Machine Learning

Decision Tree Classification

Ayşe Ceren Çiçek

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 - Internal nodes represent the features of a dataset - Branches represent the decision rules - Each leaf node represents the outcome

Dataset: https://www.kaggle.com/rakeshrau/social-network-ads

This dataset shows which of the users purchased/not purchased a particular product.

The columns are:

- User ID
- Gender
- Age
- EstimatedSalary
- Purchased

Importing libraries

```
library(caTools)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(rpart)
```

Loading dataset

```
dataset = read.csv('dataset.csv')
dataset = dataset[, 3:5]
head(dataset, n=5)
```

```
##   Age EstimatedSalary Purchased
## 1  19           19000         0
## 2  35           20000         0
## 3  26           43000         0
## 4  27           57000         0
## 5  19           76000         0
```

Data Preprocessing

Turn the target feature to factor

```
dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1))
dataset$Purchased
```

```
##   [1] 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0
##  [38] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
##  [75] 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0
## [112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
## [149] 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
## [186] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 1 0 0 0 1 0 1
## [223] 1 1 0 0 1 1 0 1 1 0 1 1 0 1 0 0 0 1 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 0 0 1
## [260] 1 0 1 1 0 1 1 0 0 1 0 0 1 1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 1 0 1 1 1 1 0 0 0
## [297] 1 1 0 1 1 1 1 1 1 0 0 0 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 0 1 1 0 0 0 1 1 0 1 0
## [334] 0 1 0 1 0 0 1 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1 1 1 1 0 1
## [371] 1 1 0 1 0 1 0 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1
## Levels: 0 1
```

We will only use Age, Salary and Purchased columns.

Split data into Train and Test set

Purchased column is our dependent variable.

```
set.seed(123)
splitted = sample.split(dataset$Purchased, SplitRatio = 0.75)
train_Set = subset(dataset, splitted == TRUE)
test_Set = subset(dataset, splitted == FALSE)
```

Feature Scaling

Feature scaling is a method used to normalize the range of independent variables or features of data.

We will scale all the features except our dependent variable, Purchased.

```
trainSet = train_Set
testSet = test_Set
trainSet[-3] = scale(train_Set[-3])
testSet[-3] = scale(test_Set[-3])
```

Apply Decision Tree

```
model = rpart(formula = Purchased ~ ., data = trainSet)
```

Prediction

Probability prediction show us predicted probabilities that the user will buy the product.

```
probability.prediction = predict(model, newdata = testSet[-3], type = 'class')
probability.prediction
```

```
##   2   4   5   9  12  18  19  20  22  29  32  34  35  38  45  46  48  52  66  69
##   0   0   0   0   0   0   1   1   0   0   1   0   1   0   0   0   0   0   0   0
##  74  75  82  84  85  86  87  89 103 104 107 108 109 117 124 126 127 131 134 139
##   1   0   0   1   0   1   0   0   1   1   0   1   1   0   0   0   0   0   0   0
## 148 154 156 159 162 163 170 175 176 193 199 200 208 213 224 226 228 229 230 234
##   0   0   0   0   1   0   0   0   0   0   0   0   1   1   1   0   1   0   0   1
## 236 237 239 241 255 264 265 266 273 274 281 286 292 299 302 305 307 310 316 324
##   1   0   1   1   0   0   1   1   1   1   1   1   1   0   0   0   1   0   0   1
## 326 332 339 341 343 347 353 363 364 367 368 369 372 373 380 383 389 392 395 400
##   0   1   0   1   0   1   1   0   0   1   1   0   1   0   1   1   1   1   0   1
## Levels: 0 1
```

Confusion Matrix

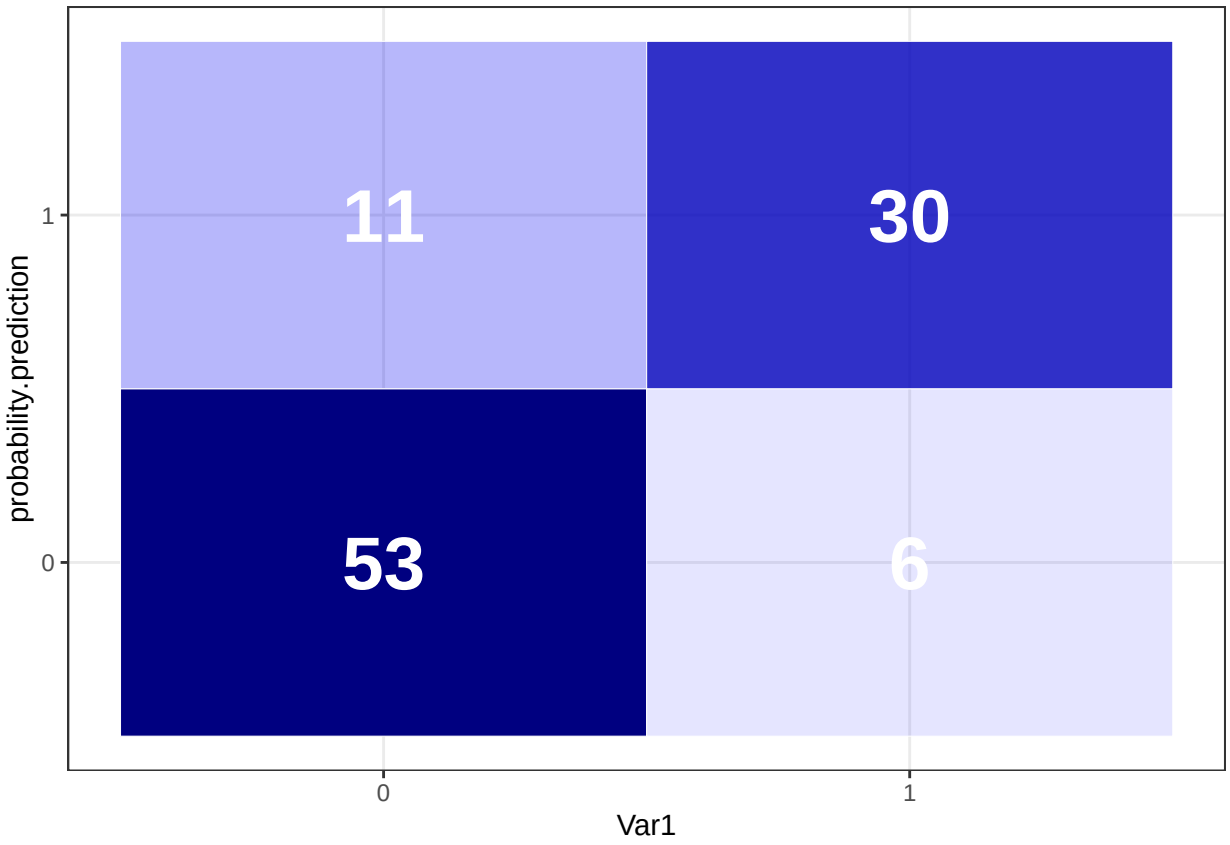
```
# Generate confusion matrix
conf.matrix <- confusionMatrix(table(testSet[, 3], probability.prediction))
conf.matrix
```

```
## Confusion Matrix and Statistics
##
##    probability.prediction
##       0  1
##   0 53 11
##   1  6 30
##
##                Accuracy : 0.83
```

```
##                  95% CI : (0.7418, 0.8977)
##     No Information Rate : 0.59
##     P-Value [Acc > NIR] : 2.321e-07
##
##                   Kappa : 0.642
##
##  Mcnemar's Test P-Value : 0.332
##
##             Sensitivity : 0.8983
##             Specificity : 0.7317
##          Pos Pred Value : 0.8281
##          Neg Pred Value : 0.8333
##              Prevalence : 0.5900
##          Detection Rate : 0.5300
##    Detection Prevalence : 0.6400
##       Balanced Accuracy : 0.8150
##
##        'Positive' Class : 0
##
```

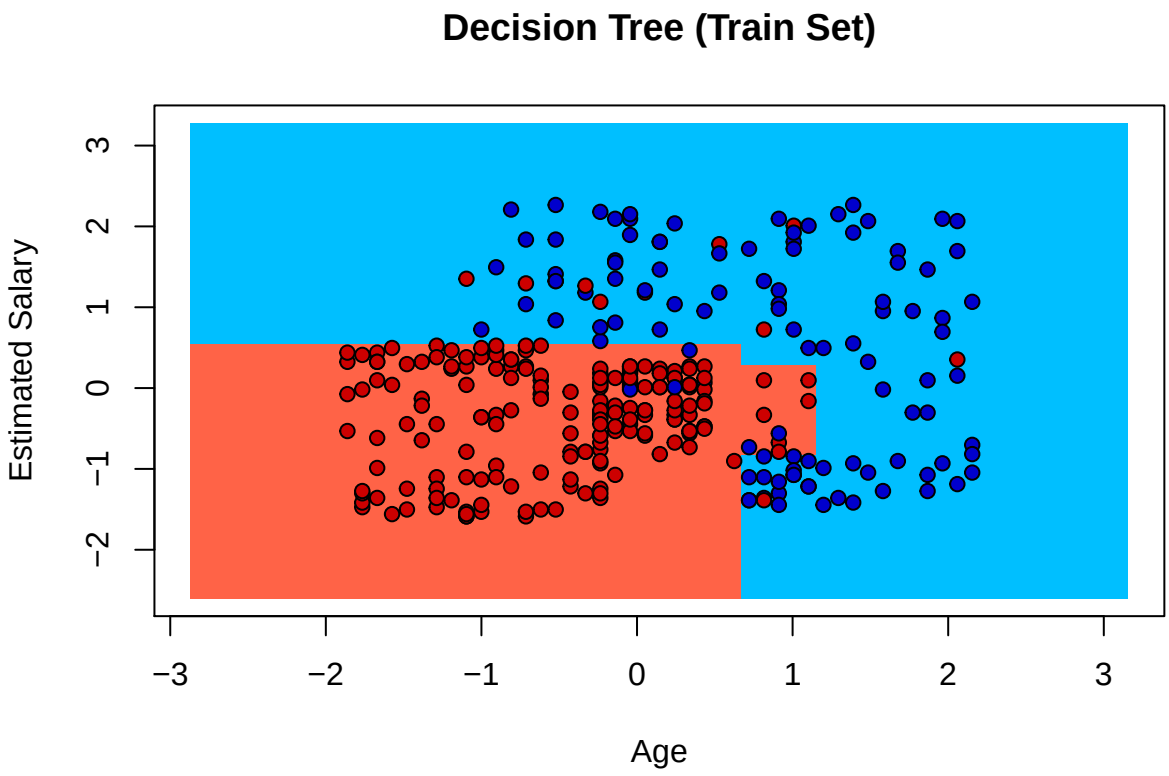
The accurary is 83%. We have 11 + 6 incorrect classifications.

```
# Heatmap visualization of confusion matrix
table <- data.frame(conf.matrix$table)


plotTable <- table %>%
  group_by(probability.prediction) %>%
  mutate(prop = Freq/sum(Freq))

ggplot(data =  plotTable, mapping = aes(x = Var1, y = probability.prediction, alpha = prop)) +
  geom_tile(aes(fill = Freq), colour = "white") +
  geom_text(aes(label = Freq), vjust = .5, fontface  = "bold", alpha = 1, color="white", size=10) +
  scale_fill_gradient(low = "blue", high = "navyblue") +
  theme_bw() + theme(legend.position = "none")
```



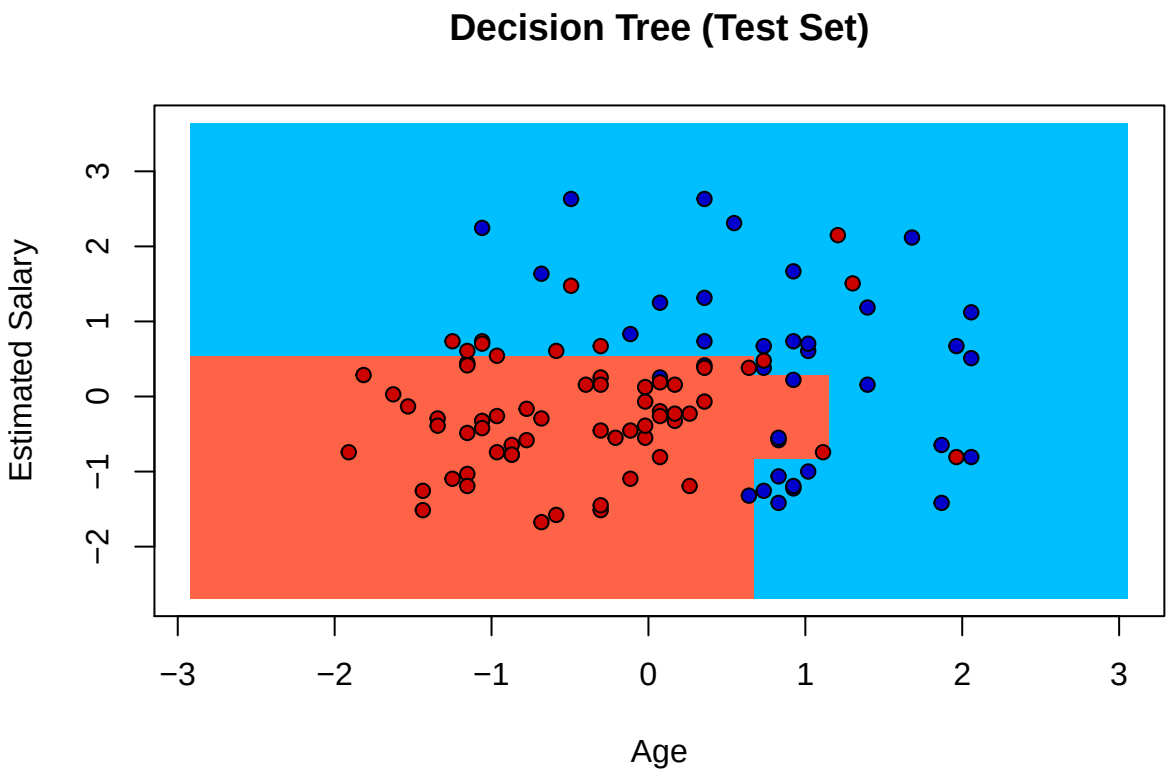
Visualize Train Set Results

```
set = trainSet
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(model, newdata = grid_set, type = 'class')

plot(set[, -3], main = 'Decision Tree (Train 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, 'deepskyblue', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'blue3', 'red3'))
```



Visualize Test Set Results

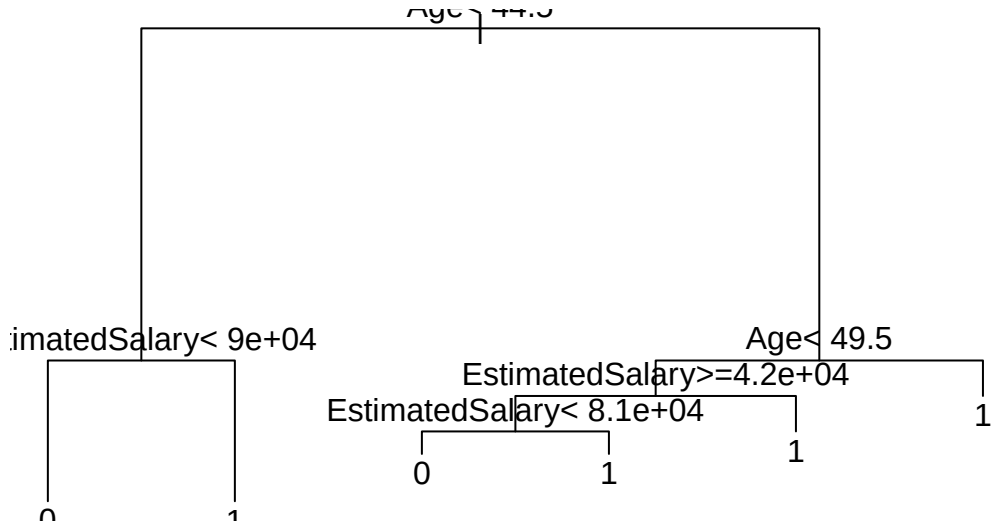
```
set = testSet
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)

grid_set = expand.grid(X1, X2)

colnames(grid_set) = c('Age', 'EstimatedSalary')

y_grid = predict(model, newdata = grid_set, type = 'class')

plot(set[, -3], main = 'Decision Tree (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, 'deepskyblue', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'blue3', 'red3'))
```



Plot Decision Tree

We will apply the decision tree to data that was not applied feature scaling.

```
model = rpart(formula = Purchased ~ ., data = train_Set)
plot(model)
text(model)
```


imatedSalary< 9e+04
Age< 49.5
EstimatedSalary>=4.2e+04
EstimatedSalary< 8.1e+04
1
1
0
1