Stat 652: Statistical Learning and Prediction

Brad McNeney

Final Project: Flight Delay Prediction

December 2019

Submitted by:

Saumya Dwivedi

301367213

Introduction:

Flight delays are a common problem which we all face on a regular basis. If there was a way by which we could predict the delay time of a flight the Airline companies could prepare in advance and make necessary preparations for an expected delay. We could also solve the problem where people miss important appointments due to flight delays by letting them know that there is a chance of a flight being delayed by specified time thereby allowing them to choose different flights or rescheduling their appointments. Fliers could use this information while booking flights based on the expected delay time of flights and their own schedule,
The objective of this project is to analyse data from three New York Airports for the year 2013 and build a model using that data to predict the flight delay time for the given test data.

Data:

We collected our data on flights from four datasets from the nycflights13 package.
The flight information is from three New York City airports in 2013.The entire training dataset made from these datasets is of the size 11,952 KB and is of the type csv.gz. Our complete training dataset has 200000 flight information(rows) and 43 features(columns)The four datasets which were used are:

Dataset	Information
flights	This dataset has information about the planned and actual flight schedules. It has variables such as:Date of departure, dep_time, arr_time, sched_arr_time, sched_dep_time, carrier, flight, tailnum, origin, destination, air time, distance and **dep_delay** which tells us the delay time in minutes and is our **response variable.**
weather	This dataset gives hourly information on the weather. In this dataset we have variables such as: origin, time of recording, temp, humidity, wind direction, wind speed, precip, pressure, visibility
airports	This dataset gives us the details of the airport. It has the following variables: faa, name, lat, lon. alt, tz, dst, tzone
planes	This dataset gives us information about the planes. The variables which this dataset has are:Tailnum, year, type, manufacturer, model, engine, seats, speed

We also used test data of the size 5,868 KB and 136776 records to test our model.

Methods:

EDA: Exploratory Data Analysis

We began our process by analysing our data to understand the distribution and any possible patterns.

Missing data:
The dataset has a lot of missing values. We measured the number of missing values in each column and removed columns which have too many missing values, which is 5% as per general rule of thumb which for our dataset is 10,000. All columns with more than 10,000 missing values were removed. We then removed all the rows with missing values.
After handling all the missing values, we were left with 184,316 records. (Check appendix section [3])

Response Variable:
As the next step we analysed the response variable dep_delay.The distribution of this variable is highly right skewed with very high max value. The number of flights with an average departure delay time more than 60 mins is 8%. The long right tail makes it a very uneven distribution and resulted in an ineffective model. To solve this problem, we scaled the response variable's quantiles to the quantiles of a normal distribution. (Check appendix section [7])
An analysis of response variable with respect to predictors was done. With respect to date we see that the relationship is non-linear. The delays are higher in summer, as compared to Fall. Not sure about winter. With respect to time we see that delays increase as the day passes by. We also see that delays increase with larger distances and with extreme temperatures. (Check appendix section[9])

Feature Engineering:
To build the best model we choose the following predictors in the required format:

- The year, month, day telling scheduled time of flight was converted to a date object.
- We remove dep_time, arr_time and arr_delay since we are using dep_delay as the response variable and these columns will be heavily associated with the response variable.
- We also remove sched_arr_time, tailnum, flight number since we can see that they are of low importance.
- We remove name since we already have the destination name called dest
- We remove air_time since it is highly correlated with distance
- We also replace the numeric precip column with a boolean column which indicates if there was any precipitation or not since the values are very small between 0 and 1 making the boolean indicator more useful.
- Upon analysing the relationship between the response variable and other predictors we also see that lat, lon and alt do not have a significant relationship with the response

variable therefore we take log(lon), log(lat) and log(alt) are taken as predictors.(Check appendix section[8])

Regression Models:

A number of regression models were tried to get the predicted delay departure time of flights. I started with linear regression, decision trees and GAM and GBM with GBM giving the best results therefore for this project I focus on GBM as the main algorithm to build the model.
The models which were used:

Multilinear Linear Regression:
Multiple linear regression is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is to model the linear relationship between the explanatory (independent) variables and the response variable. This model was used as a baseline model.

GAM - Generalised Additive Method
To improve the results achieved from the linear model we use GAM, GAM is effective when linear models are not. They are also called wriggly models. It is an improvement on linear generalised model and also has a smoothing function. GAM also showed the most important features and here lan, lon and alt were least important and were removed. Loss function used to measure the accuracy of this model showed considerable improvement from the linear model.

Random Forest
The random forest algorithm works by aggregating the predictions made by multiple decision trees of varying depth. Every decision tree in the forest is trained on a subset of the dataset called the bootstrapped dataset. Random Forest can be used for classification as well as regression and by selecting probability as true we used it to get the probability of flight's departure delay. In our case the random forest did give good results however the time taken was too long

GBM - Gradient Boosting Method
GBM is a very popular machine learning algorithm and it has proven to be successful across many domains.GBMs build an ensemble of weak successive trees with each tree learning and improving on the previous. Boosting here is done such that the weaker trees are given more weights are are included in the next successive sample. This ensemble of trees result in a powerful model. It is different and better than random forests in the sense that random forests build independent trees.
The main advantages of GBM are as follows:

- High accuracy
- It is flexible in the sense that we can tune several parameters such as the number of trees, samples to be included, the size of the tree.
- Works with both categorical and numerical features

For the models used, rmse was used as the loss function to calculate the accuracy of the models.
GBM gave the best results with respect to other models with GAM coming in second and polynomial regression proving to be the worst.
Random Forest also gave sufficiently accurate results however the run time was very high proving it to be ineffective.
As a result, I tuned GBM to get the best results. With higher number of trees, the results proved to be better. I started from 1000 trees which gave an mse value of 0.07, increasing it to 2000 reduced the mse value a little. Increasing the shrinkage value allowed for collecting more samples at each step but slowly training the model proved to be more effective.
(Check appendix section[12])

Results:

To test the model built we were given a test dataset which had 136776 records.
Similar feature engineering as performed on training data set were performed on the test dataset as well. The same columns removed,and log of lat, lon, alt taken.
These ensured that the test data set had the same format as the training dataset.
We then performed predictions for these test data using the models fitted on the above mentioned algorithms.
I started with linear model so as to get a baseline result. Loss function used to measure the accuracy of each of the model is RMSE. To increase the accuracy of the multiple linear regression I used GAM next to improve upon the linear model and as expected the accuracy of GAM is considerably better than the linear model.
Random Forest can also be used for our use case and I tried that next. Random Forest was expected to give good results but due to the large data size and the limited machine power I had the run time was very long(More than 60 mins) therefore I look to work with it in the future with a GPU to get faster results
Finally I used GBM to build the model. GBM has proven to be one of the most affecting boosting methods and as expected, it gave the best results. Tuning the GBM parameters to number of trees to 2000 and shrinkage = 0.01 gave the lowest RMSE values and thus GBM proved to be the best model.

Experiment Results:

Model	MSE	RMSE
Multiple regression	0.0723	0.268
GAM	0.0700	0.264
GBM	0.0699	0.264

Conclusion and Discussion:

In this paper, we analysed flight related data for three New York Airports in 2013 and used that data to build a model which can be used to predict the Deepdale of flights. I worked on building regression models which could then be used to predict the departure delay of flights given in the test dataset.

I began by analysing the data to observe the distribution of the response variable, which was heavily right tailed. To solve this issue, the quantiles of the response variable were scaled to quantiles of a normal curve. During the analysis many features proved to be unimportant with respect to the response variable and were thus removed.

To solve the problem of missing values we first removed the columns which had more than 5% missing null values which allowed us to work with features which did not have many missing values. We then removed the remaining rows which had null values.

A series of regression models were then built in order to get the result with the maximum accuracy. I started with a multiple linear model to get a baseline result and then proceeded to improve that by using GAM. Random Forest was also explored as a good option however it the computation time for random forest was very large. In the future I would use a GPU to get faster results and would also try to tune the model better. The last approach was GBM and it gave the best results. I was able to run GBM for 2000 trees which gave a considerably low RMSE.

For future work, with a GPU or a system with greater RAM I would like to further tune GBM for a larger number of trees which would give us even better results. I would also like to try out other regression models which may give better results such as ridge regression.

I believe without these hardware limitations I will be able to achieve a better accuracy for the predictions.

References:

- An Introduction to Statistical Learning with Applications in R by Gareth James ,Daniel Witten, Trevor Hastie and Robert Tibshirani
- Rdocumentation.org https://www.rdocumentation.org/packages/mgcv/versions/1.8-31/topicss/
- Towards Data Science- Random Forest in R https://towardsdatascience.com/random-forest-in-r-f66adf80ec9

Appendix:

R version used: 3.6.1

Packages Used:

- **tidyverse**
- **nycflights13**
- **lubridate**
- **gam**
- **gbm**
- **randomForest**

Code:

Loading Data[1]

```
library(tidyverse)
library(nycflights13)
library(lubridate)
library(gam)
library(gbm)
library(randomForest)
```

```
#help(flights)
#help(weather)
#help(airports)
#help(planes)
#fltrain            <-            read_csv("../../Project652/fltrain.csv.gz")

fltrain                      <-                    read_csv("fltrain.csv.gz")
fltest <- read_csv("fltest.csv.gz")
```

Data Dimensions[2]

Training Data:

```
fltrain
dim(fltrain)
```

Testing Data:

```
fltest
dim(fltest)
```

Function to encode columns as factor

```
fl                                  <-                                  fltrain
for(i                    in                    1:ncol(fl))                    {
  if(typeof(fl[[i]])              ==              "character")              {
    fl[[i]]                          <-                          factor(fl[[i]])
  }
}
```

Missing Values[3]

Finding the number of missing values for each column:

```
num_miss        <-        function(x)        {        sum(is.na(x))        }
sapply(fl,num_miss)
```

Removing columns with high mising variables:

```
fl                                    <-                                    fl%>%
  select(-year.y,-type,-manufacturer,-model,-engines,-seats,     -speed,     -
engine,-wind_gust,-pressure)
summary(fl)
```

Remving rows with missing variables after remving columns with very high null values

```
fl <- na.omit(fl)
summary(fl)
```

Response variable analysis[4]

Departure delays over the months

```
ggplot(fl, aes(x = factor(month), y = dep_delay)) +
  geom_boxplot()
```

Distribution of response variable - dep_delay [5]

```
range(fl$dep_delay, na.rm =TRUE)
fivenum(fl$dep_delay)
```

Histogram for distribution of dep_delay

```
ggplot(data = fl, aes(x = dep_delay)) +
  geom_histogram()
```

Grouping delays by carrier, airport, data[6]

Summaries of departure delay by NYC airport:

```
Q3 <- function(x) { quantile(x,probs=.75) }
fl %>% group_by(origin) %>%
  summarize(n=n(),med_d = median(dep_delay),Q3_d = Q3(dep_delay), max_d =
max(dep_delay)) %>%
  arrange(desc(Q3_d)) %>% head(10)
```

Summaries of departure delay by airline (carrier).

```
fl %>% group_by(carrier) %>%
  summarize(n=n(),med_d = median(dep_delay),Q3_d = Q3(dep_delay), max_d =
max(dep_delay)) %>%
  arrange(desc(Q3_d)) %>% head(10)
fl %>% group_by(origin,carrier) %>%
  summarize(n=n(),med_d = median(dep_delay),Q3_d = Q3(dep_delay), max_d =
max(dep_delay)) %>%
  arrange(desc(Q3_d)) %>% head(10)
```

```
fl                %>%                group_by(dest,carrier)                %>%
  summarize(n=n(),med_d  =  median(dep_delay),Q3_d  =  Q3(dep_delay),  max_d  =
max(dep_delay))                                                          %>%
  arrange(desc(Q3_d)) %>% head(10)

dl_aa_ua                      <-                      fl                      %>%
  filter(carrier   ==   "AA"   |   carrier   ==   "DL"   |   carrier   ==   "UA")


ggplot(dl_aa_ua,  aes(x  =  dep_delay,  y  =  arr_delay,  color  =  carrier))  +
        xlim(-25,                              100)                              +
        geom_point()
```

Summaries of departure delay by date:

```
fl                  %>%                  group_by(month,day)                %>%
  summarize(n=n(),med_d   =   mean(dep_delay),max_d   =   max(dep_delay))   %>%
  arrange(desc(med_d)) %>% head(10) # what happened on march 8?
```

Summaries of departure delay by precipitation:

```
fl  %>%  mutate(haveprecip  =  factor(precip>0))  %>%  group_by(haveprecip)  %>%
  summarize(n=n(),med_d  =  median(dep_delay),Q3_d  =  Q3(dep_delay),  max_d  =
max(dep_delay))                                                          %>%
  arrange(desc(med_d)) %>% head(10)
```

Scaling the response variable to lower extremes[7]

```
#fl     <-     fl     %>%     mutate(dep_delay     =     qqnorm(dep_delay)$x)
den                                  <-                                  nrow(fl)+1
fl     <-     fl     %>%     mutate(dep_delay     =     rank(dep_delay)/den)
 ggplot(fl,aes(x=dep_delay)) + geom_histogram(binwidth=.01)
```

Featureengineering[8]

```
library(lubridate)
fl                        <-                        fl                        %>%
  mutate(dep_date          =          make_date(year.x,month,day))          %>%
  select(-year.x,-month,-day,-dep_time,-arr_time,-arr_delay,
         -sched_arr_time,-tailnum,-flight,-name,-air_time,
         -hour,-minute,-time_hour,-tz,-dst,-tzone)                          %>%
  mutate(precip = as.numeric(precip>0))
```

Associations between dep_delay and other predictors[9]

```
ggplot(fl,aes(x=dep_date,y=dep_delay))      +      geom_point(alpha=.01)      +
geom_smooth()
ggplot(fl,aes(x=sched_dep_time,y=dep_delay))    +    geom_point(alpha=0.01)    +
geom_smooth()
ggplot(fl,aes(x=distance,y=dep_delay))      +      geom_point(alpha=0.01)      +
geom_smooth()
ggplot(fl,aes(x=log(distance),y=dep_delay))    +    geom_point(alpha=0.01)    +
geom_smooth()
fl   <-   mutate(fl,logdistance   =   log(distance))   %>%   select(-distance)
ggplot(fl,aes(x=temp,y=dep_delay)) + geom_point(alpha=0.01) + geom_smooth()
ggplot(fl,aes(x=dewp,y=dep_delay)) + geom_point(alpha=0.01) + geom_smooth()
fl <- mutate(fl,logalt = log(alt)) %>% select(-alt)

set.seed(123)
```

Formatting Testing Data[10]

```
fl_test                                  <-                                  fltest
for(i                  in                  1:ncol(fl_test))                  {
  if(typeof(fl_test[[i]])            ==            "character")            {
    fl_test[[i]]                  <-                  factor(fl_test[[i]])
  }
}
```

Number of missing values for each column:

```
num_miss_test       <-       function(x)       {       sum(is.na(x))       }
sapply(fl_test,num_miss_test)
```

Removing variables with high mising variables:

```
fl_test                                  <-                                  fl_test%>%
  select(-year.y,-type,-manufacturer,-model,-engines,-seats,     -speed,     -
engine,-wind_gust,-pressure)
summary(fl_test)
```

Remving rows with missing variables after remving columns with very high null values

```
fl_test                          <-                          na.omit(fl_test)
summary(fl_test)
```

```
den_test                          <-                          nrow(fl_test)+1
fl_test <- fl_test %>% mutate(dep_delay = rank(dep_delay)/den_test)

library(lubridate)
fl_test                   <-                   fl_test                   %>%
  mutate(dep_date         =         make_date(year.x,month,day))          %>%
  select(-year.x,-month,-day,-dep_time,-arr_time,-arr_delay,
         -sched_arr_time,-tailnum,-flight,-name,-air_time,
         -hour,-minute,-time_hour,-tz,-dst,-tzone)                        %>%
  mutate(precip = as.numeric(precip>0))

fl_test <- mutate(fl_test,logdistance = log(distance)) %>% select(-distance)
fl_test <- mutate(fl_test,logalt = log(alt)) %>% select(-alt)
```

Removing the 1 record with dep == LEX from the test data

```
fl_test                                 <-                                 fl
fl_test<-fl_test[!(fl_test$dest=="LEX"),]
```

Learning models

GAM[11]

```
library(gam)
form <- formula(dep_delay ~ s(dep_date) + s(sched_dep_time) + carrier +
origin          +          dest          +          s(logdistance)          +
                  s(temp) + s(dewp) + s(humid) + s(wind_dir) + s(wind_speed)
+                    precip                    +                    s(visib))
gam_fit            <-            gam(form,            data=fl,family=gaussian)
summary(gam_fit)
plot(gam_fit,se=TRUE)
gam_pred                  <-                predict(gam_fit,newdata=fl_test)
mse_gam               <-               mean((fl_test$dep_delay-gam_pred)^2)
mse_gam
rmse_gam                          <-                          sqrt(mse_gam)
rmse_gam
```

GBM[12]

```
library(gbm)
dep_date_numeric                  <-                  as.numeric(fl$dep_date)
```

```
dep_date_numeric <- dep_date_numeric - mean(dep_date_numeric)
fl_tr_tem <- mutate(fl,dep_date = dep_date_numeric)
gbm_fit <-gbm(dep_delay ~ .,data=fl_tr_tem,distribution="gaussian",
n.trees = 2000, shrinkage = 0.01)
summary(gbm_fit)
#
dep_date_numeric <- as.numeric(fl_test$dep_date)
dep_date_numeric <- dep_date_numeric - mean(dep_date_numeric)
fl_te_tem <- mutate(fl_test,dep_date = dep_date_numeric)
#
gbm_pred <- predict(gbm_fit,newdata=fl_te_tem,n.trees = 2000)
mse_gbm <- mean((fl_test$dep_delay-gbm_pred)^2)
mse_gbm
rmse_gbm <- sqrt(mse_gbm)
rmse_gbm
```

Multiple Linear Regression[13]

```
lr_fit <- lm(dep_delay ~ ., data = fl, family="binomial")
summary(lr_fit)
lr_pred <- predict(lr_fit, newdata = fl_test)
mse_pl <- mean((fl_test$dep_delay-lr_pred)^2)
mse_pl
rmse_pl <- sqrt(mse_pl)
rmse_pl
```

Random Forest[14]

```
library(randomForest)
rf_fit <- randomForest(dep_delay ~ dep_date + sched_dep_time + logdistance +
temp + dewp + humid+ wind_dir + wind_speed + precip + visib, data = fl,
prob=TRUE, importance = TRUE)
summary(rf_fit)
rf_pred <- predict(rf_fit, newdata = fl_test)
mse_rf <- mean((fl_test$dep_delay-rf_pred)^2)
mse_rf
rmse_rf <- sqrt(mse_rf)
rmse_rf
```