DEPARTMENT OF COMPUTER SCIENCE

University of Karachi

MCS Final (Morning) 2019-2021

University of Karachi
Department of Computer Science
Karachi-75270, Pakistan.

Sobia Mustafa (P19101068)

Maria Javed (P19101033)

Rida Hashmi (P19101055)

Group Members: **Safia Arshad (P19101061)**

Submitted To: Sir Tahseen Ahmed Jilani

Course Title: Data Warehousing and Data Mining **Course No:** 626

Abstract

In this report, we describe and analyze the performance of six iris encoding techniques. Principle Component Analysis (PCA) Encoding Method, K-Mean Clustering, K Near Neighbor, Random Forest, Decision Tree and Naïve Bayes. The classification tree method is a method of test design as it is used in various fields of software development. PCA is a method used to reduce the number of variables in your data by extracting a key from a large pool. Naive Bayes is a Bayes theory-based classification technique that assumes freedom among predictors. KNN (K - Nearest Neighbors) is one of the many (supervised learning) algorithms used in data mining and machine learning, a ranking algorithm where learning is based on "how many similar" data.

GROUP PROFILE

Sobia Mustafa smustafa2233@gmail.com

P19101068

Maria Javed Ur Rehman mariajaved722@gmail.com

P19101033

Rida Hashmi ridahashmi01@gmail.com

P19101055

Safia Arshad safiarain123@gmail.com

P19101061

Batch MCS (Final Year), Morning

(2020 - 2021)

Contents

Figure Contents

1. INTRODUCTION

1.1 IRIS DATASET

This is the "Iris" dataset. Originally published in UCI Machine Learning Repository: Iris Data Set, this small data set from 1936 is often used to test machine learning algorithms and concepts (e.g., Scatter Plot). Each row on the table represents an iris flower, which includes its species and its vegetative parts, the sepal and the petal dimensions in centimeters.

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
    - Iris Sentosa
    - Iris Versicolor
    - Iris Virginica

- Why we use IRIS dataset

Iris dataset is often used to test data mining, classification, and clustering examples and algorithms. That's why we choose the IRIS dataset to implement our machine learning techniques.

1.2 K-MEAN CLUSTERING ANALYSIS

The k-mean algorithm is used to create and analyze clusters. In this algorithm, the 'n' number of data points is divided into 'k' clusters based on some similarity measurement criteria. However, the results generated using this algorithm mainly depend on the selection of the initial cluster centroids. There is a wide range of techniques for finding subgroups of observations within a clustering dataset. When we cluster observations, we want the observations to be the same in the same group and the observations to be different in different groups. Since the response variable is not, it is a non-supervised method, which means that it tries to find the relationship between the observations without being trained by the response

variable. Clustering allows us to identify which observations are identical, and possibly classify them. K-Clustering is the simplest and most widely used clustering method for dividing a data set into sets of k groups.

1.3 PRINCIPAL COMPONENT ANALYSIS

Principal component analysis, or PCA, is a dimension reduction method often used to reduce the dimensions of large data sets, by converting a large set of variables into smaller ones which are still more in the larger set. More information is available. Reducing the number of variables in the dataset naturally comes at the cost of accuracy, but the trick to reducing the dimension is to trade a little accuracy for simplicity. Because discovering and imagining small data sets and analyzing data makes it much easier and faster for machine learning algorithms without any variable process. The Principal Component Analysis (PCA) feature is a method of extracting orthogonal linear estimates to capture fundamental variations of data. By far, the most popular dimension reduction approach is the principal component regression. (PCR).

1.4 K-NEAREST NEIGHBORS ALGORITHM

K Nearest Neighbor is a simple algorithm that stores all available cases and classifies new cases based on similarity measurements (e.g., distance functions). KNN has been used as a non-parametric technique in the early 1970's for statistical estimation and pattern identification. It belongs to the supervised learning domain and seeks intensive application in pattern identification, data mining and intrusion detection. It is largely disposable in real life scenarios because it is non-parametric, meaning it makes no basic assumptions about the distribution of data (unlike other algorithms like GMM, which assume a Gaussian distribution of the given data). Are). We are given some prior data (also called training data), which ranks the coordinates in groups identified by a feature.

1.5 RANDOM FOREST MODEL

Random Forests or Random Deciding Forests is a pair of learning for grading, regression, and other tasks that build up a large number of decision-making trees during training. For classification tasks, random forest production is mostly selected by the tree class. For

regression works, the average or average forecast of individual trees is returned. Random decisions are appropriate for the habit of fitting more than the training set of forest deciding trees. Random forests generally perform better than deciduous trees, but their accuracy is lower than gradient boosted trees. However, data features can affect their performance.

1.6 DECISION TREE ALGORITHM

The decision tree is a flow chart-like structure in which each internal node represents a "test" on an attribute (for example, whether the coin turns on the heads or tails), each branch of the test results Represents, and represents each leaf node. A class label (decision taken after counting all attributes). The paths from root to leaf represent the principles of classification. In decision analysis, an outline of the decision tree and its closely related influence is used as a visual and analytical decision auxiliary tool, where the expected values (or expected utility) of the competitive alternative are calculated.

1.7 NAÏVE BAYES ALGORITHM

In the statistics, the bayes classify Bayes as a family of simple "probability classifications" based on applying Bayes' theorem with assumptions of strong (null) freedom between features (see Bayes classifier). - These are one of the simplest models of the Bassian network, but combined with kernel density estimates, they can achieve high accuracy levels. Naïve Bayes classifiers are highly extensible, requiring a number of parameters linear in the number of variables (features / predictors) in the learning problem. The maximum probability training can be done by examining the closed form expression, which takes linear time, rather than through expensive repetitive estimates as is used for many other types of classification.

2. PROPOSED ALGORITHM

2.1 K-MEAN CLUSTERING ALGORITHM

algorithm proceeds to update the centroids and their clusters for balance while minimizing total differences within the clusters. It is mainly used in scenarios with real valuable properties as it relies on Euclidean distances to detect cluster centroids.

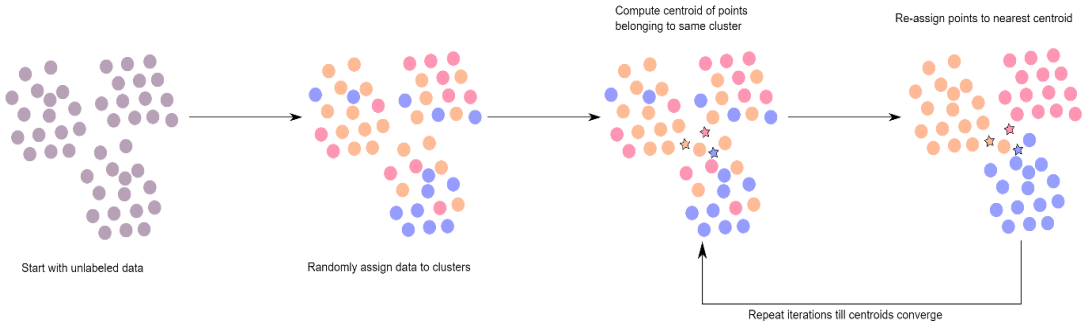


Figure 1: k-Mean cluster centroid

Here's how the K-Mean algorithm works:

- Specify number of clusters K.
- Start the centroids by first shuffling the dataset and then randomly selecting K data points for the centroid without any changes.
- Keep iterating until there is no change to the centroids. i.e., assignment of data points to clusters isn't changing.
- Compute the sum of the squared distance between data points and all centroids.
- Assign each data point to the closest cluster (centroid).
- Compute the centroids for the clusters by taking the average of all data points that belong to each cluster.
- The approach k-means follows to solve the problem is called Expectation-Maximization. The E-step is assigning the data points to the closest cluster. The M-step is computing the centroid of each cluster. Below is a breakdown of how we can solve it mathematically (feel free to skip it).

The objective function is:

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik} \|x^i - \mu_k\|^2 \qquad (1)$$

Where wik = 1 for data point xi if it belongs to cluster k; Otherwise, wik = 0. In addition, μk is the centroid of the xi cluster.

It is a matter of reducing the two parts. We reduce Jwrt first. wik and treatment μk fixed. Then we minimize Jw.r.t. μk and treatment wick fixed. Technically, we differentiate between Jw.r.t. Sell first and update cluster assignments (e-steps). Then we differentiate between Jw.r.t. μk and recount the centroids after cluster assignments from the previous step (M-step). Therefore, the e-step is:

$$\frac{\partial J}{\partial w_{ik}} = \sum_{i=1}^{m} \sum_{k=1}^{K} \|x^i - \mu_k\|^2$$

$$\Rightarrow w_{ik} = \begin{cases} 1 & \text{if } k = argmin_j \|x^i - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases} \qquad (2)$$

In other words, assign data point xi to the nearest cluster as measured by the sum of the squares of the cluster's centroids. And M stage is:

$$\frac{\partial J}{\partial \mu_k} = 2 \sum_{i=1}^{m} w_{ik}(x^i - \mu_k) = 0$$

$$\Rightarrow \mu_k = \frac{\sum_{i=1}^{m} w_{ik} x^i}{\sum_{i=1}^{m} w_{ik}} \qquad (3)$$

The translation is to re-compute the centroid of each cluster to reflect the new assignments.

2.2 PRINCIPAL COMPONENT ALGORITHM

Steps for PCA algorithm:

1. Getting the dataset:

First, we need to take the input dataset and divide it into two subdivisions, X and Y, where X is the training set, and Y is the validation set.

2. Representing data into a structure:

We will now present our dataset in the framework. As we will represent the two-dimensional matrix of the independent variable X. Here each row corresponds to the data items, and the columns correspond to the attributes. The number of columns is the dimensions of the dataset.

3. Standardizing the data:

In this step, we will standardize our dataset. As in a particular column, high-variable properties are more important than low-variable properties. If the significance of the attributes is independent of the variability of the attribute, we will divide each data item into columns with a standard deviation from the column. Here we will name the matrix Z.

$$z = \frac{value - mean}{standard\ deviation}$$

4. Calculating the Covariance of Z:

To calculate the symmetry of Z, we take the matrix Z, and move it. After transponding, we will multiply it by Z. The output matrix will be the covariance matrix of Z.

5. Calculating the Eigen Values and Eigen Vectors:

Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix Z. Eigenvectors, or covariance matrix are the axis directions with high information. And the coefficients of these eigenvectors are described as eigenvalues.

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix}$$

$$v1 = \begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix} \qquad \lambda_1 = 1.284028$$

$$v2 = \begin{bmatrix} -0.7351785 \\ 0.6778736 \end{bmatrix} \qquad \lambda_2 = 0.04908323$$

6. Sorting the Eigen Vectors:

In this step, we will take all the eigenvalues and sort them in descending order, which means from the largest to the smallest. And also arrange the eigenvectors in matrix P of eigenvalues. The resulting matrix will be named P *. Calculating new features or principal components. Here we will calculate the new features. To do this, we will multiply the P * matrix by Z. In the resulting matrix Z *, each observation is a linear sum of the original properties. Each column of the Z * matrix is independent of each other. Remove lesser or less important features from the new dataset. The new feature is set, so here we will decide what to put and what to remove. This means that we will keep only relevant or important features in the new dataset, and unimportant features will be removed.

2.3 K-NEAREST NEIGHBORS ALGORITHM

A case is classified by a majority vote of its neighbors, the case being assigned to the most common class in its K nearest neighbors measured by distance function. If K = 1, then the case is assigned only to the class next to it.

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

It should also be noted that all three distance measures are only valid for continuous variables. In the example of explicit variables, the hemming distance should be used. It also raises the issue of standardization of numerical variables between 0 and 1 when the dataset is a mixture of numeric and categorical variables.

Hamming Distance

$$D_H = \sum_{i=1}^{k}|x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

X	Y	Distance
Male	Male	0
Male	Female	1

The best value for K is chosen by examining the data first. In general, a larger K value is more accurate because it reduces overall noise but is not guaranteed. Cross-validation is another way

to determine the previously good K value by using an independent dataset to validate the K value. Historically, the best K for most data sets has been between 3-10. It produces much better results than 1NN.

2.4 RANDOM FOREST ALGORITHM

Before understanding the workings of random forest, we must consider the technique of weaving. Assembling simply means combining multiple models. Thus, a set of models is used to make predictions rather than individual models. Ensemble uses two methods:

1. Bagging:

It creates a different training subset with alternatives to sample training data and the final output is based on majority voting. For example, Random Forest

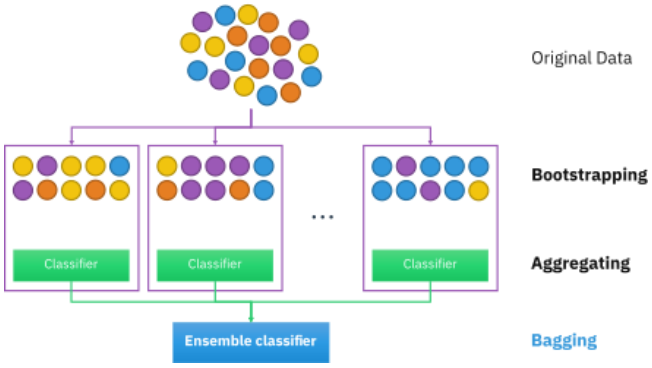


Figure 2: Random Forest Algorithm

2. Boosting:

It combines weak learners with strong learners by creating sequential models so that the final model has the most accuracy. For example, ADABOOST, XGBOOST.

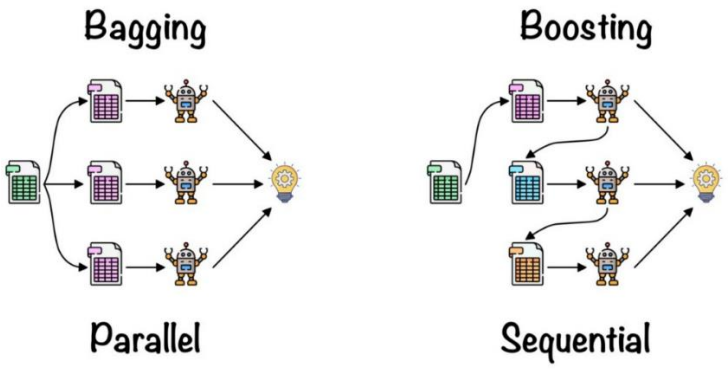


Figure 3: Bagging and Boosting

We can understand the workings of random forest algorithms with the help of the following steps.

Step 1: First, start with the selection of random patterns from the given dataset.

Step 2: Next, this algorithm will create a decision-making tree for each sample. Then he will get the result of prediction from every decision tree.

Step 3: At this stage, voting will take place for each of the predicted results.

Step 4: Finally, select the result of the most voted prediction as a result of the final prediction.

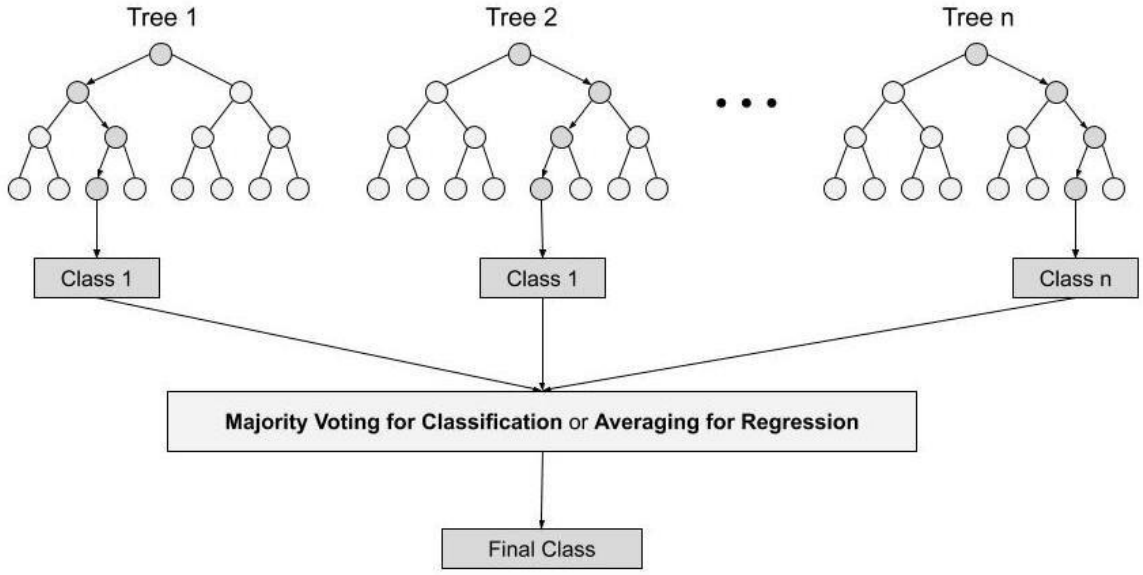


Figure 4: Tree Representation

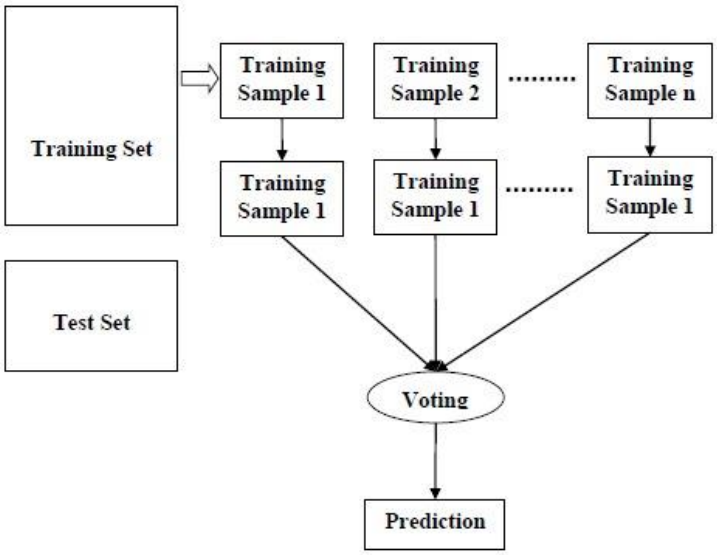


Figure 5: Random Forest Algorithm

2.5 DECISION TREE ALGORITHM

In the decision tree, to predict the class of a given dataset, the algorithm starts from the root node of the tree. This algorithm compares the root attribution values with the record (actual dataset) attribute and, based on the comparison, follows the branch, and jumps to the next node.

Step-1: Start the tree with the root node, called S, which contains the complete data set.

Step-2: Find the best attribute in a dataset using attribution selection measurement (ASM).

Step-3: Divide S into subsets that contain possible values for best attributes.

Step-4: Create a decision tree node, which has the best attribute.

Step-5: Create new decision trees again and again using the dataset subsets created in Step-3. Continue this process until you reach a stage where you can no longer classify the nodes and the final node is called the leaf node.

Attribute Selection Measures

- When implementing the decision-making tree, the main problem arises how to choose the best attribute for the root node and sub node. Therefore, there is a technique for

solving such problems which is called attribution selection measurement or ASM. With this measurement, we can easily select the best attribute for tree nodes. There are two popular techniques for ASM, which are:

- Information Gain
- Gini Index

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy (each feature)

Entropy: Entropy is a metric for measuring impurity in a particular attribute. This explains the randomness in the data. Entropy can be calculated as follows:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Were,

S= Total number of samples

P(yes)= probability of yes

P(no)= probability of no

Gini Index: The Gini index is a measure of purity or cleanliness that is used in the CART (Classification and Regression Tree) algorithm to create a decision tree.

Gini Index= 1- ∑jPj2

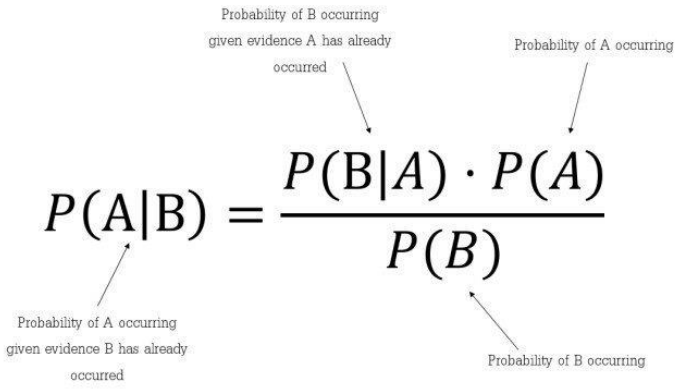
Pruning: (Getting an Optimal Decision tree)

Pruning is the process of removing unnecessary nodes from a tree to get the best decision-making tree.

- A large tree increases the risk of overfitting, and a small tree cannot capture all the important features of a dataset. Therefore, a technique that reduces the size of the learning tree without reducing the accuracy is known as pruning. There are basically two types of deforestation techniques:
- Cost Complexity Pruning
- Reduced Error Pruning.

2.6 NAÏVE BAYES ALGORITHM

Bayes Theorem is a simple mathematical formula used to calculate conditional probabilities. Conditional probability is a measure of the probability of an event occurring that caused another event (in terms of assumption, conjecture, claim, or evidence) to occur. The formula is:



Which tells us: A is B in view of how many times it occurs, the written P (A | B) is also called later probability, when we know: How many times B occurs when A occurs, P (B | A) is written and what is the probability of A being on its own, written P (A) and how likely is B to be written on its own, written P (B).6-

Assumptions made by Naïve Bayes:

The basic premise of Naïve Bayes is that each feature makes one:

- independent
- equal
- contribution to the outcome.

Let's take an example to get a better conscience. Consider the issue of car theft with color, type, originality, and purpose, theft can be either yes or no.

Bayes theorem can be rewritten as:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

The variable y is the class variable (theft?), Which represents whether the car was stolen or not. Variable X represents parameters / properties.

X is given as,

$$X = (x_1, x_2, x_3, ..., x_n)$$

Here x1, x2….xn represents attributes, meaning they can be mapped to color, type and actually. By replacing the X and expanding using the Chinese principle that we get,

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

Now, you can look up the dataset to get the values for each one and convert them into equations. For all entries in the dataset, the denominator does not change, it remains static. Therefore, the denominator can be removed, and the ratio can be injected.

$$P(y|x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

Among them, the class variable (y) is just two results, yes or no, there are cases where the classification is multiple approaches. So, to find us with at least one part to find.

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

Using the above function, we can get the class by looking at the predictors / features.

Subsequent probability P (y | X) can be calculated in advance by making a frequency table for each attribute against the target. Then, use the Naïve Bayesian equation to sort the frequency tables into the probability tables and, finally, to calculate the subsequent probability for each class. The most probable class is the result of prediction. Below are tables of frequency and

probability for all three predictions.

Frequency Table

		Stolen?	
		Yes	No
Color	Red	3	2
	Yellow	2	3

Likelihood Table

		Stolen?	
		P(Yes)	P(No)
Color	Red	3/5	2/5
	Yellow	2/5	3/5

3. MATERIALS AND DATASET

3.1 IRIS Dataset

Global data was collected using a number of measurements in taxonomic issues, and reports and data were updated from the Kaggle website.

Data View

Untitled1* | Untitled2* | iris | PC

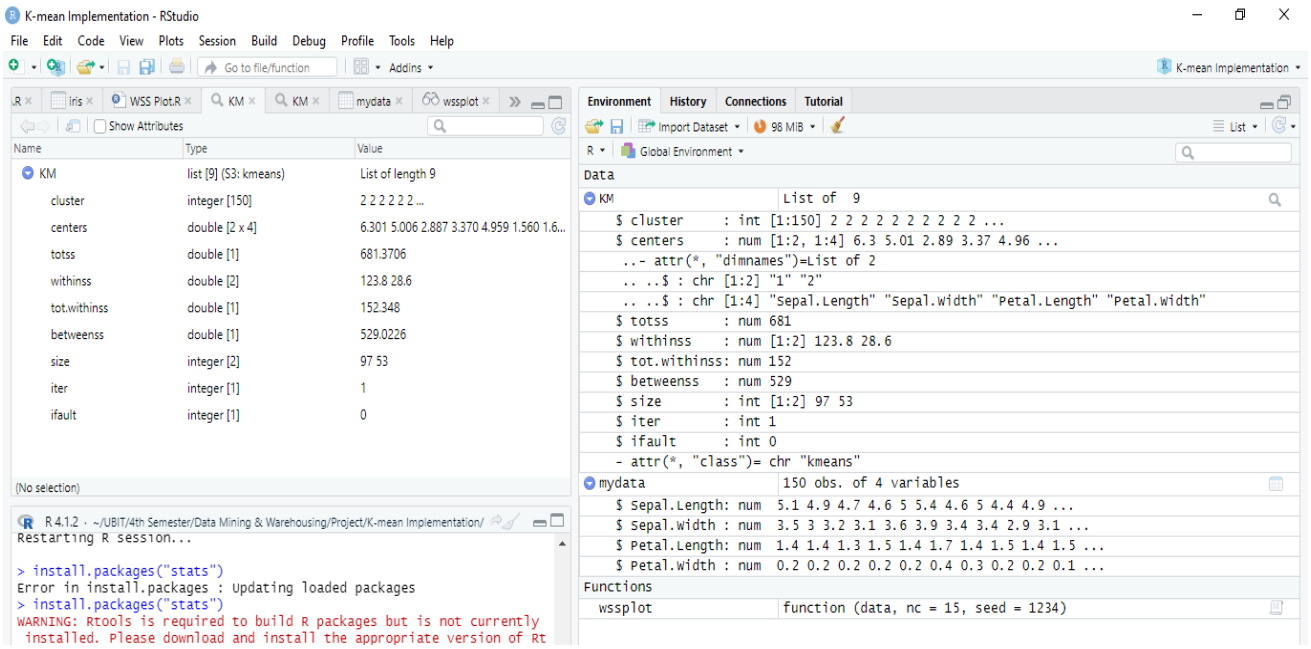
Filter

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa
13	4.8	3.0	1.4	0.1	setosa
14	4.3	3.0	1.1	0.1	setosa
15	5.8	4.0	1.2	0.2	setosa
16	5.7	4.4	1.5	0.4	setosa
17	5.4	3.9	1.3	0.4	setosa
18	5.1	3.5	1.4	0.3	setosa
19	5.7	3.8	1.7	0.3	setosa
20	5.1	3.8	1.5	0.3	setosa
21	5.4	3.4	1.7	0.2	setosa
22	5.1	3.7	1.5	0.4	setosa
23	4.6	3.6	1.0	0.2	setosa
24	5.1	3.3	1.7	0.5	setosa
25	4.8	3.4	1.9	0.2	setosa
26	5.0	3.0	1.6	0.2	setosa
27	5.0	3.4	1.6	0.4	setosa

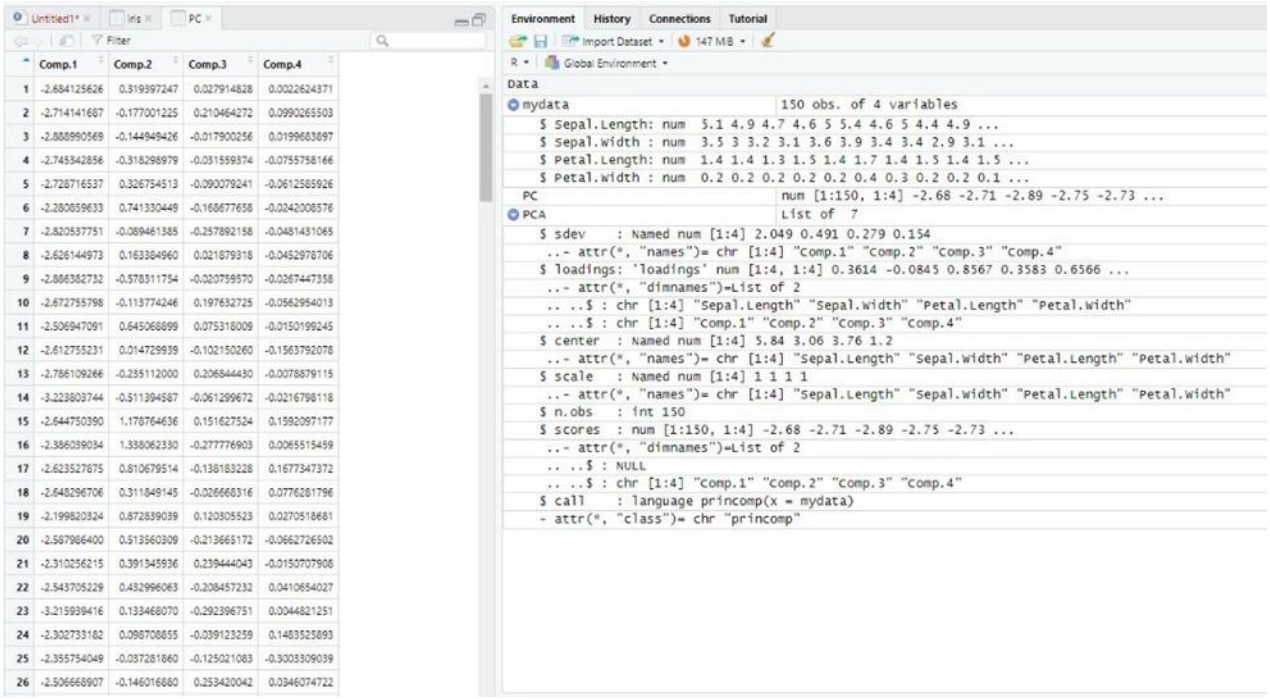
Showing 1 to 27 of 150 entries, 5 total columns

3.2 DATAVIEW OF METHODS USED

Data View of K-Mean Clustering

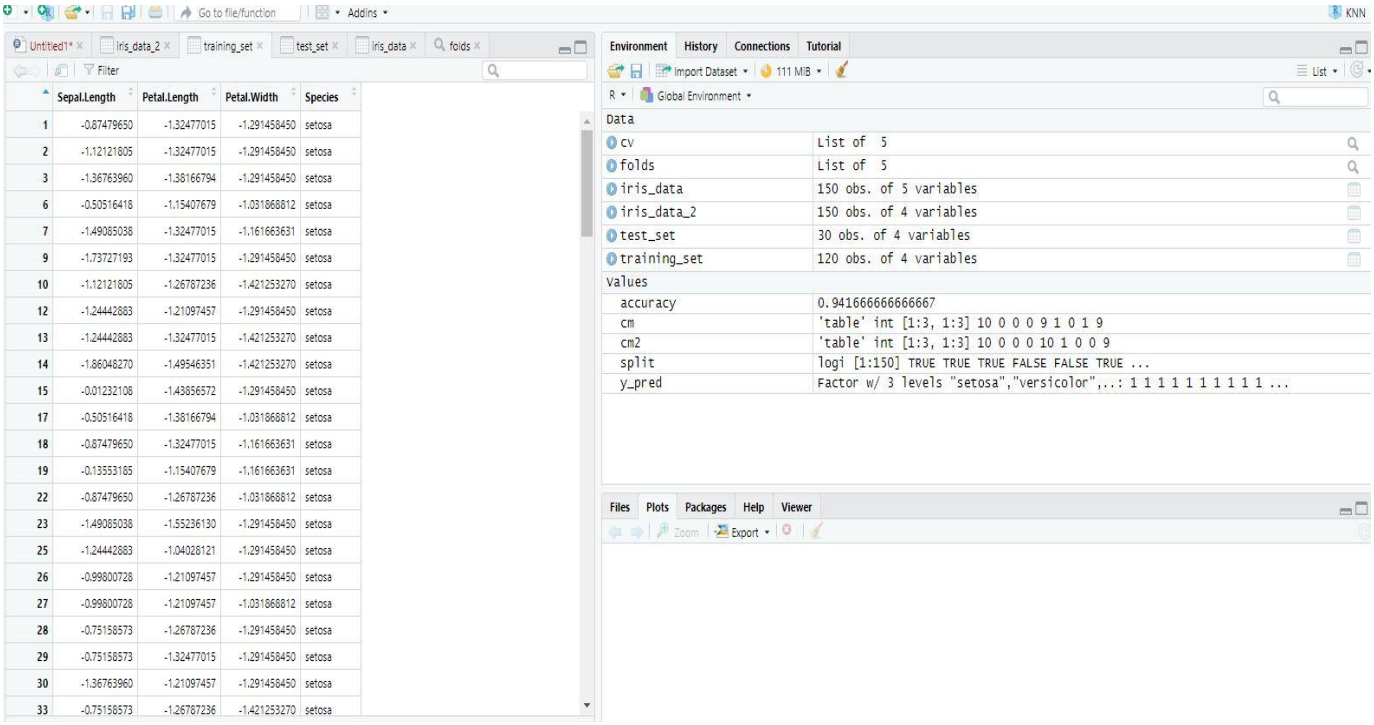


Data View of PCA



	Comp.1	Comp.2	Comp.3	Comp.4
1	-2.684125626	0.319397247	0.027914828	0.022624371
2	-2.714141687	-0.177001225	0.210464272	0.099026503
3	-2.888990569	-0.144949426	-0.017900256	0.019968397
4	-2.745342856	-0.318298979	-0.031559374	-0.075758166
5	-2.728716537	0.326754513	-0.090079241	-0.061258592
6	-2.280859633	0.741330449	-0.168677658	-0.024200657
7	-2.820537751	-0.089461385	-0.257892158	-0.048143106
8	-2.626144973	0.163384960	0.021879318	-0.045297670
9	-2.886382732	-0.578311754	-0.020759570	-0.026744735
10	-2.672755798	-0.113774246	0.197632725	-0.056295401
11	-2.506947091	0.645068899	0.075318009	-0.015019924
12	-2.612755231	0.014729939	-0.102150260	-0.156379207
13	-2.786109266	-0.235112000	0.206844430	-0.007887911
14	-3.223803744	-0.511394587	-0.061299672	-0.021679811
15	-2.644750390	1.178764636	0.151627524	0.159209717
16	-2.386039034	1.338062330	-0.277776903	0.006551545
17	-2.623527875	0.810679514	-0.138183228	0.167734737
18	-2.648296706	0.311849145	-0.026668316	0.077628179
19	-2.199820324	0.872839039	0.120305523	0.027051868
20	-2.587986400	0.513560309	-0.213665172	-0.066272650
21	-2.310256215	0.391345936	0.239444043	-0.015070790
22	-2.543705229	0.432996063	-0.208457232	0.041065402
23	-3.215939416	0.133468070	-0.292396751	0.004482125
24	-2.302733182	0.098708855	-0.039123259	0.148352569
25	-2.355754049	-0.037281860	-0.125021083	-0.300330903
26	-2.506668907	-0.146016680	0.253420042	0.034607472

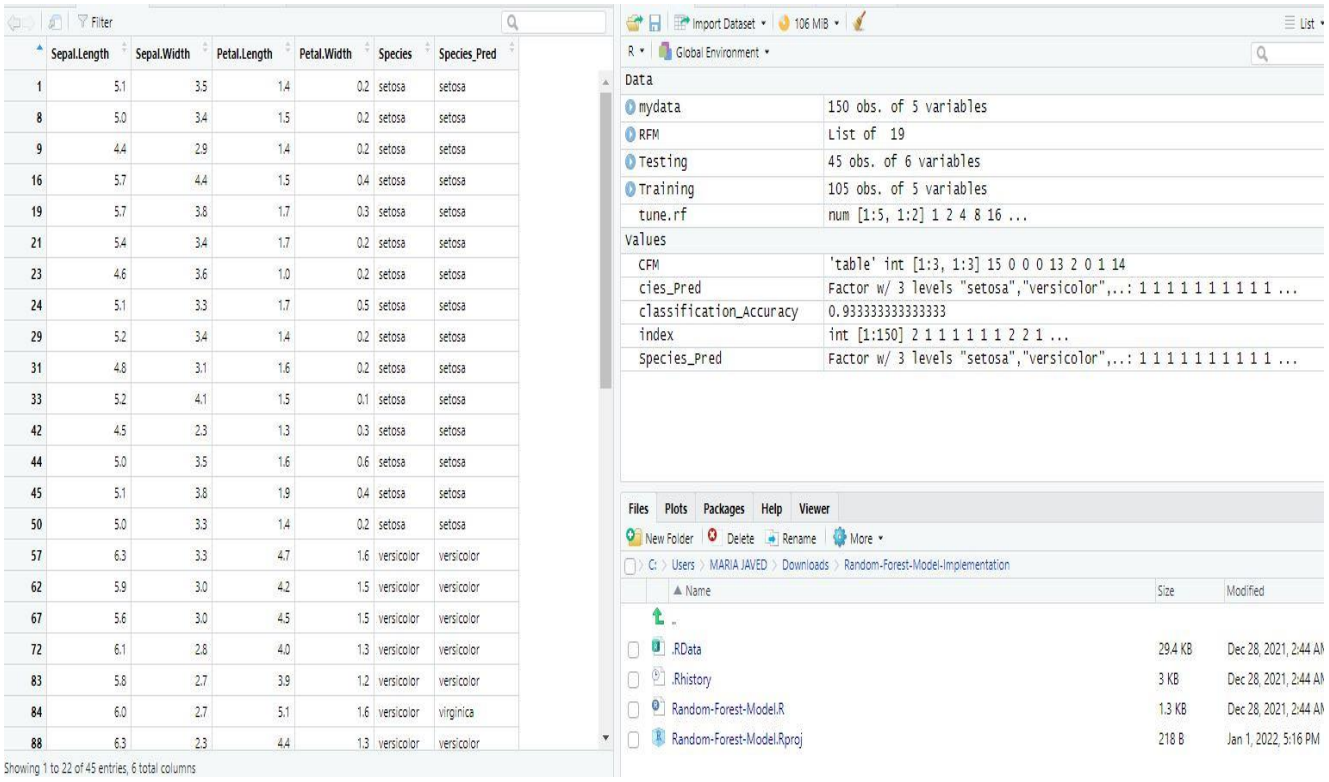
Data View of KNN



	Sepal.Length	Petal.Length	Petal.Width	Species
1	-0.87479650	-1.32477015	-1.291458450	setosa
2	-1.12121805	-1.32477015	-1.291458450	setosa
3	-1.36763960	-1.38166794	-1.291458450	setosa
6	-0.50516418	-1.15407679	-1.031868812	setosa
7	-1.49085038	-1.32477015	-1.161663631	setosa
9	-1.73727193	-1.32477015	-1.291458450	setosa
10	-1.12121805	-1.26787236	-1.421253270	setosa
12	-1.24442863	-1.21097457	-1.291458450	setosa
13	-1.24442863	-1.32477015	-1.421253270	setosa
14	-1.86048270	-1.49546351	-1.421253270	setosa
15	-0.01232108	-1.43856572	-1.291458450	setosa
17	-0.50516418	-1.38166794	-1.031868812	setosa
18	-0.87479650	-1.32477015	-1.161663631	setosa
19	-0.13553185	-1.15407679	-1.161663631	setosa
22	-0.87479650	-1.26787236	-1.031868812	setosa
23	-1.49085038	-1.55236130	-1.291458450	setosa
25	-1.24442863	-1.04028121	-1.291458450	setosa
26	-0.99800728	-1.21097457	-1.291458450	setosa
27	-0.99800728	-1.21097457	-1.031868812	setosa
28	-0.75158573	-1.26787236	-1.291458450	setosa
29	-0.75158573	-1.32477015	-1.291458450	setosa
30	-1.36763960	-1.21097457	-1.291458450	setosa
33	-0.75158573	-1.26787236	-1.421253270	setosa

Data	
cv	List of 5
folds	List of 5
iris_data	150 obs. of 5 variables
iris_data_2	150 obs. of 4 variables
test_set	30 obs. of 4 variables
training_set	120 obs. of 4 variables
Values	
accuracy	0.941666666666667
cm	'table' int [1:3, 1:3] 10 0 0 0 9 1 0 1 9
cm2	'table' int [1:3, 1:3] 10 0 0 0 10 1 0 0 9
split	logi [1:150] TRUE TRUE TRUE FALSE FALSE TRUE ...
y_pred	Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...

Data View of Random Forest



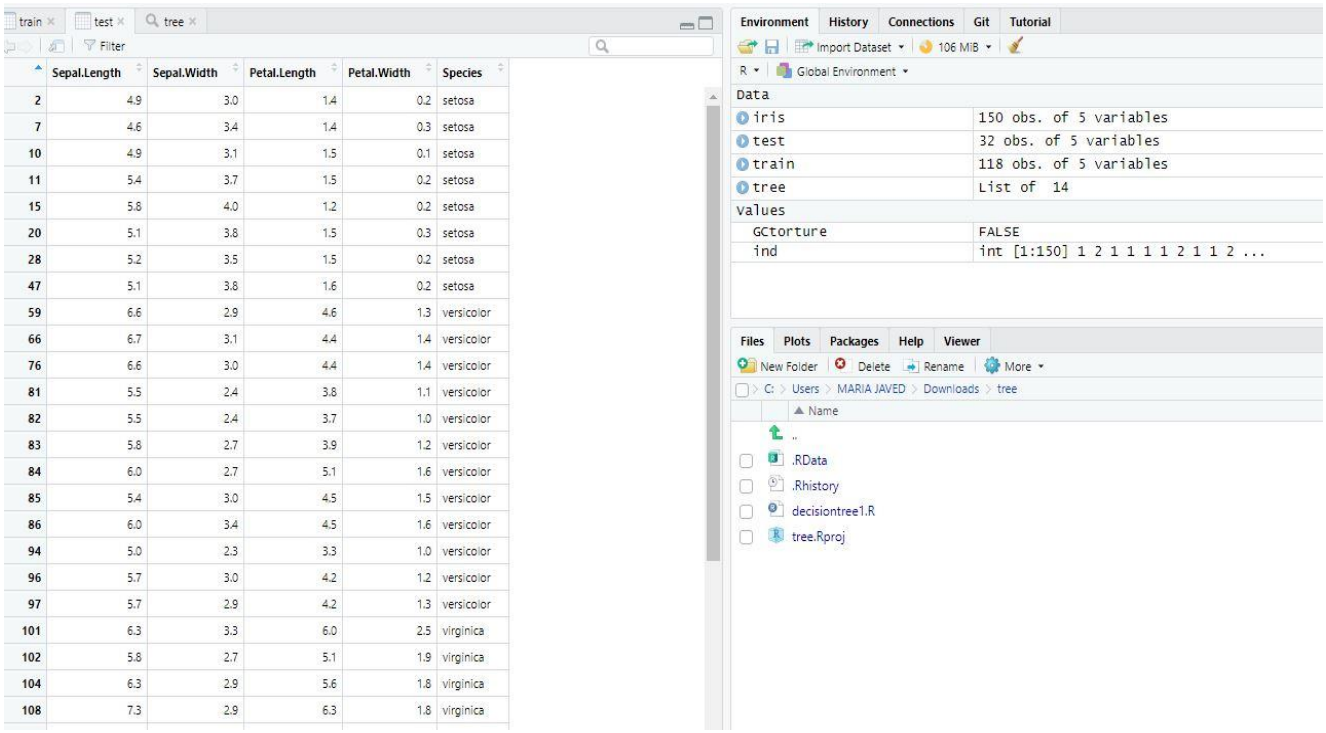
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species	Species_Pred
1	5.1	3.5	1.4	0.2	setosa	setosa
8	5.0	3.4	1.5	0.2	setosa	setosa
9	4.4	2.9	1.4	0.2	setosa	setosa
16	5.7	4.4	1.5	0.4	setosa	setosa
19	5.7	3.8	1.7	0.3	setosa	setosa
21	5.4	3.4	1.7	0.2	setosa	setosa
23	4.6	3.6	1.0	0.2	setosa	setosa
24	5.1	3.3	1.7	0.5	setosa	setosa
29	5.2	3.4	1.4	0.2	setosa	setosa
31	4.8	3.1	1.6	0.2	setosa	setosa
33	5.2	4.1	1.5	0.1	setosa	setosa
42	4.5	2.3	1.3	0.3	setosa	setosa
44	5.0	3.5	1.6	0.6	setosa	setosa
45	5.1	3.8	1.9	0.4	setosa	setosa
50	5.0	3.3	1.4	0.2	setosa	setosa
57	6.3	3.3	4.7	1.6	versicolor	versicolor
62	5.9	3.0	4.2	1.5	versicolor	versicolor
67	5.6	3.0	4.5	1.5	versicolor	versicolor
72	6.1	2.8	4.0	1.3	versicolor	versicolor
83	5.8	2.7	3.9	1.2	versicolor	versicolor
84	6.0	2.7	5.1	1.6	versicolor	virginica
88	6.3	2.3	4.4	1.3	versicolor	versicolor

Showing 1 to 22 of 45 entries, 6 total columns

Data	
mydata	150 obs. of 5 variables
RFM	List of 19
Testing	45 obs. of 6 variables
Training	105 obs. of 5 variables
tune.rf	num [1:5, 1:2] 1 2 4 8 16 ...
Values	
CFM	'table' int [1:3, 1:3] 15 0 0 0 13 2 0 1 14
cies_Pred	Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
classification_Accuracy	0.933333333333333
index	int [1:150] 2 1 1 1 1 1 1 2 2 1 ...
Species_Pred	Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...

Name	Size	Modified
.RData	29.4 KB	Dec 28, 2021, 2:44 AM
.Rhistory	3 KB	Dec 28, 2021, 2:44 AM
Random-Forest-Model.R	1.3 KB	Dec 28, 2021, 2:44 AM
Random-Forest-Model.Rproj	218 B	Jan 1, 2022, 5:16 PM

Data View of Decision Tree



train × test × tree ×

Filter

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
2	4.9	3.0	1.4	0.2	setosa
7	4.6	3.4	1.4	0.3	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa
15	5.8	4.0	1.2	0.2	setosa
20	5.1	3.8	1.5	0.3	setosa
28	5.2	3.5	1.5	0.2	setosa
47	5.1	3.8	1.6	0.2	setosa
59	6.6	2.9	4.6	1.3	versicolor
66	6.7	3.1	4.4	1.4	versicolor
76	6.6	3.0	4.4	1.4	versicolor
81	5.5	2.4	3.8	1.1	versicolor
82	5.5	2.4	3.7	1.0	versicolor
83	5.8	2.7	3.9	1.2	versicolor
84	6.0	2.7	5.1	1.6	versicolor
85	5.4	3.0	4.5	1.5	versicolor
86	6.0	3.4	4.5	1.6	versicolor
94	5.0	2.3	3.3	1.0	versicolor
96	5.7	3.0	4.2	1.2	versicolor
97	5.7	2.9	4.2	1.3	versicolor
101	6.3	3.3	6.0	2.5	virginica
102	5.8	2.7	5.1	1.9	virginica
104	6.3	2.9	5.6	1.8	virginica
108	7.3	2.9	6.3	1.8	virginica

Environment | History | Connections | Git | Tutorial

Import Dataset | 106 MiB

R | Global Environment

Data

iris	150 obs. of 5 variables
test	32 obs. of 5 variables
train	118 obs. of 5 variables
tree	List of 14

values

GCtorture	FALSE
ind	int [1:150] 1 2 1 1 1 1 2 1 1 2 ...

Files | Plots | Packages | Help | Viewer

New Folder | Delete | Rename | More

C: > Users > MARIA JAVED > Downloads > tree

Name

..

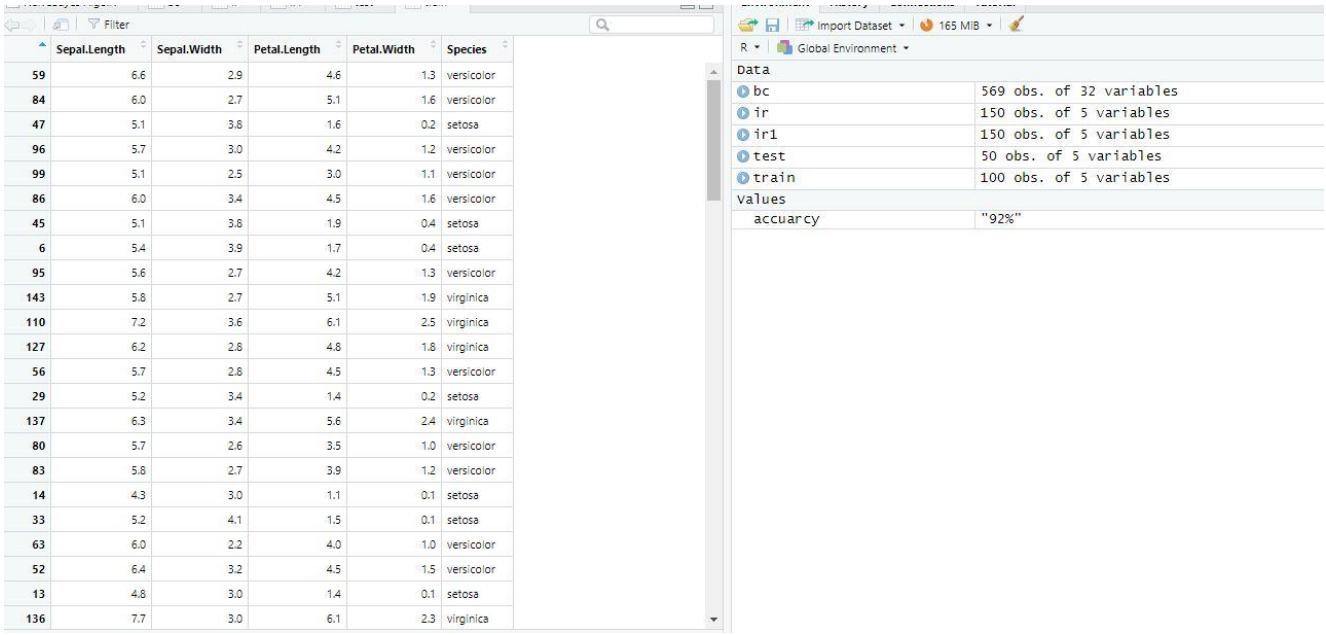
.RData

.Rhistory

decisiontree1.R

tree.Rproj

Data View of Naïve Bayes



Filter

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
59	6.6	2.9	4.6	1.3	versicolor
84	6.0	2.7	5.1	1.6	versicolor
47	5.1	3.8	1.6	0.2	setosa
96	5.7	3.0	4.2	1.2	versicolor
99	5.1	2.5	3.0	1.1	versicolor
86	6.0	3.4	4.5	1.6	versicolor
45	5.1	3.8	1.9	0.4	setosa
6	5.4	3.9	1.7	0.4	setosa
95	5.6	2.7	4.2	1.3	versicolor
143	5.8	2.7	5.1	1.9	virginica
110	7.2	3.6	6.1	2.5	virginica
127	6.2	2.8	4.8	1.8	virginica
56	5.7	2.8	4.5	1.3	versicolor
29	5.2	3.4	1.4	0.2	setosa
137	6.3	3.4	5.6	2.4	virginica
80	5.7	2.6	3.5	1.0	versicolor
83	5.8	2.7	3.9	1.2	versicolor
14	4.3	3.0	1.1	0.1	setosa
33	5.2	4.1	1.5	0.1	setosa
63	6.0	2.2	4.0	1.0	versicolor
52	6.4	3.2	4.5	1.5	versicolor
13	4.8	3.0	1.4	0.1	setosa
136	7.7	3.0	6.1	2.3	virginica

Import Dataset | 165 MiB

R | Global Environment

Data

bc	569 obs. of 32 variables
ir	150 obs. of 5 variables
ir1	150 obs. of 5 variables
test	50 obs. of 5 variables
train	100 obs. of 5 variables

values

accuarcy	"92%"

4 TECHNOLOGIES USED

4.1 INTRODUCTION TOR

R is a language and environment for statistical computing and graphics. This is a GNU project similar to S Language and Environment, developed by John Chambers and colleagues at Bell Laboratories (formerly AT&T, now Lucent Technologies). R can be thought of as a different implementation of S. There are some major differences, but most of the code written for S runs unchanged under R.

R provides a wide variety of data (linear and nonlinear modeling, classical statistical tests, time series analysis, classification, clustering,) and graphical techniques, and is highly scalable. The S language is often the vehicle of choice for statistical method research, and R provides an open-source way to participate in this activity.

One of the strengths of R is the ease with which well-designed publication quality plots can be created, including mathematical symbols and formulas where needed. Defaults are considered in the selection of minor designs in graphics, but the user retains complete control.

R Free software is available in the form of source code under the terms of the GNU General Public License of the Free Software Foundation. It compiles and runs on a variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and MacOS.

The R environment

R is an integrated suite of software facilities for data manipulation, calculation, and graphical displays. It includes.

- An efficient data handling and storage facility.
- A set of operators to calculate rows, specifically the matrix.
- A large, integrated, integrated set of intermediate tools for data analysis.
- Graphical facilities for data analysis and display on screen or hard copy.
- A well-developed, simple, and efficient programming language that includes conditional, loops, user-defined recreational functions, and input and output features.

4.2 INTRODUCTION TO RSTUDIO

RStudio R is an integrated development environment (IDE). It includes a console, a syntax highlighting editor that executes code directly, as well as tools for plotting, history, debugging and workspace management. RStudio is available in open source and commercial editions and runs on desktop (Windows, Mac, and Linux) or browsers connected to RStudio Server or RStudio Server Pro (Debian / Ubuntu, RedHat / CentOS, and SUSE Linux). RStudio R is a free and open-source Integrated Development Environment (IDE) for R, a programming language for statistical computing and graphics. RStudio was founded by JJ Allaire, creator of the programming language ColdFusion. Hadley Wickham is the Chief Scientist at RStudio. RStudio is available in two editions: RStudio Desktop, where the program is run locally as a regular desktop application. And RStudio Server, which allows access to RStudio using a web browser while running on a remote Linux server. RStudio desktop pre-packaged distributions are available for Windows, OS X, and Linux.

RStudio is written in the C ++ programming language and uses the Qt framework for its graphical user interface. Work on RStudio began around December 2010, and the first public beta version (v0.92) was officially announced in February 2011.

4.3 WHY WE USE RSTUDIO?

RStudio helps the world realize data regardless of the ability to pay. The main goal of RStudio is to create free and open-source software for data science, scientific research and technical communication. It allows anyone with computer access to participate freely in the global economy, which rewards data literacy. Increases the production and use of knowledge; And facilitates collaboration and reproductive research in science, education and industry. We spend more than 50% of our engineering resources on open-source software development and provide extensive support to the open-source data science community.

5 ANALYSIS PHASES

5.1 INITIAL STATE

5.1.1 Installation of Packages

Why we need that package?

- **Stats** The package contains the necessary function to perform the analysis of the meaning of K.
- **Dplyr** Shortcuts required for subdivision, summary, rearrangement, and data sets. Dplyr is our outgoing package for high-speed data manipulation.
- **ggplot2** R's popular package for creating beautiful graphics. ggplot2 lets you use graphics grammar to create layered, custom plots.
- **e1071** A package provides functionality for statistics and potential algorithms such as physical classifier, navy bise classifier, bagged clustering, short time foyer transform, support vector machine etc.
- **random Forest** Packages are used to create and analyze random forests.
- **caTools** Access control is easy to provide and related classes are easy to find.**caret** Use to work on large datasets, we can use some machine learning packages directly.

```
#Install pre-requisite packages #
install.packages("stats")
install.packages("dplyr")
install.packages("ggplot2")
install.packages("ggfortify")
install.packages("randomForest")
install.packages("e1071")
install.packages('caTools')
install.packages('caret')
```

5.1.2 Installation of Libraries

Following are the libraries which used:

```
#Load required libraries #
library(stats)
library(dplyr)
library(ggplot2)
library(ggfortify)
library(party)
library(rpart)
library(rpart.plot)
library(caTools)
library(datasets)
library(class)
library(caret)
```

5.2 SECOND STATE

5.2.1 Insert Data

Untitled1* | Untitled2* | iris | PC

Filter

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa
13	4.8	3.0	1.4	0.1	setosa
14	4.3	3.0	1.1	0.1	setosa
15	5.8	4.0	1.2	0.2	setosa
16	5.7	4.4	1.5	0.4	setosa
17	5.4	3.9	1.3	0.4	setosa
18	5.1	3.5	1.4	0.3	setosa
19	5.7	3.8	1.7	0.3	setosa
20	5.1	3.8	1.5	0.3	setosa
21	5.4	3.4	1.7	0.2	setosa
22	5.1	3.7	1.5	0.4	setosa
23	4.6	3.6	1.0	0.2	setosa
24	5.1	3.3	1.7	0.5	setosa
25	4.8	3.4	1.9	0.2	setosa
26	5.0	3.0	1.6	0.2	setosa
27	5.0	3.4	1.6	0.4	setosa

Showing 1 to 27 of 150 entries, 5 total columns

5.2.2 Convert IRIS data to Unlabeled Data

```
#Load required libraries #
library(stats)
library(dplyr)
library(ggplot2)
library(ggfortify)
```

5.3 LAST STATE

Implement the following methods:

- K Mean Clustering
- Principle Component Analysis
- K Nearest Neighbor
- Random Forest
- Decision Tree
- Naïve Bayes Classification

6 EXPERIMENTAL RESULTS

6.1 EXPERIMENTAL RESULT OF K-MEAN

The PCA creates a new coordinate system by converting highly integrated variables into orthogonal or unrelated variables. The dataset contains 150 observations that are evenly distributed among three species - Satosa, Versicular and Virginica.

Loading data

data(iris)

Structure

str(iris)

```
> # Loading data
> data(iris)
>
> # Structure
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

summary(iris)

```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

df <- iris

head(iris)

```
> df <- iris
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

6.1.1 Scatter plot

```
# make a scatterplot.
ggplot(iris,aes(x = Sepal.Length, y = Sepal.Width, col= Species)) +
geom_point(aes(col=Species),size = 3)
```

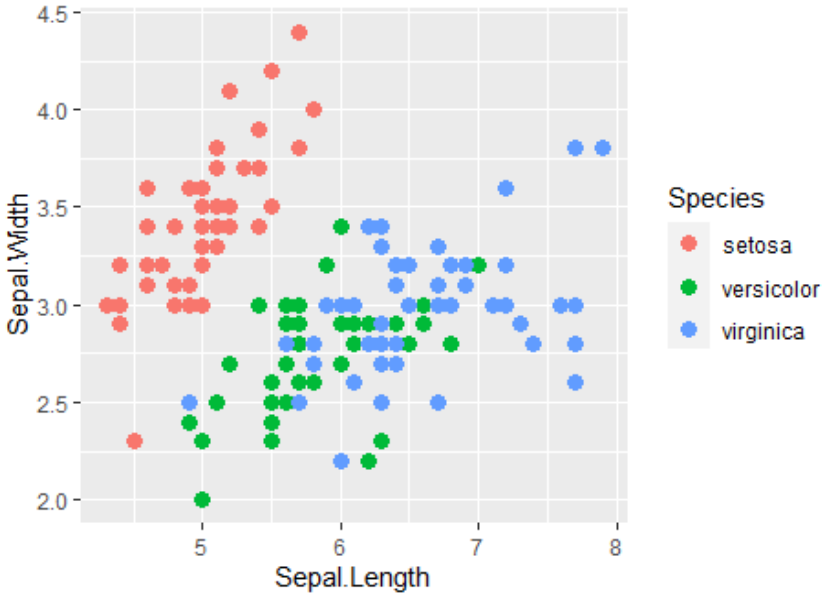


Figure 6: Scatter Plot between sepal length and sepal width in K Mean Clustering Method

```
ggplot(iris,aes(x = Petal.Length, y = Petal.Width, col= Species)) +
geom_point(aes(col=Species),size = 3)
```

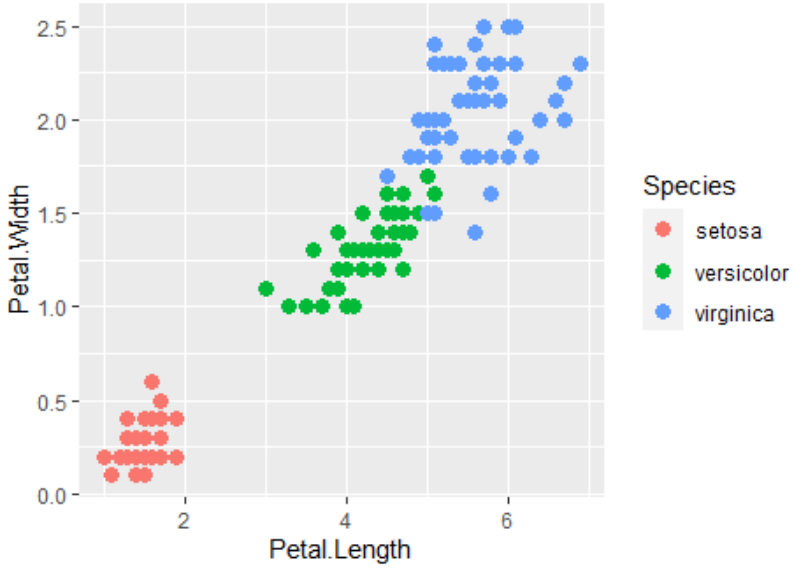


Figure 7: Scatter Plot between petal length and petal width in K Mean Clustering Method

As we can see, setosa is going to be clustered easier. Meanwhile, there is noise between versicolor and virginica even when they look like perfectly clustered.

```
# Fitting K-Means clustering Model
# to training dataset
set.seed(101)
irisCluster <- kmeans(df[,1:4], center=3, nstart=20)
irisCluster
```

```
> irisCluster
K-means clustering with 3 clusters of sizes 38, 62, 50

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     6.850000    3.073684     5.742105    2.071053
2     5.901613    2.748387     4.393548    1.433871
3     5.006000    3.428000     1.462000    0.246000

Clustering vector:
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [49] 3 3 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [97] 2 2 2 2 1 2 1 1 1 1 2 1 1 1 1 1 1 2 2 1 1 1 1 2 1 2 1 2 1 1 2 2 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2 1
[145] 1 1 2 1 1 2

Within cluster sum of squares by cluster:
[1] 23.87947 39.82097 15.15100
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
```

The 3 clusters are made which are of 38, 62, and 50 sizes respectively. Within the cluster, the sum of squares is 88.4%.

6.1.2 Confusion Matrix

```
# Confusion Matrix
cm <- table(df$Species, irisCluster$cluster)
cm
```

```
> cm

    setosa versicolor virginica
  1      0          2        36
  2      0         48        14
  3     50          0         0
```

Therefore, out of 50 cetosas, 50 centsosa is classified as Virginica. Out of 62 Versicolor, 2 Versicolor is classified as Setosa and 48 Versicolor is correctly classified as Versicolor. Of the 36 Virginicas, 19 are classified as Citusa and 14 as Versace color.

6.1.3 Model Evaluation and visualization

```
# Model Evaluation and visualization
plot(iris[c("Sepal.Length", "Sepal.Width")])
```

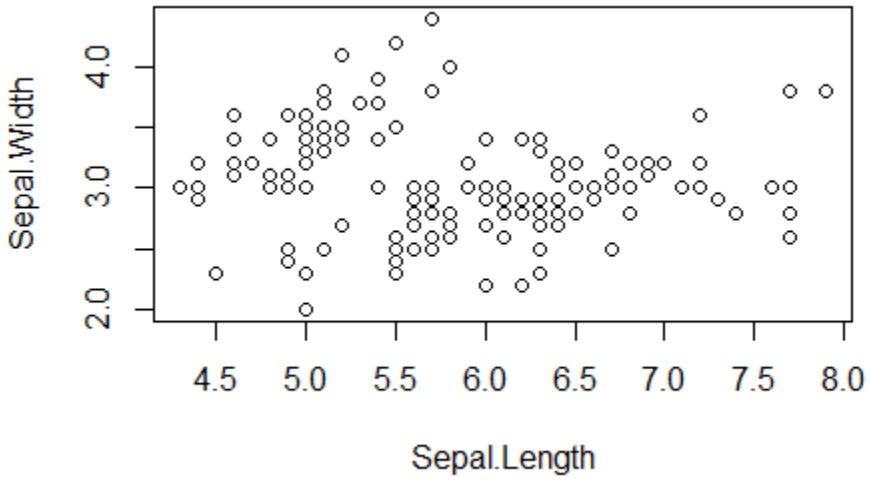


Figure 8: Model evaluation plot of sepal in K Mean Clustering Method

```
plot(iris[c("Sepal.Length", "Sepal.Width")], col = irisCluster$cluster)
```

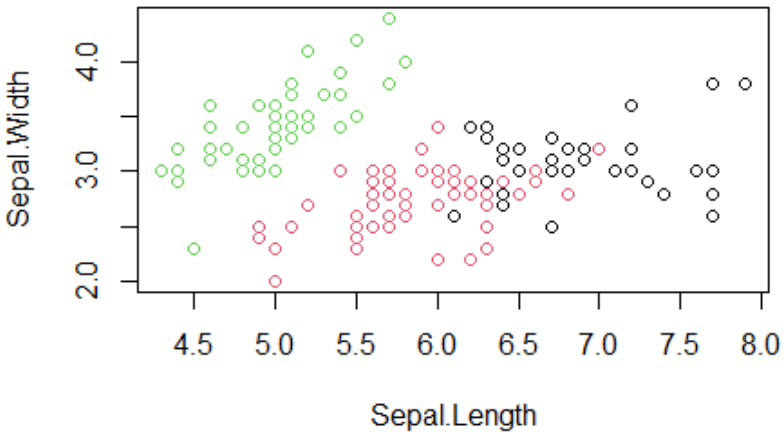


Figure 9: Model Visualize plot of sepal in K Mean Clustering Method

plot(iris[c("Sepal. Length", "Sepal. Width")], col = iris Cluster $cluster, main = "K-means with 3 clusters")

The model showed 3 cluster plots with three different colors and Sepal. Length and Sepal. Width.

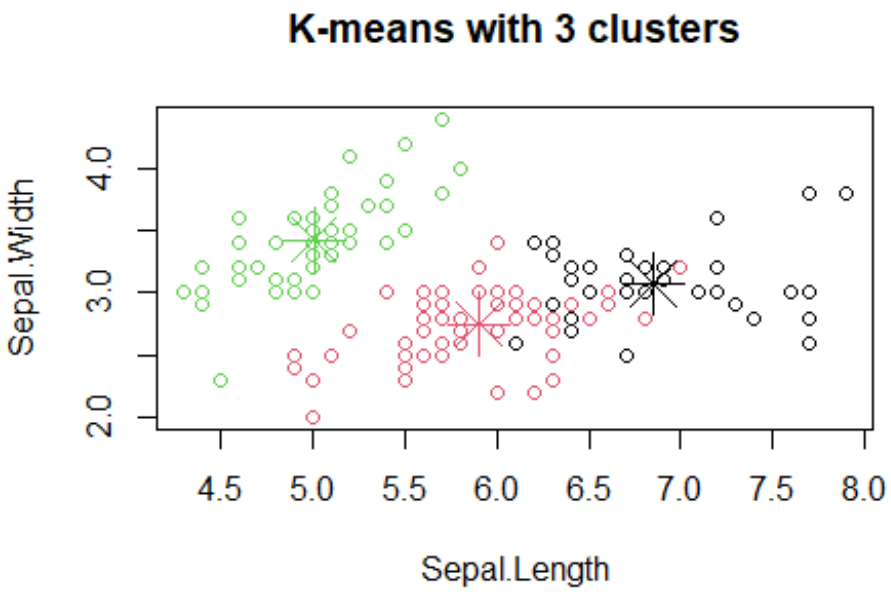


Figure 10: K-mean with 3 clusters

In the plot, the cluster centers are marked with the same color as the cross marks.

6.1.4 Plotting cluster centers

Plotting cluster centers
Iris Cluster$ centers

```
> irisCluster$centers
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     6.850000    3.073684     5.742105    2.071053
2     5.901613    2.748387     4.393548    1.433871
3     5.006000    3.428000     1.462000    0.246000
```

irisCluster$centers[, c("Sepal. Length", "Sepal. Width")]

```
> irisCluster$centers[, c("Sepal.Length", "Sepal.Width")]
  Sepal.Length Sepal.Width
1     6.850000    3.073684
2     5.901613    2.748387
3     5.006000    3.428000
```

Finding the center of cluster one, we can conclude on average that the length of all observations inside the cluster is approximately 6.850000cm, the width of the sepel is approximately 3.073684cm, the petal length is approximately 5.742105cm and the petal width is 2.071053cm. - Looking at the center of the average 2, the length of the sapling is 5.901613 cm, the width of the sapel is 2.748387 cm, the length of the petal is 4.393548 cm and the width of the petal is 1.433871 cm. Looking at the center of the average 3, the length of the apple is 5.006000 cm, the width of the apple is 3.428000 cm, the length of the petal is 1.462000 cm and the width of the petal is 0.246000 cm.

```
# cex is font size, pch is symbol
points(irisCluster$centers[, c("Sepal. Length", "Sepal.Width")], col = 1:3, pch = 8, cex = 3)
```

6.1.5 elbow Plot

```
#  If we wanted to know the exact number of centers, we should have made an elbow
method.
tot.withinss <- vector(mode="character", length=10)
for (i in 1:10){
  irisCluster <- kmeans(df[,1:4], center=i, nstart=20)
  tot.withinss[i] <- irisCluster$tot.withinss
}
# visualize WSS Plot
plot(1:10, tot.withinss, type="b", pch=19)
#screeplot or elbow graph
screeplot(pc, type="lines")
```

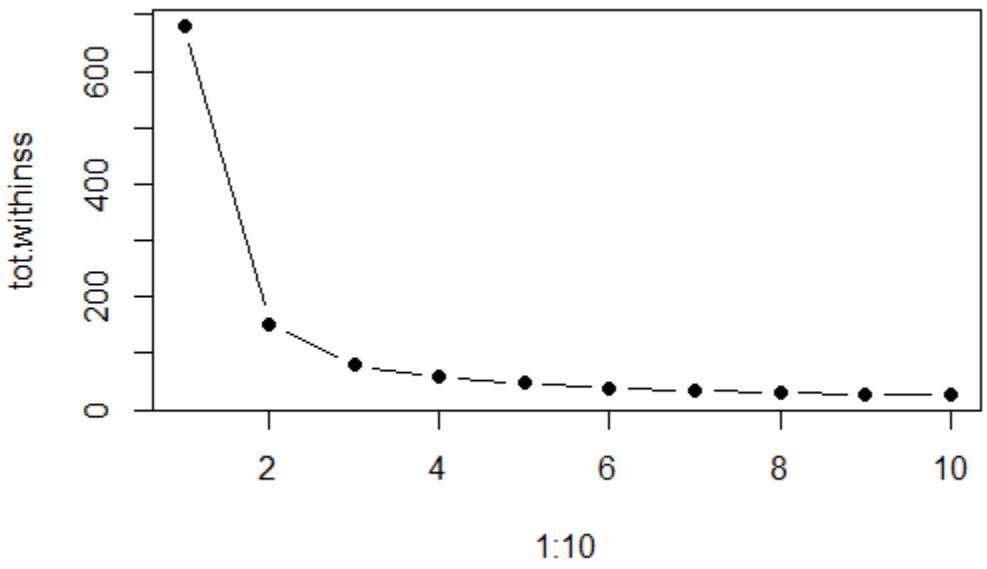


Figure 11: Elbow Plot Of K-mean cluster

The final cluster model is created using the final model k mean and k = 3. The nstart value is also defined as 20 which means that R20 will test different random starting assignments and then select the lowest one within the cluster variable.

6.1.6 Visualizing clusters

```
# Visualizing clusters
clusplot(iris, irisCluster$cluster,
        color=T,
        shade=T,
        labels=0,
        lines=0,
        plotchar = FALSE,
        span = TRUE,
        main = paste("Cluster iris"),
        xlab = 'Sepal.Length',
        ylab = 'Sepal.Width')
```

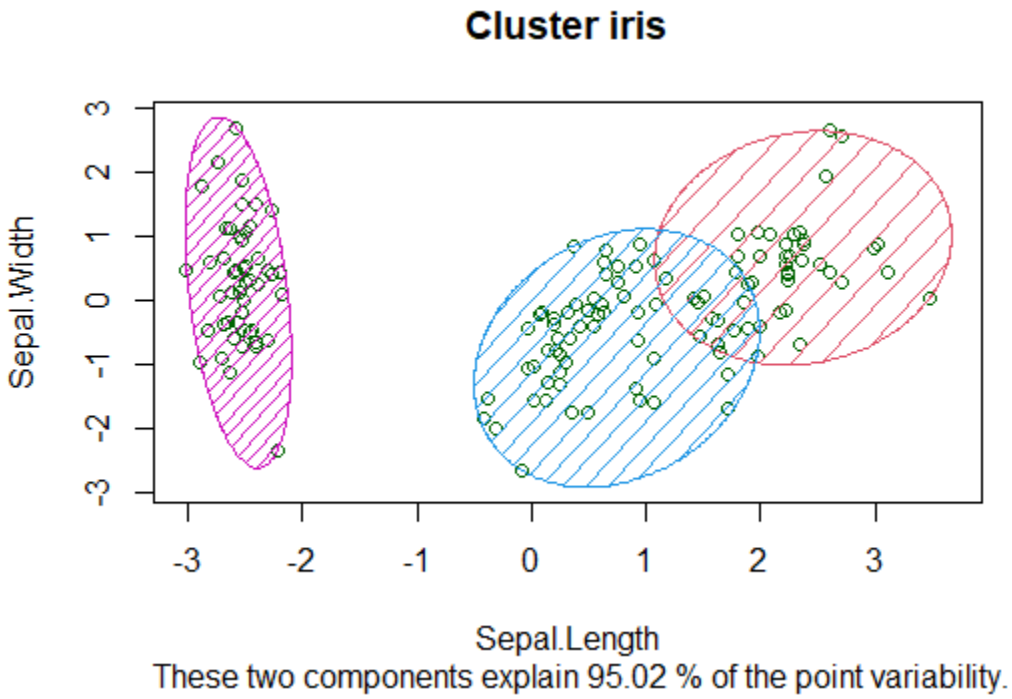


Figure 12: Cluster iris of k-mean

Therefore, 3 clusters are formed with different sapling lengths and sapling widths. Therefore, K-Means clustering algorithm is widely used in industry.

6.1.7 Conclusion

If we observe carefully, we can see that the values of apple length, apple width, petal length and petal width are all different but cluster 2 and cluster 3 have the same value which is the center of these clusters overlapping. There is overlapping so we suggest that these clusters are non-existent.

Data	
df	150 obs. of 5 variables
iris	150 obs. of 5 variables
irisCluster	List of 9
Values	
cm	'table' int [1:3, 1:3] 0 0 50 2 48 0…
i	10L
tot.withinss	chr [1:10] "681.370599999999" "152.3…

6.2 EXPERIMENTAL RESULT OF PCA

PCA creates new coordinate system by converting highly correlated variables into orthogonal or uncorrelated variables.

```
data("iris")
str(iris)
```

```
> data("iris")
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

summary(iris)

```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length
 Min.   :4.300   Min.   :2.000   Min.   :1.000
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600
 Median :5.800   Median :3.000   Median :4.350
 Mean   :5.843   Mean   :3.057   Mean   :3.758
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100
 Max.   :7.900   Max.   :4.400   Max.   :6.900
  Petal.Width          Species
 Min.   :0.100   setosa    :50
 1st Qu.:0.300   versicolor:50
 Median :1.300   virginica :50
 Mean   :1.199
 3rd Qu.:1.800
 Max.   :2.500
```

```
#data partition
set.seed(419)
ind <- sample(2, nrow(iris), replace = TRUE, prob = c(0.8,0.2))
train <- iris[ind==1,]
test <- iris[ind==2,]
dim(train)
dim(test)
```

```
> dim(train)
[1] 113   5
> dim(test)
[1] 37  5
```

6.2.1 scatter plot & correlations

```
#scatter plot & correlations
install.packages('psych')
library(psych)
pairs.panels(train[,-5], gap=0, bg=c("red","yellow", "blue")[train$Species], pch=21)
```

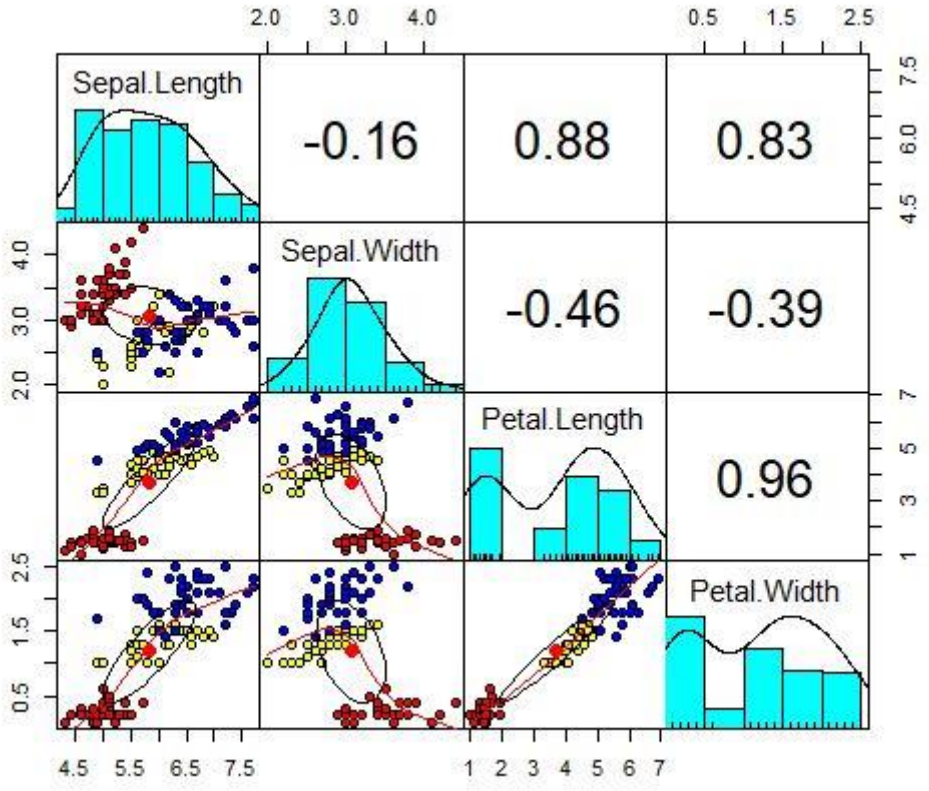


Figure 13 Scatter plot PCA

```
#Principal Component Analysis
pc <- prcomp(train[,-5], center = TRUE, scale. = TRUE)
pc
attributes(pc)
```

```
Standard deviations (1, .., p=4):
[1] 1.7215199 0.9373729 0.3734702 0.1349866

Rotation (n x k) = (4 x 4):
                    PC1         PC2        PC3        PC4
Sepal.Length  0.5192954 -0.38123664  0.7175786  0.2647109
Sepal.Width  -0.2898304 -0.91973272 -0.2315345 -0.1283815
Petal.Length  0.5763578 -0.03442750 -0.1385648 -0.8046280
Petal.Width   0.5604840 -0.08697755 -0.6420845  0.5157714
> attributes(pc)
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

#average of four variables
pc$center
mean(train$Sepal.Length)

#calculate standard deviations of four variales
pc$scale
sd(train$Sepal.Length)

#sd, rotations also called loadings
print(pc)

```
> summary(pc)
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     1.7215 0.9374 0.37347 0.13499
Proportion of Variance 0.7409 0.2197 0.03487 0.00456
Cumulative Proportion  0.7409 0.9606 0.99544 1.00000
> attributes(pc)
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

print(pc)

```
> print(pc)
Standard deviations (1, .., p=4):
[1] 1.7215199 0.9373729 0.3734702 0.1349866

Rotation (n x k) = (4 x 4):
                    PC1         PC2        PC3        PC4
Sepal.Length  0.5192954 -0.38123664  0.7175786  0.2647109
Sepal.Width  -0.2898304 -0.91973272 -0.2315345 -0.1283815
Petal.Length  0.5763578 -0.03442750 -0.1385648 -0.8046280
Petal.Width   0.5604840 -0.08697755 -0.6420845  0.5157714
```

6.2.2 elbow graph

```
#screeplot or elbow graph
screeplot(pc, type="lines")
```

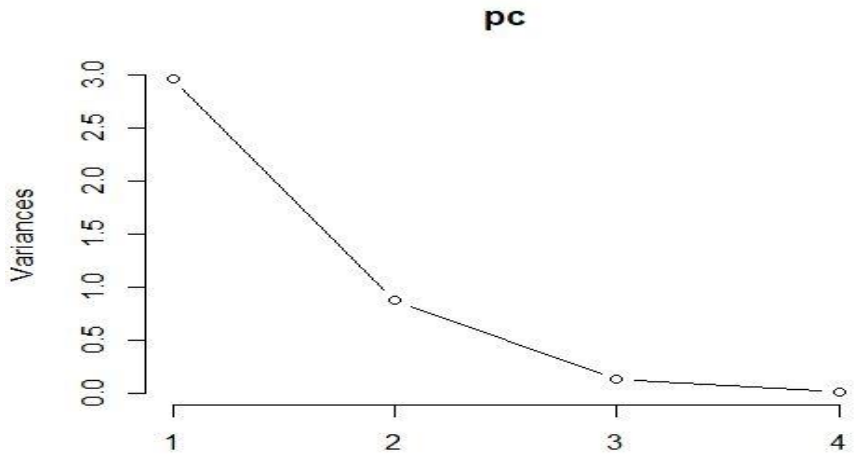


Figure 14 Elbow graph of PCA

```
#Kaiser's Criterion
pc$sdev^2

#prediction with Principal components to create training dataset
trg <- predict(pc, train)
head(trg)

#prediction with principal components to create testing dataset
tst <- predict(pc, test)
head(tst)
#now with the above components we'll add target variable, species to the train dataset
trg <- data.frame(trg, train[5])
head(trg)
#adding target variable to the test dataset
tst <- data.frame(tst, test[5])
head(tst)
```

```
> #Kaiser's Criterion
> pc$sdev^2
[1] 2.96363071 0.87866793 0.13947996 0.01822139
> trg <- predict(pc, train)
> head(trg)
        PC1          PC2          PC3          PC4
2 -1.980261  0.711941701  0.235638645  0.113500649
4 -2.199130  0.638306376 -0.080593426 -0.054751770
5 -2.311431 -0.580471994  0.007109791 -0.029063873
6 -2.020666 -1.413559178  0.003772130  0.008178089
7 -2.355695  0.005532621 -0.312294066 -0.030839036
9 -2.223785  1.146646098 -0.139052280 -0.015010599
> tst <- predict(pc, test)
> head(tst)
         PC1        PC2         PC3         PC4
1  -2.184131 -0.4180047  0.14483426  0.03145219
3  -2.266899  0.3889209 -0.03210782  0.03719577
8  -2.148401 -0.1667055  0.10405113 -0.01576810
15 -2.143058 -1.7709879  0.49644764  0.19638099
19 -1.841719 -1.3306705  0.39473913  0.06548324
20 -2.276619 -1.0546059 -0.10227132 -0.03408958
> trg <- data.frame(trg, train[5])
> head(trg)
        PC1          PC2          PC3          PC4 Species
2 -1.980261  0.711941701  0.235638645  0.113500649  setosa
4 -2.199130  0.638306376 -0.080593426 -0.054751770  setosa
5 -2.311431 -0.580471994  0.007109791 -0.029063873  setosa
6 -2.020666 -1.413559178  0.003772130  0.008178089  setosa
7 -2.355695  0.005532621 -0.312294066 -0.030839036  setosa
9 -2.223785  1.146646098 -0.139052280 -0.015010599  setosa
> tst <- data.frame(tst, test[5])
> head(tst)
         PC1        PC2         PC3         PC4 Species
1  -2.184131 -0.4180047  0.14483426  0.03145219  setosa
3  -2.266899  0.3889209 -0.03210782  0.03719577  setosa
8  -2.148401 -0.1667055  0.10405113 -0.01576810  setosa
15 -2.143058 -1.7709879  0.49644764  0.19638099  setosa
19 -1.841719 -1.3306705  0.39473913  0.06548324  setosa
20 -2.276619 -1.0546059 -0.10227132 -0.03408958  setosa
```

```
install.packages('nnet')
library(nnet)

#relevel - below code indicates that the reference value is the first Species which is Setosa
trg$Species <- relevel(trg$Species, ref = "setosa")

#multinominal Logistic regression
#we are using first two PCs because they have more than 95% of variability capture
model <- multinom(Species ~ PC1+PC2, data = trg)

#now very quickly model converges and we get some solution
summary(model)
```

```
> library(nnet)
> trg$Species <- relevel(trg$Species, ref = "setosa")
> model <- multinom(Species ~ PC1+PC2, data = trg)
# weights:  12 (6 variable)
initial  value 124.143189
iter  10 value 18.039300
iter  20 value 16.937119
final  value 16.858234
converged
> summary(model)
Call:
multinom(formula = Species ~ PC1 + PC2, data = trg)

Coefficients:
           (Intercept)       PC1      PC2
versicolor   6.9718797  9.756222 3.767176
virginica    0.2414769 15.918514 4.717151

Std. Errors:
           (Intercept)      PC1      PC2
versicolor    38.16614 35.93577 52.88910
virginica     38.20266 35.96631 52.89339

Residual Deviance: 33.71647
AIC: 45.71647
```

6.2.3 confusion matrix & misclassification error

```
#confusion matrix & misclassification error - train
p.trg <- predict(model, trg)
tab <- table(p.trg, trg$Species)
tab

#calculate misclassification error with train
1-sum(diag(tab))/sum(tab)

#confusion matrix & misclassification error - test
p.tst <- predict(model,tst)
tab.tst <- table(p.tst, tst$Species)
tab.tst

#calculate misclassification error with test
1-sum(diag(tab.tst))/sum(tab.tst)
```

```
p.trg        setosa versicolor virginica
  setosa         40          0         0
  versicolor      0         30         4
  virginica       0          4        35
> 1-sum(diag(tab))/sum(tab)
[1] 0.07079646
> p.tst <- predict(model,tst)
> tab.tst <- table(p.tst, tst$Species)
> tab.tst

p.tst        setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         12         1
  virginica       0          4        10
> 1-sum(diag(tab.tst))/sum(tab.tst)
[1] 0.1351351
```

6.2.4 Conclusion

value 4 below virginica - actual value 4 belongs to virginica, but model predicted to versicolor, so this is a misclassification error. value 4 below versicolor - actual value 4 belongs to versicolor, but model predicted to virginica. Misclassification error with train data is about 7.07%. Misclassification error for testing data is about 13.5%.

6.3 EXPERIMENTAL RESULT OF KNN:

```
install.packages('class')
install.packages('ggplot2')
install.packages('GGally')
library(class)
library(ggplot2)
library(GGally)

#summary
summary(iris)
apply(iris[,1:4], 2, sd)
```

```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
> apply(iris[,1:4], 2, sd)
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
   0.8280661    0.4358663    1.7652982    0.7622377
```

6.3.1 Scatter plot

```
#Scatter plot
ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width, col = Species)) +
geom_point()
```

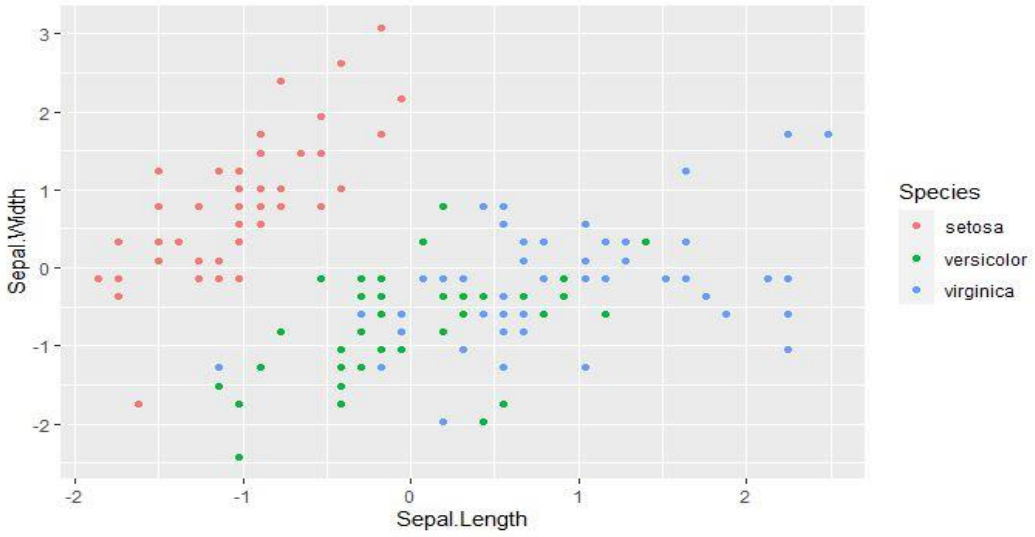


Figure 15 Scatter plot of Sepal in PCA

```
ggplot(data = iris, aes(x = Petal.Length, y = Petal.Width, col = Species)) +
geom_point()
```

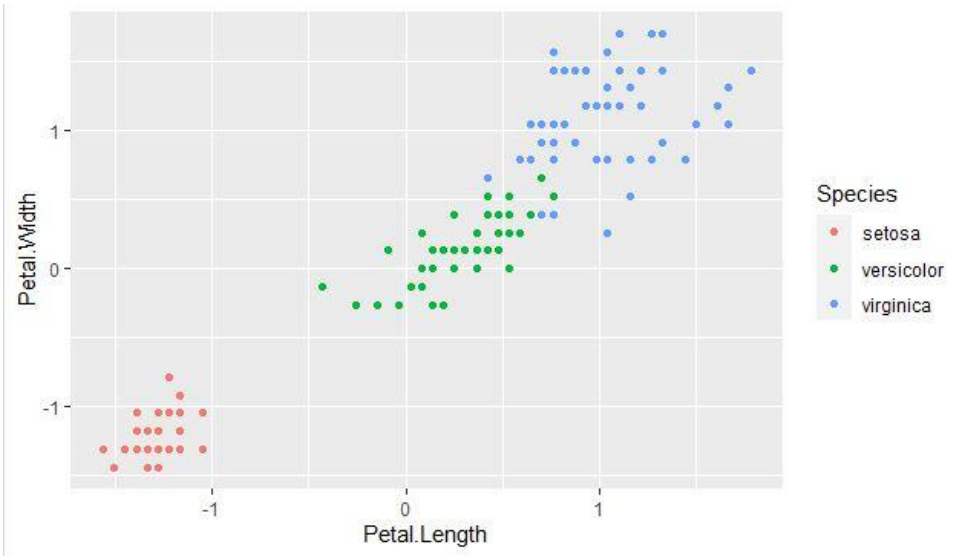


Figure 16 Scatter plot of Petal in PCA

6.3.2 Correlation Matrix

```
#Correlation Matrix
ggpairs(iris)
```

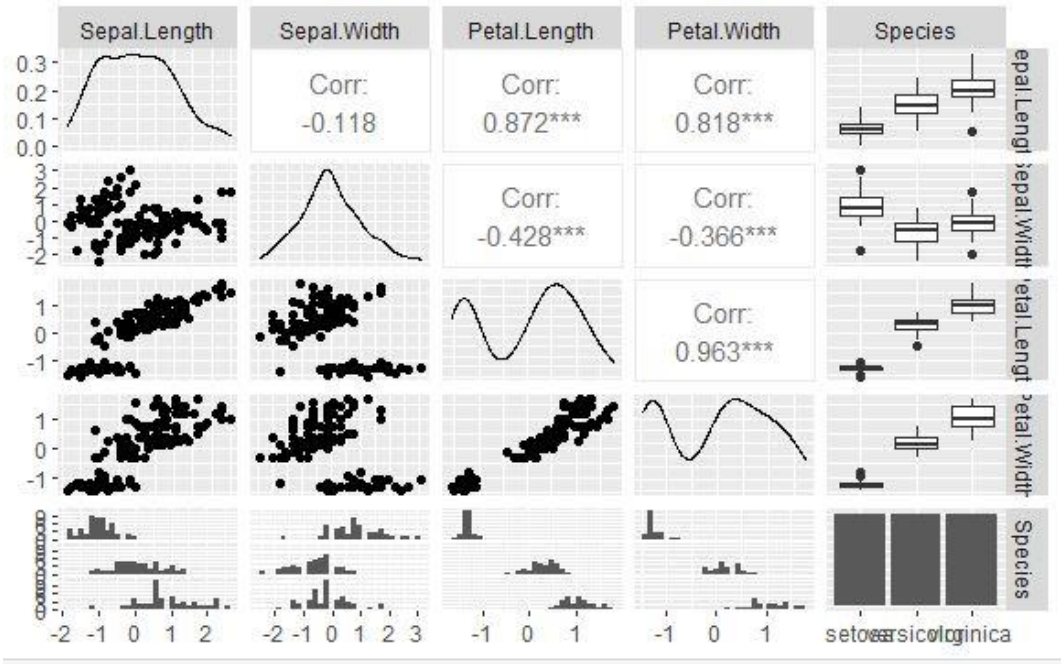


Figure 17 Correlation Matrix of PCA

6.3.3 Splitting the dataset

```
#splitting the dataset
set.seed(12420352)
iris[,1:4] <- scale(iris[,1:4])
setosa<- rbind(iris[iris$Species=="setosa",])
versicolor<- rbind(iris[iris$Species=="versicolor",])
virginica<- rbind(iris[iris$Species=="virginica",])

ind <- sample(1:nrow(setosa), nrow(setosa)*0.8)
iris.train<- rbind(setosa[ind,], versicolor[ind,], virginica[ind,])
iris.test<- rbind(setosa[-ind,], versicolor[-ind,], virginica[-ind,])
iris[,1:4] <- scale(iris[,1:4])
```

Data	
iris	150 obs. of 5 variables
iris.test	30 obs. of 5 variables
iris.train	120 obs. of 5 variables
setosa	50 obs. of 5 variables
versicolor	50 obs. of 5 variables
virginica	50 obs. of 5 variables

```
error <- c()
for (i in 1:15)
{
  knn.fit <- knn(train = iris.train[,1:4], test = iris.test[,1:4], cl = iris.train$Species, k = i)
  error[i] = 1- mean(knn.fit == iris.test$Species)
}
```

Values	
error	num [1:15] 0.0667 0.0667 0.0333 0.0333 0.0333 ...
i	15L
ind	int [1:40] 21 39 16 20 19 4 49 17 3 22 ...
iris_pred	Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 ..
knn.fit	Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 ..

```
ggplot(data = data.frame(error), aes(x = 1:15, y = error)) +
geom_line(color = "Blue")
```

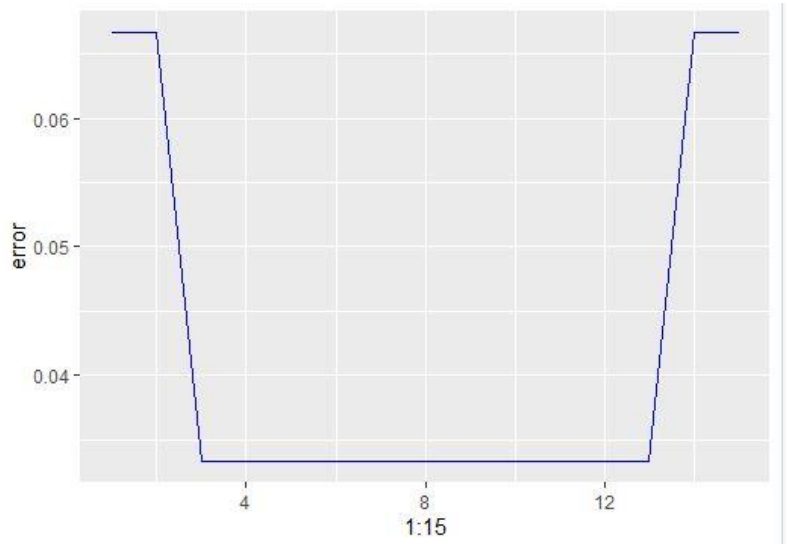


Figure 18 Error plot of PCA

6.3.4 Confusion Matrix

```
#Confusion Matrix
iris_pred <- knn(train = iris.train[,1:4], test = iris.test[,1:4], cl = iris.train$Species, k=5)
table(iris.test$Species,iris_pred)
```

```
            iris_pred
             setosa versicolor virginica
setosa           10          0         0
versicolor        0          9         1
virginica         0          0        10
```

6.3.5 Conclusion

End has been given an accuracy for every variation of our data. It has an overall accuracy of eighty-seven percent to one hundred percent and an overall accuracy of 96.6 percent. I would say that it is very good in the future. - We can use this whole process to make sure that whenever we follow a model that we see fit for the future, we have a good measure of success. We can create a KNN rating that gives a prediction accuracy of 96.67% on the test data set.

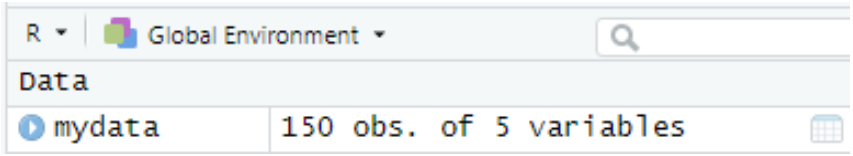
6.4 EXPERIMENTAL RESULT OF RANDOM FOREST

```
#Load required libraries #
library(stats)
library(dplyr)
library(randomForest)

summary(iris)
```

```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

```
# Load data into mydata object #
mydata = iris
```




```
# Inspect mydata #
View(mydata)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa

```
# Variable selection
str(mydata)
```

```
> str(mydata)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

6.4.1 Splitting Data in Training and Testing

```
# Splitting Data in Training and Testing #
# A vector that has random sample of training values #
index = sample(2,nrow(mydata),replace = TRUE,prob=c(0.7,0.3))
```

Data	
mydata	150 obs. of 5 variables

```
# Training data #
Training = mydata[index==1,]
# Testing data #
Testing = mydata[index==2,]
# Random Forest Model #
RFM = randomForest(Species~.,data = Training)
```

RFM	List of 19
Testing	42 obs. of 6 variables
Training	108 obs. of 5 variables

```
# Evaluating Model Accuracy #
Species_Pred = predict(RFM,Testing)
Testing$Species_Pred = Species_Pred
View(Testing)
```

Values	
CFM	'table' int [1:3, 1:3] 16 0 0 0 12 1 0 0 …
index	int [1:150] 2 2 1 2 2 1 1 2 1 1 ...
Species_Pred	Factor w/ 3 levels "setosa","versicolor",…

6.4.2 Confusion Matrix

# Building Confusion Matrix #
CFM = table(Testing$Species,Testing$Species_Pred)
CFM

```
> CFM

            setosa versicolor virginica
  setosa        16          0         0
  versicolor     0         12         0
  virginica      0          1        13
>
```

So,16 Sentosa are correctly classified as Setosa. 12 Versicolor are correctly classified as Versicolor and 1 Versicolor is classified as Virginca. 13 virginica are correctly classified as Virginica.

6.4.3 Random Forest model

# print Random Forest model #
print(RFM)

```
> print(RFM)

Call:
 randomForest(formula = Species ~ ., data = Training)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 7.41%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         34          0         0   0.0000000
versicolor      0         34         4   0.1052632
virginica       0          4        32   0.1111111
```

The number of variables tried at each split is 2. The OOB estimate of error rate is 7.41%.

where Setosa having error is about 0.0000000, versicolor having class error 0.1052632 and

virginica having class error 0.1111111.

6.4.4 Plot Random Forest Model

# Plotting #
plot(RFM)

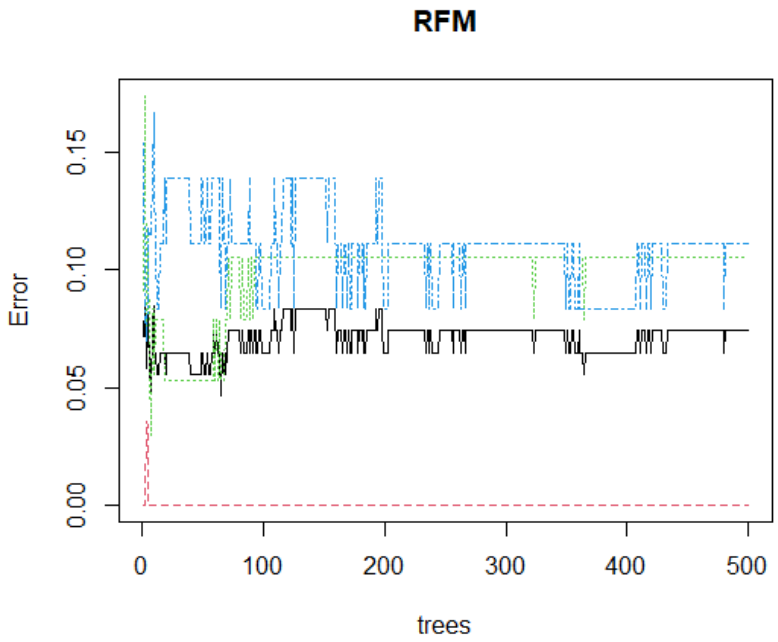


Figure 19: Random Forest Model Plot

# see the importance features
importance(RFM)

```
> importance(RFM)
             MeanDecreaseGini
Sepal.Length         6.725483
Sepal.Width          2.347941
Petal.Length        30.285161
Petal.Width         31.792144
```

varImpPlot(RFM)

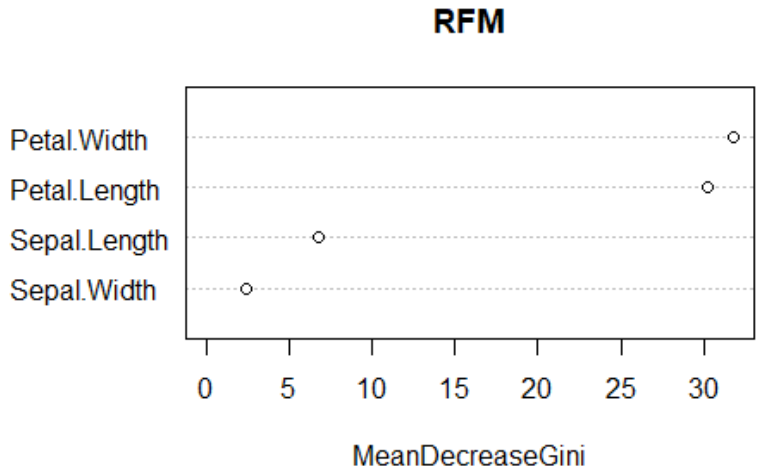


Figure 20: Plot of RFM between petal, sepal, and mean decrease Gini

6.4.5 Model Evaluation and visualization

# Try to see the margin, positive or negative, if positive it means correct classification #
plot(margin(RFM,Testing$Species))

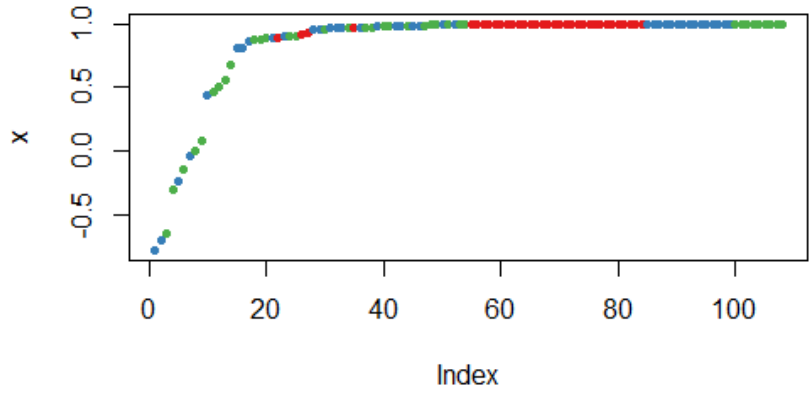


Figure 21: Plot to see the margin between RFM, Testing and species

# Try to tune Random Forest #
tune.rf <- tuneRF(iris[,-5],iris[,5], stepFactor=0.5)

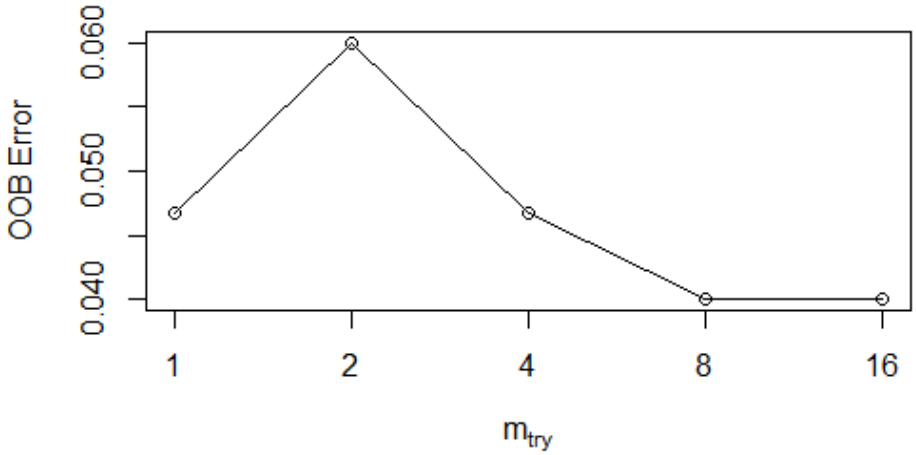


Figure 22: OOB Error in RFM

# Try to tune Random Forest #
tune.rf <- tuneRF(iris[,-5],iris[,5], stepFactor=0.5)

```
> tune.rf <- tuneRF(iris[,-5],iris[,5], stepFactor=0.5)
mtry = 2  OOB error = 6%
Searching left ...
mtry = 4        OOB error = 4.67%
0.2222222 0.05
mtry = 8        OOB error = 4%
0.1428571 0.05
mtry = 16       OOB error = 4%
0 0.05
Searching right ...
mtry = 1        OOB error = 4.67%
-0.1666667 0.05
```

print(tune.rf)

```
> print(tune.rf)
        mtry   OOBError
1.OOB      1 0.04666667
2.OOB      2 0.06000000
4.OOB      4 0.04666667
8.OOB      8 0.04000000
16.OOB    16 0.04000000
```

6.4.6 classification Accuracy

classification_Accuracy = sum(diag(CFM)/sum(CFM))
classification_Accuracy

Values	
CFM	'table' int [1:3, 1:3] 16 0 0 0 12 1 0 0 …
classification…	0.976190476190476

```
> classification_Accuracy
[1] 0.9761905
> 
```

6.4.7 Conclusion

The accuracy of the random forest model designed to predict species using input variables such as sapling length, sapling width, petal length and petal width is approximately 97% which is very good so basic Because we have a high degree of accuracy that we do not need. Go and look at the predictors again or change the predictors or see how we change the variables that we have already achieved high accuracy we can only leave this model.

6.5 RESULT OF DECISION TREE:

The accuracy of the model is better when you cut down the tree. Depending on the model both pre-pruning and subsequent pruning may be required to fit more, prevent less fitting, and fix the model.

6.6 EXPERIMENTAL RESULT OF NAÏVE BAYES:

We have shown the high accuracy that the Naive Bayes algorithm has for classifying text base data, and that it is less accurate on numerical data. The reason I chose the Iris dataset was because I really like working with it and I wanted to try to model a second time with numeric values.

7 REFERENCES

- Published in Towards Data Science. Peter Nistrup. Jan 29, 2019
  From: https://towardsdatascience.com/principal-component-analysis-pca-101-using-r-361f4c53a9ff

- Published in Towards Data Science. Rina Buoy. Jul 28, 2019
  From: https://towardsdatascience.com/introduction-to-principle-component-analysis-d705d27b88b6

- Published in Learn Data Science. Andrea Trevino | December 6, 2016
  From: https://blogs.oracle.com/ai-and-datascience/post/introduction-to-k-means-clustering

- Published in Wikipedia, the free encyclopedia 24 November 2021, at 12:52 (UTC).
  From: https://en.wikipedia.org/wiki/K-means_clustering

- Published in JavaTpoint 2011-2021 www.javatpoint.com. All rights reserved. Developed by JavaTpoint.
- From: https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning

- Published in Wikipedia, the free encyclopedia 20 November 2021, at 15:01 (UTC).
  From: https://en.wikipedia.org/wiki/RStudio

- Published in R Foundation
  From: https://www.r-project.org/about.html

- Published in Builtin. Zakaria Jaadi. April 1, 2021 Updated: December 1, 2021
  From: https://builtin.com/data-science/step-step-explanation-principal-component-analysis

- Published in JavaTpoint 2011-2021 www.javatpoint.com. All rights reserved. Developed by JavaTpoint.
  From: https://www.javatpoint.com/principal-component-analysis

- Published in Towards Data Science. Imad Dabbura. Sep 17, 2018
  From: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a

- Published in Kaggle. UCI Machine Learning. updated 5 years ago
  From: https://www.kaggle.com/uciml/iris

- Published in Data Science Blogathon. Sruthi E R — June 17, 2021
  From: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

- Published in geeksforgeeks. Last Updated : 05 Jun, 2020
  From: https://www.geeksforgeeks.org/random-forest-approach-in-r-programming/

- Published in Wikipedia, the free encyclopedia 12 December 2021, at 03:08 (UTC).
  From: https://en.wikipedia.org/wiki/Random_forest

- Published in Decision Tree Analysis
  From: https://www.omnisci.com/technical-glossary/decision-tree-analysis

- Published in JavaTpoint 2011-2021 www.javatpoint.com. All rights reserved. Developed by JavaTpoint.
  From: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

- Published in machine learning plus. November 4, 2018. Selva Prabhakaran
  From: https://www.machinelearningplus.com/predictive-modeling/how-naive-bayes-algorithm-works-with-example-and-full-code/

- Published in geeksforgeeks. Last Updated : 13 Jul, 2021
  From: https://www.geeksforgeeks.org/naive-bayes-classifier-in-r-programming/

- Published in R · Iris Flower Data Set Cleaned
  From: https://www.kaggle.com/larsen0966/decision-tree-with-the-iris-dataset

- Published in edureka. Zulaikha Lateef. Last updated on May 26,2020
  From: https://www.edureka.co/blog/naive-bayes-in-r/

- Published in Data Science Enthusiast. Nagesh Singh Chauhan.
  From: https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html

- Published in Analytics Vidhya. Sunil Ray — Sep 13th, 2015 and updated on Sept 11th, 2017
  From: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/