CLASSIFICATION TECHNIQUES IN R

Ashwin Bharathwaj Sekar

This project uses a public data set containing information from social marketing agency on the people they targeted. There are features such as Gender, Age & Estimated salary along with an indicator attribute on whether they purchased an item

To explore how different classification algorithms work, I have used only two variables (Age & Estimated Salary) to predict if a particular customer will purchase the item. This will enable an easier way to visualize how the algorithm creates the prediction boundaries.

PRE PROCESSING

```
library(caTools)
library(ggplot2)
library(ggpubr)
library(ElemStatLearn)
library(readr)

# Loading the dataset
Social_Network_Ads <- read_csv("Social_Network_Ads.csv")
dataset=Social_Network_Ads[,3:5]
dataset$Purchased=factor(dataset$Purchased,levels=c(0 , 1))
print(head(dataset,10))
```

```
## # A tibble: 10 x 3
##      Age EstimatedSalary Purchased
##    <dbl>           <dbl> <fct>
##  1    19           19000 0
##  2    35           20000 0
##  3    26           43000 0
##  4    27           57000 0
##  5    19           76000 0
##  6    27           58000 0
##  7    27           84000 0
##  8    32          150000 1
##  9    25           33000 0
## 10    35           65000 0
```

```
set.seed(123)
# Splitting into training and test datasets
training_set=subset(dataset,sample.split(dataset$Purchased,SplitRatio = 0.75)==TRUE)
test_set=subset(dataset,sample.split(dataset$Purchased,SplitRatio = 0.75)==FALSE)
# Standardization
training_set[,1:2]=scale(training_set[,1:2])
test_set[,1:2]=scale(test_set[,1:2])
```

LOGISTIC REGRESSION

In Logistic regression, the output of the lm function is predicted probability, which has to be converted to the binary output using an appropriate threshold. We will use the ROC curve to determine this optimal threshold, but here I’ve used 0.5 as an example value.

```
# Creating the Logit regressor model
logreg=glm(formula = Purchased~Age+EstimatedSalary, data=training_set, family='binomial')
summary(logreg)
```

```
## 
## Call:
## glm(formula = Purchased ~ Age + EstimatedSalary, family = "binomial", 
##     data = training_set)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -3.0753  -0.5235  -0.1161   0.3224   2.3977  
## 
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)    
## (Intercept)      -1.1923     0.2018  -5.908 3.47e-09 ***
## Age               2.6324     0.3461   7.606 2.83e-14 ***
## EstimatedSalary   1.3947     0.2326   5.996 2.03e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 390.89  on 299  degrees of freedom
## Residual deviance: 199.78  on 297  degrees of freedom
## AIC: 205.78
## 
## Number of Fisher Scoring iterations: 6
```

```
# Predicting the target variable for test & training datasets
training_set$log_prob=predict(logreg,type='response',newdata = training_set[,c('Age', 'Estima
tedSalary')])
training_set$log_pred=as.numeric(ifelse(training_set$log_prob>0.5,1,0))

test_set$log_prob=predict(logreg,type='response',newdata = test_set[,c('Age', 'EstimatedSalar
y')])
test_set$log_pred=as.numeric(ifelse(test_set$log_prob>0.5,1,0))
```

Visualizing the results

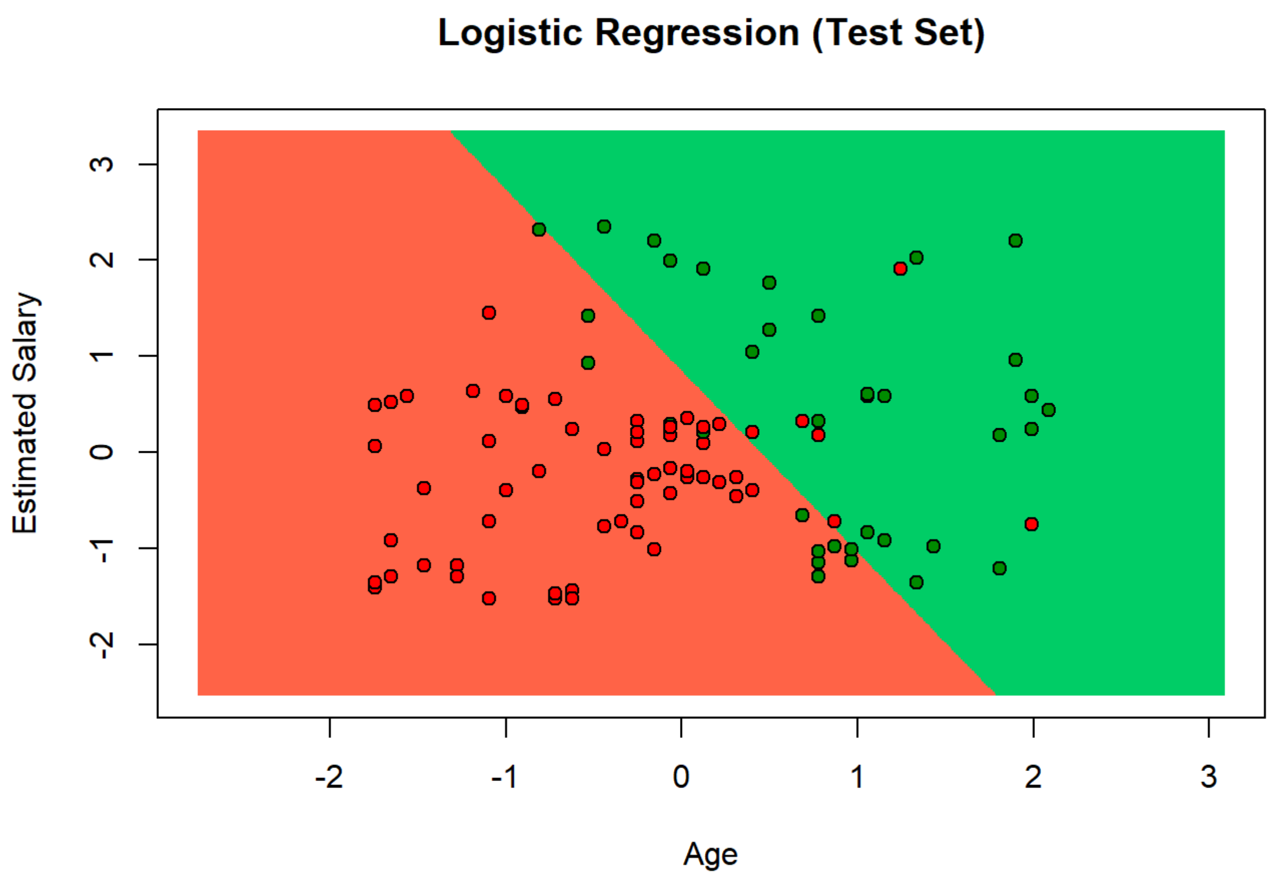
```r
# Confusion Matrix
cm_train=table(training_set$Purchased,training_set$log_pred)
cm_test=table(test_set$Purchased,test_set$log_pred)

# Creating a dummy sequence to visualize the prediction boundaries
set=test_set
X1=seq(min(set$Age)-1,max(set$Age)+1,by=0.01)
X2=seq(min(set$EstimatedSalary)-1,max(set$EstimatedSalary)+1,by=0.01)
grid_set=expand.grid(X1,X2)
colnames(grid_set)=c('Age','EstimatedSalary')

grid_set$log_prob=predict(logreg,type='response',newdata=grid_set[,c('Age','EstimatedSalary'
)])
grid_set$log_pred=as.numeric(ifelse(grid_set$log_prob>0.5,1,0))

#Plotting Logistic Regression

plot(set[,c('Age','EstimatedSalary')],
     main='Logistic Regression (Test Set)',
     xlab='Age',ylab='Estimated Salary',
     xlim=range(X1),ylim=range(X2)) +
  contour(X1,X2,matrix(as.numeric(grid_set$log_pred),length(X1),length(X2)),add=TRUE) +
  points(grid_set, pch= '.', col=ifelse(grid_set$log_pred==1,'springgreen3','tomato')) +
  points(set, pch=21, bg=ifelse(set$Purchased==1,'green4','red'))
```



```
## integer(0)
```

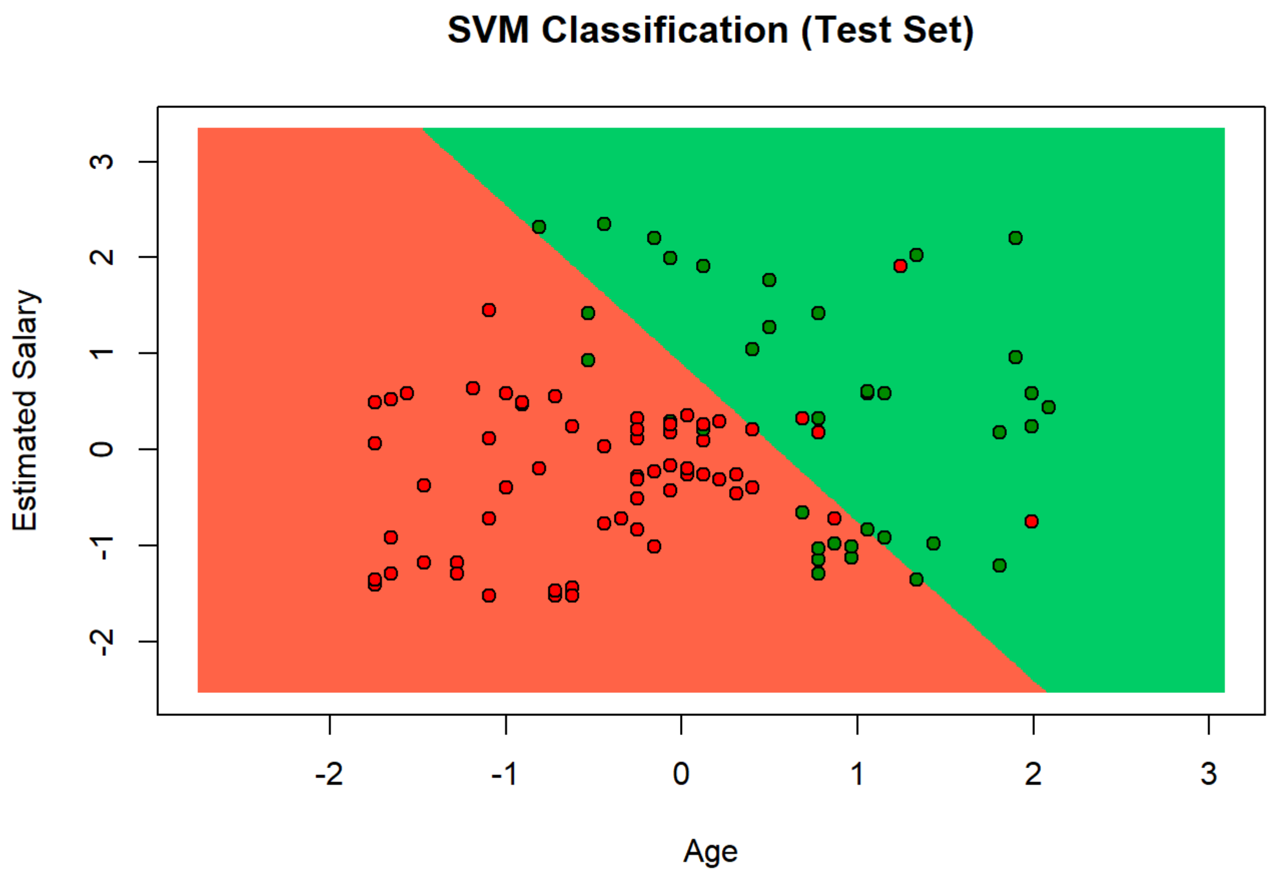
SUPPORT VECTOR MACHINE CLASSIFIER

SVM’s are typically used a non-probabilistic binary linear classifiers, where the two output data points are divided by the hyperplane where maximises the closest datapoints of the two classes. This hyperplane in SVM is called as maximum-margin classifier.

```
library(e1071)
svm_reg = svm(formula = Purchased ~ Age + EstimatedSalary,data = training_set, type = 'C-clas
sification',kernel = 'linear')
training_set$svm_pred=predict(svm_reg,newdata = training_set[,c('Age', 'EstimatedSalary')])
test_set$svm_pred=predict(svm_reg,newdata = test_set[,c('Age', 'EstimatedSalary')])
grid_set$svm_pred=predict(svm_reg,newdata = grid_set[,c('Age', 'EstimatedSalary')])
```

Visualizing the prediction boundary of the SVM classification model

```
plot(set[,c('Age','EstimatedSalary')],
     main='SVM Classification (Test Set)',
     xlab='Age',ylab='Estimated Salary',
     xlim=range(X1),ylim=range(X2)) +
  contour(X1,X2,matrix(as.numeric(grid_set$svm_pred),length(X1),length(X2)),add=TRUE) +
  points(grid_set, pch= '.', col=ifelse(grid_set$svm_pred==1,'springgreen3','tomato')) +
  points(set, pch=21, bg=ifelse(set$Purchased==1,'green4','red'))
```



```
## integer(0)
```

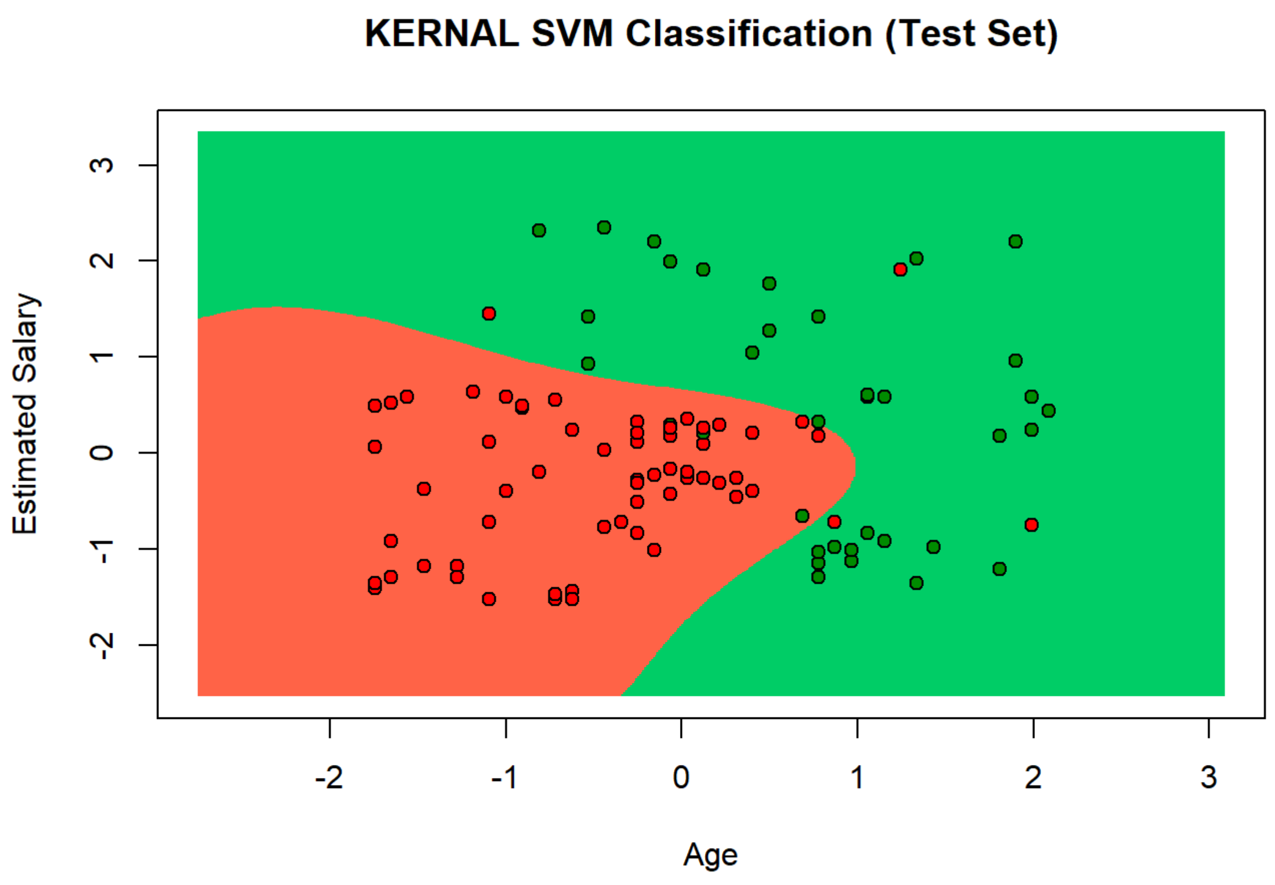
KERNEL SVM CLASSIFIER

Apart from linear classifiers, SVMs can also use the kernal trick to maximum-margin hyperplanes constructed on a transformed feature world.

```
library(e1071)
kernal_svm=svm(formula=Purchased~Age+EstimatedSalary,data=training_set,type='C-classificatio
n',kernal='radial')
training_set$ksvm_pred=predict(kernal_svm,newdata = training_set[,c('Age', 'EstimatedSalary'
)])
test_set$ksvm_pred=predict(kernal_svm,newdata = test_set[,c('Age', 'EstimatedSalary')])
grid_set$ksvm_pred=predict(kernal_svm,newdata = grid_set[,c('Age', 'EstimatedSalary')])
```

Visualizing the prediction boundary of the SVM classification model

```
plot(set[,c('Age','EstimatedSalary')],
     main='KERNAL SVM Classification (Test Set)',
     xlab='Age',ylab='Estimated Salary',
     xlim=range(X1),ylim=range(X2)) +
  contour(X1,X2,matrix(as.numeric(grid_set$ksvm_pred),length(X1),length(X2)),add=TRUE) +
  points(grid_set, pch= '.', col=ifelse(grid_set$ksvm_pred==1,'springgreen3','tomato')) +
  points(set, pch=21, bg=ifelse(set$Purchased==1,'green4','red'))
```



```
## integer(0)
```

NAIVE BAYES CLASSIFICATION

```r
library(caTools)
library(ggplot2)
library(ggpubr)
library(ElemStatLearn)
library(e1071)

dataset=read.csv("Social_Network_Ads.csv")
dataset=dataset[,3:5]
dataset$Purchased=factor(dataset$Purchased,levels=c(0 , 1))
print(head(dataset,10))
```

```
##    Age EstimatedSalary Purchased
## 1   19           19000         0
## 2   35           20000         0
## 3   26           43000         0
## 4   27           57000         0
## 5   19           76000         0
## 6   27           58000         0
## 7   27           84000         0
## 8   32          150000         1
## 9   25           33000         0
## 10  35           65000         0
```

```r
set.seed(123)
training_set=subset(dataset,sample.split(dataset$Purchased,SplitRatio = 0.75)==TRUE)
test_set=subset(dataset,sample.split(dataset$Purchased,SplitRatio = 0.75)==FALSE)
training_set[,1:2]=scale(training_set[,1:2])
test_set[,1:2]=scale(test_set[,1:2])

naive_bayes=naiveBayes(x=training_set[,1:2],y=training_set[,3])
summary(naive_bayes)
```

```
##           Length Class  Mode
## apriori   2      table  numeric
## tables    2      -none- list
## levels    2      -none- character
## isnumeric 2      -none- logical
## call      3      -none- call
```

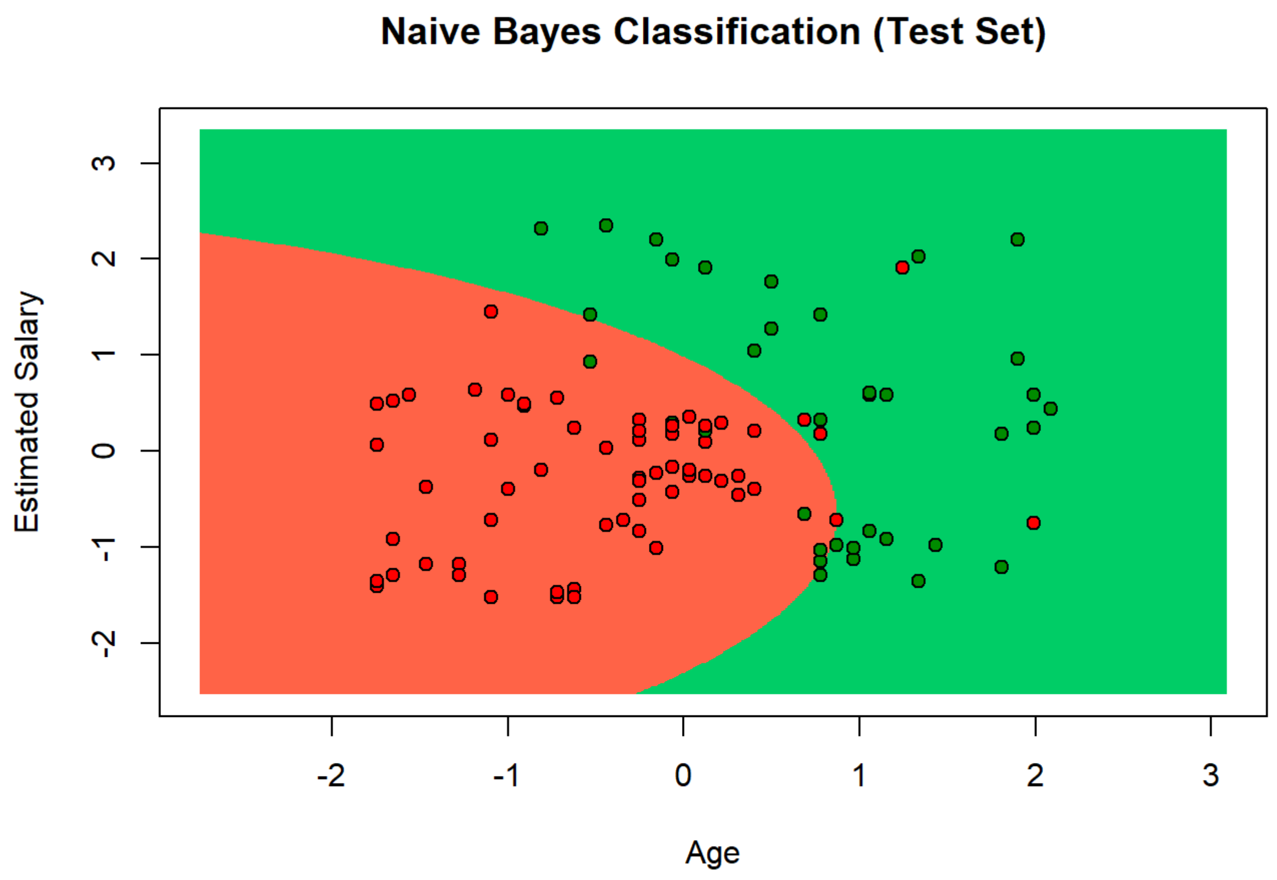
```r
training_set$naive_pred=predict(naive_bayes,newdata = training_set[,c('Age', 'EstimatedSalar
y')])
test_set$naive_pred=predict(naive_bayes,newdata = test_set[,c('Age', 'EstimatedSalary')])

set=test_set
X1=seq(min(set$Age)-1,max(set$Age)+1,by=0.01)
X2=seq(min(set$EstimatedSalary)-1,max(set$EstimatedSalary)+1,by=0.01)
grid_set=expand.grid(X1,X2)
colnames(grid_set)=c('Age','EstimatedSalary')

grid_set$naive_pred=predict(naive_bayes,newdata = grid_set[,c('Age', 'EstimatedSalary')])
```

Visualizing the prediction boundary of the Naive Bayes model

```
plot(set[,c('Age','EstimatedSalary')],
     main='Naive Bayes Classification (Test Set)',
     xlab='Age',ylab='Estimated Salary',
     xlim=range(X1),ylim=range(X2)) +
  contour(X1,X2,matrix(as.numeric(grid_set$naive_pred),length(X1),length(X2)),add=TRUE) +
  points(grid_set, pch= '.', col=ifelse(grid_set$naive_pred==1,'springgreen3','tomato')) +
  points(set, pch=21, bg=ifelse(set$Purchased==1,'green4','red'))
```



```
## integer(0)
```

DECISION TREES

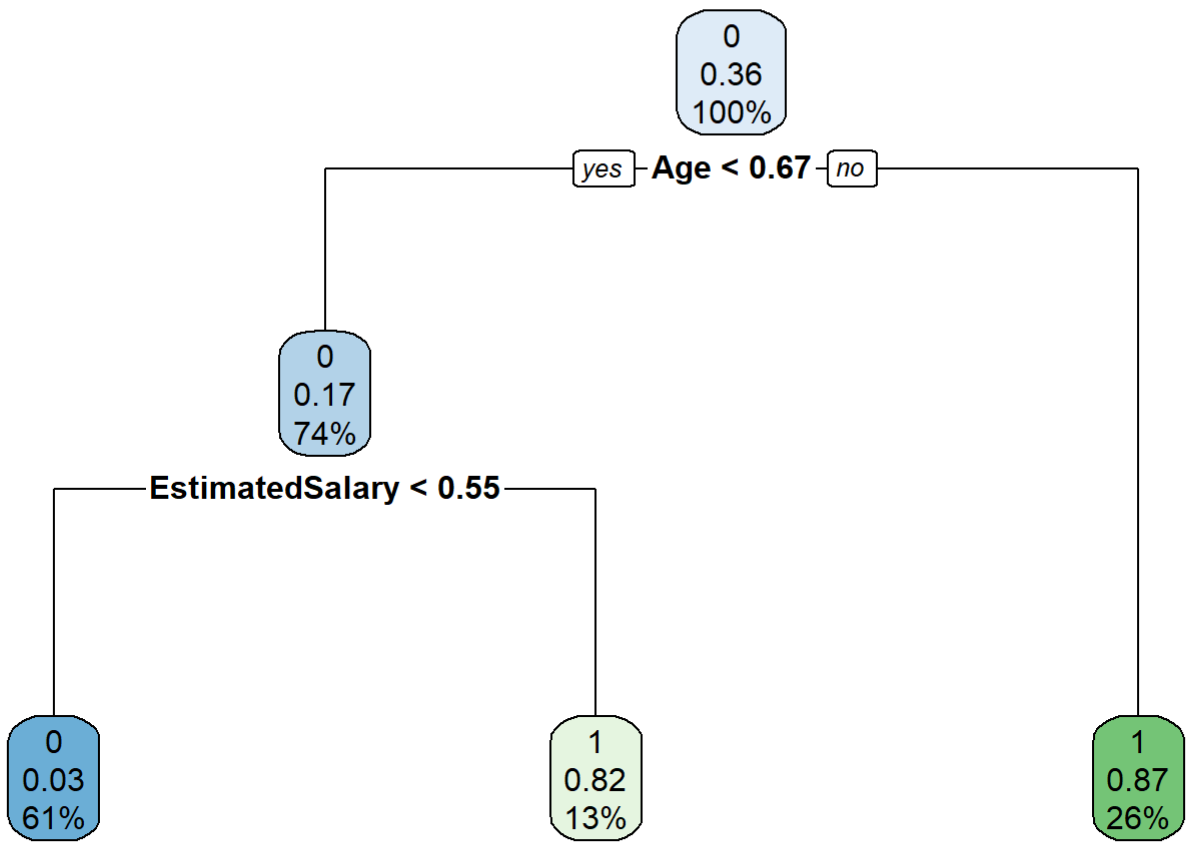
Decision trees uses various metrics to find the splitting points, but here it minimizes the Gini Impurity at each node to find the ideal split. Since we have only two variables, the final plot looks like a rectangular segment.

```
library(rpart)
library(rpart.plot)

dtree=rpart(formula=Purchased~Age+EstimatedSalary,data=training_set,method='class',control=rp
art.control(minbucket=10,minsplit=10))
summary(dtree)
```

```
## Call:
## rpart(formula = Purchased ~ Age + EstimatedSalary, data = training_set,
##     method = "class", control = rpart.control(minbucket = 10,
##         minsplit = 10))
##   n= 300
##
##          CP nsplit rel error    xerror       xstd
## 1 0.5514019      0 1.0000000 1.0000000 0.07754006
## 2 0.2336449      1 0.4485981 0.4672897 0.06032684
## 3 0.0100000      2 0.2149533 0.2336449 0.04473958
##
## Variable importance
##             Age EstimatedSalary
##              59              41
##
## Node number 1: 300 observations,    complexity param=0.5514019
##   predicted class=0  expected loss=0.3566667  P(node) =1
##     class counts:   193   107
##    probabilities: 0.643 0.357
##   left son=2 (221 obs) right son=3 (79 obs)
##   Primary splits:
##       Age             < 0.6724378  to the left,  improve=57.27285, (0 missing)
##       EstimatedSalary < 0.5387442  to the left,  improve=47.14670, (0 missing)
##   Surrogate splits:
##       EstimatedSalary < 1.908675   to the left,  agree=0.747, adj=0.038, (0 split)
##
## Node number 2: 221 observations,    complexity param=0.2336449
##   predicted class=0  expected loss=0.1719457  P(node) =0.7366667
##     class counts:   183    38
##    probabilities: 0.828 0.172
##   left son=4 (182 obs) right son=5 (39 obs)
##   Primary splits:
##       EstimatedSalary < 0.5530144  to the left,  improve=39.840550, (0 missing)
##       Age             < -0.5704567 to the left,  improve= 4.113636, (0 missing)
##   Surrogate splits:
##       Age < 0.4812232  to the left,  agree=0.833, adj=0.051, (0 split)
##
## Node number 3: 79 observations
##   predicted class=1  expected loss=0.1265823  P(node) =0.2633333
##     class counts:    10    69
##    probabilities: 0.127 0.873
##
## Node number 4: 182 observations
##   predicted class=0  expected loss=0.03296703  P(node) =0.6066667
##     class counts:   176     6
##    probabilities: 0.967 0.033
##
## Node number 5: 39 observations
##   predicted class=1  expected loss=0.1794872  P(node) =0.13
##     class counts:     7    32
##    probabilities: 0.179 0.821
```

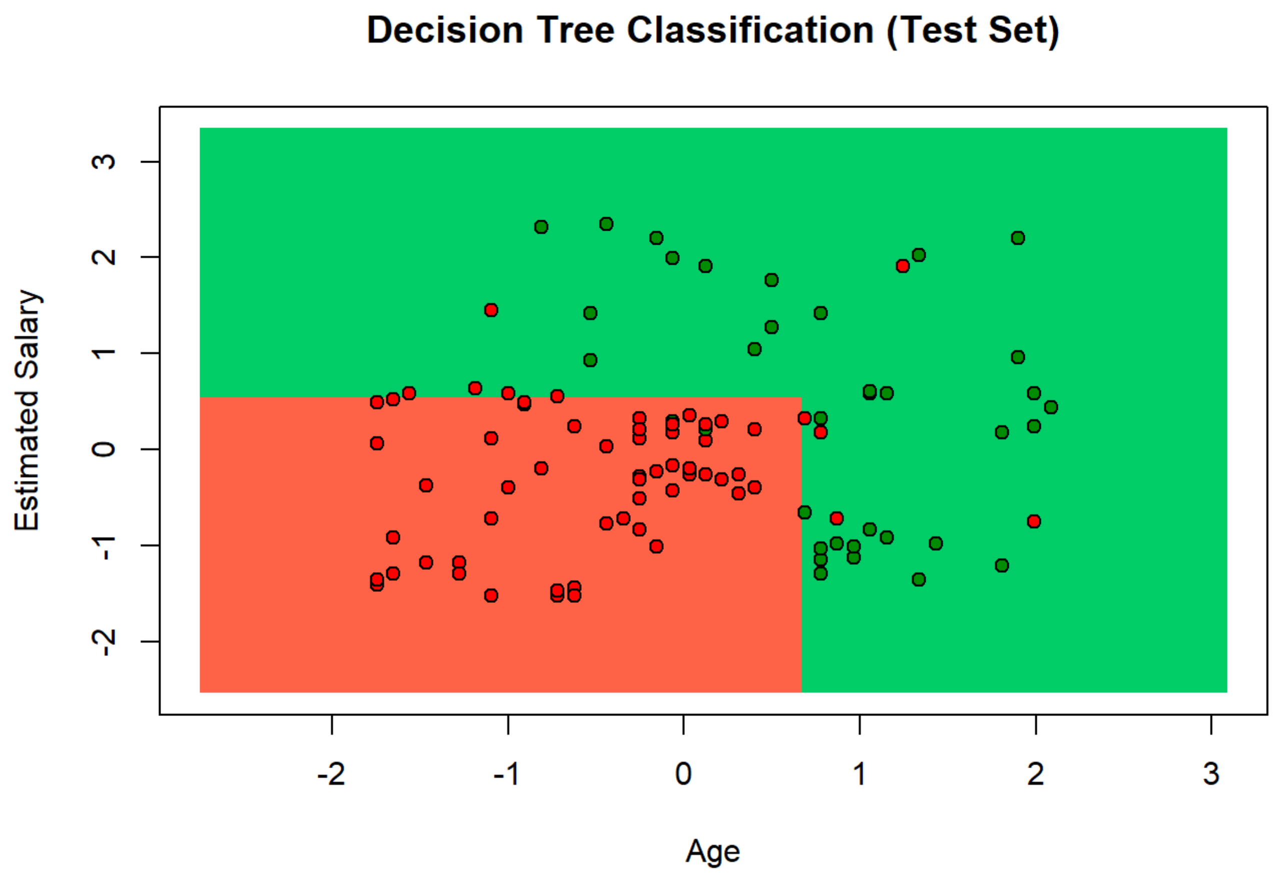
```
rpart.plot(dtree,extra=106)
```



```
training_set$dtree_pred=predict(dtree,newdata = training_set[,1:2],type='class')
test_set$dtree_pred=predict(dtree,newdata=test_set[1:2],type='class')
grid_set$dtree_pred=predict(dtree,newdata = grid_set[,c('Age', 'EstimatedSalary')],type='clas
s')
```

Visualizing the prediction boundary of the Decision Tree

```
plot(set[,c('Age','EstimatedSalary')],
     main='Decision Tree Classification (Test Set)',
     xlab='Age',ylab='Estimated Salary',
     xlim=range(X1),ylim=range(X2)) +
  contour(X1,X2,matrix(as.numeric(grid_set$dtree_pred),length(X1),length(X2)),add=TRUE) +
  points(grid_set, pch= '.', col=ifelse(grid_set$dtree_pred==1,'springgreen3','tomato')) +
  points(set, pch=21, bg=ifelse(set$Purchased==1,'green4','red'))
```



```
## integer(0)
```

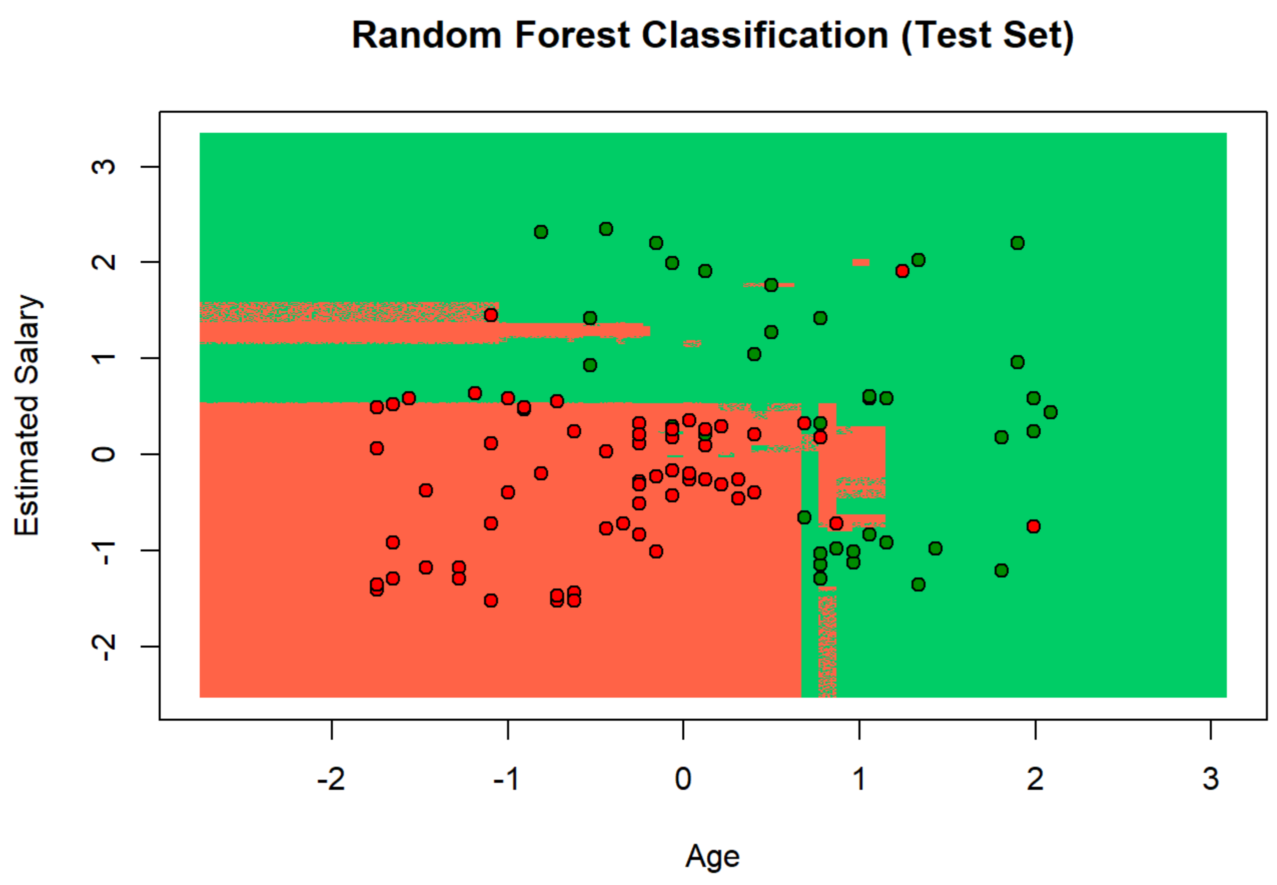
RANDOM FOREST CLASSIFIER

```
library(randomForest)

ranforest=randomForest(x=training_set[1:2],y=training_set$Purchased,ntree=10)
test_set$rf_pred=predict(ranforest,newdata = test_set[1:2])
grid_set$rf_pred=predict(ranforest,newdata = grid_set[,c('Age', 'EstimatedSalary')])
```

Visualizing the prediction boundary of the RF classifier

```
plot(set[,c('Age','EstimatedSalary')],
     main='Random Forest Classification (Test Set)',
     xlab='Age',ylab='Estimated Salary',
     xlim=range(X1),ylim=range(X2)) +
  contour(X1,X2,matrix(as.numeric(grid_set$rf_pred),length(X1),length(X2)),add=TRUE) +
  points(grid_set, pch= '.', col=ifelse(grid_set$rf_pred==1,'springgreen3','tomato')) +
  points(set, pch=21, bg=ifelse(set$Purchased==1,'green4','red'))
```

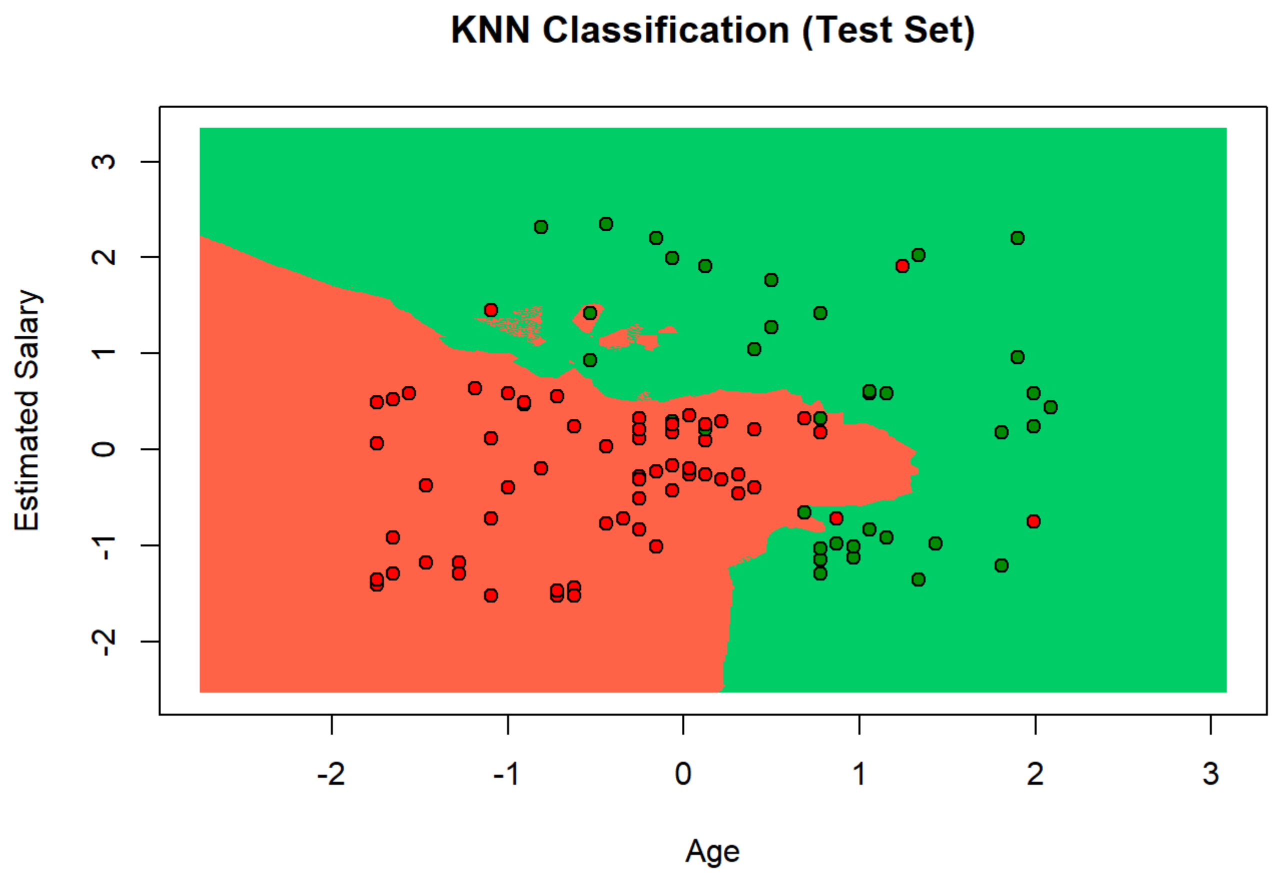


```
## integer(0)
```

K-NEAREST NEIGHBOURS

```
library(class)
test_set$knn_pred=knn(train=training_set[,c('Age','EstimatedSalary')],test=test_set[c('Age',
'EstimatedSalary')]
           ,cl=training_set[,3],k=5,prob=TRUE)
grid_set$knn_pred=knn(train=training_set[,1:2],test=grid_set[,1:2],cl=training_set[,3],k=5)

plot(set[,c('Age','EstimatedSalary')],
     main='KNN Classification (Test Set)',
     xlab='Age',ylab='Estimated Salary',
     xlim=range(X1),ylim=range(X2)) +
  contour(X1,X2,matrix(as.numeric(grid_set$knn_pred),length(X1),length(X2)),add=TRUE) +
  points(grid_set, pch= '.', col=ifelse(grid_set$knn_pred==1,'springgreen3','tomato')) +
  points(set, pch=21, bg=ifelse(set$Purchased==1,'green4','red'))
```


KNN Classification (Test Set)
Estimated Salary
3
2
1
0
-1
-2
-2
-1
0
1
2
3
Age

```
## integer(0)
```