SVM vs Random Forest - A Literature Review and Analysis

CUNY 622 - Assignment 3

Eric Hirsch

10/25/2022

Contents

I. Essay

For this assignment we discuss the advantages and disadvantages of random forest versus SVM, both in the literature and in practice with the previous assignment's data set. As a starting point, two example articles were offered which predicted Covid positive cases using decision tree ensembles and SVM, respectively.

Because of the extreme imbalance between positive and negative cases, the authors of the decision tree ensemble article used decision tree ensembles specifically designed for imbalanced data sets (for example, balanced random forest). Special sampling techniques were also applied to address the imbalance of the data set. The best AUC score (.881) was achieved with RUSbagging.

The SVM article's analysis was slightly more complex, in that it predicted whether a client had no infection, mild infection or serious infection. The model was effective, particularly in predicting severe cases, with an F1 score of .97. The article makes the claim that "SVM works best in predicting COVID-19 cases with maximum accuracy." To support this claim, they performed a comparative study of supervised learning models, including Random Forest. SVM achieved higher F1 scores than the others.

That SVM is the superior model in predicting Covid cases generally is a rather bold statement which can't be justified by one study. There are too many variables – the particular kind and nature of the data collected for the study, the types and tunings of the machine learning algorithms (for example, this study did not

consider the " balanced" random forest algorithms that were featured in the first article), and the study design (such as breaking down the classes into no infection, mild or serious).

An examination of articles which compare SVM and random forest in the field of Geography (while I am no longer in this field, it's where I got my education – Berkeley PhD 1996) do not show the superiority of one algorithm over the other. Indeed, there is little consistency either for which algorithm is favored, or for how performance is to be predicted and compared. For example, a study of multispectral images to predict canopy nitrogen weight in corn showed that random forests were only marginally better than SVR (Support Vector Regression) but that the concepts and analysis were easier to interpret with random forests (Lee et al.). A study of groundwater mapping showed a higher AUC for random forest than for SVM tested with a linear, polynomial, sigmoid, and radial kernel (Naghibi et al.). Likewise, a study of sediment transport in the Tigris-Euphrates river also found that random forest predicted better than SVM (Al Mukhtar et al.).

In contrast, a study predicting dominant tree species from Landsat images taken of the Krkonose mountains in the Czech Republic, found that SVM performed better than random forest in that particular study, but acknowledged that random forest consistently provided better results in other studies when spatial resolution was low, while SVM appeared to perform better when there were significantly more features (Zagajewski et al.). A study of images by the satellite remote-sensing Sentinel-2A also found that SVM was most accurate (95.17%) (Kavzoglu et al.).

A systematic review conducted in 2020 of 250 peer-reviewed papers on remote-sensing image classification showed that despite some inroads from deep learning, SVM and random forest remained the two most popular image classification techniques, mainly due to lower computational complexity (Sheykhmousa et al.). SVM is seen to be particularly effective where there is high dimensionality and limited training samples, while random forest is easier to use (fewer hyper parameters to tune) and more flexible with more complex classifications. Both tend to be highly accurate, although, some researchers still tout one or the other as the more superior without any strong basis.

As the researchers pointed out, SVM largely depends on the selection of the right kernel function – radial and polynomial tend to be popular in the remote-sensing field. The binary nature of SVM also creates complications for its use in multiclass scenarios, which occur frequently in remote-sensing, although these can be overcome. The ability of SVMs to manage small training sets and operate within high dimensional spaces (which is particularly applicable to remote-sensing image data) make SVMs attractive.

Random Forest, on the other hand, is popular because the decision-making process behind it is clearer and more understandable, it is easily implemented in parallel structure for geo-big data computing, it can handle thousands of input variables, is robust to outliers and noise, and is computationally lighter than other tree ensemble methods. These recommendations echo those of the machine learning literature.

Beyond these general recommendations, however, the superiority of one algorithm over the other may depend on the idiosyncrasies of the data set under examination. It is wise to examine the performance of both before assuming one will perform better than the other. We can illustrate this with the database from the previous assignment.

The database contains 466 records of small towns in the Northeast, together with statistics on poverty, industrialization, pollution, crime rate and so on. In the previous assignment, we performed regression analysis on the pupil teacher ratio (PT ratio) using random forest. In this exercise we created a binary variable containing high PT ratio (the mean and above), and low PT ratio (below the mean) in order to run a more simple classification analysis. We ran three support vector machines (linear, radial, and polynomial) as well as a random forest and a decision tree. All of the algorithms were optimized for the parameters which may be tuned.

When all of the variables were included in the data set, random forest performed significantly better than SVM (98% AUC vs. 84% for SVM using a radial kernel). Even a simple decision tree yielded a higher AUC (85%).

We investigated possible reasons that SVM performs more poorly in this data set. It was found that for certain variables (pollution, e.g), there were anomalous cohorts that were driving the analysis. For example,

there was a sizable cohort of schools all at a PT ratio of exactly 21. Further, all of the schools above a pollution index of .65 were in this cohort.

Although pollution was barely linearly correlated with PT ratio (1.7), the random forest algorithm relied on it, as anomalies like these lend themselves well to a binary decision point. The SVM algorithm (AUC=82%), which only looks at the support vectors, did not appear to benefit from this information. The evidence for this was that when the cohort was removed, the AUC for random forest dropped significantly while the AUC for SVM remained the same. In addition, when the algorithms were run on a data set with minimal columns with much smoother distributions, radial SVM outperformed all of the other algorithms and polynomial SVM was second.

We also tested the assertion that SVM performs better in high dimensional, low sample data by taking only every fifth record in the data set, reducing it to 94 rows with 12 columns. In this case, SVM does in fact approach, though does not overtake, Random Forest.

This analysis shows that it's important to test both algorithms in the data as the reason for the superiority of one performance over the other may not be readily apparent. It can, however, be worthwhile to investigate further what it is about the data shape that favors one over the other.

Sources:

Al-Mukhtar, Mustafa. (2019) "Random Forest, Support Vector Machine, and Neural Networks to Modelling Suspended Sediment in Tigris River-Baghdad - Environmental Monitoring and Assessment." SpringerLink, 25 Oct. 2019, link.springer.com/article/10.1007/s10661-019-7821-5.

Kavzoglu, Taskin, et al. (2020) "COMPARISON OF SUPPORT VECTOR MACHINES, RANDOM FOREST AND DECISION TREE METHODS FOR CLASSIFICATION OF SENTINEL - 2A IMAGE USING DIFFERENT BAND COMBINATIONS." Conference: 41st Asian Conference on Remote Sensing (ACRS 2020), 9 Nov. 2020.

Lee, H. Wang, et al. (2022) Using Linear Regression, Random Forests, and Support Vector Machine With Unmanned Aerial Vehicle Multispectral Images to Predict Canopy Nitrogen Weight in Corn – DOAJ, doaj.org/article/70fb513d21214e4387057cba745e6c0d. Accessed 4 Nov. 2022.

Naghibi, S.A., Ahmadi, K. & Daneshi, (2017) A. Application of Support Vector Machine, Random Forest, and Genetic Algorithm Optimized Random Forest Models in Groundwater Potential Mapping. Water Resour Manage 31, 2761–2775 (2017). https://doi.org/10.1007/s11269-017-1660-3

Rienow, A, Mustafa, A, Krelaus, L, Lindner, C. (2021) Modeling urban regions: Comparing random forest and support vector machines for cellular automata. Transactions in GIS. 2021; 25: 1625– 1645. https://doi.org/10.1111/tgis.12756

Sheykhmousa, M. and Mahdianpari, M. (2020) Support Vector Machine vs. Random Forest for Remote Sensing Image Classification: A Meta-analysis and systematic review, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing.

Zagajewski, Bogdan, et al. (2021) Comparison of Random Forest, Support Vector Machines, and Neural Networks for Post-Disaster Forest Species Mapping of the Krkonoše/Karkonosze Transboundary Biosphere Reserve – DOAJ, 7 Jan. 2021, doaj.org/article/e1836d4bb940454291eb1600f64afbd7.

II. Analysis

In this part of the assignment we compare SVM to the Random Forest analysis we performed last week.

```
knitr::opts_chunk$set()
```

```
library(tidyverse)
```

A. Description

```
## Warning: package 'tidyverse' was built under R version 4.0.5
```

```
## Warning: replacing previous import 'lifecycle::last_warnings' by
## 'rlang::last_warnings' when loading 'hms'
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.0     v forcats 0.5.1
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
## Warning: package 'tibble' was built under R version 4.0.5
```

```
## Warning: package 'tidyr' was built under R version 4.0.5
```

```
## Warning: package 'readr' was built under R version 4.0.5
```

```
## Warning: package 'dplyr' was built under R version 4.0.5
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
devtools::install_github("ericonsi/EHData")
```

```
## Skipping install of 'EHData' from a github remote, the SHA1 (651a607f) has not changed since last ins
##   Use `force = TRUE` to force installation
```

```
library(ggsci)
```

```
## Warning: package 'ggsci' was built under R version 4.0.5
```

```
library(EHData)
library(patchwork)
```

```
## Warning: package 'patchwork' was built under R version 4.0.5
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.5

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.0.5

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(car)
```

```
## Warning: package 'car' was built under R version 4.0.5

## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode

## The following object is masked from 'package:purrr':
##
##     some
```

```
df1 <- read.csv("D:\\RStudio\\CUNY_622\\3\\CrimeRate.csv")
```

The data set consists of 466 observations of data realted to small towns in the North East. There are 11 numeric variables and two binary variables. There are no missing values. The variables include the level of industrialization, average tax rates, pollution levels, and so on. This data set is often used to predict crime rates, but we won't use it for that purpose.

These are the variables:

- zn: proportion of residential land zoned for large lots (over 25000 square feet)
- indus: proportion of non-retail business acres per suburb
- chas: a dummy var. for whether the suburb borders the Charles River (1) or not (0)
- nox: nitrogen oxides concentration (parts per 10 million)
- rm: average number of rooms per dwelling
- age: proportion of owner-occupied units built prior to 1940
- dis: weighted mean of distances to five Boston employment centers
- rad: index of accessibility to radial highways
- tax: full-value property-tax rate per $10,000
- ptratio: pupil-teacher ratio by town
- lstat: lower status of the population (percent)
- medv: median value of owner-occupied homes in $1000s
- crime: whether the crime rate is above the median crime rate (1) or not (0)

We will drop taxes (because they are 90% correlated with radial highways) and create a new binary variable, HighPTRatio (pupil teacher ratio above the mean = 1), so that we can perform a binary analysis.

B. The Relationship Between Distribution and Algorithm Effectiveness When we examine histograms we see that a number of variables have distributions that are broken and uneven (zn, indus, nox and rad), suggesting possible hidden groupings. As we will see later, this may lend itself well to decision tree/random forest algorithms. Many of the distributions are also skewed and we can see some likely outliers. However, both models are robust to outliers so we don't do transformations here.

```
library(psych)
```

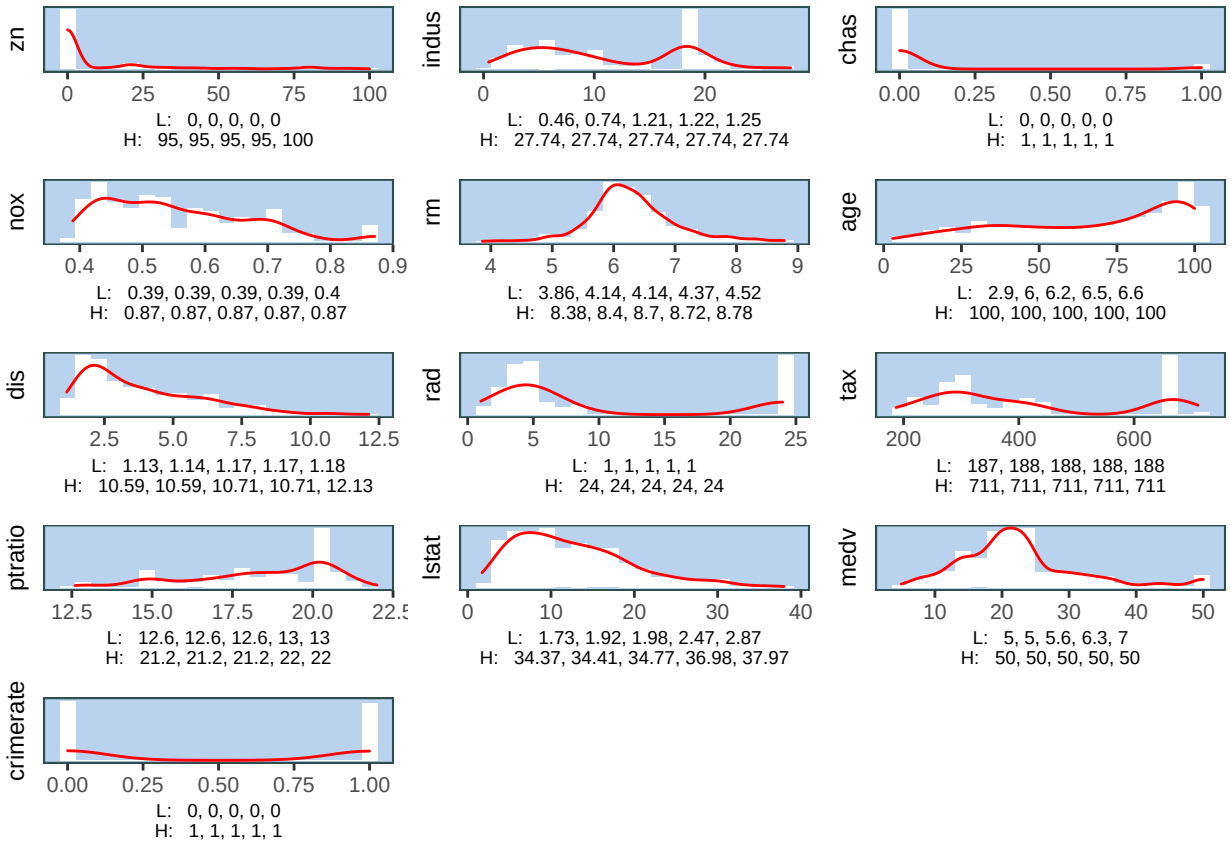
```
## Warning: package 'psych' was built under R version 4.0.5
```

```
##
## Attaching package: 'psych'
```

```
## The following object is masked from 'package:car':
##
##     logit
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

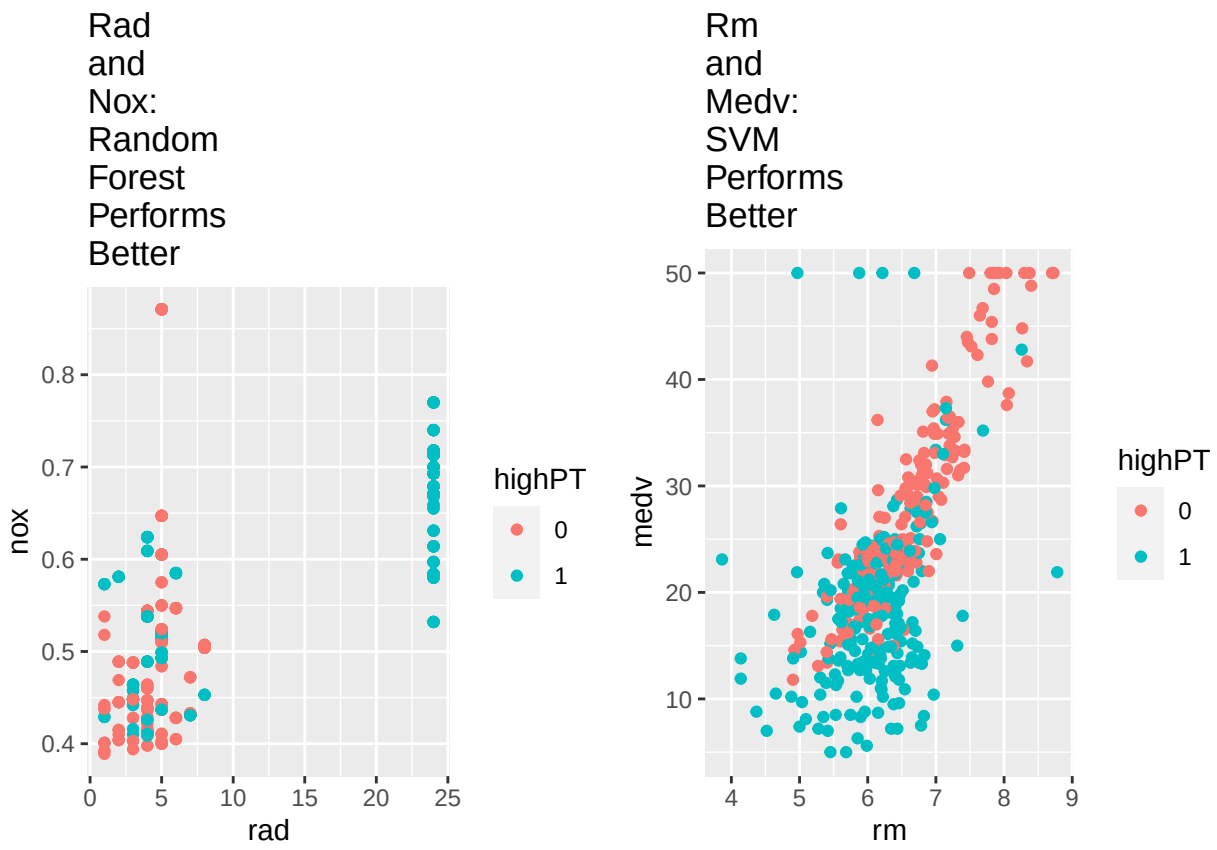
```
a <- EHSummarize_SingleColumn_Histograms(df1)
grid.arrange(grobs=a, ncol=3, nrow=5)
```



When we examine some of these broken distributions further, we see the existence of idiosyncratic cohorts in the relationship with PTRatio. In particluar, when we look at a scatterplot for PTRatio on two broken distributions (nox and rad) vs two smooth ones (medv and rm), we see two distinctly different data shapes underlying this dataset. This will become a factor later when it appears that Random Forest handles these idiosyncratic regions better than SVM, and that SVM handles the smoother regions better.

```
df2 <- df1 %>%
  dplyr:: select(-tax)

dfAll <- df2 %>%
  mutate(highPT = as.numeric(ifelse(ptratio>18.4, 1, 0))) %>%
    dplyr::select(-ptratio)

df3x <- dfAll %>%
  mutate(highPT = as.factor(highPT))

df3Nox <- dfAll %>%
  dplyr::select(highPT, nox)

g2 <- ggplot(df3x, aes(x=rad, y=nox, color=highPT)) +
  geom_point() +
    labs(title = str_wrap("Rad and Nox: Random Forest Performs Better", 5))

g3 <- ggplot(df3x, aes(x=rm, y=medv, color=highPT)) +
  geom_point() +
    labs(title = str_wrap("Rm and Medv: SVM Performs Better", 5))
```

```
grid.arrange(g2, g3, ncol=2)
```



C. Analysis We prepare various datasets for analysis:

```
dfRM_and_MEDV_only <- df2 %>%
  mutate(highPT= as.numeric(ifelse(ptratio>18.4, 1, 0))) %>%
  dplyr::select(highPT, rm, medv)

df3High_Nox_Filtered_Out <- dfAll %>%
  filter(nox<=.65)

df3Sparse <- dfAll %>%
 filter(row_number() %% 5 == 1) %>%
  dplyr::select(-chas)

df3CohortsLabeled <- dfAll %>%
  mutate(highNoxCohort = as.numeric(ifelse(nox>.65, 1, 0))) %>%
  mutate(highRadCohort = as.numeric(ifelse(rad>20, 1, 0)))
```

```
RunAnalyses <- function(df, title="") {

devtools::install_github("ericonsi/EHData", force = TRUE)
library(EHData)
```

```
library("caret")
library(tidyverse)
library(ggsci)
library(EHData)
library(patchwork)
library(gridExtra)
library(pROC)
library(car)

dfq <- df

print (" ")
print ("SVM - LINEAR")
print (" ")
a <- EHModel_SVM(dfq, "highPT", method="linear", printPlot = FALSE,  printSVM = TRUE, printConfusionMat
print (" ")
print ("SVM - RADIAL")
print (" ")
b <- EHModel_SVM(dfq, "highPT", method="radial", printSVM =  TRUE, printPlot = FALSE, printConfusionMat
print (" ")
print ("SVM - POLY")
print (" ")
c <- EHModel_SVM(dfq, "highPT", method="poly", printPlot=FALSE, printSVM = TRUE, printConfusionMatrix =
print (" ")
print ("SVM - RANDOM FOREST")
print (" ")
d <- EHModel_RandomForest(dfq, "highPT", categorical = TRUE, printPlot=FALSE, printRF = TRUE, printConf
print (" ")
print ("DECISION TREE")
print (" ")
e <- EHModel_DecisionTree(dfq, "highPT", categorical=TRUE, printDT=TRUE, printFancyTree = FALSE, printC
#f <- EHModel_Regression_Logistic(dfq, "highPT")

#devtools::install_github("ericonsi/EHData", force = TRUE)
library(EHData)
  library(caTools)
  library(ROCR)

a1 <- EHCalculate_AUC_ForBinaryClasses(a$errors, printPlot=FALSE, printConfusionMatrix = FALSE)
b1 <- EHCalculate_AUC_ForBinaryClasses(b$errors, printPlot=FALSE, printConfusionMatrix = FALSE)
c1 <- EHCalculate_AUC_ForBinaryClasses(c$errors, printPlot=FALSE, printConfusionMatrix = FALSE)
d1 <- EHCalculate_AUC_ForBinaryClasses(d$errors, printPlot=FALSE, printConfusionMatrix = FALSE)
e1 <- EHCalculate_AUC_ForBinaryClasses(e$errors, printPlot=FALSE, printConfusionMatrix = FALSE)

tab <- matrix(c(a1$AUC, a1$ConfusionMatrix$overall["Accuracy"], b1$AUC, b1$ConfusionMatrix$overall["Accu
colnames(tab) <- c('AUC','Accuracy')
rownames(tab) <- c('SVM - linear','SVM - radial','SVM - poly', 'Random Forest', 'Decision Tree')
dfTab <- as.data.frame(tab) %>%
  arrange(desc(AUC))

q <- knitr::kable(dfTab, desc=TRUE, caption=title, digits=2)
q
```

```
return(q)

}
```

We run three varieties of SVM (linear, radial and poly), as well as Random Forest and Decision Tree on several datasets and compare performance. Hyperparameters have been tuned and algorithms maximized using a grid approach.

```
an1 <- RunAnalyses(dfAll, "Predicting PTRatio, Full Dataset")
```

1. The Full Dataset

```
## Downloading GitHub repo ericonsi/EHData@HEAD

##          checking for file 'C:\Users\Eric\AppData\Local\Temp\Rtmp8kraBx\remotes21fc4551b4e\ericonsi-
##       -  preparing 'EHData':
##    checking DESCRIPTION meta-information ...     checking DESCRIPTION meta-information ...   v  check
##       -  checking for LF line-endings in source and make files and shell scripts
##   -  checking for empty or unneeded directories
##   -  creating default NAMESPACE file
##   -  building 'EHData_0.1.0.tar.gz'
##
##

## Warning: package 'EHData' is in use and will not be installed

## [1] " "
## [1] "SVM - LINEAR"
## [1] " "
## Support Vector Machines with Linear Kernel
##
## 374 samples
##  11 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 337, 337, 337, 335, 337, 336, ...
## Resampling results across tuning parameters:
##
##   C          Accuracy   Kappa
##   0.0100000  0.7121038  0.4201827
##   0.1147368  0.7406007  0.4659236
##   0.2194737  0.7360974  0.4580579
##   0.3242105  0.7290076  0.4440101
##   0.4289474  0.7316628  0.4501167
##   0.5336842  0.7352664  0.4568860
##   0.6384211  0.7316865  0.4497515
##   0.7431579  0.7325412  0.4511485
```

```
##   0.8478947  0.7307619  0.4470459
##   0.9526316  0.7325400  0.4505215
##   1.0573684  0.7334409  0.4521702
##   1.1621053  0.7352427  0.4559064
##   1.2668421  0.7334646  0.4519200
##   1.3715789  0.7326112  0.4501941
##   1.4763158  0.7317103  0.4482318
##   1.5810526  0.7326112  0.4499971
##   1.6857895  0.7334883  0.4519069
##   1.7905263  0.7352427  0.4554572
##   1.8952632  0.7352202  0.4554292
##   2.0000000  0.7334421  0.4516096
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.1147368.
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 29  4
##          1 11 48
##
##                Accuracy : 0.837
##                  95% CI : (0.7454, 0.9058)
##     No Information Rate : 0.5652
##     P-Value [Acc > NIR] : 2.707e-08
##
##                   Kappa : 0.6614
##
##  Mcnemar's Test P-Value : 0.1213
##
##             Sensitivity : 0.7250
##             Specificity : 0.9231
##          Pos Pred Value : 0.8788
##          Neg Pred Value : 0.8136
##              Prevalence : 0.4348
##          Detection Rate : 0.3152
##    Detection Prevalence : 0.3587
##       Balanced Accuracy : 0.8240
##
##        'Positive' Class : 0
##
## [1] " "
## [1] "SVM - RADIAL"
## [1] " "
## Support Vector Machines with Radial Basis Function Kernel
##
## 374 samples
##  11 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 337, 337, 337, 335, 337, 336, ...
```

```
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.7560583  0.5057840
##   0.50  0.7738155  0.5391238
##   1.00  0.7970019  0.5855073
##
## Tuning parameter 'sigma' was held constant at a value of 0.1412225
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.1412225 and C = 1.
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 30  4
##          1 10 48
##
##                Accuracy : 0.8478
##                  95% CI : (0.7579, 0.9142)
##     No Information Rate : 0.5652
##     P-Value [Acc > NIR] : 6.602e-09
##
##                   Kappa : 0.6849
##
##  Mcnemar's Test P-Value : 0.1814
##
##             Sensitivity : 0.7500
##             Specificity : 0.9231
##          Pos Pred Value : 0.8824
##          Neg Pred Value : 0.8276
##              Prevalence : 0.4348
##          Detection Rate : 0.3261
##    Detection Prevalence : 0.3696
##       Balanced Accuracy : 0.8365
##
##        'Positive' Class : 0
##
## [1] " "
## [1] "SVM - POLY"
## [1] " "
## Support Vector Machines with Polynomial Kernel
##
## 374 samples
##  11 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 337, 337, 337, 335, 337, 336, ...
## Resampling results across tuning parameters:
##
##   degree  scale  C     Accuracy   Kappa
##   1       0.001  0.25  0.5642278  0.00000000
##   1       0.001  0.50  0.5686387  0.01142828
```

```
##   1       0.001  1.00  0.6811589  0.31852946
##   1       0.010  0.25  0.7113202  0.41732616
##   1       0.010  0.50  0.7085227  0.41389105
##   1       0.010  1.00  0.7121038  0.42018267
##   1       0.100  0.25  0.7237657  0.43280500
##   1       0.100  0.50  0.7451289  0.47559060
##   1       0.100  1.00  0.7424262  0.47010014
##   2       0.001  0.25  0.5686387  0.01142828
##   2       0.001  0.50  0.6811589  0.31852946
##   2       0.001  1.00  0.7085938  0.40962509
##   2       0.010  0.25  0.7147591  0.42553378
##   2       0.010  0.50  0.7201882  0.43370787
##   2       0.010  1.00  0.7256137  0.43810525
##   2       0.100  0.25  0.7577403  0.50150193
##   2       0.100  0.50  0.7558674  0.50220871
##   2       0.100  1.00  0.7621512  0.51529815
##   3       0.001  0.25  0.6178077  0.14646116
##   3       0.001  0.50  0.6935369  0.36781369
##   3       0.001  1.00  0.7139044  0.42245147
##   3       0.010  0.25  0.7174381  0.42931276
##   3       0.010  0.50  0.7255912  0.43981979
##   3       0.010  1.00  0.7585239  0.50124024
##   3       0.100  0.25  0.7666320  0.52234444
##   3       0.100  0.50  0.7782015  0.54595099
##   3       0.100  1.00  0.7747152  0.53866549
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were degree = 3, scale = 0.1 and C = 0.5.
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 29  3
##          1 11 49
##
##                Accuracy : 0.8478
##                  95% CI : (0.7579, 0.9142)
##     No Information Rate : 0.5652
##     P-Value [Acc > NIR] : 6.602e-09
##
##                   Kappa : 0.6831
##
##  Mcnemar's Test P-Value : 0.06137
##
##             Sensitivity : 0.7250
##             Specificity : 0.9423
##          Pos Pred Value : 0.9062
##          Neg Pred Value : 0.8167
##              Prevalence : 0.4348
##          Detection Rate : 0.3152
##    Detection Prevalence : 0.3478
##       Balanced Accuracy : 0.8337
##
##        'Positive' Class : 0
```

```
##
## [1] " "
## [1] "SVM - RANDOM FOREST"
## [1] " "
## Random Forest
##
## 374 samples
##  11 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 337, 337, 337, 335, 337, 336, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9390324  0.8784536
##    6    0.9496298  0.8996511
##   11    0.9550352  0.9104231
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 11.
## rf variable importance
##
##            Overall
## indus     100.0000
## rad        95.1269
## nox        47.9100
## dis        42.3349
## medv       25.0556
## zn         14.8095
## rm         12.0677
## age        10.6848
## lstat       9.9190
## crimerate   0.5005
## chas        0.0000
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 39  1
##          1  1 51
##
##                Accuracy : 0.9783
##                  95% CI : (0.9237, 0.9974)
##     No Information Rate : 0.5652
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9558
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9750
##             Specificity : 0.9808
```

```
##          Pos Pred Value : 0.9750
##          Neg Pred Value : 0.9808
##              Prevalence : 0.4348
##          Detection Rate : 0.4239
##    Detection Prevalence : 0.4348
##       Balanced Accuracy : 0.9779
##
##        'Positive' Class : 0
##
## [1] "Parameters:   mtry =  11 , ntree =  500 , nrnodes =  79"
## [1] " "
## [1] "DECISION TREE"
## [1] " "
```

```
## Warning: package 'rpart.plot' was built under R version 4.0.5
```

```
## Warning: package 'rattle' was built under R version 4.0.5
```

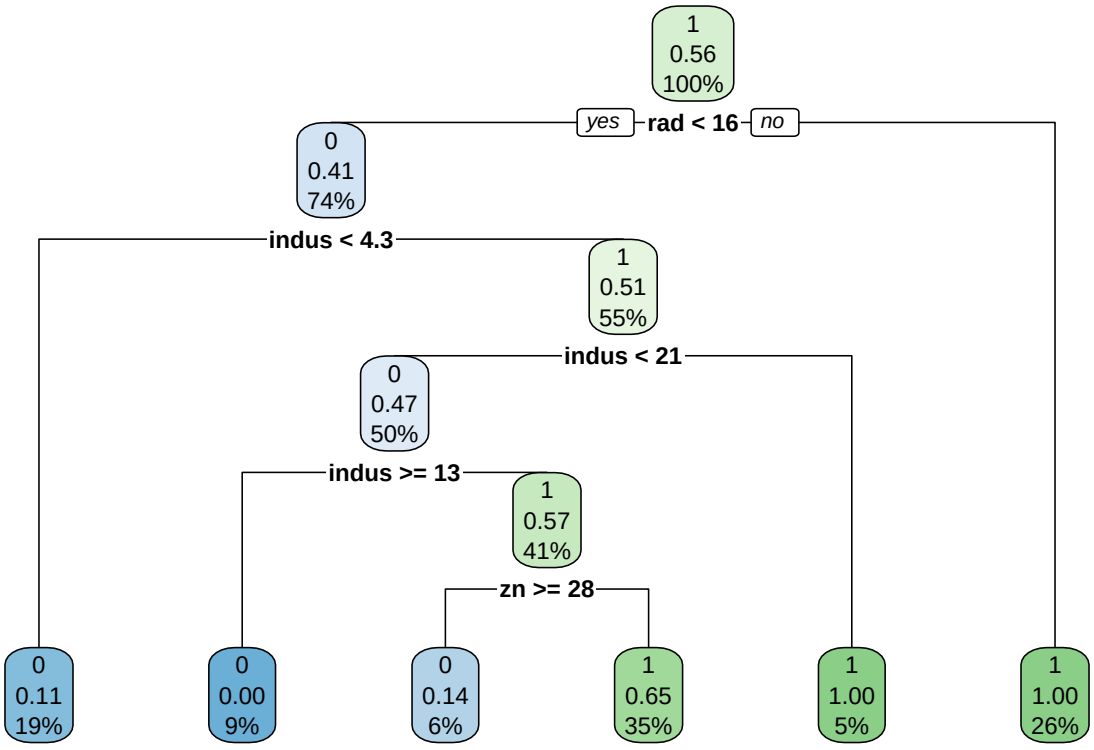
```
## Loading required package: bitops
```

```
## Warning: package 'bitops' was built under R version 4.0.5
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 29  1
##          1 11 51
##
##                Accuracy : 0.8696
##                  95% CI : (0.7832, 0.9307)
##     No Information Rate : 0.5652
##     P-Value [Acc > NIR] : 3.068e-10
##
##                   Kappa : 0.7267
##
##  Mcnemar's Test P-Value : 0.009375
##
##             Sensitivity : 0.7250
##             Specificity : 0.9808
##          Pos Pred Value : 0.9667
##          Neg Pred Value : 0.8226
##              Prevalence : 0.4348
##          Detection Rate : 0.3152
##    Detection Prevalence : 0.3261
##       Balanced Accuracy : 0.8529
##
##        'Positive' Class : 0
##
```

```
## Warning: package 'caTools' was built under R version 4.0.5

## Warning: package 'ROCR' was built under R version 4.0.5

## Setting levels: control = 1, case = 2

## Setting direction: controls < cases

## Setting levels: control = 1, case = 2

## Setting direction: controls < cases

## Setting levels: control = 1, case = 2

## Setting direction: controls < cases

## Setting levels: control = 1, case = 2

## Setting direction: controls < cases

## Setting levels: control = 1, case = 2

## Setting direction: controls < cases
```



```
print(an1)
```

```
##
##
## Table: Predicting PTRatio, Full Dataset
##
## |              |  AUC| Accuracy|
## |:-------------|----:|--------:|
## |Random Forest | 0.98|     0.98|
## |Decision Tree | 0.85|     0.87|
## |SVM - radial  | 0.84|     0.85|
## |SVM - poly    | 0.83|     0.85|
## |SVM - linear  | 0.82|     0.84|
```

Both decision tree algorithms are the highest performers, with Random Forest showing 97% accuracy. Rad and Nox figure prominently in the analysis.

```
an1a <- RunAnalyses(df3Nox, "Prediciting PTRatio: NOX Only")
```

1. NOX Only

```
## Downloading GitHub repo ericonsi/EHData@HEAD
```

```
##          checking for file 'C:\Users\Eric\AppData\Local\Temp\Rtmp8kraBx\remotes21fc53983556\ericonsi
##       -  preparing 'EHData':
##    checking DESCRIPTION meta-information ...     checking DESCRIPTION meta-information ...   v  check
##       -  checking for LF line-endings in source and make files and shell scripts
## -  checking for empty or unneeded directories
##   -  creating default NAMESPACE file
##   -  building 'EHData_0.1.0.tar.gz'
##
##
```

```
## Warning: package 'EHData' is in use and will not be installed
```

```
## [1] " "
## [1] "SVM - LINEAR"
## [1] " "
## Support Vector Machines with Linear Kernel
##
## 374 samples
##   1 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (1), scaled (1)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 337, 337, 337, 335, 337, 336, ...
## Resampling results across tuning parameters:
##
```

```
##   C          Accuracy   Kappa
##   0.0100000  0.5642278  0.0000000
##   0.1147368  0.6331479  0.2722886
##   0.2194737  0.6474438  0.2984170
##   0.3242105  0.6429393  0.2897589
##   0.4289474  0.6429393  0.2897589
##   0.5336842  0.6429393  0.2897589
##   0.6384211  0.6429393  0.2897589
##   0.7431579  0.6438165  0.2915440
##   0.8478947  0.6474201  0.2979424
##   0.9526316  0.6474201  0.2979424
##   1.0573684  0.6510237  0.3045085
##   1.1621053  0.6510237  0.3045085
##   1.2668421  0.6510237  0.3045085
##   1.3715789  0.6510237  0.3045085
##   1.4763158  0.6510237  0.3045085
##   1.5810526  0.6510237  0.3045085
##   1.6857895  0.6519246  0.3061320
##   1.7905263  0.6519246  0.3061320
##   1.8952632  0.6519246  0.3061320
##   2.0000000  0.6519246  0.3061320
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 1.685789.
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 27 16
##          1 13 36
##
##                Accuracy : 0.6848
##                  95% CI : (0.5796, 0.7777)
##     No Information Rate : 0.5652
##     P-Value [Acc > NIR] : 0.01271
##
##                   Kappa : 0.3642
##
##  Mcnemar's Test P-Value : 0.71035
##
##             Sensitivity : 0.6750
##             Specificity : 0.6923
##          Pos Pred Value : 0.6279
##          Neg Pred Value : 0.7347
##              Prevalence : 0.4348
##          Detection Rate : 0.2935
##    Detection Prevalence : 0.4674
##       Balanced Accuracy : 0.6837
##
##        'Positive' Class : 0
##
## [1] " "
## [1] "SVM - RADIAL"
## [1] " "
```

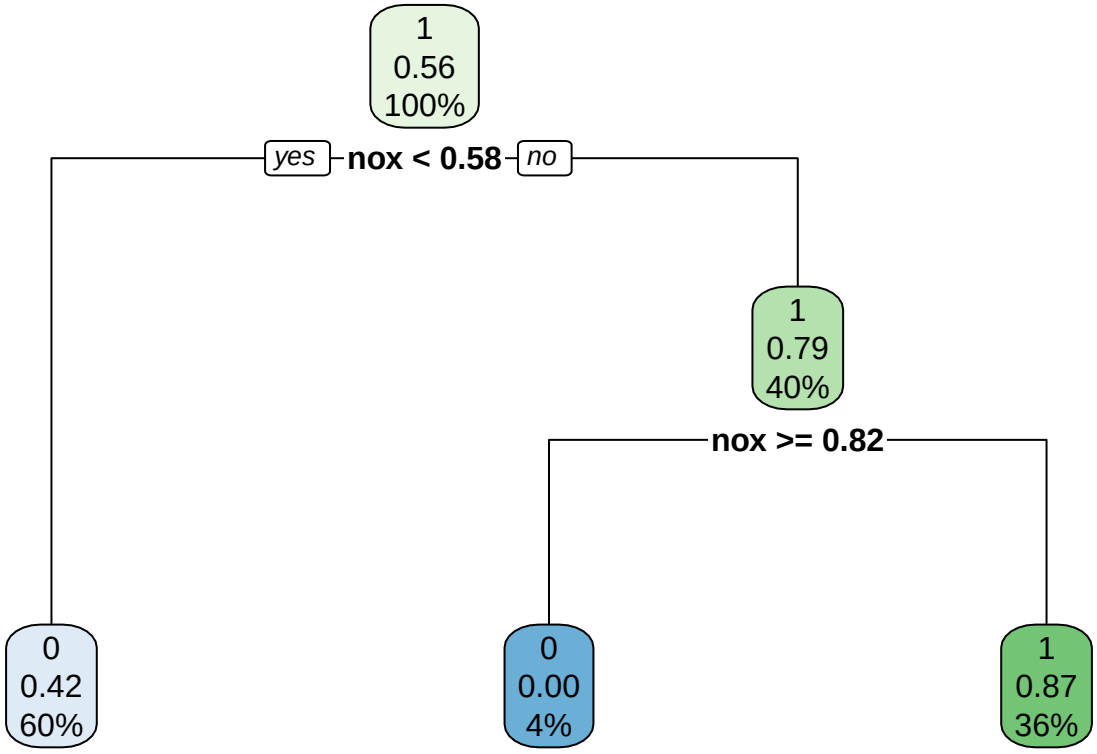
```
## Support Vector Machines with Radial Basis Function Kernel
##
## 374 samples
##   1 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (1), scaled (1)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 337, 337, 337, 335, 337, 336, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.6689469  0.3427597
##   0.50  0.6635415  0.3343851
##   1.00  0.6788094  0.3641411
##
## Tuning parameter 'sigma' was held constant at a value of 12.29308
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 12.29308 and C = 1.
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 36 13
##          1  4 39
##
##                Accuracy : 0.8152
##                  95% CI : (0.7207, 0.8885)
##     No Information Rate : 0.5652
##     P-Value [Acc > NIR] : 3.634e-07
##
##                   Kappa : 0.6336
##
##  Mcnemar's Test P-Value : 0.05235
##
##             Sensitivity : 0.9000
##             Specificity : 0.7500
##          Pos Pred Value : 0.7347
##          Neg Pred Value : 0.9070
##              Prevalence : 0.4348
##          Detection Rate : 0.3913
##    Detection Prevalence : 0.5326
##       Balanced Accuracy : 0.8250
##
##        'Positive' Class : 0
##
## [1] " "
## [1] "SVM - POLY"
## [1] " "
## Support Vector Machines with Polynomial Kernel
##
## 374 samples
##   1 predictor
##   2 classes: '0', '1'
```

```
##
## Pre-processing: centered (1), scaled (1)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 337, 337, 337, 335, 337, 336, ...
## Resampling results across tuning parameters:
##
##   degree  scale  C     Accuracy   Kappa
##   1       0.001  0.25  0.5642278  0.00000000
##   1       0.001  0.50  0.5642278  0.00000000
##   1       0.001  1.00  0.5642278  0.00000000
##   1       0.010  0.25  0.5642278  0.00000000
##   1       0.010  0.50  0.5642278  0.00000000
##   1       0.010  1.00  0.5642278  0.00000000
##   1       0.100  0.25  0.6069756  0.17747369
##   1       0.100  0.50  0.6358956  0.27815507
##   1       0.100  1.00  0.6278374  0.26285804
##   2       0.001  0.25  0.5642278  0.00000000
##   2       0.001  0.50  0.5642278  0.00000000
##   2       0.001  1.00  0.5642278  0.00000000
##   2       0.010  0.25  0.5642278  0.00000000
##   2       0.010  0.50  0.5642278  0.00000000
##   2       0.010  1.00  0.5963557  0.12329024
##   2       0.100  0.25  0.6350446  0.24470861
##   2       0.100  0.50  0.6536103  0.27602405
##   2       0.100  1.00  0.6884159  0.35471512
##   3       0.001  0.25  0.5642278  0.00000000
##   3       0.001  0.50  0.5642278  0.00000000
##   3       0.001  1.00  0.5642278  0.00000000
##   3       0.010  0.25  0.5642278  0.00000000
##   3       0.010  0.50  0.5678788  0.02040772
##   3       0.010  1.00  0.6321060  0.24361799
##   3       0.100  0.25  0.6875150  0.35269394
##   3       0.100  0.50  0.6883448  0.35698041
##   3       0.100  1.00  0.6751608  0.33548542
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were degree = 2, scale = 0.1 and C = 1.
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 18  6
##          1 22 46
##
##                Accuracy : 0.6957
##                  95% CI : (0.591, 0.7873)
##     No Information Rate : 0.5652
##     P-Value [Acc > NIR] : 0.007080
##
##                   Kappa : 0.3508
##
##  Mcnemar's Test P-Value : 0.004586
##
##             Sensitivity : 0.4500
```

```
##             Specificity : 0.8846
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 0.6765
##              Prevalence : 0.4348
##          Detection Rate : 0.1957
##    Detection Prevalence : 0.2609
##       Balanced Accuracy : 0.6673
##
##        'Positive' Class : 0
##
## [1] " "
## [1] "SVM - RANDOM FOREST"
## [1] " "
## Random Forest
##
## 374 samples
##   1 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 337, 337, 337, 335, 337, 336, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9250319  0.8485841
##
## Tuning parameter 'mtry' was held constant at a value of 2
## rf variable importance

## Warning in FUN(newX[, i], ...): no non-missing arguments to max; returning -Inf

##     Overall
## nox     NaN
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 36  2
##          1  4 50
##
##                Accuracy : 0.9348
##                  95% CI : (0.8634, 0.9757)
##     No Information Rate : 0.5652
##     P-Value [Acc > NIR] : 2.59e-15
##
##                   Kappa : 0.8665
##
##  Mcnemar's Test P-Value : 0.6831
##
##             Sensitivity : 0.9000
##             Specificity : 0.9615
##          Pos Pred Value : 0.9474
##          Neg Pred Value : 0.9259
```

```
##              Prevalence : 0.4348
##          Detection Rate : 0.3913
##    Detection Prevalence : 0.4130
##       Balanced Accuracy : 0.9308
##
##        'Positive' Class : 0
##
## [1] "Parameters:   mtry =  1 , ntree =  500 , nrnodes =  73"
## [1] " "
## [1] "DECISION TREE"
## [1] " "

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 36 23
##          1  4 29
##
##                Accuracy : 0.7065
##                  95% CI : (0.6024, 0.7969)
##     No Information Rate : 0.5652
##     P-Value [Acc > NIR] : 0.003764
##
##                   Kappa : 0.4339
##
##  Mcnemar's Test P-Value : 0.000532
##
##             Sensitivity : 0.9000
##             Specificity : 0.5577
##          Pos Pred Value : 0.6102
##          Neg Pred Value : 0.8788
##              Prevalence : 0.4348
##          Detection Rate : 0.3913
##    Detection Prevalence : 0.6413
##       Balanced Accuracy : 0.7288
##
##        'Positive' Class : 0
##

## Setting levels: control = 1, case = 2

## Setting direction: controls < cases

## Setting levels: control = 1, case = 2

## Setting direction: controls < cases

## Setting levels: control = 1, case = 2

## Setting direction: controls < cases

## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases

## Setting levels: control = 1, case = 2

## Setting direction: controls < cases
```




```
print(an1a)
```

```
##
##
## Table: Prediciting PTRatio: NOX Only
##
## |              |  AUC| Accuracy|
## |:-------------|----:|--------:|
## |Random Forest | 0.93|     0.93|
## |SVM - radial  | 0.82|     0.82|
## |Decision Tree | 0.73|     0.71|
## |SVM - linear  | 0.68|     0.68|
## |SVM - poly    | 0.67|     0.70|
```

Even using NOX alone, Random Forest achieves a 93% accuracy rate.

```
an2 <-RunAnalyses(df3High_Nox_Filtered_Out, "Predicting PTRatio: High Nox Removed")
```

2. High NOX Removed

```
## Downloading GitHub repo ericonsi/EHData@HEAD

##          checking for file 'C:\Users\Eric\AppData\Local\Temp\Rtmp8kraBx\remotes21fc6acd4b31\ericonsi
##       -  preparing 'EHData':
##    checking DESCRIPTION meta-information ...     checking DESCRIPTION meta-information ...   v  chec
##       -  checking for LF line-endings in source and make files and shell scripts
## -  checking for empty or unneeded directories
##   -  creating default NAMESPACE file
##   -  building 'EHData_0.1.0.tar.gz'
##
##

## Warning: package 'EHData' is in use and will not be installed

## [1] " "
## [1] "SVM - LINEAR"
## [1] " "
## Support Vector Machines with Linear Kernel
##
## 294 samples
##  11 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 265, 265, 265, 265, 264, 264, ...
## Resampling results across tuning parameters:
##
##   C          Accuracy   Kappa
##   0.0100000  0.6952874  0.3915401
##   0.1147368  0.6806130  0.3632806
##   0.2194737  0.6851341  0.3721660
##   0.3242105  0.6851341  0.3722630
##   0.4289474  0.6851341  0.3722630
##   0.5336842  0.6885441  0.3789806
##   0.6384211  0.6908046  0.3835534
##   0.7431579  0.6919157  0.3857756
##   0.8478947  0.6919157  0.3857756
##   0.9526316  0.6919157  0.3857756
##   1.0573684  0.6919157  0.3857756
##   1.1621053  0.6919157  0.3857756
##   1.2668421  0.6919157  0.3857756
##   1.3715789  0.6908046  0.3835534
##   1.4763158  0.6908046  0.3835534
##   1.5810526  0.6908046  0.3835534
##   1.6857895  0.6896935  0.3813311
##   1.7905263  0.6896935  0.3813311
```

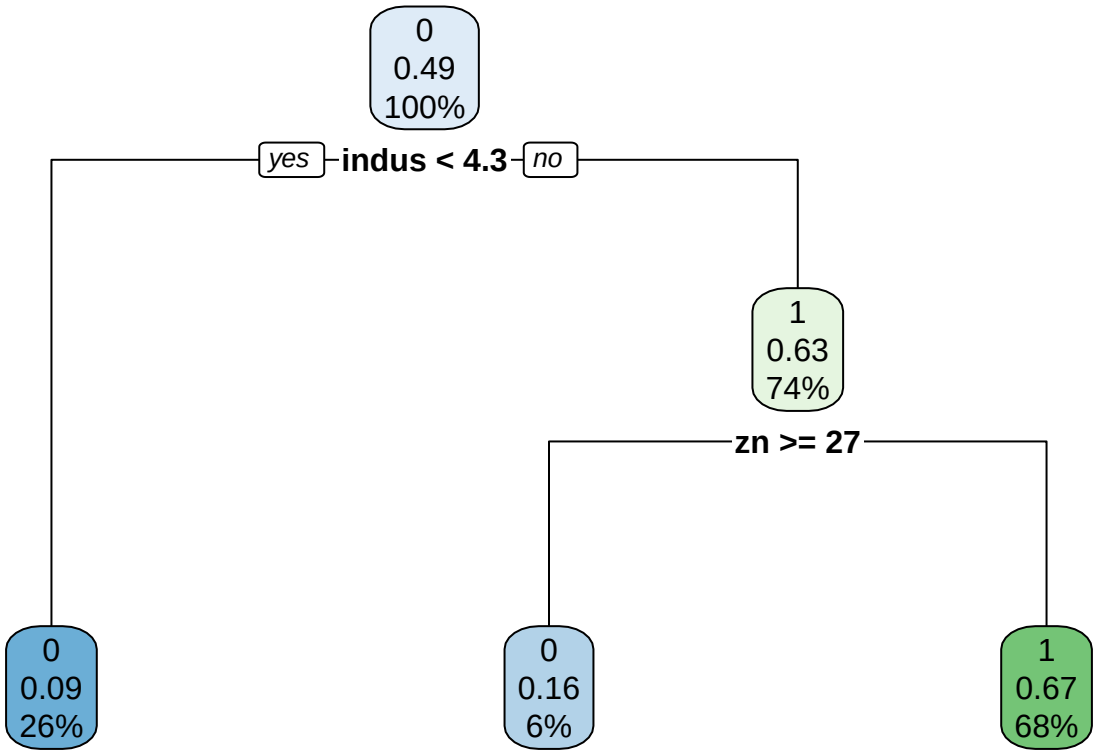
```
##   1.8952632  0.6896935  0.3813311
##   2.0000000  0.6896935  0.3813311
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.01.
## Confusion Matrix and Statistics
##
##          Reference
## Prediction  0  1
##          0 29  5
##          1  8 30
##
##                Accuracy : 0.8194
##                  95% CI : (0.7111, 0.9002)
##     No Information Rate : 0.5139
##     P-Value [Acc > NIR] : 6.749e-08
##
##                   Kappa : 0.6394
##
##  Mcnemar's Test P-Value : 0.5791
##
##             Sensitivity : 0.7838
##             Specificity : 0.8571
##          Pos Pred Value : 0.8529
##          Neg Pred Value : 0.7895
##              Prevalence : 0.5139
##          Detection Rate : 0.4028
##    Detection Prevalence : 0.4722
##       Balanced Accuracy : 0.8205
##
##        'Positive' Class : 0
##
## [1] " "
## [1] "SVM - RADIAL"
## [1] " "
## Support Vector Machines with Radial Basis Function Kernel
##
## 294 samples
##  11 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 265, 265, 265, 265, 264, 264, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.7146743  0.4310351
##   0.50  0.7212644  0.4436817
##   1.00  0.7360153  0.4725963
##
## Tuning parameter 'sigma' was held constant at a value of 0.1140188
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.1140188 and C = 1.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 24  6
##          1 13 29
##
##                Accuracy : 0.7361
##                  95% CI : (0.619, 0.833)
##     No Information Rate : 0.5139
##     P-Value [Acc > NIR] : 9.722e-05
##
##                   Kappa : 0.4747
##
##  Mcnemar's Test P-Value : 0.1687
##
##             Sensitivity : 0.6486
##             Specificity : 0.8286
##          Pos Pred Value : 0.8000
##          Neg Pred Value : 0.6905
##              Prevalence : 0.5139
##          Detection Rate : 0.3333
##    Detection Prevalence : 0.4167
##       Balanced Accuracy : 0.7386
##
##        'Positive' Class : 0
##
## [1] " "
## [1] "SVM - POLY"
## [1] " "
## Support Vector Machines with Polynomial Kernel
##
## 294 samples
##  11 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 265, 265, 265, 265, 264, 264, ...
## Resampling results across tuning parameters:
##
##   degree  scale  C     Accuracy   Kappa
##   1       0.001  0.25  0.5103448  0.0000000
##   1       0.001  0.50  0.5103448  0.0000000
##   1       0.001  1.00  0.6562069  0.3071751
##   1       0.010  0.25  0.6766667  0.3519644
##   1       0.010  0.50  0.6888123  0.3774108
##   1       0.010  1.00  0.6952874  0.3915401
##   1       0.100  0.25  0.6953257  0.3934293
##   1       0.100  0.50  0.6942146  0.3908206
##   1       0.100  1.00  0.6862835  0.3745481
##   2       0.001  0.25  0.5103448  0.0000000
##   2       0.001  0.50  0.6562069  0.3071751
##   2       0.001  1.00  0.6654023  0.3289505
```

```
##   2       0.010  0.25  0.6855172  0.3703699
##   2       0.010  0.50  0.6977011  0.3961071
##   2       0.010  1.00  0.7032950  0.4088891
##   2       0.100  0.25  0.7088123  0.4189675
##   2       0.100  0.50  0.7155939  0.4321508
##   2       0.100  1.00  0.7280460  0.4567480
##   3       0.001  0.25  0.6014176  0.1905505
##   3       0.001  0.50  0.6527969  0.3023135
##   3       0.001  1.00  0.6778161  0.3541953
##   3       0.010  0.25  0.6931418  0.3861536
##   3       0.010  0.50  0.7010345  0.4037434
##   3       0.010  1.00  0.7010728  0.4045462
##   3       0.100  0.25  0.7471648  0.4945381
##   3       0.100  0.50  0.7495402  0.4990079
##   3       0.100  1.00  0.7494253  0.4986417
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were degree = 3, scale = 0.1 and C = 0.5.
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 27  8
##          1 10 27
##
##                Accuracy : 0.75
##                  95% CI : (0.634, 0.8446)
##     No Information Rate : 0.5139
##     P-Value [Acc > NIR] : 3.485e-05
##
##                   Kappa : 0.5004
##
##  Mcnemar's Test P-Value : 0.8137
##
##             Sensitivity : 0.7297
##             Specificity : 0.7714
##          Pos Pred Value : 0.7714
##          Neg Pred Value : 0.7297
##              Prevalence : 0.5139
##          Detection Rate : 0.3750
##    Detection Prevalence : 0.4861
##       Balanced Accuracy : 0.7506
##
##        'Positive' Class : 0
##
## [1] " "
## [1] "SVM - RANDOM FOREST"
## [1] " "
## Random Forest
##
## 294 samples
##  11 predictor
##   2 classes: '0', '1'
##
```

```
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 265, 265, 265, 265, 264, 264, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9390805  0.8780949
##    6    0.9627586  0.9253936
##   11    0.9525287  0.9050490
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 6.
## rf variable importance
##
##           Overall
## indus     100.000
## nox        47.806
## dis        45.773
## medv       42.094
## lstat      25.300
## rad        25.016
## rm         24.933
## zn         22.398
## age        19.122
## crimerate   1.775
## chas        0.000
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 34  2
##          1  3 33
##
##                Accuracy : 0.9306
##                  95% CI : (0.8453, 0.9771)
##     No Information Rate : 0.5139
##     P-Value [Acc > NIR] : 1.747e-14
##
##                   Kappa : 0.8611
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9189
##             Specificity : 0.9429
##          Pos Pred Value : 0.9444
##          Neg Pred Value : 0.9167
##              Prevalence : 0.5139
##          Detection Rate : 0.4722
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.9309
##
##        'Positive' Class : 0
##
## [1] "Parameters:   mtry =  6 , ntree =  500 , nrnodes =  97"
```

```
## [1] " "
## [1] "DECISION TREE"
## [1] " "

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 24  2
##          1 13 33
##
##                Accuracy : 0.7917
##                  95% CI : (0.6798, 0.8784)
##     No Information Rate : 0.5139
##     P-Value [Acc > NIR] : 1.041e-06
##
##                   Kappa : 0.5865
##
##  Mcnemar's Test P-Value : 0.009823
##
##             Sensitivity : 0.6486
##             Specificity : 0.9429
##          Pos Pred Value : 0.9231
##          Neg Pred Value : 0.7174
##              Prevalence : 0.5139
##          Detection Rate : 0.3333
##    Detection Prevalence : 0.3611
##       Balanced Accuracy : 0.7958
##
##        'Positive' Class : 0
##

## Setting levels: control = 1, case = 2

## Setting direction: controls < cases

## Setting levels: control = 1, case = 2

## Setting direction: controls < cases

## Setting levels: control = 1, case = 2

## Setting direction: controls < cases

## Setting levels: control = 1, case = 2

## Setting direction: controls < cases

## Setting levels: control = 1, case = 2

## Setting direction: controls < cases
```




```
print(an2)
```

```
##
##
## Table: Predicting PTRatio: High Nox Removed
##
## |               |  AUC| Accuracy|
## |:--------------|----:|--------:|
## |Random Forest  | 0.93|     0.93|
## |SVM - linear   | 0.82|     0.82|
## |Decision Tree  | 0.80|     0.79|
## |SVM - poly     | 0.75|     0.75|
## |SVM - radial   | 0.74|     0.74|
```

Now the performance for Random Forest drops considerably. For SVM there is little change. SVM is now the second best algorithm.

```
an3 <-RunAnalyses(dfRM_and_MEDV_only, "Predicting PTRatio: RM and MEDV Only")
```

3. RM and MEDV Only

```
## Downloading GitHub repo ericonsi/EHData@HEAD
```

```
##          checking for file 'C:\Users\Eric\AppData\Local\Temp\Rtmp8kraBx\remotes21fc7ec810cf\ericonsi
##       -  preparing 'EHData':
##    checking DESCRIPTION meta-information ...     checking DESCRIPTION meta-information ...   v  check
##       -  checking for LF line-endings in source and make files and shell scripts
## -  checking for empty or unneeded directories
##   -  creating default NAMESPACE file
##   -  building 'EHData_0.1.0.tar.gz'
##
##

## Warning: package 'EHData' is in use and will not be installed

## [1] " "
## [1] "SVM - LINEAR"
## [1] " "
## Support Vector Machines with Linear Kernel
##
## 374 samples
##   2 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (2), scaled (2)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 337, 337, 337, 335, 337, 336, ...
## Resampling results across tuning parameters:
##
##   C          Accuracy   Kappa
##   0.0100000  0.6658235  0.2654267
##   0.1147368  0.6900518  0.3343811
##   0.2194737  0.6891746  0.3327628
##   0.3242105  0.6909527  0.3366573
##   0.4289474  0.6891746  0.3330857
##   0.5336842  0.6891746  0.3330857
##   0.6384211  0.6891983  0.3334329
##   0.7431579  0.6900755  0.3350512
##   0.8478947  0.6900755  0.3350512
##   0.9526316  0.6900755  0.3350512
##   1.0573684  0.6891983  0.3339090
##   1.1621053  0.6882974  0.3319558
##   1.2668421  0.6891983  0.3339090
##   1.3715789  0.6882974  0.3319558
##   1.4763158  0.6891983  0.3339090
##   1.5810526  0.6882974  0.3319558
##   1.6857895  0.6882974  0.3319558
##   1.7905263  0.6882974  0.3319558
##   1.8952632  0.6882974  0.3319558
##   2.0000000  0.6882974  0.3319558
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.3242105.
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
```

```
##          0 23  4
##          1 17 48
##
##                Accuracy : 0.7717
##                  95% CI : (0.6725, 0.8528)
##     No Information Rate : 0.5652
##     P-Value [Acc > NIR] : 2.925e-05
##
##                   Kappa : 0.5175
##
##  Mcnemar's Test P-Value : 0.008829
##
##             Sensitivity : 0.5750
##             Specificity : 0.9231
##          Pos Pred Value : 0.8519
##          Neg Pred Value : 0.7385
##              Prevalence : 0.4348
##          Detection Rate : 0.2500
##    Detection Prevalence : 0.2935
##       Balanced Accuracy : 0.7490
##
##        'Positive' Class : 0
##
## [1] " "
## [1] "SVM - RADIAL"
## [1] " "
## Support Vector Machines with Radial Basis Function Kernel
##
## 374 samples
##   2 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (2), scaled (2)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 337, 337, 337, 335, 337, 336, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.6908816  0.3467012
##   0.50  0.6908579  0.3511681
##   1.00  0.6863758  0.3438766
##
## Tuning parameter 'sigma' was held constant at a value of 3.78593
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 3.78593 and C = 0.25.
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 22  5
##          1 18 47
##
##                Accuracy : 0.75
##                  95% CI : (0.6489, 0.8345)
```

```
##     No Information Rate : 0.5652
##     P-Value [Acc > NIR] : 0.0001829
##
##                   Kappa : 0.4715
##
##  Mcnemar's Test P-Value : 0.0123434
##
##             Sensitivity : 0.5500
##             Specificity : 0.9038
##          Pos Pred Value : 0.8148
##          Neg Pred Value : 0.7231
##              Prevalence : 0.4348
##          Detection Rate : 0.2391
##    Detection Prevalence : 0.2935
##       Balanced Accuracy : 0.7269
##
##        'Positive' Class : 0
##
## [1] " "
## [1] "SVM - POLY"
## [1] " "
## Support Vector Machines with Polynomial Kernel
##
## 374 samples
##   2 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (2), scaled (2)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 337, 337, 337, 335, 337, 336, ...
## Resampling results across tuning parameters:
##
##   degree  scale  C     Accuracy   Kappa
##   1       0.001  0.25  0.5642278  0.000000000
##   1       0.001  0.50  0.5642278  0.000000000
##   1       0.001  1.00  0.5642278  0.000000000
##   1       0.010  0.25  0.6016437  0.098587811
##   1       0.010  0.50  0.6355710  0.190659516
##   1       0.010  1.00  0.6658235  0.265426682
##   1       0.100  0.25  0.6802130  0.310020449
##   1       0.100  0.50  0.6882737  0.329694120
##   1       0.100  1.00  0.6900518  0.334381139
##   2       0.001  0.25  0.5642278  0.000000000
##   2       0.001  0.50  0.5642278  0.000000000
##   2       0.001  1.00  0.5909728  0.069436395
##   2       0.010  0.25  0.6328920  0.184240808
##   2       0.010  0.50  0.6658235  0.264926444
##   2       0.010  1.00  0.6784112  0.303550747
##   2       0.100  0.25  0.6811376  0.313182846
##   2       0.100  0.50  0.6856184  0.323685613
##   2       0.100  1.00  0.6909053  0.336929717
##   3       0.001  0.25  0.5642278  0.000000000
##   3       0.001  0.50  0.5651287  0.002345059
##   3       0.001  1.00  0.6034930  0.107690733
```

```
##   3       0.010  0.25  0.6435144  0.210696225
##   3       0.010  0.50  0.6757322  0.292687241
##   3       0.010  1.00  0.6802367  0.310599633
##   3       0.100  0.25  0.6882974  0.329067421
##   3       0.100  0.50  0.6935606  0.342383818
##   3       0.100  1.00  0.6962396  0.349879819
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were degree = 3, scale = 0.1 and C = 1.
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 24  4
##          1 16 48
##
##                Accuracy : 0.7826
##                  95% CI : (0.6844, 0.8619)
##     No Information Rate : 0.5652
##     P-Value [Acc > NIR] : 1.072e-05
##
##                   Kappa : 0.5418
##
##  Mcnemar's Test P-Value : 0.01391
##
##             Sensitivity : 0.6000
##             Specificity : 0.9231
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 0.7500
##              Prevalence : 0.4348
##          Detection Rate : 0.2609
##    Detection Prevalence : 0.3043
##       Balanced Accuracy : 0.7615
##
##        'Positive' Class : 0
##
## [1] " "
## [1] "SVM - RANDOM FOREST"
## [1] " "
## note: only 1 unique complexity parameters in default grid. Truncating the grid to 1 .
##
## Random Forest
##
## 374 samples
##   2 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 337, 337, 337, 335, 337, 336, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.6470365  0.2808172
```

```
##
## Tuning parameter 'mtry' was held constant at a value of 2
## rf variable importance
##
##      Overall
## medv     100
## rm         0
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 27 17
##          1 13 35
##
##                Accuracy : 0.6739
##                  95% CI : (0.5682, 0.768)
##     No Information Rate : 0.5652
##     P-Value [Acc > NIR] : 0.0218
##
##                   Kappa : 0.3441
##
##  Mcnemar's Test P-Value : 0.5839
##
##             Sensitivity : 0.6750
##             Specificity : 0.6731
##          Pos Pred Value : 0.6136
##          Neg Pred Value : 0.7292
##              Prevalence : 0.4348
##          Detection Rate : 0.2935
##    Detection Prevalence : 0.4783
##       Balanced Accuracy : 0.6740
##
##        'Positive' Class : 0
##
## [1] "Parameters:   mtry =  2 , ntree =  500 , nrnodes =  187"
## [1] " "
## [1] "DECISION TREE"
## [1] " "

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 30 13
##          1 10 39
##
##                Accuracy : 0.75
##                  95% CI : (0.6489, 0.8345)
##     No Information Rate : 0.5652
##     P-Value [Acc > NIR] : 0.0001829
##
##                   Kappa : 0.4957
##
##  Mcnemar's Test P-Value : 0.6766573
```

```
##
##             Sensitivity : 0.7500
##             Specificity : 0.7500
##          Pos Pred Value : 0.6977
##          Neg Pred Value : 0.7959
##              Prevalence : 0.4348
##          Detection Rate : 0.3261
##    Detection Prevalence : 0.4674
##       Balanced Accuracy : 0.7500
##
##        'Positive' Class : 0
##
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = 1, case = 2
```

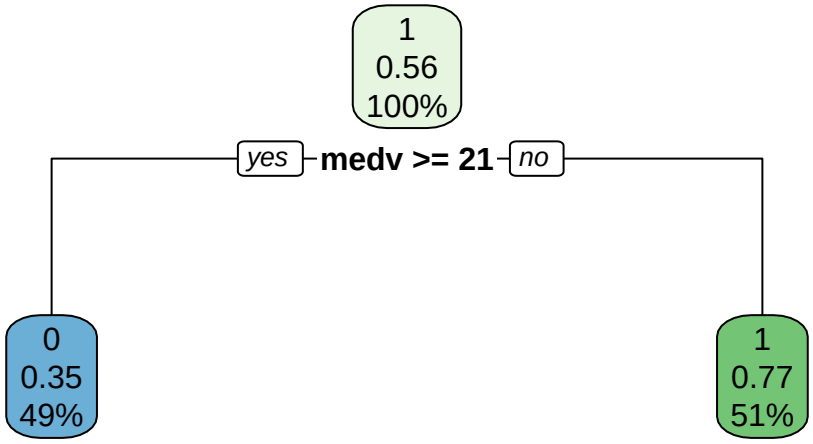
```
## Setting direction: controls < cases
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```




```
print(an3)
```

```
##
##
## Table: Predicting PTRatio: RM and MEDV Only
##
## |              |  AUC| Accuracy|
## |:-------------|----:|--------:|
## |SVM - poly    | 0.76|     0.78|
## |Decision Tree | 0.75|     0.75|
## |SVM - linear  | 0.75|     0.77|
## |SVM - radial  | 0.73|     0.75|
## |Random Forest | 0.67|     0.67|
```

With RM and MEDV, two smoothly distributed variables, SVM poly now performs best. Random Forest is worse than the other SVM algorithms. That one decision tree performs better than random forest is probably a matter of luck - since random forest averages performance it stands to reason a particular decision tree may perform better than average.

Now the performance for Random Forest drops considerably. For SVM there is little change. SVM is now the second best algorithm.

```
an4 <-RunAnalyses(df3Sparse, "Predicting PTRatio: Sparse Dataset")
```

4. Sparse Dataset

```
## Downloading GitHub repo ericonsi/EHData@HEAD

##          checking for file 'C:\Users\Eric\AppData\Local\Temp\Rtmp8kraBx\remotes21fc50103b00\ericonsi
##       -  preparing 'EHData':
##    checking DESCRIPTION meta-information ...     checking DESCRIPTION meta-information ...   v  check
##       -  checking for LF line-endings in source and make files and shell scripts
## -  checking for empty or unneeded directories
##   -  creating default NAMESPACE file
##   -  building 'EHData_0.1.0.tar.gz'
##
##


## Warning: package 'EHData' is in use and will not be installed

## [1] " "
## [1] "SVM - LINEAR"
## [1] " "
## Support Vector Machines with Linear Kernel
##
## 76 samples
## 10 predictors
##  2 classes: '0', '1'
##
## Pre-processing: centered (10), scaled (10)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 69, 68, 69, 68, 68, 68, ...
## Resampling results across tuning parameters:
##
##   C          Accuracy   Kappa
##   0.0100000  0.7434524  0.4972108
##   0.1147368  0.6916667  0.4012707
##   0.2194737  0.6392857  0.2956353
##   0.3242105  0.6267857  0.2682005
##   0.4289474  0.6089286  0.2315111
##   0.5336842  0.6089286  0.2277202
##   0.6384211  0.6035714  0.2173868
##   0.7431579  0.6178571  0.2437975
##   0.8478947  0.6130952  0.2327303
##   0.9526316  0.6130952  0.2306562
##   1.0573684  0.6178571  0.2385536
##   1.1621053  0.6220238  0.2468870
##   1.2668421  0.6172619  0.2368357
##   1.3715789  0.6172619  0.2345024
##   1.4763158  0.6220238  0.2429565
##   1.5810526  0.6172619  0.2345024
##   1.6857895  0.6214286  0.2428357
##   1.7905263  0.6261905  0.2521352
##   1.8952632  0.6261905  0.2521352
##   2.0000000  0.6261905  0.2521352
##
## Accuracy was used to select the optimal model using the largest value.
```

```
## The final value used for the model was C = 0.01.
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 9 3
##          1 0 6
##
##                Accuracy : 0.8333
##                  95% CI : (0.5858, 0.9642)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.003769
##
##                   Kappa : 0.6667
##
##  Mcnemar's Test P-Value : 0.248213
##
##             Sensitivity : 1.0000
##             Specificity : 0.6667
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5000
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.6667
##       Balanced Accuracy : 0.8333
##
##        'Positive' Class : 0
##
## [1] " "
## [1] "SVM - RADIAL"
## [1] " "
## Support Vector Machines with Radial Basis Function Kernel
##
## 76 samples
## 10 predictors
##  2 classes: '0', '1'
##
## Pre-processing: centered (10), scaled (10)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 69, 68, 69, 68, 68, 68, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.7339286  0.4848262
##   0.50  0.7208333  0.4581083
##   1.00  0.7130952  0.4411453
##
## Tuning parameter 'sigma' was held constant at a value of 0.09602855
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.09602855 and C = 0.25.
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
```

```
##          0 9 4
##          1 0 5
##
##                Accuracy : 0.7778
##                  95% CI : (0.5236, 0.9359)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.01544
##
##                   Kappa : 0.5556
##
##  Mcnemar's Test P-Value : 0.13361
##
##             Sensitivity : 1.0000
##             Specificity : 0.5556
##          Pos Pred Value : 0.6923
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5000
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.7222
##       Balanced Accuracy : 0.7778
##
##        'Positive' Class : 0
##
## [1] " "
## [1] "SVM - POLY"
## [1] " "
## Support Vector Machines with Polynomial Kernel
##
## 76 samples
## 10 predictors
##  2 classes: '0', '1'
##
## Pre-processing: centered (10), scaled (10)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 69, 68, 69, 68, 68, 68, ...
## Resampling results across tuning parameters:
##
##   degree  scale  C     Accuracy   Kappa
##   1       0.001  0.25  0.5285714  0.00000000
##   1       0.001  0.50  0.5285714  0.00000000
##   1       0.001  1.00  0.5285714  0.00000000
##   1       0.010  0.25  0.5607143  0.10143303
##   1       0.010  0.50  0.6928571  0.39543305
##   1       0.010  1.00  0.7434524  0.49721083
##   1       0.100  0.25  0.7250000  0.46444729
##   1       0.100  0.50  0.6863095  0.38958120
##   1       0.100  1.00  0.7000000  0.41793732
##   2       0.001  0.25  0.5285714  0.00000000
##   2       0.001  0.50  0.5285714  0.00000000
##   2       0.001  1.00  0.5428571  0.03939394
##   2       0.010  0.25  0.7053571  0.42043305
##   2       0.010  0.50  0.7380952  0.48901140
##   2       0.010  1.00  0.7250000  0.46444729
##   2       0.100  0.25  0.7083333  0.43460399
```

```
##   2       0.100  0.50  0.7047619  0.42655271
##   2       0.100  1.00  0.6970238  0.40466382
##   3       0.001  0.25  0.5285714  0.00000000
##   3       0.001  0.50  0.5285714  0.00000000
##   3       0.001  1.00  0.6297619  0.26115496
##   3       0.010  0.25  0.7392857  0.48887749
##   3       0.010  0.50  0.7202381  0.45589174
##   3       0.010  1.00  0.7029762  0.42291453
##   3       0.100  0.25  0.7255952  0.46614530
##   3       0.100  0.50  0.7059524  0.41947863
##   3       0.100  1.00  0.7101190  0.42697436
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were degree = 1, scale = 0.01 and C = 1.
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 9 3
##          1 0 6
##
##                Accuracy : 0.8333
##                  95% CI : (0.5858, 0.9642)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.003769
##
##                   Kappa : 0.6667
##
##  Mcnemar's Test P-Value : 0.248213
##
##             Sensitivity : 1.0000
##             Specificity : 0.6667
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5000
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.6667
##       Balanced Accuracy : 0.8333
##
##        'Positive' Class : 0
##
## [1] " "
## [1] "SVM - RANDOM FOREST"
## [1] " "
## Random Forest
##
## 76 samples
## 10 predictors
##  2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 69, 68, 69, 68, 68, 68, ...
## Resampling results across tuning parameters:
```

```
##
##   mtry  Accuracy   Kappa
##    2    0.7750000  0.5559744
##    6    0.7892857  0.5858889
##   10    0.7892857  0.5858889
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 6.
## rf variable importance
##
##           Overall
## indus      100.00
## rad         85.33
## dis         54.24
## nox         46.66
## lstat       44.83
## medv        32.99
## rm          25.26
## age         23.48
## zn          12.71
## crimerate    0.00
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 9 2
##          1 0 7
##
##                Accuracy : 0.8889
##                  95% CI : (0.6529, 0.9862)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.0006561
##
##                   Kappa : 0.7778
##
##  Mcnemar's Test P-Value : 0.4795001
##
##             Sensitivity : 1.0000
##             Specificity : 0.7778
##          Pos Pred Value : 0.8182
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5000
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.6111
##       Balanced Accuracy : 0.8889
##
##        'Positive' Class : 0
##
## [1] "Parameters:   mtry =  6 , ntree =  500 , nrnodes =  31"
## [1] " "
## [1] "DECISION TREE"
## [1] " "

## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction 0 1
##          0 9 5
##          1 0 4
##
##                Accuracy : 0.7222
##                  95% CI : (0.4652, 0.9031)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.04813
##
##                   Kappa : 0.4444
##
##  Mcnemar's Test P-Value : 0.07364
##
##             Sensitivity : 1.0000
##             Specificity : 0.4444
##          Pos Pred Value : 0.6429
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5000
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.7778
##       Balanced Accuracy : 0.7222
##
##        'Positive' Class : 0
##
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = 1, case = 2
```

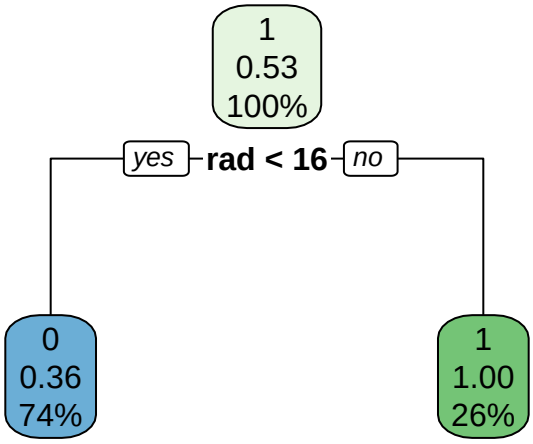
```
## Setting direction: controls < cases
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```




```
print(an4)
```

```
##
##
## Table: Predicting PTRatio: Sparse Dataset
##
## |              |  AUC| Accuracy|
## |:-------------|----:|--------:|
## |Random Forest | 0.89|     0.89|
## |SVM - linear  | 0.83|     0.83|
## |SVM - poly    | 0.83|     0.83|
## |SVM - radial  | 0.78|     0.78|
## |Decision Tree | 0.72|     0.72|
```

With 20% of the dataset chosen at random, SVM's performance is a lot closer to Random Forest, though it still falls behind.

Conclusion

The literature suggests that both algorithms may perfom well under different circumstances. In this dataset, Random Forest performs better than SVM. However, the literature does not generally point to exactly why one algorithm outperforms the other, unless the dataset is sparse or contains low-level imagery, etc. The existence of many broken distributions and idiosyncratic cohorts may help to explain Random Forest's success

in this dataset. When run on only the smoothly distributed data, SVM was superior. This observation may not be generalizable, however - more resaerch would be needed to support it..