# Asteroid Data Analysis Project

Annika Lin

2023-04-21

```
df <- read.csv("~/Documents/Georgetown/Spring23/Statistical Learning & Data Science/Proj
ect/NASA-asteroid-Classification-master/nasa_4_4_23.csv")
df <- df[ , !(names(df) %in% c("X"))]
```

# 1. Lasso-penalized Logistic Regression

## (1.a) Perform lasso variable selection using the area under the curve (AUC) for the receiver operating characteristic (ROC) curve as criterion for choosing the penalty parameter λ.

Use set.seed(1). List the variables selected using lambda.1se. [Note: Do not print out the coefficients of all covariates. Provide only the names of the selected variables.]

```
library(glmnet)
```

```
## Loading required package: Matrix
```

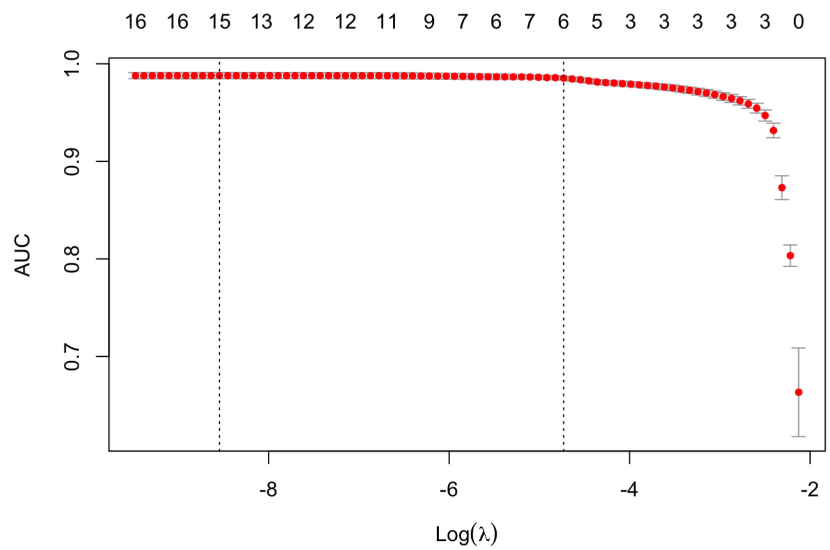
```
## Loaded glmnet 4.1-7
```

```
# we use the function model.matrix to create the design matrix
X = model.matrix(Hazardous ~ ., data=df)
Y = as.numeric(df$Hazardous=="True")

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








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

```
## [1] "Absolute.Magnitude"          "Est.Dia.in.KM.min."
## [3] "Orbit.Uncertainity"          "Minimum.Orbit.Intersection"
## [5] "Jupiter.Tisserand.Invariant" "Range.Dia.in.KM"
```

## (1.b) Fit a 5-fold cross-validated (CV) logistic regression model using the lasso-selected variables with set.seed(1).

```
library(caret)
```

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

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

```
  # paste(sel.names, collapse = "+")

set.seed(1)
fit.df <- train(Hazardous ~ Absolute.Magnitude+Est.Dia.in.KM.min.+Orbit.Uncertainity+Min
imum.Orbit.Intersection+Jupiter.Tisserand.Invariant+Range.Dia.in.KM, method = "glm",
  trControl = trainControl(method="cv", number=5, savePredictions = TRUE),
  data=df)
```

```
## 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.df
```

```
## Generalized Linear Model
##
## 3079 samples
##    6 predictor
##    2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2463, 2464, 2463, 2463, 2463
## Resampling results:
##
##   Accuracy   Kappa
##   0.9626534  0.8453003
```

**i. Assess if the final model has multicollinearity problems.**

```
summary(fit.df$finalModel)
```





```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3451  -0.0350  -0.0026   0.0000   6.6127
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  7.138e+01  5.218e+00  13.680  < 2e-16 ***
## Absolute.Magnitude          -3.027e+00  2.275e-01 -13.305  < 2e-16 ***
## Est.Dia.in.KM.min.          -5.929e+09  1.826e+09  -3.248  0.00116 **
## Orbit.Uncertainity          -1.332e-01  4.746e-02  -2.806  0.00502 **
## Minimum.Orbit.Intersection  -1.227e+02  8.280e+00 -14.815  < 2e-16 ***
## Jupiter.Tisserand.Invariant -1.264e-01  8.953e-02  -1.411  0.15815
## Range.Dia.in.KM              4.796e+09  1.477e+09   3.248  0.00116 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2508.03  on 3078  degrees of freedom
## Residual deviance:  546.34  on 3072  degrees of freedom
## AIC: 560.34
##
## Number of Fisher Scoring iterations: 15
```

There are no categorical covariates with more than two levels in the model so we use VIF.

```
library(car)
```

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

```
vif(fit.df$finalModel)
```

```
##          Absolute.Magnitude          Est.Dia.in.KM.min.
##                2.758298e+00                3.002775e+15
##          Orbit.Uncertainity  Minimum.Orbit.Intersection
##                1.208552e+00                2.045395e+00
## Jupiter.Tisserand.Invariant             Range.Dia.in.KM
##                1.128041e+00                3.002775e+15
```

There does not appear to be an issue of multicollinearity since VIF < 5 for all variables.

ii. Assess the goodness-of-fit of the final model.

```
library(ResourceSelection)
```

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

```
res = hoslem.test(fit.df$finalModel$y, fit.df$finalModel$fitted.values)
res
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  fit.df$finalModel$y, fit.df$finalModel$fitted.values
## X-squared = 265520, df = 8, p-value < 2.2e-16
```

Since there are continuous variables in this model, we use Hosmer-Lemeshow goodness-of-fit test. With a p-value < 2.2e-16, we reject H0. The model does not appear to fit the data well.

iii. Interpret the regression coefficient of the predictor with smallest p-value [Note: the intercept is not a predictor].

Absolute.Magnitude -3.027e+00 2.275e-01 -13.305 < 2e-16 ***Est.Dia.in.KM.min. -5.929e+09 1.826e+09 -3.248 0.00116*** *Orbit.Uncertainity -1.332e-01 4.746e-02 -2.806 0.00502* ***Minimum.Orbit.Intersection -1.227e+02 8.280e+00 -14.815 < 2e-16*** Range.Dia.in.KM 4.796e+09 1.477e+09 3.248 0.00116 **

Absolute.Magnitude and Minimum.Orbit.Intersection are significant at the .1% level.

0 ‘’ ***0.001*** ‘’ *0.01* ‘’ 0.05 ‘.’ 0.1 ‘ ’ 1

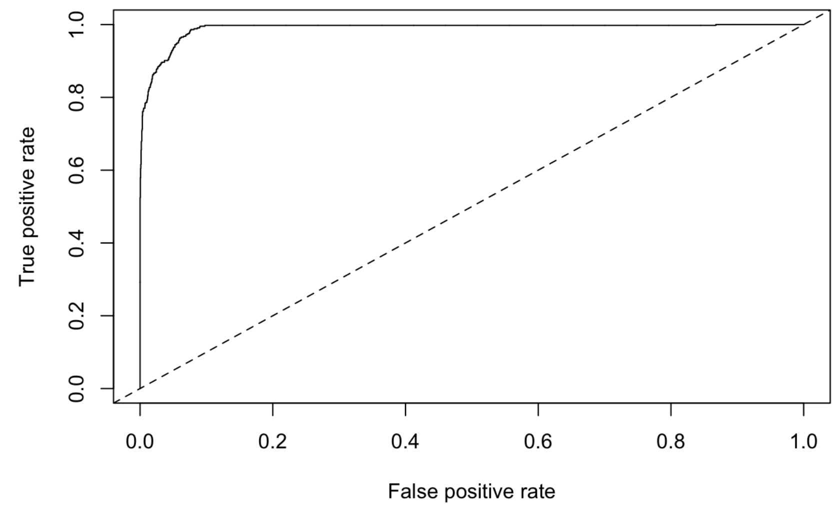
## (1.c) Provide the cross-validated ROC curve and its AUC [Note: you should use the cross-validated prediction to construct the ROC curve].

```
#
pihat <- predict(fit.df, type="prob")
# Using cutoff of pi_0=0.5
yhat <- pihat>0.5

library(ROCR)
# Plot ROC curve
pred = prediction(fitted(fit.df), df$Hazardous)
perf = performance(pred, "tpr", "fpr")
plot(perf)
abline(a=0, b=1, lty=2)
```








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

```
## [[1]]
## [1] 0.9894517
```

## (1.d) Let π0 be the cut-off for predicting hazard What range of π0 values lead to a true positive rate (TPR) > 0.75 and a false positive rate (FPR) < 0.25?

```
# pi0.cut <- cbind(unlist(perf@y.values),
# unlist(perf@x.values),
# unlist(perf@alpha.values))
# pi0.cut[pi0.cut[,1]>0.75 & pi0.cut[,2]<0.25,]
```

## (1.e) Using a cut-off of π0 = 0.35, calculate the cross-validated misHazardousification error rate and the Matthew correlation coefficient [Note: you should use the cross-validated prediction

to calculate these metrics].

```
# library(boot)
#
# mycost <- function(r, pi = 0) mean(abs(r-pi) > 0.35)
#
# set.seed(10)
# nrep <- 5
# cv.5foldRep <- sapply(1:nrep, function(i) {cv.err <- cv.glm(df, fit.df$finalModel, myc
ost, K=5)
# cv.err$delta[1]})
# cv.5foldRep
```

# 2. k-nearest neighbors (kNN)

## (2.a) Process the data using min-max normalization. Show the data for the first 5 covariates in the first

3 subjects before and after normalization.

```
library(caret)
# function to normalize data
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x))) }

df[1:3,1:5]
```

```
##   Absolute.Magnitude Est.Dia.in.KM.min. Est.Dia.in.KM.max. Close.Approach.Date
## 1               21.6          0.1272199          0.2844723            19950101
## 2               21.3          0.1460680          0.3266179            19950101
## 3               20.3          0.2315021          0.5176545            19950108
##   Relative.Velocity.km.per.sec
## 1                     6.115834
## 2                    18.113985
## 3                     7.590711
```

```
arr.norm <- apply(df[,-21], 2, normalize)
arr.norm[1:3,1:5]
```

```
##      Absolute.Magnitude Est.Dia.in.KM.min. Est.Dia.in.KM.max.
## [1,]          0.4067797         0.03602979         0.03602979
## [2,]          0.3898305         0.04141048         0.04141048
## [3,]          0.3333333         0.06579992         0.06579992
##      Close.Approach.Date Relative.Velocity.km.per.sec
## [1,]        0.000000e+00                    0.1236499
## [2,]        0.000000e+00                    0.4027430
## [3,]        3.320573e-05                    0.1579575
```





```
arr.norm <- data.frame(arr.norm, df$Hazardous)
# names(arr.norm) <- names(df[1:5])
colnames(arr.norm)[colnames(arr.norm) == "df.Hazardous"] ="Hazardous"
```

## (2.b) Fit kNN using 5-fold CV over a grid of values between 1 and 21 for the number of neighbors k,

using set.seed(1). How many 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 = T
RUE),
  metric = "Accuracy",
  data = arr.norm)

fit.knn
```

```
## k-Nearest Neighbors
##
## 3079 samples
##   20 predictor
##    2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2463, 2464, 2463, 2463, 2463
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##    1  0.8639193  0.4226974
##    2  0.8561203  0.3795259
##    3  0.8808067  0.4467633
##    4  0.8752846  0.4106403
##    5  0.8739895  0.3746612
##    6  0.8762612  0.3806889
##    7  0.8733402  0.3514732
##    8  0.8713879  0.3468392
##    9  0.8743105  0.3409216
##   10  0.8756103  0.3428819
##   11  0.8801579  0.3639023
##   12  0.8804825  0.3661709
##   13  0.8821064  0.3688452
##   14  0.8791812  0.3446355
##   15  0.8814566  0.3610334
##   16  0.8778846  0.3182547
##   17  0.8808072  0.3340695
##   18  0.8814571  0.3369320
##   19  0.8801531  0.3233616
##   20  0.8782050  0.3070310
##   21  0.8778804  0.2970209
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 13.
```

13 neighbors are used in the final model.

## (2.c) Which are the 10 most important variables using kNN? Is there any overlap with the variables you

selected using the lasso penalized logistic regression?

```
imp2 <- varImp(fit.knn)$importance
head(imp2[order(-imp2[,2]),,drop=FALSE], 10)
```

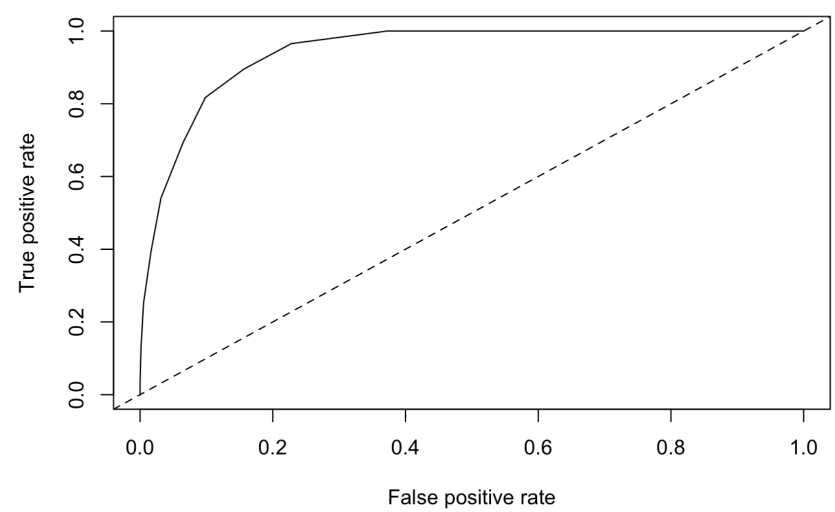




```
##                                 False      True
## Absolute.Magnitude           100.00000 100.00000
## Est.Dia.in.KM.min.           100.00000 100.00000
## Est.Dia.in.KM.max.           100.00000 100.00000
## Range.Dia.in.KM              100.00000 100.00000
## Orbit.Uncertainity            93.67117  93.67117
## Minimum.Orbit.Intersection    76.15651  76.15651
## Relative.Velocity.km.per.sec  58.97766  58.97766
## Perihelion.Distance           58.33724  58.33724
## Eccentricity                  53.31434  53.31434
## Close.Approach.Date           26.52959  26.52959
```

## (2.d) Provide the cross-validated ROC curve and its AUC.

```
pihatfin.knn <- predict(fit.knn, type="prob")
predfin.knn <- prediction(pihatfin.knn[,2], df$Hazardous)
perffin.knn <- performance(predfin.knn, "tpr", "fpr")
plot(perffin.knn)
abline(a=0, b=1, lty=2)
```



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

```
## [[1]]
## [1] 0.9431865
```

### (2.e) Let π0 be the cut-off for predicting the risk of collision What range of π0 values lead to a true positive rate (TPR) > 0.70 and a false positive rate (FPR) < 0.30?

```
# pi0.cut <- cbind(unlist(perffin.knn@y.values),
# unlist(perffin.knn@x.values),
# unlist(perffin.knn@alpha.values))
# pi0.cut[pi0.cut[,1]>0.7 & pi0.cut[,2]<0.3,]
```

### (2.f) What TPR and FPR is achieved using π0 = 0.5?

```
# pihat <- predict(fit.knn, type="prob")
# # Using cutoff of pi_0=0.5
# yhat <- pihat>0.5
# table(yhat, df$Hazardous[fit.knn$pred$rowIndex])
# pi0.cut[pi0.cut[,3]>0.45 & pi0.cut[,3]<0.55,]
```

# 3. Classification tree

### (3.a) Fit a decision tree with 5-fold CV using set.seed(1) and the one-SE rule. Plot the final Classification tree.

```
library(rpart)
set.seed(1)
arr.CVrpart <- train(Hazardous ~ ., data=df,
method="rpart",
tuneGrid = expand.grid(cp = seq(0.005, 0.05, length=10)),
trControl = trainControl(method = "cv", number=5,
savePredictions = TRUE,
selectionFunction = "oneSE") )
arr.CVrpart
```





```
## CART
##
## 3079 samples
##   20 predictor
##    2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2463, 2464, 2463, 2463, 2463
## Resampling results across tuning parameters:
##
##   cp     Accuracy   Kappa
##   0.005  0.9938296  0.9743396
##   0.010  0.9938296  0.9743396
##   0.015  0.9938296  0.9743396
##   0.020  0.9938296  0.9743396
##   0.025  0.9938296  0.9743396
##   0.030  0.9938296  0.9743396
##   0.035  0.9938296  0.9743396
##   0.040  0.9938296  0.9743396
##   0.045  0.9938296  0.9743396
##   0.050  0.9938296  0.9743396
##
## Accuracy was used to select the optimal model using  the one SE rule.
## The final value used for the model was cp = 0.05.
```

```
library(rattle)
```

```
## Loading required package: tibble
```

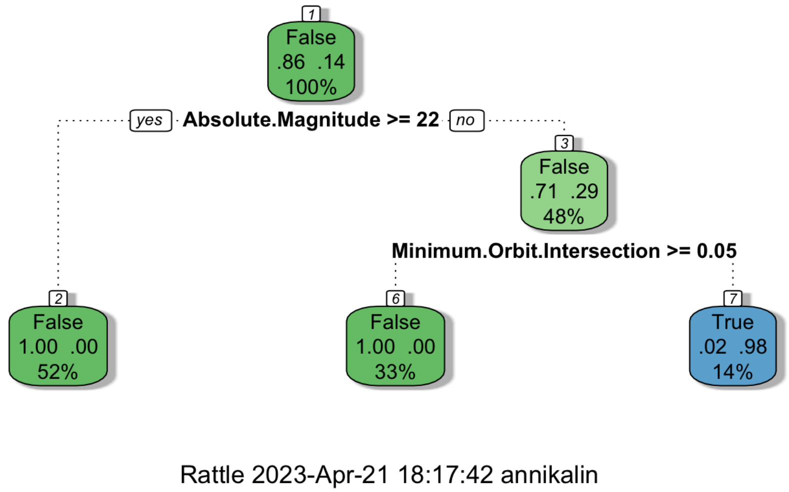
```
## Loading required package: bitops
```

```
##
## Attaching package: 'bitops'
```

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

```
## 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.
```

```
fancyRpartPlot(arr.CVrpart$finalModel)
```




## (3.b) Which are the 10 most important variables for the Hazardousification tree? Is there any overlap with the variables you selected using the lasso penalized logistic regression?

```
imp3 <- varImp(arr.CVrpart)$importance
head((imp3[order(-imp3$Overall),,drop=FALSE]), 10)
```

```
##                                Overall
## Minimum.Orbit.Intersection   100.000000
## Absolute.Magnitude            18.003784
## Est.Dia.in.KM.max.            18.003784
## Est.Dia.in.KM.min.            18.003784
## Range.Dia.in.KM               18.003784
## Perihelion.Distance           12.614216
## Miss.Dist..kilometers.         8.894239
## Inclination                    6.350764
## Relative.Velocity.km.per.sec   2.234861
## Close.Approach.Date            0.000000
```

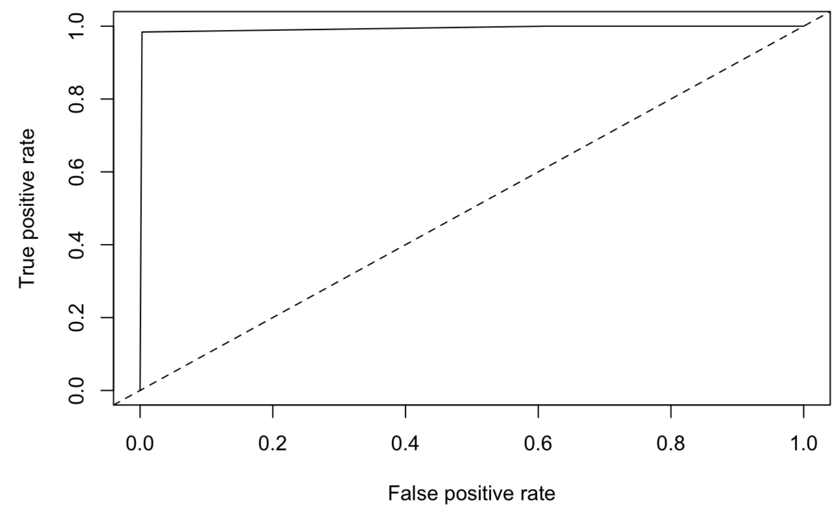




## (3.c) Provide the cross-validated ROC curve and its AUC.

```
pihat <- predict(arr.CVrpart, type="prob")

pred <- prediction(pihat[,2], df$Hazardous)
perf <- performance(pred, "tpr", "fpr")
plot(perf)
abline(a=0, b=1, lty=2)
```




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

```
## [[1]]
## [1] 0.9935725
```

## (3.d) Let π0 be the cut-off for predicting the risk of df. What range of π0 values lead to a true

positive rate (TPR) > 0.70 and a false positive rate (FPR) < 0.30?

```
# pi0.cut <- cbind(unlist(perf@y.values),
# unlist(perf@x.values),
# unlist(perf@alpha.values))
# pi0.cut[pi0.cut[,1]>0.7 & pi0.cut[,2]<0.3,]
```

# 4. Random forest

## (4.a) Fit a random forest with 5-fold CV using set.seed(1) and consider a range of values between 85 and 125 with steps of 10 for mtry, the number of randomly selected variables used at each node splitting. What value of mtry is used in the final model?

```
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
```

```
set.seed(1)
arr.RF <- train(Hazardous ~ .,
  method = "rf",
  tuneGrid = expand.grid(mtry=seq(85,125, 10)),
  trControl = trainControl(method="cv", number=5, savePredictions = TRUE, classProbs = T
RUE),
  metric = "Accuracy",
  data = df)
```





```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
```

```
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range
```

```
arr.RF
```

```
## Random Forest
##
## 3079 samples
##   20 predictor
##    2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2463, 2464, 2463, 2463, 2463
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    85   0.9961028  0.9838069
##    95   0.9954535  0.9810779
##   105   0.9954535  0.9810779
##   115   0.9948041  0.9783752
##   125   0.9951288  0.9797462
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 85.
```

The final value used for the model was mtry = 85.





## (4.b) Which are the 10 most important variables identified by random forest?

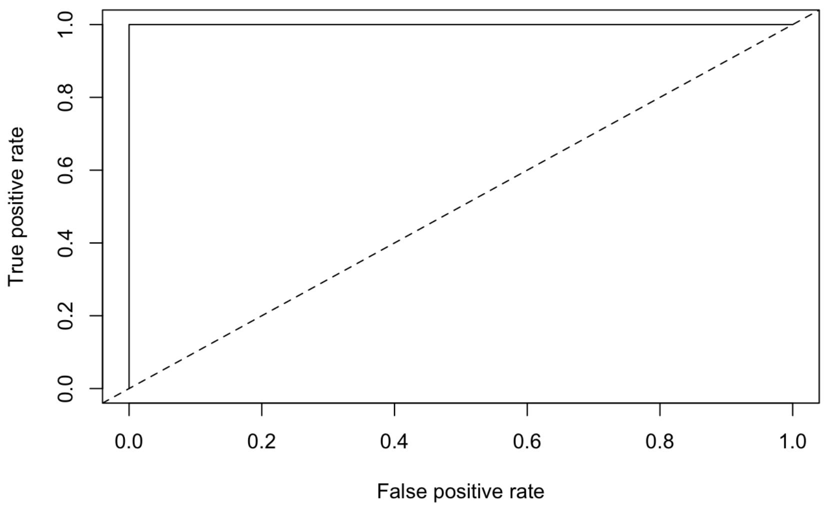
```
imp4 <- varImp(arr.RF)$importance
head((imp4[order(-imp4$Overall),,drop=FALSE]), 10)
```

```
##                                 Overall
## Minimum.Orbit.Intersection 100.0000000
## Est.Dia.in.KM.min.           7.8947189
## Est.Dia.in.KM.max.           7.5703971
## Absolute.Magnitude           7.3704056
## Range.Dia.in.KM              6.9641430
## Close.Approach.Date          0.4412355
## Perihelion.Distance          0.3386739
## Aphelion.Dist                0.2417534
## Miss.Dist..kilometers.       0.1887247
## Mean.Anomaly                 0.1550934
```

## (4.c) Provide the cross-validated ROC curve and its AUC.

```
pihat <- predict(arr.RF, type="prob")

pred <- prediction(pihat[,2], df$Hazardous)
perf <- performance(pred, "tpr", "fpr")
plot(perf)
abline(a=0, b=1, lty=2)
```




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

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

## (4.d) Let π0 be the cut-off for predicting the risk of df. What range of π0 values lead to a true positive rate (TPR) > 0.75 and a false positive rate (FPR) < 0.25?

```
# pi0.cut <- cbind(unlist(perf@y.values),
# unlist(perf@x.values),
# unlist(perf@alpha.values))
# pi0.cut[pi0.cut[,1]>0.75 & pi0.cut[,2]<0.25,]
```

# Comparison of models





## Provide a table summarizing the AUC for the cross-validated ROC curve for each of the methods

considered (penalized logistic, PC logistic, kNN, PC kNN, CART, random forest).

```
Method <- c("penalized logistic", "kNN", "CART", "random forest")
auc_score <- c(unlist(pen_log_auc),  unlist(knn_auc), unlist(cart_auc), unlist(rf_auc))

auc_tab <- data.frame(Method, auc_score)
# auc_tab

auc_tab[order(-auc_tab$auc_score),,drop=FALSE]
```

```
##               Method auc_score
## 4      random forest 1.0000000
## 3               CART 0.9935725
## 1 penalized logistic 0.9894517
## 2                kNN 0.9431865
```

## Which variables are deemed important by the four methods using the covariate data (penalized logistic, kNN, CART, random forest)?

```
imp1 <- varImp(fit.df)$importance



mylist <- list(rownames(head((imp1[order(-imp1$Overall),,drop=FALSE]), 10)),
               rownames(head(imp2[order(-imp2[,2]),,drop=FALSE], 10)),
               rownames(head((imp3[order(-imp3$Overall),,drop=FALSE]), 10)),
               rownames(head((imp4[order(-imp4$Overall),,drop=FALSE]), 10))
               )


mydf <- stack(setNames(mylist, seq_along(mylist)))
mydf$ind <- as.numeric(mydf$ind)


names(mydf) <- c("Variables", "count.in.methods")

mydf[order(-mydf$count.in.methods),,drop=FALSE]
```

```
##                       Variables count.in.methods
## 27   Minimum.Orbit.Intersection                4
## 28           Est.Dia.in.KM.min.                4
## 29           Est.Dia.in.KM.max.                4
## 30           Absolute.Magnitude                4
## 31              Range.Dia.in.KM                4
## 32           Close.Approach.Date               4
## 33          Perihelion.Distance                4
## 34                 Aphelion.Dist               4
## 35        Miss.Dist..kilometers.               4
## 36                 Mean.Anomaly                4
## 17   Minimum.Orbit.Intersection                3
## 18           Absolute.Magnitude                3
## 19           Est.Dia.in.KM.max.                3
## 20           Est.Dia.in.KM.min.                3
## 21              Range.Dia.in.KM                3
## 22          Perihelion.Distance                3
## 23        Miss.Dist..kilometers.               3
## 24                  Inclination                3
## 25 Relative.Velocity.km.per.sec                3
## 26           Close.Approach.Date               3
## 7            Absolute.Magnitude                2
## 8            Est.Dia.in.KM.min.                2
## 9            Est.Dia.in.KM.max.                2
## 10              Range.Dia.in.KM                2
## 11           Orbit.Uncertainity                2
## 12   Minimum.Orbit.Intersection                2
## 13 Relative.Velocity.km.per.sec                2
## 14          Perihelion.Distance                2
## 15                 Eccentricity                2
## 16           Close.Approach.Date               2
## 1    Minimum.Orbit.Intersection                1
## 2            Absolute.Magnitude                1
## 3            Est.Dia.in.KM.min.                1
## 4               Range.Dia.in.KM                1
## 5            Orbit.Uncertainity                1
## 6   Jupiter.Tisserand.Invariant                1
```