SmartFly: Prepare Data For Modeling

Cindy Lamm

12:15, Friday 16th January, 2015

Load preprocessed data from the previous step "Exploratory Analysis For Historic Flight Data"

```
rm(list=ls())   #clear memory
load("../01_exploratory_data_analysis/trainDataTyped.RData")
modelData <- trainDataTyped
rm(trainDataTyped)
```

1 Remove variables that don't help with prediction

Before I do an analysis of how many rows contain any missing values, I remove the variables that I won't use for estimating the model. I exclude the variable `id` because I assume it is randomly assigned to the observation and has no predictive power regarding the delay of a flight. Furthermore I exlcude the variables `departure_delay`, `taxi_time_in`, `taxi_time_out`, `cancelled`, `cancellation_code` because they are not available in the scheduled flight data.

```
nonAvailable <- c("id", "departure_delay", "taxi_time_in", "taxi_time_out",
                  "cancelled", "cancellation_code")
excludeIdx <- sapply(nonAvailable, FUN=function(v, x){ which(v==x)}, v=names(modelData))
modelData <- modelData[,-excludeIdx]
str(modelData)

## 'data.frame': 7374365 obs. of  15 variables:
##  $ year                    : Factor w/ 2 levels "2013","2014": 1 1 1 1 1 1 1 1 1 1 ...
##  $ month                   : Factor w/ 12 levels "01","02","03",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ day_of_month            : Factor w/ 31 levels "01","02","03",..: 11 17 18 24 25 31 1 2 3 4 ...
##  $ day_of_week             : Factor w/ 7 levels "1","2","3","4",..: 7 6 7 6 7 6 4 5 6 7 ...
##  $ scheduled_departure_time: Factor w/ 24 levels "00","01","02",..: 11 11 11 11 11 11 8 8 8 8 ...
##  $ scheduled_arrival_time  : Factor w/ 24 levels "00","01","02",..: 12 12 12 12 12 12 9 9 9 9 ...
##  $ airline                 : Factor w/ 17 levels "AA","AS","B6",..: 15 15 15 15 15 15 15 15 15 15 ...
##  $ flight_number           : Factor w/ 6889 levels "1","10","100",..: 6744 6744 6744 6744 6744 6744 6
##  $ tail_number             : Factor w/ 5035 levels "0","000000","N050AA",..: 3898 3963 3806 3810 4008
##  $ plane_model             : Factor w/ 6 levels "737","747","757",..: 3 3 5 2 5 2 2 3 2 6 ...
##  $ seat_configuration      : Factor w/ 6 levels "Standard","Three Class",..: 2 1 4 5 4 5 2 1 5 2 ...
##  $ origin_airport          : Factor w/ 279 levels "ABE","ABI","ABQ",..: 46 46 46 46 46 46 133 133 133
##  $ destination_airport     : Factor w/ 279 levels "ABE","ABI","ABQ",..: 61 61 61 61 61 61 61 61 61 61
##  $ distance_travelled      : num  361 361 361 361 361 361 185 185 185 185 ...
##  $ is_delayed              : Factor w/ 2 levels "on_time","delayed": 1 2 1 1 1 1 1 1 2 1 ...
```

2 Remove variables due to randomForest constraint

Since I will use the `randomForest`[1] package I also remove the factor variables that have more than 53 levels since otherwise an error occurs.

```
modelFactorIdx <- which(sapply(modelData, FUN=class) == "factor")
modelFactorLevels <- sapply(modelData, FUN=levels)
nbLevels <- sapply(modelFactorLevels, FUN=length)
suitable <- which(nbLevels < 53) # condition for this randomForest implementation
rfModelData <- modelData[,suitable]
rm(modelData)
```

So the variables that I use for the estimation of a random forest are as follows:

```
str(rfModelData)

## 'data.frame': 7374365 obs. of  11 variables:
##  $ year                    : Factor w/ 2 levels "2013","2014": 1 1 1 1 1 1 1 1 1 1 ...
##  $ month                   : Factor w/ 12 levels "01","02","03",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ day_of_month            : Factor w/ 31 levels "01","02","03",..: 11 17 18 24 25 31 1 2 3 4 ...
##  $ day_of_week             : Factor w/ 7 levels "1","2","3","4",..: 7 6 7 6 7 6 4 5 6 7 ...
##  $ scheduled_departure_time: Factor w/ 24 levels "00","01","02",..: 11 11 11 11 11 11 8 8 8 8 ...
##  $ scheduled_arrival_time  : Factor w/ 24 levels "00","01","02",..: 12 12 12 12 12 12 9 9 9 9 ...
##  $ airline                 : Factor w/ 17 levels "AA","AS","B6",..: 15 15 15 15 15 15 15 15 15 15 ...
##  $ plane_model             : Factor w/ 6 levels "737","747","757",..: 3 3 5 2 5 2 2 3 2 6 ...
##  $ seat_configuration      : Factor w/ 6 levels "Standard","Three Class",..: 2 1 4 5 4 5 2 1 5 2 ...
##  $ distance_travelled      : num  361 361 361 361 361 361 185 185 185 185 ...
##  $ is_delayed              : Factor w/ 2 levels "on_time","delayed": 1 2 1 1 1 1 1 1 2 1 ...
```

I save these variable names for the prediction step since I use the same variable base for prediction and modeling:

```
modelVariables <- names(rfModelData)
save(modelVariables, file="../02_prepare_data_for_modeling/modelVariables.RData")
```

3 Convert date and time related variables from factors to numbers

When predicting from a randomForest it expects the same factor levels to be present in the training and prediction data set. For dates and times I can work around that by using numeric values instead of factor levels.

```
str(rfModelData)

## 'data.frame': 7374365 obs. of  11 variables:
##  $ year                    : num  2013 2013 2013 2013 2013 ...
##  $ month                   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ day_of_month            : num  11 17 18 24 25 31 1 2 3 4 ...
##  $ day_of_week             : num  7 6 7 6 7 6 4 5 6 7 ...
##  $ scheduled_departure_time: num  10 10 10 10 10 10 7 7 7 7 ...
##  $ scheduled_arrival_time  : num  11 11 11 11 11 11 8 8 8 8 ...
##  $ airline                 : Factor w/ 17 levels "AA","AS","B6",..: 15 15 15 15 15 15 15 15 15 15 ...
```

[1] http://www.stat.berkeley.edu/~breiman/RandomForests/cc_manual.htm

```
##  $ plane_model             : Factor w/ 6 levels "737","747","757",..: 3 3 5 2 5 2 2 3 2 6 ...
##  $ seat_configuration      : Factor w/ 6 levels "Standard","Three Class",..: 2 1 4 5 4 5 2 1 5 2 ...
##  $ distance_travelled      : num  361 361 361 361 361 361 185 185 185 185 ...
##  $ is_delayed              : Factor w/ 2 levels "on_time","delayed": 1 2 1 1 1 1 1 1 2 1 ...
```

4 Analyse & deal with missing values

Check for missing values in any of the remaining variables:

```
nbRows <- dim(rfModelData)[1]
rowHasNa <- apply(rfModelData, MARGIN=1, FUN=function(row){ any(is.na(row)) })
nbRowsWithNa <- sum(rowHasNa)
nbRowsLeft <- nbRows - nbRowsWithNa
# proportion of NA rows:
nbRowsWithNa / nbRows

## [1] 0
```

There are no rows with missing values (after we deleted variables that we won't use for modeling anyway).

I save the model data for the estimation step:

```
save(rfModelData, file="../02_prepare_data_for_modeling/rfModelData.RData")
```