## Ex.No.5a Implement SVM Classification Techniques

**Aim**

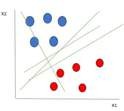
To implement support vector machine (SVM) to find optimum hyper plane

Line in 2D, 3Dhyper plane) which maximize the margin between two classes.

#### Pre-Lab Discussion Theory

**Support Vector Machine(SVM):**

Support Vector Machine(SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well it’s best suited for classification. The objective of the SVM algorithm is to find a hyperplane in an N- dimensional space that distinctly classifies the data points. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three. Let’s consider two independent variables x1, x2, and one dependent variable which is either a blue circle or a red circle.



From the figure above it’s very clear that there are multiple lines (our hyperplane here is a line because we are considering only two input features x1, x2) that segregate our data points or do a classification between red and blue circles.

#### Types of SVM

SVM can be of two types:

* Linear SVM:

Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

* Non-linear SVM:

Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

#### Hyperplane and Support Vectors in the SVM algorithm:

**Hyperplane:** There can be multiple lines/decision boundaries to segregate the classes in n- dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

#### Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

# PROGRAM:

plot(iris) iris

install.packages("e1071")

plot(iris$Sepal.Length, iris$Sepal.width, col=iris$Species) plot(iris$Petal.Length, iris$Petal.width, col=iris$Species) s<-sample(150,100)

col<- c("Petal.Length", "Petal.Width", "Species") iris\_train<- iris[s,col]

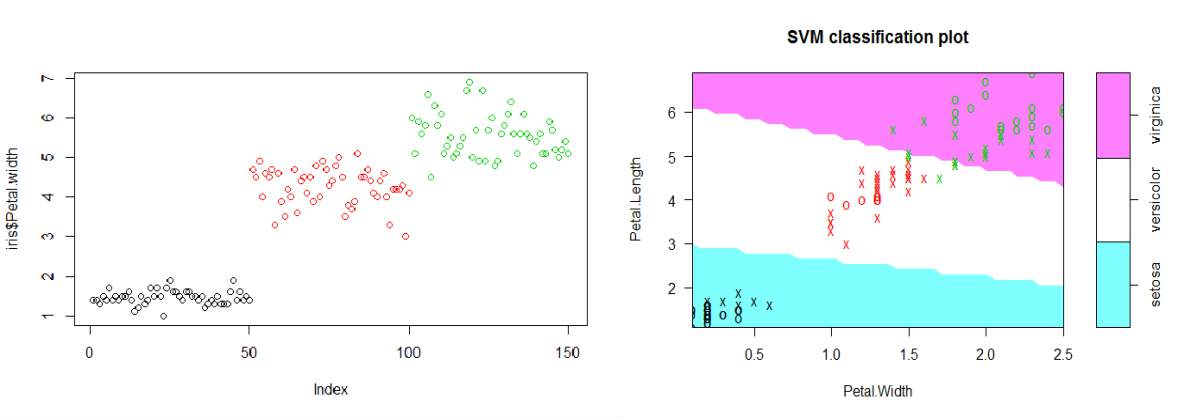
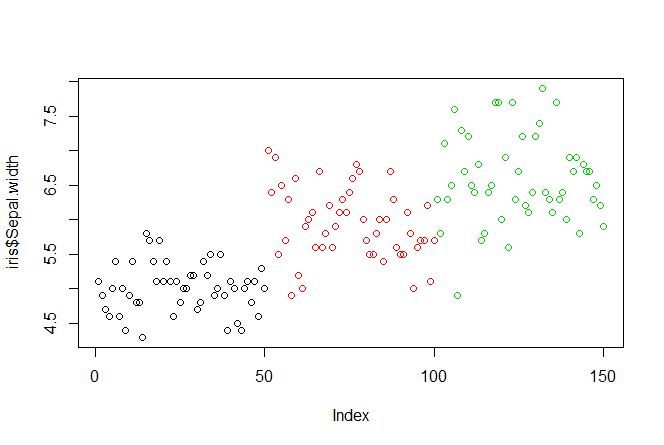
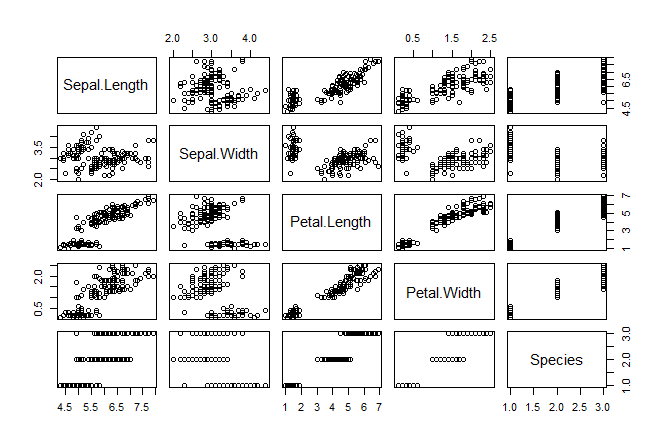
iris\_test<- iris[-s,col]

svmfit<- svm(Species ~., data = iris\_train, kernel = "linear", cost = .1, scale = FALSE) print(svmfit)

plot(svmfit, iris\_train[,col])

tuned <- tune(svm, Species~., data = iris\_train, kernel = "linear", ranges=

list(cost=c(0.001,0.01,.1,.1,10,100)))



summary(tuned)

p<-predict(svmfit, iris\_test[,col], type="class") plot(p)

table(p,iris\_test[,3] ) mean(p== iris\_test[,3])

**OUTPUT:**

## Result

Thus , the implementation of support vector machine (SVM) to find optimum hyper plane (Line in 2D, 3Dhyper plane) which maximize the margin between two classes was executed.

## Ex.No.5b Implement Decision Tree Classification Techniques

### AIM

To implement a decision tree used to representing a decision situation in visually and show all those factors within the analysis that are considered relevant to the decision.

#### Pre-Lab Discussion Theory

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

#### Steps:

Begin the tree with the root node, says S, which contains the complete dataset.

1. Find the best attribute in the dataset using Attribute Selection Measure (ASM).
2. Divide the S into subsets that contains possible values for the best attributes.
3. Generate the decision tree node, which contains the best attribute.
4. Recursively make new decision trees using the subsets of the dataset created in step -

3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

#### Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as Attribute selection measure or ASM. By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

* + Information Gain
  + Gini Index

#### Information Gain:

* Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
* It calculates how much information a feature provides us about a class.
* According to the value of information gain, we split the node and build the decision tree.
* A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first.

#### Gini Index:

* Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
* An attribute with the low Gini index should be preferred as compared to the high Gini index.

# PROGRAM:

library(MASS) library(rpart) head(birthwt) hist(birthwt$bwt) table(birthwt$low)

cols <- c('low', 'race', 'smoke', 'ht', 'ui') birthwt[cols] <- lapply(birthwt[cols], as.factor) set.seed(1)

train<- sample(1:nrow(birthwt), 0.75 \* nrow(birthwt))

birthwtTree<- rpart(low ~ . - bwt, data = birthwt[train, ], method = 'class')

plot(birthwtTree) text(birthwtTree, pretty = 0) summary(birthwtTree)

birthwtPred<- predict(birthwtTree, birthwt[-train, ], type = 'class') table(birthwtPred, birthwt[-train, ]$low)

# OUTPUT:

low age lwt race smoke ptl ht ui ftv bwt

85 0 19 182 2 0 0 0 1 0 2523

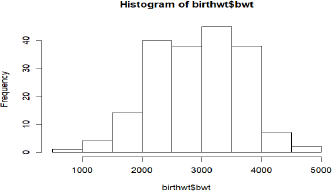
86 0 33 155 3 0 0 0 0 3 2551

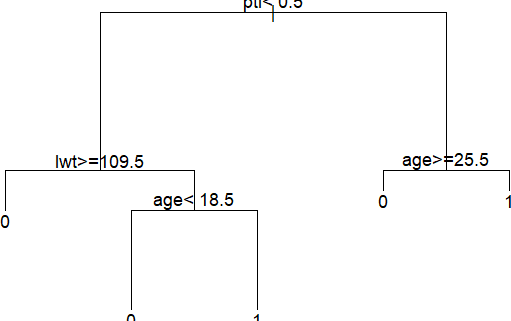
87 0 20 105 1 1 0 0 0 1 2557

88 0 21 108 1 1 0 0 1 2 2594

89 0 18 107 1 1 0 0 1 0 2600

91 0 21 124 3 0 0 0 0 0 2622





## Result

Thus , the implementation of a decision tree used to representing a decision situation in visually and show all those factors within the analysis that are considered relevant to the decision was executed.