STAT 652 Project report
Rishabh Jain
2019-12-05

1 . Introduction

Every year in the United State of America, millions of passengers experience delay in flights, resulting in missing flight connections which squander the valuable time of passengers. Reports have shown that this is a serious problem not just for the passengers but also for the airlines and the US economy. Our main objective in this project will be to identify the factors which have the most influence on flight on-time performance.

This project intent to predict the departure delay time for flights departing from NYC based on the hourly meteorological data for each airport, construction information about each plane, airport locations and the Flight characteristics.

2. Data

The data used here contains information about all flights that departs from NYC (i.e. JFK, LGA or EWR) in 2013.This data is available as part of the nycflights13 R package. It includes information about planes, airports, weather and flights. The training dataset has 200000 rows with 43 features.

2.1 Data Cleansing

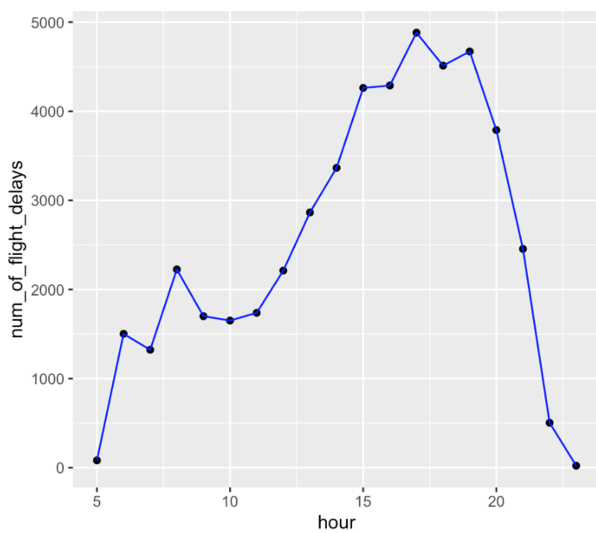
The dataset has a lot of missing values which can be dealt by either removing the records (complete case analysis) or by imputing data with mean/mode values. In this case we will first remove the irrelevant features and then omit the records with missing data. As a thumb rule we discard variables with more than 5% missing values, which is 10,000 of these data. Based on this we remove year.y,type,manufacturer,model,engines,seats,speed,engine,wind_gust,pressure columns.

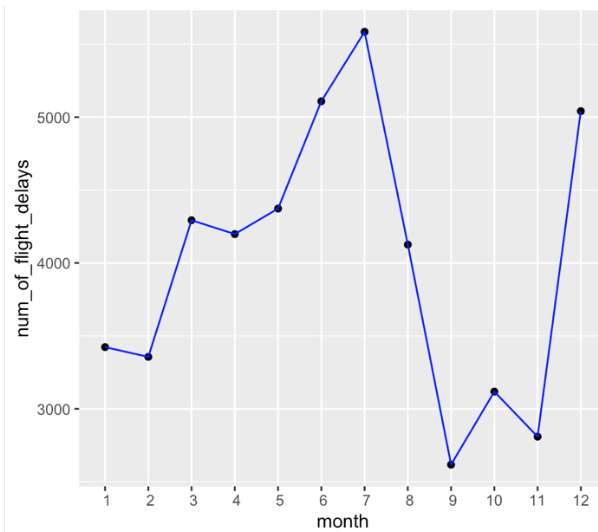
Few of the variables like dep_time, arr_time and arr_delay are not known to us when predicting departure delays, therefore we remove these from the dataset. Features like tailnum and flight (flight number) are not really needed for prediction of flight delay, air_time is highly correlated with distance , hour and minute are already considered in sched_dep_time, time_hour is same as sched_dep_time and tz ,dst, tzone are highly correlated with dest. Therefore, we remove these variables from our dataset.

```
> fivenum(fl$dep_delay)
[1]  -43    -5    -2    11 1301
```

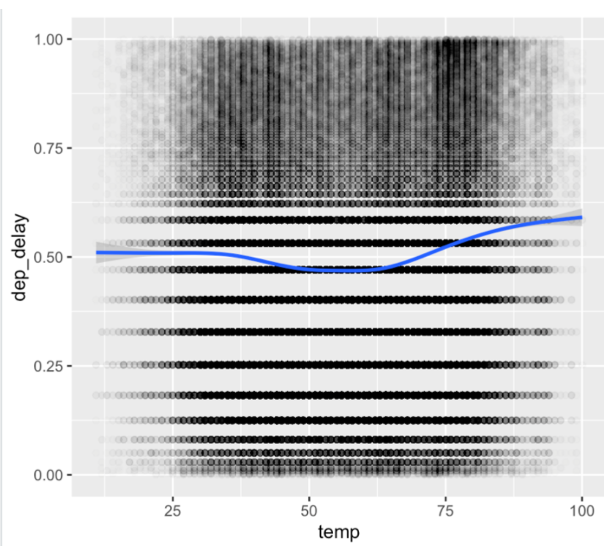
As we can see from above function that flights are normally on time with median of -2. Maximum delay of 1302 minutes shows that Departure delay is highly right-skewed. Therefore we standardize the column for better prediction accuracy. Ranks are scaled by n+1 to get the empirical quantiles, which will be comparable to those in the test set to get better predictions.

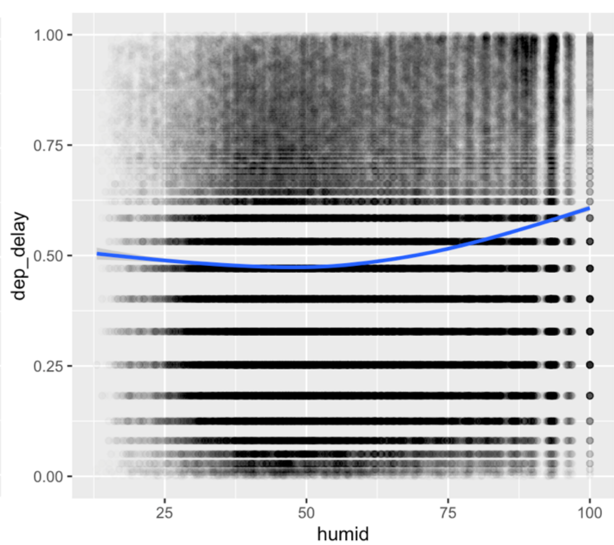
2.2 Data Exploration





The above left figure shows the distribution of flight delays grouped by hours in a day. We can see that the flights leaving around 16:00-20:00 gets delayed most often. The right figure shows the distribution of flight delays grouped by months. Flights departing in the month of July is most likely to get delayed probably because of summer holiday season. Both the plots were drawn with delays of more than 10 minutes. From the above two plots we can say that people traveling around evening in the month of July has high chance of facing departure delays.





Above plots compares temperature and dewpoint with departure delay and show signs of non linearity in data. We can observe a pattern of increase in Departure delays in lower and higher range of temperature and dewpoint. This also proves that weather plays an important role in predicting flight delays. Refer Appendix Part 2 for more Data exploration plots.

3. Methods

Before fitting the model, the dataset is split into training and validation sets in random fashion of ratio 70:30. Firstly we fit the model on train data and then use it on validation data to predict the validation error and tune the hyperparameters. Then we use the model on the provided test set to calculate the test error.

In order to predict the value of a continuous variable i.e. Departure delay we tried different regression methods. Here, we use xgboost method by tuning hyperparameters.

Extreme Gradient Boosting (xgboost) is similar to gradient boosting framework but more efficient. It has both linear model solver and tree learning algorithms. So, what makes it fast is its capacity to do parallel computation on a single machine. This makes xgboost at least 10 times faster than existing gradient boosting implementations. It supports various objective functions, including regression, classification and ranking. Below are the hyperparameters which was tuned using validation set to improve the performance of the model.

Eta[default = 0.3] - It controls the learning rate, i.e., the rate at which our model learns patterns in data. After every round, it shrinks the feature weights to reach the best optimum.
Nrounds[default = 100] - It controls the maximum number of iterations.
Max_Depth[default = 6] - It controls the depth of the tree. Larger the depth, more complex the model; higher chances of overfitting
The above parameters was tuned with the validation set and lowest error was noted in Eta = 0.1, nrounds=160 and max_depth = 9.

Below are some more models which was applied on this data.
GAM : A generalized additive model (GAM) is a generalized linear model in which the linear predictor is given by a user specified sum of smooth functions of the covariates plus a conventional parametric component of the linear predictor. This method was applied on the dataset.

GBM : It is an algorithm which converts a weak leaner to a strong learner, it uses multiple decision tree sequentially. Parameters were tuned and for ntree=2000 and shrinkage=0.01, the validation error was lowest.

Decision Tree : This is a non-parametric supervised learning method used for classification and regression. It models the conditional probability of the predictors given the response variable. Tree pruning was done for best=3 to obtain the lowest error.

Random Forest : Random forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. We were unable to perform this method because of huge volume of training data and low computational power.

Code for all the models described is attached in appendix below.

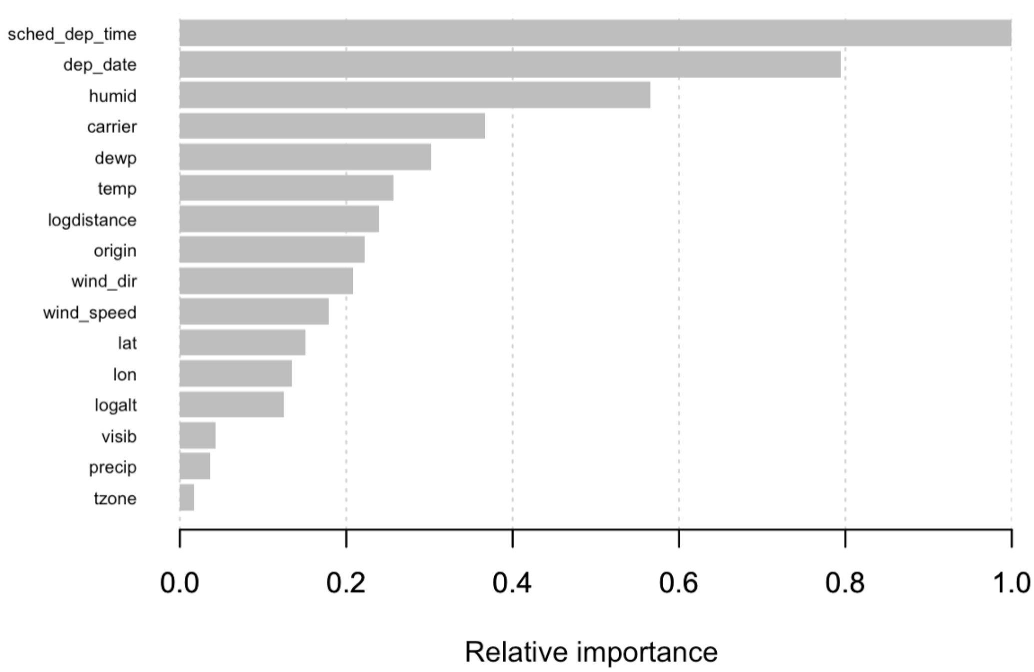
4. Result

Following table shows the model accuracy of different models :

Model	Validation Error	Test Error
XG Boost	0.06239184	0.06215232
Decision Tree	0.07628095	0.07586896
GAM	0.07066494	0.07037731
GBM	0.07068754	0.07074504

Comparing the models above we can see that XG Boost performs the best among all with the lowest test and validation error. Random forest was also used but because of computational inefficiency the code did not complete successfully.

Below figure show the relative feature importance for XG Boost method.



```
> importance_matrix
          Feature        Gain       Cover  Frequency
 1: sched_dep_time 0.219237246 0.140417285 0.16992707
 2:       dep_date 0.175056606 0.273803496 0.11980553
 3:          humid 0.121206297 0.108430669 0.10691052
 4:        carrier 0.081073645 0.102445703 0.06658178
 5:           dewp 0.065002274 0.051313098 0.06887371
 6:           temp 0.052896470 0.047657513 0.09760389
 7:    logdistance 0.051092053 0.049475574 0.05035305
 8:         origin 0.049894112 0.027843565 0.02062739
 9:       wind_dir 0.042867806 0.035678985 0.07213798
10:     wind_speed 0.036752271 0.034222662 0.06403519
11:            lat 0.030841612 0.035483522 0.05671953
12:            lon 0.028122667 0.019013481 0.04287533
13:         logalt 0.025452812 0.055369028 0.04002778
14:          visib 0.008965449 0.011535299 0.01458502
15:         precip 0.008154738 0.004678712 0.00391249
16:          tzone 0.003383941 0.002631407 0.00502373
```

The gain in the above chart implies the relative contribution of the corresponding feature to the model calculated by taking each feature's contribution for each tree in the model. A higher value of this metric when compared to another feature implies it is more important.
The Cover metric means the relative number of observations related to this feature.
The Frequency is the percentage representing the relative number of times a particular feature occurs in the trees of the model
Thus, the chart here shows that sched_dep_time is the most important feature in this method followed by others in descending order.

5. Conclusion and Discussion

The aim of this study was to construct a model that predicts the departure delay in all flights that departs from New York City. But with the data we have and this reliable analysis, it's not easy to predict the U.S flight delay of an unknown data. Modeling assumptions did not turn out to be accurate. For future work , we can try using Random forest with more computing power to get better results. Instead of doing complete case analysis we might try different methods of imputing the data to get better predictions.

Besides that, I also want to find different machine learning algorithms such as Neural Network to improve the prediction rates comparing to the models in this project.

Appendix

Software Version -> All analysis for this project was done on RStudio 3.6.1

OS -> Mac OS v 10.15

Part 1 : Data Loading and Cleansing

```
library(tidyverse)

library(nycflights13)
#install.packages("tree")
fltrain <- read_csv("fltrain.csv")
```

Converting Categorical variables into factors.

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

Count the missing values in each variable and removing variables with more than 10000 missing values.

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

##          year.x          month            day       dep_time sched_dep_time 
##               0              0              0           4898              0 
##       dep_delay       arr_time sched_arr_time      arr_delay        carrier 
##            4898           5169              0           5584              0 
##          flight        tailnum         origin           dest       air_time 
##               0           1492              0              0           5584 
##        distance           hour         minute      time_hour           temp 
##               0              0              0              0            948 
##            dewp          humid       wind_dir     wind_speed      wind_gust 
##             948            948           5862            982         152260 
##          precip       pressure          visib           name            lat 
##             937          23092            937           4484           4484 
##             lon            alt             tz            dst          tzone 
##            4484           4484           4484           4484           4484 
##          year.y           type   manufacturer          model        engines 
##           34298          31163          31163          31163          31163 
##           seats          speed         engine 
##           31163         199415          31163

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

Omitting the remaining rows

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

Some analysis on the predictor variable showing the data is highly right skewed.

```
range(fl$dep_delay)

## [1]  -43 1301

fivenum(fl$dep_delay)

## [1]  -43   -5   -2   11 1301

quantile(fl$dep_delay,probs = c(0.01,0.05,0.1,0.25,.5,.75,.90,.95,.99))

##  1%  5% 10% 25% 50% 75% 90% 95% 99%
## -12  -9  -7  -5  -2  11  49  88 193

mean(fl$dep_delay >= 60)

## [1] 0.08210356
```

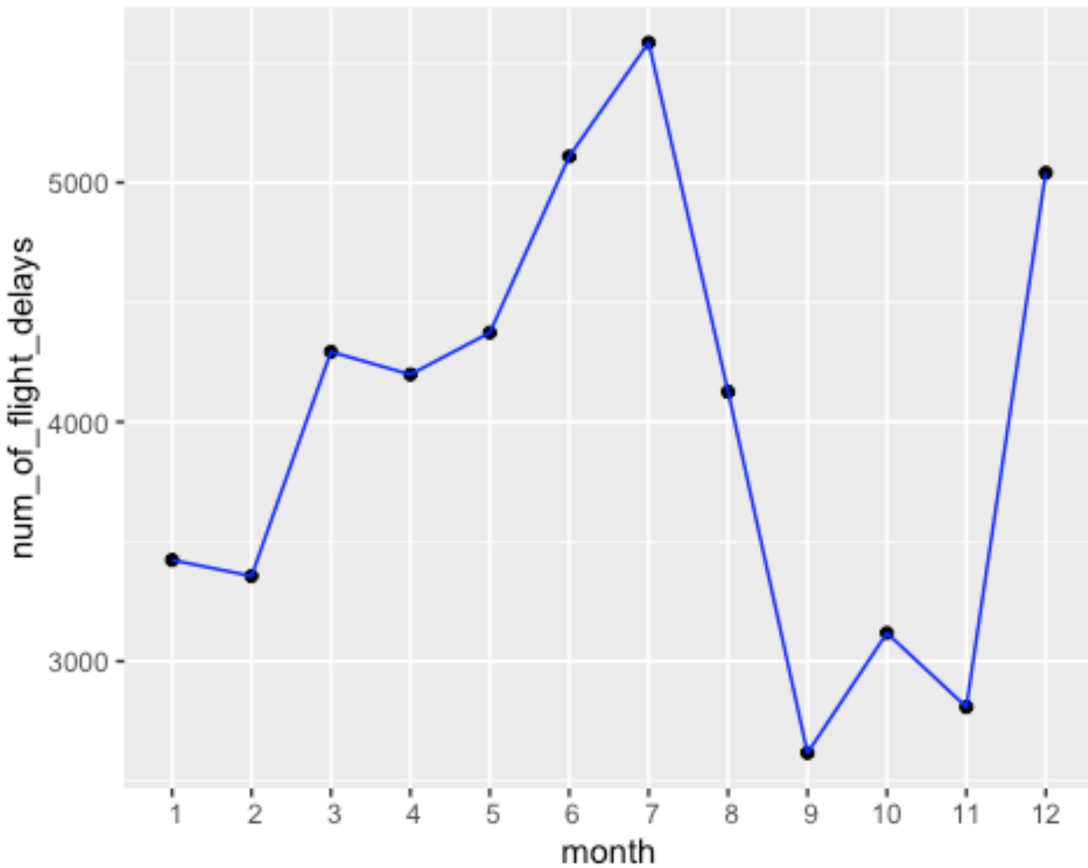
Top 10 delays.

```
fl%>% arrange(desc(dep_delay)) %>% head(10)

## # A tibble: 10 x 33
##    year.x month   day dep_time sched_dep_time dep_delay arr_time
##     <dbl> <dbl> <dbl>    <dbl>          <dbl>     <dbl>    <dbl>
##  1   2013     1     9      641            900      1301     1242
##  2   2013     9    20     1139           1845      1014     1457
##  3   2013     3    17     2321            810       911      135
##  4   2013     7    22     2257            759       898      121
##  5   2013    12     5      756           1700       896     1058
##  6   2013     5    19      713           1700       853     1007
##  7   2013     2    10     2243            830       853      100
##  8   2013    12    19      734           1725       849     1046
##  9   2013    12    17      705           1700       845     1026
## 10   2013    12    14      830           1845       825     1210
## # … with 26 more variables: sched_arr_time <dbl>, arr_delay <dbl>,
## #   carrier <fct>, flight <dbl>, tailnum <fct>, origin <fct>, dest <fct>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>, temp <dbl>, dewp <dbl>, humid <dbl>, wind_dir <dbl>,
## #   wind_speed <dbl>, precip <dbl>, visib <dbl>, name <fct>, lat <dbl>,
## #   lon <dbl>, alt <dbl>, tz <dbl>, dst <fct>, tzone <fct>
```

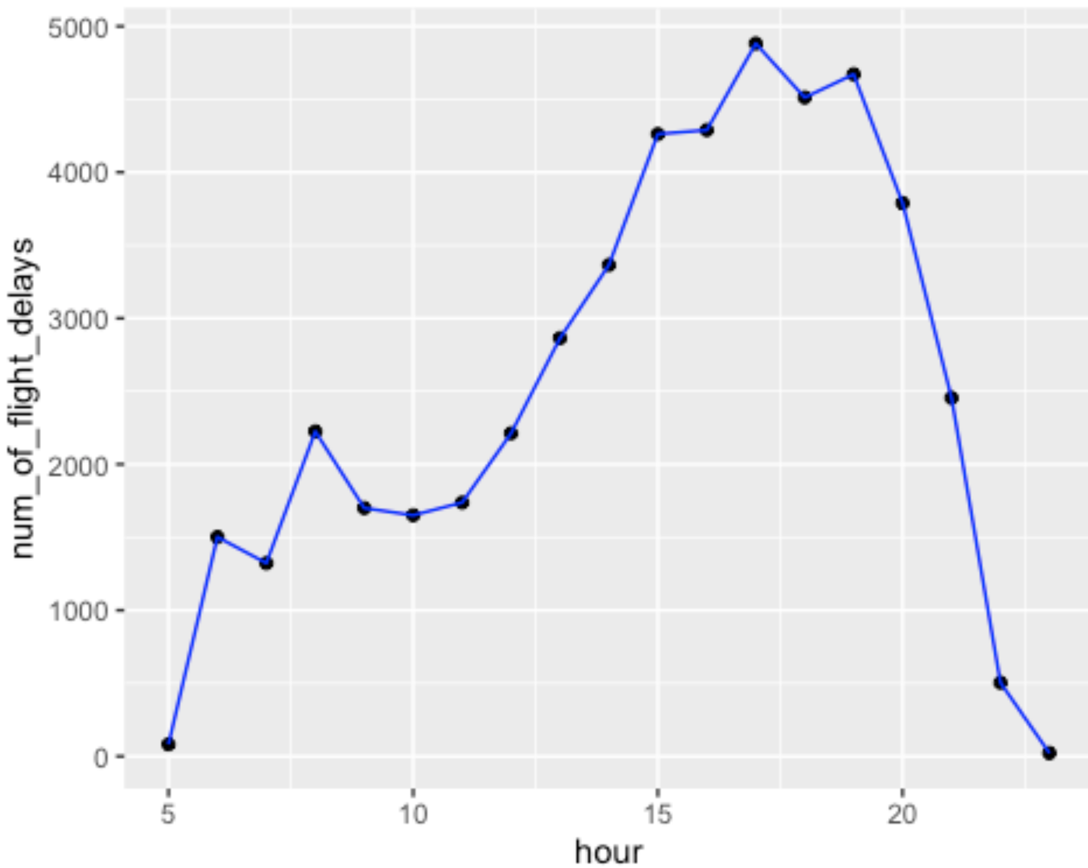
Distribution of flight delays and cancellations by months and hours in a day

```
fl %>% filter(dep_delay >= 10) %>% group_by(month) %>% summarize(num_of_fligh
t_delays = n()) %>%
  ggplot(aes(x= month, y = num_of_flight_delays)) +
  geom_point() +
```

```
  geom_line(col = "blue") +
  scale_x_discrete(limits=1:12)
```




```
fl %>% filter(dep_delay >= 10) %>% group_by(hour) %>% summarize(num_of_flight
_delays = n()) %>%
  ggplot(aes(x= hour, y = num_of_flight_delays)) +
  geom_point() +
  geom_line(col = "blue")
```



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)
```

```
## # A tibble: 3 x 5
##   origin     n med_d  Q3_d max_d
##   <fct>  <int> <dbl> <dbl> <dbl>
## 1 EWR    65512    -1    15   896
## 2 JFK    60327    -1    10  1301
## 3 LGA    58477    -3     7   911
```

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)
```

```
## # A tibble: 10 x 5
##    carrier     n med_d  Q3_d max_d
```

```
##     <fct>    <int> <dbl> <dbl> <dbl>
##   1 EV       29137    -1  25      536
##   2 WN        6897     1  18      471
##   3 F9         388     0  17.2    853
##   4 9E       10179    -2  16      430
##   5 FL        1832     1  16      602
##   6 YV         312    -3  13      387
##   7 B6       29282    -1  12      502
##   8 UA       32252     0  11      483
##   9 MQ       14382    -3   9      486
##  10 VX        2991     0   7      653
```

```
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)
```

```
## # A tibble: 10 x 6
## # Groups:   origin [3]
##     origin carrier     n med_d  Q3_d max_d
##     <fct>  <fct>   <int> <dbl> <dbl> <dbl>
##   1 EWR    OO          3     4  67.5   131
##   2 EWR    EV      23565    -1  26     443
##   3 LGA    EV       4769    -2  22     473
##   4 JFK    9E       8126    -1  20     430
##   5 JFK    EV        803    -2  19     536
##   6 EWR    WN       3487     2  18     440
##   7 LGA    WN       3410     1  18     471
##   8 LGA    F9        388     0  17.2   853
##   9 EWR    MQ       1156    -2  17     381
##  10 LGA    FL       1832     1  16     602
```

```
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)
```

```
## # A tibble: 10 x 6
## # Groups:   dest [10]
##     dest  carrier     n med_d  Q3_d max_d
##     <fct> <fct>   <int> <dbl> <dbl> <dbl>
##   1 STL   UA          2  77.5 116.    155
##   2 DTW   OO          2  61    96     131
##   3 TYS   EV        183   8    68.5   285
##   4 PBI   EV          3  50    67.5    85
##   5 ORD   OO          1  67    67      67
##   6 RDU   UA          1  60    60      60
##   7 TUL   EV        185   3    53     251
##   8 OKC   EV        184   8.5  51.5   207
##   9 BHM   EV        175   3    50     325
##  10 CAE   EV         57  10    48     163
```

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?
```

```
## # A tibble: 10 x 5
## # Groups:   month [7]
##    month   day     n med_d max_d
##    <dbl> <dbl> <int> <dbl> <dbl>
##  1     3     8   461  79.5   470
##  2     7     1   505  58.1   363
##  3     7    10   471  56.6   576
##  4     9     2   438  53.7   696
##  5    12     5   458  52.2   896
##  6     5    23   453  51.5   410
##  7     4    19   511  50.4   812
##  8     9    12   444  50.4   602
##  9     6    13   469  50.3   388
## 10     7    22   476  49.9   898
```

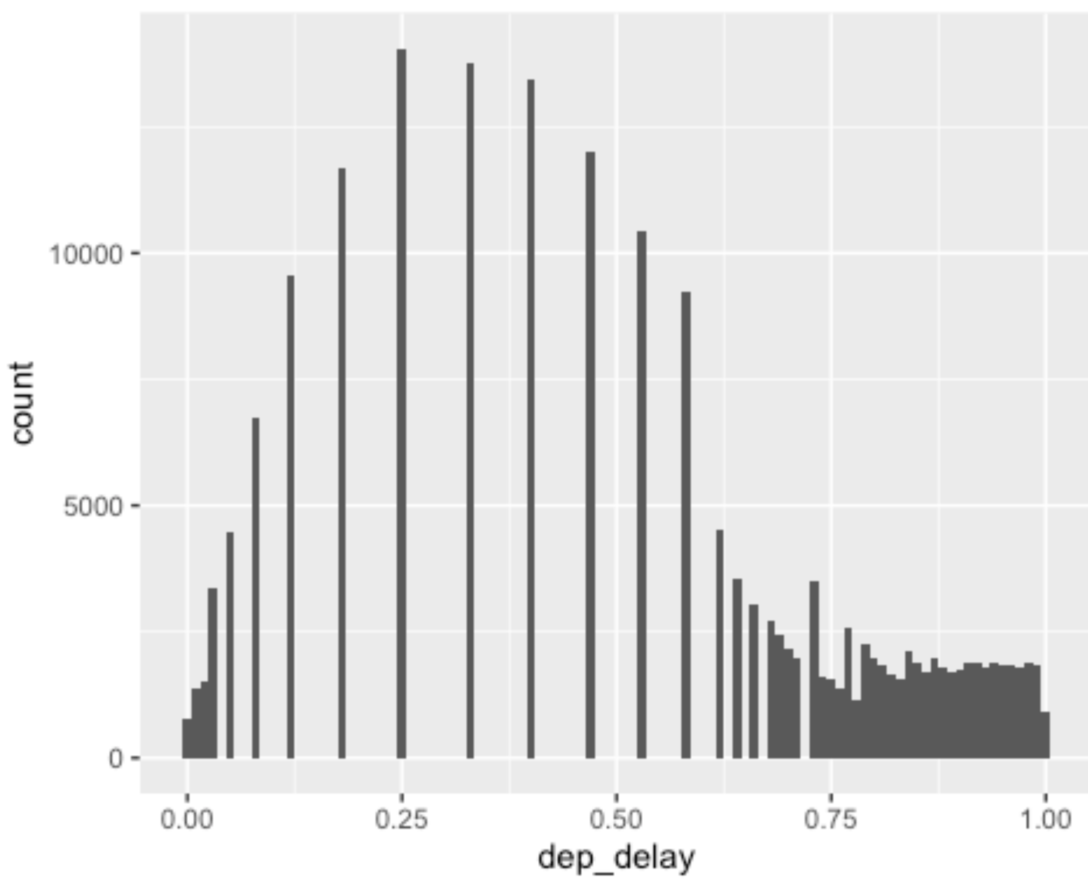
Summaries of departure delay by precipitation:

```
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)
```

```
## # A tibble: 2 x 5
##   haveprecip      n med_d  Q3_d max_d
##   <fct>       <int> <dbl> <dbl> <dbl>
## 1 TRUE        11804     5    41   853
## 2 FALSE      172512    -2     9  1301
```

Ranking the departure delays for making better predictions

```
#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)
```



Making dep_date column from year,month and date

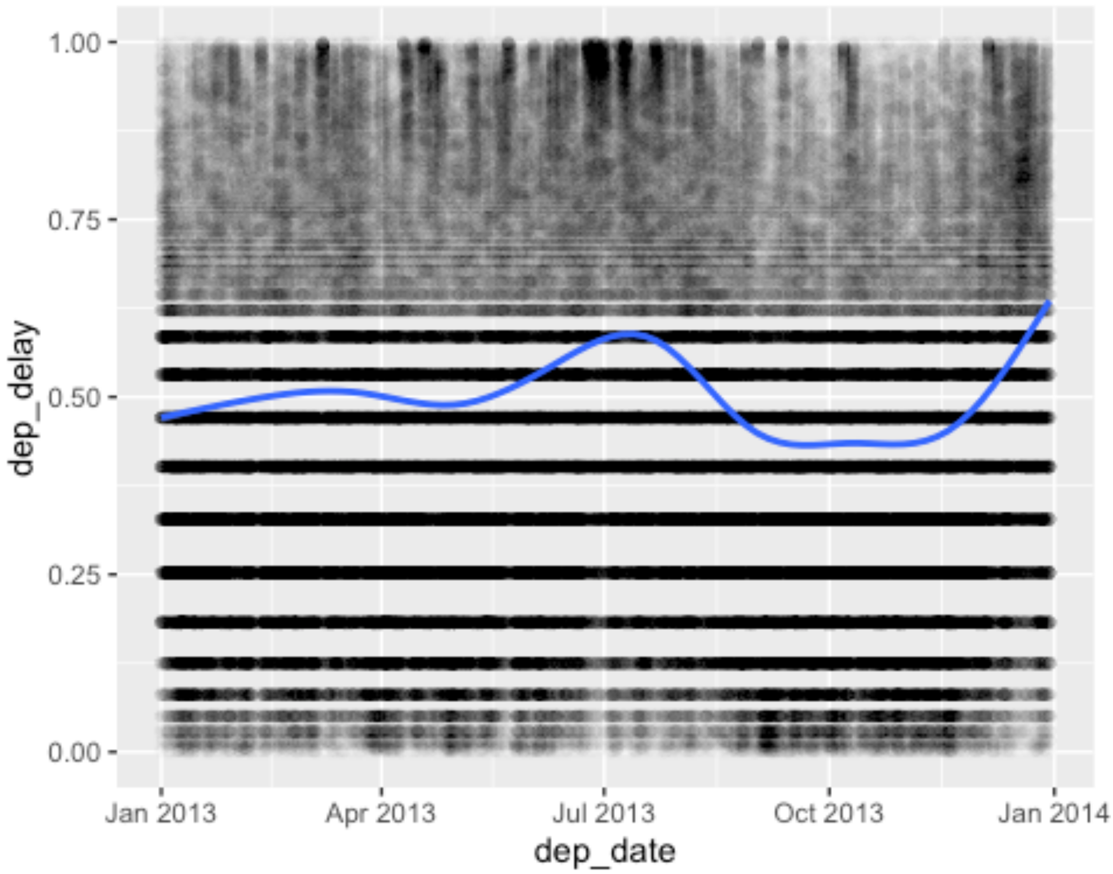
```
library(lubridate)

##
## Attaching package: 'lubridate'

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

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,-dest) %>%
  mutate(precip = as.numeric(precip>0))
```

Plots to see relation between departure delays and the feature

Part 2 : Data Exploration

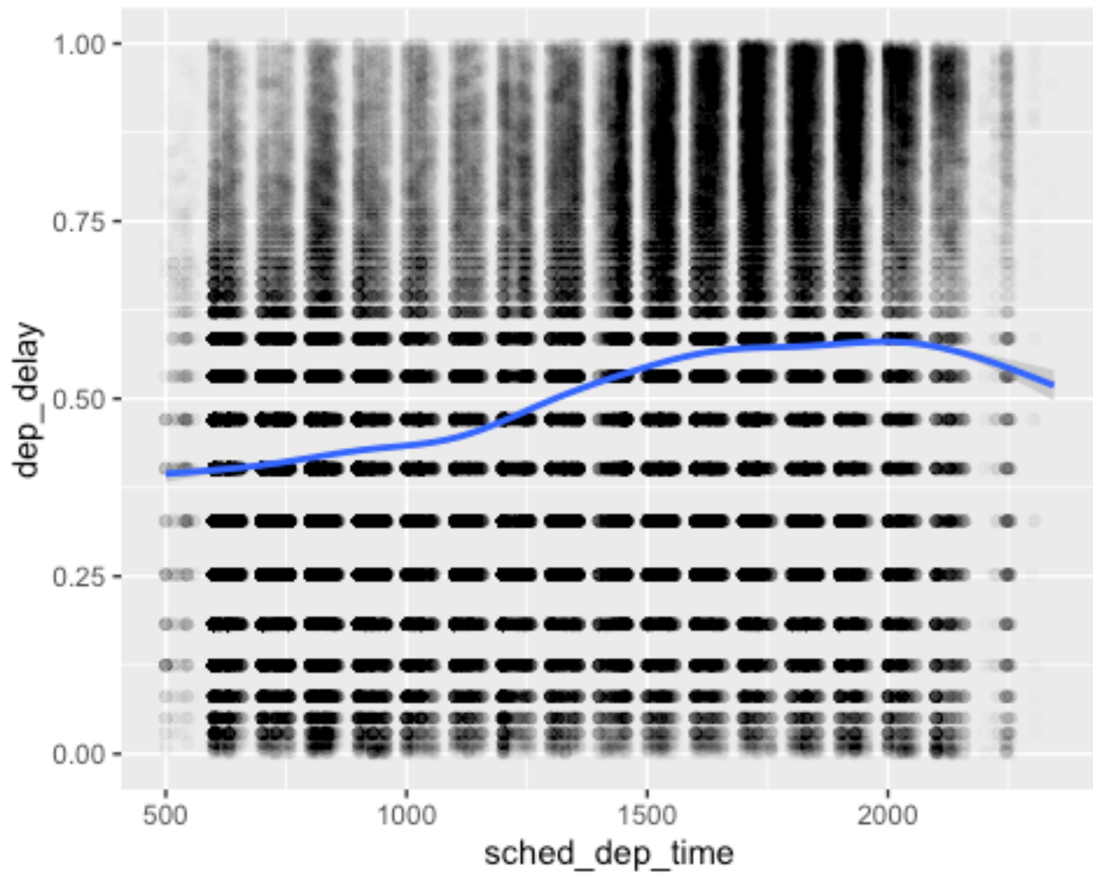
```
ggplot(fl,aes(x=dep_date,y=dep_delay)) + geom_point(alpha=.01) + geom_smooth(
)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

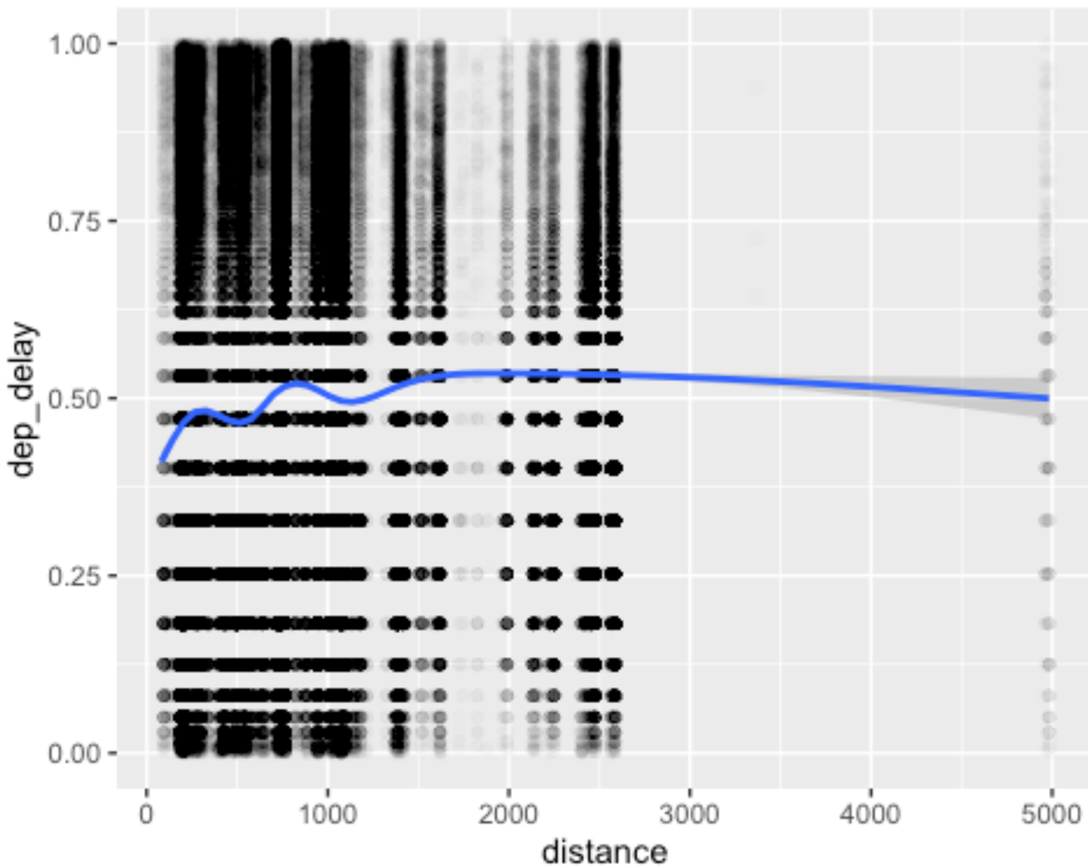



```r
ggplot(fl,aes(x=sched_dep_time,y=dep_delay)) + geom_point(alpha=0.01) + geom_
smooth()
```

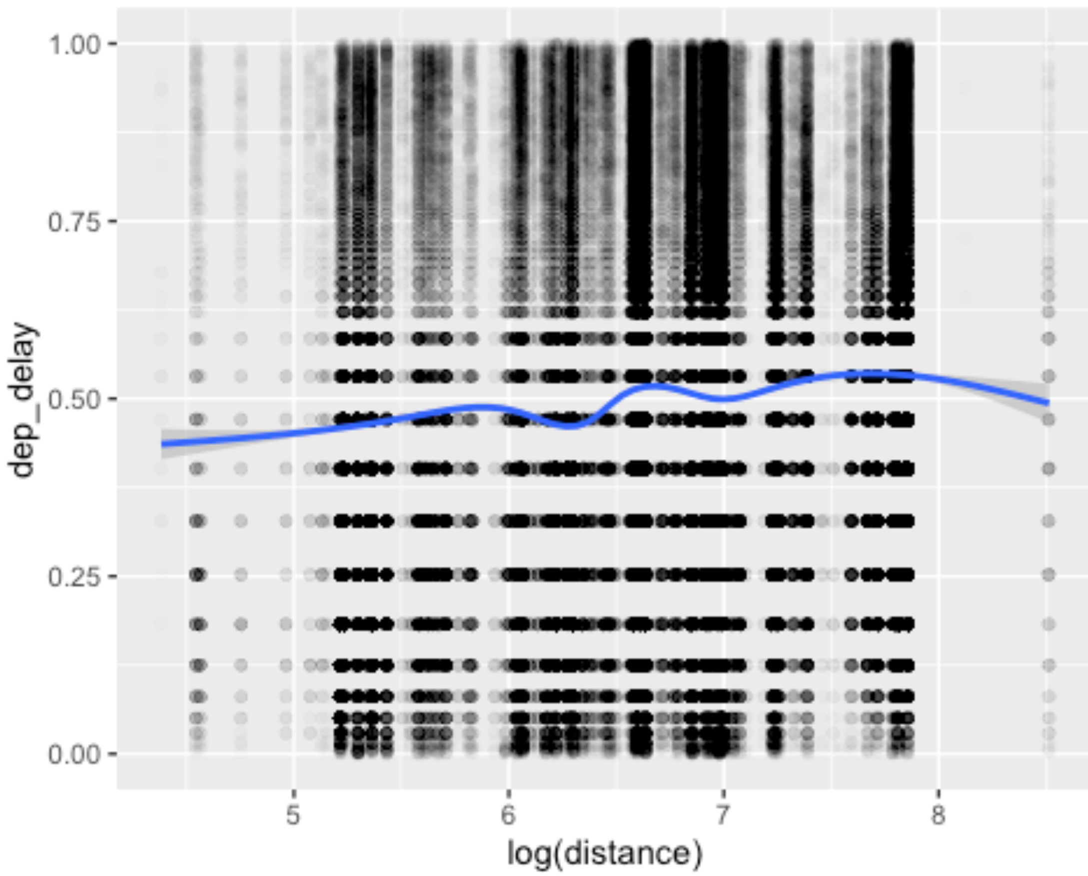
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
ggplot(fl,aes(x=distance,y=dep_delay)) + geom_point(alpha=0.01) + geom_smooth
()

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



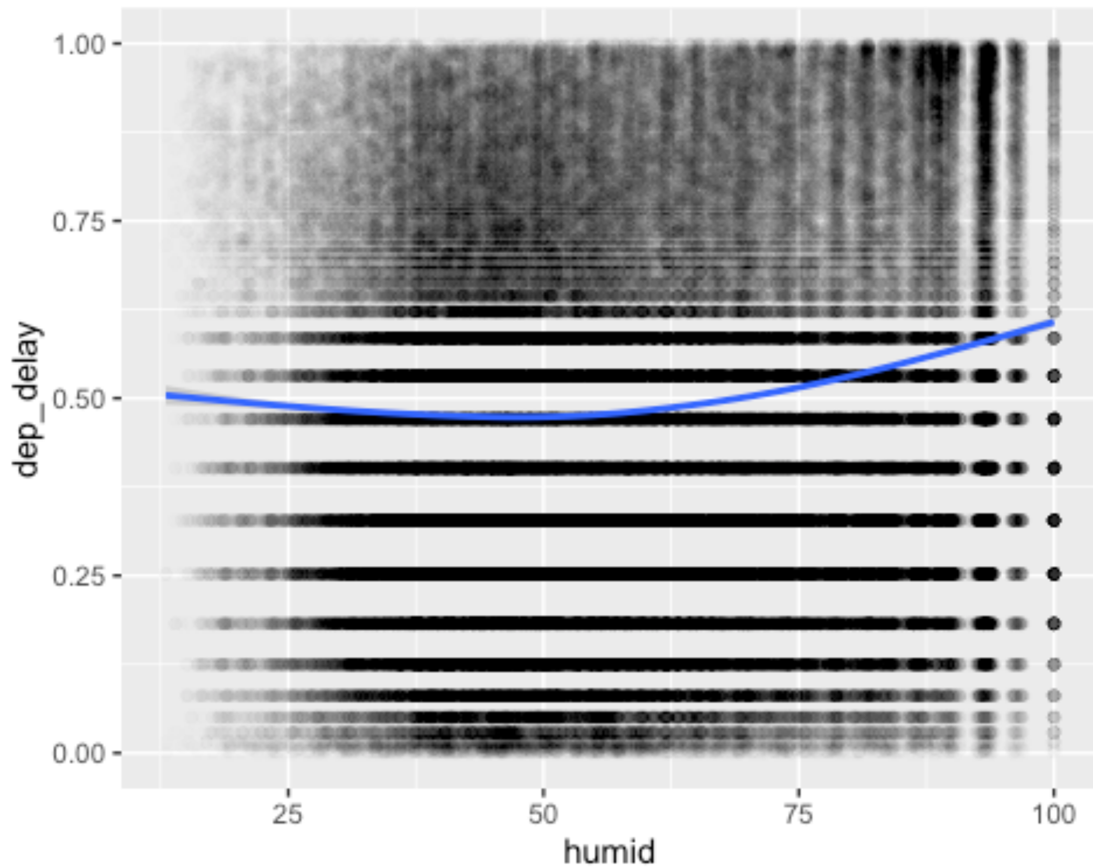

```
ggplot(fl,aes(x=log(distance),y=dep_delay)) + geom_point(alpha=0.01) + geom_s
mooth()

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
ggplot(fl,aes(x=dewp,y=dep_delay)) + geom_point(alpha=0.01) + geom_smooth()

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
# Increase in departure delay  with increase in dewp
ggplot(fl,aes(x=temp,y=dep_delay)) + geom_point(alpha=0.01) + geom_smooth()

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
# delays when too hot or too cold
ggplot(fl,aes(x=humid,y=dep_delay)) + geom_point(alpha=0.01) + geom_smooth()

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Replace alt with log(alt)

```
fl <- mutate(fl,logalt = log(alt)) %>% select(-alt)
# increases with distance -- use log distance
fl <- mutate(fl,logdistance = log(distance)) %>% select(-distance)
```

Before fitting the model the dataset is split into training and validation sets in random fashion of ratio 70:30. The model is fitted on train data and then fit on validation data to predict the validation error.

```
set.seed(123)
tr_size <- ceiling(2*nrow(fl)/3)
train <- sample(1:nrow(fl),size=tr_size)
fl_tr <- fl[train,]
fl_te <- fl[-train,]
# baseline to compare learning methods to:
var_dd <- var(fl_te$dep_delay)
var_dd

## [1] 0.08311941
```

Part 3 : Learning methods

1) XG Boost

```r
#install.packages("xgboost")
library(xgboost)

library(data.table)

dtrain <- xgb.DMatrix(label = fl_tr$dep_delay, data = data.matrix(fl_tr[-2]))

xgb <- xgboost(data = dtrain,
               max_depth = 9,
               eta = 0.1,
               nround=160,
               seed = 1,
               eval_metric = "rmse",
)

dtest <- xgb.DMatrix(label = fl_te$dep_delay, data = data.matrix(fl_te[-2]))
xg_pred= predict(xgb,dtest)
mse_xg <- mean((fl_te$dep_delay-xg_pred)^2)
mse_xg

## [1] 0.06239184
```

Getting the summary of the important features used in XG Boost

```r
names <- dimnames(data.matrix(fl_tr[,-2]))[[2]]
importance_matrix <- xgb.importance(names, model = xgb)
xgb.plot.importance(importance_matrix, rel_to_first = TRUE, xlab = "Relative
importance")
```

Using for loops to get best value for **parameter tuning**

```r
cv <- matrix(nrow = 80, ncol = 3)
nroundlist <-c(100,120,135,150,160)
i  <- 1
etas <-c(0.1,0.2,0.3)
max_depthlist <-c(7,8,9,10)
for(x in etas){
  for(round in nroundlist){
    for(depth in max_depthlist){
      xgb <- xgboost(data = dtrain,
                     max_depth = depth,
                     eta = x,
                     nround=round,
                     seed = 1,
                     eval_metric = "rmse",
      )
      xg_pred= predict(xgb,dtest)
      mse_xg <- mean((fl_te$dep_delay-xg_pred)^2)
      cv[i,1] <- round
      cv[i,2] <- depth
      cv[i,3] <- mse_xg
```

```
      i <- i+1
    }
  }
}
```

2) GAM

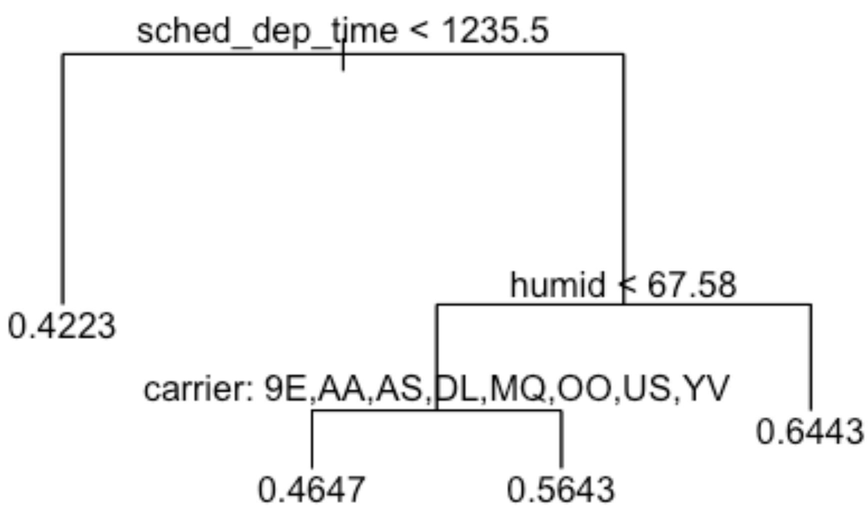
```
#install.packages("gam")
library(gam)
form <- formula(dep_delay ~ s(dep_date) + s(sched_dep_time) + carrier + origi
n + tzone + s(logdistance) +
                    s(temp) + s(dewp) + s(humid) + s(wind_dir) + s(wind_speed)
+ precip + s(visib))
gam_fit <- gam(form, data=fl_tr,family=gaussian)

## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts
## argument ignored

gam_pred <- predict(gam_fit,newdata=fl_te)
mse_gam <- mean((fl_te$dep_delay-gam_pred)^2)
mse_gam

## [1] 0.07066494

abs(mse_gam - var_dd)/var_dd

## [1] 0.1498383
```

3) GBM

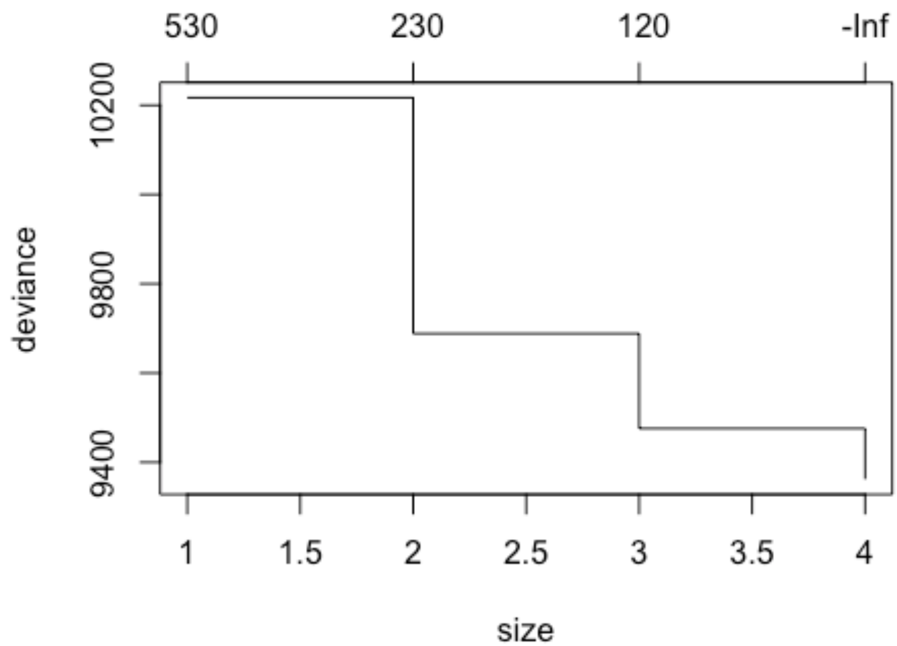
```
library(gbm)
dep_date_numeric <- as.numeric(fl_tr$dep_date)
dep_date_numeric <- dep_date_numeric - mean(dep_date_numeric)
fl_tr_tem <- mutate(fl_tr,dep_date = dep_date_numeric)
gbm_fit <-gbm(dep_delay ~ .,data=fl_tr_tem,distribution="gaussian",
              n.trees = 2000, shrinkage = 0.01)
#summary(gbm_fit)

dep_date_numeric <- as.numeric(fl_te$dep_date)
dep_date_numeric <- dep_date_numeric - mean(dep_date_numeric)
fl_te_tem <- mutate(fl_te,dep_date = dep_date_numeric)

gbm_pred <- predict(gbm_fit,newdata=fl_te_tem,n.trees = 200)
mse_gbm <- mean((fl_te$dep_delay-gbm_pred)^2)
mse_gbm

## [1] 0.07712024

#in 1000 : 0.07206
#in 2000 : 0.07068754
abs(mse_gbm - var_dd)/var_dd

## [1] 0.0721753
```

4) Decision Tree

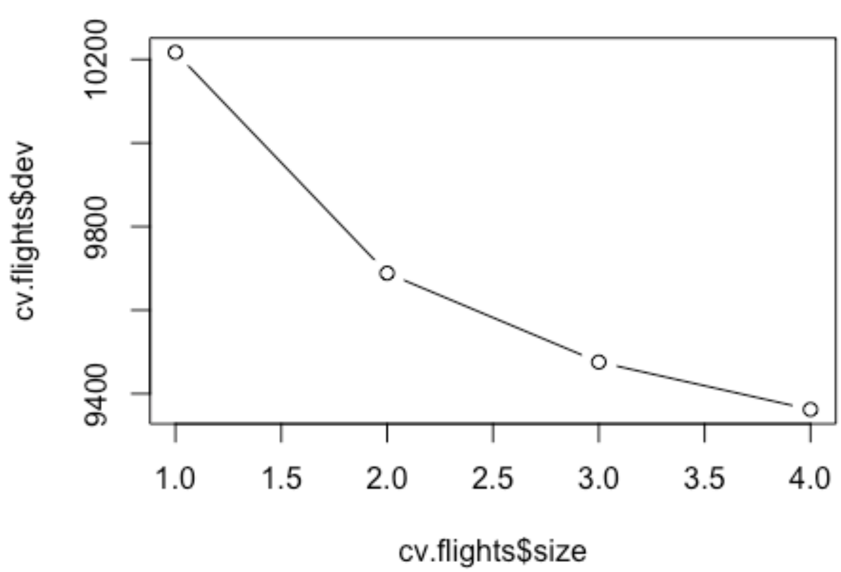
```
library(tree)

## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli

decision_tree=tree(dep_delay ~ ., data = fl_tr)

## Warning in tree(dep_delay ~ ., data = fl_tr): NAs introduced by coercion

summary(decision_tree)

##
## Regression tree:
## tree(formula = dep_delay ~ ., data = fl_tr)
## Variables actually used in tree construction:
## [1] "sched_dep_time" "humid"          "carrier"
## Number of terminal nodes:  4
## Residual mean deviance:  0.07595 = 9332 / 122900
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.644300 -0.239700  0.006016  0.000000  0.228100  0.577700

plot(decision_tree)
text(decision_tree, pretty = 0)
```



```
cv.flights=cv.tree(decision_tree)
plot(cv.flights)
```




```
plot(cv.flights$size ,cv.flights$dev ,type='b')
```



```
prune.flights3=prune.tree(decision_tree ,best=3)
tree_pred2=predict(decision_tree ,newdata=fl_te)

mse_tree2 <- mean((fl_te$dep_delay-tree_pred2)^2)
mse_tree2

## [1] 0.07628095
```

Part 4 : Evaluating on Test set

Applying all the same feature selections which was applied on training data.

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

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

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

flt <- na.omit(flt)


den <- nrow(flt)+1
flt <- flt %>% mutate(dep_delay = rank(dep_delay)/den)

flt <- flt %>%
  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,-dest) %>%
  mutate(precip = as.numeric(precip>0))

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

Testing Predictions

```
library(gam)

form <- formula(dep_delay ~ s(dep_date) + s(sched_dep_time) + carrier + origi
n + tzone + s(logdistance) +
                  s(temp) + s(dewp) + s(humid) + s(wind_dir) + s(wind_speed)
+ precip + s(visib))
gam_fit <- gam(form, data=fl_tr,family=gaussian)
```

```r
#summary(gam_fit)
#plot(gam_fit,se=TRUE)
gam_pred <- predict(gam_fit,newdata=flt)
mse_gam <- mean((flt$dep_delay-gam_pred)^2)
mse_gam

## [1] 0.0703773

abs(mse_gam - var_dd)/var_dd

## [1] 0.1532988

##For GBM
library(gbm)

## Loaded gbm 2.1.5

dep_date_numeric <- as.numeric(fl_tr$dep_date)
dep_date_numeric <- dep_date_numeric - mean(dep_date_numeric)
fl_tr_tem <- mutate(fl_tr,dep_date = dep_date_numeric)
gbm_fit <-gbm(dep_delay ~ .,data=fl_tr_tem,distribution="gaussian",
              n.trees = 2000, shrinkage = 0.01)
dep_date_numeric <- as.numeric(flt$dep_date)
dep_date_numeric <- dep_date_numeric - mean(dep_date_numeric)
fl_te_tem1 <- mutate(flt,dep_date = dep_date_numeric)
#
gbm_pred1 <- predict(gbm_fit,newdata=fl_te_tem1,n.trees = 2000)
mse_gbm1 <- mean((flt$dep_delay-gbm_pred1)^2)
mse_gbm1

## [1] 0.07048635

##For XGBoost
library(xgboost)
library(data.table)

dtrain <- xgb.DMatrix(label = fl_tr$dep_delay, data = data.matrix(fl_tr[-2]))

xgb <- xgboost(data = dtrain,
               max_depth = 9,
               eta = 0.1,
               nround=160,
               seed = 1,
               eval_metric = "rmse",
)

dtest1 <- xgb.DMatrix(label = flt$dep_delay, data = data.matrix(flt[-2]))
xg_pred1= predict(xgb,dtest1)

mse_xg_test <- mean((flt$dep_delay-xg_pred1)^2)
mse_xg_test
```

```
## [1] 0.06215232

##For Decision Tree
library(tree)
decision_tree=tree(dep_delay ~ ., data = fl_tr)

## Warning in tree(dep_delay ~ ., data = fl_tr): NAs introduced by coercion

tree_pred2=predict(decision_tree ,newdata=flt)

## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by
## coercion

mse_tree2 <- mean((flt$dep_delay-tree_pred2)^2)
mse_tree2

## [1] 0.07586896
```