Job applicant decision tree

##Load library  
library(readxl)  
library(rpart)

library(rpart.plot)

library(caret)

library(e1071)

library(Metrics)

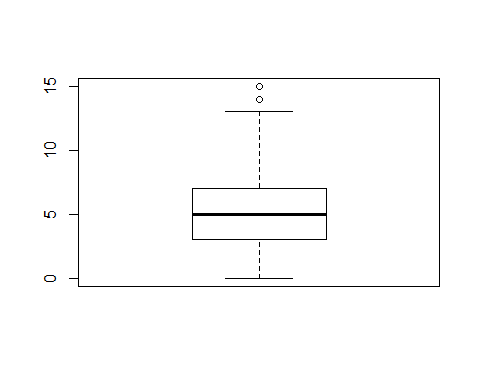
##Import data  
applicants = read\_excel('JobRaw.xls')  
  
##Inspect data table  
head(applicants)

## # A tibble: 6 x 8  
## Age State Degree Major Experience Outcome X\_\_1 X\_\_2  
## <dbl> <chr> <chr> <chr> <dbl> <chr> <chr> <dbl>  
## 1 20 AZ Cert none 0 minimal <NA> NA  
## 2 27 TX Cert none 5 minimal <NA> NA  
## 3 28 CO MS other 6 minimal <NA> NA  
## 4 28 MA UG IS 6 adequate <NA> NA  
## 5 30 CA MS Csci 8 adequate <NA> NA  
## 6 21 TX Cert none 0 minimal <NA> NA

### Columns 7 and 8 are empty and needs to be removed  
  
## Remove columns 7 & 8  
applicants = applicants[-8]  
applicants = applicants[-7]  
  
## Inspect data for missing values and anomalies  
summary(applicants)

## Age State Degree Major   
## Min. :19.00 Length:500 Length:500 Length:500   
## 1st Qu.:25.00 Class :character Class :character Class :character   
## Median :28.00 Mode :character Mode :character Mode :character   
## Mean :27.69   
## 3rd Qu.:30.00   
## Max. :43.00   
## Experience Outcome   
## Min. : 0.000 Length:500   
## 1st Qu.: 3.000 Class :character   
## Median : 5.000 Mode :character   
## Mean : 5.188   
## 3rd Qu.: 7.000   
## Max. :15.000

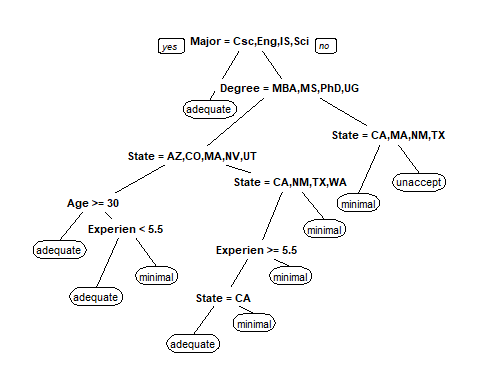
## The max of experience seems high in relation to the median and 3rd quartile, check for outliers  
boxplot(applicants$Experience)



### There are in fact 2 outliers at 15 and 14  
### May consider removing at a later time  
  
## Separate data into training and test sets  
train\_app = applicants[1:250,]  
test\_app = applicants[251:500,]  
  
  
  
## Setting tree control parameters  
fitControl = trainControl(method = "cv", number = 5)  
cartGrid = expand.grid(.cp=(1:50)\*0.01)  
  
## Creating the decision tree model  
tree\_model = train(Outcome ~ ., data = train\_app, method = "rpart", trControl = fitControl, tuneGrid = cartGrid)  
  
## Getting the complexity parameter (cp) of the model  
print(tree\_model)

## CART   
##   
## 250 samples  
## 5 predictor  
## 4 classes: 'adequate', 'excellent', 'minimal', 'unacceptable'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 199, 200, 200, 201, 200   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.01 0.6124946 0.32612439  
## 0.02 0.6201809 0.32154831  
## 0.03 0.6081809 0.27953915  
## 0.04 0.5959360 0.22561381  
## 0.05 0.5960144 0.22261375  
## 0.06 0.5960144 0.22261375  
## 0.07 0.5960144 0.22261375  
## 0.08 0.5960144 0.22261375  
## 0.09 0.5960144 0.22261375  
## 0.10 0.5960144 0.22261375  
## 0.11 0.5960144 0.22261375  
## 0.12 0.5960144 0.22261375  
## 0.13 0.5960144 0.22261375  
## 0.14 0.5960144 0.22261375  
## 0.15 0.5960144 0.22261375  
## 0.16 0.5960144 0.22261375  
## 0.17 0.5680144 0.12832246  
## 0.18 0.5480864 0.04421365  
## 0.19 0.5400864 0.00000000  
## 0.20 0.5400864 0.00000000  
## 0.21 0.5400864 0.00000000  
## 0.22 0.5400864 0.00000000  
## 0.23 0.5400864 0.00000000  
## 0.24 0.5400864 0.00000000  
## 0.25 0.5400864 0.00000000  
## 0.26 0.5400864 0.00000000  
## 0.27 0.5400864 0.00000000  
## 0.28 0.5400864 0.00000000  
## 0.29 0.5400864 0.00000000  
## 0.30 0.5400864 0.00000000  
## 0.31 0.5400864 0.00000000  
## 0.32 0.5400864 0.00000000  
## 0.33 0.5400864 0.00000000  
## 0.34 0.5400864 0.00000000  
## 0.35 0.5400864 0.00000000  
## 0.36 0.5400864 0.00000000  
## 0.37 0.5400864 0.00000000  
## 0.38 0.5400864 0.00000000  
## 0.39 0.5400864 0.00000000  
## 0.40 0.5400864 0.00000000  
## 0.41 0.5400864 0.00000000  
## 0.42 0.5400864 0.00000000  
## 0.43 0.5400864 0.00000000  
## 0.44 0.5400864 0.00000000  
## 0.45 0.5400864 0.00000000  
## 0.46 0.5400864 0.00000000  
## 0.47 0.5400864 0.00000000  
## 0.48 0.5400864 0.00000000  
## 0.49 0.5400864 0.00000000  
## 0.50 0.5400864 0.00000000  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.02.

## complexity parameter = .01  
  
## Build decision tree using the .01 cp  
main\_tree = rpart(Outcome ~ ., data = train\_app, control = rpart.control(cp=0.01))  
  
## View display of the decision tree  
prp(main\_tree)



## Predict outcome of the training data  
train\_pred = predict(main\_tree, train\_app, type = "vector")  
  
## View the confusion matrix of the training prediction  
table(train\_app$Outcome, train\_pred)

## train\_pred  
## 1 3 4  
## adequate 123 10 2  
## excellent 7 0 0  
## minimal 21 50 8  
## unacceptable 3 9 17

## 76% accuracy  
  
## Predict the outcome of the test data  
pre\_score = predict(main\_tree, test\_app, type = "vector")  
  
## View the confusion matrix of the model  
table(test\_app$Outcome, pre\_score)

## pre\_score  
## 1 3 4  
## adequate 93 35 0  
## excellent 16 0 0  
## minimal 23 40 13  
## unacceptable 3 19 8

## 56.4% accuracy

**Conclusion**

The model did not predict any of the outcomes as excellent because all the records with outcome as excellent falls under the root node of the tree Major = Csc, Eng, IS, Sci. Due to the number of “adequates” being proportionally high, the model automatically assigns an outcome of adequate for any records with these Majors.

The confusion matrix for the training data is:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Modeled** | **Modeled** | **Modeled** | **Modeled** |
| **Actual** | *Adequate* | *Excellent* | *Minimal* | *Unacceptable* |
| *Adequate* | 123 | 0 | 10 | 2 |
| *Excellent* | 7 | 0 | 0 | 0 |
| *Minimal* | 21 | 0 | 50 | 8 |
| *Unacceptable* | 3 | 0 | 9 | 17 |

The training model results in a 76% accuracy

The confusion matrix for the training data is:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Modeled** | **Modeled** | **Modeled** | **Modeled** |
| **Actual** | *Adequate* | *Excellent* | *Minimal* | *Unacceptable* |
| *Adequate* | 63 | 0 | 35 | 0 |
| *Excellent* | 16 | 0 | 0 | 0 |
| *Minimal* | 23 | 0 | 40 | 13 |
| *Unacceptable* | 3 | 0 | 19 | 8 |

Running the model on the test data however, results in a 56.4% accuracy rating