Asteriod Data Analysis (working title)

Annika Lin, Hannah Norman, and Madeline Pfister

2023-05-02

Load Packages and Functions

```
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)
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following object is masked from 'package:bitops':
##
##     %&%
```

```
## Loaded glmnet 4.1-7
```

```
library(car)
```

```
## Loading required package: carData
```

```
library(ResourceSelection)
```

```
## ResourceSelection 0.3-5   2019-07-22
```

Predictive Performance Stats

We created a function in which the output is a table summarizing all of the predictive performance stats. This function was used throughtout the analysis.

```
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's CC")
  value <- c(acc, err, pre, sen, spe, fme, mcc)
  stats <- data.frame(name, value)

  return (stats)
}
```

Data Cleaning and Initital Analysis

We began by choosing one date to focus on for our data set. The data we chose was April 6, 2017, so we filtered our data to only include observations from that date.

```
#import dataset filtered for '2017-04-06'
nasa <- read.csv("nasa.csv")
```

Analysis of Predictors

We did a simple numerical summary and box plots to determine if there are any large outliers in the data set.

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

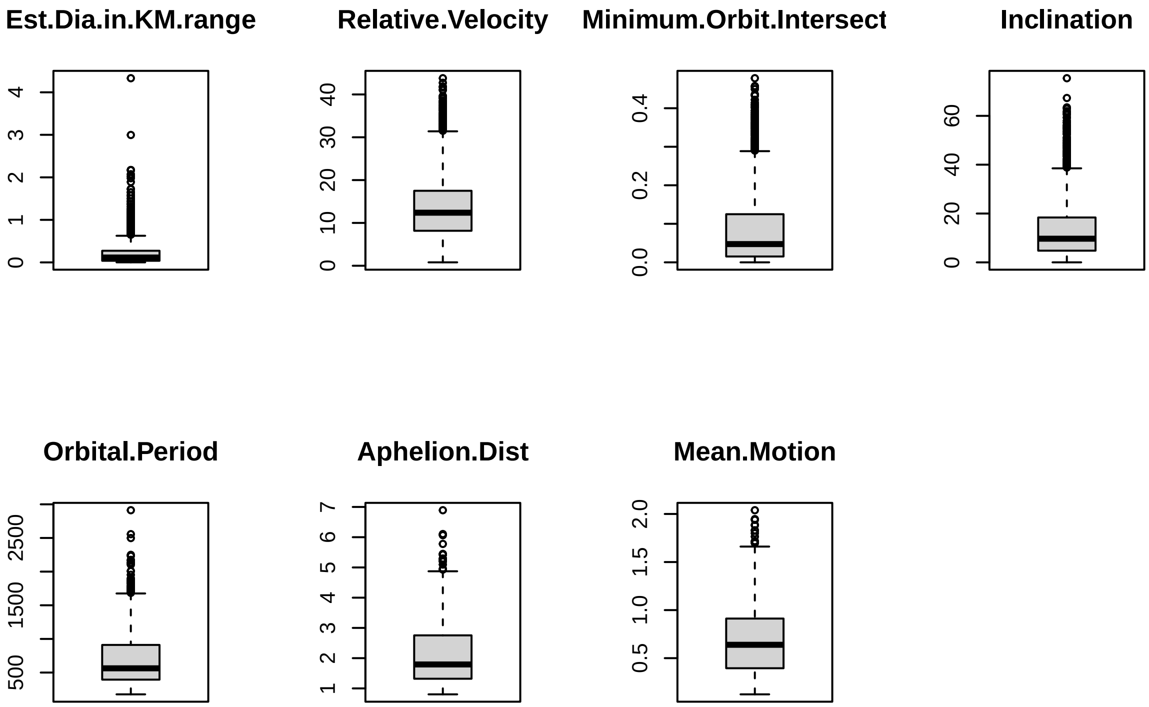
```
##  Absolute.Magnitude Est.Dia.in.KM.min. Est.Dia.in.KM.max. Est.Dia.in.KM.range
##  Min.   :14.40      Min.   :0.001011   Min.   :0.00226    Min.   :0.001249
##  1st Qu.:20.40      1st Qu.:0.030518   1st Qu.:0.06824    1st Qu.:0.037722
##  Median :22.30      Median :0.092163   Median :0.20608    Median :0.113919
```

```
##  Mean   :22.53      Mean   :0.172936   Mean   :0.38670    Mean   :0.213761
##  3rd Qu.:24.70      3rd Qu.:0.221083   3rd Qu.:0.49436    3rd Qu.:0.273273
##  Max.   :32.10      Max.   :3.503926   Max.   :7.83502    Max.   :4.331091
##  Close.Approach.Date Relative.Velocity.in.KM.per.sec Miss.Dist.in.KM
##  Min.   :19950101    Min.   : 0.8002                 Min.   :   26610
##  1st Qu.:20010765    1st Qu.: 8.1683                 1st Qu.:16224266
##  Median :20070908    Median :12.3900                 Median :37032388
##  Mean   :20066238    Mean   :13.6152                 Mean   :36468199
##  3rd Qu.:20120922    3rd Qu.:17.5084                 3rd Qu.:56281652
##  Max.   :20160908    Max.   :43.7899                 Max.   :74781600
##  Orbit.Uncertainity Minimum.Orbit.Intersection Jupiter.Tisserand.Invariant
##  Min.   :0.000      Min.   :0.0000021          Min.   :2.196
##  1st Qu.:1.000      1st Qu.:0.0151384          1st Qu.:3.804
##  Median :5.000      Median :0.0473248          Median :4.798
##  Mean   :4.099      Mean   :0.0823078          Mean   :4.852
##  3rd Qu.:7.000      3rd Qu.:0.1248985          3rd Qu.:5.774
##  Max.   :9.000      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.:0.24918   1st Qu.:1.0530   1st Qu.: 4.78919   1st Qu.: 83.6762
##  Median :0.38253   Median :1.3349   Median : 9.68518   Median :173.6808
##  Mean   :0.39329   Mean   :1.4850   Mean   :12.83906   Mean   :173.5626
##  3rd Qu.:0.52595   3rd Qu.:1.8388   3rd Qu.:18.38036   3rd Qu.:258.6316
##  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.:0.67808     1st Qu.: 95.6430   1st Qu.:1.3191
##  Median : 563.3   Median :0.87390     Median :188.5239   Median :1.7918
##  Mean   : 692.9   Mean   :0.84320     Mean   :184.1443   Mean   :2.1268
##  3rd Qu.: 910.7   3rd Qu.:1.01935     3rd Qu.:272.5059   3rd Qu.:2.7545
##  Max.   :2912.0   Max.   :1.29983     Max.   :359.9931   Max.   :6.8918
##   Mean.Anomaly       Mean.Motion       Hazardous
##  Min.   :  0.0032   Min.   :0.1236   Length:3079
##  1st Qu.: 83.5492   1st Qu.:0.3953   Class :character
##  Median :183.9847   Median :0.6391   Mode  :character
##  Mean   :180.1871   Mean   :0.6810
##  3rd Qu.:276.3719   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.
## 695               14.4           3.503926           7.835018
##     Est.Dia.in.KM.range Close.Approach.Date Relative.Velocity.in.KM.per.sec
## 695            4.331091            20001222                        21.19854
##     Miss.Dist.in.KM Orbit.Uncertainity Minimum.Orbit.Intersection
```

```
## 695        21340634                    0                   0.0282524
##     Jupiter.Tisserand.Invariant Eccentricity Semi.Major.Axis Inclination
## 695                       3.573    0.6343498        1.982214    6.705068
##     Asc.Node.Longitude Orbital.Period Perihelion.Distance Perihelion.Arg
## 695           294.8956       1019.352           0.7247968       236.3403
##     Aphelion.Dist Mean.Anomaly Mean.Motion Hazardous
## 695      3.239631       338.28   0.3531656      True
```



Oberservation 695 was a very large outlier. We decided to remove it from the data, so it did not skew our model results.

```r
#remove outlier
nasa <- nasa[-695,]
```

Create a Proportion Table

We wanted to create a proportion table to get an understanding of how many asteriods were classified as hazardous and how many were classified as non-hazardous before beginning our analysis.

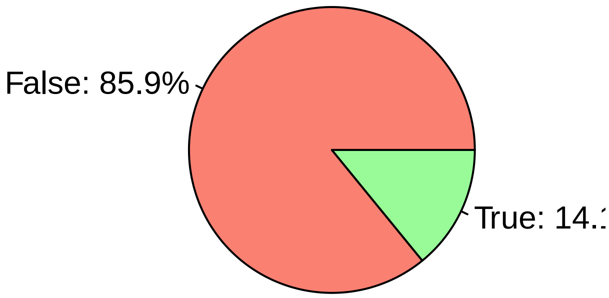
```r
prop.hazardous <- prop.table(table(nasa$Hazardous))
prop.hazardous
```

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

We found that the majority of our asteriods (85.9%) were classified as non-hazardous. All of our observations were classified as either hazardous or non-hazardous. This is also shown by the visulaizations below.

```r
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"))
```



Data Analysis

Lasso's Penalized Regression

Create the design matrix.

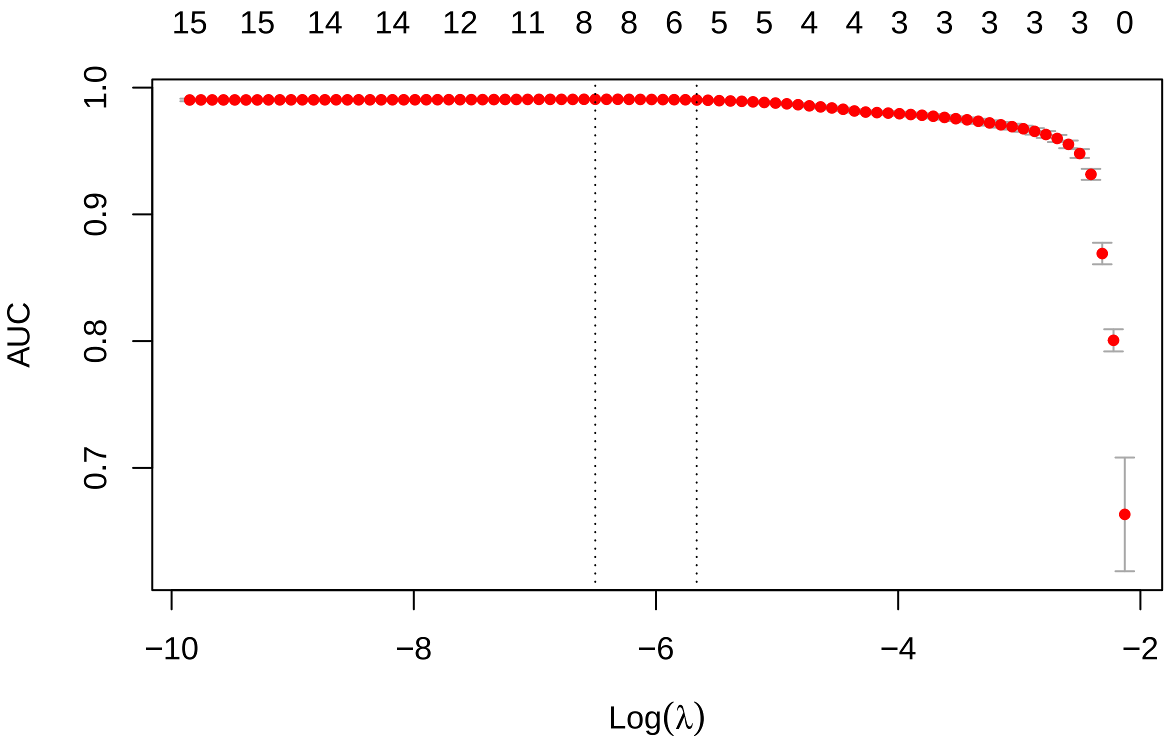
```
X = model.matrix(Hazardous ~ ., data=nasa)
Y = as.numeric(nasa$Hazardous=="True")
```

Conduct the cross-validation.

```
set.seed(1)
cvfit = cv.glmnet(x=X[,-1], y=Y, family="binomial", type.measure="auc")
cvfit
```

```
##
## Call:  cv.glmnet(x = X[, -1], y = Y, type.measure = "auc", family = "binomial")
##
## Measure: AUC
##
##       Lambda Index Measure        SE Nonzero
## min 0.001501    48  0.9908 0.0007380       8
## 1se 0.003468    39  0.9902 0.0008217       5
```

```
plot(cvfit)
```



Determine which variables were selected using lambda.1se. These are the variables with which the lasso models will be built.

```
sel.vars <- which(coef(cvfit, s=cvfit$lambda.1se)!=0)[-1]-1
sel.names <- colnames(nasa)[sel.vars]
sel.names
```

```
## [1] "Absolute.Magnitude"         "Est.Dia.in.KM.min."
## [3] "Orbit.Uncertainity"         "Minimum.Orbit.Intersection"
## [5] "Mean.Motion"
```

We built our initial lasso model using the variables selected.

```
#fit a lasso model using the selected variables
fit.lasso <- glm(as.factor(Hazardous) ~ Absolute.Magnitude + Est.Dia.in.KM.min. +
                   Orbit.Uncertainity + Minimum.Orbit.Intersection +
                   Mean.Motion, family="binomial", data=nasa)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(fit.lasso)
```

```
##
## Call:
## glm(formula = as.factor(Hazardous) ~ Absolute.Magnitude + Est.Dia.in.KM.min. +
##     Orbit.Uncertainity + Minimum.Orbit.Intersection + Mean.Motion,
##     family = "binomial", data = nasa)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.22020  -0.02263  -0.00116  -0.00001   2.99174
##
## Coefficients:
```

```
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  83.83471    6.10198  13.739  < 2e-16 ***
## Absolute.Magnitude           -3.57931    0.26542 -13.486  < 2e-16 ***
## Est.Dia.in.KM.min.          -17.12997    1.54537 -11.085  < 2e-16 ***
## Orbit.Uncertainity           -0.13717    0.04917  -2.790  0.00528 **
## Minimum.Orbit.Intersection -129.95192    8.97244 -14.483  < 2e-16 ***
## Mean.Motion                  -0.49633    0.33275  -1.492  0.13580
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2504.11  on 3077  degrees of freedom
## Residual deviance:  504.81  on 3072  degrees of freedom
## AIC: 516.81
##
## Number of Fisher Scoring iterations: 10
```

The model assumptions for independence, normaility, and no influential points are satisfied by the data. However, we also need to check for multicollinearity to ensure that all model assumptions are satisfied.

```
#address the model assumptions of the lasso model -- multicollinearity
vif(fit.lasso)
```

```
##         Absolute.Magnitude         Est.Dia.in.KM.min.
##                  13.374033                   8.520165
##         Orbit.Uncertainity Minimum.Orbit.Intersection
##                   1.365700                   3.027934
##                Mean.Motion
##                   1.130061
```

Both Absoulte.Magnitude and Est.Dia.in.KM.min. have a VIF>5 indicating multicollinearity. We dedcided to intially adjust our model by removing Abosulte.Magnitude as it has a larger VIF.

```
#adjust the model based on multicollinearlity issues
#remove Absolute.Magnitude
fit.lasso2 <- glm(as.factor(Hazardous) ~ Est.Dia.in.KM.min. + Orbit.Uncertainity +
                   Minimum.Orbit.Intersection + Mean.Motion,
                family="binomial", data=nasa)
summary(fit.lasso2)
```

```
##
## Call:
## glm(formula = as.factor(Hazardous) ~ Est.Dia.in.KM.min. + Orbit.Uncertainity +
##     Minimum.Orbit.Intersection + Mean.Motion, family = "binomial",
##     data = nasa)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1854  -0.3306  -0.0984  -0.0036   3.1953
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  2.34246    0.27245   8.598  < 2e-16 ***
## Est.Dia.in.KM.min.           4.65616    0.51422   9.055  < 2e-16 ***
## Orbit.Uncertainity          -0.51699    0.03237 -15.974  < 2e-16 ***
## Minimum.Orbit.Intersection -62.10827    4.01705 -15.461  < 2e-16 ***
```

```
## Mean.Motion                 -1.41537    0.23374  -6.055  1.4e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2504.1  on 3077  degrees of freedom
## Residual deviance: 1189.8  on 3073  degrees of freedom
## AIC: 1199.8
##
## Number of Fisher Scoring iterations: 8
```

```
#assess multicollinearity of the adjusted model
vif(fit.lasso2)
```

```
##        Est.Dia.in.KM.min.         Orbit.Uncertainity
##                  2.155630                   1.479609
## Minimum.Orbit.Intersection                Mean.Motion
##                  2.135797                   1.080829
```

Removing Absolute.Magnitude fixed the multicollinearity issues. All assumptions are satisfied for the adjusted model. We then tested how well the model fit the data using the Hosmer and Lemeshow goodness of fit test.

$$H_0\text{: the model fits the data well}$$

$$H_1\text{: the model does not fit the data well}$$

$$\alpha = 0.05$$

```
#test the goodness of fit of the adjusted model
hoslem.test(fit.lasso2$y, fit.lasso2$fitted.values)
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  fit.lasso2$y, fit.lasso2$fitted.values
## X-squared = 9.4684, df = 8, p-value = 0.3043
```

There is statistically sufficent evidence (p = 0.436, df = 8) to conclude that the model fits the data well.

We then replaced Est.Dia.in.KM.min. with Absoulte.Magnitude (adjuting for multicollinearity) to see if this model would be a better fit for the data.

```
#adjust the model based on multicollinearlity issues
#remove Est.Dia.in.KM.min
fit.lasso3 <- glm(as.factor(Hazardous) ~ Absolute.Magnitude + Orbit.Uncertainity +
                    Minimum.Orbit.Intersection + Mean.Motion,
                family="binomial", data=nasa)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(fit.lasso3)
```

```
##
## Call:
## glm(formula = as.factor(Hazardous) ~ Absolute.Magnitude + Orbit.Uncertainity +
##     Minimum.Orbit.Intersection + Mean.Motion, family = "binomial",
##     data = nasa)
##
## Deviance Residuals:
```

```
##      Min        1Q    Median        3Q       Max
## -2.84132  -0.10369  -0.01797  -0.00008   2.56744
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  35.50759    2.16373  16.410  < 2e-16 ***
## Absolute.Magnitude           -1.49105    0.09507 -15.684  < 2e-16 ***
## Orbit.Uncertainity           -0.16626    0.04068  -4.087 4.37e-05 ***
## Minimum.Orbit.Intersection -109.77605    6.87597 -15.965  < 2e-16 ***
## Mean.Motion                  -0.53944    0.28509  -1.892   0.0585 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2504.11  on 3077  degrees of freedom
## Residual deviance:  690.15  on 3073  degrees of freedom
## AIC: 700.15
##
## Number of Fisher Scoring iterations: 9
```

```r
#assess multicollinearity of the adjusted model
vif(fit.lasso3)
```

```
##         Absolute.Magnitude         Orbit.Uncertainity
##                   3.045975                   1.406947
## Minimum.Orbit.Intersection                Mean.Motion
##                   2.861872                   1.094045
```

There are no multicollinearity issues in the adjusted model.

Hosmer and Lemeshow goodness of fit test:

$$H_0\text{: the model fits the data well}$$

$$H_1\text{: the model does not fit the data well}$$

$$\alpha = 0.05$$

```r
#test the goodness of fit of the adjusted model
hoslem.test(fit.lasso3$y, fit.lasso3$fitted.values)
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  fit.lasso3$y, fit.lasso3$fitted.values
## X-squared = 7.0768, df = 8, p-value = 0.5284
```

There is statistically sufficient evidence (p = 0.5285, df = 8) to conclude that the model fits the data well.

Fit.lasso3 is the best fit of the lasso model as the Hosmer and Lemeshow goodness of fit (GOF) test results in a higher p-value for fit.lasso3 than fit.lasso2. This is the model in which we base the cross-validated predicition off of.

```r
#create the cross-validated model using the selected variables
set.seed(1)
fit.cv <- train(as.factor(Hazardous) ~ Est.Dia.in.KM.min. + Orbit.Uncertainity +
                   Minimum.Orbit.Intersection + Mean.Motion ,
                   method = "glm", family = "binomial",
                   trControl = trainControl(method="cv", number=5,
```

```r
                    savePredictions = TRUE, classProbs = TRUE),data=nasa)
fit.cv
```

```
## Generalized Linear Model
##
## 3078 samples
##    4 predictor
##    2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2462, 2463, 2463, 2462, 2462
## Resampling results:
##
##   Accuracy   Kappa
##   0.9204107  0.6397316
```

```r
#determine the final model
summary(fit.cv$finalModel)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1854  -0.3306  -0.0984  -0.0036   3.1953
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   2.34246    0.27245   8.598  < 2e-16 ***
## Est.Dia.in.KM.min.            4.65616    0.51422   9.055  < 2e-16 ***
## Orbit.Uncertainity           -0.51699    0.03237 -15.974  < 2e-16 ***
## Minimum.Orbit.Intersection -62.10827    4.01705 -15.461  < 2e-16 ***
## Mean.Motion                  -1.41537    0.23374  -6.055  1.4e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2504.1  on 3077  degrees of freedom
## Residual deviance: 1189.8  on 3073  degrees of freedom
## AIC: 1199.8
##
## Number of Fisher Scoring iterations: 8
```

Verify that there are no multicollinearity issues in the final model and that the model fits the data well.

```r
#asess the mulitcollinearity of the final model
vif(fit.cv$finalModel)
```

```
##         Est.Dia.in.KM.min.         Orbit.Uncertainity
##                   2.155630                   1.479609
## Minimum.Orbit.Intersection                Mean.Motion
##                   2.135797                   1.080829
```

$$H_0\text{: the model fits the data well}$$

$$H_1\text{: the model does not fit the data well}$$

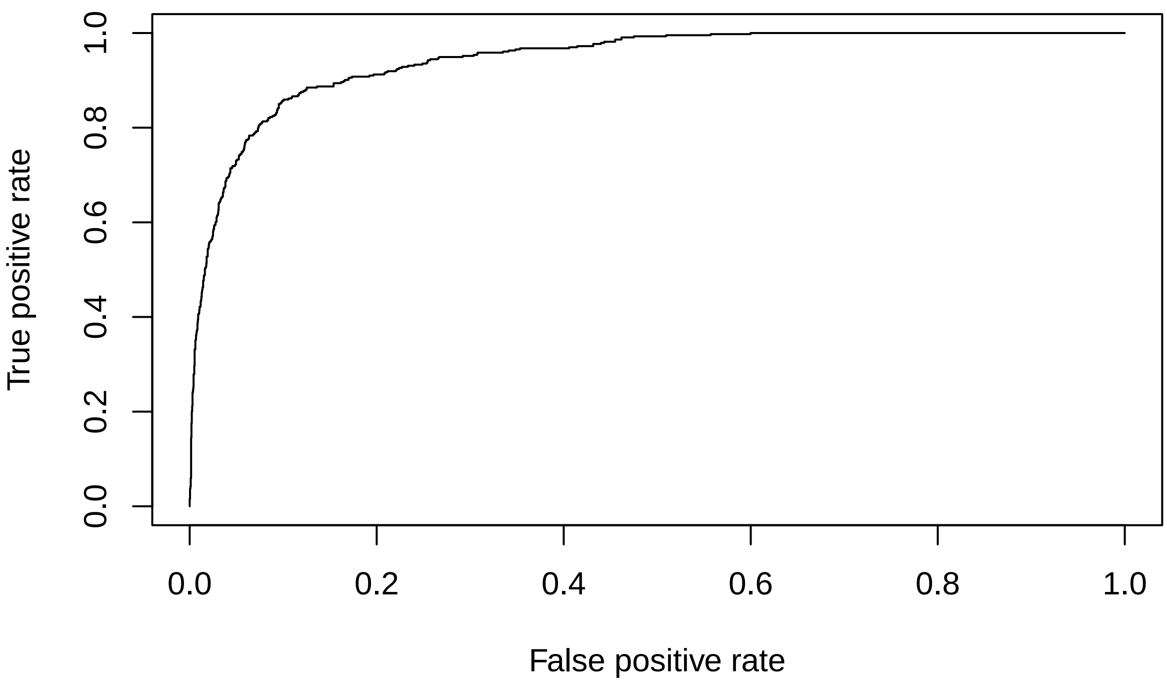
$$\alpha = 0.05$$

```r
#goodness of fit
hoslem.test(fit.cv$finalModel$y, fit.cv$finalModel$fitted.values)
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  fit.cv$finalModel$y, fit.cv$finalModel$fitted.values
## X-squared = 9.4684, df = 8, p-value = 0.3043
```

We then assessed the predictive performance of the model by plotting the ROC curve and finding the area under the curve.

```r
#plot the ROC curve
pihatcv <- fit.cv$pred
predcv <- prediction(pihatcv$True, pihatcv$obs)
perfcv <- performance(predcv, "tpr", "fpr")
plot(perfcv)
```




```r
#find the area under the ROC curve
auccv <- performance(predcv, "auc")@y.values
auccv
```

```
## [[1]]
## [1] 0.9435641
```

Analyze the data analysis by creating a confusion matrix and generating performance statistics.

```r
confusionMatrix(pihatcv$pred, pihatcv$obs, positive="True")
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction False True
```

```
##      False  2567  168
##      True     77  266
##
##                Accuracy : 0.9204
##                  95% CI : (0.9103, 0.9297)
##     No Information Rate : 0.859
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6399
##
##  Mcnemar's Test P-Value : 8.93e-09
##
##             Sensitivity : 0.61290
##             Specificity : 0.97088
##          Pos Pred Value : 0.77551
##          Neg Pred Value : 0.93857
##              Prevalence : 0.14100
##          Detection Rate : 0.08642
##    Detection Prevalence : 0.11144
##       Balanced Accuracy : 0.79189
##
##        'Positive' Class : True
##
```

```
# Confusion matrix
conf.lasso <- table(pihatcv$pred, pihatcv$obs)
conf.lasso
```

```
##
##         False True
##   False  2567  168
##   True     77  266
```

```
lasso.stats <- get_stats(conf.lasso)
lasso.stats
```

```
##           name      value
## 1     accuracy 0.92040286
## 2   error rate 0.07959714
## 3    precision 0.61290323
## 4  sensitivity 0.77551020
## 5  specificity 0.93857404
## 6    F-measure 0.68468468
## 7 Matthew's CC 0.64565361
```

k-Nearest Neighbor

The KNN makes no assumptions and has relatively few parameters to specify (k and a distance measure). We normalized our data before performing the analysis.

```
# function to normalize data
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x))) }
arr.norm <- apply(nasa[,-21], 2, normalize)
arr.norm <- data.frame(arr.norm, nasa$Hazardous)
colnames(arr.norm)[colnames(arr.norm) == "nasa.Hazardous"] ="Hazardous"
```

We used the 5-fold cross validated model to choose k, and we found that 5 neighbors are used in the final model.

```
# 5-fold CV to choose k
set.seed(1)
arr.norm$Hazardous <- as.factor(arr.norm$Hazardous)
fit.knn <- train(Hazardous ~ .,
  method = "knn",
  tuneGrid = expand.grid(k = 1:21),
  trControl = trainControl(method="cv", number=5, savePredictions = TRUE, classProbs = TRUE),
  metric = "Accuracy",
  data = arr.norm)
fit.knn
```

```
## k-Nearest Neighbors
##
## 3078 samples
##   20 predictor
##    2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2462, 2463, 2463, 2462, 2462
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.8645222  0.4227282
##    2  0.8534785  0.3734312
##    3  0.8804429  0.4254110
##    4  0.8749203  0.3926299
##    5  0.8814180  0.4071912
##    6  0.8778418  0.3926183
##    7  0.8801167  0.3847148
##    8  0.8797931  0.3796854
##    9  0.8775209  0.3589633
##   10  0.8804466  0.3685311
##   11  0.8801193  0.3552413
##   12  0.8804440  0.3575683
##   13  0.8797936  0.3410263
##   14  0.8797936  0.3371958
##   15  0.8801177  0.3317969
##   16  0.8762200  0.3033070
##   17  0.8788180  0.3181018
##   18  0.8781718  0.3167519
##   19  0.8794684  0.3145760
##   20  0.8768699  0.2957819
##   21  0.8791442  0.3117543
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

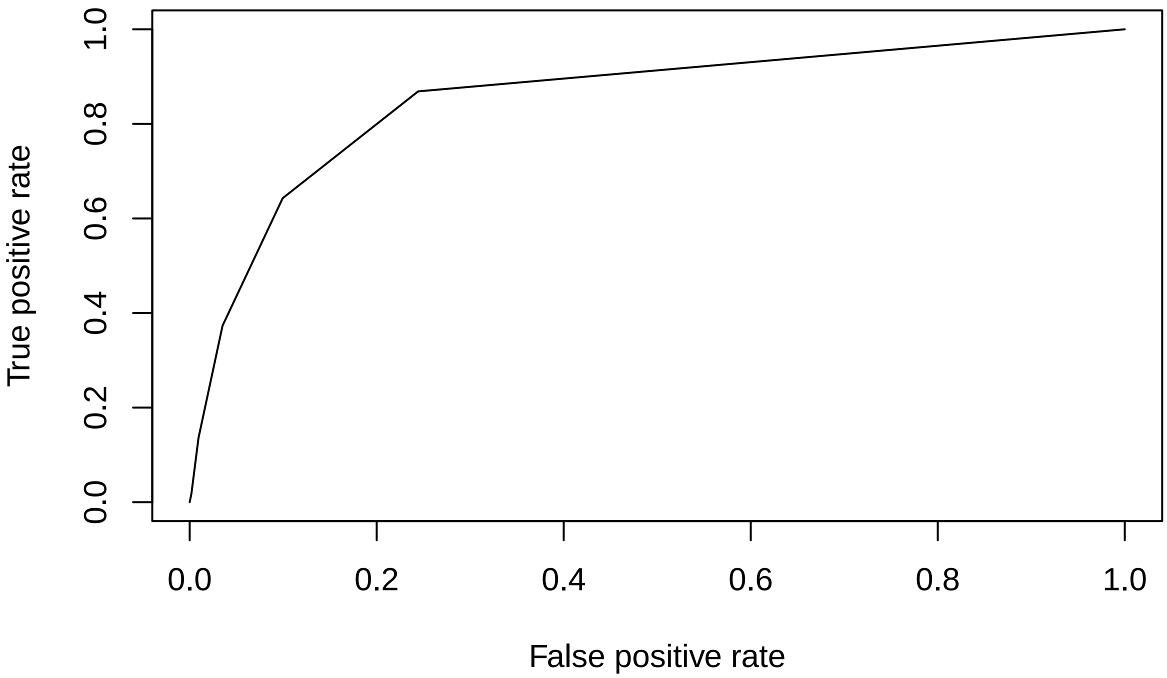
The 10 most important variables in the k-Nearest neighbor models we built are:

```
imp.knn <- rownames(varImp(fit.knn)$importance)
sel.knn <- imp.knn[order(varImp(fit.knn)$importance[,1], decreasing=T)][1:10]
sel.knn
```

```
##  [1] "Absolute.Magnitude"              "Est.Dia.in.KM.min."
##  [3] "Est.Dia.in.KM.max."              "Est.Dia.in.KM.range"
##  [5] "Orbit.Uncertainity"              "Minimum.Orbit.Intersection"
##  [7] "Relative.Velocity.in.KM.per.sec" "Perihelion.Distance"
##  [9] "Eccentricity"                    "Close.Approach.Date"
```

We then assessed the predictive performance of the model by plotting the ROC curve and finding the area under the curve, plotting the error rate, and plotting the most important variables.

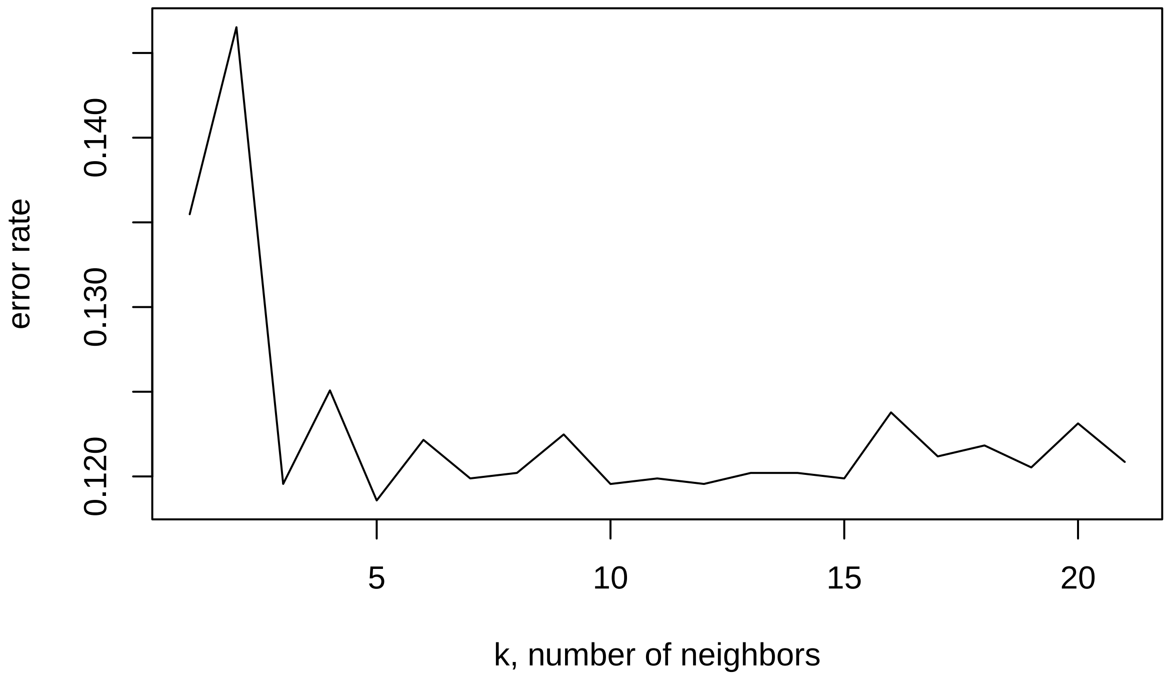
```
#plot the ROC curve
pihatcv.knn <- fit.knn$pred[fit.knn$pred$k == 5,]
pred <- prediction(pihatcv.knn$True, pihatcv.knn$obs)
perf <- performance(pred, "tpr", "fpr")
plot(perf)
```



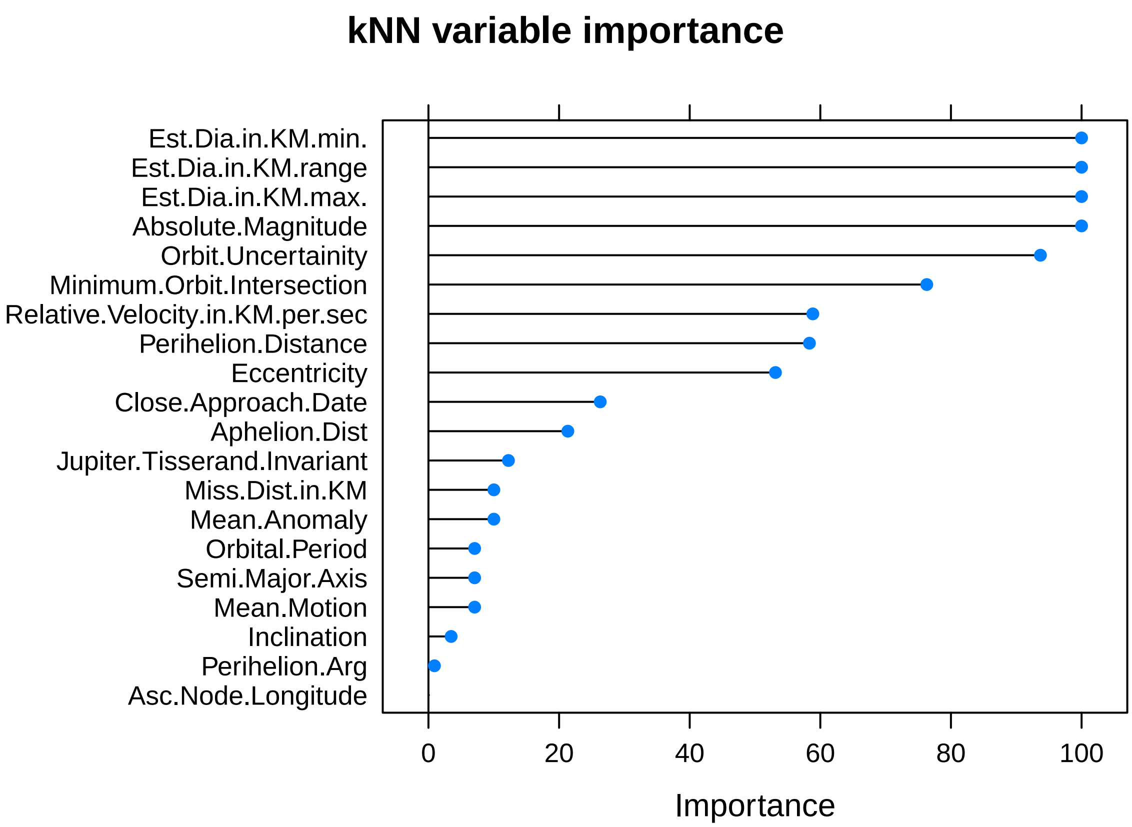

```
# Area under ROC curve (AUC) = concordance index
auc.perf = performance(pred, "auc")
knn_auc <- auc.perf@y.values
knn_auc
```

```
## [[1]]
## [1] 0.8553424
```

```
plot(fit.knn$results[,1], 1-fit.knn$results[,2], type="l",
xlab="k, number of neighbors", ylab="error rate")
```




```
plot(varImp(fit.knn), main="kNN variable importance")
```




```
varImp(fit.knn)
```

```
## ROC curve variable importance
##
##                                  Importance
```

```
## Est.Dia.in.KM.range              100.0000
## Est.Dia.in.KM.max.               100.0000
## Est.Dia.in.KM.min.               100.0000
## Absolute.Magnitude               100.0000
## Orbit.Uncertainity                93.7065
## Minimum.Orbit.Intersection        76.2993
## Relative.Velocity.in.KM.per.sec   58.8524
## Perihelion.Distance               58.3352
## Eccentricity                      53.1304
## Close.Approach.Date               26.3045
## Aphelion.Dist                     21.3395
## Jupiter.Tisserand.Invariant       12.2465
## Miss.Dist.in.KM                   10.0205
## Mean.Anomaly                      10.0178
## Mean.Motion                        7.0710
## Semi.Major.Axis                    7.0710
## Orbital.Period                     7.0710
## Inclination                        3.4734
## Perihelion.Arg                     0.9156
## Asc.Node.Longitude                 0.0000
```

Analyze the data by creating a confusion matrix and generating performance statistics.

```
confusionMatrix(pihatcv.knn$pred, pihatcv.knn$obs, positive="True")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction False True
##      False  2551  272
##      True     93  162
##
##                Accuracy : 0.8814
##                  95% CI : (0.8695, 0.8926)
##     No Information Rate : 0.859
##     P-Value [Acc > NIR] : 0.0001443
##
##                   Kappa : 0.4085
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.37327
##             Specificity : 0.96483
##          Pos Pred Value : 0.63529
##          Neg Pred Value : 0.90365
##              Prevalence : 0.14100
##          Detection Rate : 0.05263
##    Detection Prevalence : 0.08285
##       Balanced Accuracy : 0.66905
##
##        'Positive' Class : True
##
```

```
# Confusion matrix
conf.knn <- table(pihatcv.knn$pred, pihatcv.knn$obs)
conf.knn
```

```
##
##          False True
##    False  2551  272
##    True     93  162
```

```
knn.stats <- get_stats(conf.knn)
knn.stats
```

```
##            name     value
## 1      accuracy 0.8814165
## 2    error rate 0.1185835
## 3     precision 0.3732719
## 4   sensitivity 0.6352941
## 5   specificity 0.9036486
## 6     F-measure 0.4702467
## 7 Matthew's CC 0.4268670
```

Classification Tree Analysis

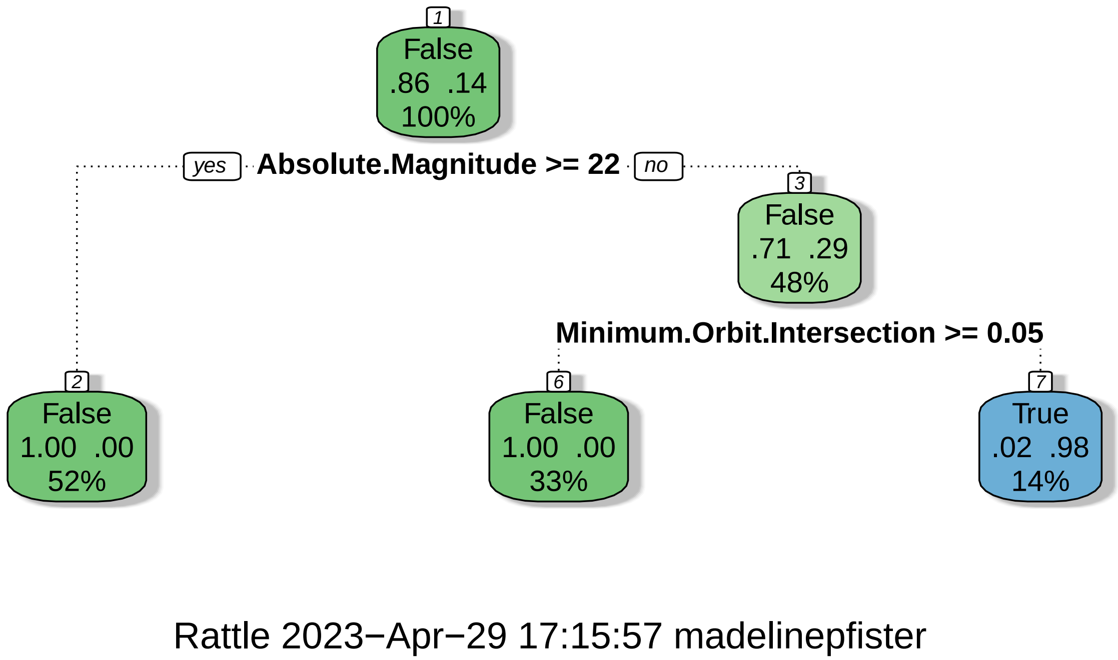
There are no model assumptions to be satisfied for decision trees. We are fitting a classification tree as opposed to a regression tree because the response variable is categorical and binary.

```
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=5,
                                             savePredictions=TRUE,
                                             classProbs=TRUE,
                                             selectionFunction = "oneSE"))
nasa.CVrpart
```

```
## CART
##
## 3078 samples
##   20 predictor
##    2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2462, 2463, 2463, 2462, 2462
## Resampling results across tuning parameters:
##
##   cp     Accuracy   Kappa
##   0.005  0.9948026  0.978491
##   0.010  0.9948026  0.978543
##   0.015  0.9948026  0.978543
##   0.020  0.9948026  0.978543
##   0.025  0.9948026  0.978543
##   0.030  0.9948026  0.978543
##   0.035  0.9948026  0.978543
##   0.040  0.9948026  0.978543
##   0.045  0.9948026  0.978543
##   0.050  0.9948026  0.978543
##
```

```
## Accuracy was used to select the optimal model using  the one SE rule.
## The final value used for the model was cp = 0.05.
```

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



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.

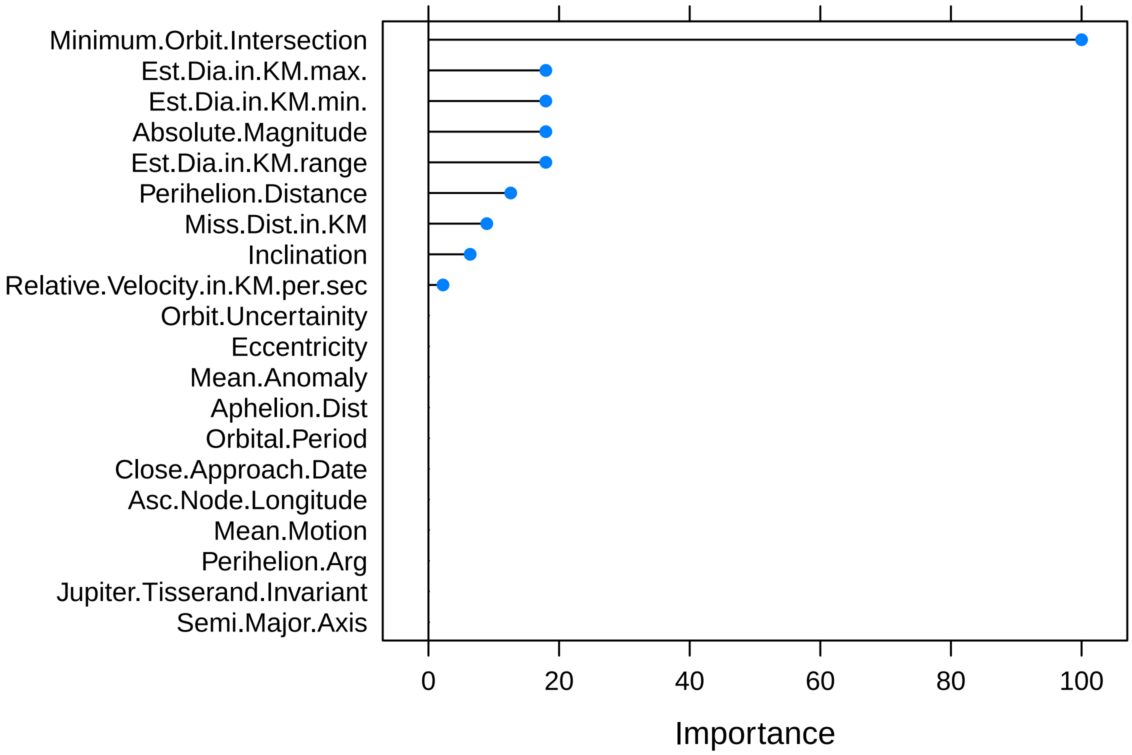
The 10 most important variables in the model are:

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

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

```
## Eccentricity                     0.000
## Aphelion.Dist                    0.000
## Orbit.Uncertainity               0.000
```

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

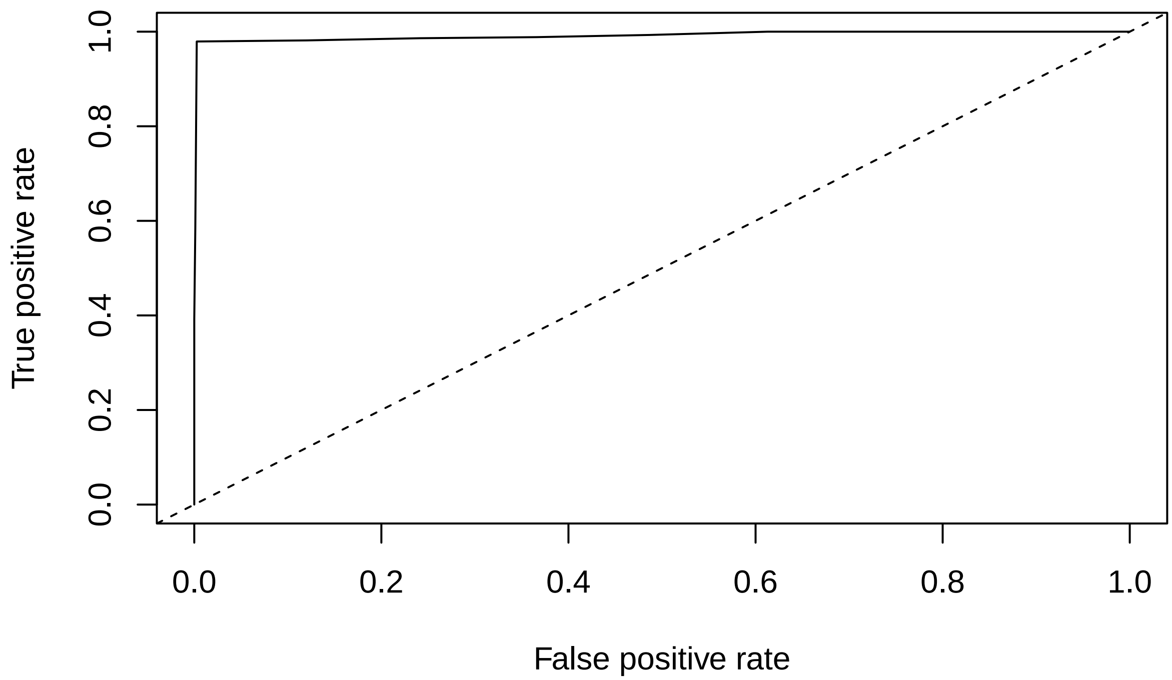


We then assessed the predictive performance by plotting the ROC curve and finding the area under the curve.

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

```
##    pred   obs rowIndex      False        True    cp Resample
## 1  True  True        5 0.01436782 0.985632184 0.005    Fold1
## 2 False False       11 1.00000000 0.000000000 0.005    Fold1
## 3 False False       12 1.00000000 0.000000000 0.005    Fold1
## 4 False False       13 0.99686766 0.003132341 0.005    Fold1
## 5 False False       25 0.99686766 0.003132341 0.005    Fold1
## 6 False False       30 0.99686766 0.003132341 0.005    Fold1
```

```
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] 0.9917045
```

Analyze the data by creating a confusion matrix and generating performance statistics.

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

```
##
##          False True
##   False   2637    7
##   True       9  425
```

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

```
##            name       value
## 1      accuracy 0.994801819
## 2    error rate 0.005198181
## 3     precision 0.983796296
## 4   sensitivity 0.979262673
## 5   specificity 0.997352496
## 6     F-measure 0.981524249
## 7 Matthew's CC 0.978503227
```

Random Forest Analysis

There are no model assumptions to be satisfied for random forest analysis.

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

```
set.seed(1)
nasa.CVrf <- train(Hazardous ~ ., data=nasa,
                   method="rf",
                   trControl=trainControl(method="cv", number=5,
                                          savePredictions=TRUE,
                                          classProbs=TRUE))
nasa.CVrf
```

```
## Random Forest
##
## 3078 samples
##   20 predictor
##    2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2462, 2463, 2463, 2462, 2462
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9944774  0.9769498
##   11    0.9954519  0.9811428
##   20    0.9961013  0.9838326
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 20.
```

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

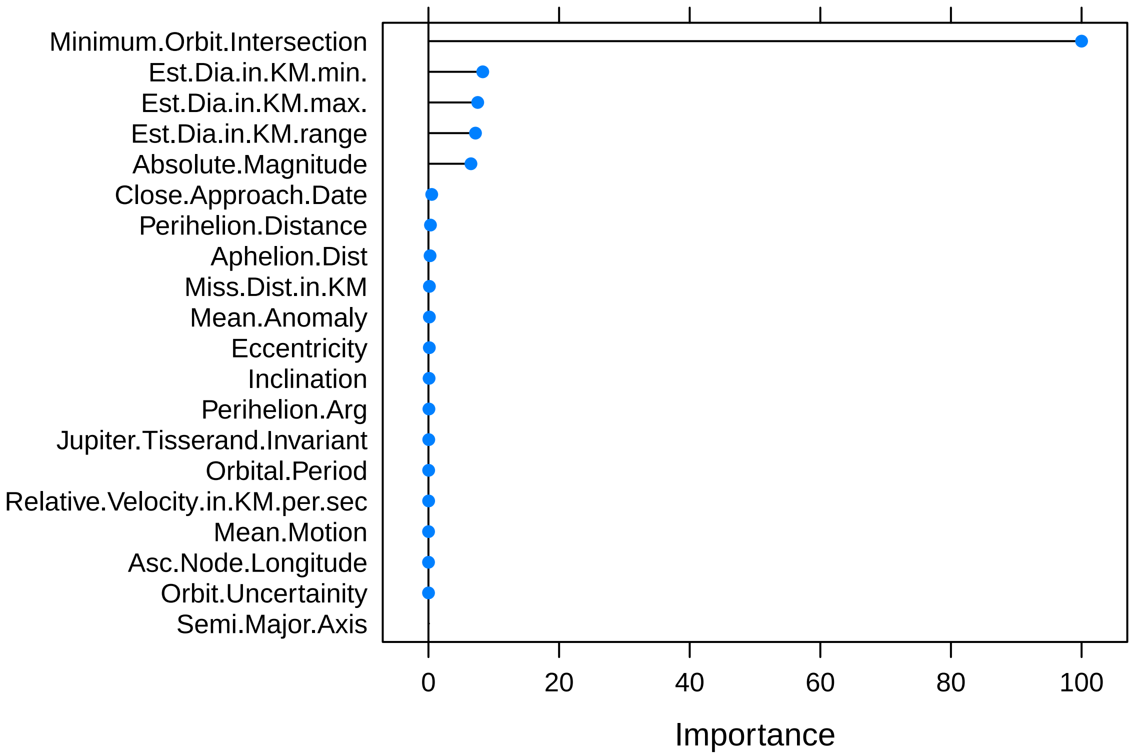
The 10 most important variables is the random forest model are:

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

```
## rf variable importance
##
##                                Overall
## Minimum.Orbit.Intersection      100.00000
## Est.Dia.in.KM.min.                8.31765
## Est.Dia.in.KM.max.                7.54595
## Est.Dia.in.KM.range               7.19967
## Absolute.Magnitude                6.50834
## Close.Approach.Date               0.50215
## Perihelion.Distance               0.31000
## Aphelion.Dist                     0.24640
```

```
## Miss.Dist.in.KM                    0.14536
## Mean.Anomaly                       0.14268
## Eccentricity                       0.13382
## Inclination                        0.09681
## Perihelion.Arg                     0.07256
## Jupiter.Tisserand.Invariant        0.04657
## Orbital.Period                     0.03027
## Relative.Velocity.in.KM.per.sec    0.02670
## Mean.Motion                        0.01485
## Asc.Node.Longitude                 0.01116
## Orbit.Uncertainity                 0.01089
## Semi.Major.Axis                    0.00000
```

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



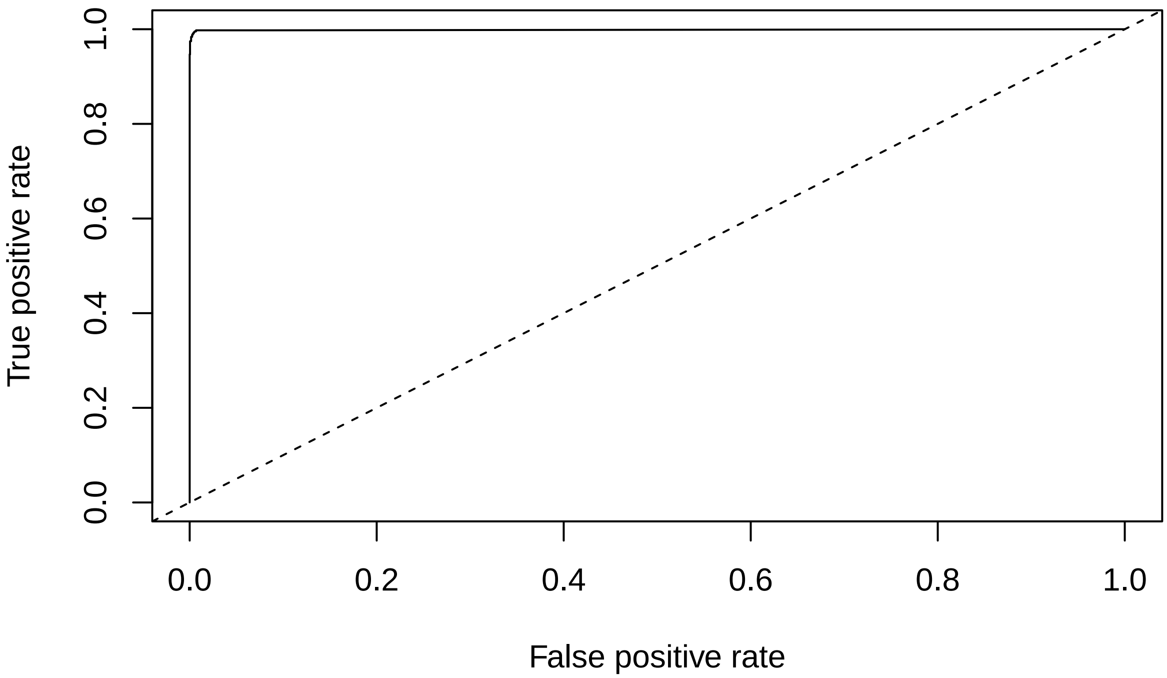
We then assessed the model predictive performance by plotting the ROC curve and finding the model under the curve.

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

```
##    pred   obs False  True rowIndex mtry Resample
## 1  True  True 0.200 0.800        5    2    Fold1
## 2 False False 0.970 0.030       11    2    Fold1
## 3 False False 0.770 0.230       12    2    Fold1
## 4 False False 0.980 0.020       13    2    Fold1
## 5 False False 0.976 0.024       25    2    Fold1
## 6 False False 0.998 0.002       30    2    Fold1
```

```
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)
```

```r
abline(a=0, b=1, lty=2)
```




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

```
## [[1]]
## [1] 0.9986715
```

Analyze the data by creating a confusion matrix and generating performance statistics.

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

```
##
##          False True
##   False  2640    4
##   True      8  426
```

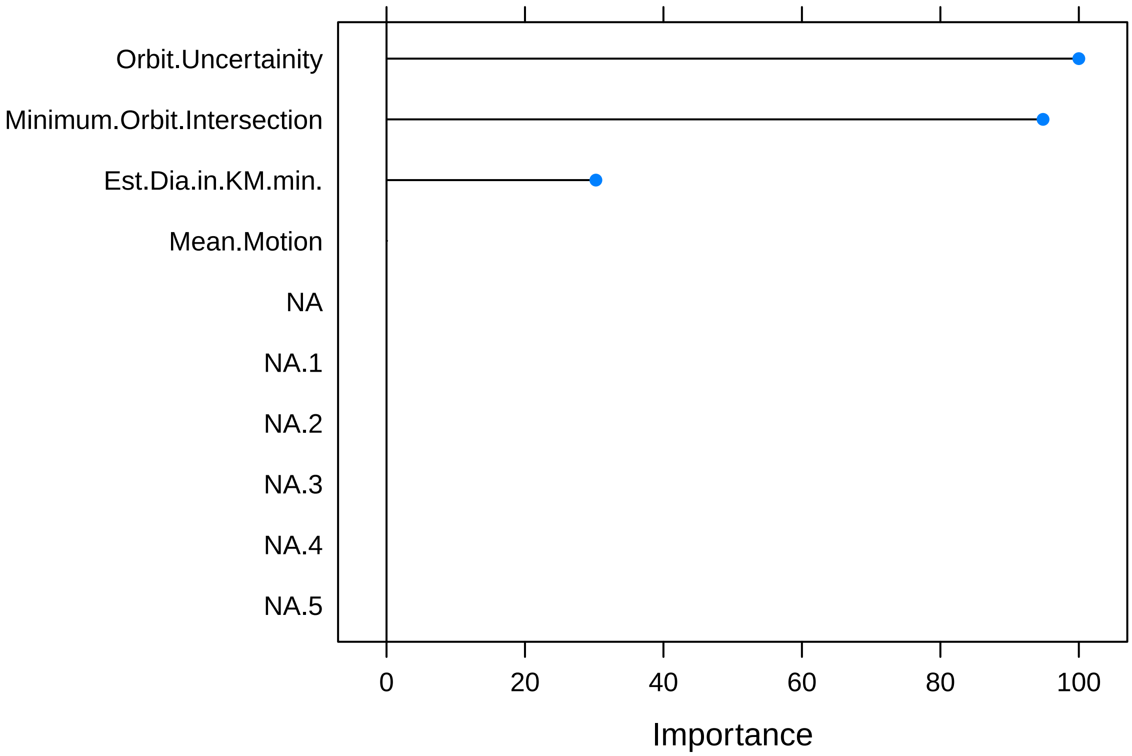
```r
rf.stats <- get_stats(confMat)
rf.stats
```

```
##           name       value
## 1     accuracy 0.996101365
## 2   error rate 0.003898635
## 3    precision 0.990697674
## 4  sensitivity 0.981566820
## 5  specificity 0.998487141
## 6    F-measure 0.986111111
## 7 Matthew's CC 0.983857862
```

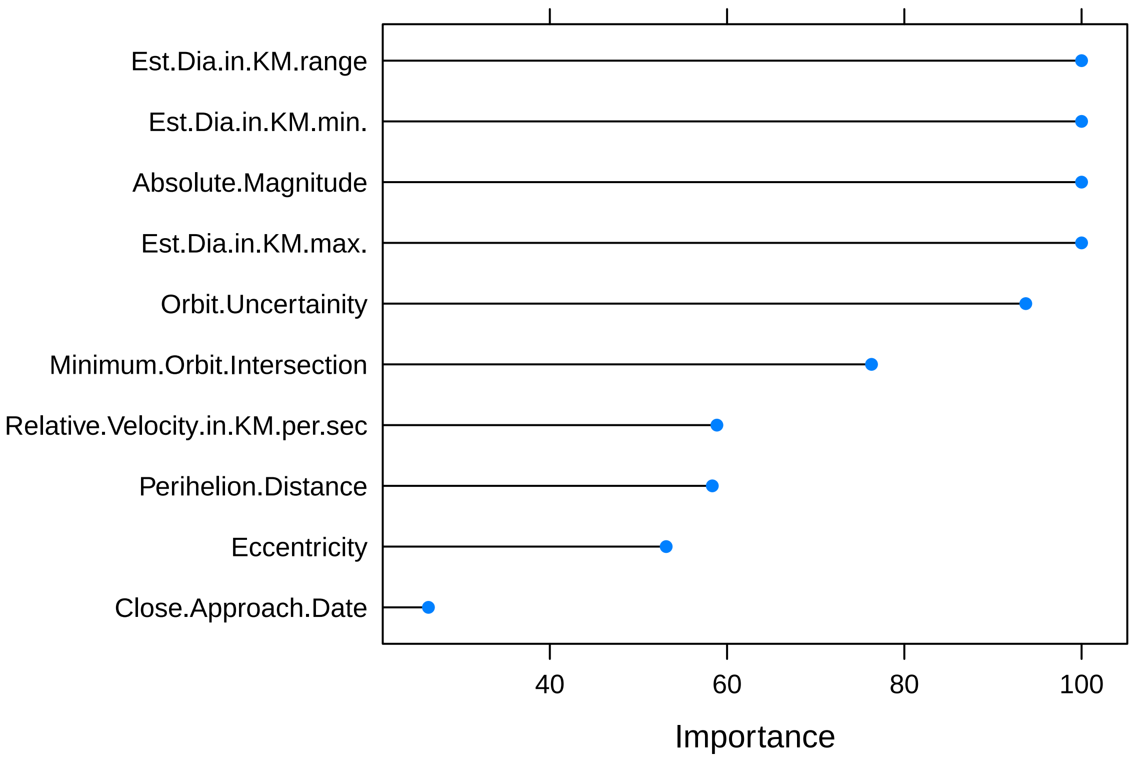
Model Comparisons

Variable Importance Comparison

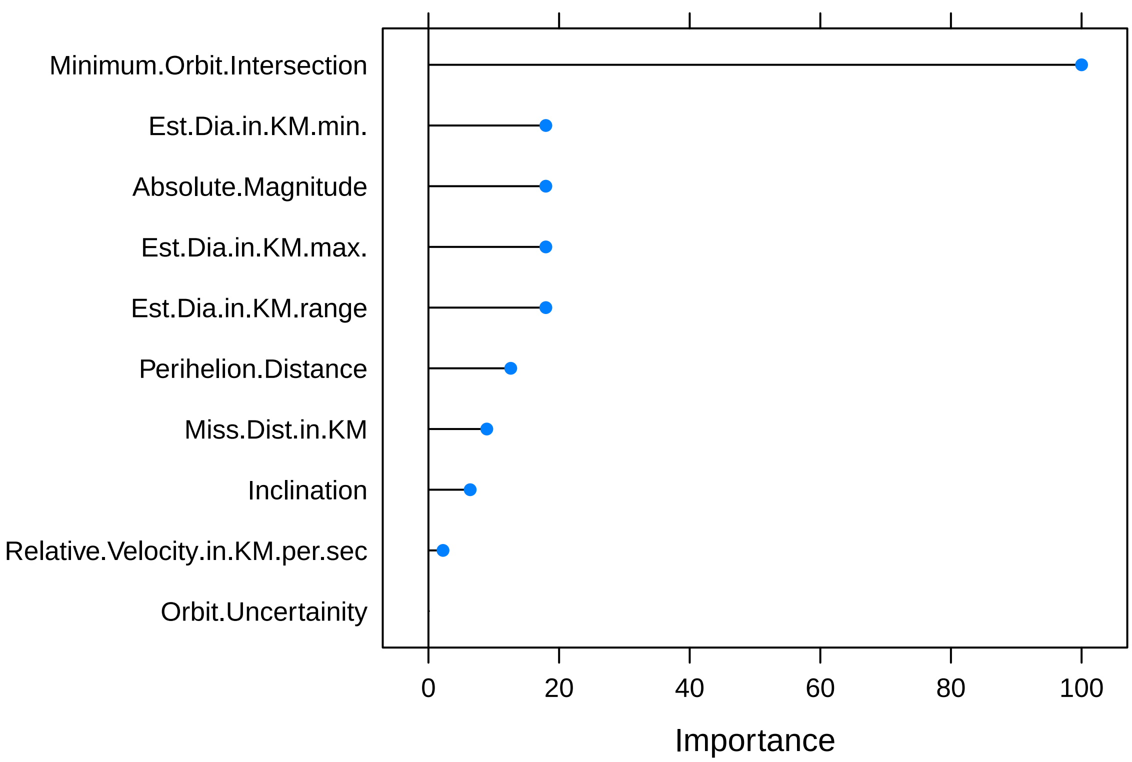
```
par(mfrow=c(2,2))
plot(varImp(fit.cv), top=10)
```



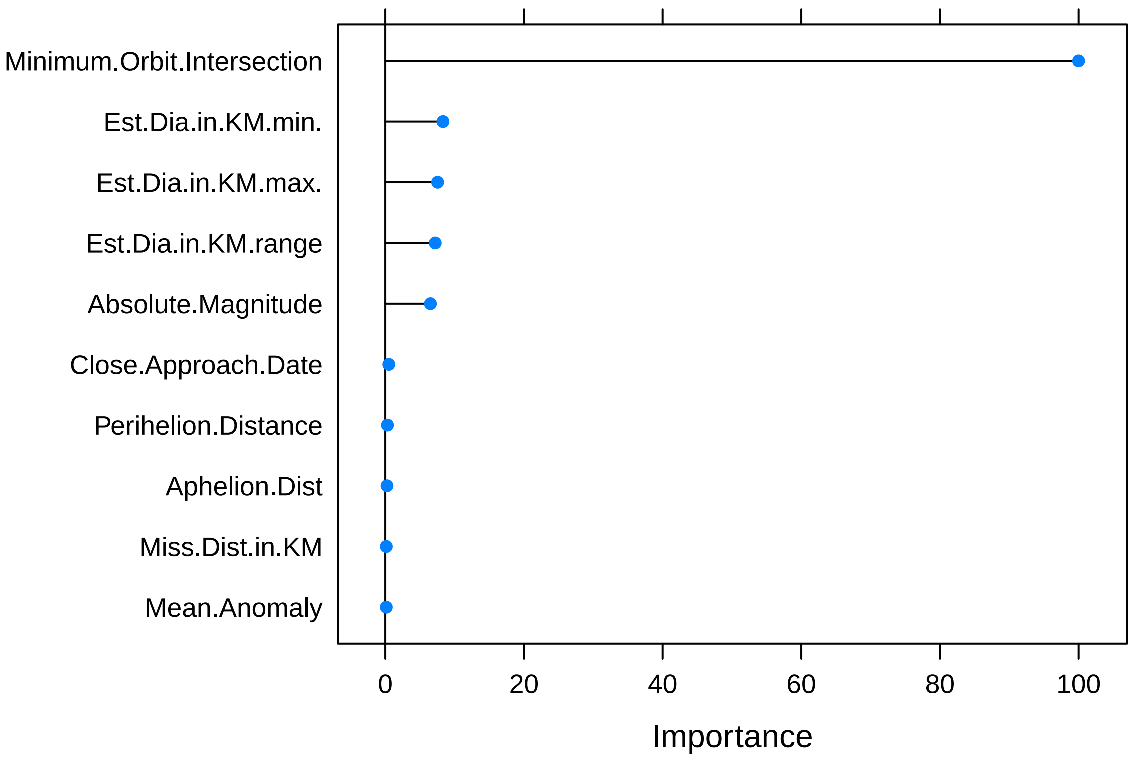

```
plot(varImp(fit.knn), top=10)
```




```
plot(varImp(nasa.CVrpart), top=10)
```




```
plot(varImp(nasa.CVrf), top=10)
```




```
imp.knn <- rownames(varImp(fit.knn)$importance)
sel.knn <- imp.knn[order(varImp(fit.knn)$importance[,1], decreasing=T)][1:10]
imp.rp <- rownames(varImp(nasa.CVrpart)$importance)
sel.rp <- imp.rp[order(varImp(nasa.CVrpart)$importance[,1], decreasing=T)][1:10]
imp.rf <- rownames(varImp(nasa.CVrf)$importance)
sel.rf <- imp.rf[order(varImp(nasa.CVrf)$importance[,1], decreasing=T)][1:10]
intersect(sel.names, intersect(sel.knn, intersect(sel.rp, sel.rf)))
```

```
## [1] "Absolute.Magnitude"         "Est.Dia.in.KM.min."
## [3] "Minimum.Orbit.Intersection"
```

Predicitive Performance Stat Comparison

```
lasso.stats
```

```
##            name      value
## 1      accuracy 0.92040286
## 2    error rate 0.07959714
## 3     precision 0.61290323
## 4   sensitivity 0.77551020
## 5   specificity 0.93857404
## 6     F-measure 0.68468468
## 7 Matthew's CC 0.64565361
```

```
knn.stats
```

```
##            name     value
## 1      accuracy 0.8814165
## 2    error rate 0.1185835
## 3     precision 0.3732719
## 4   sensitivity 0.6352941
```

```
## 5  specificity 0.9036486
## 6    F-measure 0.4702467
## 7 Matthew's CC 0.4268670
```

```r
rpart.stats
```

```
##           name       value
## 1     accuracy 0.994801819
## 2   error rate 0.005198181
## 3    precision 0.983796296
## 4  sensitivity 0.979262673
## 5  specificity 0.997352496
## 6    F-measure 0.981524249
## 7 Matthew's CC 0.978503227
```

```r
rf.stats
```

```
##           name       value
## 1     accuracy 0.996101365
## 2   error rate 0.003898635
## 3    precision 0.990697674
## 4  sensitivity 0.981566820
## 5  specificity 0.998487141
## 6    F-measure 0.986111111
## 7 Matthew's CC 0.983857862
```

```r
df_merge <- merge(lasso.stats,knn.stats,by="name")
colnames(df_merge)[colnames(df_merge) == "value.x"] ="Lasso"
colnames(df_merge)[colnames(df_merge) == "value.y"] ="kNN"
df_merge <- merge(df_merge,rpart.stats,by="name")
df_merge <- merge(df_merge,rf.stats,by="name")
colnames(df_merge)[colnames(df_merge) == "value.x"] ="Classification Tree"
colnames(df_merge)[colnames(df_merge) == "value.y"] ="Random Forest"
df_merge
```

```
##           name      Lasso       kNN Classification Tree Random Forest
## 1     accuracy 0.92040286 0.8814165         0.994801819   0.996101365
## 2   error rate 0.07959714 0.1185835         0.005198181   0.003898635
## 3    F-measure 0.68468468 0.4702467         0.981524249   0.986111111
## 4 Matthew's CC 0.64565361 0.4268670         0.978503227   0.983857862
## 5    precision 0.61290323 0.3732719         0.983796296   0.990697674
## 6  sensitivity 0.77551020 0.6352941         0.979262673   0.981566820
## 7  specificity 0.93857404 0.9036486         0.997352496   0.998487141
```