Caret / Recursive Partitioning

Dennis Raj

May 16, 2017

Exercise 1: caret/logistic regression (5 points)

Rebuild your logistic regression model from the previous week, this time using the `caret` package.

- Calculate the training or apparent performance of the model.
- Calculate an unbiased measure of performance
- Create a ROC Curve for your model

Show all work.

```
# create the data set used for the analysis and set various features to factors
flights <- tbl_df(read_csv("flights_clean.csv", col_names = TRUE))
```

```
## Parsed with column specification:
## cols(
##   .default = col_integer(),
##   carrier = col_character(),
##   tailnum = col_character(),
##   origin = col_character(),
##   dest = col_character(),
##   time_hour = col_character(),
##   fdate = col_character(),
##   type = col_character(),
##   manufacturer = col_character(),
##   model = col_character(),
##   engine = col_character(),
##   name = col_character(),
##   lat = col_double(),
##   lon = col_double(),
##   dst = col_character(),
##   temp = col_double(),
##   dewp = col_double(),
##   humid = col_double(),
##   wind_speed = col_double(),
##   wind_gust = col_double(),
##   pressure = col_double()
##   # ... with 1 more columns
## )
```

```
## See spec(...) for full column specifications.
```

```
flights$month.x <- as.factor(flights$month.x)
flights$carrier <- as.factor(flights$carrier)
flights$origin <- as.factor(flights$origin)
flights$dest <- as.factor(flights$dest)
flights$late <- as.factor(flights$late)



#narrow down the data set to features of interest
```

```r
flights_clean <- flights %>% select(late, month.x, dep_delay, carrier, origin, dest, distance, humid, wi
glimpse(flights_clean)
```

```
## Observations: 327,346
## Variables: 10
## $ late       <fctr> FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, FALSE, FA...
## $ month.x    <fctr> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ dep_delay  <int> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2...
## $ carrier    <fctr> UA, UA, AA, B6, DL, UA, B6, EV, B6, AA, B6, B6, UA...
## $ origin     <fctr> EWR, LGA, JFK, JFK, LGA, EWR, EWR, LGA, JFK, LGA, ...
## $ dest       <fctr> IAH, IAH, MIA, BQN, ATL, ORD, FLL, IAD, MCO, ORD, ...
## $ distance   <int> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 7...
## $ humid      <dbl> 62.21, 57.33, 59.50, 59.50, 57.33, 62.21, 62.21, 57...
## $ wind_speed <dbl> 10.3570, 14.9601, 17.2617, 17.2617, 18.4125, 10.357...
## $ temp       <dbl> 39.92, 39.92, 39.92, 39.92, 39.92, 39.92, 39.92, 39...
```

```r
#set the seed for reproducability with caret's inTraining
set.seed(1234)

# set the level for the training and the test data sets on a 75 / 25 split
inTraining <- createDataPartition(flights_clean$late, p = .75, list = FALSE)
train <- flights_clean[inTraining,]
test <- flights_clean[-inTraining,]
# setup the cross validation for the fite
tc <- trainControl(method = "cv", 10, savePredictions=T)
# create the fit
fit1 <- train(late ~ month.x + dep_delay + carrier + origin + dest, data = train,
                        method = "glm",
                        family = binomial,
                        trControl = tc)
```

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

```r
#double check that the predictions were made with different folds
summary(fit1)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2213  -0.3361  -0.2520  -0.1745   3.3433
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.316e+00  2.841e-01 -11.671  < 2e-16 ***
## month.x2    -1.489e-01  4.184e-02  -3.560 0.000370 ***
## month.x3    -3.213e-01  4.095e-02  -7.845 4.33e-15 ***
## month.x4     2.722e-01  3.872e-02   7.029 2.08e-12 ***
## month.x5    -4.674e-01  4.144e-02 -11.278  < 2e-16 ***
## month.x6     5.908e-02  3.930e-02   1.503 0.132804
## month.x7     1.945e-02  3.888e-02   0.500 0.616846
## month.x8    -1.068e-01  3.935e-02  -2.713 0.006659 **
## month.x9    -7.530e-01  4.658e-02 -16.167  < 2e-16 ***
## month.x10   -3.875e-01  4.244e-02  -9.130  < 2e-16 ***
## month.x11   -1.614e-01  4.151e-02  -3.889 0.000101 ***
## month.x12    3.437e-01  3.721e-02   9.236  < 2e-16 ***
## dep_delay    1.060e-01  5.165e-04 205.234  < 2e-16 ***
## carrierAA    1.261e-01  6.016e-02   2.097 0.036009 *
## carrierAS   -4.407e-01  2.557e-01  -1.723 0.084804 .
## carrierB6    4.583e-01  5.142e-02   8.912  < 2e-16 ***
## carrierDL    2.971e-02  5.325e-02   0.558 0.576883
## carrierEV    3.854e-01  5.348e-02   7.206 5.77e-13 ***
## carrierF9    8.280e-01  1.587e-01   5.216 1.82e-07 ***
## carrierFL    5.161e-01  1.004e-01   5.141 2.73e-07 ***
## carrierHA    2.443e-01  4.314e-01   0.566 0.571187
## carrierMQ    7.812e-01  5.348e-02  14.607  < 2e-16 ***
## carrierOO    1.555e+00  8.430e-01   1.845 0.065099 .
## carrierUA   -4.702e-02  5.851e-02  -0.804 0.421606
## carrierUS    5.190e-01  6.481e-02   8.008 1.16e-15 ***
## carrierVX   -2.535e-01  9.182e-02  -2.761 0.005766 **
## carrierWN   -3.614e-01  7.836e-02  -4.612 3.98e-06 ***
## carrierYV    4.917e-01  2.056e-01   2.391 0.016791 *
## originJFK   -5.734e-03  3.123e-02  -0.184 0.854308
## originLGA    1.143e-01  2.743e-02   4.167 3.09e-05 ***
## destACK      4.253e-02  4.104e-01   0.104 0.917455
## destALB     -5.564e-01  3.749e-01  -1.484 0.137826
## destANC     -1.808e+00  2.851e+00  -0.634 0.526079
## destATL      4.929e-01  2.804e-01   1.758 0.078752 .
## destAUS      4.905e-01  2.910e-01   1.686 0.091868 .
## destAVL      4.718e-01  3.990e-01   1.182 0.237070
## destBDL     -9.490e-01  4.010e-01  -2.366 0.017962 *
## destBGR     -6.337e-01  3.913e-01  -1.619 0.105350
## destBHM     -4.219e-01  4.200e-01  -1.005 0.315096
## destBNA      1.960e-01  2.844e-01   0.689 0.490768
## destBOS     -2.304e-02  2.800e-01  -0.082 0.934410
## destBQN     -2.920e-02  3.169e-01  -0.092 0.926597
## destBTV     -1.643e-01  2.941e-01  -0.559 0.576397
## destBUF     -3.677e-02  2.861e-01  -0.129 0.897715
## destBUR      6.414e-01  3.399e-01   1.887 0.059207 .
## destBWI      3.697e-01  3.020e-01   1.224 0.220842
## destBZN      8.264e-01  7.115e-01   1.161 0.245446
## destCAE      1.851e+00  4.426e-01   4.183 2.88e-05 ***
## destCAK      2.068e-01  3.277e-01   0.631 0.528105
## destCHO     -8.090e-01  9.117e-01  -0.887 0.374889
## destCHS      1.379e-01  2.923e-01   0.472 0.637137
## destCLE      6.532e-02  2.874e-01   0.227 0.820229
## destCLT      3.260e-01  2.814e-01   1.158 0.246752
## destCMH     -5.262e-02  2.900e-01  -0.181 0.856027
## destCRW      2.136e-01  4.590e-01   0.465 0.641704
## destCVG      3.096e-01  2.891e-01   1.071 0.284237
## destDAY     -1.704e-02  3.091e-01  -0.055 0.956031
## destDCA      3.025e-01  2.827e-01   1.070 0.284514
## destDEN      5.095e-01  2.826e-01   1.803 0.071389 .
## destDFW      4.018e-01  2.842e-01   1.414 0.157411
## destDSM      2.384e-01  3.543e-01   0.673 0.500964
## destDTW     -2.805e-03  2.829e-01  -0.010 0.992091
## destEGE     -1.153e-01  4.437e-01  -0.260 0.795044
## destEYW      1.444e-01  1.134e+00   0.127 0.898678
## destFLL      4.413e-01  2.791e-01   1.581 0.113821
## destGRR      5.684e-01  3.207e-01   1.773 0.076300 .
## destGSO      1.518e-01  3.036e-01   0.500 0.617114
## destGSP      4.301e-01  3.179e-01   1.353 0.176100
## destHDN     -7.511e-01  1.482e+00  -0.507 0.612245
## destHNL      2.551e-03  3.760e-01   0.007 0.994586
## destHOU      5.129e-01  2.949e-01   1.739 0.082002 .
## destIAD      7.566e-02  2.862e-01   0.264 0.791505
## destIAH      5.375e-01  2.828e-01   1.900 0.057379 .
## destILM     -1.009e+00  6.187e-01  -1.631 0.102861
## destIND      1.471e-01  2.981e-01   0.494 0.621632
## destJAC      2.910e-01  7.997e-01   0.364 0.715985
## destJAX      2.244e-01  2.909e-01   0.771 0.440430
## destLAS      2.315e-01  2.838e-01   0.816 0.414657
## destLAX      4.606e-01  2.794e-01   1.648 0.099253 .
## destLEX     -7.249e+00  1.970e+02  -0.037 0.970643
## destLGB     -1.462e-01  3.273e-01  -0.447 0.655059
## destMCI      3.242e-01  2.978e-01   1.089 0.276319
## destMCO      1.853e-01  2.791e-01   0.664 0.506840
## destMDW      3.458e-01  2.936e-01   1.178 0.238948
## destMEM      1.995e-01  3.007e-01   0.663 0.507146
## destMHT     -4.181e-01  3.232e-01  -1.293 0.195869
## destMIA      1.394e-01  2.820e-01   0.494 0.621003
## destMKE      4.001e-01  2.921e-01   1.370 0.170746
## destMSN     -4.616e-02  3.450e-01  -0.134 0.893572
## destMSP      1.054e-01  2.840e-01   0.371 0.710611
## destMSY      9.436e-02  2.885e-01   0.327 0.743574
## destMTJ      5.902e-01  1.101e+00   0.536 0.591839
## destMVY      2.523e-01  4.286e-01   0.589 0.556069
## destMYR      6.161e-02  6.759e-01   0.091 0.927362
## destOAK     -5.413e-02  3.743e-01  -0.145 0.885002
## destOKC      1.110e+00  3.501e-01   3.170 0.001526 **
## destOMA      3.921e-01  3.204e-01   1.224 0.221038
## destORD      2.070e-01  2.799e-01   0.739 0.459678
## destORF     -1.340e-01  3.070e-01  -0.436 0.662566
## destPBI      4.149e-01  2.814e-01   1.474 0.140361
## destPDX      3.106e-01  3.009e-01   1.032 0.301941
## destPHL      8.717e-01  2.987e-01   2.918 0.003522 **
## destPHX      8.862e-02  2.864e-01   0.309 0.756984
## destPIT      1.699e-01  2.930e-01   0.580 0.562045
## destPSE     -1.658e-01  3.752e-01  -0.442 0.658656
## destPSP     -8.057e+00  5.242e+01  -0.154 0.877843
## destPVD      9.064e-02  3.685e-01   0.246 0.805692
## destPWM     -2.016e-01  2.948e-01  -0.684 0.493993
## destRDU      8.234e-02  2.829e-01   0.291 0.771043
## destRIC      3.296e-01  2.946e-01   1.119 0.263136
## destROC      4.771e-02  2.943e-01   0.162 0.871191
## destRSW      4.656e-02  2.881e-01   0.162 0.871625
## destSAN      5.521e-01  2.883e-01   1.915 0.055534 .
## destSAT      2.262e-01  3.303e-01   0.685 0.493429
## destSAV      2.122e-01  3.274e-01   0.648 0.516763
## destSBN     -1.955e+00  1.712e+00  -1.142 0.253373
## destSDF      2.493e-01  3.154e-01   0.790 0.429383
## destSEA      2.586e-01  2.886e-01   0.896 0.370227
## destSFO      4.127e-01  2.801e-01   1.473 0.140665
## destSJC      2.141e-01  3.619e-01   0.592 0.554092
## destSJU      1.508e-01  2.828e-01   0.533 0.593774
## destSLC      3.443e-02  2.950e-01   0.117 0.907081
## destSMF      5.192e-01  3.614e-01   1.437 0.150803
## destSNA      5.177e-02  3.316e-01   0.156 0.875936
## destSRQ      3.939e-02  3.125e-01   0.126 0.899705
## destSTL      2.700e-01  2.878e-01   0.938 0.348052
## destSTT      9.956e-02  3.674e-01   0.271 0.786422
## destSYR     -2.681e-01  3.014e-01  -0.890 0.373732
## destTPA      3.572e-01  2.813e-01   1.270 0.204203
## destTUL      7.573e-01  3.744e-01   2.023 0.043090 *
## destTVC     -3.385e-01  5.841e-01  -0.579 0.562258
## destTYS      2.898e-01  3.471e-01   0.835 0.403734
## destXNA      8.630e-01  3.055e-01   2.825 0.004734 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 244645  on 245510  degrees of freedom
## Residual deviance: 107108  on 245378  degrees of freedom
## AIC: 107374
##
## Number of Fisher Scoring iterations: 10
```

```r
head(fit1$pred)
```

```
##    pred   obs rowIndex parameter Resample
## 1 FALSE FALSE        3      none   Fold01
## 2 FALSE FALSE       50      none   Fold01
## 3  TRUE  TRUE       65      none   Fold01
## 4 FALSE FALSE       68      none   Fold01
## 5 FALSE  TRUE       71      none   Fold01
## 6 FALSE FALSE       72      none   Fold01
```

```r
tail(fit1$pred)
```

```
##         pred   obs rowIndex parameter Resample
## 245506  TRUE  TRUE   245437      none   Fold10
## 245507 FALSE FALSE   245480      none   Fold10
## 245508 FALSE FALSE   245483      none   Fold10
## 245509 FALSE FALSE   245490      none   Fold10
## 245510  TRUE  TRUE   245505      none   Fold10
## 245511 FALSE FALSE   245507      none   Fold10
```

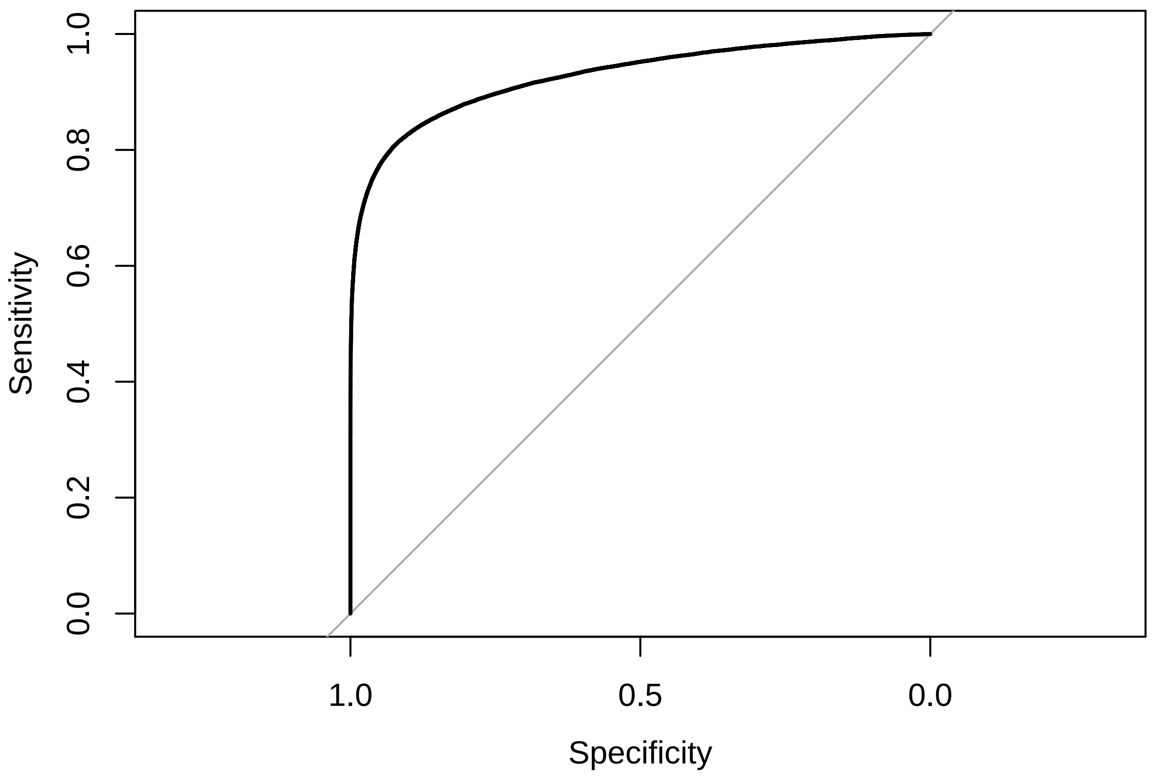
```r
# use the final fitted values for the prediction and set a threshold level for 'late'
fitpred <- ifelse(fit1$finalModel$fitted.values > 0.5, TRUE, FALSE)
fitpred <- as.factor(fitpred)

# setup a confusion matrix
fit1_cm <- confusionMatrix(data=fitpred, reference = train$late)
# review the confusion matrix
fit1_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  FALSE   TRUE
##      FALSE 192395  14395
##      TRUE    4396  34325
##
##                Accuracy : 0.9235
##                  95% CI : (0.9224, 0.9245)
##     No Information Rate : 0.8016
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7393
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9777
##             Specificity : 0.7045
##          Pos Pred Value : 0.9304
##          Neg Pred Value : 0.8865
##              Prevalence : 0.8016
##          Detection Rate : 0.7837
##    Detection Prevalence : 0.8423
##       Balanced Accuracy : 0.8411
##
##        'Positive' Class : FALSE
##
```

```
# the kappa of 0.7393 is an unbiased (or less biased) measure of accuracy

# calculate and plot an ROC curve
fit1_auc <- roc(train$late, fit1$finalModel$fitted.values)
# the area under the curve of 0.9272 is another unbiased measure of accuracy
fit1_auc
```

```
##
## Call:
## roc.default(response = train$late, predictor = fit1$finalModel$fitted.values)
##
## Data: fit1$finalModel$fitted.values in 196791 controls (train$late FALSE) < 48720 cases (train$late
## Area under the curve: 0.9272
```

```
plot(fit1_auc)
```



Exercise 2: caret/rpart (5 points)

Using the `caret` and `rpart` packages, create a **classification** model for flight delays using your NYC FLight data. Your solution should include:

- The use of `caret` and `rpart` to train a model.
- An articulation of the the problem your are
- An naive model
- An unbiased calculation of the performance metric
- A plot of your model – (the actual tree; there are several ways to do this)
- A discussion of your model

Show and describe all work

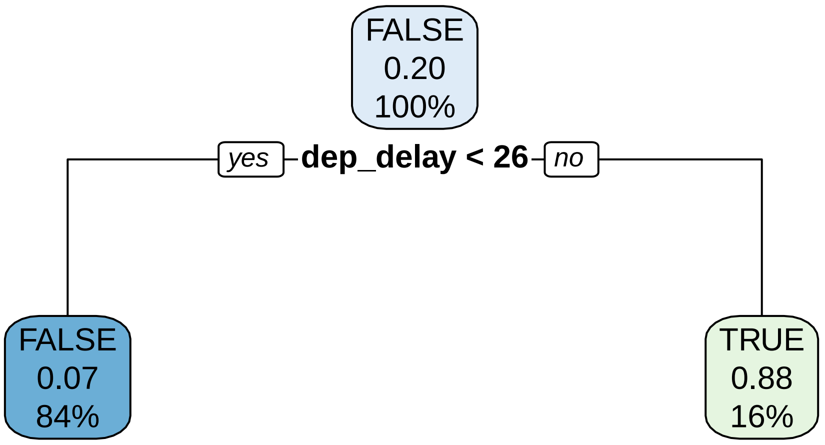
```
# Articulation of the problem
# We are looking to find a classification model that can accurately predict whether flights are late ar

# The naive model would be extending the rate of the observed values in the training data to the test d

set.seed(2345)

inTraining2 <- createDataPartition(flights_clean$late, p = .75, list = FALSE)
train2 <- flights_clean[inTraining2,]
test2 <- flights_clean[-inTraining2,]
#tc2 <- trainControl(method = "repeatedcv",
                   # number = 10,
                   # classProbs = TRUE,
                  #  summaryFunction = twoClassSummary)
```

```
#fit2 <- train(cat_late ~ month.x + dep_delay + carrier + origin + dest,
         #      data = train2,
         #     method = "rpart",
         #     trControl = tc2)

#fit2
#rpart.plot(fit2)

#fitpred2 <- predict(fit2, newdata=test2, type="class")

#fit2_cm <- confusionMatrix(data=fitpred2, reference = test2$late)
#fit2_cm
```

```
#This pat was created because I couldn't get the caret resampling to work with the rpart method to crea

fit3 <- rpart(late ~ month.x + dep_delay + carrier + origin + dest,
               data = train2)

fit3
```

```
## n= 245511
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 245511 48720 FALSE (0.80155675 0.19844325)
##   2) dep_delay< 26.5 206137 14252 FALSE (0.93086151 0.06913849) *
##   3) dep_delay>=26.5 39374  4906 TRUE (0.12459999 0.87540001) *
```

```
rpart.plot(fit3)
```



```
# The model is pretty straightforward -- it uses just the departure delay to determine what the outcome
fitpred3 <- predict(fit3, newdata=test2, type="class")

fit3_cm <- confusionMatrix(data=fitpred3, reference = test2$late)
# the Kappa of 0.7363 is an unbiased metric (based only on rpart), which is pretty good.
fit3_cm
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction FALSE  TRUE
##      FALSE 63888  4684
##      TRUE   1708 11555
##
##                Accuracy : 0.9219
##                  95% CI : (0.92, 0.9237)
##     No Information Rate : 0.8016
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7363
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9740
##             Specificity : 0.7116
##          Pos Pred Value : 0.9317
##          Neg Pred Value : 0.8712
##              Prevalence : 0.8016
##          Detection Rate : 0.7807
##    Detection Prevalence : 0.8379
##       Balanced Accuracy : 0.8428
##
##        'Positive' Class : FALSE
##
```

Questions:

- Discuss the difference between the models and why you would use one model over the other?

It depends on what my audience was and who was going to be using the model. Since rules-based / decision tree models are both highly interpretable and easy to use to predict new data, if this was a model that was going to be used by non-programmer/analytics people I would prefer to use this model. However, I think there are better ways to tweak the logsitic regression model to get a prefered business outcome by changing the specificy and sensitivy (aka moving along the ROC curve) to limit Type I and Type II errors, depending on their cost.

- How might you produce an ROC type curve for the *rpart* model?

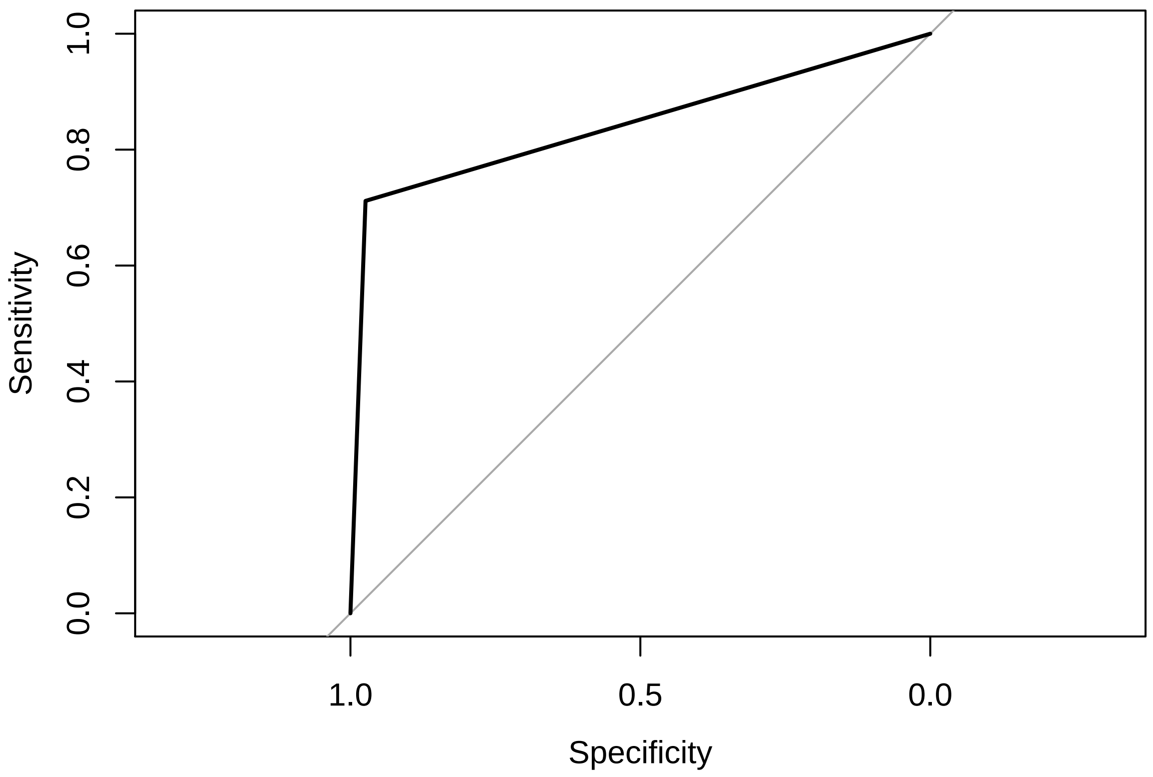
```
# calculate and plot an ROC curve for the rpart model

pred_test <- predict(fit3, newdata=test2, type="prob")[,2]

str(pred_test)
```

```
##  Named num [1:81835] 0.0691 0.0691 0.0691 0.0691 0.0691 ...
##  - attr(*, "names")= chr [1:81835] "1" "2" "3" "4" ...
```

```
fit3_auc <- roc(test2$late, pred_test)
fit3_auc
```

```
##
## Call:
## roc.default(response = test2$late, predictor = pred_test)
##
## Data: pred_test in 65596 controls (test2$late FALSE) < 16239 cases (test2$late TRUE).
## Area under the curve: 0.8428
```

```
plot(fit3_auc)
```


Sensitivity
Specificity