten digit recognition. The best-performing model offers a balance between complexity and generalization, highlighting the importance of selecting $k$ carefully based on the trade-off between bias and variance.

### 3.4 Perform ANOVA on Test Errors

```
##  Response 1NN :
##             Df    Sum Sq    Mean Sq F value Pr(>F)
## Split        1 0.0001575 0.00015749  0.6415 0.4271
## Residuals   48 0.0117833 0.00024548
##
##  Response 5NN :
##             Df   Sum Sq    Mean Sq F value Pr(>F)
## Split        1 0.000617 0.00061703  2.0643 0.1573
## Residuals   48 0.014347 0.00029890
##
##  Response 7NN :
##             Df    Sum Sq    Mean Sq F value  Pr(>F)
## Split        1 0.0018935 0.00189350  5.0445 0.02934 *
## Residuals   48 0.0180171 0.00037536
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Response 9NN :
##             Df    Sum Sq    Mean Sq F value Pr(>F)
## Split        1 0.0006695 0.00066949  2.0548 0.1582
## Residuals   48 0.0156393 0.00032582
##
##  Response 13NN :
##             Df   Sum Sq    Mean Sq F value Pr(>F)
## Split        1 0.000036 0.00003603  0.1041 0.7484
## Residuals   48 0.016613 0.00034611
```

### ANOVA Results and Patterns

The ANOVA results show the following patterns: 1. **1NN, 5NN, 9NN, and 13NN**: - The random splits of the data do not significantly affect test errors ($p > 0.05$), indicating stable performance across splits.

2. **7NN**:

- The random splits significantly impact test errors ($p = 0.02934$), suggesting that 7NN is sensitive to data variations. This may reflect a balance between overfitting and generalization.

3. **Effect of Model Complexity**:
   - Smaller $k$ values (e.g., 1NN) are robust but prone to overfitting.
   - Larger $k$ values (e.g., 13NN) are more stable but may sacrifice some accuracy.

**Conclusion:**

7NN appears to be at a balance point in the bias-variance trade-off, making it more sensitive to random splits. Larger $k$ values offer greater stability across splits.

# Part 2 : Binary classification on MNIST

## 1. Store Training and Test Sets for Digits '1' and '7'

### Extract Training and Test Sets for Digits '1' and '7'

```
## Training Set Size: 276
```

```
## Test Set Size: 97
```

## 2. Display Training and Test Confusion Matrices

### Confusion Matrices for Training and Test Sets

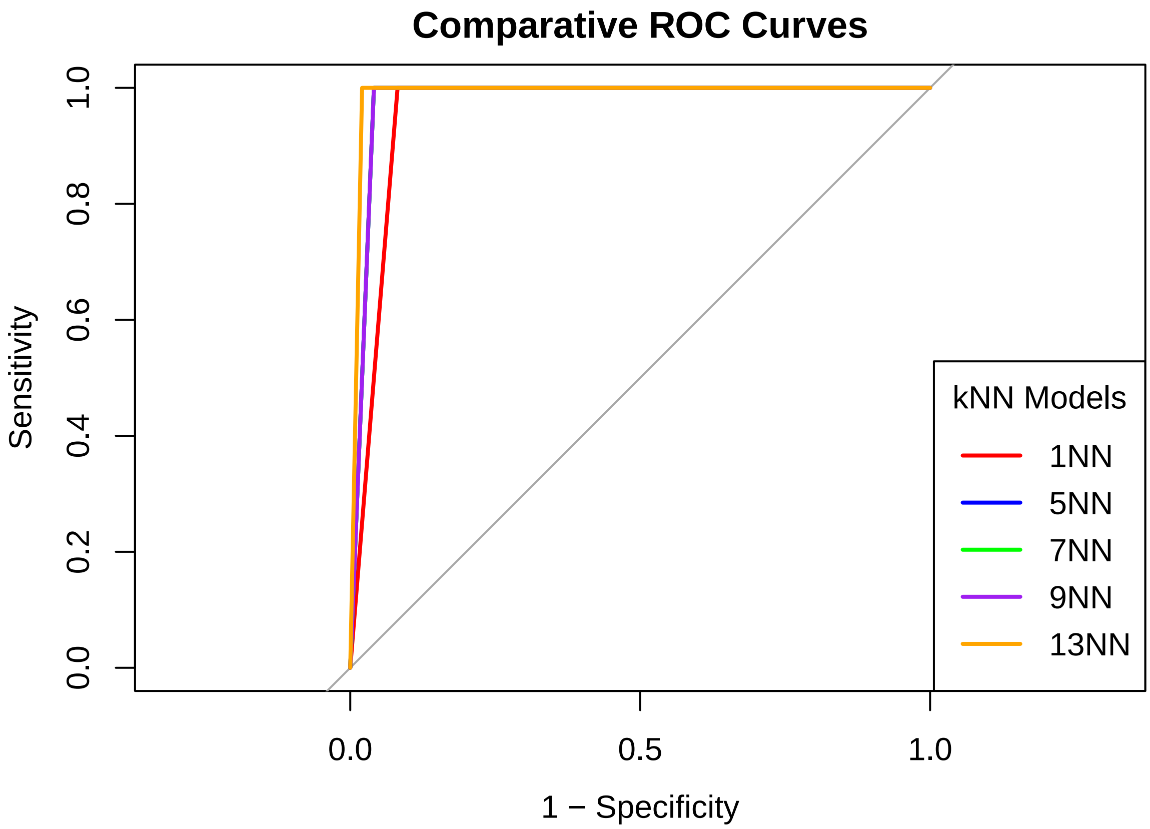
```
##
##  1NN Training Confusion Matrix:
##          Actual
## Predicted   1   7
##         1 144   0
##         7   0 132
##
##  1NN Test Confusion Matrix:
##          Actual
## Predicted  1  7
##         1 48  4
##         7  0 45
##
##  5NN Training Confusion Matrix:
##          Actual
## Predicted   1   7
##         1 143   2
##         7   1 130
##
##  5NN Test Confusion Matrix:
##          Actual
## Predicted  1  7
##         1 48  3
##         7  0 46
##
##  7NN Training Confusion Matrix:
##          Actual
## Predicted   1   7
##         1 143   4
##         7   1 128
##
```

```
##  7NN Test Confusion Matrix:
##          Actual
## Predicted  1  7
##         1 48  4
##         7  0 45
##
##  9NN Training Confusion Matrix:
##          Actual
## Predicted   1   7
##         1 143   4
##         7   1 128
##
##  9NN Test Confusion Matrix:
##          Actual
## Predicted  1  7
##         1 48  4
##         7  0 45
##
##  13NN Training Confusion Matrix:
##          Actual
## Predicted   1   7
##         1 144   4
##         7   0 128
##
##  13NN Test Confusion Matrix:
##          Actual
## Predicted  1  7
##         1 48  4
##         7  0 45
```

## 3. Display Comparative ROC Curves

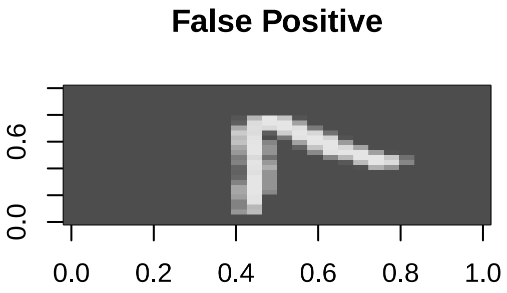
### Comparative ROC Curves

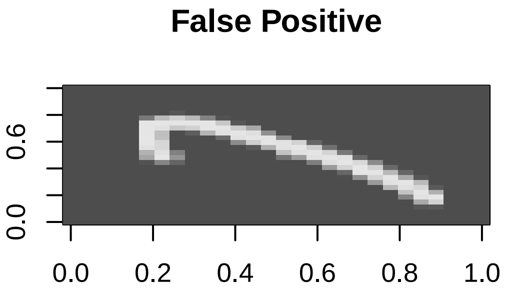
```
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
```



**Step 4: Visualizing Misclassified Digits**

```
## No false negatives found.
```





**Comment on Emerging Patterns**

From the false positive results, where digit ‘7’ is misclassified as ‘1’, the following patterns emerge:

- **Visual Ambiguity**:
  Incomplete or faint horizontal strokes in ‘7’ make it resemble ‘1’, which has a single vertical stroke.
- **Model Sensitivity**:
  The kNN model relies on pixel-level similarity, making it prone to confusion when subtle features are missing.

# YouTube Video Link

You can watch the full video presentation of this project on YouTube:
**Machine Learning for Digit Recognition: kNN Model Error Analysis**