Asteroid Compiled

Annika Lin

2023-04-04

```
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)
```

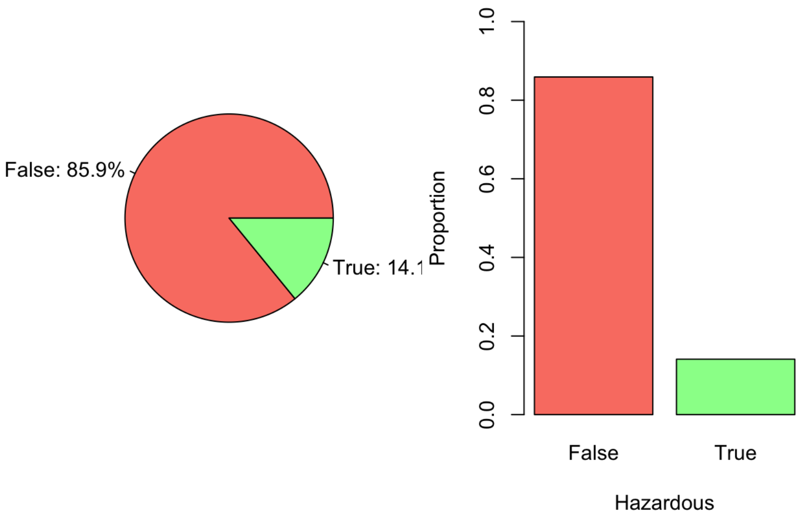
```
## ResourceSelection 0.3-5   2019-07-22
```

```
#import dataset filtered for '2017-04-06'
nasa <- read.csv("~/Documents/Georgetown/Spring23/Statistical Learning & Data Science/Pr
oject/NASA-asteroid-Classification-master/final/nasa.csv")

nasa <- nasa[ , !(names(nasa) %in% c("X"))]

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

```
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)

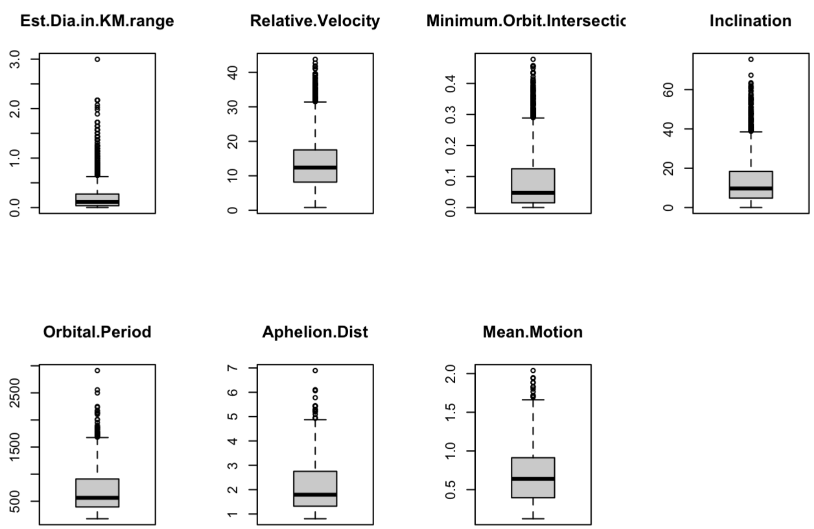
```
# numerical summary + box plots
summary(nasa)
```



```
##  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.
## 2729               15.2           2.424125           5.420508
##      Est.Dia.in.KM.range Close.Approach.Date Relative.Velocity.in.KM.per.sec
## 2729            2.996383            20141222                         23.5172
##      Miss.Dist.in.KM Orbit.Uncertainity Minimum.Orbit.Intersection
## 2729        45467472                  0                   0.153116
##      Jupiter.Tisserand.Invariant Eccentricity Semi.Major.Axis Inclination
## 2729                       4.864    0.3462176        1.261473    36.90474
##      Asc.Node.Longitude Orbital.Period Perihelion.Distance Perihelion.Arg
## 2729           111.2844       517.5058           0.8247287       349.1364
##      Aphelion.Dist Mean.Anomaly Mean.Motion Hazardous
## 2729      1.698217     328.5237   0.6956443     False
```



Predictive Performance Stats

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

  return (stats)
}
```

Lasso’s Penalized Regression

Create the design matrix

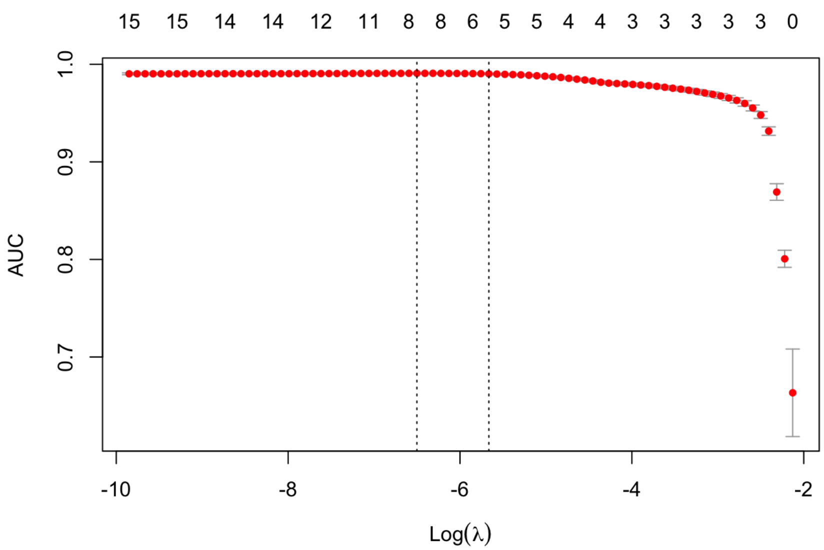
```r
X = model.matrix(Hazardous ~ ., data=nasa)
Y = as.numeric(nasa$Hazardous=="True")
```

Conduct the cross-validation

```r
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
```

```r
plot(cvfit)
```



Variables selected using lambda.1se

```
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"
```

```
#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 + Jupiter.Tisserand.Invariant,
                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 + Jupiter.Tisserand.Invariant,
##     family = "binomial", data = nasa)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.21356  -0.02287  -0.00118  -0.00001   2.97254
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  83.95094    6.09896  13.765  < 2e-16 ***
## Absolute.Magnitude           -3.57347    0.26573 -13.448  < 2e-16 ***
## Est.Dia.in.KM.min.          -17.11936    1.54653 -11.070  < 2e-16 ***
## Orbit.Uncertainity           -0.13672    0.04963  -2.755  0.00587 **
## Minimum.Orbit.Intersection -129.67467    8.94106 -14.503  < 2e-16 ***
## Jupiter.Tisserand.Invariant  -0.12295    0.09307  -1.321  0.18645
## ---
## 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:  505.29  on 3072  degrees of freedom
## AIC: 517.29
##
## Number of Fisher Scoring iterations: 10
```

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

```
##          Absolute.Magnitude          Est.Dia.in.KM.min.
##                   13.402908                    8.509262
##          Orbit.Uncertainity  Minimum.Orbit.Intersection
##                    1.392265                    3.010432
## Jupiter.Tisserand.Invariant
##                    1.159019
```

```
#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 + Jupiter.Tisserand.Invariant,
                family="binomial", data=nasa)
summary(fit.lasso2)
```

```
##
## Call:
## glm(formula = as.factor(Hazardous) ~ Est.Dia.in.KM.min. + Orbit.Uncertainity +
##     Minimum.Orbit.Intersection + Jupiter.Tisserand.Invariant,
##     family = "binomial", data = nasa)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1507  -0.3249  -0.0973  -0.0035   3.1983
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   3.56403    0.41170   8.657  < 2e-16 ***
## Est.Dia.in.KM.min.            4.49382    0.51569   8.714  < 2e-16 ***
## Orbit.Uncertainity           -0.52498    0.03271 -16.049  < 2e-16 ***
## Minimum.Orbit.Intersection  -62.34047    4.02315 -15.495  < 2e-16 ***
## Jupiter.Tisserand.Invariant  -0.43722    0.06464  -6.764 1.34e-11 ***
## ---
## 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: 1180.2  on 3073  degrees of freedom
## AIC: 1190.2
##
## Number of Fisher Scoring iterations: 8
```

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

```
##         Est.Dia.in.KM.min.          Orbit.Uncertainity
##                   2.133114                    1.508538
##  Minimum.Orbit.Intersection Jupiter.Tisserand.Invariant
##                   2.116395                    1.110887
```

```
#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 = 7.974, df = 8, p-value = 0.436
```



```
#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 + Jupiter.Tisserand.Invariant,
               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 + Jupiter.Tisserand.Invariant,
##     family = "binomial", data = nasa)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.83507  -0.10414  -0.01809  -0.00008   2.57268
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   35.69555    2.16622  16.478  < 2e-16 ***
## Absolute.Magnitude            -1.48606    0.09551 -15.558  < 2e-16 ***
## Orbit.Uncertainity            -0.16696    0.04104  -4.069 4.73e-05 ***
## Minimum.Orbit.Intersection  -109.62272    6.86539 -15.967  < 2e-16 ***
## Jupiter.Tisserand.Invariant   -0.13840    0.07946  -1.742   0.0816 .
## ---
## 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:  690.72  on 3073  degrees of freedom
## AIC: 700.72
##
## Number of Fisher Scoring iterations: 9
```

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

```
##         Absolute.Magnitude          Orbit.Uncertainity
##                   3.081047                    1.432653
##  Minimum.Orbit.Intersection Jupiter.Tisserand.Invariant
##                   2.856256                    1.123920
```

```
#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 = 7.0759, df = 8, p-value = 0.5285
```

Fit.lasso3 is the best fit of the lasso model

```
#create the cross-validated model using the selected variables

set.seed(1)
fit.cv <- train(as.factor(Hazardous) ~ Absolute.Magnitude + Est.Dia.in.KM.min. + Orbit.U
ncertainity +
                   Minimum.Orbit.Intersection + Jupiter.Tisserand.Invariant ,
                   method = "glm", family = "binomial", trControl = trainControl(method
="cv", number=5,
                                                 savePredictions = TRUE),data=nasa)
```

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

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

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

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

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

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

```
fit.cv
```

```
## Generalized Linear Model
##
## 3078 samples
##    5 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
##   0.9606937  0.8371297
```



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

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.21356  -0.02287  -0.00118  -0.00001   2.97254
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   83.95094    6.09896  13.765  < 2e-16 ***
## Absolute.Magnitude            -3.57347    0.26573 -13.448  < 2e-16 ***
## Est.Dia.in.KM.min.           -17.11936    1.54653 -11.070  < 2e-16 ***
## Orbit.Uncertainity            -0.13672    0.04963  -2.755  0.00587 **
## Minimum.Orbit.Intersection  -129.67467    8.94106 -14.503  < 2e-16 ***
## Jupiter.Tisserand.Invariant   -0.12295    0.09307  -1.321  0.18645
## ---
## 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:  505.29  on 3072  degrees of freedom
## AIC: 517.29
##
## Number of Fisher Scoring iterations: 10
```

```
#asses the mullitcollinearity of the final model
vif(fit.cv$finalModel)
```

```
##          Absolute.Magnitude          Est.Dia.in.KM.min.
##                   13.402908                    8.509262
##          Orbit.Uncertainity  Minimum.Orbit.Intersection
##                    1.392265                    3.010432
## Jupiter.Tisserand.Invariant
##                    1.159019
```

```
#create a new cross-validated model removing Est.Dia.in.KM.min.
fit.cv2 <- train(as.factor(Hazardous) ~ Absolute.Magnitude + Orbit.Uncertainity +
                   Minimum.Orbit.Intersection + Jupiter.Tisserand.Invariant ,
                   method = "glm", family = "binomial", trControl = trainControl(method
="cv", number=5,
                                                                              classPr
obs = TRUE,
                                                savePredictions = TRUE),data=nasa)
```

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

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

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

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

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

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

```
fit.cv2
```

```
## 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
##   0.9496442  0.7864174
```

```
#determine the final model
summary(fit.cv2$finalModel)
```



```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.83507  -0.10414  -0.01809  -0.00008   2.57268
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   35.69555    2.16622  16.478  < 2e-16 ***
## Absolute.Magnitude            -1.48606    0.09551 -15.558  < 2e-16 ***
## Orbit.Uncertainity            -0.16696    0.04104  -4.069 4.73e-05 ***
## Minimum.Orbit.Intersection  -109.62272    6.86539 -15.967  < 2e-16 ***
## Jupiter.Tisserand.Invariant   -0.13840    0.07946  -1.742   0.0816 .
## ---
## 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:  690.72  on 3073  degrees of freedom
## AIC: 700.72
##
## Number of Fisher Scoring iterations: 9
```

```
#assess the multicollinearity of the adjusted cross-validated model
vif(fit.cv2$finalModel)
```

```
##          Absolute.Magnitude          Orbit.Uncertainity
##                    3.081047                    1.432653
##  Minimum.Orbit.Intersection Jupiter.Tisserand.Invariant
##                    2.856256                    1.123920
```

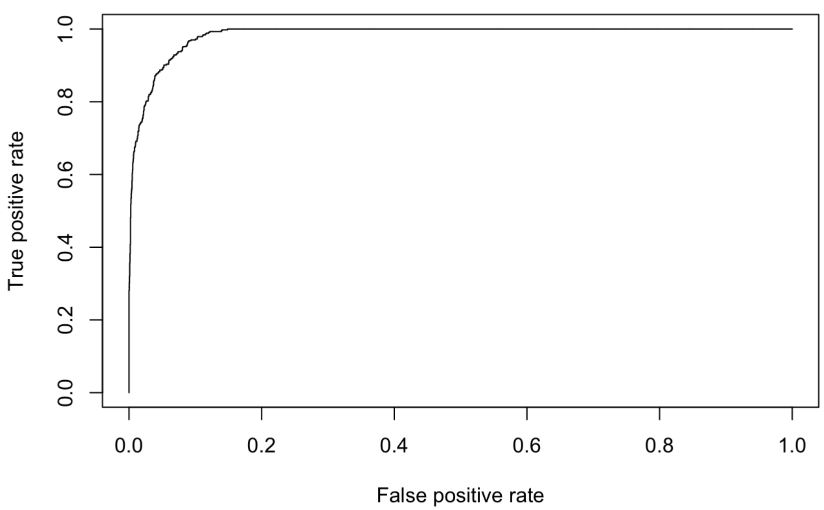
```
#goodness of fit
hoslem.test(fit.cv2$finalModel$y, fit.cv2$finalModel$fitted.values)
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  fit.cv2$finalModel$y, fit.cv2$finalModel$fitted.values
## X-squared = 7.0759, df = 8, p-value = 0.5285
```

```
#assess the predictive performance using the optimized model
pihat <- predict(fit.cv2, type="prob")
head(cbind(nasa$Hazardous, pihat, predict(fit.cv2)))
```

```
##   nasa$Hazardous     False         True predict(fit.cv2)
## 1           True 0.6572134 3.427866e-01            False
## 2          False 1.0000000 2.043709e-08            False
## 3           True 0.4556698 5.443302e-01             True
## 4          False 0.9993470 6.530091e-04            False
## 5           True 0.7499228 2.500772e-01            False
## 6          False 0.9871813 1.281874e-02            False
```

```
pred <- prediction(pihat[,2], nasa$Hazardous)
perf <- performance(pred, "tpr", "fpr")
plot(perf)
```




```
auc <- performance(pred, "auc")@y.values
auc
```

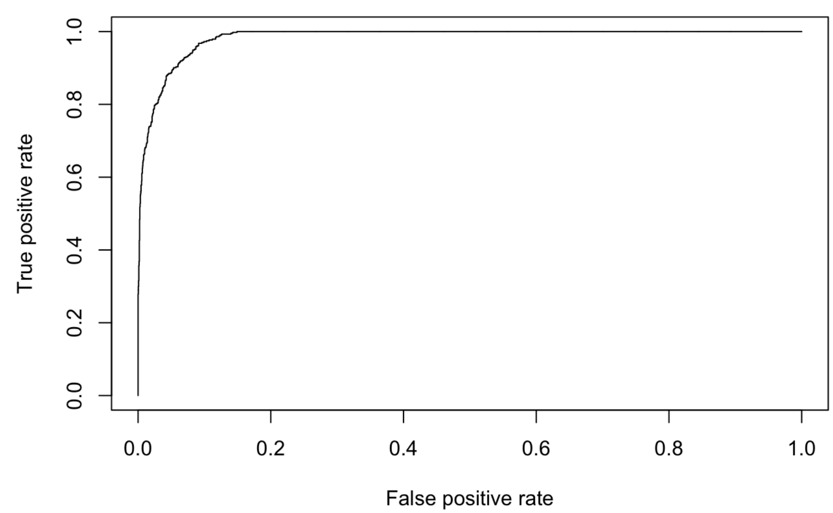
```
## [[1]]
## [1] 0.9840792
```



```
#assess the predictive performance using the predictive model
# pihatcv <- fit.cv2$pred
# head(cbind(nasa$Hazardous[pihatcv$rowIndex], pihatcv))
# predcv <- prediction(pihatcv$True, pihatcv$obs)
# perfcv <- performance(predcv, "tpr", "fpr")
# plot(perfcv)
# auccv <- performance(predcv, "auc")@y.values
# auccv

predprob.lasso <- fit.cv2$pred
head(predprob.lasso)
```

```
##    pred   obs      False        True rowIndex parameter Resample
## 1  True  True 0.44764628 0.552353716        3      none    Fold1
## 2  True  True 0.27155987 0.728440134        7      none    Fold1
## 3 False False 0.99548421 0.004515794       16      none    Fold1
## 4 False False 0.99883589 0.001164109       17      none    Fold1
## 5 False False 0.99724971 0.002750294       20      none    Fold1
## 6  True False 0.04122548 0.958774516       32      none    Fold1
```

```
pred.lasso = prediction(predprob.lasso$True, predprob.lasso$obs)
perf.lasso = performance(pred.lasso, "tpr", "fpr")
plot(perf.lasso)
```



```
auc.lasso = performance(pred.lasso, "auc")@y.values
auc.lasso
```

```
## [[1]]
## [1] 0.983586
```

```
confusionMatrix(predprob.lasso$pred, predprob.lasso$obs, positive="True")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction False True
##      False  2581   92
##      True     63  342
##
##                Accuracy : 0.9496
##                  95% CI : (0.9413, 0.9571)
##     No Information Rate : 0.859
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7861
##
##  Mcnemar's Test P-Value : 0.02451
##
##             Sensitivity : 0.7880
##             Specificity : 0.9762
##          Pos Pred Value : 0.8444
##          Neg Pred Value : 0.9656
##              Prevalence : 0.1410
##          Detection Rate : 0.1111
##    Detection Prevalence : 0.1316
##       Balanced Accuracy : 0.8821
##
##        'Positive' Class : True
##
```

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

```
##
##         False True
##   False  2581   92
##   True     63  342
```

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



```
##            name      value
## 1      accuracy 0.94964263
## 2    error rate 0.05035737
## 3     precision 0.78801843
## 4   sensitivity 0.84444444
## 5   specificity 0.96558174
## 6     F-measure 0.81525626
## 7 Matthew's CC 0.78677483
```

k-Nearest Neighbor

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

KNN makes no assumptions and has relatively few parameters to specify (k and a distance measure).

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

```
# 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"
```

```
# 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 = T
RUE),
  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.
```

5 neighbors are used in the final model.

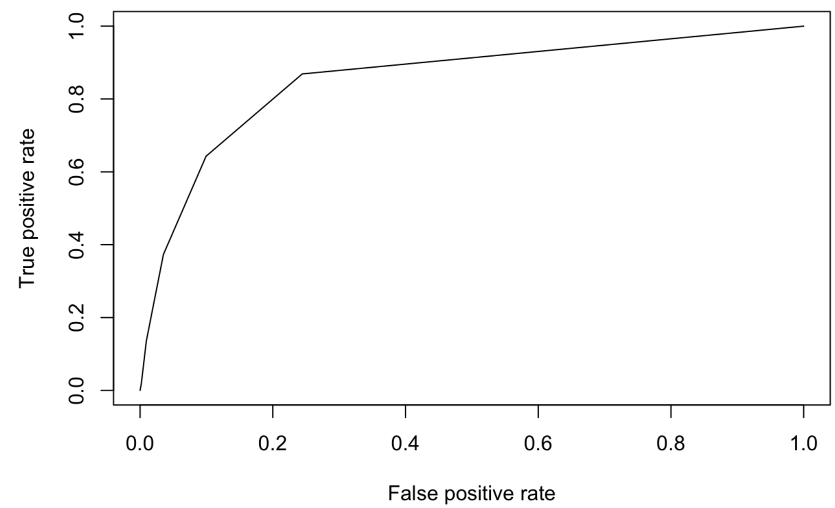
10 most important variables are:

```
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"
```



```
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)
```

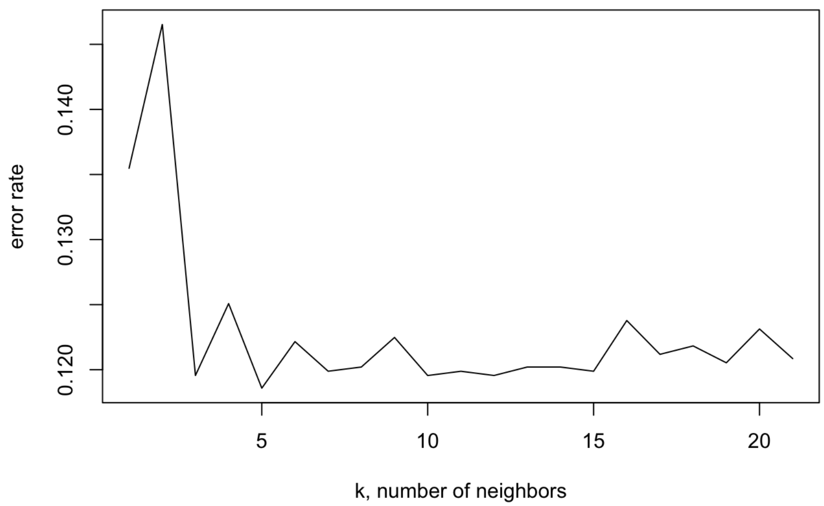



```
# Area under ROC curve (AUC) = concordance index
auc.perf = performance(pred, "auc")
knn_auc <- auc.perf@y.values
knn_auc
```

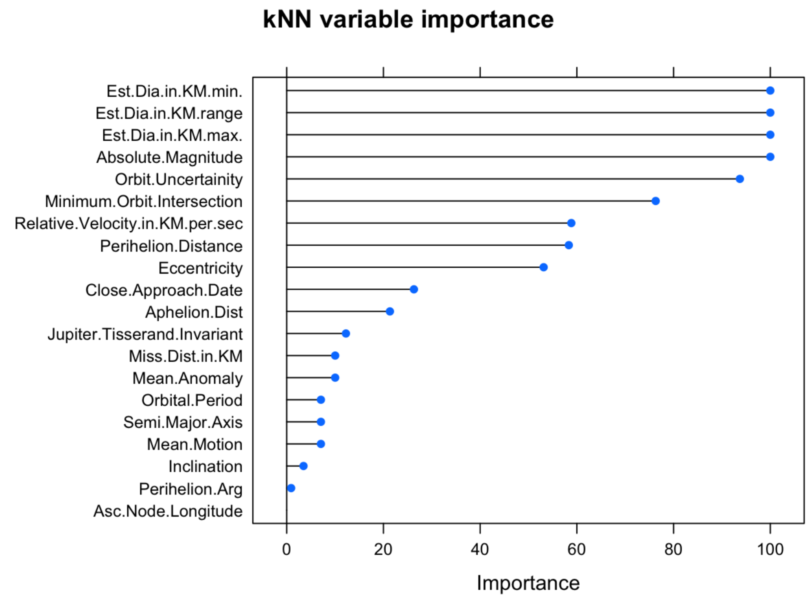
```
## [[1]]
## [1] 0.8553424
```

KNN has an AUC of 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
```

```
predprob.knn <- fit.knn$pred
predprob.knn <- predprob.knn[predprob.knn$k==as.numeric(fit.knn$bestTune),]

pred.knn = prediction(predprob.knn$True, predprob.knn$obs)
perf.knn = performance(pred.knn, "tpr", "fpr")
knn.cut <- cbind(unlist(perf.knn@y.values), unlist(perf.knn@x.values), unlist(perf.knn@a
lpha.values))
knn.cut[knn.cut[,1]>0.7 & knn.cut[,2]<0.3,]
```

```
## [1] 0.8686636 0.2443268 0.2000000
```

```
knn.cut[knn.cut[,3] == 0.5,]
```

```
##      [,1] [,2] [,3]
```

```
confusionMatrix(predprob.knn$pred, predprob.knn$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(predprob.knn$pred, predprob.knn$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

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.

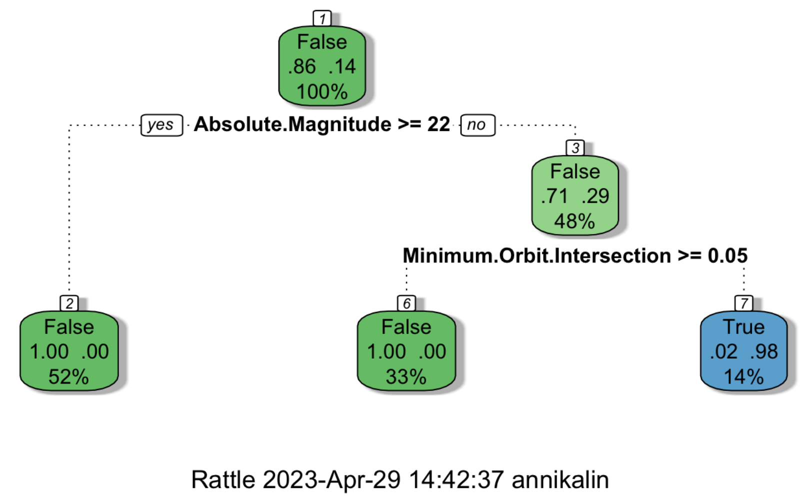
2. Assess the model fit and perform diagnostics, if appropriate.

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

```
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)
```

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

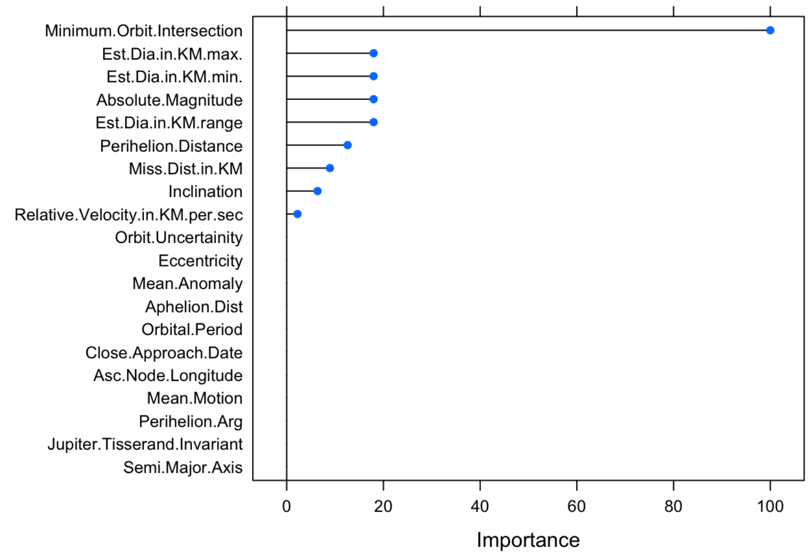
If the plain rpart method is utilized, it selects a cp (complexity parameter) value of 0.05 to maximize accuracy for the final model. Note that the accuracy and kappa values are identical for each of the cp values tried by the model, so any choice within that range should produce comparable results.

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.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))
```

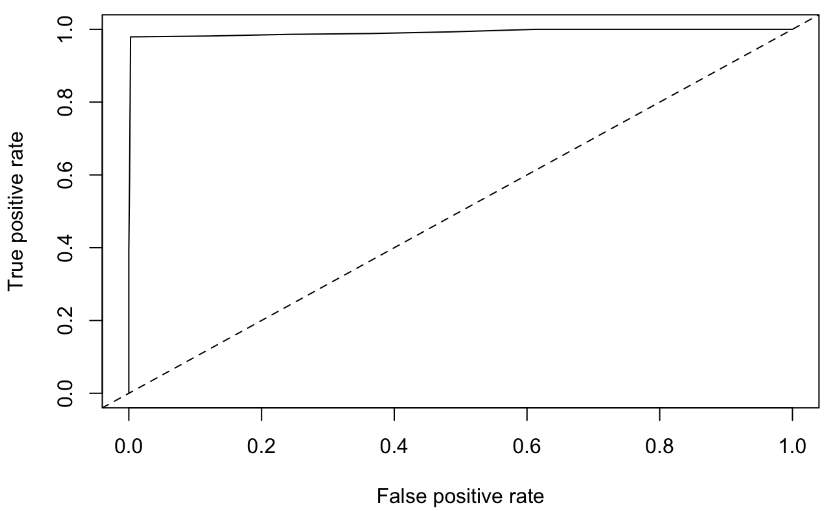



5. Evaluate the cross-validated (CV) predictive performance.

```
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
```

```
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

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)
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.
```

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.

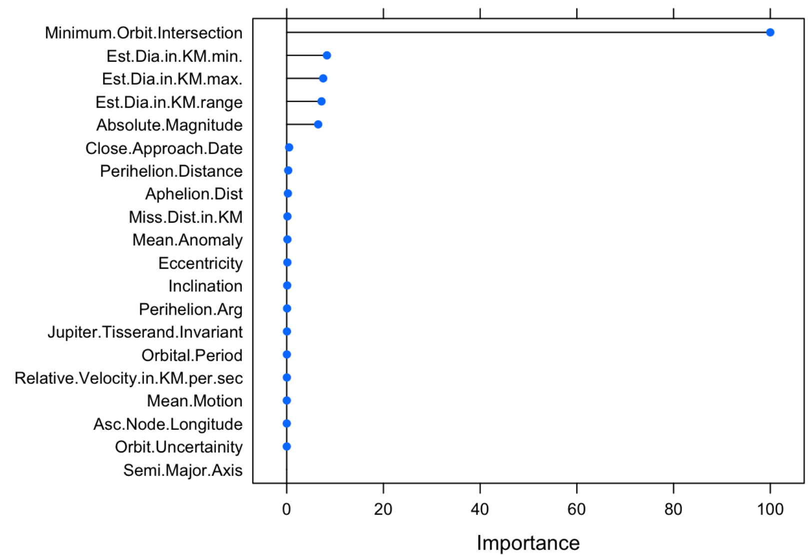
4. Identify and interpret the effect of selected variables.

```
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))
```

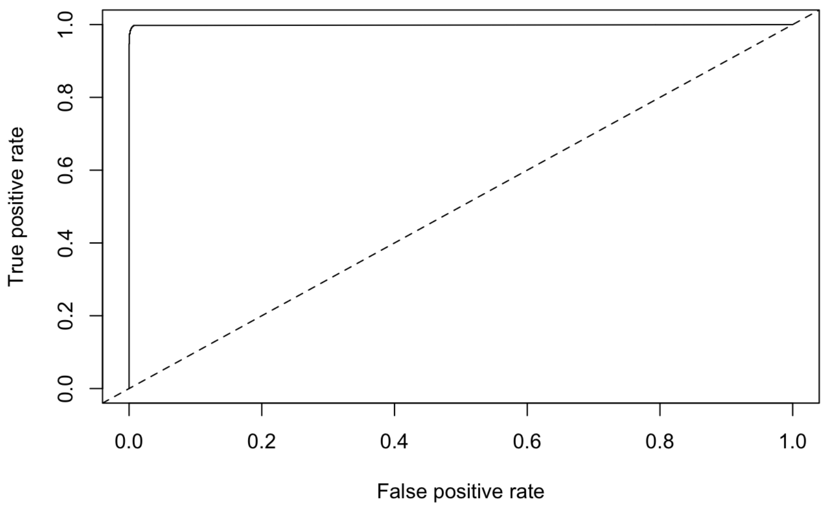


5. Evaluate the cross-validated (CV) predictive performance.

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




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

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

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

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

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

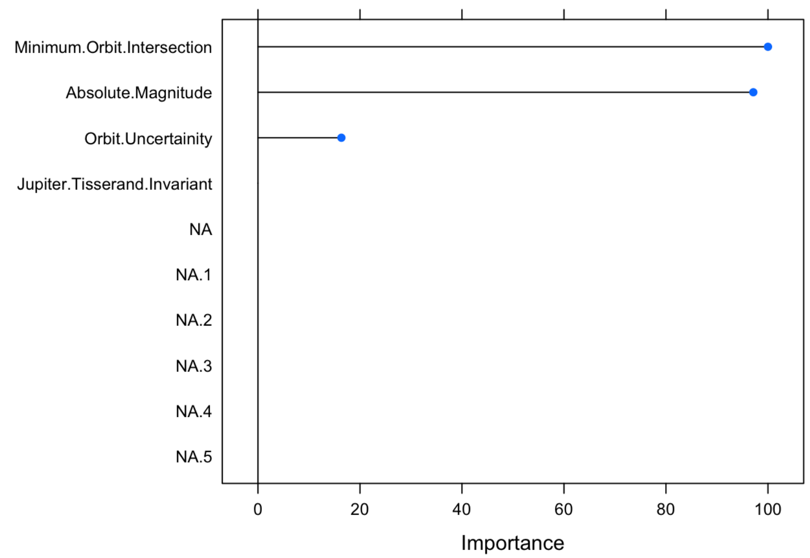


```
##           name       value
## 1     accuracy 0.996101365
## 2   error rate 0.003898635
## 3    precision 0.990697674
## 4  sensitivity 0.981566820
## 5  specificity 0.998487141
## 6    F-measure 0.986111111
## 7 Matthew's CC 0.983857862
```

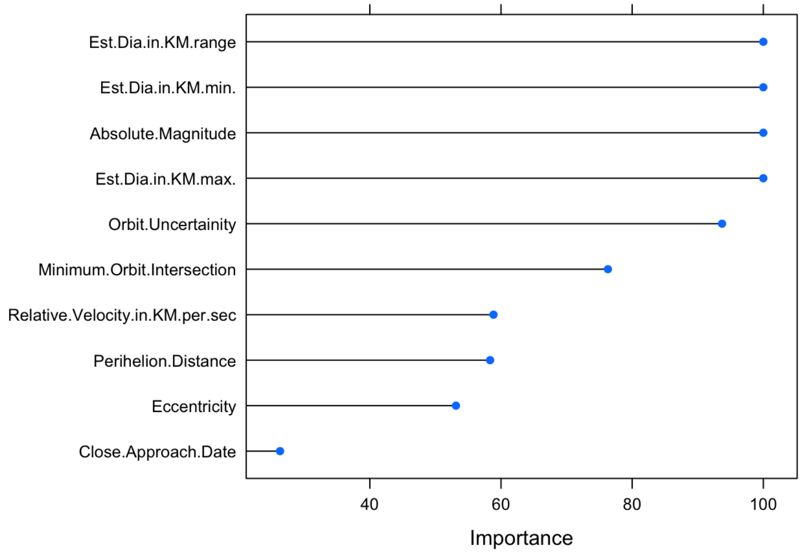
Model Comparisons

variable importance comparison

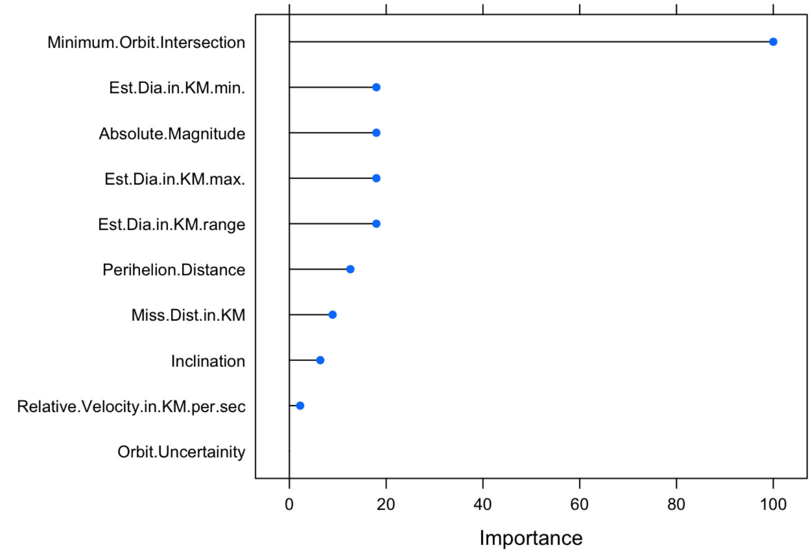
```
par(mfrow=c(2,2))
plot(varImp(fit.cv2), top=10)
```



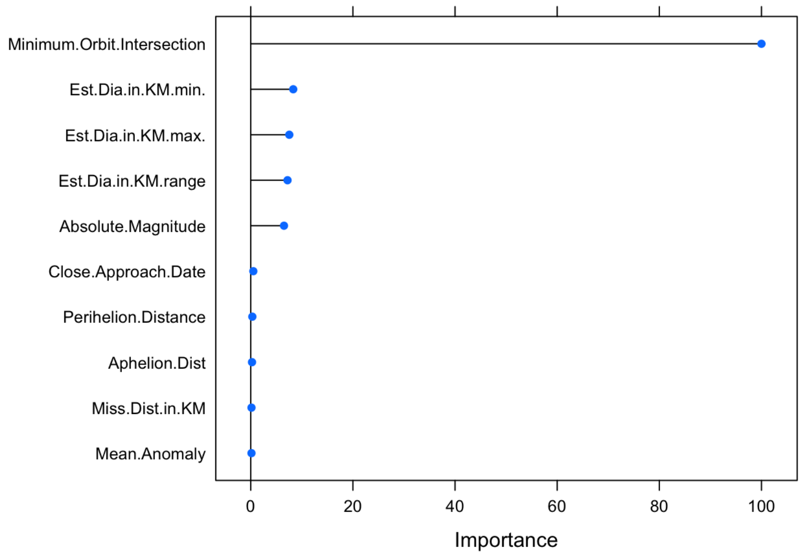

```
plot(varImp(fit.knn), top=10)
```




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


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



stats table

```
lasso.stats
```

```
##           name      value
## 1     accuracy 0.94964263
## 2   error rate 0.05035737
## 3    precision 0.78801843
## 4  sensitivity 0.84444444
## 5  specificity 0.96558174
## 6    F-measure 0.81525626
## 7 Matthew's CC 0.78677483
```

```
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
```

```
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
```

```
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
```

```
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 0.94964263 0.8814165         0.994801819   0.996101365
## 2   error rate 0.05035737 0.1185835         0.005198181   0.003898635
## 3    F-measure 0.81525626 0.4702467         0.981524249   0.986111111
## 4 Matthew's CC 0.78677483 0.4268670         0.978503227   0.983857862
## 5    precision 0.78801843 0.3732719         0.983796296   0.990697674
## 6  sensitivity 0.84444444 0.6352941         0.979262673   0.981566820
## 7  specificity 0.96558174 0.9036486         0.997352496   0.998487141
```

```
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"
```