Predicting Train Arrival Status - On Time or Late

Introduction

Adanna Alutu

June 6, 2017

At the beginning of the project, it was hard to come up with a good data to analyze and predict the outcome. Initially I wanted to work on data from my job but after we coudn't see much dependence among the fields that made sense, my mentor Dr. Shmuel Naaman advised me to scout for data from other internet sites he recommended.

Since I take the train most of the time and experienced delay issues many times that has ranged from 10 mins to 2 hours, I became interested in working on transportation data for trains. This is because I want to experience the process of predicting outcomes which is made possible through Data Science. I want to focus on the steps that will make it possible for me and my mentor Dr Shmuel Naaman to predict the arrival times of the train. The possibility of cutting down the delays experienced in waiting for the train no longer seems to be far fetched. My mentor agreed with me and the Septa Train data from Kaggle website was a good option to work on. There were 3 different datasets available to work on but I chose the "on time performance" which I felt has more relevant features, variables and observations and also has sufficient data for the analysis, tests involved.

The variables in the dataset include: 1. train_id 2. status 3. origin 4. direction 5. next_station 6. timeStamp 7. date

subtitle:

Data Exploration

Several steps were taken to ensure elaborate data analysis and wrangling. Every bit of the data was maximized. We went beyond using the provided variables by creating new ones, removing unnecessary data and testing with reliable tools to get quality, reliable results that can be tested with any dataset.

It was necessary to take the following steps to ensure that all the combinations, dicing and testing would yield a meaningful interpretation and prediction that will help tell us with high confidence when the train will be late:

I. We first tried to plot charts with the entire data but the plots were too crowded and blurry to make any sense. The scales were distorted with big units affected by the outliers.

II. GGPlot bar charts were used to plot and observe the trends and statistics summary but the dataset was too huge for the charts.

III. My mentor suggested shuffling the data and taking the first 20percent as sample to work on. Using the formula below, the row-wise shuffling was done first before the column was then shuffled:

IV. We used the data to fit in several models which include:

- GGPlot with different combination of the variables.
- Linear regression model which was used different ways to get the best stastical summary. Including using some of the observations as variables.
- CART model with focus on the classification method because most of the variables in the data are categorical and the prediction is binary with 0 as "on Time" and 1 as "Late"
- Random Forest which created it's own model that highlighted the top more meaningful variables that contributed majorly in predicting the outcome.

Each of these models were implemented because the train dataset contains a mixture of numerical and categorical variables. Converting their types to either numeric or factors wasn't sufficient. To get the benefit of all the variables, it was essential to test these models.

subtitle:

Data Wrangling

Some data manipulations were done which include: + splitting some of the original variables nto separate variables. For example, time stamp variable was split into six variables. year, month day, hour, min, seconds. + Irrelevant variables were removed or set to null so they would not appear in the dataframe used for the predictions. + Some of observations from the wkday and day of month variables were converted to variables and they significantly improved the statistics of the models. The additions however increased the number of variables from 11 to 58. + Units attached to the dependent variable observations were removed to enable convesions to different types and allow plotting with only the observations of the same type. + The dependent variable "status" observations of "on Time" were replaced with "0" using gsub so that all the observations for the variable will match and easier to manipulate."On time" meant the train arrived as scheduled so it made sense to use "0" to represent no delay.

subtitle:

A Peek into some new variables

This section shows the summary of the SEPTA train data and the first few records using the head().

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
## Warning: Too many values at 150009 locations: 1, 2, 3, 4, 5, 6, 7, 8, 9,
## 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...
```

```
## 'data.frame':    150009 obs. of  45 variables:
##  $ status      : num  1.61 1.79 0 0 0 ...
##  $ origin      : Factor w/ 177 levels "16th St Jct",..: 54 141 115 65 32 151 49 49 151 101 ...
##  $ hour        : chr  "16" "08" "05" "09" ...
##  $ minute      : chr  "55" "54" "04" "32" ...
##  $ month       : num  4 9 6 3 6 10 9 6 10 6 ...
##  $ wkday1      : num  1 0 0 0 1 0 0 0 0 0 ...
##  $ wkday2      : num  0 0 1 0 0 0 0 1 0 0 ...
##  $ wkday3      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ wkday4      : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ wkday5      : num  0 1 0 0 0 0 1 0 0 1 ...
##  $ wkday6      : num  0 0 0 0 0 1 0 0 1 0 ...
##  $ wkday7      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday1   : num  1 0 0 0 0 0 0 0 0 0 ...
##  $ monthday2   : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ monthday3   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday4   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday5   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday6   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday7   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday8   : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ monthday9   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday10  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday11  : num  0 0 0 0 0 1 0 0 1 0 ...
##  $ monthday12  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday13  : num  0 0 1 0 0 0 0 1 0 0 ...
##  $ monthday14  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday15  : num  0 0 0 0 0 0 1 0 0 0 ...
##  $ monthday16  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday17  : num  0 0 0 0 1 0 0 0 0 0 ...
##  $ monthday18  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday19  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday20  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday21  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday22  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday23  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday24  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday25  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday26  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday27  : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ monthday28  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday29  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday30  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthday31  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ next_station: Factor w/ 155 levels "30th Street Station",..: 87 125 96 39 120 43 99 91 19 47 ...
##  $ direction   : Factor w/ 2 levels "N","S": 2 2 2 1 2 1 2 2 1 1 ...
```

subtitle:

CART Model /Decision Tree

:

In this section, the CART model is implemented. The two options considered are Classification and Regression CART models/trees but the reression model is preferred so that the results can be compared with the linear regression model used above. It's like comparing apples to apples or oranges to oranges.

I found this site very helpful because they explained in detail the conditions for the variables before a succesful model can be achieved - https://rstudio-pubs-static.s3.amazonaws.com/27179_e64f0de316fc4f169d6ca300f18ee2aa.html.

Prior to finding this site, only the root or just one circle with a number (4.6) was drawn.

```
library(caTools)
set.seed(3000)

smp_size <- floor (0.8 *nrow(Data))

train_ind <- sample (seq_len(nrow(Data)), size=smp_size)

train <- Data[train_ind, ]
test  <- Data[-train_ind,]


#build CART model

library(rpart)
library(rpart.plot)

#now create the CART model. Use rpart to build a linear regression tree since the status variable being

#use rpart formula to fit the data.

#logvar <- log(as.numeric( train_data$status)+1)
TraindataTree2 = rpart(status ~. , data = train)



actual <- test$status
predicted <- predict(TraindataTree2, newdata = test )

R2_test <- 1 - (sum((actual-predicted )^2)/sum((actual-mean(actual))^2))



actual <- train$status
predicted <- predict(TraindataTree2, newdata = train )

R2_train <- 1 - (sum((actual-predicted )^2)/sum((actual-mean(actual))^2))
R2_test
```

```
## [1] 0.207283
```

```
R2_train
```

```
## [1] 0.2086067
```

```
#print R2
```

Check r-squared for Training and Test data for Regression model

```
set.seed(2)

rm(R2lmtrn)
```

```
## Warning in rm(R2lmtrn): object 'R2lmtrn' not found
```

```
rm(R2lmtst)
```

```
## Warning in rm(R2lmtst): object 'R2lmtst' not found
```

```
rm(jj)
```

```
## Warning in rm(jj): object 'jj' not found
```

```
x <- .01 * 150000
#j
R2lmtrn <- 0
R2lmtst <- 0
jj<-0
k<-0
for(b in 1:x){

    if(x >=150000){
      #get out of loop once all data has been processed
      break
    }

train_lm = train[1:x,]

#k=k+1
jj[k] <- x

TraindataLM <- lm(status ~ ., data = train_lm)

#data = Trainset[complete.cases(b),])

predstrn <- predict(TraindataLM, data = train)

predstest <- predict(TraindataLM, data = test)

R2lmtrn[k] <- 1-sum((train$status-predstrn)^2)/sum((train$status-mean(train$status))^2)

R2lmtst[k] <- 1-sum((test$status-predstest)^2)/sum((test$status-mean(test$status))^2)

k=k+1

x <- x+x

}
```

```
## Warning in train$status - predstrn: longer object length is not a multiple
## of shorter object length

## Warning in test$status - predstest: longer object length is not a multiple
## of shorter object length

## Warning in train$status - predstrn: longer object length is not a multiple
## of shorter object length

## Warning in test$status - predstest: longer object length is not a multiple
## of shorter object length

## Warning in train$status - predstrn: longer object length is not a multiple
## of shorter object length

## Warning in test$status - predstest: longer object length is not a multiple
## of shorter object length
## Warning in train$status - predstrn: longer object length is not a multiple
## of shorter object length

## Warning in test$status - predstest: longer object length is not a multiple
## of shorter object length

## Warning in train$status - predstrn: longer object length is not a multiple
## of shorter object length

## Warning in test$status - predstest: longer object length is not a multiple
## of shorter object length

## Warning in train$status - predstrn: longer object length is not a multiple
## of shorter object length

## Warning in test$status - predstest: longer object length is not a multiple
## of shorter object length

## Warning in train$status - predstrn: longer object length is not a multiple
## of shorter object length

## Warning in test$status - predstest: longer object length is not a multiple
## of shorter object length
```

```
#list r-squared for Linear Regression Training and Test data
R2lmtrn
```

```
## [1] -0.30380775 -0.25512980 -0.20361653 -0.15246999 -0.04970087  0.14952646
```

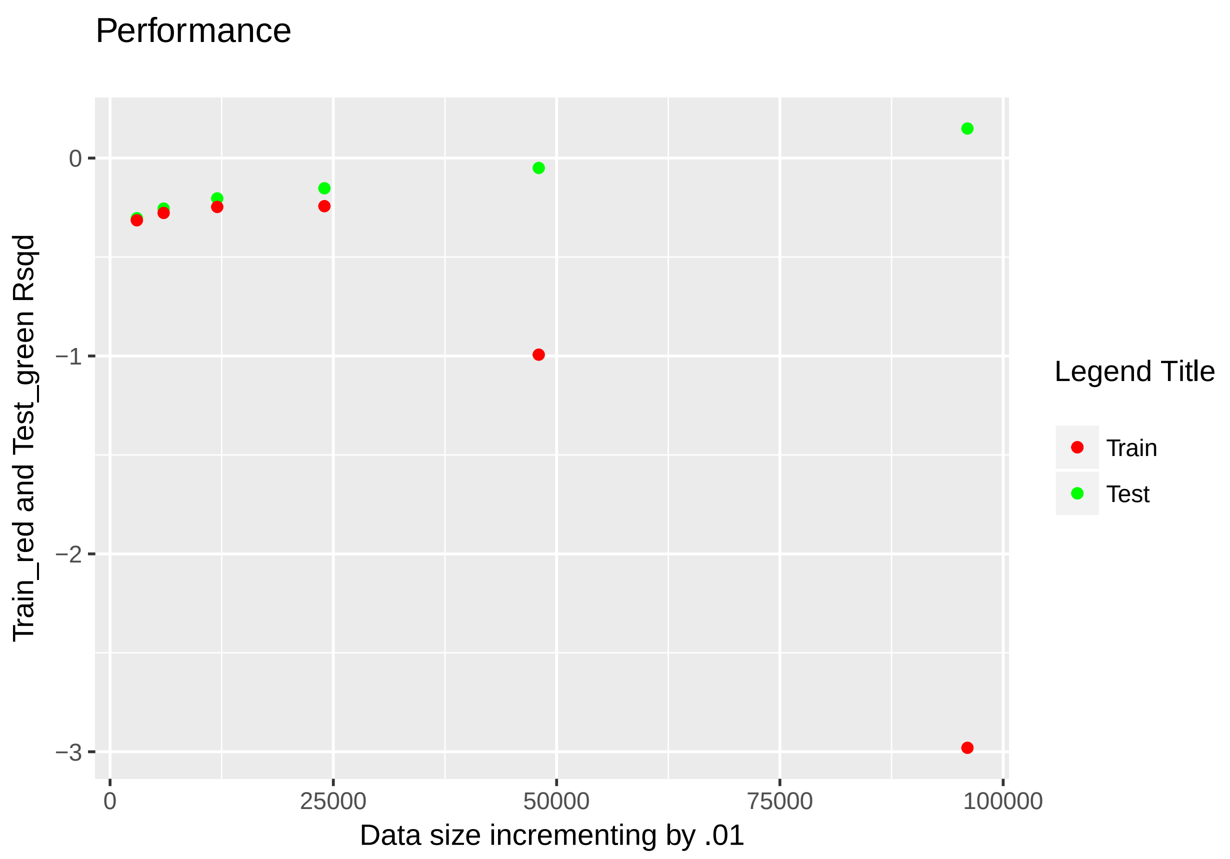
```
R2lmtst
```

```
## [1] -0.3143426 -0.2770520 -0.2467380 -0.2429246 -0.9933706 -2.9801395
```

```
md <- data.frame(jj, R2lmtrn, R2lmtst)
print(md)
```

```
##      jj     R2lmtrn    R2lmtst
## 1  3000 -0.30380775 -0.3143426
## 2  6000 -0.25512980 -0.2770520
## 3 12000 -0.20361653 -0.2467380
## 4 24000 -0.15246999 -0.2429246
## 5 48000 -0.04970087 -0.9933706
## 6 96000  0.14952646 -2.9801395
```

```
  ggplot(md) +
    geom_point(aes(x = jj, y = R2lmtrn, color = "red")) +
  geom_point(aes(x = jj, y = R2lmtst, color = "green"))+
    labs(title = "Performance \n", x = "Data size incrementing by .01", y = "Train_red and Test_green R
  scale_color_manual(labels = c("Train", "Test"), values = c("red", "green"))
```



subtitle:

Conclusion from the models

The r-squared value from the Linear Regression model is 0.33506 ~34%. The Training set performance (r-squared) is .308 ~31%. The Testing set r-squared is 0.

The r-squared value from CART Regression model/ Decision Tree is .318 ~ 32%. The Training set best performance is 18% while the Test performance is 0.

The r-squared value from the Random Forest Training model at ntree=22 is .214 ~ 21% and as expected, the performance on the Training set is better than the performance on the Testing set.The r-squared value from the Random Forest Testing model is .101~ 10%. The word document named "finalproject_randomforest.docx" lists the different r-squared values at ntree values = 15, 10, 5 and 1. At the ntree value of 5, the r-squared or performance is 21% which is slightly better than the performance at ntree = 22 which is .202 ~ 20%.

The model that gave us the best performance is the Linear Regression model because it's R-squared is slightly higher than the CART regression and Random Forest models. Rhe Testing set performance is 31%, which is higher than the training performance of the other models.