Caret / Recursive Partitioning

Jennifer Lee

May 13, 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.

```
#Read the joined NYC flights data set with added "delay" column
nycflights2 <- read.csv("nycflights2.csv")

#Task 1. Calculate training or apparent performance of the model.
# create a stratified random sample of the data into training and test sets using caret
set.seed(1000)
inTraining <- createDataPartition(nycflights2$delay22min, p = .75, list = FALSE)
training <- nycflights2[ inTraining,]
testing <- nycflights2[-inTraining,]

#build the logistic regression model
glmMod <- glm(formula = delay22min ~ month.x + dep_time + dep_delay + arr_time + air_time + distance + 
```

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

```
summary(glmMod)
```

```
##
## Call:
## glm(formula = delay22min ~ month.x + dep_time + dep_delay + arr_time +
##     air_time + distance + hour.x + temp + dewp + humid, family = binomial,
##     data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2459  -0.2946  -0.1784  -0.0889   3.9423
##
## Coefficients:
##               Estimate Std. Error  z value Pr(>|z|)
## (Intercept) -1.017e+01  2.073e-01  -49.061  < 2e-16 ***
## month.x     -3.499e-03  2.574e-03   -1.359    0.174
## dep_time    -7.916e-04  8.307e-05   -9.531  < 2e-16 ***
## dep_delay    1.229e-01  6.459e-04  190.215  < 2e-16 ***
## arr_time    -9.385e-05  2.320e-05   -4.046 5.21e-05 ***
## air_time     9.272e-02  7.966e-04  116.386  < 2e-16 ***
```

```
## distance    -1.200e-02  1.045e-04 -114.821  < 2e-16 ***
## hour.x       1.043e-01  8.257e-03   12.626  < 2e-16 ***
## temp         9.879e-02  4.766e-03   20.728  < 2e-16 ***
## dewp        -8.424e-02  5.054e-03  -16.668  < 2e-16 ***
## humid        5.222e-02  2.316e-03   22.545  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 242873  on 243878  degrees of freedom
## Residual deviance:  89994  on 243868  degrees of freedom
##   (1632 observations deleted due to missingness)
## AIC: 90016
##
## Number of Fisher Scoring iterations: 7
```

```
#prediction
predglm <- predict(glmMod, newdata=testing, type = "response")

#label predictions as same level as last column in NYCflights2 data set and change class to factor
model_predglm <- rep("delay < 22min", 81835)
model_predglm[predglm > 0.5] <- "delay >= 22 min"
model_predglm <- as.factor(model_predglm)

#Task 2. Calculate Unbiased Measure of Performance
#create a confusion matrix using the caret package
confusionMatrix(data=model_predglm, testing$delay22min)
```

```
## Confusion Matrix and Statistics
##
##                  Reference
## Prediction        delay < 22min delay >= 22 min
##   delay < 22min           64169            4137
##   delay >= 22 min          1427           12102
##
##                Accuracy : 0.932
##                  95% CI : (0.9303, 0.9337)
##     No Information Rate : 0.8016
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.772
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9782
##             Specificity : 0.7452
##          Pos Pred Value : 0.9394
##          Neg Pred Value : 0.8945
##              Prevalence : 0.8016
##          Detection Rate : 0.7841
##    Detection Prevalence : 0.8347
##       Balanced Accuracy : 0.8617
##
##        'Positive' Class : delay < 22min
```

```
##
```

```r
#Use lift function to evaluate probabilities thresholds that can capture a certain percentage of hits b
set.seed(2)
lift_training <- twoClassSim(1000)
lift_testing <- twoClassSim(1000)
ctrl <- trainControl(method = "cv", classProbs = TRUE, summaryFunction = twoClassSummary)

set.seed(1045)
fda_lift <- train(Class ~ ., data = lift_training, method = "fda",
                  metric = "ROC", tuneLength = 20, trControl = ctrl)
```

```
## Loading required package: earth

## Warning: package 'earth' was built under R version 3.3.2

## Loading required package: plotmo

## Warning: package 'plotmo' was built under R version 3.3.2

## Loading required package: plotrix

## Warning: package 'plotrix' was built under R version 3.3.2

##
## Attaching package: 'plotrix'

## The following object is masked from 'package:gplots':
##
##     plotCI

## Loading required package: TeachingDemos

## Loading required package: mda

## Loading required package: class

## Loaded mda 0.4-9
```

```r
set.seed(1045)
lda_lift <- train(Class ~ ., data = lift_training, method = "lda",
                  metric = "ROC", trControl = ctrl)
```

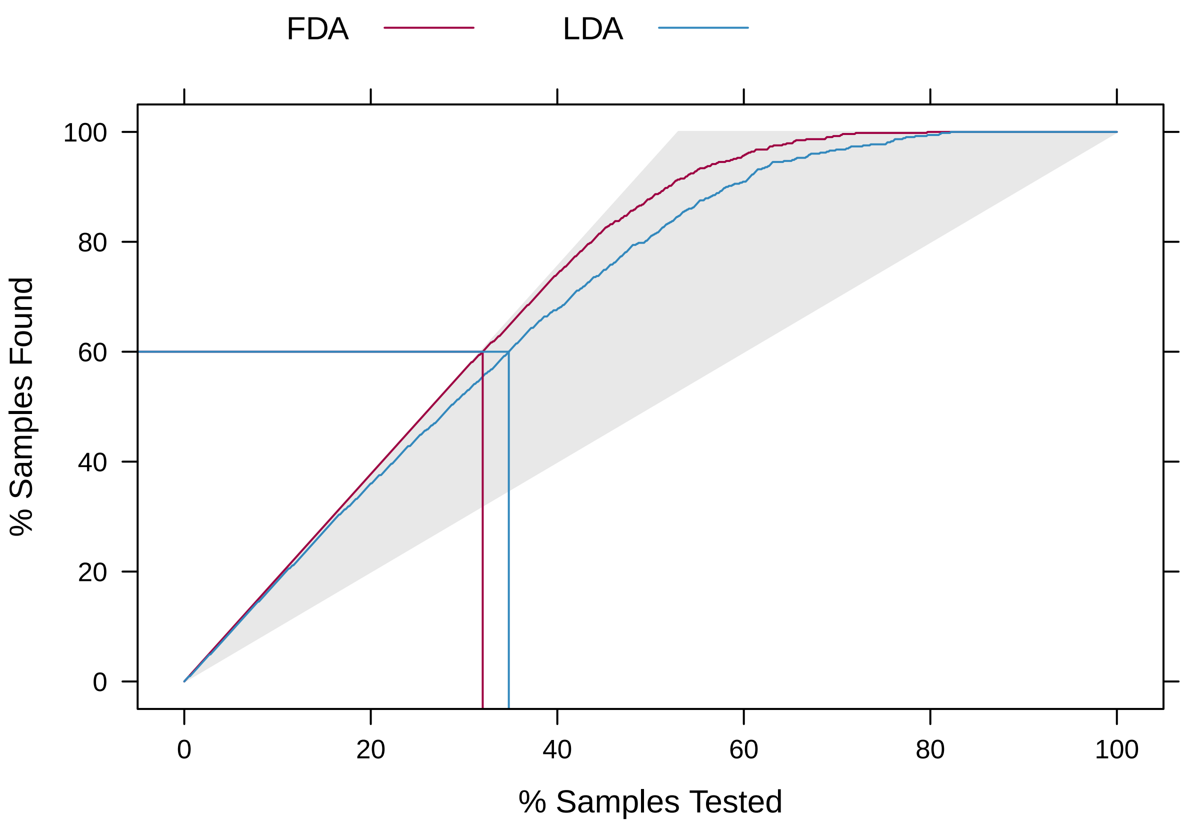
```
## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
#Generate the test set results
lift_results <- data.frame(Class = lift_testing$Class)
lift_results$FDA <- predict(fda_lift, lift_testing, type = "prob")[,"Class1"]
lift_results$LDA <- predict(lda_lift, lift_testing, type = "prob")[,"Class1"]
head(lift_results)
```

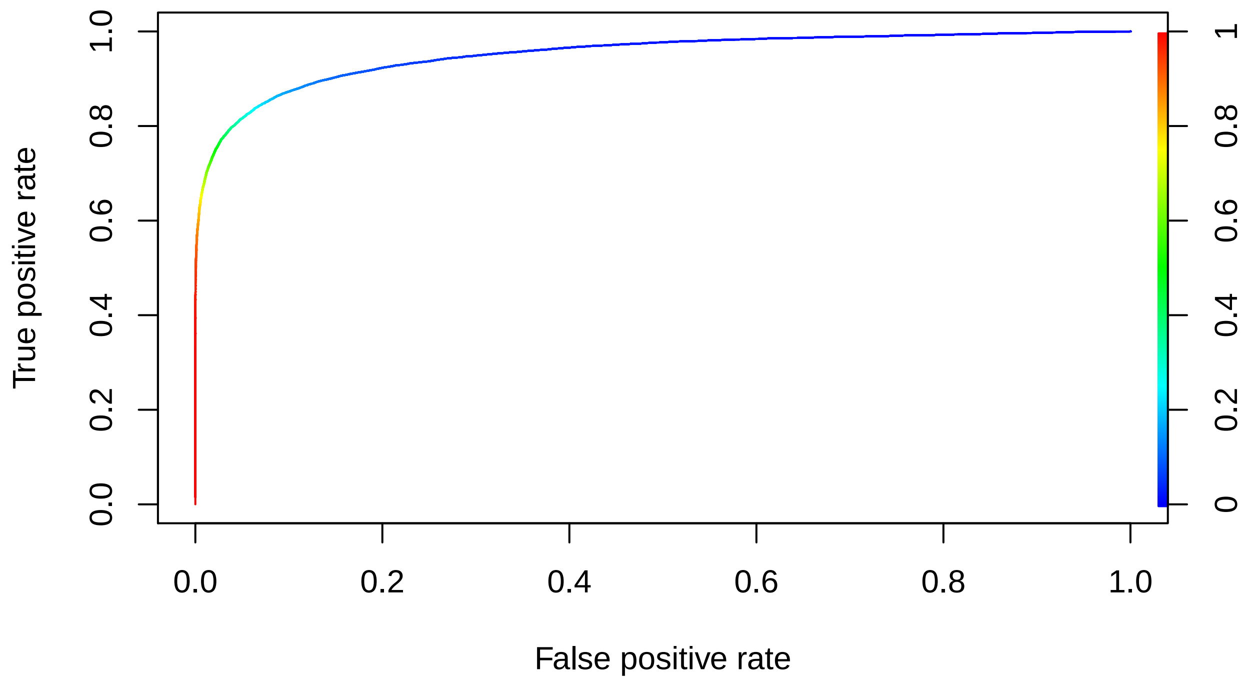
```
##    Class        FDA       LDA
## 1 Class1 0.99244077 0.8838205
## 2 Class1 0.99128497 0.7572450
## 3 Class1 0.82142101 0.8883830
## 4 Class2 0.04336463 0.0140480
## 5 Class1 0.77494981 0.9320695
## 6 Class2 0.11532541 0.0524154
```

```r
#plot lift curve
#graph shows that to find 60% of the hits, more than 30% of the data can be sampled
trellis.par.set(caretTheme())
lift_obj <- lift(Class ~ FDA + LDA, data = lift_results)
plot(lift_obj, values = 60, auto.key = list(columns = 3,
                                            lines = TRUE,
                                            points = FALSE))
```

```
## Warning in draw.key(simpleKey(...), draw = FALSE): not enough rows for
## columns
```



```r
#Compute area under the curve for predicting delay22min with the model
prob <- predict(glmMod, newdata=testing, type="response")
pred <- prediction(prob, testing$delay22min)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")

#Task 3. Create ROC curve for model
# Plot ROC curve
plot(perf, colorize = TRUE)
```




```r
auc <- performance(pred, measure = "auc")
auc <- auc@y.values[[1]]
auc #0.951452
```

```
## [1] 0.951452
```

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

```r
#Task 1. An articulation of the the problem you are trying to solve
#I am trying to find out which airport has the most amount of arrival flight delays greater than 15 min

nycflights3 <- read.csv("nycflights3.csv")
nycflights3$delay15min <- ifelse(nycflights3$arr_delay >= 15, "delay >= 15 min", "delay < 15min")
nycflights3$delay15min <- as.factor(nycflights3$delay15min)
nycflights3 <- nycflights3[complete.cases(nycflights3$delay15min),]

#Task 2. A naive model using linear regression
inTraining3 <- createDataPartition(nycflights3$delay15min, p = .75, list = FALSE)
training3 <- nycflights3[ inTraining3,]
testing3 <- nycflights3[-inTraining3,]

lmMod3 <- lm(arr_delay ~ carrier, data = training3)

#prediction
predlm <- predict(lmMod3, newdata=testing3, type = "response")

#label predictions as same level as last column in NYCflights3 data set and change class to factor
model_predlm <- rep("delay < 15min", 81836)
model_predlm[predlm > 15] <- "delay >= 15 min"
model_predlm <- as.factor(model_predlm)

#train CART model
rpart1 <- rpart(carrier ~ ., data = training3, control = rpart.control(maxdepth = 2))
rpart1
```

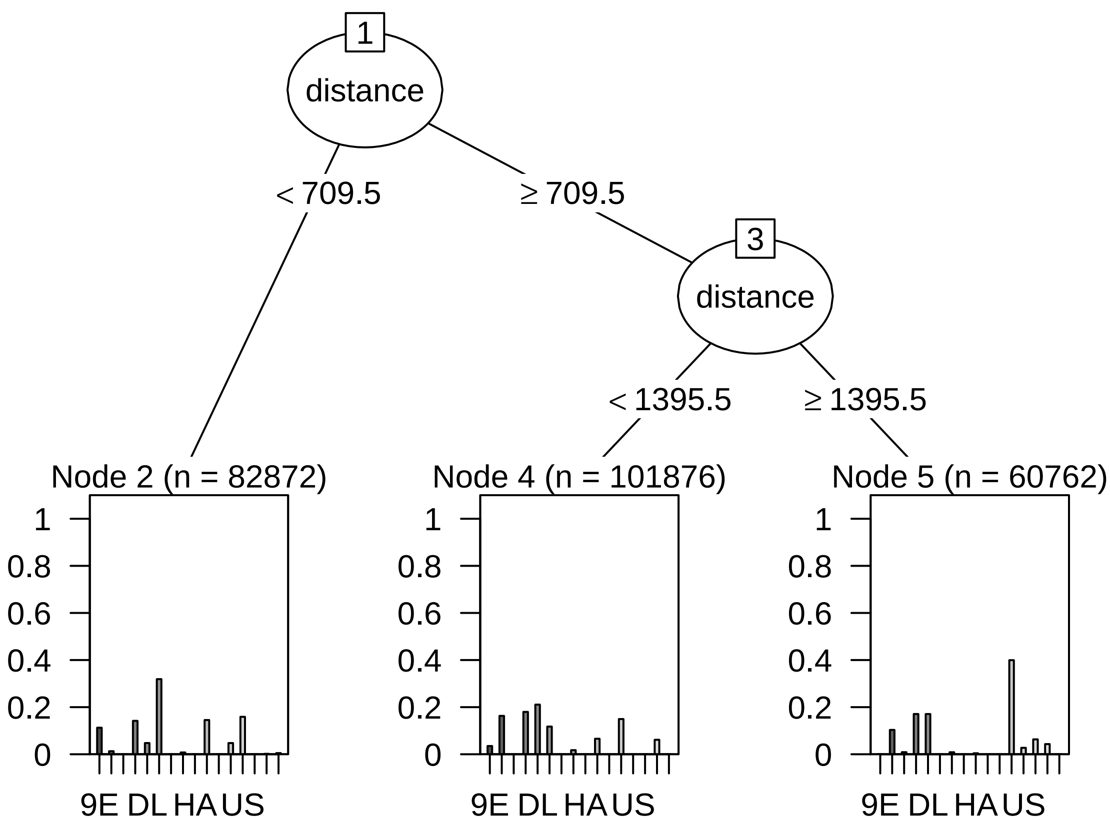
```
## n= 245510
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 245510 202067 UA (0.053 0.098 0.0022 0.16 0.15 0.16 0.0021 0.0097 0.001 0.076 8.1e-05 0.18 0
##   2) distance< 709.5 82872  56446 EV (0.11 0.013 0 0.14 0.048 0.32 0 0.0076 0 0.15 0.00021 0.048 0.1
##   3) distance>=709.5 162638 123138 UA (0.022 0.14 0.0033 0.18 0.2 0.074 0.0032 0.011 0.0015 0.041 1.
##     6) distance< 1395.5 101876  80408 DL (0.035 0.16 0 0.18 0.21 0.12 0 0.017 0 0.066 2.9e-05 0.15 0
##     7) distance>=1395.5 60762  36488 UA (0.00049 0.1 0.0088 0.17 0.17 0 0.0085 0 0.0041 0 0 0.4 0.02
```

```r
#Task 3. An unbiased calculation of the performance metric
#create a confusion matrix using the caret package
confusionMatrix(data=model_predlm, testing3$delay15min)
```

```
## Confusion Matrix and Statistics
##
##                  Reference
## Prediction        delay < 15min delay >= 15 min
##   delay < 15min           52539           15628
##   delay >= 15 min          9272            4397
##
##                Accuracy : 0.6957
##                  95% CI : (0.6926, 0.6989)
##     No Information Rate : 0.7553
##     P-Value [Acc > NIR] : 1
```

```
##
##                   Kappa : 0.0779
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8500
##             Specificity : 0.2196
##          Pos Pred Value : 0.7707
##          Neg Pred Value : 0.3217
##              Prevalence : 0.7553
##          Detection Rate : 0.6420
##    Detection Prevalence : 0.8330
##       Balanced Accuracy : 0.5348
##
##        'Positive' Class : delay < 15min
##
```

```r
#Task 4. A plot of your model -- (the actual tree; there are several ways to do this)
rpart1a <- as.party(rpart1)
plot(rpart1a)
```




```r
#Task 5. A discussion of your model
#Model shows that 24% of the flights are delayed more than 15 minutes while 76% of the flights are on-t
```

Questions:

- Discuss the difference between the models and why you would use one model over the other? I used a linear regression and k-nearest neighbors as my naive models. k-nearest neighbors is a classifcation model whereas linear regression is not.

- How might you produce an ROC type curve for the *rpart* model? I would need to plot the performance of the model which is based on the prediction and actual results.