Ontime Arrivals

Gary R Seamans

4 April 2018

Example Description

The Delayed Airplanes Dataset consists of airplane flights from Washington D.C. into New York City. This dataset can be downloaded from the United States Department of Transportation, Bureau of Transportation Statistics, TranStats.

The domestic market was selected for the analysis. In this dataset are airplane flights between NYC and D.C. If there is a delay of 15 minutes or more, then code the delay variable so that (0 = no delay and 1 = delay). For the predictor variable, the following was used:

- The arrival airports of Newark, Kennedy, LaGuardia, and Newark
- Departure airports (Baltimore, Dulles, and Reagan)
- Carriers
- Hours of departure
- Weather conditions
- Monday = 1, Tuesday = 2, ...Sunday = 7

Bayseian analysis will be used to predict on-time arrival.

Data Preparation

The data was downloaded from [@_rita_air_delay_]. Only those fields that were deemed useful for the analysis were included. The latest delay information was for June 2016. I chose the January 2016 dataset for this analysis. The dataset contained a total of 445827 records for domestic flights.

Once the data was downloaded the data set was loaded and an initial examination performed using the following commands:

```
## Read in the airline delay data
airlineDF <- read.csv(file = "january/january_ontime.csv", header = TRUE)
## Get rid of the X (logic) row
airlineDF <- airlineDF[,1:7]
## Source the code that will convert str() into a table structure
source('strtable.R')
## Create a table of the data types
print(xtable(strtable(airlineDF), caption = "Airline Data Types"))
## Read in the field descriptions and convert them into a table for documentation
termsDF <- read.csv(file = "168748919_T_ONTIME_ReadMe.csv", header = TRUE)
## Load required libraries
library(xtable)
library(plyr)
## Rename the columns
termsDF <- rename(termsDF, c("SYS_FIELD_NAME"="Field Name",
                             "X.FIELD_DESC"="Description"))
## Print the table
print(xtable(termsDF, caption = "Field Descriptions"),
      include.rownames = FALSE)
```

There are 27 variables in the selected data set show in table 1.

Table 1: Airline Data Types

	variable	class	levels	examples
1	DAY_OF_WEEK	integer		1, 1, 1, 1, ...
2	CARRIER	Factor w/ 12 levels	"AA", "AS", "B6", "DL", ...	"AA", "AA", "AA", "AA", ...
3	ORIGIN	Factor w/ 294 levels	"ABE", "ABQ", "ABR", "ABY", ...	"DFW", "DFW", "DFW", "DTW", ...
4	DEST	Factor w/ 294 levels	"ABE", "ABQ", "ABR", "ABY", ...	"DTW", "DTW", "DTW", "DFW", ...
5	DEP_TIME	integer		1107, 1055, 1059, 1509, ...
6	ARR_DEL15	numeric		0, 0, 0, 0, ...
7	WEATHER_DELAY	numeric		NA, NA, NA, NA, ...

Table 2 shows the fields and field descriptions for the airline delay data set that was selected for this project.

Table 2: Field Descriptions

Field Name	Description
DAY_OF_WEEK	Day of Week
CARRIER	Code assigned by IATA and commonly used to identify a carrier. As the same code may have been assigned to different carriers over time, the code is not always unique. For analysis, use the Unique Carrier Code.
ORIGIN_AIRPORT_ID	Origin Airport, Airport ID. An identification number assigned by US DOT to identify a unique airport. Use this field for airport analysis across a range of years because an airport can change its airport code and airport codes can be reused.
ORIGIN	Origin Airport
DEST	Destination Airport
ARR_DEL15	Arrival Delay Indicator, 15 Minutes or More (1=Yes)
WEATHER_DELAY	Weather Delay, in Minutes

The next step in the data preparation will be to select only those rows that contain the starting and ending airports as described in the assignment instructions. There were three departure and three arrival airports chosen for the assignment. Flights from any of the departure airports are tracked to any of the arrival airports.

Table 3: Arrival and Departure Airports

	Departure	ID	Arrival	ID
1	Baltimore	BWI	Kennedy	JFK
2	Dulles	IAD	LaGuardia	LGA
3	Reagan	DCA	Newark	EWR

Using the *ORIGIN* and *DEST* values from the dataset the following R code will select only those rows that originate from one of the departure airport destined for one of the arrival airports.

```
depart <- c("BWI", "IAD","DCA")
arrive <- c("JFK","LGA","EWR")
targetAirports <- airlineDF[ which(airlineDF$ORIGIN %in% depart &
                                   airlineDF$DEST %in% arrive), ]
```

Data Cleaning

Next I'll check for missing data and decide what to do about it.

```
## Check for missing data
print(xtable(data.frame(as.list(sapply(targetAirports, function(x)
  sum(is.na(x)))))), include.rownames = FALSE)
```

We can see that there are three fields with missing values.

DAY_OF_WEEK	CARRIER	ORIGIN	DEST	DEP_TIME	ARR_DEL15	WEATHER_DELAY
0	0	0	0	74	76	534

There are 76 rows where *ARR_DELAY_NEW* and/or *ARR_DEL15* are *N/A*. Since these fields should always have a value they are not useful for our analysis. I'll eliminate them and see if there are any *N/As* remaining other than weather.

```
## Check the number of rows with our target departure and arrival airports
nrow(targetAirports)
[1] 677
## Remove rows with NAs in the ARR_DEL15 field
tgtAirports2 <-
  targetAirports[ which(!(is.na(targetAirports$ARR_DEL15))), ]
nrow(tgtAirports2)
[1] 601
## Set the NAs in the WEATHER field to 0 since there was no known weather delay
tgtAirports2[is.na(tgtAirports2)] <- 0
print(xtable(data.frame(as.list(sapply(tgtAirports2, function(x)
  sum(is.na(x)))))), include.rownames = FALSE)
## Check the number of rows with our target departure and arrival airports
nrow(targetAirports)
[1] 677
## Remove rows with NAs in the ARR_DEL15 field
tgtAirports2 <-
  targetAirports[ which(!(is.na(targetAirports$ARR_DEL15))), ]
nrow(tgtAirports2)
[1] 601
## Set the NAs in the WEATHER field to 0 since there was no known weather delay
tgtAirports2[is.na(tgtAirports2)] <- 0
print(xtable(data.frame(as.list(sapply(tgtAirports2, function(x)
  sum(is.na(x)))))), include.rownames = FALSE)
## Check the number of rows with our target departure and arrival airports
nrow(targetAirports)
[1] 677
## Remove rows with NAs in the ARR_DEL15 field
tgtAirports2 <-
  targetAirports[ which(!(is.na(targetAirports$ARR_DEL15))), ]
nrow(tgtAirports2)
[1] 601
## Set the NAs in the WEATHER field to 0 since there was no known weather delay
tgtAirports2[is.na(tgtAirports2)] <- 0
print(xtable(data.frame(as.list(sapply(tgtAirports2, function(x)
  sum(is.na(x)))))), include.rownames = FALSE)
```

We can see that there were originally 677 rows containing the data of interest and after removing the rows containing *N/As* from the dataset and replacing the *WEATHER NAs* with 0 there are no more missing

Table 4: Missing Values Cleaned

DAY_OF_WEEK	CARRIER	ORIGIN	DEST	DEP_TIME	ARR_DEL15	WEATHER_DELAY
0	0	0	0	0	0	0

values and there are now 601 rows of data.

There are no remaining *N/As* in our cleaned dataset. The next step will be to drop the unused factor levels and convert the *DEP_TIMES* to factors, rounded to the nearest whole hour.

```
## Drop unused levels from the dataset
tgtAirports2 <- droplevels(tgtAirports2)

## First lets convert DEP_DEL15, DAY_OF_THE_WEEK, to factors
tgtAirports2$ARR_DEL15 <- as.factor(tgtAirports2$ARR_DEL15)
tgtAirports2$DAY_OF_WEEK <- as.factor(tgtAirports2$DAY_OF_WEEK)
tgtAirports2$WEATHER_DELAY <- as.factor(tgtAirports2$WEATHER_DELAY)

## Now we'll rename the factors
tgtAirports2$ARR_DEL15 <- mapvalues(tgtAirports2$ARR_DEL15, from =c("0","1"),
                                    to =c("No","Yes"))
tgtAirports2$DAY_OF_WEEK <-
  mapvalues(tgtAirports2$DAY_OF_WEEK, from = c("1", "2", "3", "4","5","6","7"),
            to = c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
                   "Saturday", "Sunday"))

## For the next step I decided to round all of the times to the nearest hour
## and convert them to factors. Since the data only listed the hour and minutes
## this was an interesting problem. There were values like "20" for 20 minutes after
## midnight and "623" for 6:23am, etc.
tgtAirports2$DEP_TIME <- as.factor(unlist(lapply(tgtAirports2$DEP_TIME,
       function(x) format(round(strptime(paste("2016-01-01", mypaste(x)),
                                format="%Y-%m-%d %H%M"), units="hours"),
                          format="%H%M"))))

## Print one last table to check the factors and values
print(xtable(strtable(tgtAirports2),
             caption = "Final Cleaned/Adjusted Airline Data Types"))

## Print a table containing the totoal number of arrivals and departures from our target
## airports.
print(xtable(summary(data.frame(tgtAirports2$ORIGIN,tgtAirports2$DEST)),
             caption = "Departures and Arrivals"), include.rownames = FALSE)
```

Our final, cleaned, dataset contains 601 flight instances for the month of January between our departure and arrival airports.

Table 8 shows the total number of flights originating from the target departure airports and flying non-stop to the target destination airports.

Bayesian Prediction

For the first set I'll create a naive Bayes model using the entire dataset.

Table 5: Final Cleaned/Adjusted Airline Data Types

	variable	class	levels	examples
1	DAY_OF_WEEK	Factor w/ 7 levels	"Monday", "Tuesday", ...	"Monday", "Monday", ...
2	CARRIER	Factor w/ 5 levels	"AA", "B6", "DL", "EV", ...	"AA", "AA", "AA", "AA", ...
3	ORIGIN	Factor w/ 3 levels	"BWI", "DCA", "IAD"	"DCA", "DCA", "DCA", "DCA", ...
4	DEST	Factor w/ 3 levels	"EWR", "JFK", "LGA"	"JFK", "JFK", "JFK", "LGA", ...
5	DEP_TIME	Factor w/ 19 levels	"0000", "0600", ...	"1800", "1800", ...
6	ARR_DEL15	Factor w/ 2 levels	"No", "Yes"	"No", "No", "Yes", "No", ...
7	WEATHER_DELAY	Factor w/ 7 levels	"0", "1", "6", ...	"0", "0", ...

Table 6: Departures and Arrivals

ORIGIN	DEST
BWI: 9	EWR:159
DCA:482	JFK:129
IAD:110	LGA:313

```
## Create a Naive Bayes model
delayNB <- naiveBayes(ARR_DEL15 ~ DEP_TIME + CARRIER + WEATHER_DELAY
            + DAY_OF_WEEK + ORIGIN, data = tgtAirports2)

delayNB ## Extract the A-piori

nbPredict <- predict(delayNB, tgtAirports2)
print(xtable(table(nbPredict, tgtAirports2$ARR_DEL15),caption = "Predictions for Delay"))
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
       No       Yes
0.7620632 0.2379368
```

Table 7: Predictions for Delay

	No	Yes
No	435	109
Yes	23	34

```
## Doing a simple cross table evaluation
sum(diag(nbTable))/sum(nbTable)
[1] 0.7803661
```

It appears that the model correctly predicts arrival delays of 15 minutes or more approximately 78% of the

time. Next I'll break up the data into a training and a test dataset.

Training, Test, and the Confusion Matrix

```
nbTrainIndex <- createDataPartition(tgtAirports2$ARR_DEL15, p=0.75, list=FALSE)
nbTrain <- tgtAirports2[nbTrainIndex,]
nbTest <- tgtAirports2[-nbTrainIndex,]
nbModel <- naiveBayes(ARR_DEL15 ~ DEP_TIME + CARRIER + WEATHER_DELAY +
                        DAY_OF_WEEK + ORIGIN, data = nbTrain)
## Create the test data set minus ARR_DEL15
nbTest_X <- nbTest[,-6]
## Create the list of expected results
nbTest_Y <- nbTest[,6]
## Do the prediction
nbPredict <- predict(nbModel, nbTest_X)
confusionMatrix(nbPredict, nbTest_Y)
```

Confusion Matrix

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  104  28
       Yes  10   7

               Accuracy : 0.745
                 95% CI : (0.6672, 0.8128)
    No Information Rate : 0.7651
    P-Value [Acc > NIR] : 0.75349

                  Kappa : 0.1366
 Mcnemar's Test P-Value : 0.00582

            Sensitivity : 0.9123
            Specificity : 0.2000
         Pos Pred Value : 0.7879
         Neg Pred Value : 0.4118
             Prevalence : 0.7651
         Detection Rate : 0.6980
   Detection Prevalence : 0.8859
      Balanced Accuracy : 0.5561

       'Positive' Class : No
```

The confusion matrix results from the training and test datasets shows an accuracy of approximately 75% which validates our earlier result with the entire dataset that showed approximately a 78% accuracy.

Summary

A simple Bayesian classification model was created and tested using airport delay information for January 2016. The model was able to successfully predict delays of 15 minutes or more for approximately 75% of the cases. The major challenge for this exercise was the manipulation of the data to prepare it for analysis, but, in my experience, that is quite often the case.