Project 1:
Analyzing airline data using Classification and Prediction Methods

STAT 6620: Statistical Learning Using R

Abrar Alshehry
Li Qianmin
Rakesh Sunkari

Part 1: Importing the data

The "Airline on-time performance" dataset is donated by U.S. Department of Transportation's (DOT) Bureau of Transportation Statistics (BTS).We can download the data from

http://stat-computing.org/dataexpo/2009/the-data.html.

We are going to use the data of year 1987 collected from web. The dataset is in the form of ".bz2". So, we used 7zip application to unzip the dataset and imported in R. After using the str function in R to understand the structure of the data. We got 1311826 observations with 29 variables as follows:

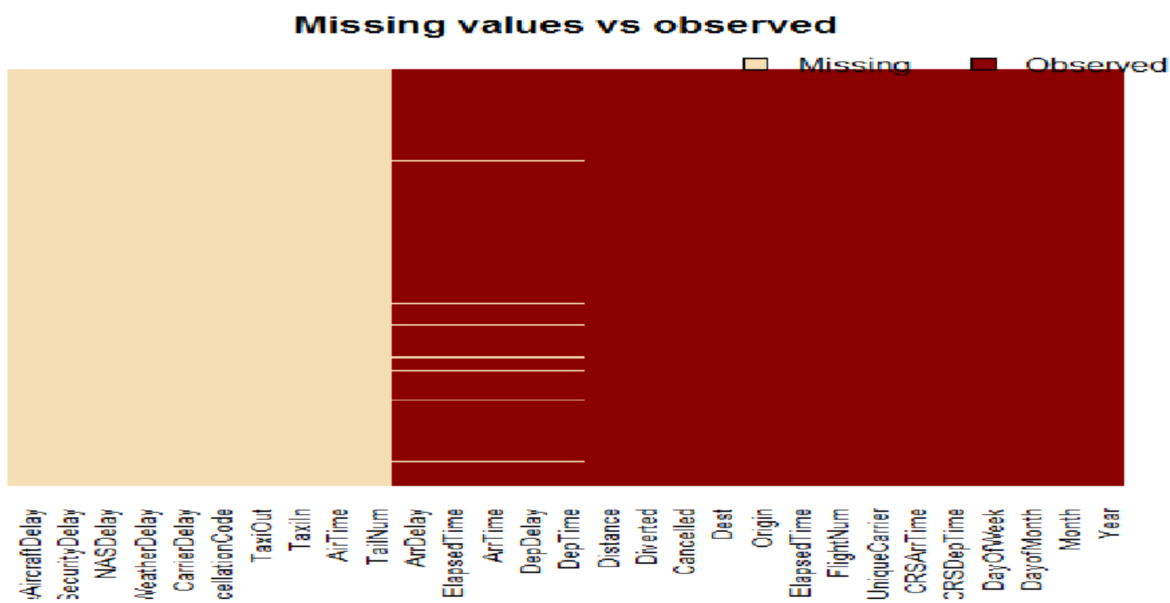
#	*Features*	*Type*	*Description*
1	Year	Integer	Year of the flight
2	Month	Integer	Month of the flight
3	DayofMonth	Integer	Day of the month
4	DayOfWeek	Integer	Day of the week
5	DepTime	Integer	Actual departure time
6	CRSDepTime	Integer	Scheduled departure time
7	ArrTime	Integer	Actual arrival time
8	CRSArrTime	Integer	Scheduled arrival time
9	UniqueCarrier	Factor (14 levels)	Airline
10	FlightNum	Integer	Flight number
11	TailNum	-	NA
12	ActualElapsedTime	Integer	Actual flight duration
13	CRSElapsedTime	Integer	Scheduled duration of the flight
14	AirTime	-	NA
15	ArrDelay	Integer	Arrival delay
16	DepDelay	Integer	Departure delay
17	Origin	Factor (237 levels)	City of departure
18	Dest	Factor (237 levels)	City of destination
19	Distance	Integer	Distance of the flight
20	TaxiIn	-	NA
21	TaxiOut	-	NA
22	Cancelled	0 or 1	Cancelled flight
23	CancellationCode	-	NA
24	Diverted	0 or 1	Diverted flight
25	CarrierDelay	-	NA
26	WeatherDelay	-	NA
27	NASDelay	-	NA
28	SecurityDelay	-	NA
29	LateAircraftDelay	-	NA

Part 2: Summarizing the data

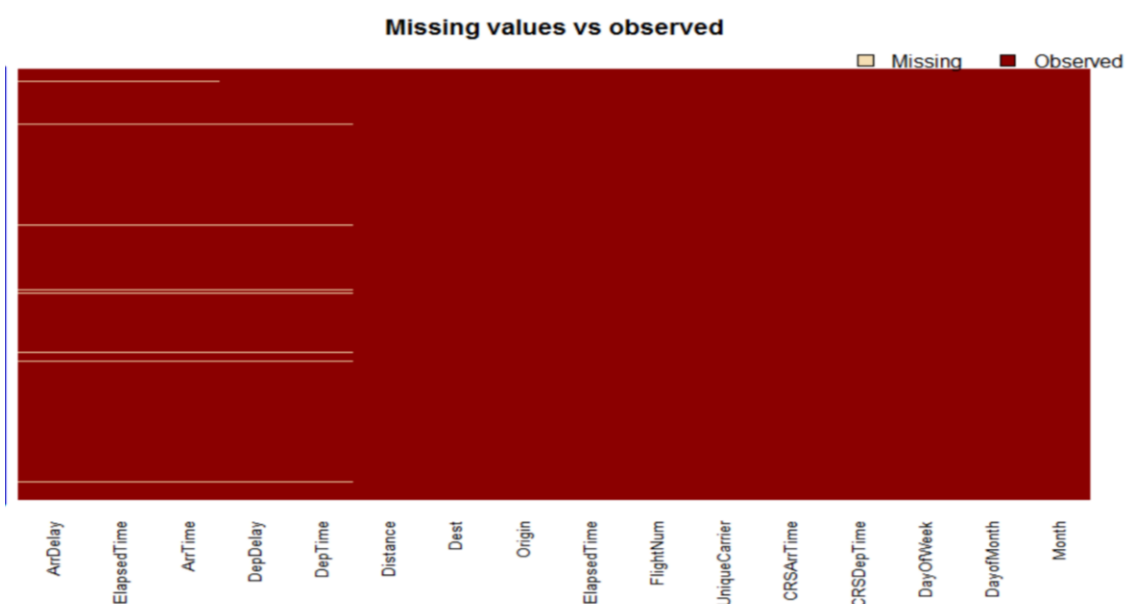
The above dataset contains 1,311,826 observations, and 29 variables. Out of 29 variables 11 are numeric features are *Departure Time, CRS Departure Time, Arrival Time, CRS Arrival Time , Actual Elapsed Time, CRS Elapsed Time, Arrival Delay, Departure Delay, Distance, Cancelled, Diverted* and 5 are characteristic variables are *Year, Month, Day of Month, Day of Week, and Flight Number* and the remaining 13 are logic variables which are eliminated because they contain only missing values. Since these missing columns are not useful for the analysis we are going to list out all those missing values and remove them using subset function in R

The following plot will visualize our data with the missing columns “the color yellow shows the missing values”, and the second plot show the data after removing the empty columns.

Plot (1): shows the entire data set



Plot (2) : shows the data after removing the empty columns



From the second plot we can see that we only have a few missing values. To calculate the missing values we used the following codes:

```
> #select the complete raws only
> complete.cases(airline)
> newdata <- airline[complete.cases(airline), ]
> str(newdata)
```

The airline dataset (with the missing values) has 1311826 observations. The new dataset (a dataset that has no missing values) has 1287333 observations. By comparing the two datasets we conclude that we have 24493 missing values out of 1,311,826 Since we have a large data set , we will simply remove the missing values.

Make tables of counts and relative frequencies for the Categorical features for each month of 1987. First, We will check for mean and standard deviation by month with respect to numerical and categorical variables:

```
> #mean and sd by month
> aggregate(data=newdata,cbind(DepTime,CRSDepTime,ArrTime, CRSArrTime,ActualElapsedTime,CRSElapsedTime,ArrDelay,DepDelay,Distance)~Month , mean)

  Month  DepTime CRSDepTime  ArrTime CRSArrTime ActualElapsedTime CRSElapsedTime  ArrDelay  DepDelay Distance
1    10 1365.294   1360.061 1493.254   1490.705          100.7280       99.58537  6.001447  5.010683 588.2018
2    11 1369.344   1361.655 1493.709   1491.679          102.1832      100.67574  8.467922  7.046977 590.9774
3    12 1373.230   1361.730 1492.643   1491.379          103.7413      101.67741 13.986140 11.945839 594.7546

> aggregate(data=newdata,cbind(DepTime,CRSDepTime,ArrTime, CRSArrTime,ActualElapsedTime,CRSElapsedTime,ArrDelay,DepDelay,Distance)~Month , sd)
  Month  DepTime CRSDepTime  ArrTime CRSArrTime ActualElapsedTime CRSElapsedTime ArrDelay DepDelay Distance
1    10 475.4528   471.7325 492.0934   484.9926          60.86199       60.65157 18.55076 19.56673 496.4846
2    11 477.4736   471.2423 497.4717   486.0279          61.53808       61.25396 24.44286 20.28832 497.9100
3    12 482.9949   472.6391 506.9758   488.6284          63.09214       61.73426 32.16863 29.31844 500.3926
```

From the above output/results, we can say that the mean values are close to each month and standard deviations are not changed. We are going to create crosstables for the categorical variables for each month of 1987.Cross tables are used to calculate relative frequencies. Here are the R codes for cross table for catagorical variables but we will only present some of them in this analysis.

```
library(gmodels)
CrossTable(y=newdata$Month, x= newdata$DayofMonth, prop.chisq=FALSE)
CrossTable(y= newdata$Month, x= newdata$DayOfWeek, prop.chisq=FALSE)
CrossTable(y= newdata$Month, x= newdata$UniqueCarrier, prop.chisq=FALSE)
CrossTable(y= newdata$Month, x= newdata$FlightNum, prop.chisq=FALSE)
CrossTable(y= newdata$Month, x= newdata$Origin, prop.chisq=FALSE)
CrossTable(y= newdata$Month, x= newdata$Dest, prop.chisq=FALSE)
```

- Cross table for ‘Day of Week’ by month.

```
> CrossTable(y= newdata$Month, x= newdata$DayOfWeek, prop.chisq=FALSE)
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|
Total Observations in Table:  1287333
                  | newdata$Month
newdata$DayOfWeek |        10 |        11 |        12 | Row Total |
```

```
-------------------|-----------|-----------|-----------|-----------|
                 1 |     58407 |     72066 |     54996 |    185469 |
                   |     0.315 |     0.389 |     0.297 |     0.144 |
                   |     0.131 |     0.173 |     0.129 |           |
                   |     0.045 |     0.056 |     0.043 |           |
-------------------|-----------|-----------|-----------|-----------|
                 2 |     58636 |     57771 |     69071 |    185478 |
                   |     0.316 |     0.311 |     0.372 |     0.144 |
                   |     0.132 |     0.139 |     0.162 |           |
                   |     0.046 |     0.045 |     0.054 |           |
-------------------|-----------|-----------|-----------|-----------|
                 3 |     58556 |     57338 |     71080 |    186974 |
                   |     0.313 |     0.307 |     0.380 |     0.145 |
                   |     0.132 |     0.138 |     0.167 |           |
                   |     0.045 |     0.045 |     0.055 |           |
-------------------|-----------|-----------|-----------|-----------|
                 4 |     73368 |     54575 |     70232 |    198175 |
                   |     0.370 |     0.275 |     0.354 |     0.154 |
                   |     0.165 |     0.131 |     0.165 |           |
                   |     0.057 |     0.042 |     0.055 |           |
-------------------|-----------|-----------|-----------|-----------|
                 5 |     73230 |     54130 |     54949 |    182309 |
                   |     0.402 |     0.297 |     0.301 |     0.142 |
                   |     0.165 |     0.130 |     0.129 |           |
                   |     0.057 |     0.042 |     0.043 |           |
-------------------|-----------|-----------|-----------|-----------|
                 6 |     66602 |     52226 |     52558 |    171386 |
                   |     0.389 |     0.305 |     0.307 |     0.133 |
                   |     0.150 |     0.125 |     0.123 |           |
                   |     0.052 |     0.041 |     0.041 |           |
-------------------|-----------|-----------|-----------|-----------|
                 7 |     55658 |     68077 |     53807 |    177542 |
                   |     0.313 |     0.383 |     0.303 |     0.138 |
                   |     0.125 |     0.164 |     0.126 |           |
                   |     0.043 |     0.053 |     0.042 |           |
-------------------|-----------|-----------|-----------|-----------|
      Column Total |    444457 |    416183 |    426693 |   1287333 |
                   |     0.345 |     0.323 |     0.331 |           |
-------------------|-----------|-----------|-----------|-----------|
```

- Cross tables for ‘Airline name’ by ‘month’.

```
> CrossTable(y= newdata$Month, x= newdata$UniqueCarrier, prop.chisq=FALSE)
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|
Total Observations in Table:  1287333
                      | newdata$Month
newdata$UniqueCarrier |        10 |        11 |        12 | Row Total |
----------------------|-----------|-----------|-----------|-----------|
                   AA |     55752 |     52705 |     54152 |    162609 |
                      |     0.343 |     0.324 |     0.333 |     0.126 |
                      |     0.125 |     0.127 |     0.127 |           |
                      |     0.043 |     0.041 |     0.042 |           |
----------------------|-----------|-----------|-----------|-----------|
                   AS |      7002 |      6586 |      6560 |     20148 |
                      |     0.348 |     0.327 |     0.326 |     0.016 |
                      |     0.016 |     0.016 |     0.015 |           |
                      |     0.005 |     0.005 |     0.005 |           |
```

CO	42480	38489	39009	119978
	0.354	0.321	0.325	0.093
	0.096	0.092	0.091	
	0.033	0.030	0.030	
DL	62735	59197	61707	183639
	0.342	0.322	0.336	0.143
	0.141	0.142	0.145	
	0.049	0.046	0.048	
EA	36207	33794	35563	105564
	0.343	0.320	0.337	0.082
	0.081	0.081	0.083	
	0.028	0.026	0.028	
HP	14860	14761	15160	44781
	0.332	0.330	0.339	0.035
	0.033	0.035	0.036	
	0.012	0.011	0.012	
NW	37107	33860	35121	106088
	0.350	0.319	0.331	0.082
	0.083	0.081	0.082	
	0.029	0.026	0.027	
PA (1)	5078	5505	5962	16545
	0.307	0.333	0.360	0.013
	0.011	0.013	0.014	
	0.004	0.004	0.005	
PI	38860	37204	39046	115110
	0.338	0.323	0.339	0.089
	0.087	0.089	0.092	
	0.030	0.029	0.030	
PS	14210	13355	13147	40712
	0.349	0.328	0.323	0.032
	0.032	0.032	0.031	
	0.011	0.010	0.010	
TW	23701	21924	22546	68171
	0.348	0.322	0.331	0.053
	0.053	0.053	0.053	
	0.018	0.017	0.018	
UA	52626	47994	48125	148745
	0.354	0.323	0.324	0.116
	0.118	0.115	0.113	
	0.041	0.037	0.037	
US	32129	30665	31105	93899
	0.342	0.327	0.331	0.073
	0.072	0.074	0.073	
	0.025	0.024	0.024	
WN	21710	20144	19490	61344
	0.354	0.328	0.318	0.048
	0.049	0.048	0.046	
	0.017	0.016	0.015	
Column Total	444457	416183	426693	1287333
	0.345	0.323	0.331	

Part 3: Create a variable ArrivedLate

We have to create a variable for arrived late. We will consider any flight delayed more than 15 minutes as arrived late flight.

```
> #create a new variable that indicated that the flight arrived late.
> #Any flight delayed more than 15 minutes classified as arrived late
> newdata$arrivelate=ifelse(newdata$ArrDelay>=15,1,0)
```

Another function is the, verify that the time delay calculations are correct. The Time Delay function compares the difference between the scheduled arrival time and the real arrival time, to the Arrival delay value. If these values match the conclusion will be (True), and if there is a different the conclusion will be (False).

```
# Using a function in R, verify that the time delay calculations are correct.
timeDelay <- function(x=ArrTime,y=CRSArrTime,z=ArrDelay) {if ((x-y)==z) return(TRUE) else{ return(FAL
SE)}
```

Part 4: Classification: Building the kNN Model

With our dataset examined and cleaned, we then split the data into training and testing sets. We randomly sampled 100k rows of the data to be used in training, and 10k rows to be used for testing. We then built a kNN classifier to predict which flights would be late, with the expectation that this classifier may not perform as well as the C5.0 Decision Tree since it must ignore the categoric data (like city of departure, origin, flight number, etc). Discarding these features loses some of the model's predictive power. For the kNN model, we selected 4 numeric features from our remaining dataset to be included (CRSDepTime, CRSArrTime, Distance, and DepDelay). We then applied min-max normalization to each of the variables in our model.

We tried several different values of k in our kNN model (k = 5,10,17,23,25). Accuracy ranged from 82.8% for k=35 to 85.4% for k=5. We were quite happy with the results, considering the fact that we do not have weather data to learn from and that our categorical features were left out of the model, this algorithm performed better than we had anticipated. The confusion matrix for our kNN model with k=5 is shown below. We can also see that we have a True Positive Rate of 11.5% with this model.

Pred / Actual	No	Yes
No	7394	287
No	0.739	0.029
———	———	———
Yes	1174	1145
Yes	0.117	0.115

Code:

```
# This step removes all remaining rows with NA values

air.df <- newdata

summary(air.df)

# KNN Model

# Create data frame based on the numeric columns to include as outlined in the report

kNN.df <- air.df[,c(5,7,13,16,17)]

dim(kNN.df)

#	Normalize the airline data

kNN.df.n <- as.data.frame(lapply(kNN.df[1:4], normalize))
summary(kNN.df.n)

#	setting seed

set.seed(12345)

#	Split the data frames into 100k for training, 10k for testing

sample(nrow(kNN.df.n), size=110000, replace=F)

kNN.rand.n <- kNN.df.n[rand.rows]

kNN.train <- kNN.rand.n[1:100000]

kNN.test<- kNN.rand.n[100001:100 ]

#	Create labels for training and testing the data

kNN.test.pred3 <- knn(train = kNN.train, test = kNN.test, cl = kNN.train.labels, k=35)

kNN.test.pred6 <- knn(train = kNN.train, test = kNN.test, cl = kNN.train.labels, k=5)

#	check the proportion of class variable

prop.table(table(kNN.train.labels))

## kNN.train     .labels

##      No      Yes

## 0.77237      0.22763

prop.table(table(kNN.test.labels))

## kNN.train     .labels

##      No      Yes

## 0.7672      0.2328
```

Training a model on the data

```
kNN.train.labels <- kNN.df[rand.rows[1:100000], 5]

kNN.test.labels <- kNN.df[rand.rows[100001:110000], 5]
```

Evaluating model performance

Create the cross tabulation of predicted vs. actual

```
CrossTable(x = kNN.test.labels, y = kNN.test.pred3, prop.chisq=FALSE)
```

```
## |  -------------------------  |
##
##
## Total Observations in Table:   10000
##
##
##                    | kNN.test.pred3
## kNN.test.labels |              No |         Yes | Row Total |
                    | ----------- |---------- | ----------- |
##    No            | 7644        |        28 | 7672        |
##                  | 0.996       |     0.004 | 0.767       |
##                  | 0.818       |     0.043 |             |
##                  | 0.764       |     0.003 |             |
                    | ----------- |---------- | ----------- |
##              Yes | 1701        |       627 | 2328        |
##                  | 0.731       |     0.269 | 0.233       |
##                  | 0.182       |     0.957 |             |
##                  | 0.170       |     0.063 |             |
                    | ----------- |---------- | ----------- |
```

```
CrossTable(x = kNN.test.labels, y = kNN.test.pred6, prop.chisq=FALSE)
```

```
## |  -------------------------  |
##
##
## Total Observations in Table:   10000
##
##
##                    | kNN.test.pred6
## kNN.test.labels |              No |         Yes | Row Total |
                    | ----------- |---------- | ----------- |
##    No            | 7386        |       286 | 7672        |
##                  | 0.963       |     0.037 | 0.767       |
##                  | 0.862       |     0.199 |             |
##                  | 0.739       |     0.029 |             |
                    | ----------- |---------- | ----------- |
##    Yes |           1179        |      1149 | 2328        |
##                  | 0.506       |     0.494 | 0.233       |
##                  | 0.138       |     0.801 |             |
##                  | 0.118       |     0.115 |             |
                    | ----------- |---------- | ----------- |
##    Column Total |  8565        |      1435 | 10000       |
##                  | 0.857       |     0.143 |             |
                    | ----------- |---------- | ----------- |
##
##
```

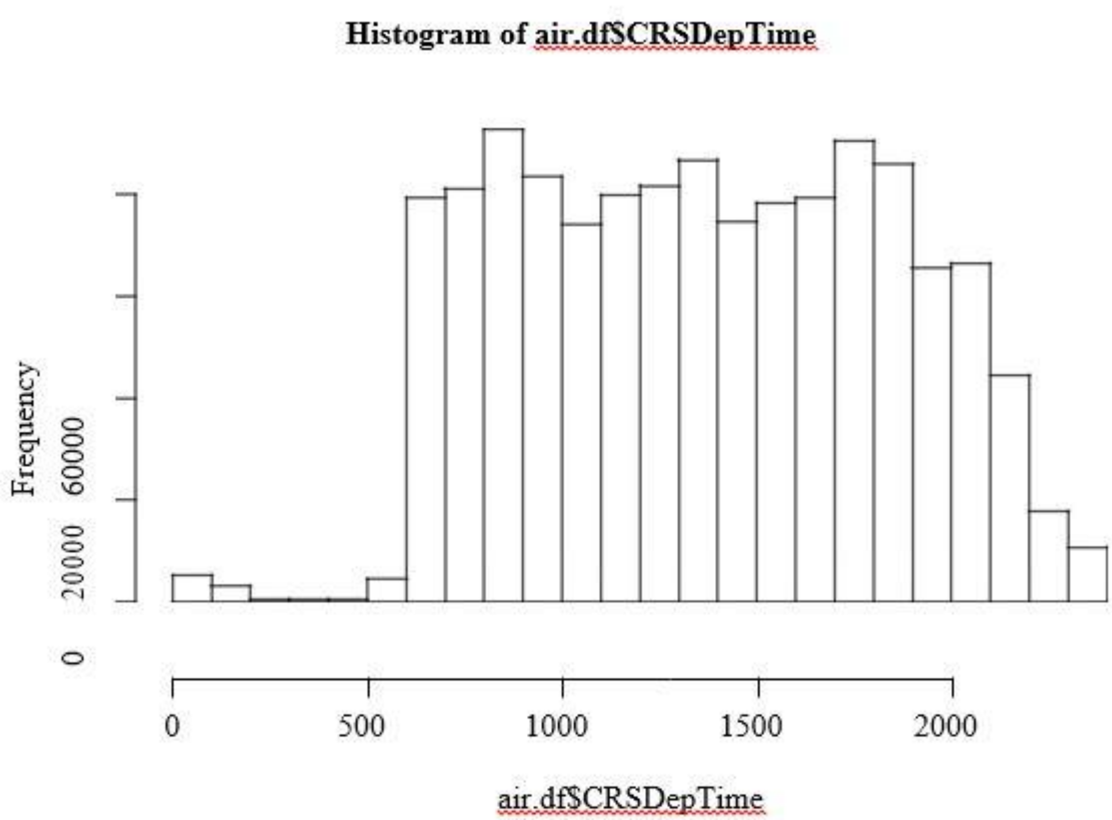
Part 5: Improving the Model

Next, we attempt to improve our classifier by using a C5.0 Decision Tree instead of a kNN classifier. The C5.0 tree is capable of using both numeric and categorical data as inputs, so we will take advantage of this to see if we can improve our predictions. We also created a few new features as described below.

We created a 'Holiday Flag' function to flag days which are close to the Thanksgiving and Christmas holidays as these are peak flying times, and may have an impact on late arrivals. We also examined the distribution of flight times to see binning them into categories makes sense. Given the shape of the histogram below, it's clear that we can group our flight departure and arrival times into 4 different categories. We found that we were able to improve our results by placing all time-based data into Morning, Afternoon, Evening, and Red-eye time bins as follows:

Time	Time Bin
600 - 1100	Morning
1100 - 1600	Afternoon
1600 - 2100	Evening
2100 - 600	Red-eye

hist(air.df$CRSDepTime)



Next, we built our C5.0 boosting tree classifier with the number of trials = 20. We trained the model and validated the results on the testing data. We then generated a confusion matrix to examine the results. As we can see below, our model's predictions have improved quite a bit. We went from having 85.4% accuracy with the kNN classifier to 88.9% accuracy with the C5.0 boosting tree with the holiday flag and the departure and arrival time bins as surrogates for actual departure and arrival times. We also improved the True Positive Rate (TPR) from 11.8% to 14.0%. We were very happy with these results.

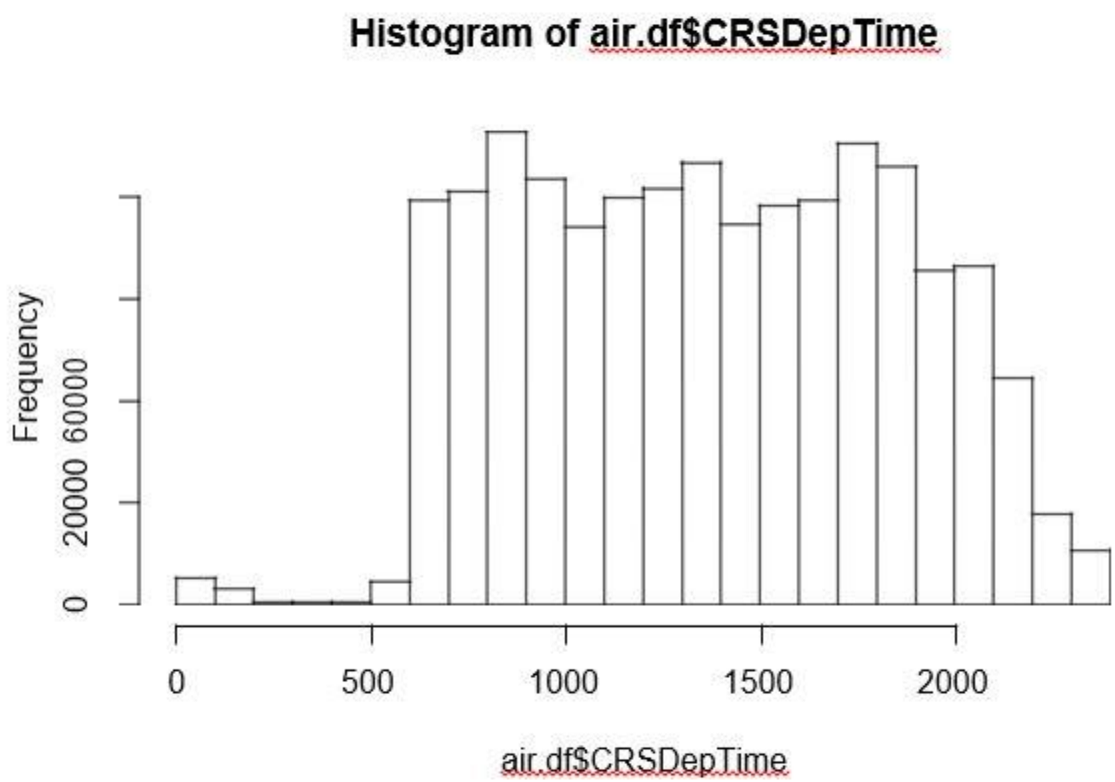
Pred / Actual	No	Yes
No	7491	231
No	0.749	0.023
Yes	877	1401
Yes	0.088	0.140

Classifier	Accuracy	TPR
kNN	85.4%	11.8%
C5.0	88.9%	14.0%

Code:

```
#  C5.0 Decision Tree #

# Create normalization function - using min-max normalization normalize <- function(x) {return ((x - min(x))
/ (max(x) - min(x)))}

#  Create function to flag late flights
late <- function(x) {if (x > 15) return (1) else
return (0)}

#  Create function to flag holiday flights for Thanksgiving and Christmas holiday <-
function(x,y) {if ((x == "Nov" & is.element(y,c(25,26,27,28,29))) | (x == "Dec" &
is.element(y,c(23, return (1) else return (0)}
#  Create binning function for times of day - use with categorical models

timebin<-function(x){ if (x >=
   600 & x < 1100) return
   ("Morning")

else if (x >= 1100 & x < 1600)
   return ("Afternoon")

else if (x >= 1600 & x < 2100) return
("Evening")

else return ("RedEye")}
```

```
# Create function for time delay verification td <-
function(x,y,z) {

if ((y-x) == z) return (TRUE) else
return (FALSE)

}

## Improving the Model With Decision Trees

# Add Holiday Flag

air.df$Holiday <- mapply(holiday, air.df$Month, air.df$DayofMonth)

air.df$Holiday <- factor(air.df$Holiday, levels = c("0","1"), labels = c("No","Yes"))

str(air.df)

#  Create bins for times of day

#         Examine distribution of departure times to select bins

hist(air.df$CRSDepTime)
```




```
# Add binned CRSDepTime

air.df$DepBin <- sapply(air.df$CRSDepTime, timebin)

air.df$DepBin <- factor(air.df$DepBin)

# Add binned CRSArrTime

air.df$ArrBin <- sapply(air.df$CRSArrTime, timebin)
```

```
air.df$ArrBin <- factor(air.df$ArrBin)

summary(air.df$DepBin)

## Afternoon   Evening   Morning    RedEye
##    399307    394293    404966     88767

summary(air.df$ArrBin)

## Afternoon   Evening   Morning    RedEye
##    400861    412992    273745    199735

# We will use the following features for our C5.0 tree: "Month","DayofMonth","DayOfWee

# "UniqueCarrier","FlightNum","DepDelay","Origin","Dest","Distance","Holiday","DepBin"

C50.df <- air.df[,c(1,2,3,8,9,13,14,15,16,18,19,20,17)]

summary(C50.df)

##   Month           DayofMonth        DayOfWeek     UniqueCarrier
##   Oct:444457     2  :  43674      Mon:185469    DL     :183639
##   Nov:416183     9  :  43663      Tue:185478    AA     :162609
##   Dec:426693     23 :  43373      Wed:186974    UA     :148745
##                  30 :  43061      Thu:198175    CO     :119978
##                  22 :  42953      Fri:182309    PI     :115110
##                  6  :  42947      Sat:171386    NW     :106088
##               (Other):1027662     Sun:177542    (Other):451164
##  FlightNum          DepDelay              Origin           Dest
##   539    :  1961    Min.   :-1345.000    ORD    : 65288    ORD    : 66011
##   416    :  1901    1st Qu.:    0.000    ATL    : 65116    ATL    : 65559
##   85     :  1861    Median :    0.000    DFW    : 51287    DFW    : 51923
##   202    :  1816    Mean   :    7.968    LAX    : 44718    LAX    : 44664
##   309    :  1793    3rd Qu.:    8.000    DEN    : 41863    DEN    : 42653
##   537    :  1779    Max.   : 1439.000    SFO    : 34387    SFO    : 34108
##  (Other):1276222                         (Other):984674    (Other):982415
##  Distance          Holiday        DepBin              ArrBin
##   Min.   :  10.0   No :1104482   Afternoon:399307    Afternoon:400861
##   1st Qu.: 247.0   Yes: 182851    Evening  :394293    Evening  :412992
##   Median : 416.0                  Morning  :404966    Morning  :273745
##   Mean   : 591.3                  RedEye   : 88767    RedEye   :199735
##   3rd Qu.: 787.0
##   Max.   :4983.
##
##     Late
##   No :995608
##   Yes:291725

dim(C50.df)

## [1] 1287333    13
```

```
# Split the data frames into 100k for training, 10k for testing
# kNN.rand.n <- kNN.df.n[order(runif(nrow(kNN.df.n))), ]
rand.rows2 <- sample(nrow(C50.df), size=110000, replace=F)
C50.rand <- C50.df[rand.rows2]
C50.train <- C50.rand[1:100000]
C50.test <- C50.rand[100001:110000]
# Create labels for training and testing the data
C50.train.labels <- C50.df[rand.rows2[1:100000], 13]
C50.test.labels <- C50.df[rand.rows2[100001:110000], 13]
# check the proportion of class variable
prop.table(table(C50.train$Late))
##    No      Yes
## 0.77339   0.22661
prop.table(table(C50.test$Late))
##    No      Yes
##  0.7746    0.2254
# Try with Flight Numbers
C50.m1 <- C5.0(C50.train[,c(1:12)], C50.train$Late, trials = 20)
summary(C50.m1)
C50.pred1 <- predict(C50.m1, C50.test)
# cross tabulation of predicted versus actual classes
CrossTable(C50.test$Late, C50.pred1, prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE, dnn = c('actual default', 'predicted default'))
```

```
## Total Observations in Table:   10000
##
##
##    | predicted default
## actual default |              No |        Yes | Row Total |
   |-----------                     | -----------   | -----------|
##    No |                     6466  | 1280          | 7746 |
##    |                        0.647 | 0.128         | |
   |-----------                     | -----------   | -----------|
##    Yes |                    1858  | 396           | 2254 |
##    |                        0.186 | 0.040         | |
   |-----------                     | -----------   | -----------|
##    Column Total |           8324  | 1676          | 10000 |
   |-----------                     | -----------   | -----------|
##
```

```
# Try 2nd model without the flight numbers

C50.df2 <- air.df[,c(1,2,3,8,13,14,15,16,18,19,20,17)]

# summary(C50.df2)

# dim(C50.df2)

# Split the data frames into 100k for training, 10k for testing

rand.rows3 <- sample(nrow(C50.df2), size=110000, replace=F)

C50.rand2 <- C50.df2[rand.rows3]

C50.train2 <- C50.rand2[1:100000]

C50.test2 <- C50.rand2[100001:110000]

# Create labels for training and testing the data

C50.train.labels2 <- C50.df2[rand.rows3[1:100000], 12]

C50.test.labels2 <- C50.df2[rand.rows3[100001:110000], 12]

# check the proportion of class variable

prop.table(table(C50.train2$Late))

prop.table(table(C50.test2$Late))

# train model without flight numbers

C50.m2 <- C5.0(C50.train2[,c(1:11)], C50.train2$Late, trials = 20)

# C50.m2

# summary(C50.m2)

C50.pred2 <- predict(C50.m2, C50.test2)

# cross tabulation of predicted versus actual classes

CrossTable(C50.test2$Late, C50.pred2, prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE, dnn =
c('actual default', 'predicted default'))
```

```
## Total Observations in Table:   10000
##
##
##    | predicted default
## actual default |    No |    Yes | Row Total |
     |----------     | ----------    | ----------|
##   No |    7501    | 247    | 7748 |
##   |    0.750    | 0.025    | |
     |----------     | ----------    | ----------|
##   Yes |    888    | 1364    | 2252 |
##   |    0.089    | 0.136    | |
     |----------     | ----------    | ----------|
##   Column Total |    8389    | 1611    | 10000 |
     |----------     | ----------    | ----------|
##
```

Summary

Both of the models we built to classify flights as either 'late' and 'on-time' performed quite well. We approached the problem from two different ways, first using only numeric features with a kNN algorithm, and second using both numeric and categorical features with a C5.0 Boosted Decision Tree. The kNN algorithm yielded an accuracy rate of 85.4%, while the C5.0 Tree outperformed the kNN model by achieving 88.9% accuracy. Our model could be improved if we were able to incorporate weather data, which we did not have in this dataset. We were able to think a bit outside the box, by creating a 'Holiday Flag', and time bins as features for our C5.0 Tree. These proved to be useful additions to the model's predictive power. We also tried predicting late flights with and without flight numbers in the models. While our trees looked quite different, the model's performance was quite similar with and without them yielding 88.7% and 88.9% accuracy respectively. With the kNN model, we were able to capture an 11.8% True Positive Rate, and with the C5.0 model we captured a 14.0% TPR.

Appendix:

R-code

```
mydata <- read.csv("1987.csv", stringsAsFactors = TRUE)
str(`mydata`)
#takeout the columns with missing data or uniqe data
airline<- subset(mydata,select = -
c(Year,TaxiIn,TaxiOut,Cancelled,CancellationCode,Diverted,CarrierDelay,WeatherDelay,NASDelay,Security
Delay,LateAircraftDelay,TailNum,AirTime,Cancelled))
str(airline)

#select the complete raws only

complete.cases(airline)
newdata <- airline[complete.cases(airline), ]
# the number of obs was 1,311,826 and the new number is 1,287,333 so we only had 24,493 missing values in
the data set

#The new data set contains 1,287,333 examples and 15 features: 13 numeric variables and 3 categorical
variables
str(newdata)

#mean and sd by month
aggregate(data=newdata,cbind(DepTime,CRSDepTime,ArrTime,
CRSArrTime,ActualElapsedTime,CRSElapsedTime,ArrDelay,DepDelay,Distance)~Month , mean)
aggregate(data=newdata,cbind(DepTime,CRSDepTime,ArrTime,
CRSArrTime,ActualElapsedTime,CRSElapsedTime,ArrDelay,DepDelay,Distance)~Month , sd)

# cross table for catagorical variables

library(gmodels)
CrossTable(y=newdata$Month, x= newdata$DayofMonth, prop.chisq=FALSE)
CrossTable(y= newdata$Month, x= newdata$DayOfWeek, prop.chisq=FALSE)
        CrossTable(y= newdata$Month, x= newdata$UniqueCarrier, prop.chisq=FALSE)
        CrossTable(y= newdata$Month, x= newdata$FlightNum, prop.chisq=FALSE)
        CrossTable(y= newdata$Month, x= newdata$Origin, prop.chisq=FALSE)
```

```
CrossTable(y= newdata$Month, x= newdata$Dest, prop.chisq=FALSE)

#create a new variable that indicated that the flight arrived late.
#Any flight delayed more than 15 minutes classified as arrived late
newdata$arrivelate=ifelse(newdata$ArrDelay>=15,1,0)

# Using a function in R, verify that the time delay calculations are correct.
timeDelay <- function(x=ArrTime,y=CRSArrTime,z=ArrDelay) {
  if ((x-y)==z) return(TRUE) else{ return(FALSE)}
  library(C50)
  library(class)
  # Create normalization function - using min-max normalization
  normalize <- function(x) {

    return ((x - min(x)) / (max(x) - min(x)))

  }

  #  Create function to flag late flights
  late <- function(x) {

    if (x > 15) return (1) else return (0)
  }

  #	Create function to flag holiday flights for Thanksgiving and Christmas
  holiday <- function(x,y)
  {

    if ((x = "Nov" & is.element(y,c(25,26,27,28,29))) | (x == "Dec" & is.element(y,c(23, return (1)

                                                                                    else return (0)

  }

  #	Create binning function for times of day - use with categorical models

  timebin <- function(x){ if (x >= 600 & x < 1100)

    return ("Morning")

    else if (x >= 1100 & x < 1600) return ("Afternoon")

    else if (x >= 1600 & x < 2100) return ("Evening")

    else return ("RedEye")

  }

  # Create function for time delay verification
  td <- function(x,y,z) {

    if ((y-x) == z) return (TRUE) else return (FALSE)
```

```
}
#	This removes all remaining rows with NA values

air.df <- newdata[complete.cases(newdata),]

summary(air.df)



# KNN Model
#  Create data frame based on the numeric columns to include as outlined in the report

#	Note: must choose between 'Distance' and 'CRSElapsedTime' as they are co-surrogates
kNN.df <- air.df[,c(5,7,13,16,17)]

dim(kNN.df)

#	Normalize the airline data

kNN.df.n <- as.data.frame(lapply(kNN.df[1:4], normalize))
summary(kNN.df.n)

#	setting  seed
set.seed(12345)

#	Split the data frames into 100k for training, 10k for testing
rand.rows <- sample(nrow(kNN.df.n), size=110000, replace=F)
kNN.rand.n <- kNN.df.n[rand.rows,]
kNN.train <- kNN.rand.n[1:100000, ]
kNN.test <- kNN.rand.n[100001:110000, ]

#	Create labels for training and testing the data
kNN.train.labels <- kNN.df[rand.rows[1:100000], 5]
kNN.test.labels <- kNN.df[rand.rows[100001:110000], 5]

#	check the proportion of class variable

prop.table(table(kNN.train.labels))

prop.table(table(kNN.test.labels))

#  Training a model on the data

kNN.test.pred3 <- knn(train = kNN.train, test = kNN.test, cl = kNN.train.labels, k=35),
kNN.test.pred6 <- knn(train = kNN.train, test = kNN.test, cl = kNN.train.labels, k=5),

## Evaluating model performance

# Create the cross tabulation of predicted vs. actual

CrossTable(x = kNN.test.labels, y = kNN.test.pred3, prop.chisq=FALSE),
```

```
CrossTable(x = kNN.test.labels, y = kNN.test.pred6, prop.chisq=FALSE),

#  C5.0 Decision Tree

## Improving the Model With Decision Trees

# Add Holiday Flag

air.df$Holiday <- mapply(holiday, air.df$Month, air.df$DayofMonth)

air.df$Holiday <- factor(air.df$Holiday, levels = c("0","1"), labels = c("No","Yes")) str(air.df)
# sanity check 2.0

#  Create bins for times of day


#	Examine distribution of departure times to select bins
hist(air.df$CRSDepTime)

# Add binned CRSDepTime

air.df$DepBin <- sapply(air.df$CRSDepTime, timebin)
air.df$DepBin <- factor(air.df$DepBin)

# Add binned CRSArrTime
air.df$ArrBin <- sapply(air.df$CRSArrTime, timebin) air.df$ArrBin <- factor(air.df$ArrBin)

summary(air.df$DepBin)

summary(air.df$ArrBin)

#	We will use the following features for our C5.0 tree: "Month","DayofMonth","DayOfWee

#	"UniqueCarrier","FlightNum","DepDelay","Origin","Dest","Distance","Holiday","DepBin"

C50.df <- air.df[,c(1,2,3,8,9,13,14,15,16,18,19,20,17)]
summary(C50.df)

dim(C50.df)

rand.rows2 <- sample(nrow(C50.df), size=110000, replace=F)
C50.rand <- C50.df[rand.rows2]
C50.train <- C50.rand[1:100000]
C50.test <- C50.rand[100001:110000]

#	Create labels for training and testing the data

C50.train.labels <- C50.df[rand.rows2[1:100000], 13]
C50.test.labels <- C50.df[rand.rows2[100001:110000], 13]

#	check the proportion of class variable
```

```
prop.table(table(C50.train$Late))

prop.table(table(C50.test$Late))

# Try with Flight Numbers
C50.m1 <-     C5.0(C50.train[,c(1:12)], C50.train$Late, trials = 20)
C50.m1
summary(C50.m1)

C50.pred1 <- predict(C50.m1, C50.test)
# cross tabulation of predicted versus actual classes

CrossTable(C50.test$Late, C50.pred1,
        prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
        dnn = c('actual default', 'predicted default'))
# Try 2nd model without the flight numbers

C50.df2 <- air.df[,c(1,2,3,8,13,14,15,16,18,19,20,17)]

#       summary(C50.df2)

#       dim(C50.df2)

#       Split the data frames into 100k for training, 10k for testing rand.rows3 <-
sample(nrow(C50.df2), size=110000, replace=F) C50.rand2 <- C50.df2[rand.rows3,]
C50.train2 <- C50.rand2[1:100000]
C50.test2 <- C50.rand2[100001:110000]

#       Create labels for training and testing the data

C50.train.labels2 <- C50.df2[rand.rows3[1:100000], 12]
C50.test.labels2 <- C50.df2[rand.rows3[100001:110000], 12]

#       check the proportion of class variable
prop.table(table(C50.train2$Late))
prop.table(table(C50.test2$Late))

#       train model without flight numbers

C50.m2 <- C5.0(C50.train2[,c(1:11)], C50.train2$Late, trials = 20)

#       C50.m2

#       summary(C50.m2)
C50.pred2 <- predict(C50.m2, C50.test2)
# cross tabulation of predicted versus actual classes

CrossTable(C50.test2$Late, C50.pred2,
        prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
        dnn = c('actual default', 'predicted default'))
```