**Syracuse University**

**IST-707 Assignment 6**

ThulasiRam RuppaKrishnan

IST 707Section: 35

Professor Ami Gates

Contents

[Introduction 3](#_Toc10174182)

[Analysis and Models 7](#_Toc10174183)

[**About the data** 7](#_Toc10174184)

[**Models** 11](#_Toc10174185)

[15](#_Toc10174186)

[Results 22](#_Toc10174187)

[27](#_Toc10174188)

[Conclusion 29](#_Toc10174189)

## 

## **Introduction**

Artificial Intelligence, Machine Learning and Deep Learning (AI | ML | DL) are at the heart of digital transformation by enabling organizations to exploit their growing wealth of big data to optimize key business and operational use cases.

* **AI** (Artificial Intelligence) is the theory and development of computer systems able to perform tasks normally requiring human intelligence (e.g. visual perception, speech recognition, **Image recognition**, translation between languages, etc.).
* **ML** (Machine Learning) is a sub-field of AI that provides systems the ability to learn and improve by itself from experience without being explicitly programmed.
* **DL** (Deep Learning) is a type of ML built on a deep hierarchy of layers, with each layer solving different pieces of a complex problem. These layers are interconnected into a “neural network.” A DL framework is SW that accelerates the development and deployment of these models.

**Image Recognition**

Image recognition refers to technologies that identify places, logos, people, objects, buildings, and several other variables in images. Users are sharing vast amounts of data through apps, social networks, and websites. Additionally, mobile phones equipped with cameras are leading to the creation of limitless digital images and videos. The large volume of digital data is being used by companies to deliver better and smarter services to the people accessing it.

The image recognition market is estimated to grow from USD 15.95 Billion in 2016 to USD 38.92 Billion by 2021, at a CAGR of 19.5% between 2016 and 2021 (source <https://www.transparencymarketresearch.com/image-recognition-market.html>). Advancements in machine learning and use of high bandwidth data services is fueling the growth of this technology. Companies in different sectors such as e-commerce, automotive, healthcare, and gaming are rapidly adopting image recognition.

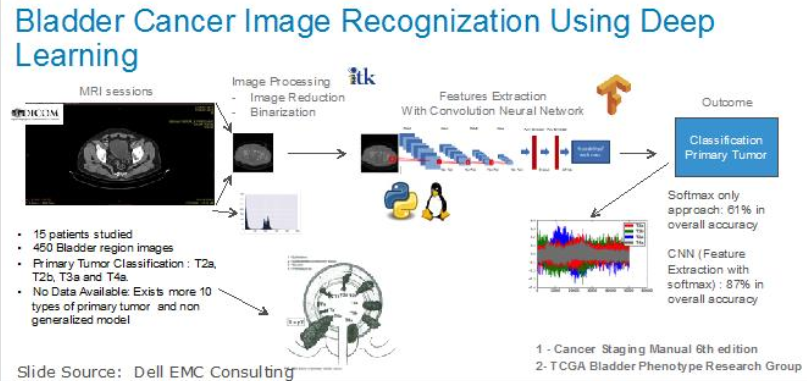


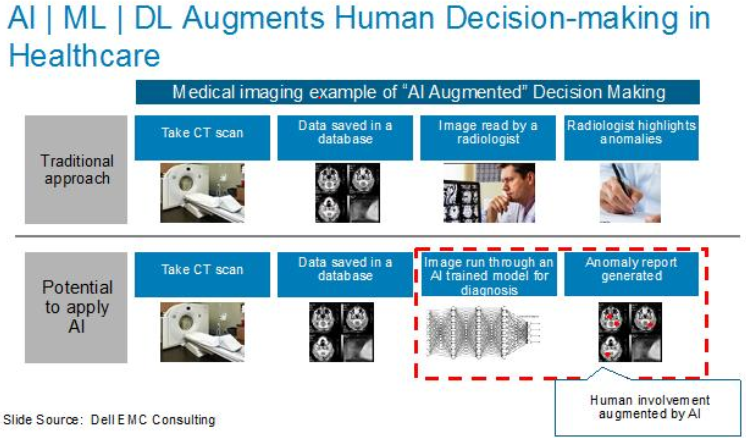
Image recognition is a part of computer vision and a process to identify and detect an object or attribute in a digital video or image. Computer vision is a broader term which includes methods of gathering, processing and analyzing data from the real world. The data is high-dimensional and produces numerical or symbolic information in the form of decisions. Apart from image recognition, computer vision also includes event detection, object recognition, learning, image reconstruction and video tracking. The major steps in image recognition process are gather and organize data, build a predictive model and use it to recognize images.

Practical applications for image recognition are very broad in nature and has numerous benefits for the human life conditions. Here are the few examples that has positive impacts on human beings and for their advancement as technological species in the known universe.

**Use Case #1:  Bladder Cancer Identification Using Medical Image Recognition**

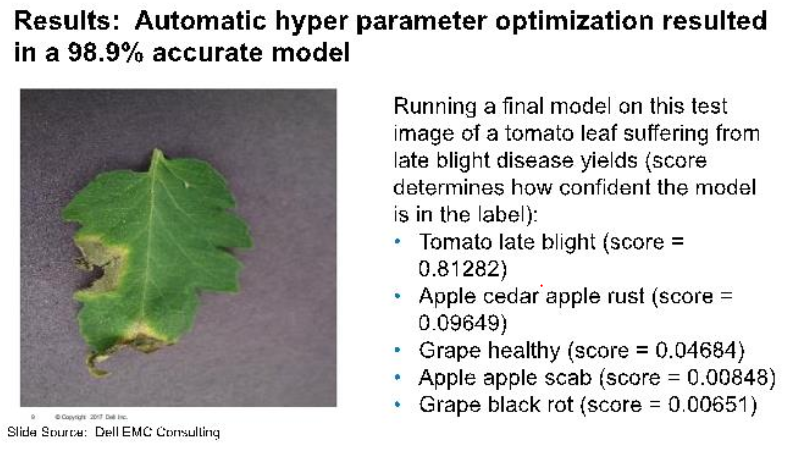
Image recognition of the human body is expected to improve drastically to help doctors with better and more accurate medical diagnostics. ML applied to image recognition of organs, even in the presence of disease, can minimize the possibility of medical errors and speed up disease diagnosis.





**Use Case #2: Crop Disease Identification**

One potential application is the development of mobile disease diagnostics through Image recognition, Deep Learning and crowdsourcing. The results of the engagement were very impressive in scoring different types of crops and their risk to unhealthy situations



Source : <https://infocus.dellemc.com/william_schmarzo/democratizing-artificial-intelligence-deep-learning-machine-learning-with-dell-emc-ready-solutions/>

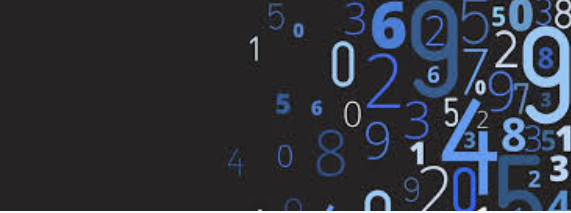
**Classification problem**

Classification between the objects is easy task for humans but it has proved to be a complex problem for machines. The raise of high-capacity computers, the availability of high quality and low-priced video cameras, and the increasing need for automatic video analysis has generated an interest in object classification algorithms. Classification system needs information that contains predefined patterns that compares with detected object to classify in to proper category.

Image classification is an important and challenging task in various application domains, including biomedical imaging, biometry, video surveillance, vehicle navigation, industrial visual inspection, robot navigation, and remote sensing

**Handwritten digit recognition and classification complexity**

Handwritten digit recognition has gained so much popularity from the aspiring beginner of machine learning and deep learning to an expert who has been practicing for years. Developing such a system includes a machine to understand and classify the images of handwritten digits as 10 digits (0–9).



The handwritten digits are not always of the same size, width, orientation and justified to margins as they differ from writing of person to person, so the general problem would be while classifying the digits due to the similarity between digits such as 1 and 7, 5 and 6, 3 and 8, 2 and 5, 2 and 7, etc. This problem is faced more when many people write a single digit with a variety of different handwritings. Lastly, the uniqueness and variety in the handwriting of different individuals also influence the formation and appearance of the digits.

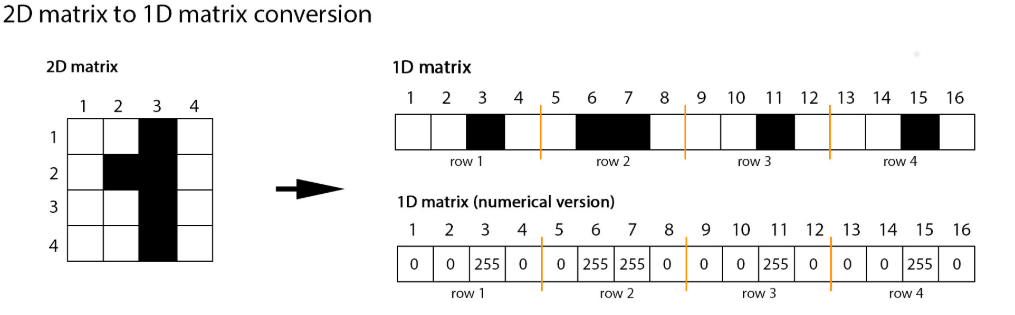
## **Analysis and Models**

### **About the data**

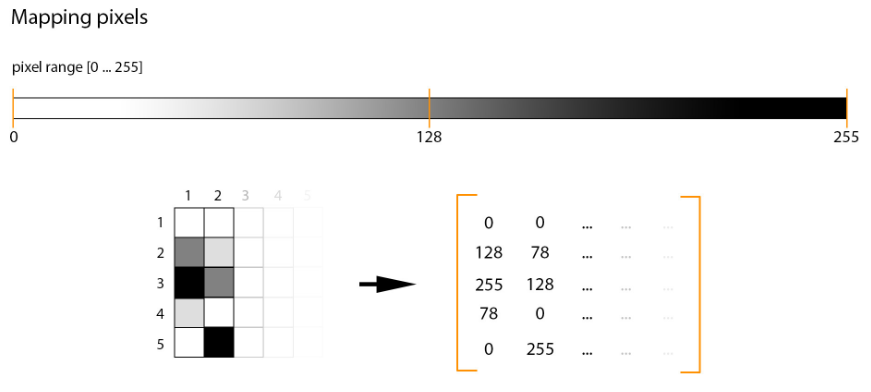
The training data set, (train.csv), has 785 columns. The first column, called "label", is the digit that was drawn by the user. The rest of the columns contain the pixel-values of the associated image. Each pixel column in the training set has a name like pixel-x, where x is an integer between 0 and 783, inclusive. To locate this pixel on the image, suppose that we have decomposed x as x = i \* 28 + j, where i and j are integers between 0 and 27, inclusive. Then pixel-x is located on row i and column j of a 28 x 28 matrix indexing by zero.

Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, inclusive.

see **Figure 1.1** and **Figure 1.2** for a similar arrangement described above using 4 x 4 matrix.



**Figure 1.1**

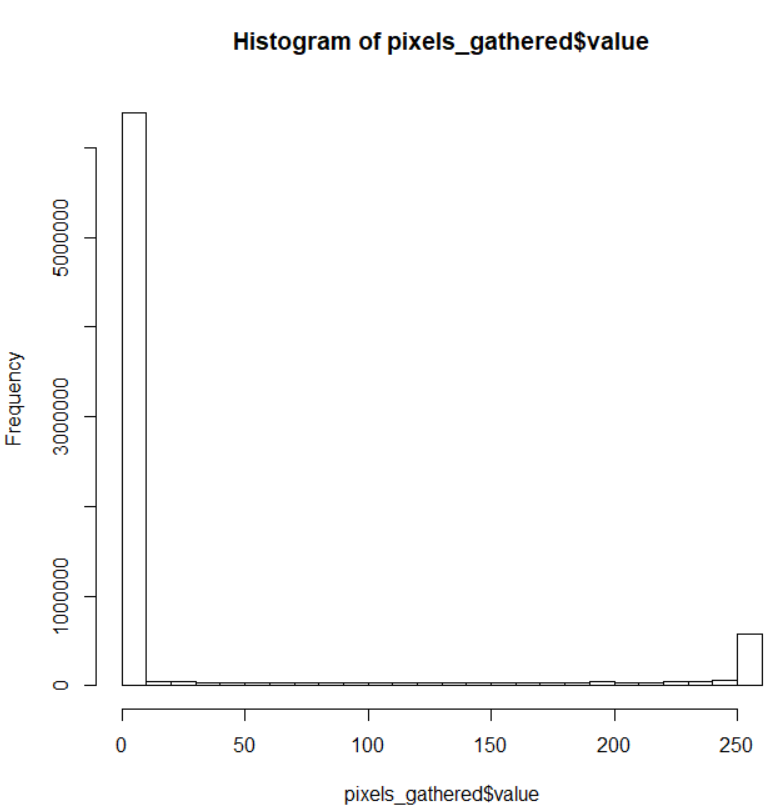


**Figure 1.2 Mapping pixels**

Following points are same in training and testing set along with the set of the images and labels files:

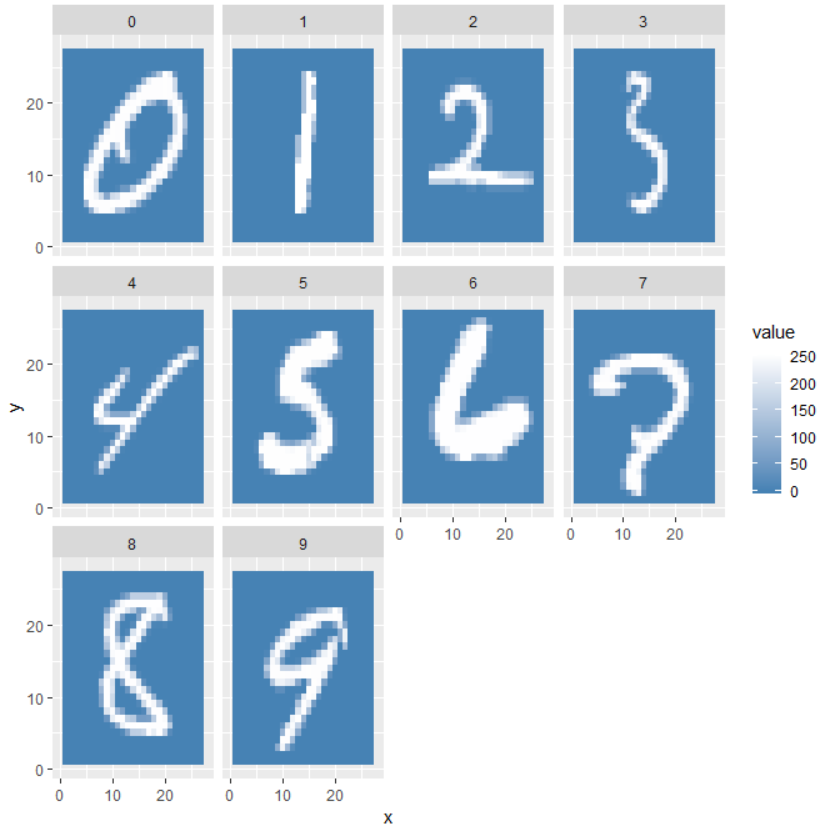
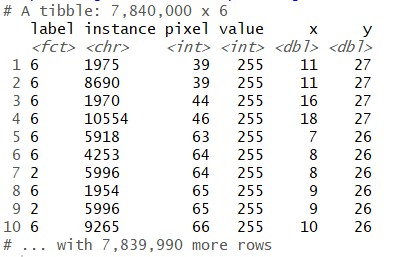
* Background as white (0 value from RGB) and foreground as black (255 value from RGB). (see **Figure 1.1 and Figure 1.2**)
* Pixels are arranged row-wise, ranging from 0 to 255, as from RGB color code. (see **Figure 1.1 and Figure 1.2**)
* Labels of digits classified from 0 to 9.

**Figure 1.3** shows the histogram of the pixel value of some sample images. The distribution is not normal and there are predominant number of attributes with pixel value of 0 and then the next large number of items are with pixel value more than 250



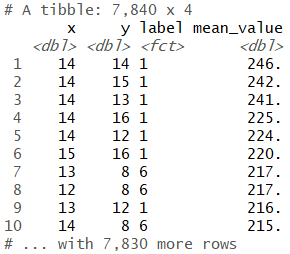
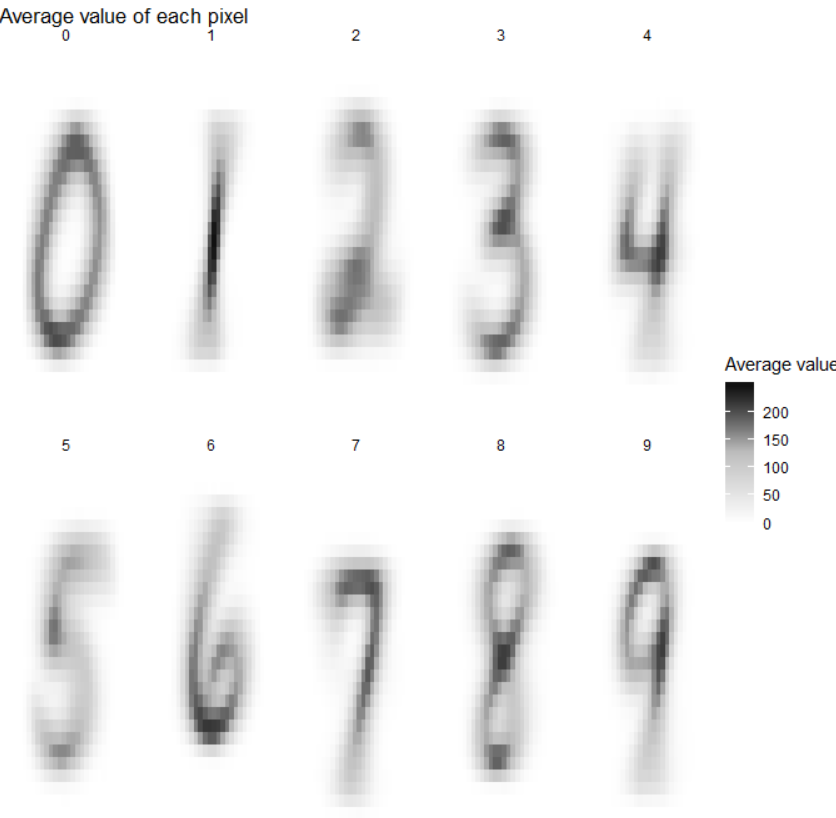
**Figure 1.3 Pixel Value Histogram**

In **Table 1.1** the input pixel value is transformed into 2D and the **Figure 1.4** shows some sample images from the hand-written image digit



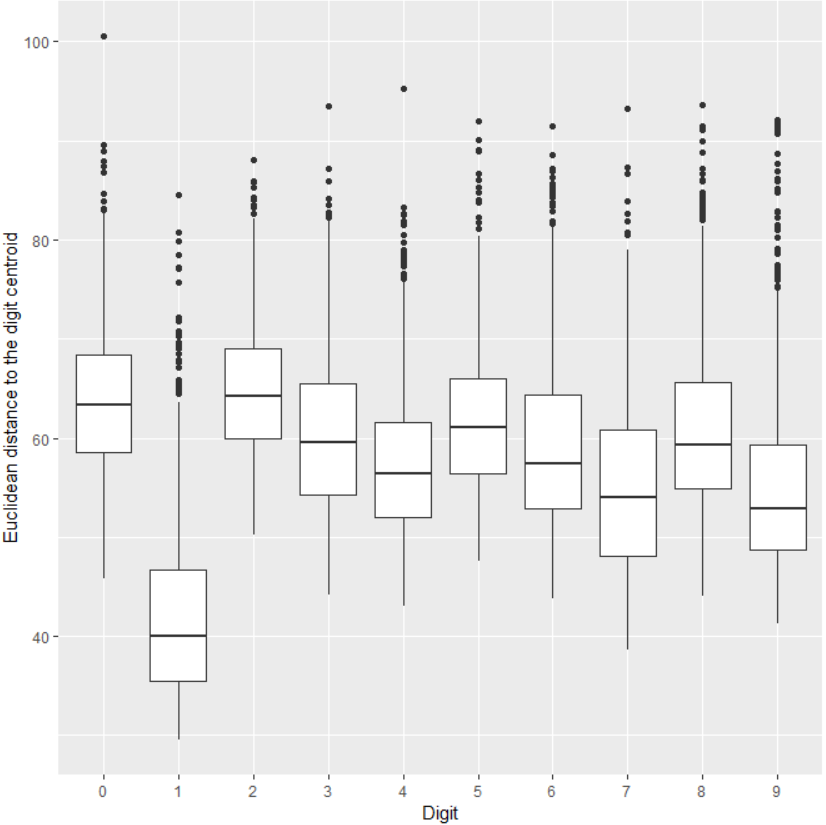
**Table 1.1 Pixel value transformed in 2D Figure 1.4 Sample image from the input dataset**

In **Table 1.2** shows the average pixel value for the given xy coordinates and for a given digit. **Figure 1.5** shows the digit 0 to 9 using average pixel values

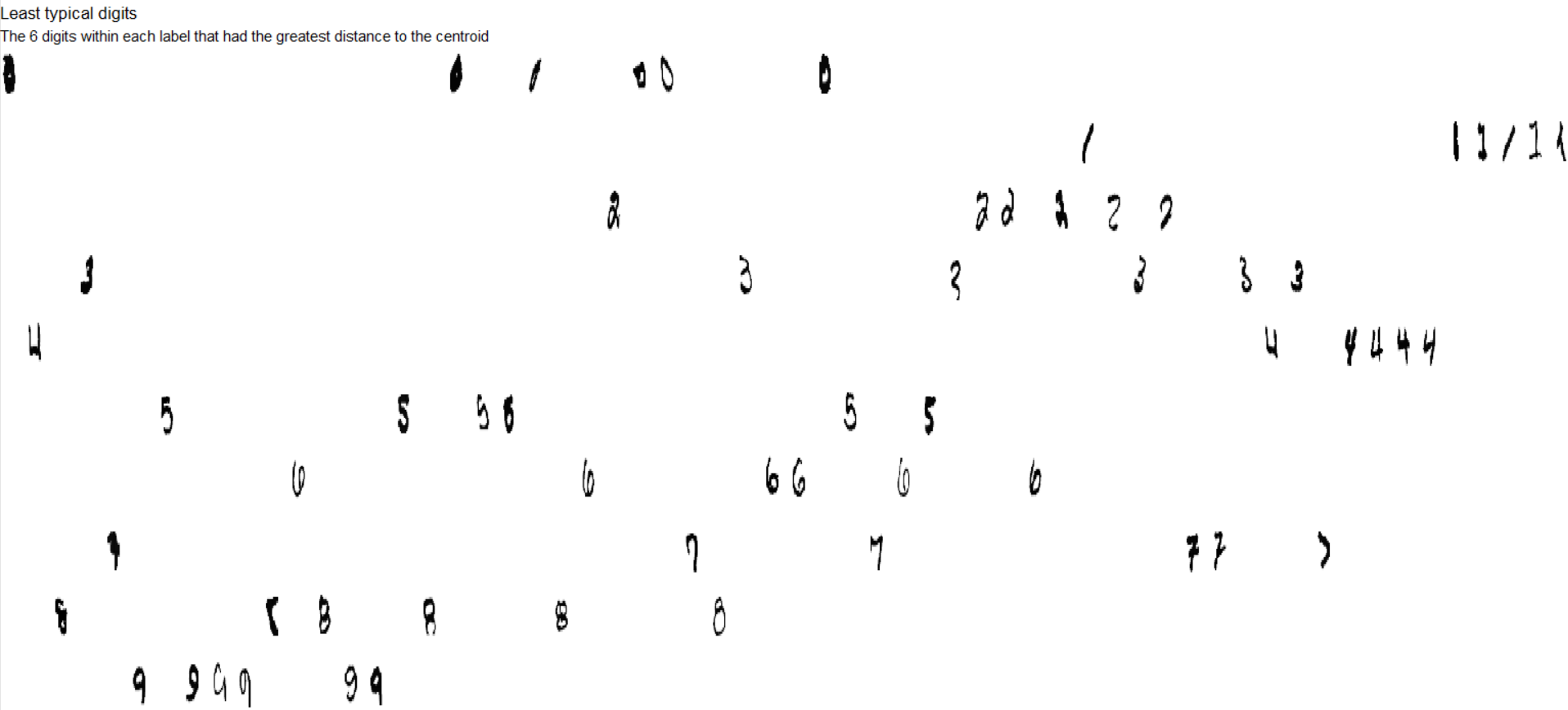
**Table 1.2 Average Pixel value Figure 1.5 Image from the average of individual pixel value**

Variations and outliers for each digit is shown in the below box chart.



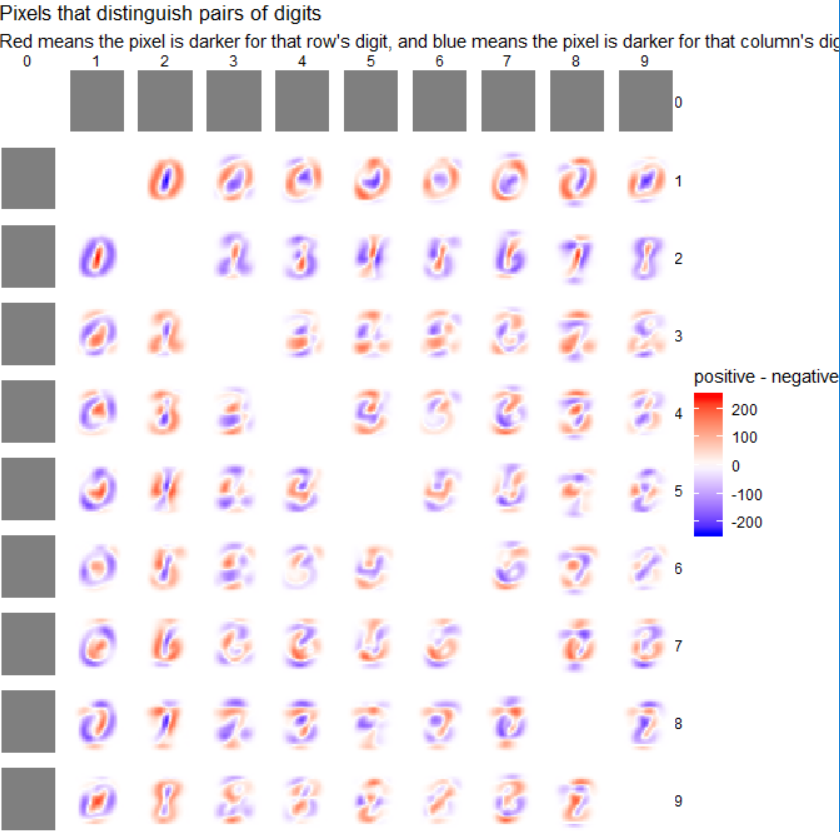
**Figure 1.6 boxplot for images 0 to 9**

Least typical digits are shown in the **Figure 1.7** showing digits with large number of variance from the mean



**Figure 1.7 Least typical digits**

Pairwise comparison of each digit with respect to one another is shown in **Figure 1.8**



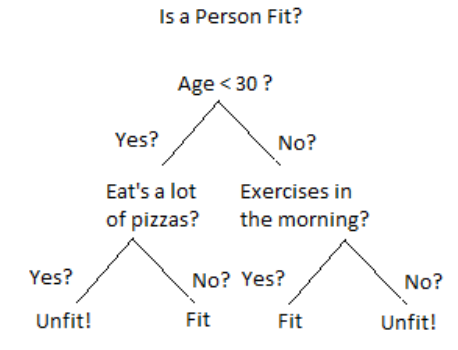
**Figure 1.8 pairwise comparison of digits**

### **Models**

In this exercise, models are developed using Decision Tree Algorithm and Naïve Bayes Algorithm to compare their efficiency and accuracy in classifying the handwritten images into their right bucket from 0 to 9.

#### **Decision Tree Classification**

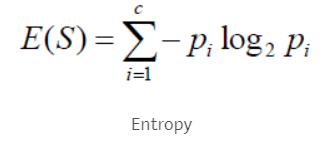
Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The result is a tree with decision nodes and leaf nodes. It follows Iterative Dichotomiser 3(ID3) algorithm structure for determining the split. Decision Tree uses Entropy and Information Gain to construct a decision tree. **Figure 2.1** shows the simple illustration on how decision tree algorithm works



**Figure 2.1 Decision Tree**

#### **Entropy**

Entropy is the degree or amount of uncertainty in the randomness of elements or in other words it is a measure of impurity. Intuitively, it tells us about the predictability of a certain event. Entropy calculates the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has an entropy of one.



#### **Information Gain**

It measures the relative change in entropy with respect to the independent attribute. It tries to estimate the information contained by each attribute. Constructing a decision tree is all about finding the attribute that returns the highest information gain (i.e., the most homogeneous branches). Where Gain (T, X) is the information gain by applying feature X. Entropy(T) is the Entropy of the entire set, while the second term calculates the Entropy after applying the feature X.

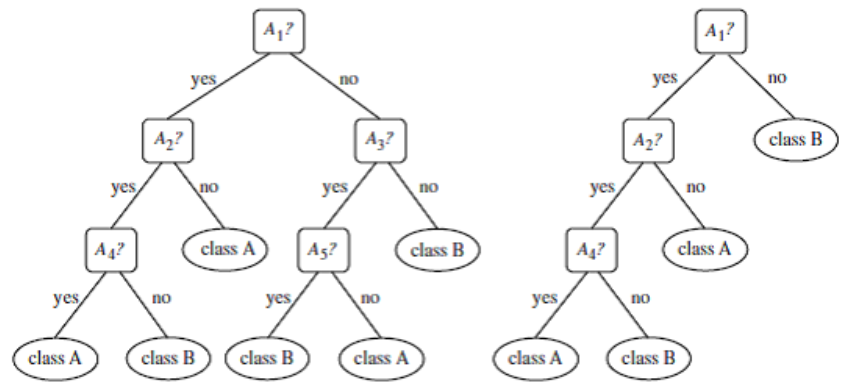


Information Gain ranks attribute for filtering at a given node in the tree. The ranking is based on the highest information gain entropy in each split.

The disadvantage of a Decision Tree Model is overfitting as it tries to fit the model by going deeper in the training set and thereby reducing test accuracy.

#### **Overfitting**

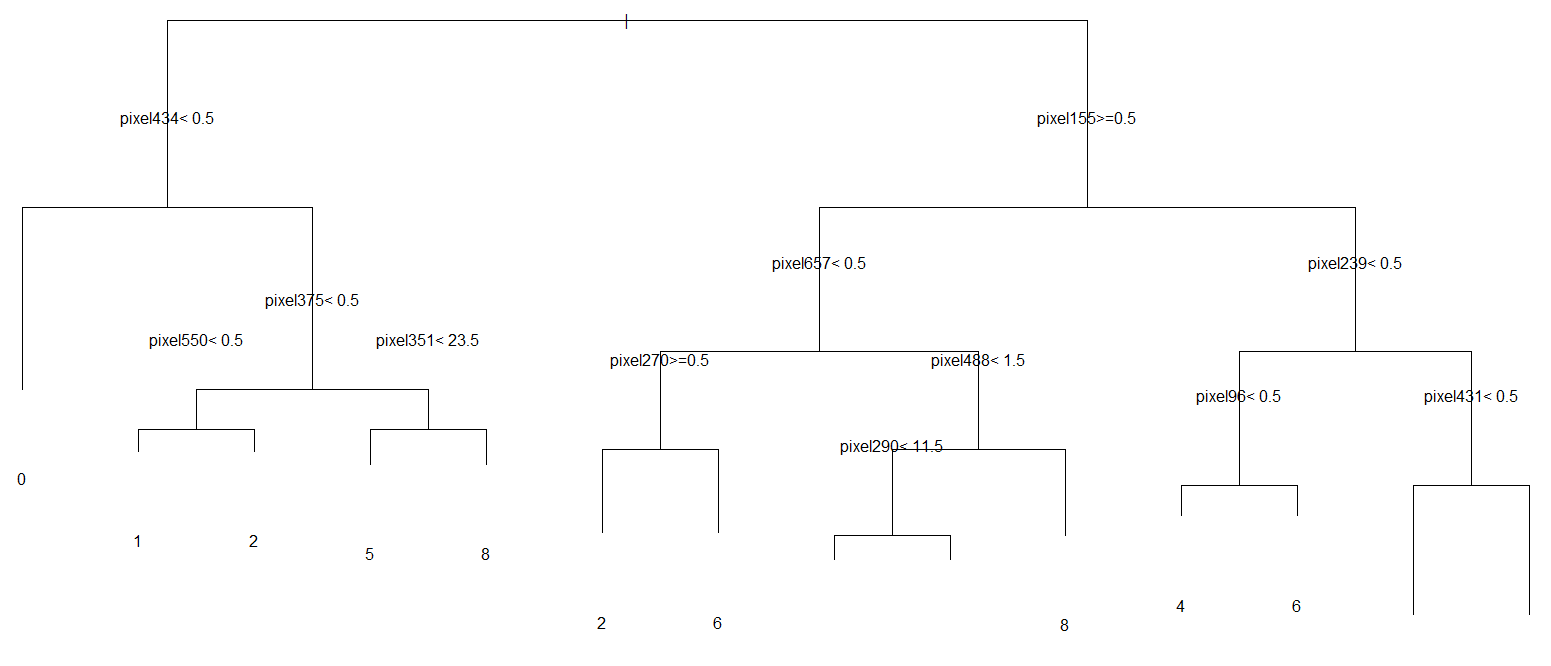
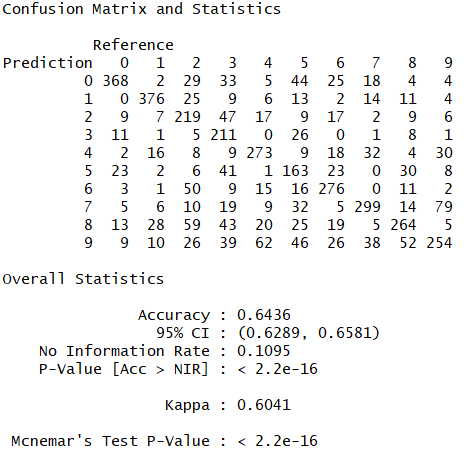
Overfitting in Decision Trees can be minimized by pruning nodes. Tree pruning is illustrated in the **Figure 2.2**



**Figure 2.2 Decision Tree Pruning**

**Model 1: Model from raw pixel values using Decision Tree Algorithm in rpart (unpruned)**

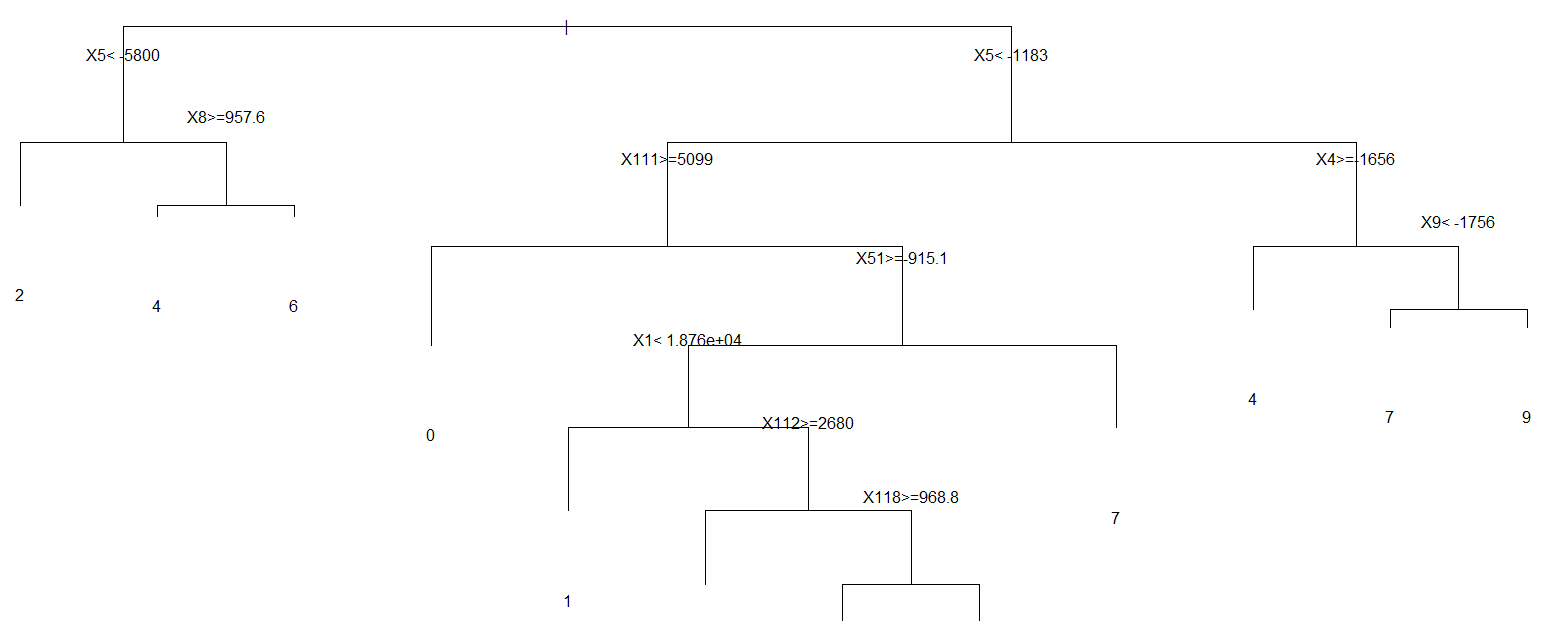
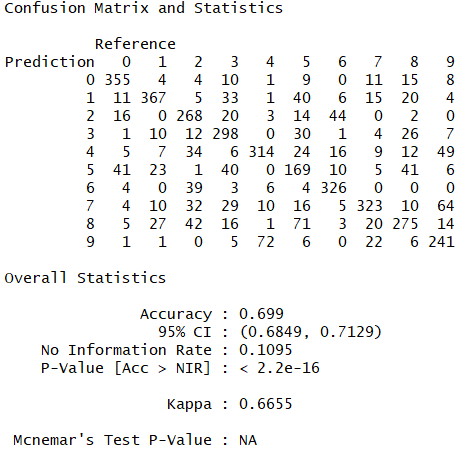
**Figure 2.3 and 2.4** shows the confusion matrix and decision tree that were created as part of the generated model using rpart decision tree algorithm and now transformation applied on the dataset



**Figure 2.3 Consfusion Matrix for Model 1 Figure 2.4 Decision Tree for Model 1**

**Model 2: Model from DCT feature extraction on pixel values 1D using Decision Tree Algorithm in rpart (unpruned)**

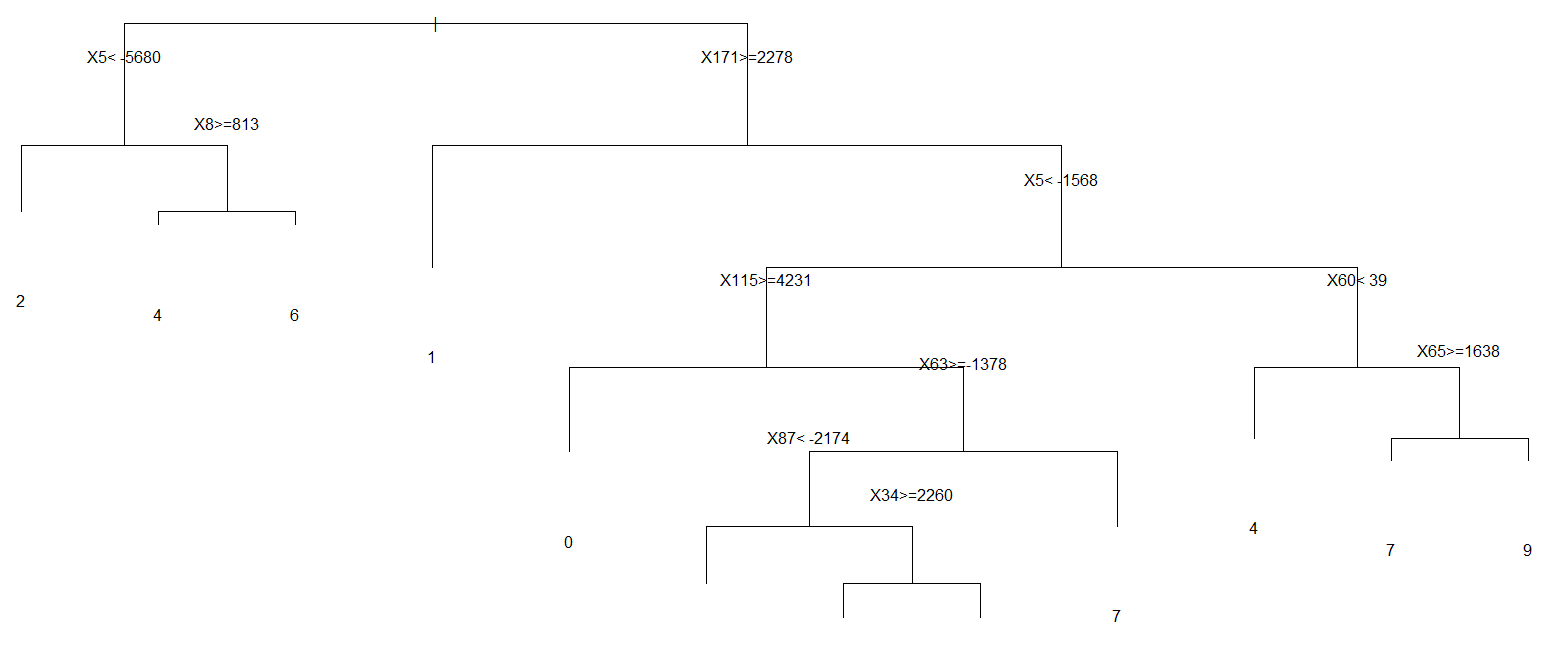
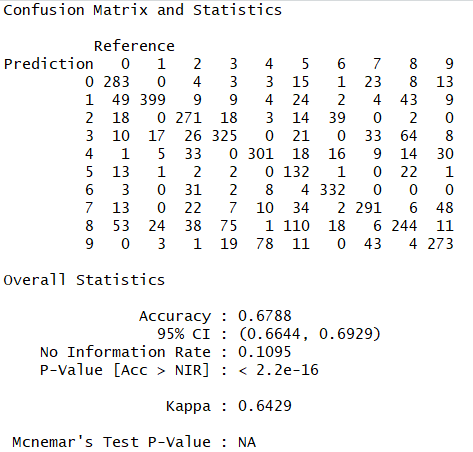
**Figure 2.5 and 2.6** shows the confusion matrix and decision tree that were created as part of the generated model using rpart decision tree algorithm and the pixel value of the dataset are transformed into a discreate cosine values before applying the algorithm



**Figure 2.5 Consfusion Matrix for Model 2 Figure 2.6 Decision Tree for Model 2**

**Model 3: Model from DCT feature extraction on pixel values 2D using Decision Tree Algorithm in rpart (unpruned)**

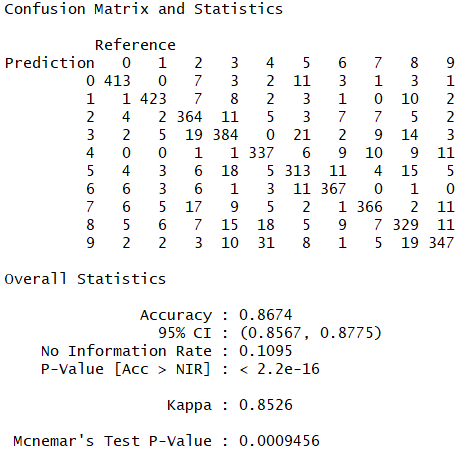
**Figure 2.7 and 2.8** shows the confusion matrix and decision tree that were created as part of the generated model using rpart decision tree algorithm and the pixel value of the dataset are transformed into a 2D discreate cosine value matrix and transposed to an individual attributes before applying the algorithm



**Figure 2.7 Consfusion Matrix for Model 3 Figure 2.8 Decision Tree for Model 3**

**Model 4: Model from raw pixel values using Decision Tree Algorithm in J48 (unpruned)**

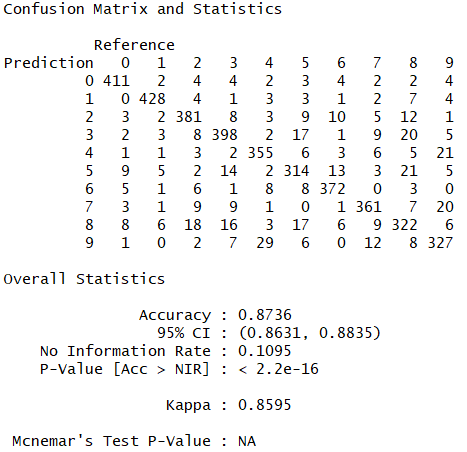
**Figure 2.9** shows the confusion matrix of the unpruned decision tree algorithm. In this model original pixel values are used to generate the decision rules.



**Figure 2.9 Confusion Matrix for Model 4**

**Model 5: Model from DCT feature extraction on pixel values 1D using Decision Tree Algorithm in J48 (unpruned)**

**Figure 2.10** shows the confusion matrix of the unpruned decision tree algorithm. In this model original pixel values are transformed to a discrete cosine values and the decision trees are built using the transformed data.



**Figure 2.10 Confusion Matrix for Model 5**

**Model 6: Model from DCT feature extraction on pixel values 2D using Decision Tree Algorithm in J48 (unpruned)**

**Figure 2.11** shows the confusion matrix of the unpruned decision tree algorithm. In this model original pixel values are transformed to a discrete cosine values in 2D matrix and then transposed from matrix to individual attribute in a row. The decision trees are then built using the transformed data.

## 

**Figure 2.11 Confusion Matrix for Model 6**

#### **Naïve Bayes Classification**

It is a classification technique based on Bayes’ Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a specific feature in a class is unrelated to the presence of any other feature. For example, a fruit may be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as ‘Naive’.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

**Bayes theorem**

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:



* P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
* P(c) is the prior probability of class.
* P(x|c) is the likelihood which is the probability of predictor given class.
* P(x) is the prior probability of predictor.

**Classification based on conditional probability**

To classify whether players will play or not based on weather condition using Naïve Bayes classification approach

Likelihood table Frequency Table are derived by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

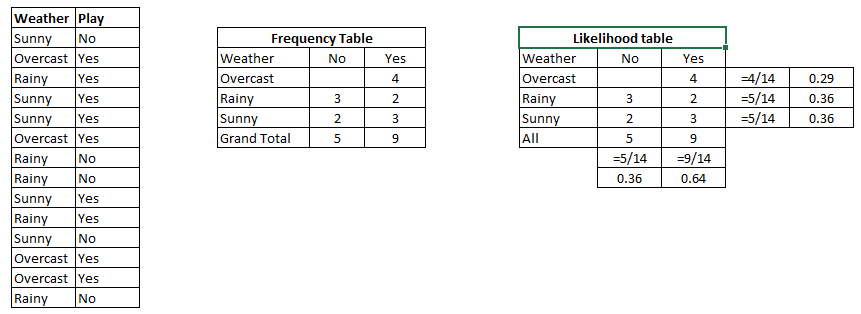
[](https://www.analyticsvidhya.com/wp-content/uploads/2015/08/Bayes_41.png)

Table 2.1

Using Naive Bayesian equation, the posterior probability for each class is calculated. The class with the highest posterior probability is the outcome of prediction.

Say if we want to find out if the Players will play when the weather is sunny?

To solve the above discussed method of posterior probability.

P (Yes | Sunny) = P (Sunny | Yes) \* P(Yes) / P (Sunny)

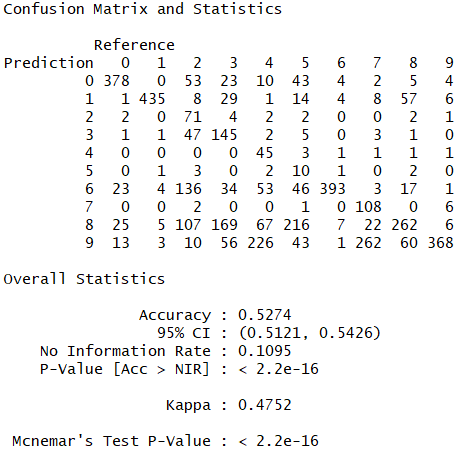
Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P(Yes)= 9/14 = 0.64

Now, P (Yes | Sunny) = 0.33 \* 0.64 / 0.36 = 0.60, which has higher probability.

Naive Bayes uses a similar method to predict the probability of different class based on various attributes.

**Model 1: Model from raw pixel values using Naïve Bayes Algorithm in e1071**

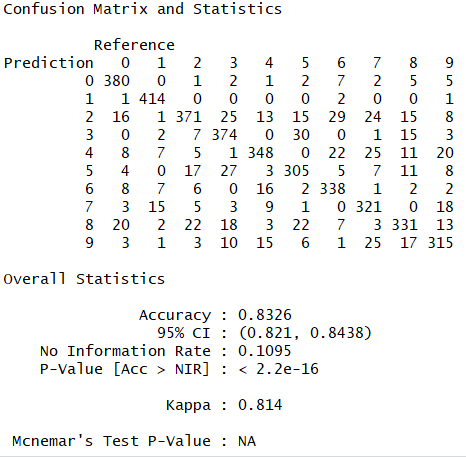
**Figure 2.12** shows the confusion matrix of the Naïve Bayes conditional probability algorithm. In this model original pixel values are used for model generation .



**Figure 2.12 Confusion Matrix for NB Model 1**

**Model 2: Model from DCT feature extraction on pixel values 1D using Naïve Bayes Algorithm in e1071**

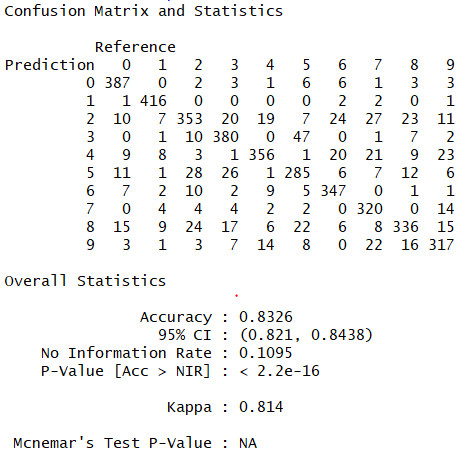
**Figure 2.13** shows the confusion matrix of the Naïve Bayes conditional probability algorithm. In this model original pixel values are transformed using Discrete Cosine Transformation (DCT) before applying Naïve Bayes Algorithm.



**Figure 2.13 Confusion Matrix for NB Model 2**

**Model 3: Model from DCT feature extraction on pixel values 2D using Naïve Bayes Algorithm in e1071**

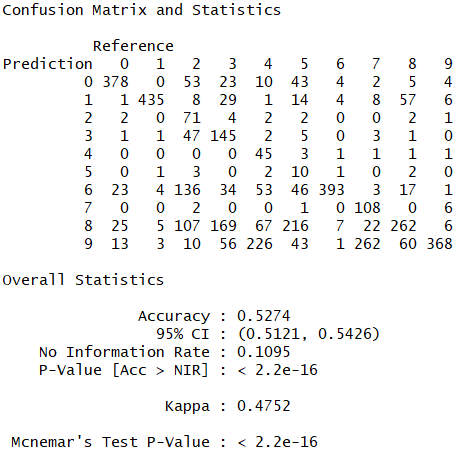
**Figure 2.14** shows the confusion matrix of the Naïve Bayes conditional probability algorithm. In this model original pixel values are transformed using Discrete Cosine Transformation (DCT) from a 2d pixel matrix before applying Naïve Bayes Algorithm.



**Figure 2.14 Confusion Matrix for NB Model 3**

**Model 4: Model from raw pixel values using Naïve Bayes Algorithm in naivebayes package**

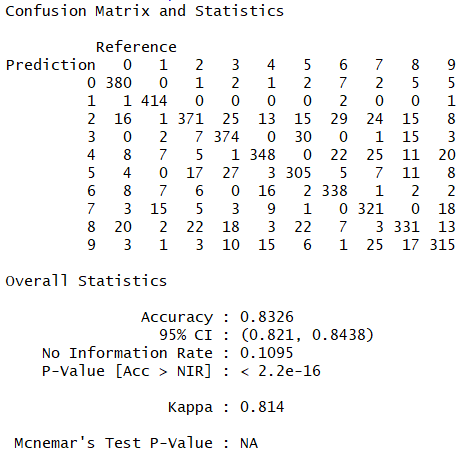
**Figure 2.15** shows the confusion matrix of the Naïve Bayes conditional probability algorithm. In this model original pixel values are used for model generation.



**Figure 2.15 Confusion Matrix for NB Model 4**

**Model 5: Model from DCT feature extraction on pixel values 1D using Naïve Bayes Algorithm in naivebayes package**

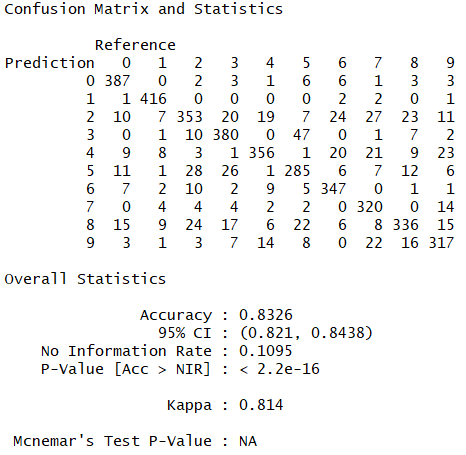
**Figure 2.16** shows the confusion matrix of the Naïve Bayes conditional probability algorithm. In this model original pixel values are transformed using Discrete Cosine Transformation (DCT) before applying Naïve Bayes Algorithm.



**Figure 2.16 Confusion Matrix for NB Model 5**

**Model 6: Model from DCT feature extraction on pixel values 2D using Naïve Bayes Algorithm in e1071**

**Figure 2.17** shows the confusion matrix of the Naïve Bayes conditional probability algorithm. In this model original pixel values are transformed using Discrete Cosine Transformation (DCT) from a 2d pixel matrix before applying Naïve Bayes Algorithm.



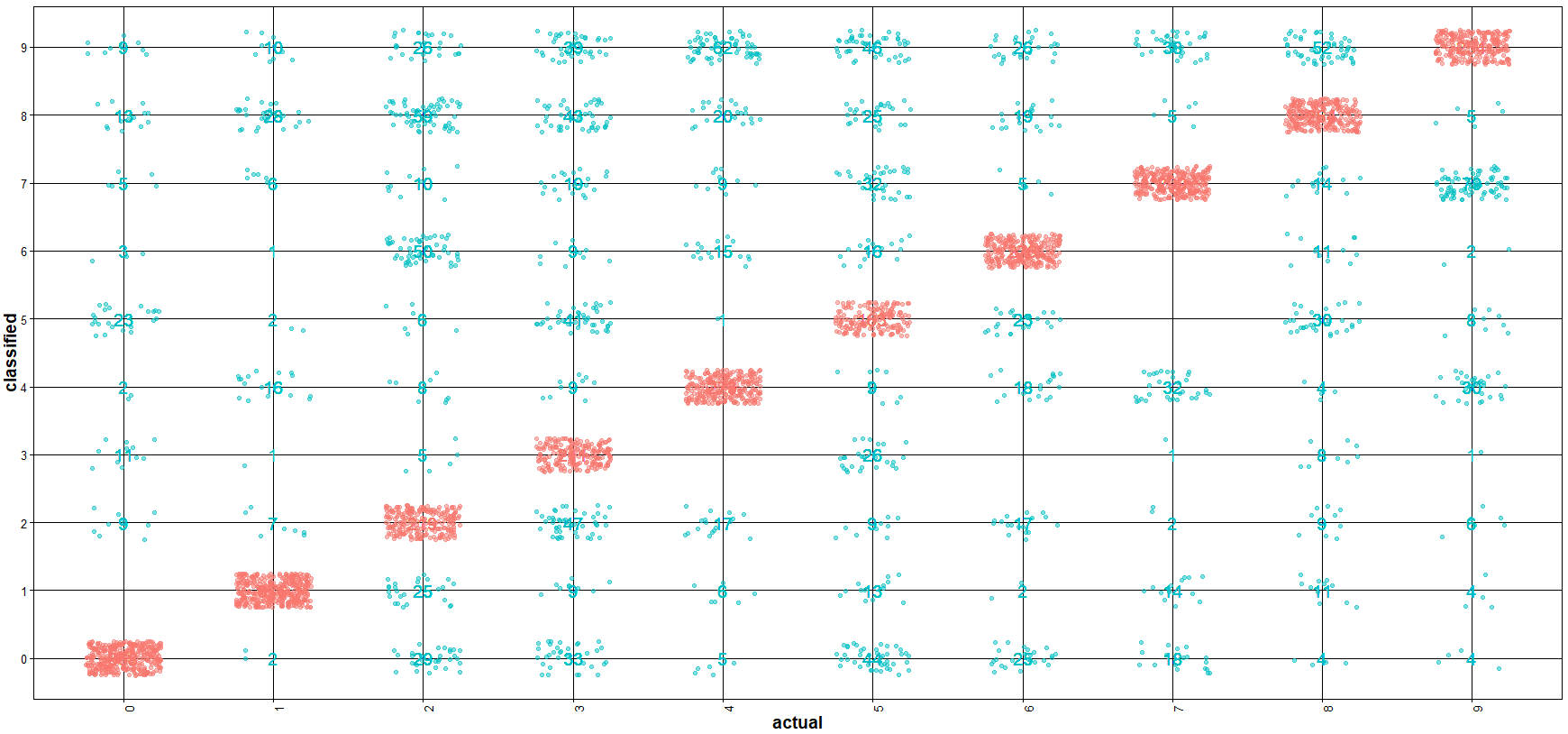
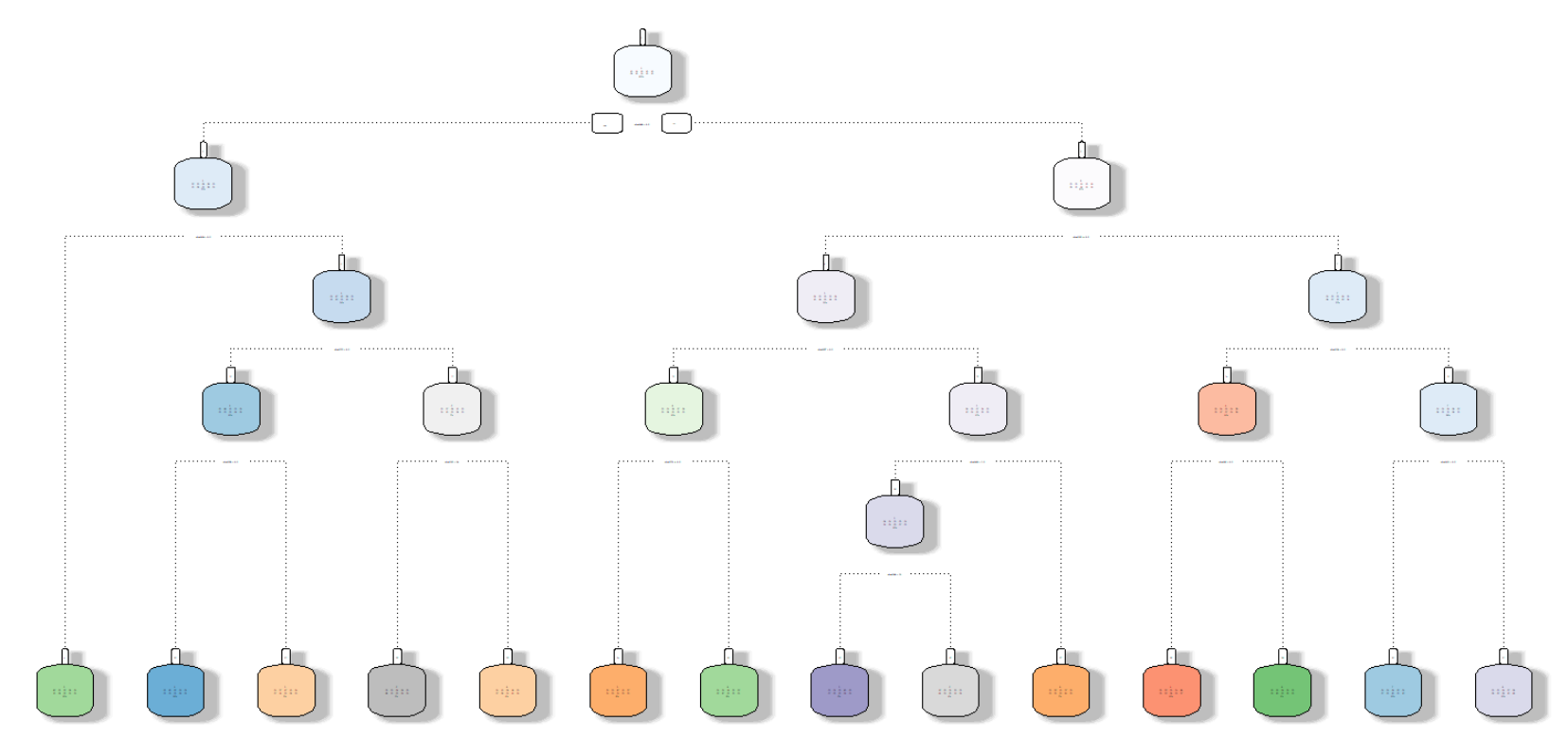
**Figure 2.17 Confusion Matrix for NB Model 6**

## **Results**

#### **Decision Tree Classification**

**Model 1: Model from raw pixel values using Decision Tree Algorithm in rpart (unpruned)**

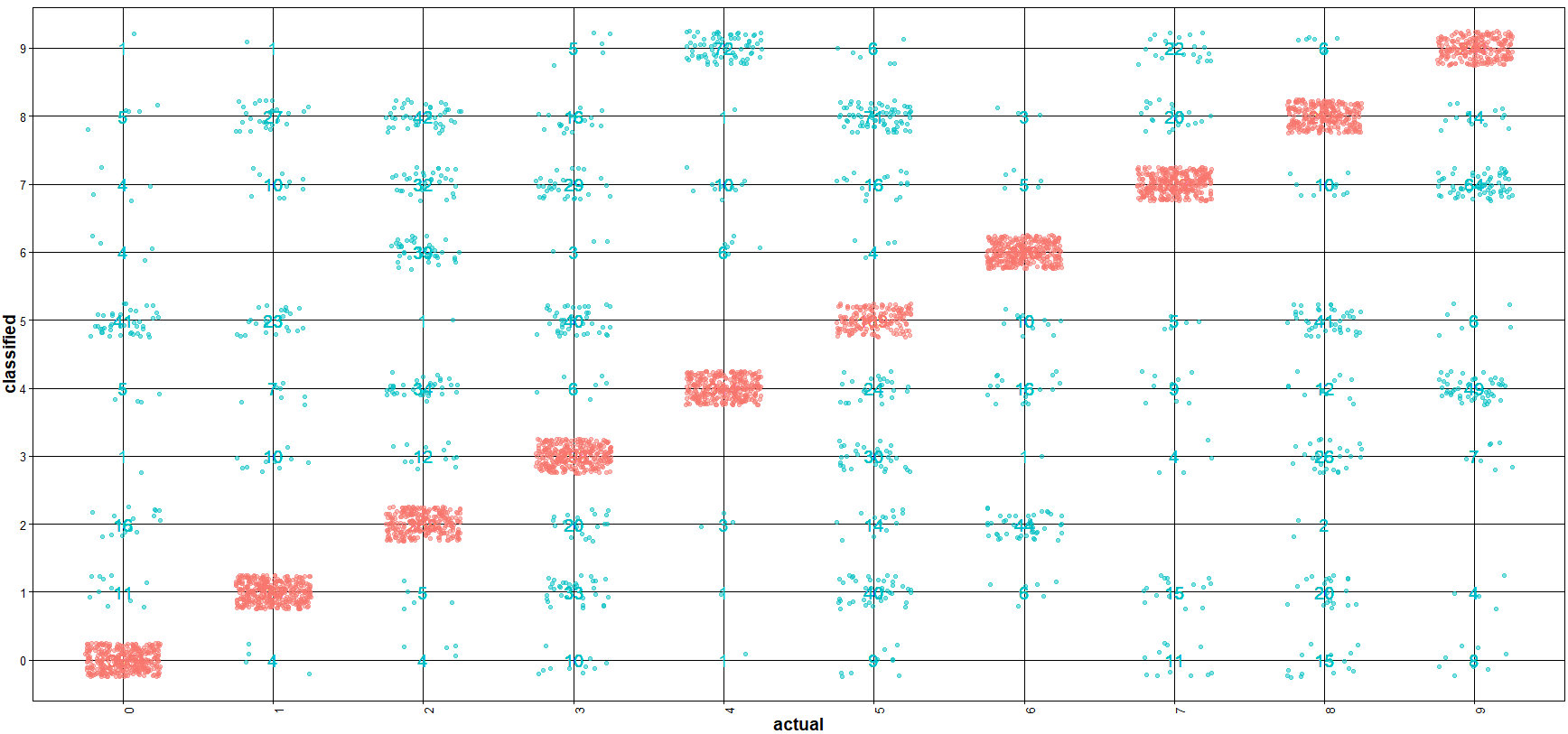
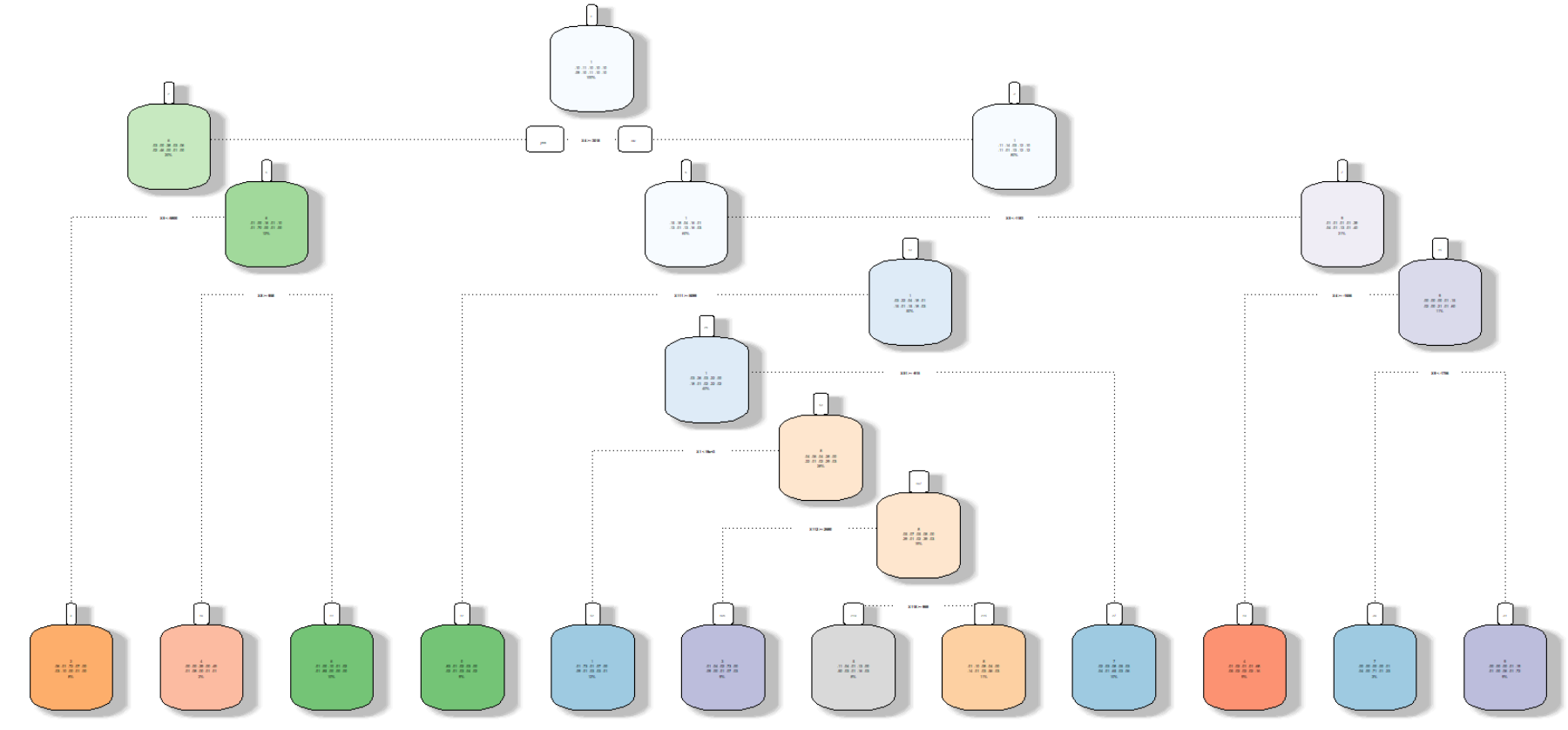
The accuracy of this model is about 54%. The cost required to generate and test this model is low and easy. **Figure 3.1** shows the decision tree rules and the distribution of prediction error for each digit image.



**Figure 3.1 Decision Tree and Accuracy of the prediction in Model 1**

**Model 2: Model from DCT feature extraction on pixel values 1D using Decision Tree Algorithm in rpart (unpruned)**

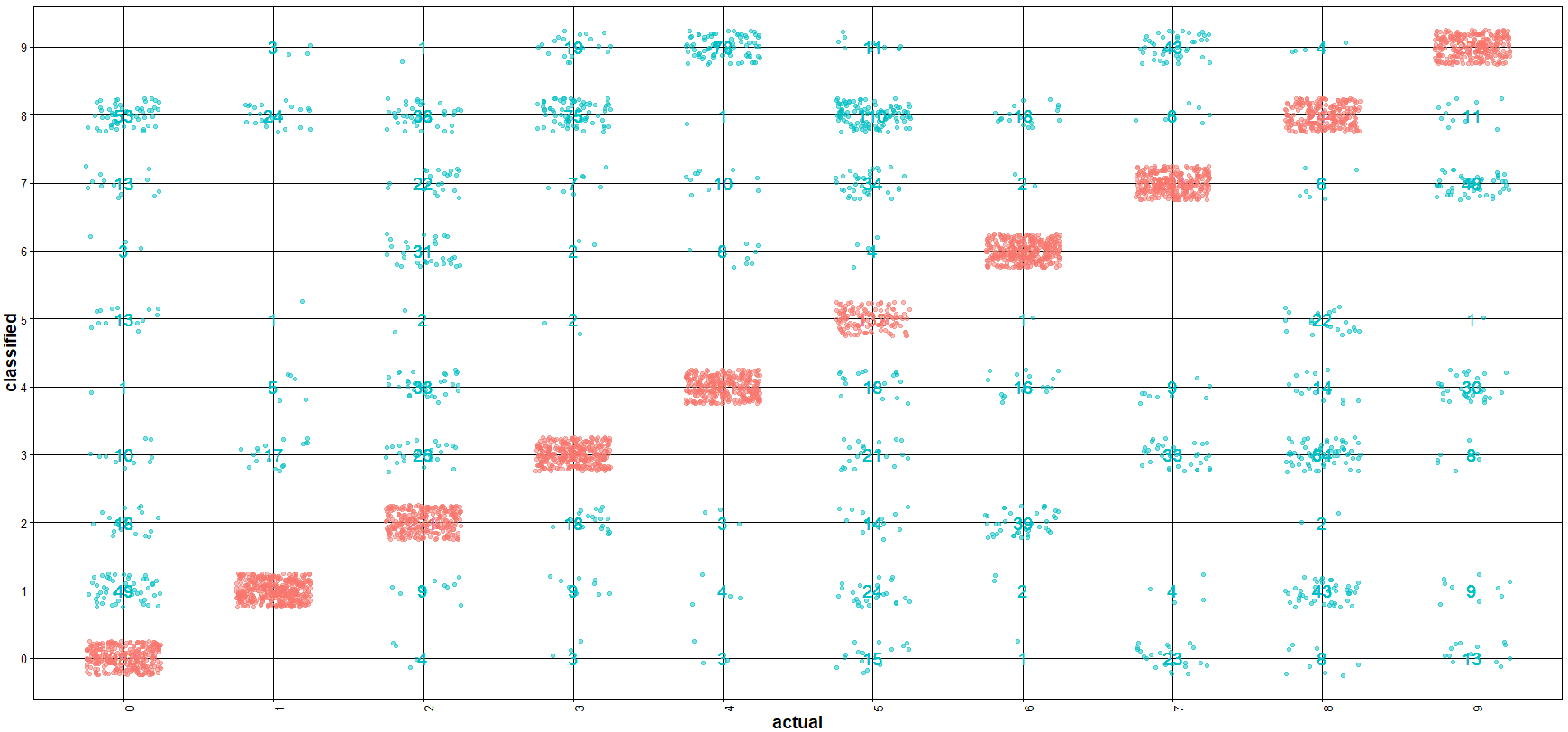
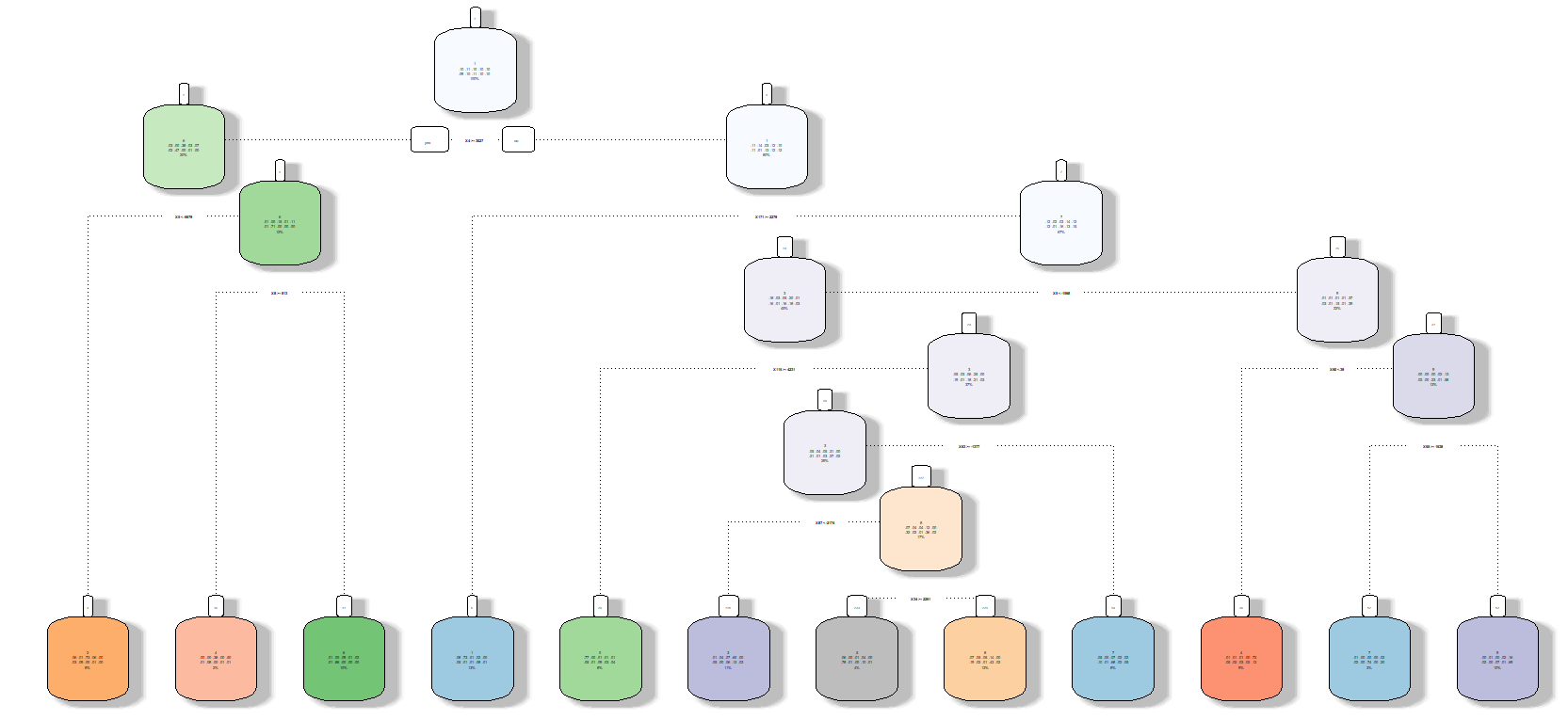
The accuracy of this model is about 83%. DCT feature generation is time consuming but the accuracy of the model increased from 54 to 83% **Figure 3.2** shows the decision tree rules and the distribution of prediction error for each digit image.



**Figure 3.2 Decision Tree and Accuracy of the prediction in Model 2**

**Model 3: Model from DCT feature extraction on pixel values 2D using Decision Tree Algorithm in rpart (unpruned)**

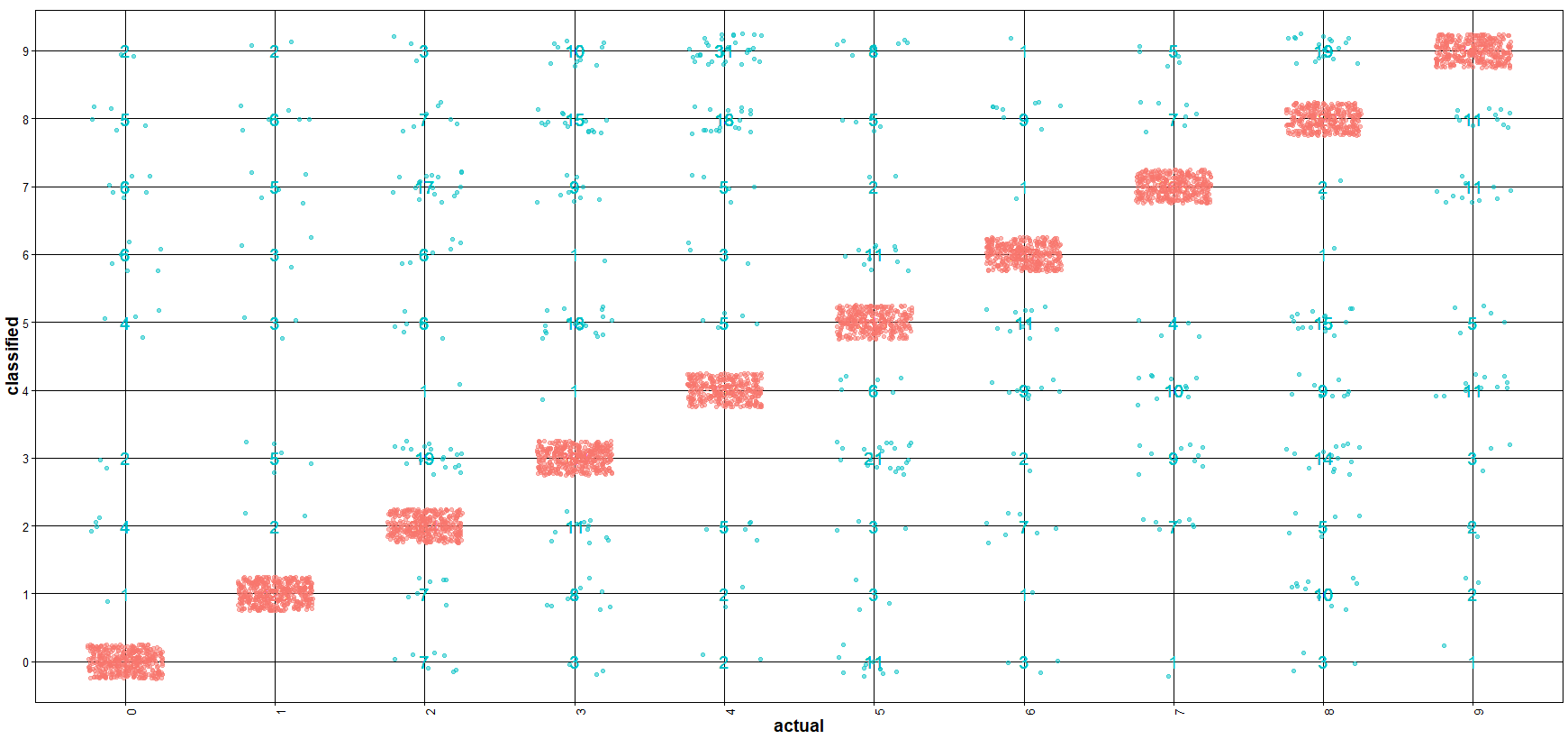
The accuracy of this model is about 82.7% which is same as the previous model. DCT feature generation from 2D matrix took same time as the feature generation from the previous model (features are extracted from a vector of pixel values in 1D). The accuracy of the model increased from 54 to 83% from the model which used raw pixel values. **Figure 3.3** shows the decision tree rules and the distribution of prediction error for each digit image.



**Figure 3.3 Decision Tree and Accuracy of the prediction in Model 3**

**Model 4: Model from raw pixel values using Decision Tree Algorithm in J48 (unpruned)**

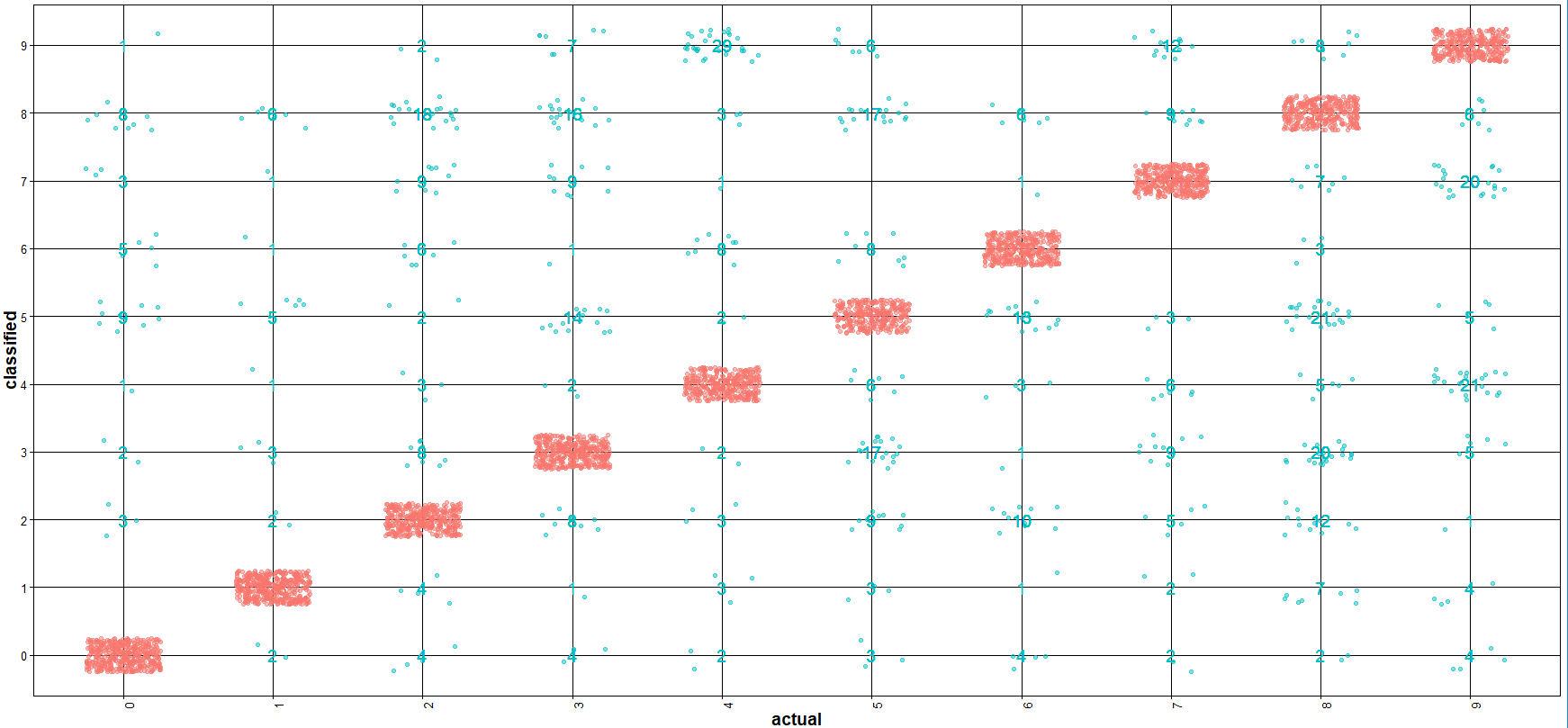
The accuracy of this model is about 86%. The cost required to generate is high and resource intensive. **Figure 3.4** shows the distribution of prediction error for each digit image.



**Figure 3.4 Accuracy of the prediction in Model 4**

**Model 5: Model from DCT feature extraction on pixel values 1D using Decision Tree Algorithm in J48 (unpruned)**

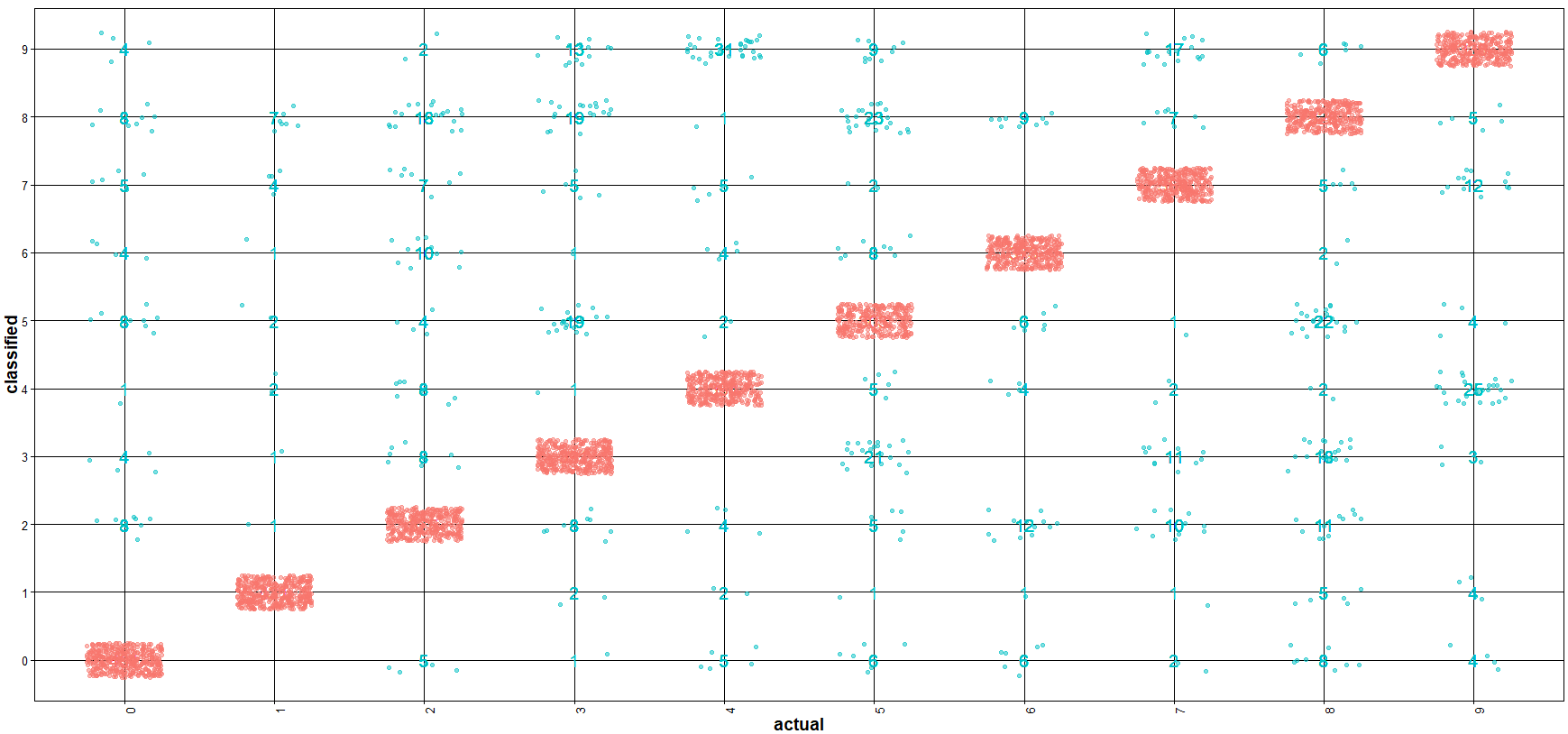
The accuracy of this model is about 86% which is same as in the previous model. DCT feature generation didn’t had much difference in improving the accuracy of the model **Figure 3.5** shows the distribution of prediction error for each digit image.



**Figure 3.5 Accuracy of the prediction in Model 5**

**Model 6: Model from DCT feature extraction on pixel values 2D using Decision Tree Algorithm in J48 (unpruned)**

The accuracy of this model is about 85% which is same as previous 2 models. DCT feature generation both from 1D vector of pixel values and 2D matrix pixel values didn’t had much difference in improving the accuracy of the model **Figure 3.6** shows the distribution of prediction error for each digit image.

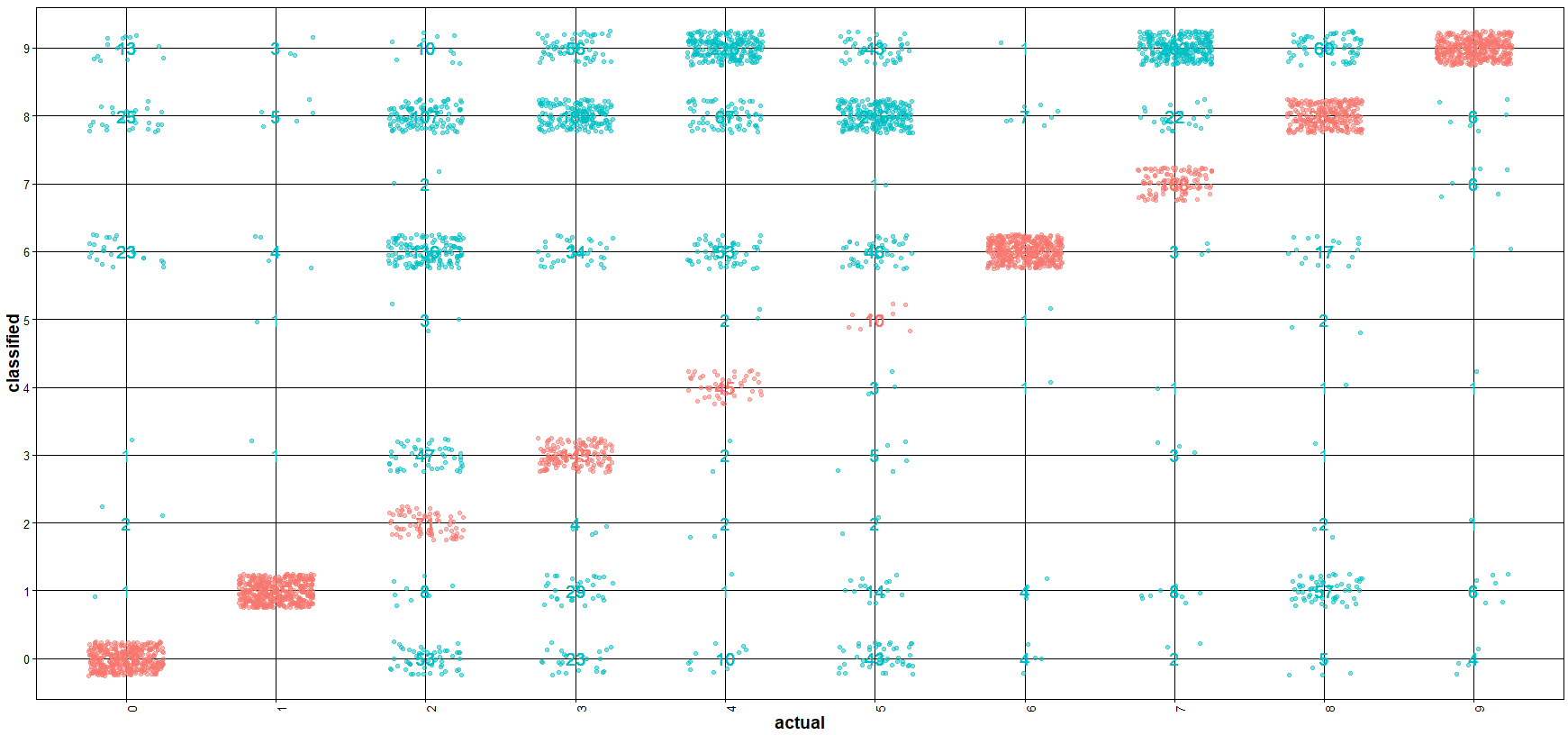


**Figure 3.6 Accuracy of the prediction in Model 6**

#### **Naïve Bayes Classification**

**Model 1: Model from raw pixel values using Naïve Bayes Algorithm in e1071**

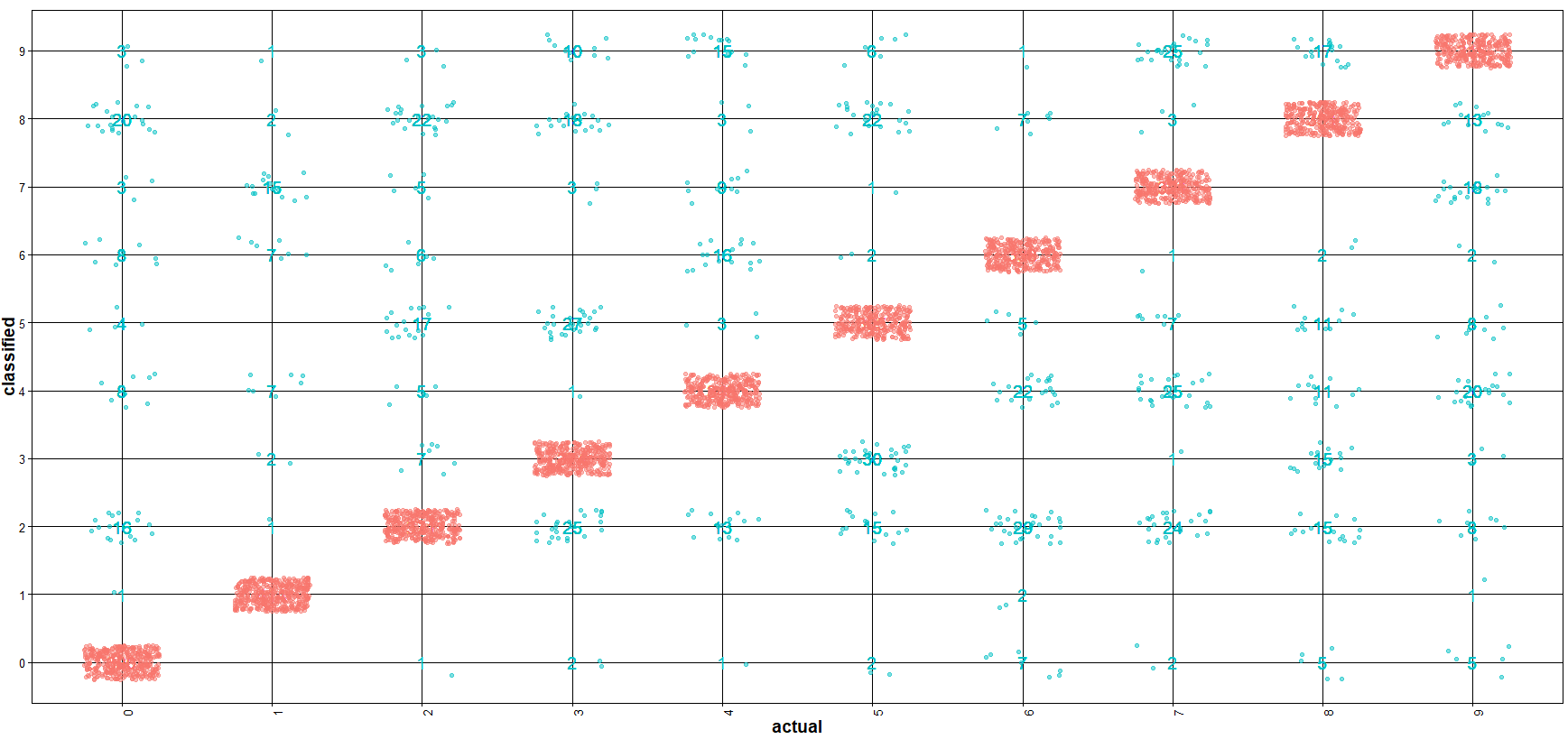
The accuracy of this model is about 52%. The cost required to generate and test this model is low and takes less time. **Figure 3.7** shows the distribution of prediction error for each digit image.



**Figure 3.7 Accuracy of the prediction in Model 1**

**Model 2: Model from DCT feature extraction on pixel values 1D using Naïve Bayes Algorithm in e1071**

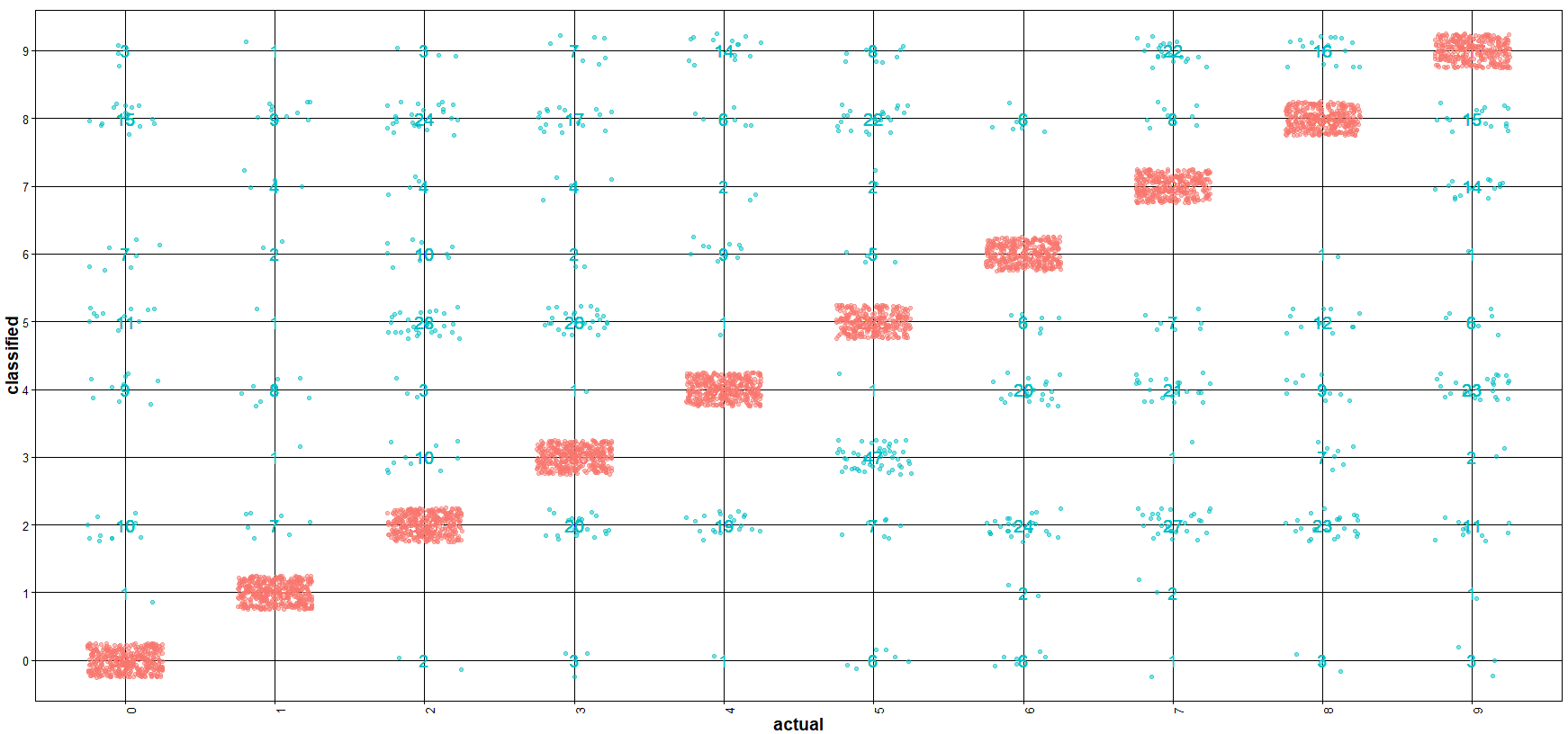
The accuracy of this model is about 82%. DCT feature generation is time consuming but the accuracy of the model increased from 52 to 82% **Figure 3.7** shows the distribution of prediction error for each digit image.



**Figure 3.8 Accuracy of the prediction in Model 2**

**Model 3: Model from DCT feature extraction on pixel values 2D using Naïve Bayes Algorithm in e1071**

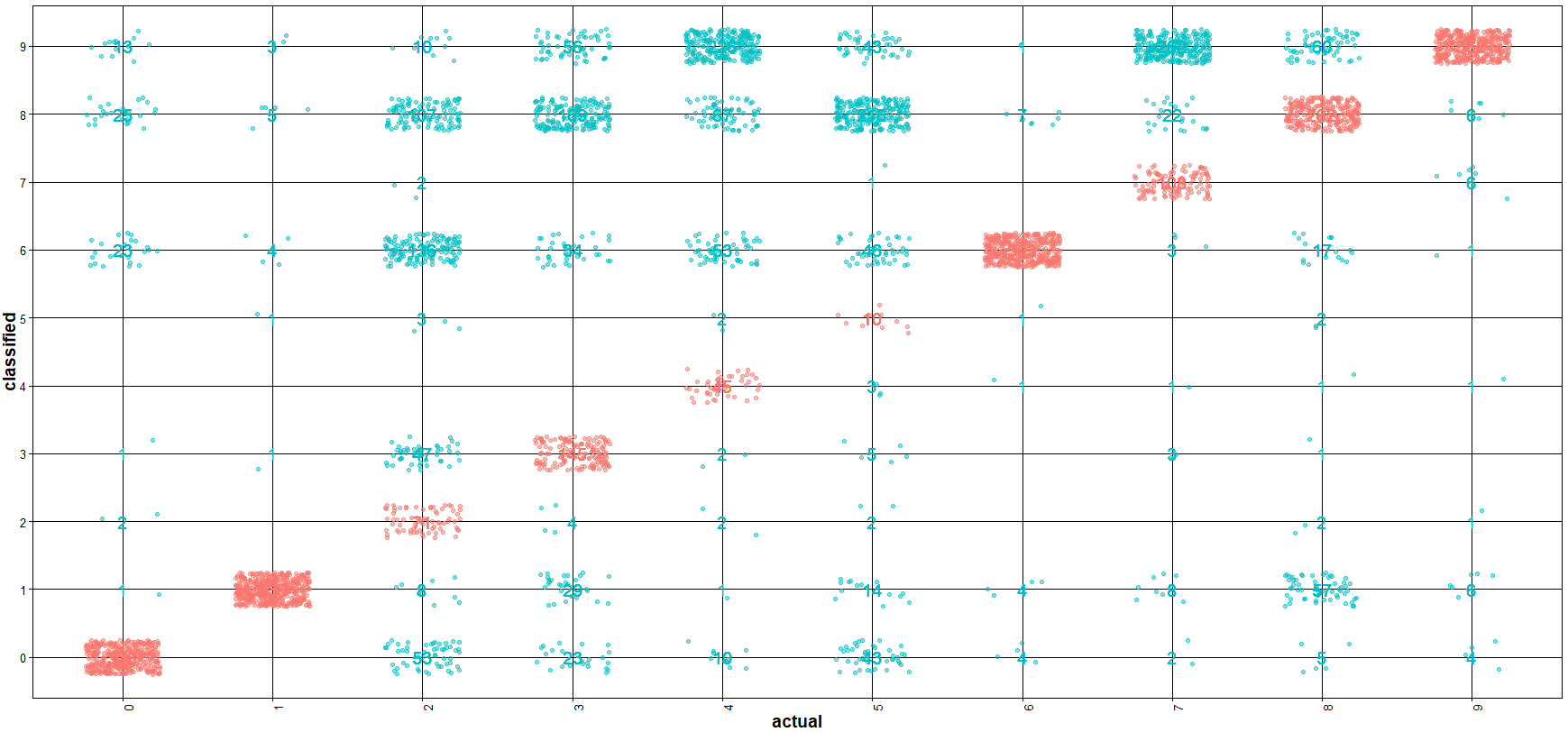
The accuracy of this model is about 81% which is same as the previous model. DCT feature generation from 2D matrix took same time as the feature generation from the previous model (features are extracted from a vector of pixel values in 1D). The accuracy of the model increased from 52 to 81% from the model which used raw pixel values. **Figure 3.9** shows the distribution of prediction error for each digit image.



**Figure 3.9 Accuracy of the prediction in Model 3**

**Model 4: Model from raw pixel values using Naïve Bayes Algorithm in naivebayes package**

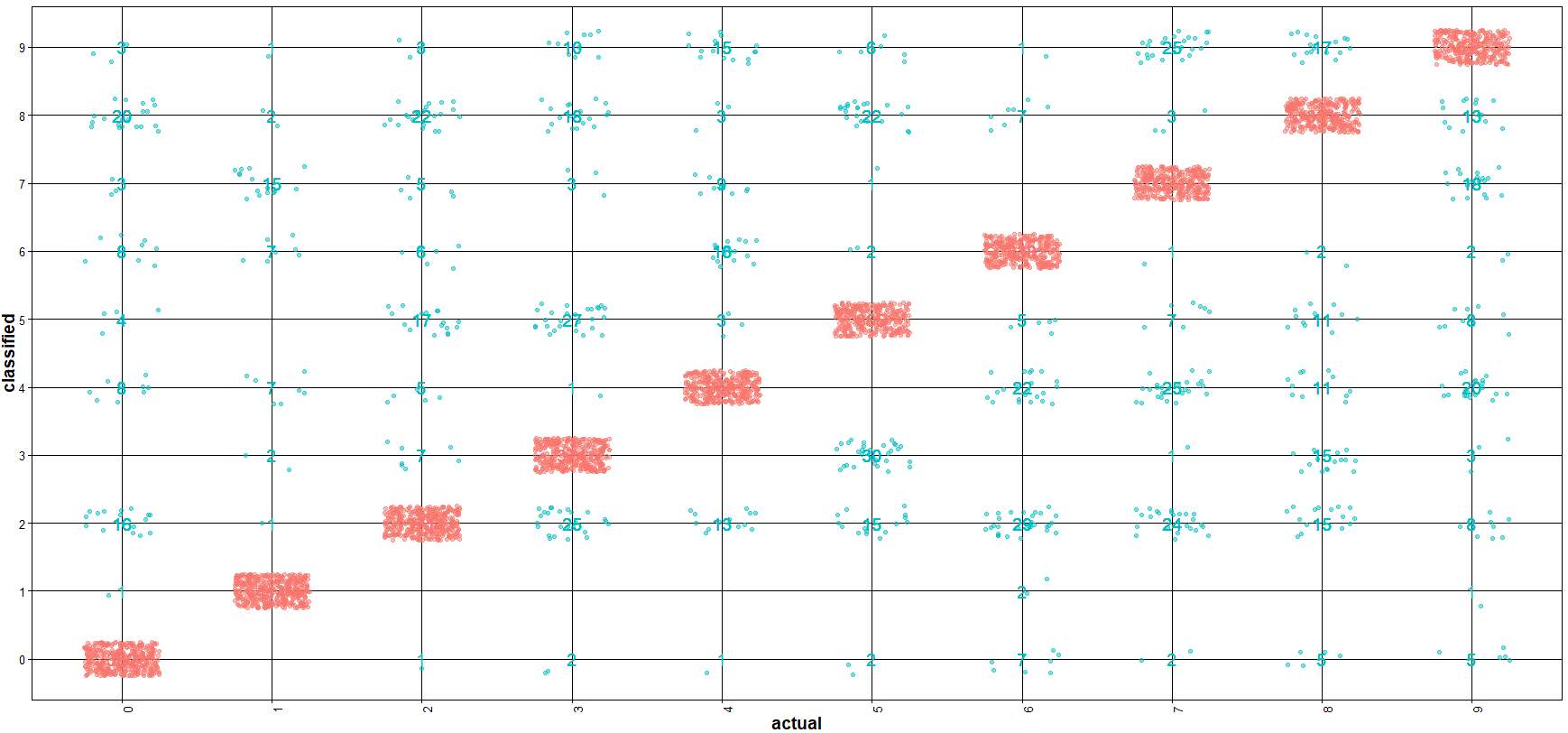
The accuracy of this model is about 52%. The cost required to generate and test this model is low and takes less time. **Figure 3.10** shows the distribution of prediction error for each digit image.



**Figure 3.10 Accuracy of the prediction in Model 4**

**Model 5: Model from DCT feature extraction on pixel values 1D using Naïve Bayes Algorithm in naivebayes package**

The accuracy of this model is about 82%. DCT feature generation is time consuming but the accuracy of the model increased from 52 to 82% **Figure 3.11** shows the distribution of prediction error for each digit image.



**Figure 3.11 Accuracy of the prediction in Model 5**

**Model 6: Model from DCT feature extraction on pixel values 2D using Naïve Bayes Algorithm in naivebayes package**

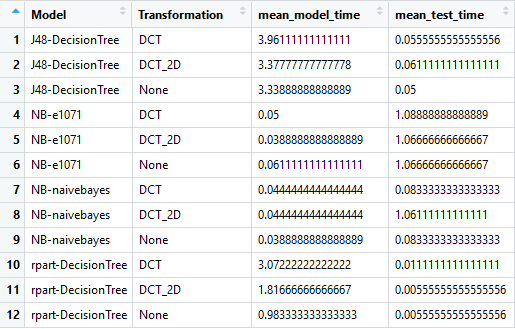
The accuracy of this model is about 81% which is same as the previous model. DCT feature generation from 2D matrix took same time as the feature generation from the previous model (features are extracted from a vector of pixel values in 1D). The accuracy of the model increased from 52 to 81% from the model which used raw pixel values. **Figure 3.12** shows the distribution of prediction error for each digit image.

## 

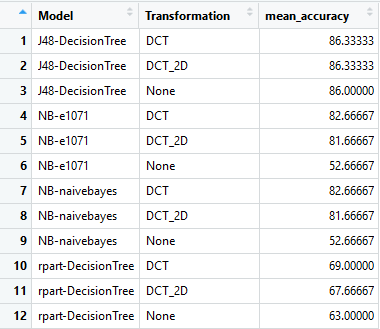
**Figure 3.12 Accuracy of the prediction in Model 6**

#### **Algorithm performance comparison**

After performing 3 cross validation, the average time took to build the model and perform the test prediction is shown in **Table 3.1.** Also, **Table 3.2** shows the average accuracy of the model for all the 3-cross validation



**Table 3.1 Model build time comparison**

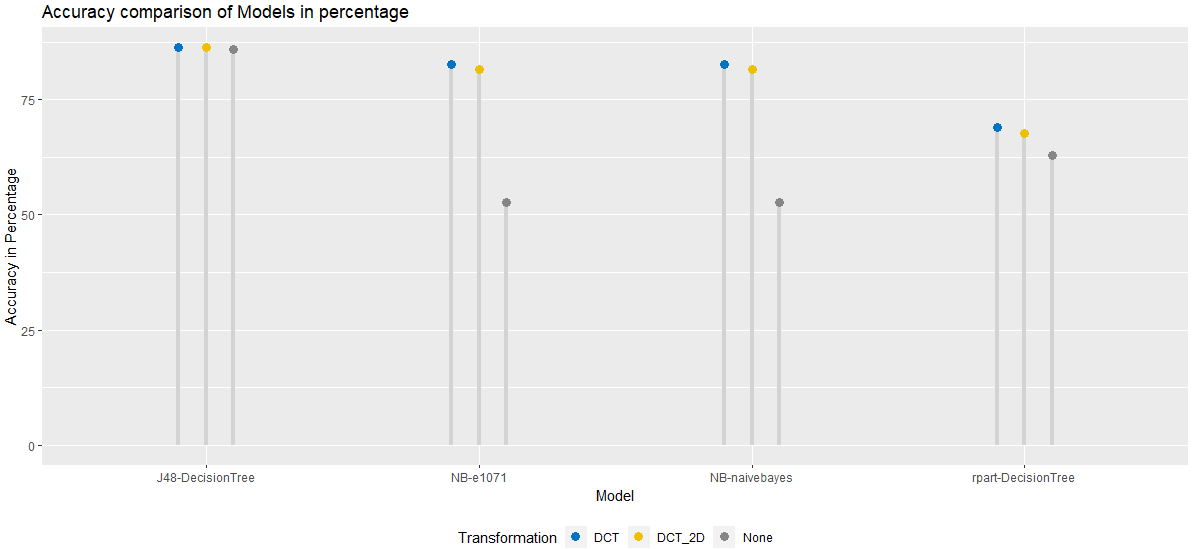


**Table 3.2 Model performance comparison**

## **Conclusion**

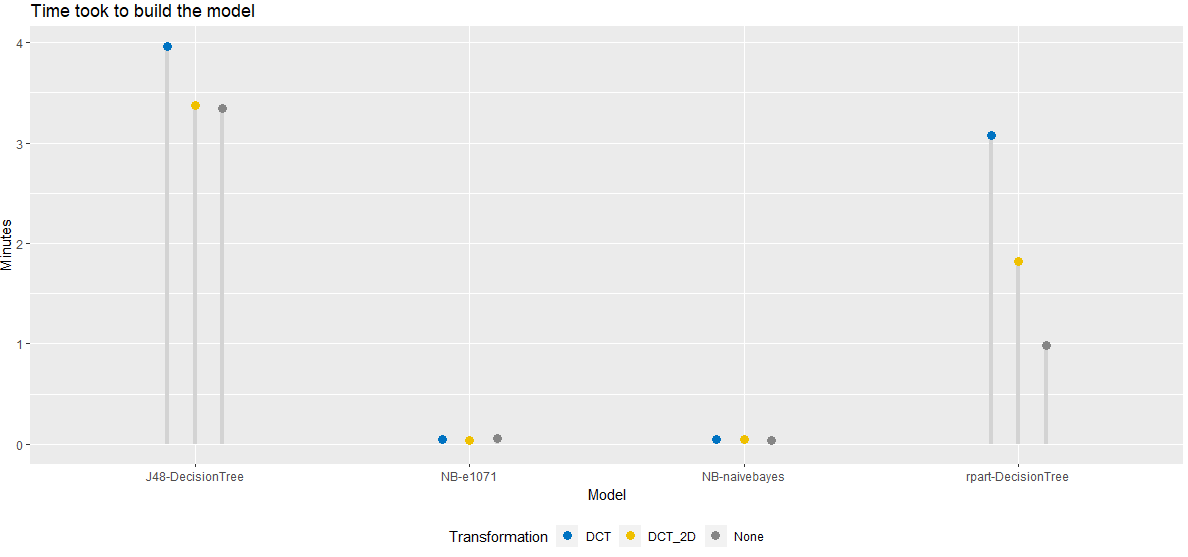
In the context of Image recognition, it is the ability of an algorithm to identify patterns in images and classify them under right buckets.

After comparing different algorithm such as Naïve Bayes and Decision Tree in identifying hand written digits present in the image, it is observed that Naïve Bayes accuracy is less (50%) when compared to Decision Tree algorithm (60% and 80%) using raw pixel values. However, when the raw pixel values are transformed into discrete cosine values, the accuracy of the Naïve Bayes algorithm improved by 30% whereas the accuracy of the Decision Tree algorithms stayed the same. Decision Tree algorithm using rpart had a slight improvement in the accuracy. Please refer **Figure 4.1** which compares the accuracy of different models and its accuracy.



**Figure 4.1 Accuracy comparison of Models**

Naïve Bayes took less time to build the model when compared to Decision Tree algorithm as shown in **Figure 4.2**. Decision Tree algorithms took more than a minute and the J48 algorithm took almost 3 to 4 minutes to build the model. Naïve Bayes took only few seconds to build the model, but the accuracy can be improved by applying proper transformation. Applying the transformation like DCT will add more time in overall while building the model.



**Figure 4.2 Performance comparison of Model build time**

The above inference shows the decision tree algorithm is outperforming Naïve Bayes in the accuracy of the image classification whereas the Naïve Bayes algorithm outperforming the Decision trees for the time required to build the model.