project.R

Yash

Sat Dec 08 15:45:17 2018

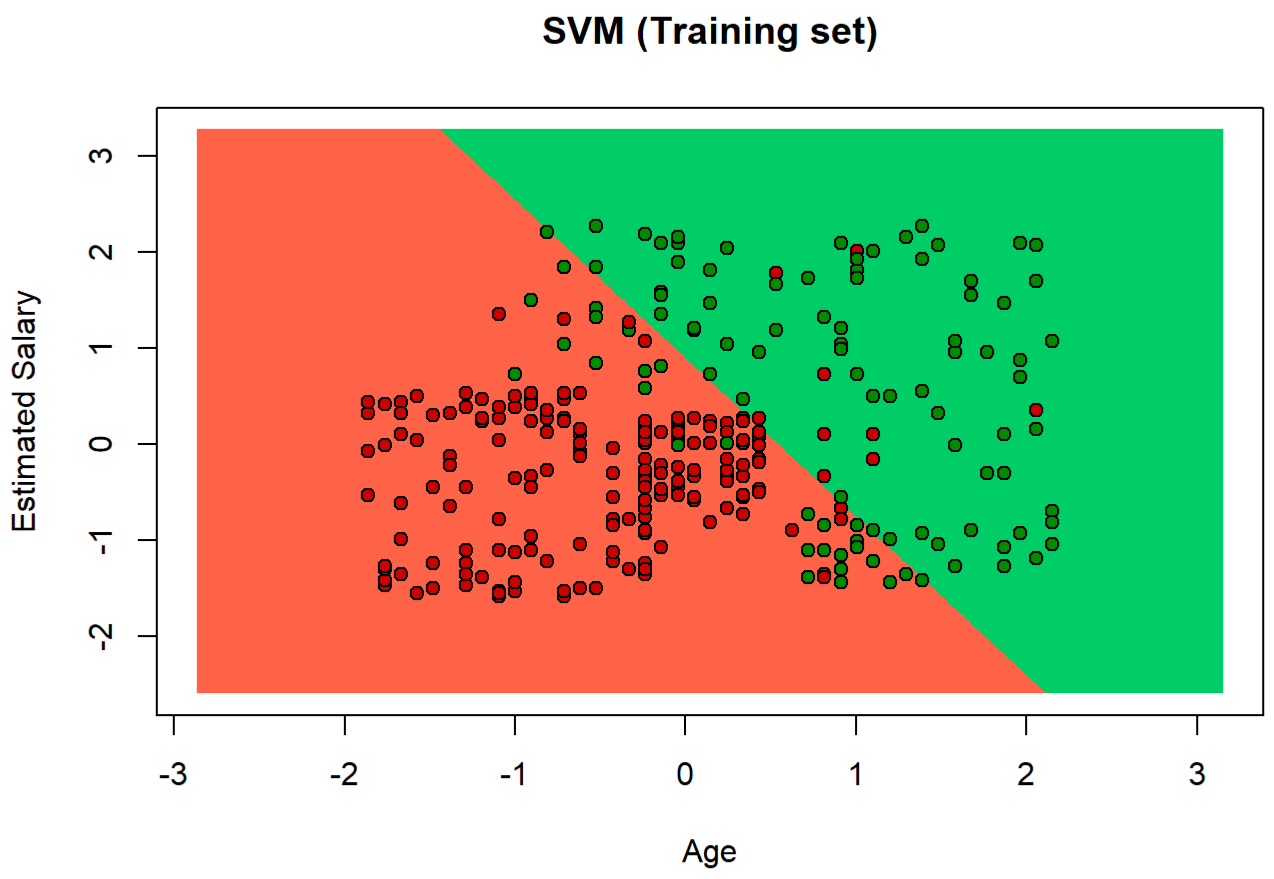
```
#GROUP PROJECT FINAL: SOCIAL NETWORKING ADS



# Importing the dataset
dataset = read.csv('C:/Users/Yash/Downloads/Social_Network_Ads.csv')
# First we have to preprocess data
#Selecting the index which is 3,4 and 5 that is age,estimated salary and purchase od the car

dataset = dataset[3:5]

# Encoding the target feature as factor
dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1))

# Splitting the dataset into the Training set and Test set
# install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(dataset$Purchased, SplitRatio = 0.75)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
#we take the split ratio as 0.75 which means we take 300 dataset as training set and 100 as test
 set
# Feature Scaling
#It is important to select feature scaling in classification
#We remove variable 3 which is dependent variable that is the purchase
#After doing this the age and estimated salary is perfectly scaled
training_set[-3] = scale(training_set[-3])
test_set[-3] = scale(test_set[-3])


# Fitting classifier to the Training set
# Create your classifier here
#downloads  the package 'e1071' from the library
#SVM
library(e1071)
#creates the "SVM" classifier here
#takes the most basic "linear" kernel type


classifier = svm(formula = Purchased ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'linear')
# Predicting the Test set results
y_pred = predict(classifier, newdata = test_set[-3])

#we make new predictions based on whether user buys car or not
#the last column shows this result
# we have incorrect predictions as well

# predicts if a user brought a car or not from the two sets of data.
# Making the Confusion Matrix
# we find out the number of incorrect predictions by using the matrix

# Making the Confusion Matrix
cm = table(test_set[, 3], y_pred)
cm
```

```
##    y_pred
##      0  1
##   0 57  7
##   1 13 23
```

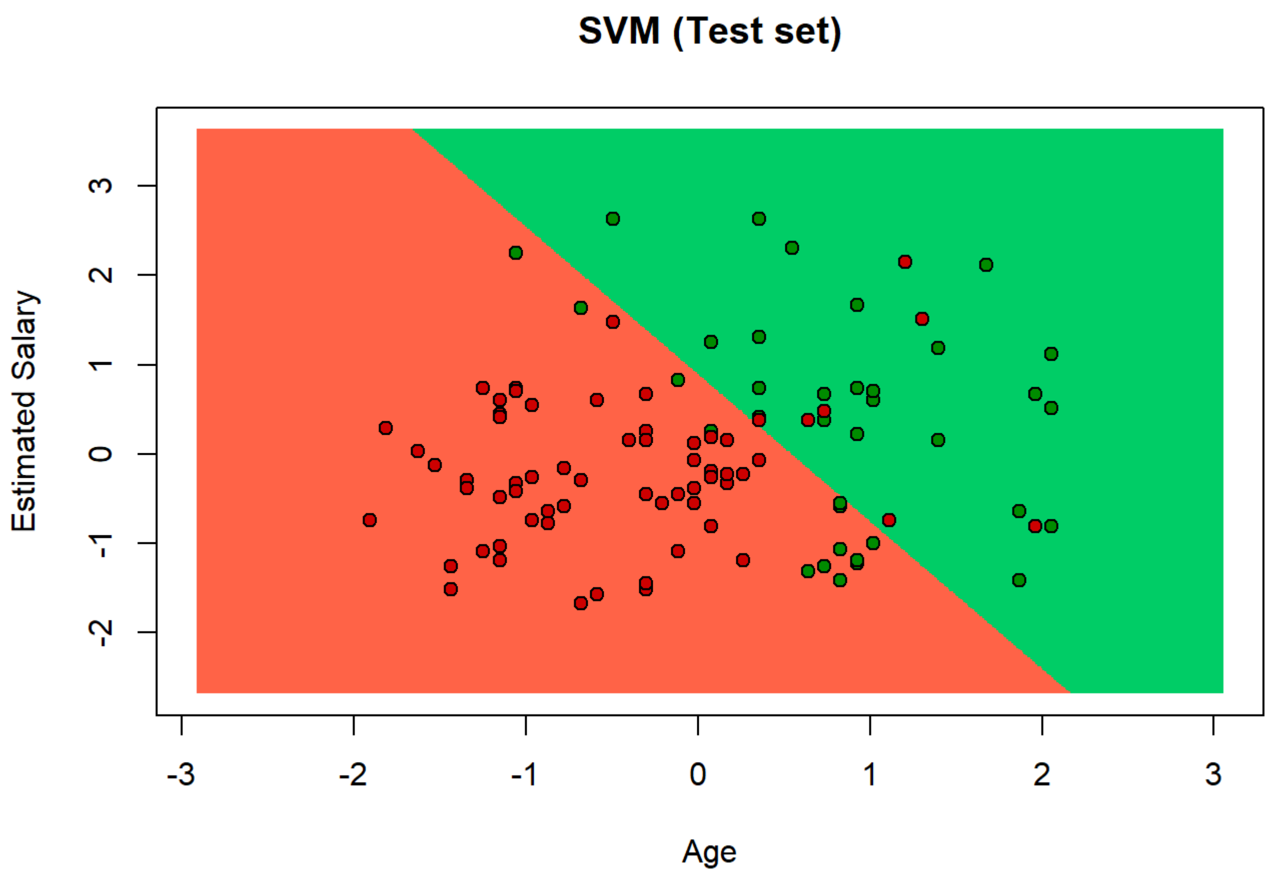
```
(57+23)/(57+7+13+23)
```

```
## [1] 0.8
```

```
# Visualising the Training set results
library(ElemStatLearn)
set = training_set
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
y_grid = predict(classifier, newdata = grid_set)
#Specifies the title classifier as "SVM" for training set
#plots the estimated salary vs age graph for the SVM training set

plot(set[, -3],
     main = 'SVM (Training set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))
#makes a contour line on the above graph
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
#divides the two sets of points using colors
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```



```
#the points in green are where the users brought the SUV car in reality
#some points fall under green because in linear classification , line cannot be a curve
# Visualising the Test set results
library(ElemStatLearn)
set = test_set
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
y_grid = predict(classifier, newdata = grid_set)
#changes the title to SVM for the test set classifier
#some users who bought the car fell into the red region as it is not a curve but a straight line

plot(set[, -3], main = 'SVM (Test set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```



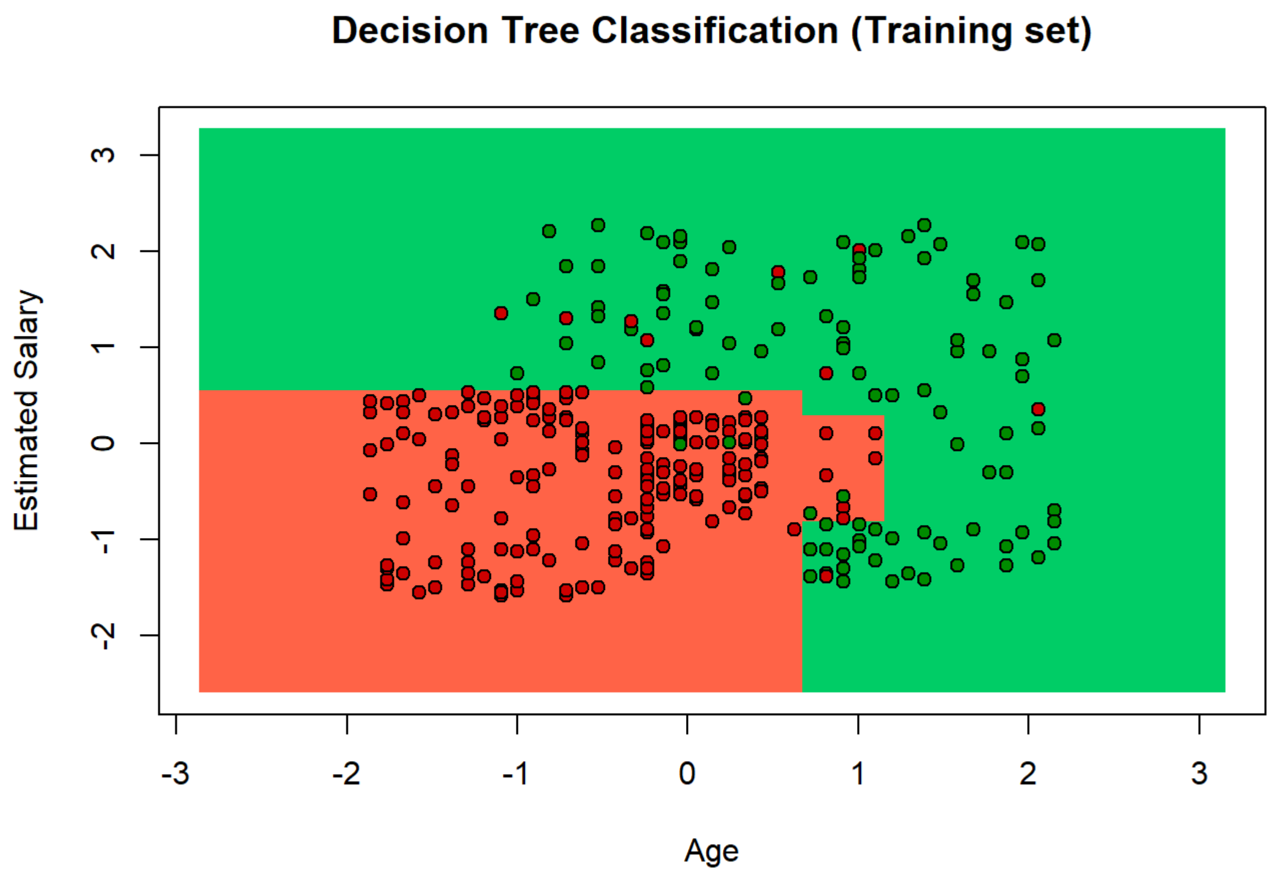
```
# Decision Tree Classification
# Fitting Decision Tree Classification to the Training set
#downloads the package "rpart" from the library
# install.packages('rpart')
library(rpart)
#we pick our training set from the data and we use independent variable purchased
classifier = rpart(formula = Purchased ~ .,
                   data = training_set)

# Predicting the Test set results
#gives a matrix of 2 columns and 100 lines
#gives the probabilities whether the user buys the car or not
y_pred = predict(classifier, newdata = test_set[-3], type = 'class')
#this prediction is correct
#we have the prediction 0 or 1 according to whether the user buys or not

#we have 17 incorrect predictions
# Making the Confusion Matrix
cm = table(test_set[, 3], y_pred)
cm
```

```
##    y_pred
##      0  1
##   0 53 11
##   1  6 30
```

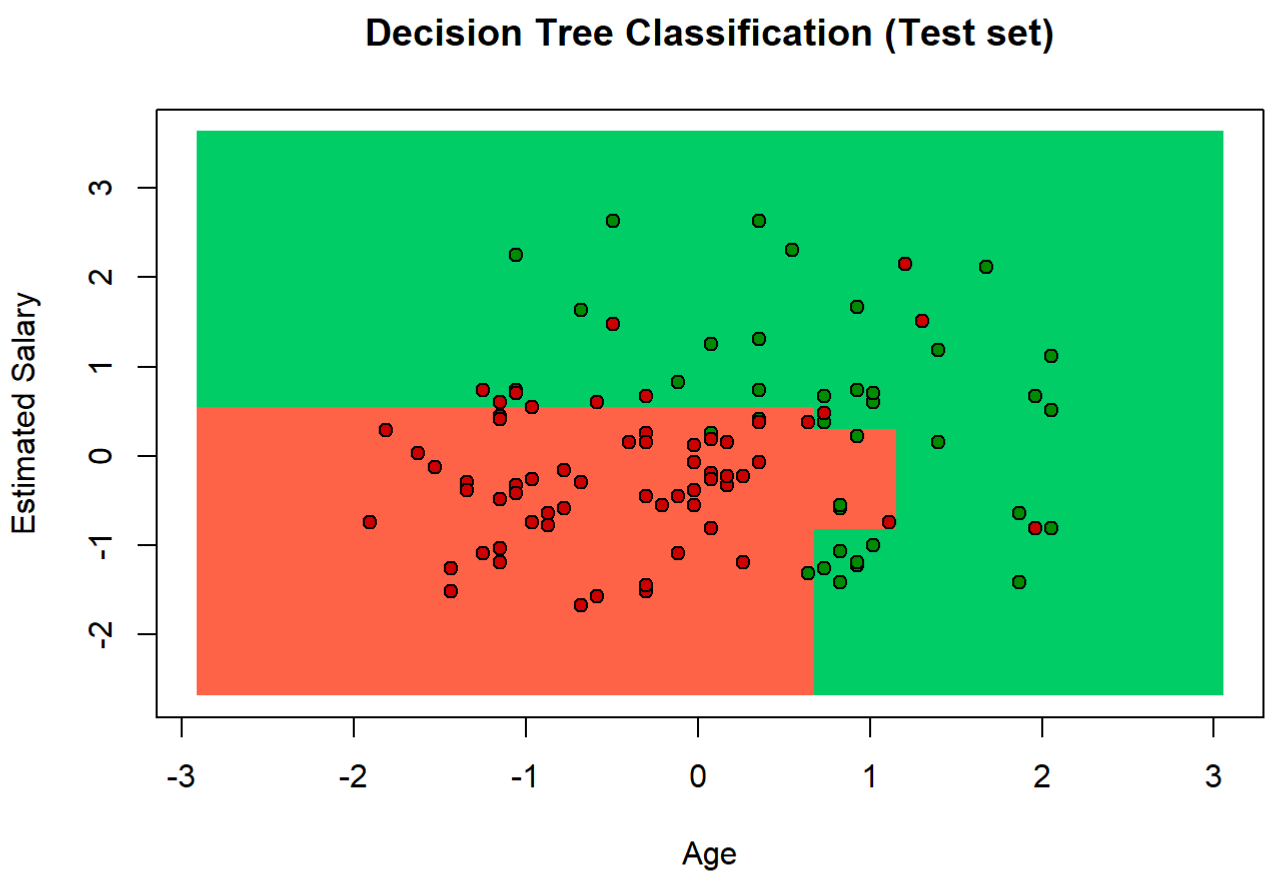
```
(53+30)/(53+11+6+30)
```

```
## [1] 0.83
```

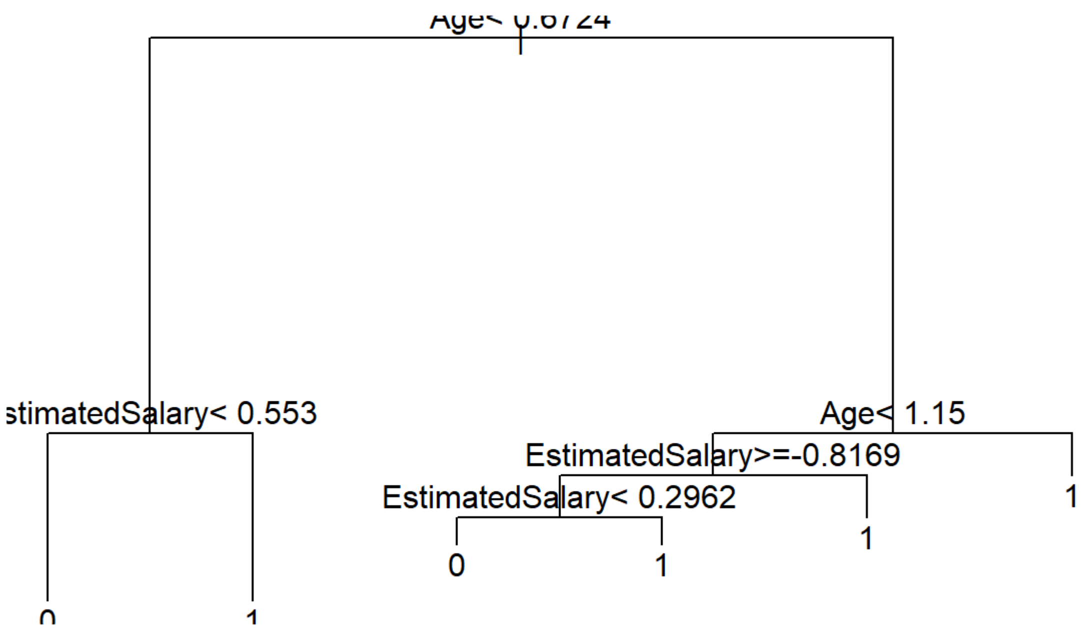
```
# Visualising the Training set results
library(ElemStatLearn)
set = training_set
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
y_grid = predict(classifier, newdata = grid_set, type = 'class')
#plots the decision tree classification of the training set using age vs estimated salary
plot(set[, -3],
     main = 'Decision Tree Classification (Training set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))
#plots a contour line to separate the points
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```



```
# Visualising the Test set results
#no overfitting in the test set results
#red region has 17 incorrect predictions

library(ElemStatLearn)
set = test_set
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
y_grid = predict(classifier, newdata = grid_set, type = 'class')
plot(set[, -3], main = 'Decision Tree Classification (Test set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```



```
# Plotting the decision tree
#plot the tree without labels

# Plotting the tree
plot(classifier)
#plots explicitly the tree
text(classifier)
```



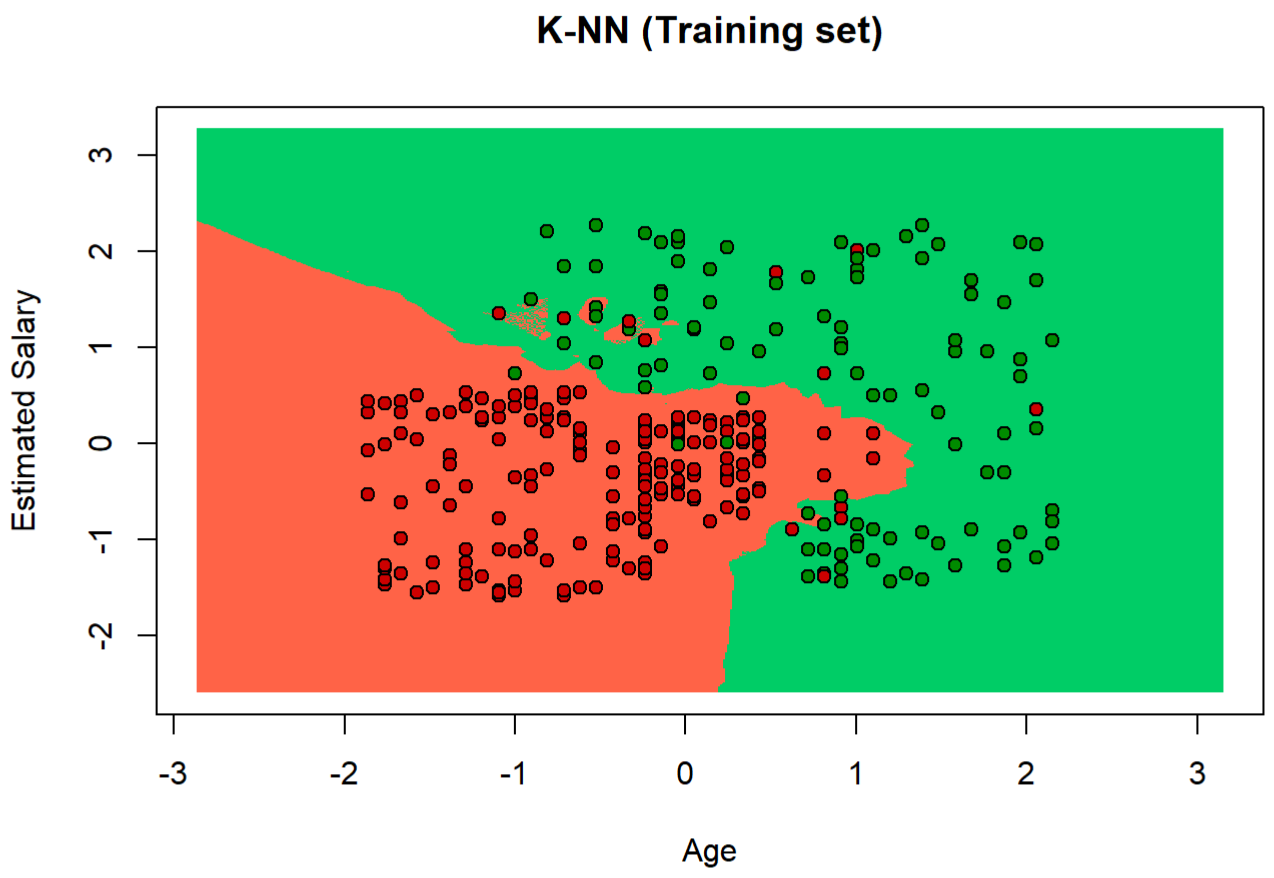
```
#if age <44 .5 yrs, est. salary < 90k$ , then this user wont buy the SUV car.
#if est. salary>90k$ , then this user will buy the SUV car, since result is 1

# K-Nearest Neighbors (K-NN)
# Fitting K-NN to the Training set and Predicting the Test set results
#downloads the package 'class' from the library
library(class)
#fitting knn to the training set and predicting the test set results
#knn function will return the predictions
y_pred = knn(train = training_set[, -3],
             test = test_set[, -3],
             cl = training_set[, 3],
             k = 5,
             prob = TRUE)
#we take 5 neighbors and first 5 users have correct predictions
# Making the Confusion Matrix
cm = table(test_set[, 3], y_pred)
cm
```

```
##    y_pred
##      0  1
##   0 59  5
##   1  6 30
```

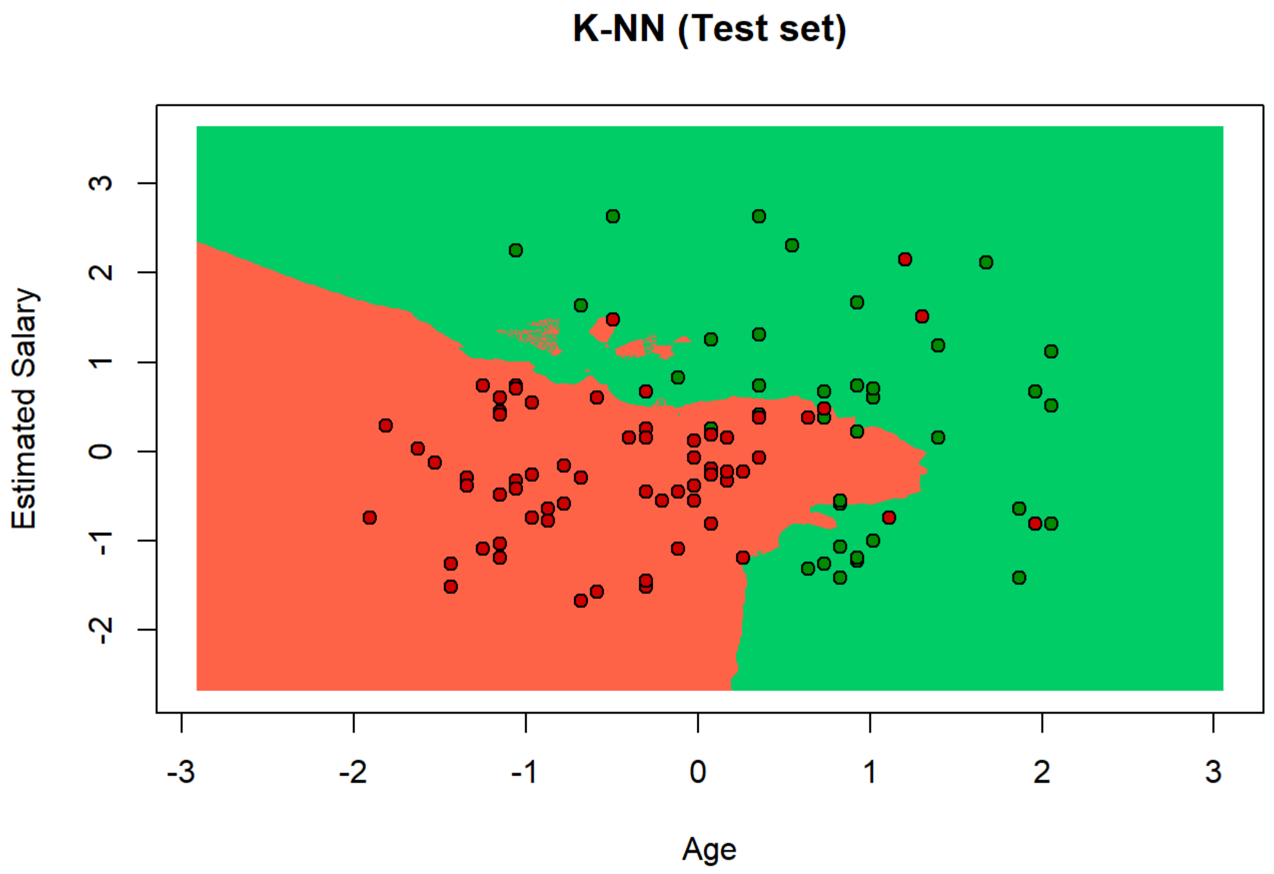
```
(59+30)/(59+30+6+5)
```

```
## [1] 0.89
```

```
# Visualising the Training set results
library(ElemStatLearn)
set = training_set
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
y_grid = knn(train = training_set[, -3], test = grid_set, cl = training_set[, 3], k = 5)
#plots the graph age vs estimated salary for the KNN Training set
plot(set[, -3],
     main = 'K-NN (Training set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))
#plots a contour line to separate the points in red and green region
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```



```
# Visualising the Test set results
library(ElemStatLearn)
set = test_set
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
y_grid = knn(train = training_set[, -3], test = grid_set, cl = training_set[, 3], k = 5)
plot(set[, -3],
     main = 'K-NN (Test set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```



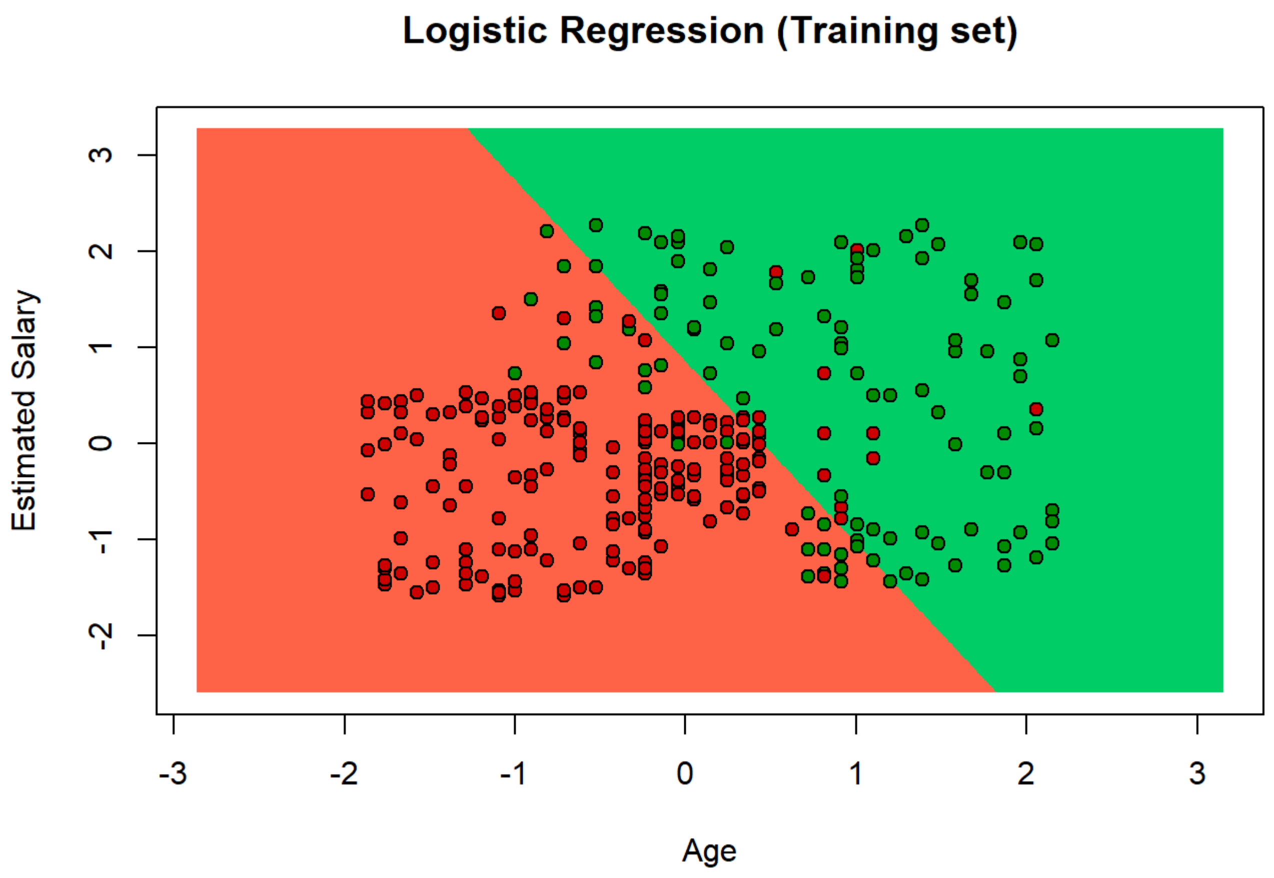
```
#the green region indicates that Suv is bought by the users
#the red region indiactes that the SUV is not bought by the users
#knn does a much better than a logistic regression model as the contour line is not linear but a
 curve , so it does differentiate the areas properly


# Logistic Regression

# Fitting Logistic Regression to the Training set
# creating our classifier we us gml function which is going to build our logistic classifier
# gml is generalized linear model thats because logistic regression is linear model classifier


classifier = glm(formula = Purchased ~ .,
                 family = binomial,
                 data = training_set)
# Predicting the Test set results by using the classifier we just build above
# vector of the predicted value of our test set observation, from our  gml classifiers
# we take 2 independent variables in this case


# Predicting the Test set results
prob_pred = predict(classifier, type = 'response', newdata = test_set[-3])
#transform the probabilities into 0 and 1's using if else function
y_pred = ifelse(prob_pred > 0.5, 1, 0)

# Making the Confusion Matrix
cm = table(test_set[, 3], y_pred > 0.5)
cm
```

```
##
##      FALSE TRUE
##    0    57    7
##    1    10   26
```

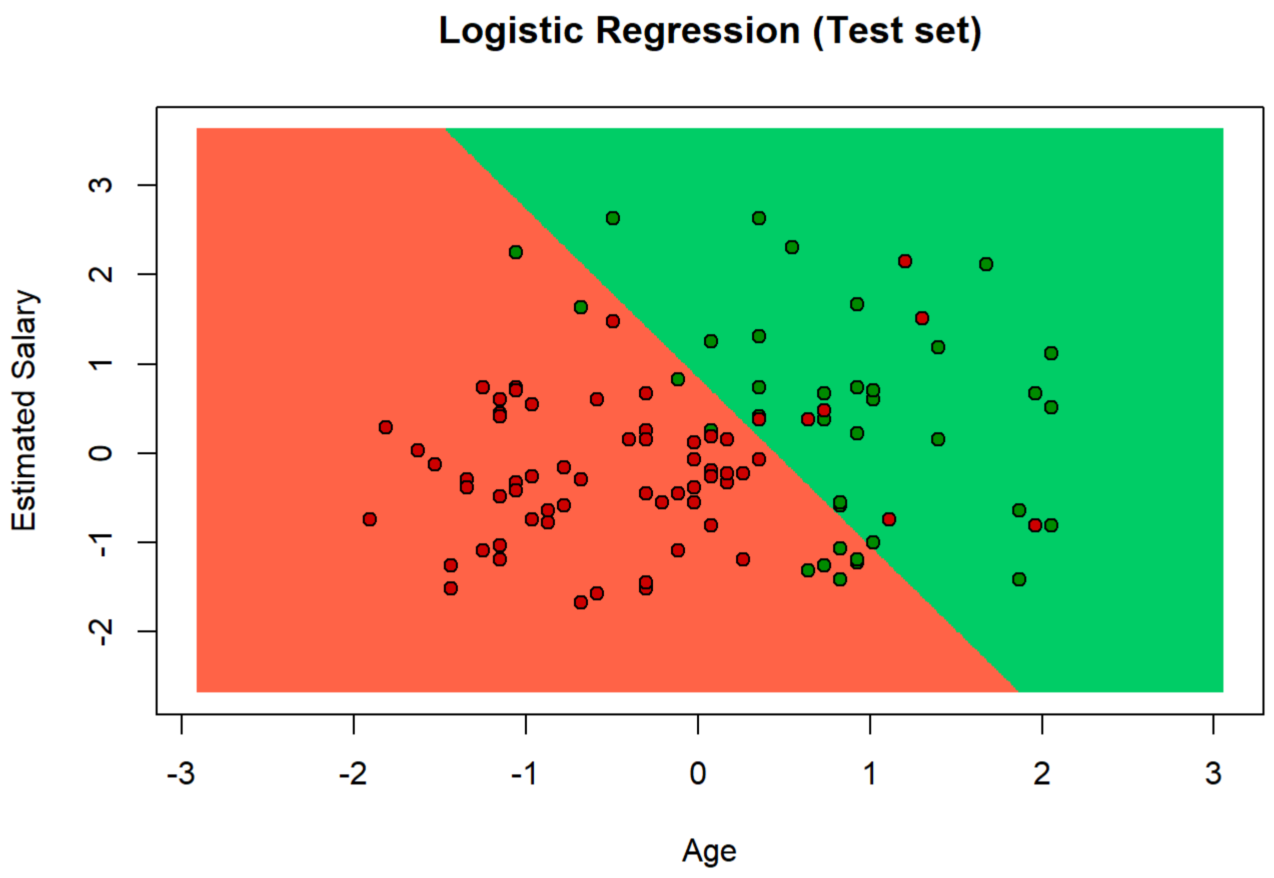
```
(57+26)/(57+7+10+26)
```

```
## [1] 0.83
```

```
# Visualising the Training set results
library(ElemStatLearn)
set = training_set
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
prob_set = predict(classifier, type = 'response', newdata = grid_set)
y_grid = ifelse(prob_set > 0.5, 1, 0)
plot(set[, -3],
     main = 'Logistic Regression (Training set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```



```
# Visualising the Test set results
library(ElemStatLearn)
set = test_set
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
prob_set = predict(classifier, type = 'response', newdata = grid_set)
y_grid = ifelse(prob_set > 0.5, 1, 0)
plot(set[, -3],
     main = 'Logistic Regression (Test set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```



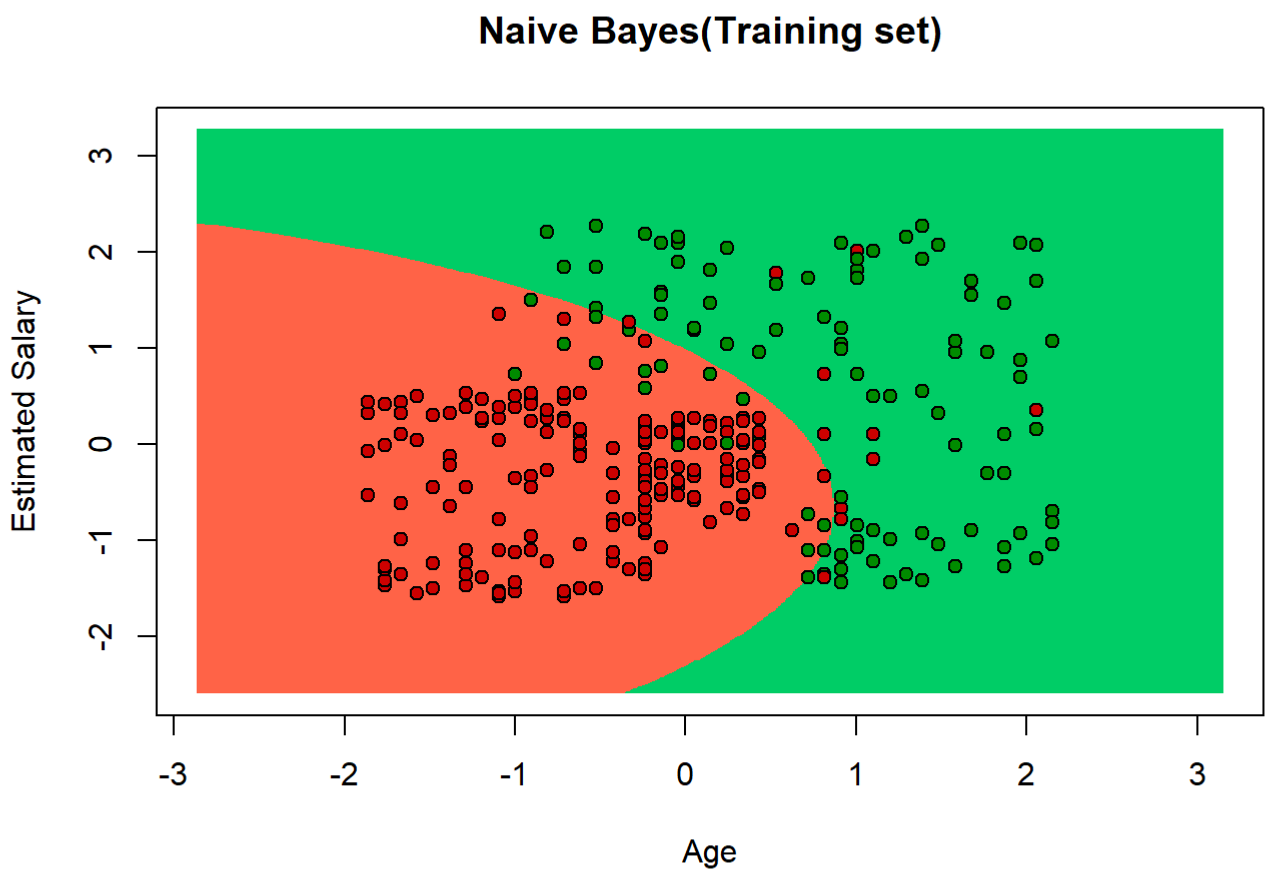
```
# Naive Bayes
# Fitting Naive Bayes to the Training set
# install.packages('e1071')
#downloads the package 'e1071' from the library
library(e1071)
classifier = naiveBayes(x = training_set[-3],
                        y = training_set$Purchased)

# Predicting the Test set results
#Fitting naive bayes to the training set
#create the predictor using 'y_pred'

y_pred = predict(classifier, newdata = test_set[-3])

# Making the Confusion Matrix
cm = table(test_set[, 3], y_pred)
cm
```

```
##    y_pred
##      0  1
##   0 57  7
##   1  7 29
```

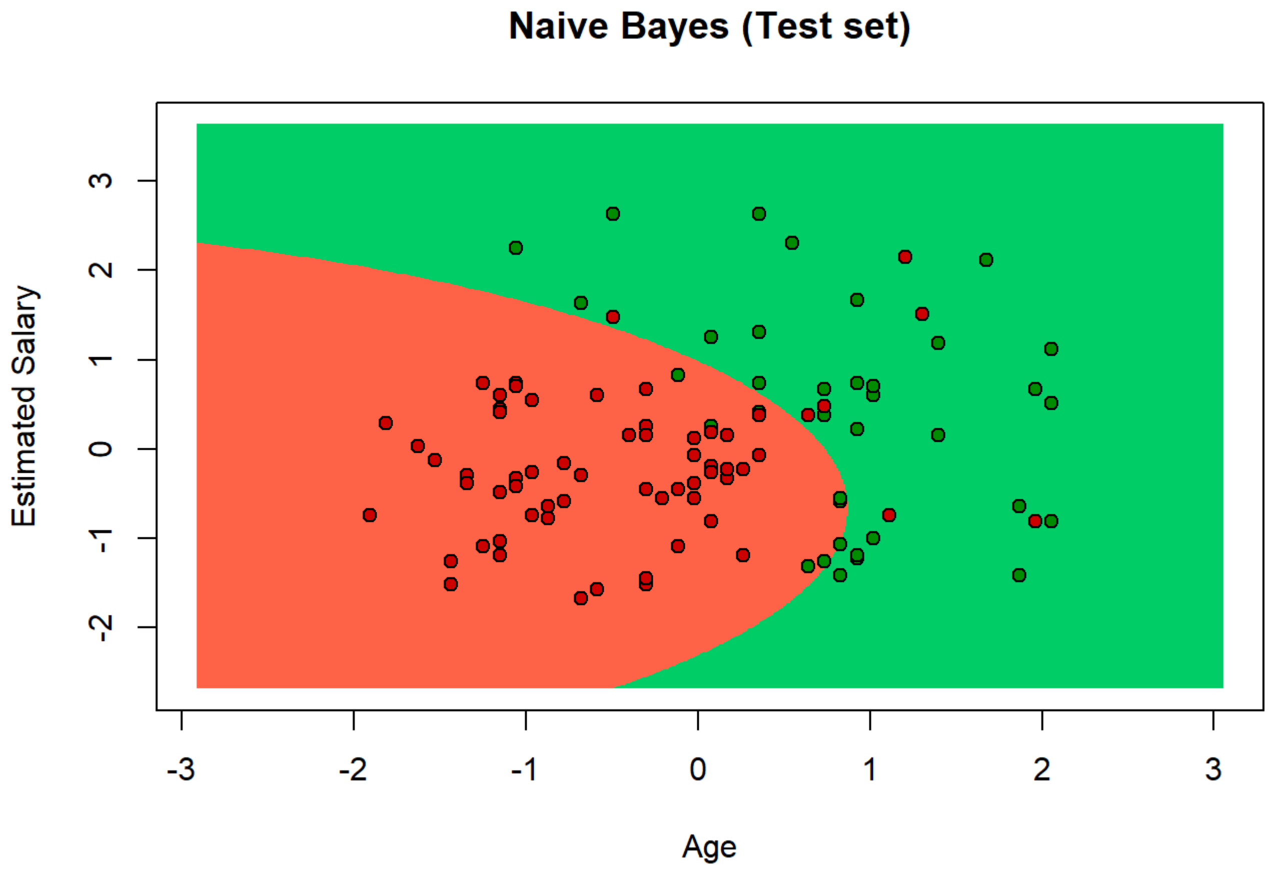
```
(57+29)/(57+29+7+7)
```

```
## [1] 0.86
```

```
# Visualising the Training set results
library(ElemStatLearn)
set = training_set
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
y_grid = predict(classifier, newdata = grid_set)
#plots a graph showing the age vs estimated salary for the Naive Bayes training set
plot(set[, -3],
     main = 'Naive Bayes(Training set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))
#draws a parabolic curve contour to separate the red and green color regions
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```



```
# Visualising the Test set results
library(ElemStatLearn)
set = test_set
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
y_grid = predict(classifier, newdata = grid_set)
#plots the smooth curve graph to show the green users and red users
#shows less incorrect predictions than other methods
plot(set[, -3], main = 'Naive Bayes (Test set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```



```
#we have a high resolution here and a good job is done to classify the points

# Random Forest Classification
# Fitting Random Forest Classification to the Training set
# install.packages('randomForest')
#using random forest library for execution of classifier
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

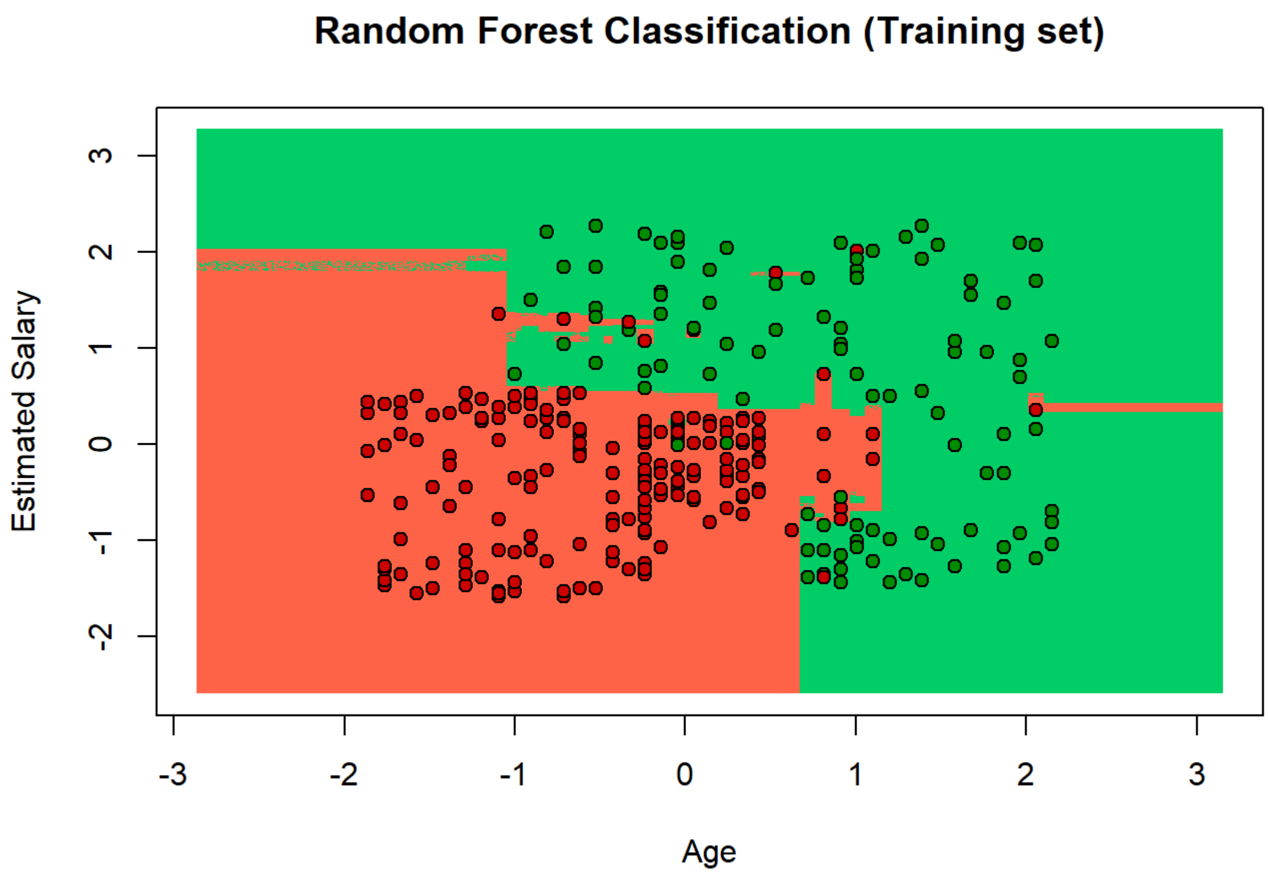
```
set.seed(123)
classifier = randomForest(x = training_set[-3],
                          y = training_set$Purchased,
                          ntree = 50)

# Predicting the Test set results
#predcition of test set results as y_pred gives you the values
y_pred = predict(classifier, newdata = test_set[-3])

# Making the Confusion Matrix
#confusion martix used to show incorrect results we have
cm = table(test_set[, 3], y_pred)
cm
```

```
##    y_pred
##      0  1
##   0 54 10
##   1  7 29
```

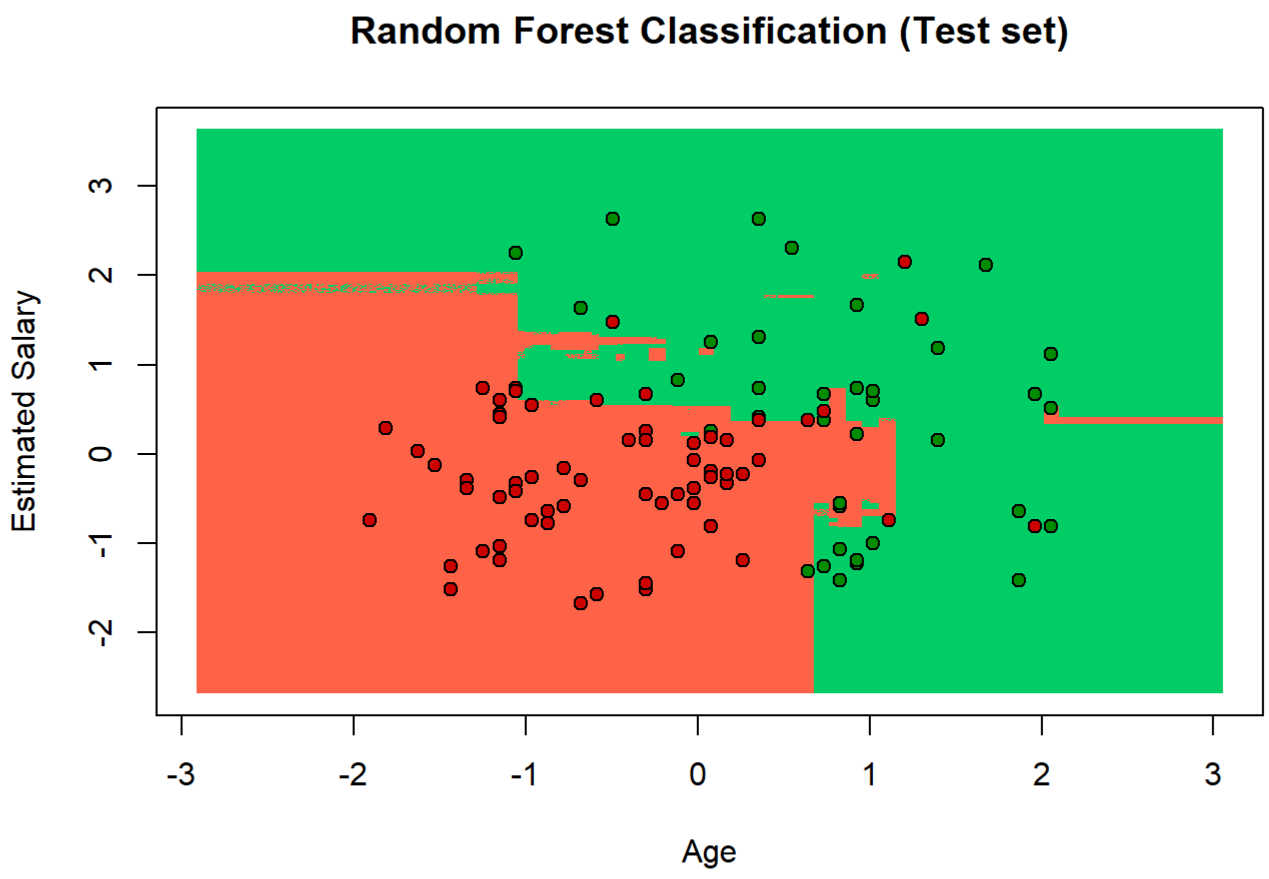
```
(56+29)/(56+29+7+8)
```

```
## [1] 0.85
```

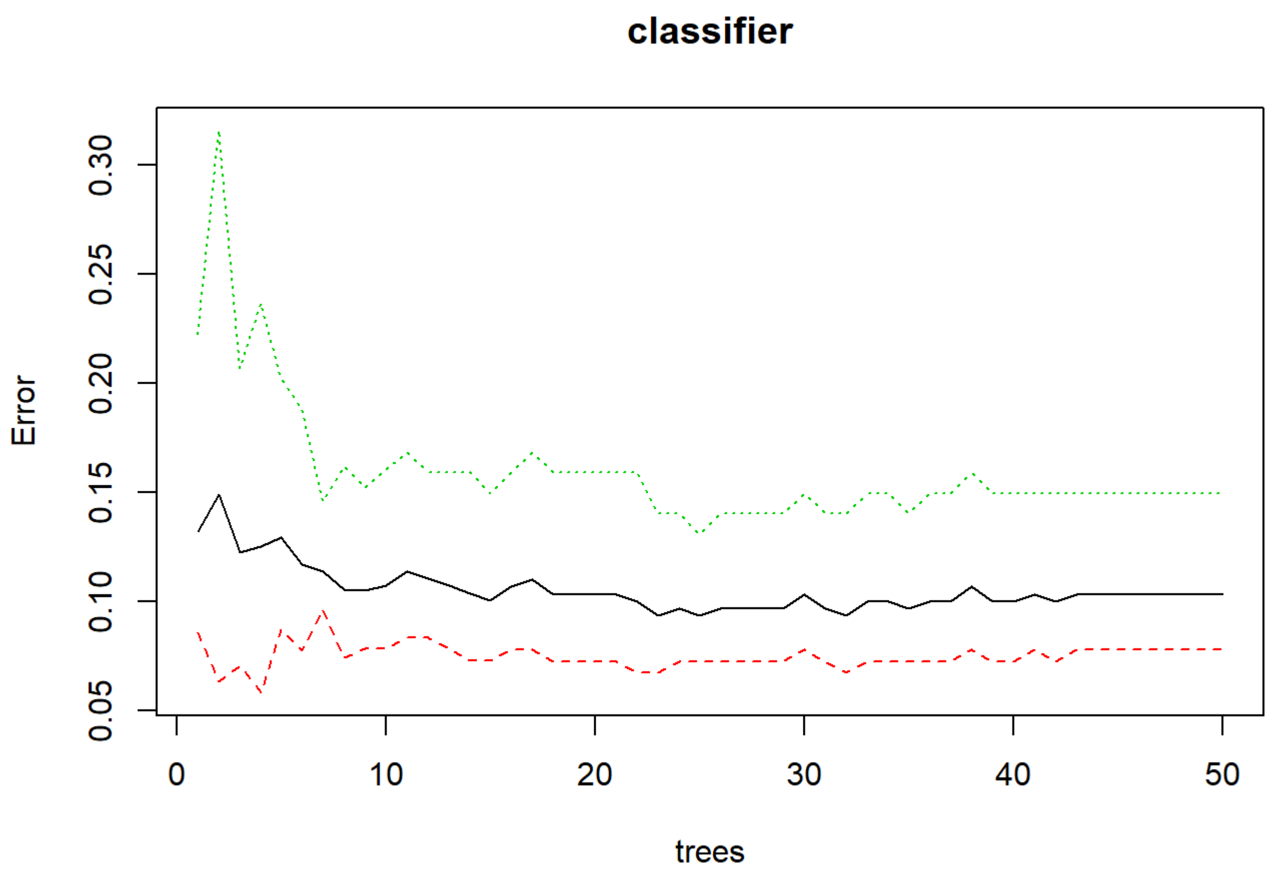
```
# Visualising the Training set results
#gives you the results by plotting red and green region where dots displays
#regions whether they have bought it through social network or no

library(ElemStatLearn)
set = training_set
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
y_grid = predict(classifier, grid_set)
plot(set[, -3],
     main = 'Random Forest Classification (Training set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```



```
# Visualising the Test set results
#execution of classifier on testing data set
library(ElemStatLearn)
set = test_set
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
y_grid = predict(classifier, grid_set)
plot(set[, -3], main = 'Random Forest Classification (Test set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```



```
# Choosing the number of trees
plot(classifier)
```


classifier
Error
0.05
0.10
0.15
0.20
0.25
0.30
0
10
20
30
40
50
trees