Analysis: Classification Trees & Random Forest

Hannah Norman

```
library(caret)
```

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

```
library(mlbench)
library(rattle)
```

```
## Loading required package: tibble
## Loading required package: bitops
## 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.
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:rattle':
##
##     importance
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(ROCR)
```

Original NASA Asteroids Data Set

```
nasa_orig <- read.csv("nasa_original.csv")
# nasa_orig
```

Cleaned NASA Asteroids Data Set

```
nasa <- read.csv("nasa.csv")
#nasa
```

Numerical and Graphical Summaries of Response Variable

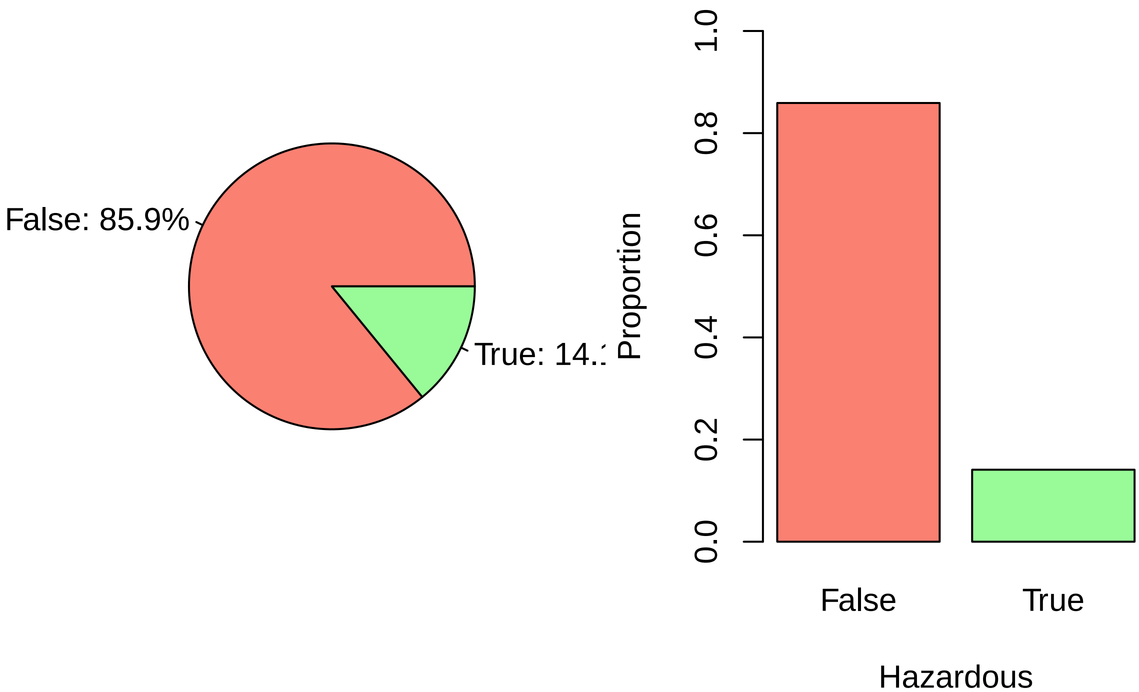
```
prop.hazardous <- prop.table(table(nasa$Hazardous))
prop.hazardous
```

```
##
##     False      True
## 0.8589994 0.1410006
```

```
par(mfrow=c(1,2))

# pie chart
count.hazardous <- table(nasa$Hazardous)
lbls <- paste(levels(as.factor(nasa$Hazardous)), ": ",
              round(prop.hazardous,3)*100, "%", sep="")
pie(count.hazardous, labels=lbls, col=c("salmon", "palegreen"))

# bar plot
barplot(prop.hazardous, xlab="Hazardous", ylab="Proportion", ylim=c(0, 1.0),
        col=c("salmon", "palegreen"))
```



Analyses of Predictors

(in search of outliers that might skew analysis)

```
# numerical summary + box plots
summary(nasa)
```

```
##  Absolute.Magnitude Est.Dia.in.KM.min. Est.Dia.in.KM.max. Est.Dia.in.KM.range
##  Min.   :15.20      Min.   :0.00101    Min.   :0.00226    Min.   :0.001249
##  1st Qu.:20.40      1st Qu.:0.03052    1st Qu.:0.06824    1st Qu.:0.037722
##  Median :22.30      Median :0.09216    Median :0.20608    Median :0.113919
##  Mean   :22.53      Mean   :0.17185    Mean   :0.38428    Mean   :0.212423
```

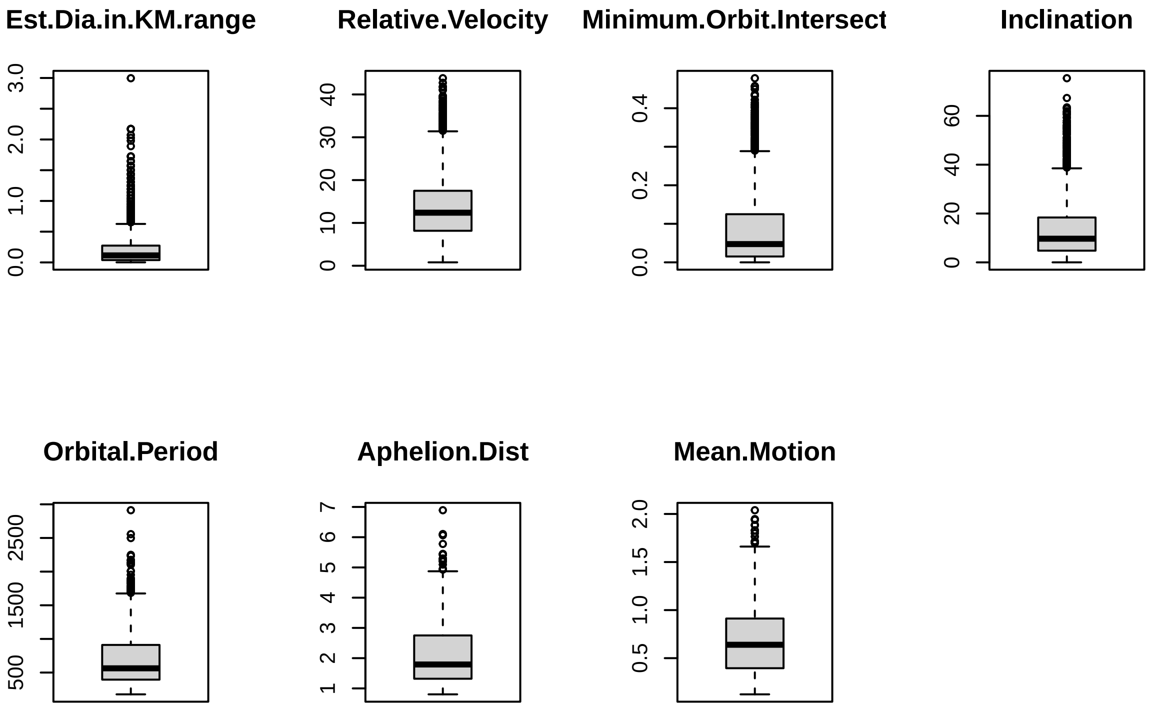
```
##  3rd Qu.:24.70      3rd Qu.:0.22108    3rd Qu.:0.49436    3rd Qu.:0.273273
##  Max.   :32.10      Max.   :2.42412    Max.   :5.42051    Max.   :2.996383
##  Close.Approach.Date Relative.Velocity.in.KM.per.sec Miss.Dist.in.KM
##  Min.   :19950101    Min.   : 0.8002                 Min.   :   26610
##  1st Qu.:20010808    1st Qu.: 8.1680                 1st Qu.:16215102
##  Median :20070912    Median :12.3870                 Median :37033462
##  Mean   :20066259    Mean   :13.6128                 Mean   :36473114
##  3rd Qu.:20120922    3rd Qu.:17.5079                 3rd Qu.:56290664
##  Max.   :20160908    Max.   :43.7899                 Max.   :74781600
##  Orbit.Uncertainity Minimum.Orbit.Intersection Jupiter.Tisserand.Invariant
##  Min.   :0.0        Min.   :0.0000021          Min.   :2.196
##  1st Qu.:1.0        1st Qu.:0.0151341          1st Qu.:3.807
##  Median :5.0        Median :0.0473452          Median :4.798
##  Mean   :4.1        Mean   :0.0823254          Mean   :4.852
##  3rd Qu.:7.0        3rd Qu.:0.1249268          3rd Qu.:5.774
##  Max.   :9.0        Max.   :0.4778910          Max.   :9.025
##   Eccentricity     Semi.Major.Axis   Inclination       Asc.Node.Longitude
##  Min.   :0.01296   Min.   :0.6159   Min.   : 0.01451   Min.   :  0.0019
##  1st Qu.:0.24907   1st Qu.:1.0530   1st Qu.: 4.78566   1st Qu.: 83.6648
##  Median :0.38236   Median :1.3347   Median : 9.68715   Median :173.5898
##  Mean   :0.39321   Mean   :1.4848   Mean   :12.84105   Mean   :173.5232
##  3rd Qu.:0.52589   3rd Qu.:1.8386   3rd Qu.:18.38238   3rd Qu.:258.4855
##  Max.   :0.96026   Max.   :3.9908   Max.   :75.40667   Max.   :359.9059
##  Orbital.Period   Perihelion.Distance Perihelion.Arg     Aphelion.Dist
##  Min.   : 176.6   Min.   :0.08074     Min.   :  0.0069   Min.   :0.8038
##  1st Qu.: 394.7   1st Qu.:0.67807     1st Qu.: 95.6381   1st Qu.:1.3191
##  Median : 563.2   Median :0.87434     Median :188.4906   Median :1.7911
##  Mean   : 692.8   Mean   :0.84323     Mean   :184.1273   Mean   :2.1264
##  3rd Qu.: 910.6   3rd Qu.:1.01936     3rd Qu.:272.5434   3rd Qu.:2.7517
##  Max.   :2912.0   Max.   :1.29983     Max.   :359.9931   Max.   :6.8918
##   Mean.Anomaly       Mean.Motion       Hazardous
##  Min.   :  0.0032   Min.   :0.1236   Length:3078
##  1st Qu.: 83.3164   1st Qu.:0.3953   Class :character
##  Median :183.8903   Median :0.6392   Mode  :character
##  Mean   :180.1357   Mean   :0.6811
##  3rd Qu.:276.3132   3rd Qu.:0.9121
##  Max.   :359.9180   Max.   :2.0390
```

```r
par(mfrow=c(2,4))
boxplot(nasa$Est.Dia.in.KM.range, main="Est.Dia.in.KM.range")
boxplot(nasa$Relative.Velocity, main="Relative.Velocity")
boxplot(nasa$Minimum.Orbit.Intersection, main="Minimum.Orbit.Intersection")
boxplot(nasa$Inclination, main="Inclination")
boxplot(nasa$Orbital.Period, main="Orbital.Period")
boxplot(nasa$Aphelion.Dist, main="Aphelion.Dist")
boxplot(nasa$Mean.Motion, main="Mean.Motion")

# to find row number of outlier observations
nasa[which.max(nasa$Est.Dia.in.KM.range),]
```

```
##      Absolute.Magnitude Est.Dia.in.KM.min. Est.Dia.in.KM.max.
## 2728               15.2           2.424125           5.420508
##      Est.Dia.in.KM.range Close.Approach.Date Relative.Velocity.in.KM.per.sec
## 2728            2.996383            20141222                         23.5172
##      Miss.Dist.in.KM Orbit.Uncertainity Minimum.Orbit.Intersection
```

```
## 2728        45467472                  0                    0.153116
##      Jupiter.Tisserand.Invariant Eccentricity Semi.Major.Axis Inclination
## 2728                       4.864    0.3462176        1.261473    36.90474
##      Asc.Node.Longitude Orbital.Period Perihelion.Distance Perihelion.Arg
## 2728           111.2844       517.5058           0.8247287       349.1364
##      Aphelion.Dist Mean.Anomaly Mean.Motion Hazardous
## 2728      1.698217     328.5237   0.6956443     False
```



Predictive Performance Stats

```
get_stats <- function(CM) {
  TP <- CM[2,2]
  FP <- CM[1,2]
  TN <- CM[1,1]
  FN <- CM[2,1]

  acc <- (TP+TN) / (TP+TN+FN+FP)
  err <- (FP+FN) / (TP+TN+FN+FP)
  pre <- (TP) / (TP+FP)
  sen <- (TP) / (TP+FN)
  spe <- (TN) / (TN+FP)
  fme <- (2*pre*sen) / (pre+sen)
  mcc_denom <- sqrt(TP+FP)*sqrt(TP+FN)*sqrt(TN+FP)*sqrt(TN+FN)
  mcc <- (TP*TN - FP*FN) / mcc_denom

  name <- c("accuracy", "error rate", "precision", "sensitivity", "specificity", "F-measure", "Matthew'
  value <- c(acc, err, pre, sen, spe, fme, mcc)
  stats <- data.frame(name, value)

  return (stats)
}
```

Classification Tree Analysis

1. Make sure that the model assumptions, if any, are satisfied.

No model assumptions of decision trees to be satisfied? We are fitting a classification tree as opposed to a regression tree because the response variable is categorical and binary.

2. Assess the model fit and perform diagnostics, if appropriate.

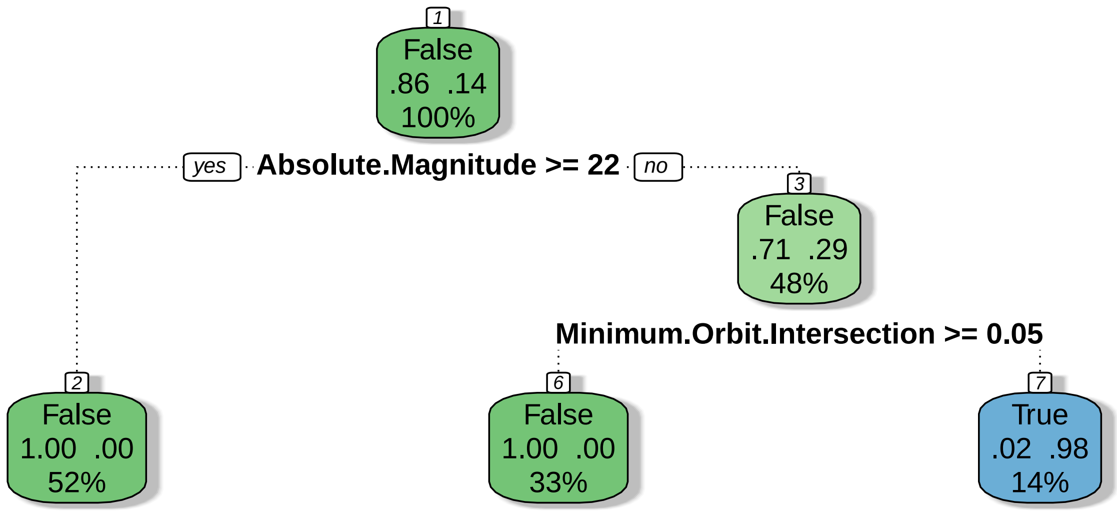
Both methods appear to give identical results. Note the identical accuracy and kappa ratings across all cp tuning parameters in the plain rpart method…

```r
set.seed(1)

# rpart
nasa.CVrpart <- train(Hazardous ~ ., data=nasa,
                      method="rpart",
                      tuneGrid = expand.grid(cp=seq(0.005, 0.05, length=10)),
                      trControl=trainControl(method="cv", number=10,
                                             savePredictions=TRUE,
                                             classProbs=TRUE,
                                             selectionFunction = "oneSE"))
nasa.CVrpart
```

```
## CART
##
## 3078 samples
##   20 predictor
##    2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2770, 2771, 2771, 2769, 2770, 2771, ...
## Resampling results across tuning parameters:
##
##   cp     Accuracy   Kappa
##   0.005  0.9947999  0.9783443
##   0.010  0.9947999  0.9783443
##   0.015  0.9947999  0.9783443
##   0.020  0.9947999  0.9783443
##   0.025  0.9947999  0.9783443
##   0.030  0.9947999  0.9783443
##   0.035  0.9947999  0.9783443
##   0.040  0.9947999  0.9783443
##   0.045  0.9947999  0.9783443
##   0.050  0.9947999  0.9783443
##
## Accuracy was used to select the optimal model using  the one SE rule.
## The final value used for the model was cp = 0.05.
```

```r
# print tree
fancyRpartPlot(nasa.CVrpart$finalModel)
```



Rattle 2023−Apr−26 23:40:22 hannah

3. Identify tuning parameters to be used, if appropriate.

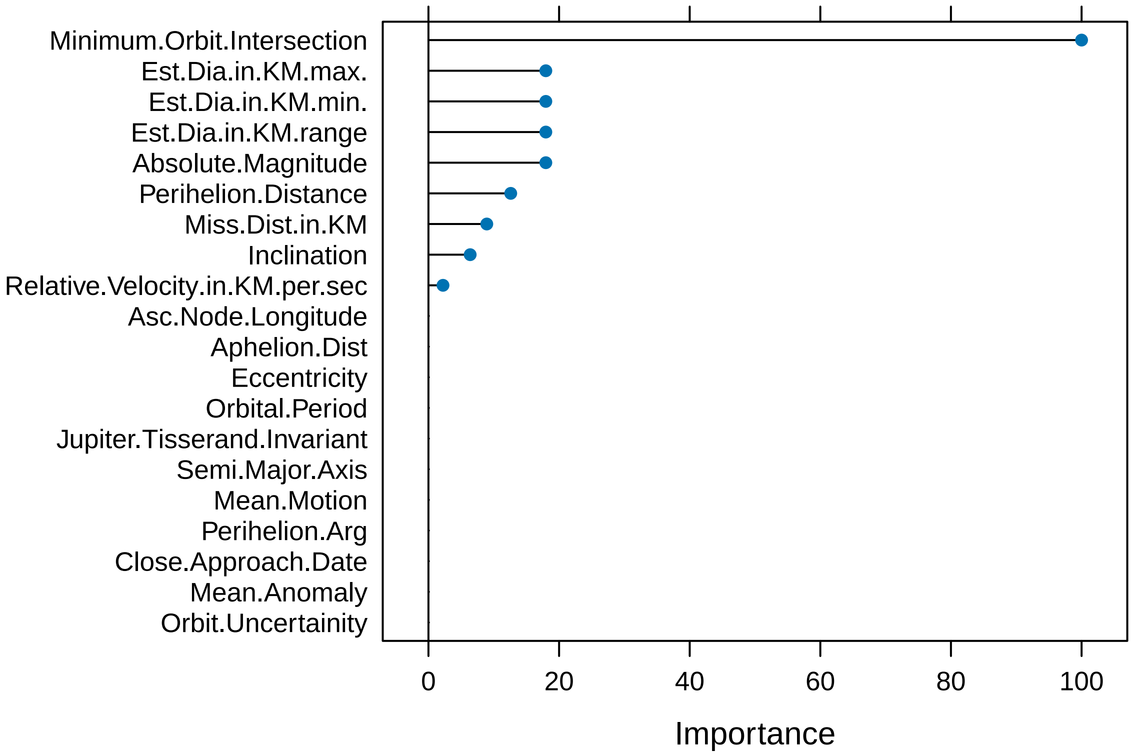
If the plain rpart method is utilized, it selects a cp (complexity parameter) value of 0.05 to maximize accuracy for the final model. Note that the accuracy and kappa values are identical for each of the cp values tried by the model, so any choice within that range should produce comparable results.

4. Identify and interpret the effect of selected variables.

```
# variable importance (rpart)
varImp(nasa.CVrpart)
```

```
## rpart variable importance
##
##                                     Overall
## Minimum.Orbit.Intersection          100.000
## Est.Dia.in.KM.range                  17.973
## Est.Dia.in.KM.min.                   17.973
## Absolute.Magnitude                   17.973
## Est.Dia.in.KM.max.                   17.973
## Perihelion.Distance                  12.601
## Miss.Dist.in.KM                       8.932
## Inclination                           6.388
## Relative.Velocity.in.KM.per.sec       2.229
## Aphelion.Dist                         0.000
## Perihelion.Arg                        0.000
## Jupiter.Tisserand.Invariant           0.000
## Mean.Motion                           0.000
## Orbit.Uncertainity                    0.000
## Close.Approach.Date                   0.000
## Mean.Anomaly                          0.000
## Semi.Major.Axis                       0.000
## Asc.Node.Longitude                    0.000
## Orbital.Period                        0.000
## Eccentricity                          0.000
```

```r
plot(varImp(nasa.CVrpart))
```

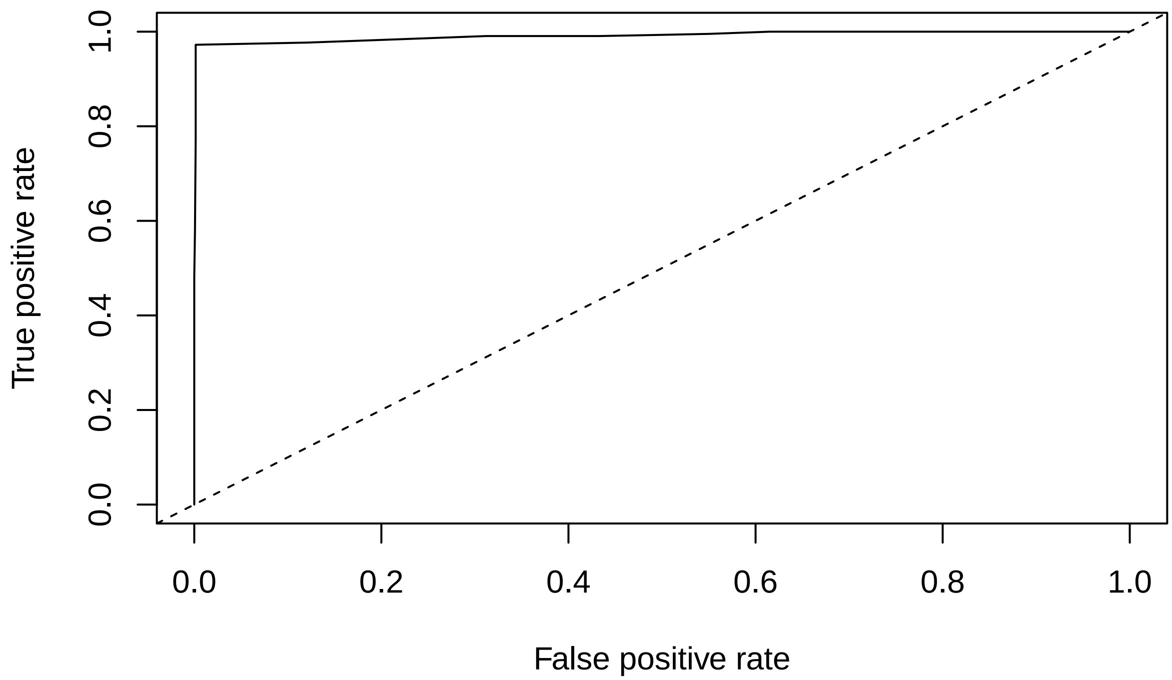


5. Evaluate the cross-validated (CV) predictive performance.

```r
head(nasa.CVrpart$pred)
```

```
##    pred   obs rowIndex       False        True    cp Resample
## 1 False False       11 1.000000000 0.000000000 0.005   Fold01
## 2 False False       13 0.993243243 0.006756757 0.005   Fold01
## 3 False False       34 1.000000000 0.000000000 0.005   Fold01
## 4 False False       38 0.993243243 0.006756757 0.005   Fold01
## 5  True  True       68 0.002624672 0.997375328 0.005   Fold01
## 6  True  True       70 0.002624672 0.997375328 0.005   Fold01
```

```r
pihatcv.rpart <- nasa.CVrpart$pred[nasa.CVrpart$pred$cp == 0.05,]
predcv.rpart <- prediction(pihatcv.rpart$True, pihatcv.rpart$obs)
perfcv.rpart <- performance(predcv.rpart, "tpr", "fpr")
plot(perfcv.rpart)
abline(a=0, b=1, lty=2)
```




```
aucCV.rpart <- performance(predcv.rpart, "auc")@y.values
aucCV.rpart
```

```
## [[1]]
## [1] 0.991193
```

```
confMat <- table(pihatcv.rpart$obs, pihatcv.rpart$pred)
confMat
```

```
##
##         False True
##   False  2640    4
##   True     12  422
```

```
rpart.stats <- get_stats(confMat)
rpart.stats
```

```
##            name       value
## 1      accuracy 0.994801819
## 2    error rate 0.005198181
## 3     precision 0.990610329
## 4   sensitivity 0.972350230
## 5   specificity 0.998487141
## 6     F-measure 0.981395349
## 7 Matthew's CC 0.978431703
```

Random Forest Analysis

1. Make sure that the model assumptions, if any, are satisfied.

No model assumptions of random forest to be satisfied?

2. Assess the model fit and perform diagnostics, if appropriate.

```
set.seed(1)
```

```r
nasa.rf <- randomForest(as.factor(Hazardous) ~ ., data=nasa)
nasa.rf
```

```
##
## Call:
##  randomForest(formula = as.factor(Hazardous) ~ ., data = nasa)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 0.49%
## Confusion matrix:
##       False True class.error
## False  2640    4 0.001512859
## True     11  423 0.025345622
```

```r
set.seed(1)

nasa.CVrf <- train(Hazardous ~ ., data=nasa,
                   method="rf",
                   trControl=trainControl(method="cv", number=10,
                                          savePredictions=TRUE,
                                          classProbs=TRUE))
nasa.CVrf
```

```
## Random Forest
##
## 3078 samples
##   20 predictor
##    2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2770, 2771, 2771, 2769, 2770, 2771, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9951256  0.9796031
##   11    0.9960986  0.9837768
##   20    0.9964264  0.9850905
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 20.
```

3. Identify tuning parameters to be used, if appropriate.

The final model selects an mtry tuning parameter value of 20 in order to maximize accuracy.

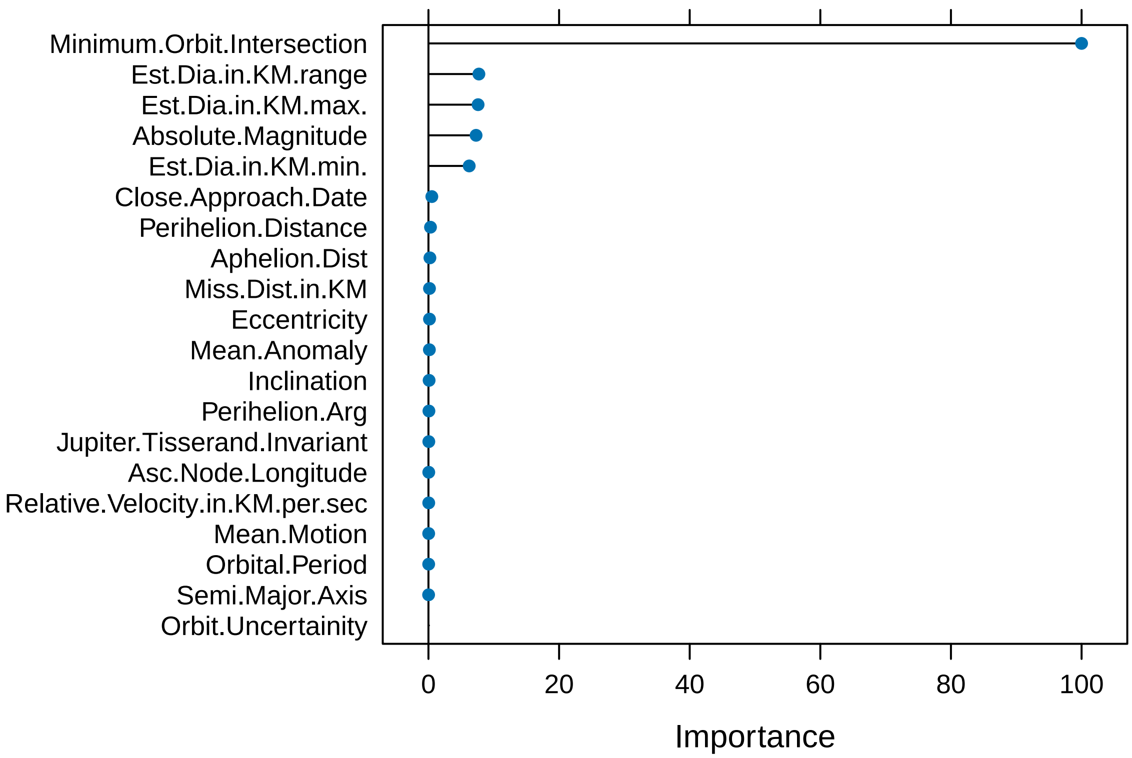
4. Identify and interpret the effect of selected variables.

```r
varImp(nasa.CVrf)
```

```
## rf variable importance
##
##                                    Overall
## Minimum.Orbit.Intersection      100.00000
## Est.Dia.in.KM.range               7.72334
```

```
## Est.Dia.in.KM.max.                 7.60017
## Absolute.Magnitude                 7.29260
## Est.Dia.in.KM.min.                 6.23992
## Close.Approach.Date                0.51729
## Perihelion.Distance                0.31224
## Aphelion.Dist                      0.22185
## Miss.Dist.in.KM                    0.15809
## Eccentricity                       0.15804
## Mean.Anomaly                       0.14305
## Inclination                        0.09922
## Perihelion.Arg                     0.07231
## Jupiter.Tisserand.Invariant        0.05422
## Asc.Node.Longitude                 0.04687
## Relative.Velocity.in.KM.per.sec    0.04404
## Mean.Motion                        0.03637
## Orbital.Period                     0.03556
## Semi.Major.Axis                    0.02316
## Orbit.Uncertainity                 0.00000
```

```
plot(varImp(nasa.CVrf))
```



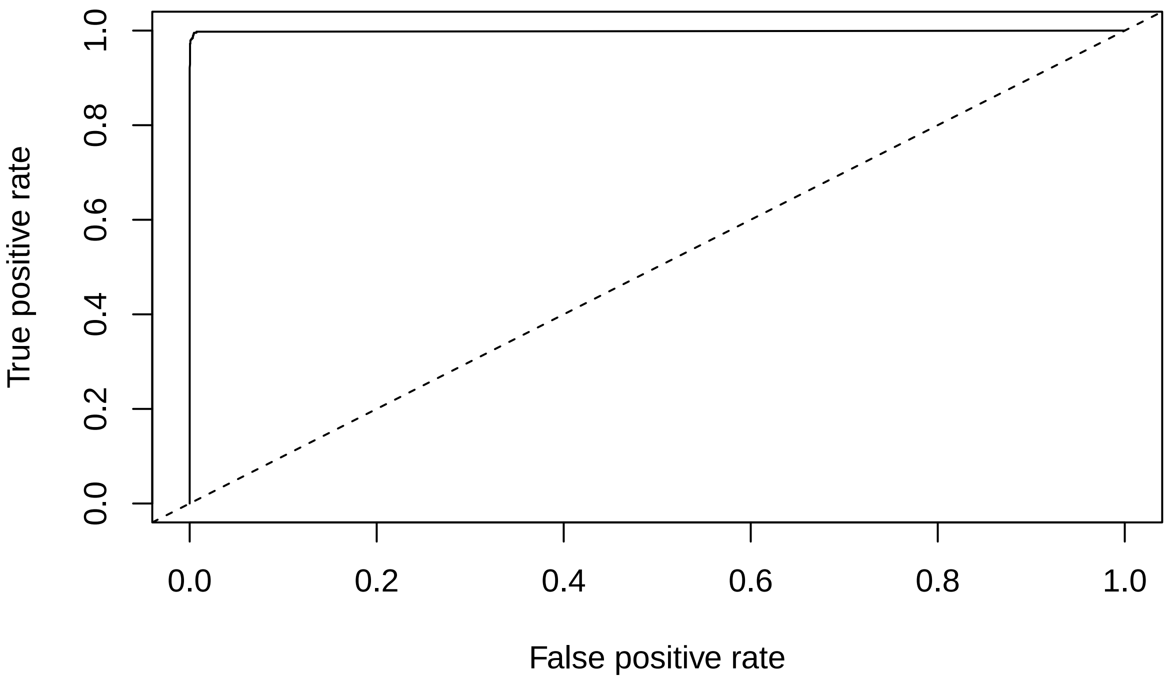
5. Evaluate the cross-validated (CV) predictive performance.

```
head(nasa.CVrf$pred)
```

```
##    pred   obs False  True rowIndex mtry Resample
## 1 False False 0.970 0.030       11    2   Fold01
## 2 False False 0.984 0.016       13    2   Fold01
## 3 False False 0.968 0.032       34    2   Fold01
## 4 False False 0.996 0.004       38    2   Fold01
## 5  True  True 0.242 0.758       68    2   Fold01
```

```
## 6  True  True 0.212 0.788      70    2   Fold01
```

```
pihatcv.rf <- nasa.CVrf$pred[nasa.CVrf$pred$mtry == 20,]
predcv.rf <- prediction(pihatcv.rf$True, pihatcv.rf$obs)
perfcv.rf <- performance(predcv.rf, "tpr", "fpr")
plot(perfcv.rf)
abline(a=0, b=1, lty=2)
```




```
aucCV.rf <- performance(predcv.rf, "auc")@y.values
aucCV.rf
```

```
## [[1]]
## [1] 0.9986688
```

```
confMat <- table(pihatcv.rf$obs, pihatcv.rf$pred)
confMat
```

```
##
##         False True
##   False  2642    2
##   True      9  425
```

```
rf.stats <- get_stats(confMat)
rf.stats
```

```
##           name       value
## 1     accuracy 0.996426251
## 2   error rate 0.003573749
## 3    precision 0.995316159
## 4  sensitivity 0.979262673
## 5  specificity 0.999243570
## 6    F-measure 0.987224158
## 7 Matthew's CC 0.985190895
```