Predicting Delayed Flights

Group 4 - Analytics in Practice

Table of Contents

0. WORKSPACE SETUP

load relevant libraries

```r
library(caret)
library(tidyverse)
library(skimr)
library(plyr)
library(data.table)
library(bit64)
library(dummies)
library(randomForest)
library(e1071)
library(pROC)
```

1.BUSINESS REQUIREMENTS

Group 4 has been recently employed as the business analytics team for a major US airline and their first task is to analyze flight data from the previous year (2019) which can be used to predict flight delay. Their objective is straightforward -

OBJECTIVE: interpret 2019 flight data and build a model that accurately predicts flight delay to be used to decrease delay in 2020 and, thereby, *increase customer satisfaction*.

The business requirements of this objective will be to correctly interpret the raw data, clean and process that data, select and engineer relevant features, use those features to predict and fine-tune the chosen machine learning model, and interpret the results for use by the airline's upper management and decision-makers. Following is Group 4's full report and model.

2. OVERVIEW OF DATA

Gathering

Our data gathering process was spent reading through and mining information from many websites that host free, accessible transportation data specific to flight travel. The primary source was **The Bureau of Transportation Statistics** website, maintained by the U.S. Federal Government. Note: Although this is the same foundational source for the data that we used during our ggplot2 visualization assignment, it's *not the same dataset*. In fact, we curated *12 new datasets*, one for each month of flight data in the year 2019, and combined them into one master dataset with over **7 million** unique observations and **35 variables**. This master dataset is where we began our exploration process for modeling flight delays, and its creation process is shown below:

```r
# importing and listing each monthly dataset from the year 2019
dir = "datasets"
files = list.files(path = dir,
                   pattern = "*.csv",
                   full.names = TRUE)
files

##  [1] "datasets/april19.csv"     "datasets/august19.csv"
##  [3] "datasets/december19.csv"  "datasets/february19.csv"
##  [5] "datasets/january19.csv"   "datasets/july19.csv"
##  [7] "datasets/june19.csv"      "datasets/march19.csv"
##  [9] "datasets/may19.csv"       "datasets/november19.csv"
## [11] "datasets/october19.csv"   "datasets/september19.csv"

# join 12 monthly datasets to one 2019 yearly dataset
flightdf = ldply(files, read_csv)
```

Summarized Overview

```
skim(flightdf)
```

Data summary

Name	flightdf
Number of rows	7422037
Number of columns	35
_______________________	
Column type frequency:	
character	11
Date	1
logical	1
numeric	22
________________________	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
OP_UNIQUE_CARRIER	1086599	0.85	1	2	0	20	0
ORIGIN	0	1.00	3	3	0	360	0
ORIGIN_CITY_NAME	0	1.00	8	34	0	352	0
ORIGIN_STATE_ABR	0	1.00	2	2	0	52	0
DEST	0	1.00	3	3	0	360	0

DEST_CITY_NAME	0	1.00	8	34	0	352	0
DEST_STATE_ABR	0	1.00	2	2	0	52	0
CRS_DEP_TIME	0	1.00	4	4	0	1379	0
DEP_TIME	130086	0.98	4	4	0	1440	0
CRS_ARR_TIME	0	1.00	4	4	0	1436	0
ARR_TIME	137646	0.98	4	4	0	1440	0

Variable type: Date

skim_variable	n_missing	complete_rate	min	max	median	n_unique
FL_DATE	0	1	2019-01-01	2019-12-31	2019-07-04	365

Variable type: logical

skim_variable	n_missing	complete_rate	mean	count
X35	7422037	0	NaN	:

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
YEAR	0	1.00	2019.00	0.00	2019	2019	2019	2019	2019	▁▁▇▁▁
MONTH	0	1.00	6.58	3.40	1	4	7	10	12	▇▇▇▇▇
DAY_OF_WEEK	0	1.00	3.94	2.00	1	2	4	6	7	▇▅▅▆▇
ORIGIN_AIRPORT_ID	0	1.00	12648.88	1523.85	10135	11292	12889	13931	16869	▇▃▇▆▁
DEST_AIRPORT_ID	0	1.00	12648.81	1523.82	10135	11292	12889	13931	16869	▇▃▇▆▁
DEP_DELAY	130110	0.98	10.92	48.96	-82	-5	-2	7	2710	▇▁▁▁▁
DEP_DEL15	130110	0.98	0.19	0.39	0	0	0	0	1	▇▁▁▁▂
ARR_DELAY	153805	0.98	5.41	51.07	-99	-15	-6	7	2695	▇▁▁▁▁
ARR_DEL15	153805	0.98	0.19	0.39	0	0	0	0	1	▇▁▁▁▂

CANCELLED	0	1.00	0.02	0.13	0	0	0	0	1	▇▁▁▁▁
DIVERTED	0	1.00	0.00	0.05	0	0	0	0	1	▇▁▁▁▁
CRS_ELAPSED_TIME	135	1.00	141.90	72.35	1	90	124	171	948	▇▃▁▁▁
ACTUAL_ELAPSED_TIME	153805	0.98	136.69	72.61	15	84	119	167	1604	▇▁▁▁▁
AIR_TIME	153805	0.98	111.57	70.56	4	60	93	141	1557	▇▁▁▁▁
FLIGHTS	0	1.00	1.00	0.00	1	1	1	1	1	▁▁▇▁▁
DISTANCE	0	1.00	800.54	592.51	31	369	640	1034	5095	▇▃▁▁▁
DISTANCE_GROUP	0	1.00	3.68	2.33	1	2	3	5	11	▇▅▁▁▁
CARRIER_DELAY	6032784	0.19	21.13	66.10	0	0	0	18	2695	▇▁▁▁▁
WEATHER_DELAY	6032784	0.19	3.80	32.36	0	0	0	0	1847	▇▁▁▁▁
NAS_DELAY	6032784	0.19	16.59	39.66	0	0	2	20	1741	▇▁▁▁▁
SECURITY_DELAY	6032784	0.19	0.10	3.36	0	0	0	0	1078	▇▁▁▁▁
LATE_AIRCRAFT_DELAY	6032784	0.19	27.41	53.46	0	0	3	33	2206	▇▁▁▁▁

Initially summarizing the 2019 flights dataset, it's apparent that it'll need some processing before use in a prediction model. Many of the variables are not of the correct type and will need to be converted. Ex: Our target variable will be DEP_DEL15, a variable that has **value = 1 for delayed** (15 minutes or longer past scheduled departure) and **value = 0 for ontime**. This is listed as a numeric type variable, but is required to be a binary categorical variable of type *factor* for effective classification results.

Further, many variables possess missing values. These will need analysis to determine if the missing values make sense given the nature of the data, need imputed and kept, or rows with missing values need dropped completely. Finally, not all variables can be relevant for prediction; feature selection will determine what to keep and what to remove.

3. DATA EXPLORATION

Trimming Data

Here, we drop irrelevant variables and observations with missing values in key variables. Variables like YEAR and DISTANCE_GROUP are not needed when all the data is from one year and when other variables account for distance. Each variable in the drop index has been determined following similar logic.

```
# dropping observations with missing airline info
df.clean <- flightdf %>% drop_na(OP_UNIQUE_CARRIER)

# creating drop index
drops <- c("YEAR", "X35", "ORIGIN_AIRPORT_ID", "DEST_AIRPORT_ID", "DISTANCE_G
ROUP", "ORIGIN_CITY_NAME", "DEST_CITY_NAME", "FLIGHTS", "ACTUAL_ELAPSED_TIME"
, "AIR_TIME", "CRS_ELAPSED_TIME", "CRS_DEP_TIME", "CRS_ARR_TIME", "DEP_TIME",
"ARR_TIME")

# dropping negligible variables by drop index of names
df.clean <- df.clean[, !(names(df.clean) %in% drops)]

# rename key variables to more descriptive name
df.clean <- df.clean %>% rename(c("OP_UNIQUE_CARRIER"="AIRLINE", "FL_DATE"="D
ATE"))
```

Missing Values

Here, we determine that the remaining variables with missing values are properly missing. Meaning SECURITY_DELAY, for example, will only have data in any given observation if that particular flight was delayed because of an airport security issue, and will otherwise be missing. Because these NA values harbor important flight delay information, we convert NA values to value = 0 and fill in delay information across the five delay variables to value = 1.

```
# replace missing values in type of delay variables with value of 0
df.clean <- df.clean %>% replace_na(list(CARRIER_DELAY = 0,
                                          WEATHER_DELAY = 0,
                                          NAS_DELAY = 0,
                                          SECURITY_DELAY = 0,
                                          LATE_AIRCRAFT_DELAY = 0))
```

```
# convert standing values in type of delay variables to 1 for yes and 0 for
no
df.clean <- df.clean %>% mutate(CARRIER_DELAY = ifelse(CARRIER_DELAY > 0, 1,
0),
                                  WEATHER_DELAY = ifelse(WEATHER_DELAY > 0,
1, 0),
                                  NAS_DELAY = ifelse(NAS_DELAY > 0, 1, 0),
                                  SECURITY_DELAY = ifelse(SECURITY_DELAY > 0,
1, 0),
                                  LATE_AIRCRAFT_DELAY =
ifelse(LATE_AIRCRAFT_DELAY > 0, 1, 0))
```

Variable Type Conversion

Next, we convert variables to their proper types. This is necessary for any prediction model to approriately interpret key variables.

```
# vector of categorical variables to convert to factor type
fconvert <- c("MONTH", "DAY_OF_WEEK", "AIRLINE", "ORIGIN", "ORIGIN_STATE_ABR"
, "DEST", "DEST_STATE_ABR", "CANCELLED", "DIVERTED", "CARRIER_DELAY", "WEATHE
R_DELAY", "NAS_DELAY", "SECURITY_DELAY", "LATE_AIRCRAFT_DELAY")

# converting categorical columns to factor type
df.clean <- df.clean %>% mutate_if(names(df.clean) %in% fconvert, as.factor)

# vector of numerical variables to convert to numeric type
nconvert <- c("CRS_DEP_TIME", "DEP_TIME", "CRS_ARR_TIME", "ARR_TIME")

# converting numerical columns to numeric type
df.clean <- df.clean %>% mutate_if(names(df.clean) %in% nconvert, as.numeric)
```

Final Cleaning

Some final revisions to the dataset are made, including dropping newly-determined negligible variables, renaming variables for clarity, and further type conversion. Mainly, CANCELLED was determined to be dropped because cancelled and delayed flights are not inclusive. Meaning, if a flight is cancelled then it has no information about delay. Vice versa, if it was delayed, it wasn't cancelled. Therefore, the CANCELLED variable is unnecessary. Similar logic was applied to ARRIVAL-related variables, which cannot predict delay because they happen *after* delayed flights occur in a timeseries.

```
# dropping CANCELLED variable
df.clean <- df.clean[!(df.clean$CANCELLED == 1),]
df.clean$CANCELLED <- NULL

# renaming ORIGIN variable to ORIGIN_AIRPORT for clarity
colnames(df.clean)[colnames(df.clean) == 'ORIGIN'] <- 'ORIGIN_AIRPORT'
```

```r
# rename target variable to STATUS
colnames(df.clean)[colnames(df.clean) == 'DEP_DEL15'] <- 'STATUS'

# rename levels of STATUS to 1 = delayed and 0 = ontime
df.clean <- df.clean %>% mutate(STATUS = ifelse(STATUS == 1, "delayed", "onti
me"))

# reclassify STATUS to factor
df.clean$STATUS <- as.factor(df.clean$STATUS)

# arrival information no longer matters, as target of prediction is a delay i
n departing flights
# thus, arrival variables are dropped
df.clean$ARR_DELAY <- NULL
df.clean$ARR_DEL15 <- NULL
df.clean$DEST <- NULL
df.clean$DEST_STATE_ABR <- NULL

# variable DATE is dropped because we have variables related to day of week a
nd month; avoiding redundnacy
df.clean$DATE <- NULL

# variables DEP_DELAY is dropped because delay is accounted for by STATUS cla
ssification
df.clean$DEP_DELAY <- NULL

# variable DISTANCE is dropped because distance travelled occurs after a flig
ht has been deemed delayed or not and
# thus, isn't causation for delay
df.clean$DISTANCE <- NULL

# rename ORIGIN_STATE_ABR for clarity
colnames(df.clean)[colnames(df.clean) == 'ORIGIN_STATE_ABR'] <- 'ORIGIN_STATE
'
```

Final Summarized Overview

```
skim(df.clean)
```

Data summary

Name	df.clean
Number of rows	6227446
Number of columns	12
_______________________	
Column type frequency:	
factor	12
________________________	
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
MONTH	0	1	FALSE	12	8: 647163, 7: 646101, 10: 630842, 6: 623464
DAY_OF_WEEK	0	1	FALSE	7	1: 943989, 5: 925015, 4: 913586, 2: 904816
AIRLINE	0	1	FALSE	20	WN: 1108842, DL: 832494, AA: 772811, OO: 688077
ORIGIN_AIRPORT	0	1	FALSE	360	ATL: 335147, ORD: 279707, DFW: 250297, DEN: 213503
ORIGIN_STATE	0	1	FALSE	52	CA: 676800, TX: 659626, FL: 494119, IL: 364855

STATUS	0	1	FALSE	2	ont: 5045728, del: 1181718
DIVERTED	0	1	FALSE	2	0: 6210623, 1: 16823
CARRIER_DELAY	0	1	FALSE	2	0: 5640279, 1: 587167
WEATHER_DELAY	0	1	FALSE	2	0: 6156310, 1: 71136
NAS_DELAY	0	1	FALSE	2	0: 5582640, 1: 644806
SECURITY_DELAY	0	1	FALSE	2	0: 6223538, 1: 3908
LATE_AIRCRAFT_DELAY	0	1	FALSE	2	0: 5598157, 1: 629289

We summarize the data one final time to check and make sure all our cleaning/processing efforts were appropriately applied. Now, there remain **12 variables**, one of which is our prediction target, **STATUS** (flight status delayed/ontime). Further, all remaining key variables are categorical and will make good features in a classification model.

4. ANALYSIS & MODELING STRATEGY

We must now analysis our final dataset and define our modeling strategy. We start by making a judgement call; before devoting hours of computational resources to a full-fledged prediction model on all 7 million+ observations in this dataset, we instead opt to focus on a subset of the dataset to initially deploy our model. This subset consists of **50,000** randomly sampled observations from our cleaned flights dataset.

Sample Full Dataset

```
# set the random seed for reproducibility
set.seed(62695)

# create subset large enough for effecctive results yet small enough for comp
utational success on laptop
df.subset <- sample_n(df.clean, 50000)

# check balance of target variable values
table(df.subset$STATUS)

##
## delayed  ontime
##    9290   40710
```

It is clear from the table output above that the proportion of delayed to ontime STATUS cases is imbalanced. Our prediction model will require a more balanced set so it doesn't introduce *bias* on the disproportionately higher class while predicting. We balance the data below.

Balance Subset

```
# balancing dataset to equal number of results in target variable
index <- sample(length(df.clean$STATUS[df.clean$STATUS == "ontime"]),
                length(df.clean$STATUS[df.clean$STATUS == "delayed"]))

df.filtered0 <- df.clean %>% filter(STATUS == "ontime")

df.filtered1 <- df.clean %>%  filter(STATUS == "delayed")

df.filtered0 <- df.filtered0[index,]

df.subset <-  as.data.frame(rbind(df.filtered0, df.filtered1))

table(df.subset$STATUS)

##
## delayed  ontime
## 1181718 1181718
```

Re-Sample Full Dataset

```
df.subset <- sample_n(df.subset, 50000)

# confirm balance in target variable values
table(df.subset$STATUS)

##
## delayed  ontime
##   25098   24902
```

It’s now clear that our random sample is balanced in terms of our target variable, STATUS.

Preserve Sample Set

As a last analysis step, we preserve our sample dataset by writing it to a csv file. This way, we can access a fresh copy of it should something go wrong and we need to restart our model.

```
# write final dataset to csv for future use
write.csv(df.subset, "flights-df.csv", row.names = FALSE)
```

Dummy Variable Encoding

Alternatively, we perform dummy-variable encoding on the variables in the dataset that have more than 32 categorical levels. This is because some prediction modeling packages cannot handle more than 32 levels per factor when running their respective prediction algorithms. We encode the necessary variables as binary categories and save this subset to a csv file for use later.

```
# dummy variable encoding for factors with >= 32 levels because randomForest
cannot handle more levels per variable than that
df.coded <- dummy.data.frame(df.subset, names = c("ORIGIN_AIRPORT","ORIGIN_ST
ATE") , sep = "_")

## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts = F
ALSE):
## non-list contrasts argument ignored

## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts = F
ALSE):
## non-list contrasts argument ignored

# write encoded dataset to csv for future use
write.csv(df.coded, "flight-df-encoded.csv", row.names = FALSE)
```

Now, we have two data subsets, both of the same random sampling of 50,000 observations from the master set, both saved as csv files, and one that has been encoded with dummy variables for an alternate approach at modeling.

Modeling Strategy

Finally, comes a discussion of our modeling strategy. We considered, and attempted, **randomForest, regression, and naiveBayes** modeling approaches to predict flight delay classification on our dataset. RandomForest ended up being insufficient for our needs; given the prevalence of categorical variables in our dataset it continually made too many splits between binary categories across all variables and the prediction results were inconclusive at best. Regression had its own downside; the variables were determined to be linearly inseparable and the prediction results faltered because of it. After contending with these drawbacks, we decided to use Naive Bayes as our machine learning method of choice. Naive Bayes is fine-tuned to handle large amounts of categorical variables to predict classifications on a binary response variable. Our dataset fits this description perfectly. Following this description are the design and results from our successful Naive Bayes model.

5. MODEL PERFORMANCE ESTIMATION

Below, we create a random partition of our prediction dataset; 70% for training and the remaining 30% for testing. We then design a naiveBayes model that learns from the training partition and then predicts the classifications of the test set. Finally, we display some metrics that allow us to gauge how our model performed.

Partition Data

```
df.index <- createDataPartition(df.subset$STATUS,
                                p=0.7,
                                list = FALSE)

train <- df.subset[df.index,]

test <- df.subset[-df.index,]
```

Build naiveBayes Model

```
# build model on training data
nb.model <- naiveBayes(STATUS ~ ., data = train)

# predict on test data
pred <- predict(nb.model, test)
pred.raw <- predict(nb.model, test, type = "raw")
```

Performance Metrics

Counts Table

```
table(pred, test$STATUS)

##
## pred      delayed ontime
##   delayed    5992    343
##   ontime     1537   7127
```

The above table shows the number of actual vs. predicted classifications in both categories of the target variable, STATUS.

Proportions Table

```
prop.table(table(pred, test$STATUS))

##
## pred         delayed     ontime
##   delayed 0.39949330 0.02286819
##   ontime  0.10247350 0.47516501
```

The above table shows the proportion of actual vs. predicted classifications in both categories.

Confusion Matrix & Metrics

```
(perf.metric<- confusionMatrix(pred, test$STATUS))

## Confusion Matrix and Statistics
##
##             Reference
## Prediction delayed ontime
##    delayed    5992    343
##    ontime     1537   7127
##
##                Accuracy : 0.8747
##                  95% CI : (0.8693, 0.8799)
##     No Information Rate : 0.502
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7495
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7959
##             Specificity : 0.9541
##          Pos Pred Value : 0.9459
##          Neg Pred Value : 0.8226
##              Prevalence : 0.5020
##          Detection Rate : 0.3995
##    Detection Prevalence : 0.4224
##       Balanced Accuracy : 0.8750
##
##        'Positive' Class : delayed
##
```

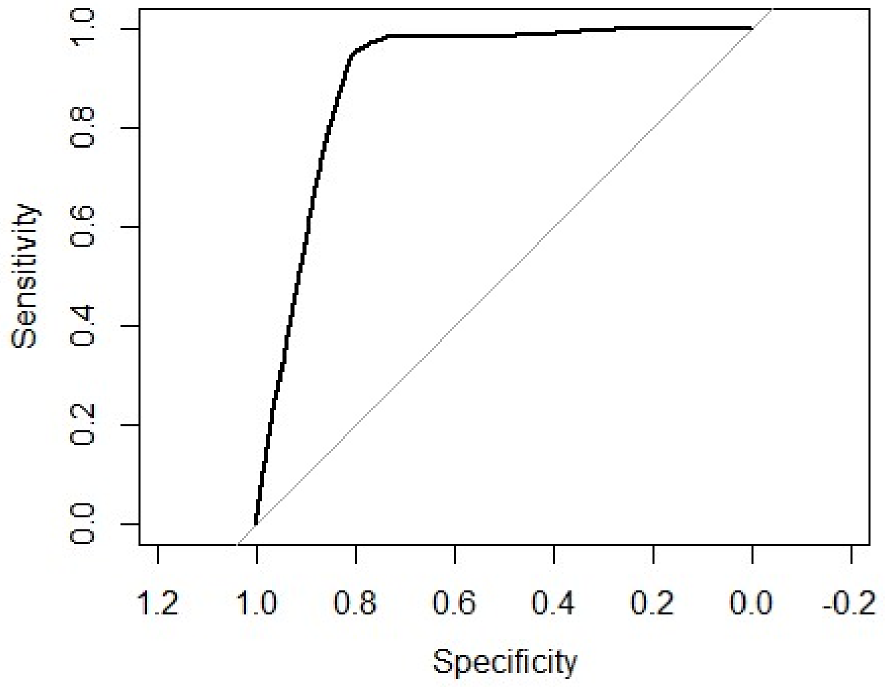
The above figure shows a confusion matrix and relevant statistics pertaining to the metrics of our naiveBayes model. We learn from this that the overall **Accuracy of prediction is 87.25%**. We can also see that both the positive and negative prediction rates are above **80%**, with the positive rate being at **94%**! These are great metrics for a model which only makes up for 50,000 subset of the original 7 million observations. Finally, we look at a ROC plot to view the AUC value of our model.

ROC Plot

```
ROCplot <- plot.roc(test$STATUS, pred.raw[,2])

## Setting levels: control = delayed, case = ontime

## Setting direction: controls < cases
```



```
ROC <- roc(test$STATUS, pred.raw[,2])

## Setting levels: control = delayed, case = ontime

## Setting direction: controls < cases

ROC

##
## Call:
## roc.default(response = test$STATUS, predictor = pred.raw[, 2])
##
## Data: pred.raw[, 2] in 7529 controls (test$STATUS delayed) < 7470 cases (t
est$STATUS ontime).
## Area under the curve: 0.9007
```

The above figure shows both an ROC plot and an output of its metrics. We can see that the curve has much vertical lift in comparison with the baseline. Further, we see a **AUC value of 90.26%**, which confirms our success in accurately predicting on this flight dataset.

6. INSIGHTS & CONCLUSIONS

Overall, the model has performed above our expectations which can be seen by the aforementioned metrics. Given that this is only a 50,000 sample set of the original 7 million observations of flight delay from 2019, the upside of this Business Analytics team is promising. If this were a real team for a real airline, we could take these results to our upper management next and present them with the successful findings; that we were able to accurately predict flight delay on un-seen data. These decision-makers could then instruct us to use this model on the master flights dataset for 2019 - maybe even the last decade - to further fine-tune and improve the performane of the naiveBayes model. Armed with this ever-improving model, the analytics department of this major airline would be able to save money and improve customer experience by accurately predicting flight delay and using that information to decrease flight delay in the future.

NOTE: We enjoyed working on this case and simulating what it would be like to work at a real company that makes real-world, important decisions with the information provided. Thank you for the opportunity, and we look forward to using these necessary skills in our future careers. - **Group 4**