STAT 652: Predicting Flight Delays Project

Vincent Chiu

11/26/2019

Contents

1 Introduction

The goal of this project is to predict the response variable, departure delays for a particular flight given the explanatory variables.

2 Data

The dataset consists of information about all the flights leaving from New York City in 2013. The dataset contains 43 variables in total. The dataset is an amalgamation of several datasets including datasets containing information on weather, the airports, the flights, and the models of airplanes. The training dataset provided to us contains 200,000 observations. Please see table 1 for what the data looks like.

3 Methods:

We will now outline the various methods used to clean and perform prediction on the data. We will discuss our techniques for data preprocessing, and cross validation, and the different models that we tried.

3.1 Data Preprocessing

I performed data preprocessing on the nycflights13 dataset. My data preprocessing steps include the following:

- loading the data from a csv
- setting the random seed for reproducibility of results
- casting all the columns with the character data type into the factor data type
- converting the shed_arr_time and sched_dep_time columns into the POSIX time format so that I can accurately take the difference of them.
- Dropping columns that contain data from after the planes' departure which may leak information about the response variable dep_delay. We drop columns "dep_time", "arr_time", "air_time", and "arr_delay".
- We drop column "year.x" because all the values are 2013
- We also drop tailnum because it produces too many dummy variable columns for one hot encoding.
- Dropping columns which consists of over 50% NAs which include the speed column. However, it should be noted that a rule of thumb suggested by Professor McNeney is to drop any columns with over 5% NAs. We use a different threshold for dropping columns leading to us keeping columns such as model instead of dropping it.
- Afterwards, I impute the missing values. However, there are limitations to this approach of imputing the missing values. It is possible, that the missingness of the plane model variable is related to dep_delay. In this scenario, we may be creating an inferior feature set by keeping the variable 'model' and imputing it. For example, say a highly unreliable plane model that frequently causes long delays has a high probability of being labeled as NA and represents the majority of NAs in the dataset. We would not be able to capture the relationship between this plane model and dep_delay if we imputed the model with the mode. To counteract this affect, we imputed NAs for the remaining columns using the imputeMissings library, adding a Boolean flag which indicates 1 if the associated value was 1 and 0 otherwise. For example, the "model_flag" for a given row is 1 if the "model" value was NA for that given row. Hence, no information is lost from our imputation.
- Normalizing the data to work well with methods like lasso regression.
- Only kept data which had a departure delay of less than 30 minutes late, which reduced the dataset from 200,000 rows to approximately 170,000. This is because we consider extreme delays of of over 30 minutes late to be freak accidents which cannot be accurately predicted by the available explanatory variables.

3.2 Exploratory Analysis

3.2.1 Correlations

First, we created a correlation plot for the numeric variables to see if there any correlations between the variables.

We see that there is very little correlation between the response variable dep_delay and any of the other variables. Some of the strongest correlations include the correlation between distance and longitude and time zone and a smaller correlation between distance and latitude. This makes sense as most of the planes are inter US flights from west to east or vice versa, there is not as much distance flown in the north south direction. Please see Figure 1 for the correlation plot (Click on the number after Figure to jump to plot).

3.3 Principal Component Analysis (PCA)

Next we performed PCA on only the numeric variables as techniques to perform PCA on mixed datasets (numerical and categorical) was not covered in class. When looking at the contribution of each variable to the first principal component, we notice that the variables lon, distance, tz, seats, alt, sched_air_time have the greatest absolute coefficients for the first principal component. The fact that the aforementioned variables have large coefficients in the first principal component suggests that they are highly correlated with each other. The fact that dep_delay has a small coefficient in the first principal component suggests that dep_delay is not highly correlated with any of the above variables.

As expected, it turns out that variables like lon, distance and tz are not important for predicting dep_delay according to the gbm model. This maybe be because although variables like lon, distance and tz help explain most of the variance in the dataset, they have a weak relationship with dep_delay.

Please see table 2 for the proportion of variance explained by each principal component. Please see table 3 for the coefficient of each variable for each principal component ordered by magnitude of coefficient.

3.4 Cross Validation

Initially, I used the most basic cross validation technique where I have a training dataset and a cross validation dataset. I split the original data into a ratio of 2/3 train and 1/3 of the data for cross validation. There is a additional data which would be provided by the professor at a later date which we will use as the holdout test set. I believe that 2/3 of the data gives enough data for the models to train on while 1/3 is enough data for us to get an accurate assessment of the error. k-folds cross validation was not initially used in order to save on compute time as we were initially only exploring the models. k-folds cross validation would increase training time for the models by a factor of k. However, k-folds cross validation would lead to a more stable estimate of holdout test set error.

3.5 Models

We first explored some basic models to establish a baseline performance and compared it to our most sophisticated model, the Generalized Boosted Regression Model (GBM).

3.5.1 Basic Models

dep_delay is the number of minutes that the plane either departs early or late. Negative numbers are for early departures and positive numbers are for the number of minutes the plane is late. First, I used a basic model of simply predicting the dep_delay to always be 0. This was done to establish baseline performance.

This model had an root mean squared error (RMSE) of 8.30571. TODO The model in which I predicted the mean for all the predictions had an RMSE of TODO.

3.5.2 Linear Regression

A linear regression model assumes a linear relationship between the explanatory variables and the response variable. The model minimizes the squared loss function. This minimization process generates coefficients which are used for a linear combintation of the explanatory variables. The linear combination is the prediction. Then I tried linear regression with dep_delay as the response variables and all the other remaining variables as the explanatory variables. This model was better than predicting the mean with an RMSE of TODO. This suggests that there is some relationship between the dep_delay and the explanatory variables.

3.5.3 Generalized Boosted Regression Model (GBM)

How boosted regression models work is we first start off with a baseline prediction. For example, we can use the mean as our baseline prediction. Then we use a base classifier to iteratively predict on the residuals multiplied by the shrinkage hyperparameter. Then this base classifier is added to the enssemble of base classifiers trained so far. A lower shrinkage effectively means a lower learning rate and therefore you need more iterations to reduce the train set residuals by the same amount. The benefit of a smaller shrinkage (with sufficient trees) is that you end up with a larger ensemble of trees that can reach a lower cross validation loss. This iterative prediction process is called boosting. In our case, our base classifiers are regressor trees. Each tree decides on its splitting criterion greedily by picking the split which results in the lowest mean squared error or some other splitting heuristic. We continue for n number of trees where n is specified by the user. Each iteration produces one tree, so the number of iterations is equal to the number of trees. Afterwards, we tried a Generalized Boosted Regression Model (GBM). This model had the lowest RMSE of 7.94458 on the cross validation set after it was tuned to have a shrinkage of 0.01 and around 16,000 trees. Shrinkage is proportional to the learning rate. 16,000 trees is the number of trees used in the model. Each iteration uses 1 tree, so 16,000 trees also refers to the number of iterations. According to the vignette, the RMSE can always be improved by decreasing shrinkage, but this provides diminishing returns. A good strategy would be to pick a small shrinkage that balances performance and compute time. Then with this fixed shrinkage value, increase the number of trees until you get diminishing returns. We decided to follow the aforementioned strategy. Please see table @4 for a table of the relative influence of each explanatory variable. Here, you can see the relative influence for each variable for gbm.

For a gbm, the improvement in the splitting criterion (which is mean squared error for regression) for a given variable is calculated at each step. The relative influence for a given variable is the average of these improvements over all the trees where the aforementioned variable is used.

4 Results

In regression and gbm, I found different features to be important.

Please see table @5 for all the models and their root mean squared error (RMSE) on the cross validation set.

4.1 GBM

For the best gbm model, dest which refers to which airport a given plane was flying to was the most important feature. However, the one hot encoding versions of carrier were the most important features for regression. Based on the relative influence scores provided by the gbm, some of the most important feature variables include dest, model, and sched_dep_time_num_minute. The dest column contains the airport code for where a given flight is flying to. Based on my run of gbm with a shrinkage of 0.01 and 16834 trees, dest was the most important feature with 49.56 relative influence. (“Gradient Boosting Machines · UC Business Analytics

R Programming Guide" 2019). dest is the destination airport code. 'sched_dep_time_num_minute' is the number of minutes since the beginning of a given day for that flight. 'model' is the plane nodel.

4.2 Linear Regression

On the other hand, dest does appear as an important feature in linear regression as well but it is not the most important feature. I surmise that if we can somehow sum up all the contributions from each of the one-hot-encoded variables derived from dest then, it might appear as the most important feature for linear regression as well. We can try using ANOVA in order to measure the statistical significance of dest. Performing ANOVA on comparing linear regression model with and without dest, it was determined that due to the low p-value of 0.0001863 associated with having dest that keeping at least one of the one hot categorical variables derived from dest is beneficial for the linear regression model.

4.3 Comparison

Dest was most important feature in gbm. It is possible that dest is important in combination with other variables which is something that the linear model without interaction terms cannot capture the relationship of whereas gbm can discover these non linear relationships.

TODO: try interaction terms , try anova.

5 Conclusion and Discussion

5.1 Discussion

We considered removing outliers in terms of 'dep_delay' in train but not in test, then use k-folds cross validation on test to determine how many outliers we should remove to boost performance on the cross-validation set. We considered removing highly influential points in order to train a better model. In this case, we consider highly influential points to be points with high cook's distances. However, this was infeasible as we did not have enough computational resources available and it took too long.

5.2 Conclusion

None of the models that we tried performed particularly well. We surmise that this may be due to the explanatory variables having a weak relationship with the 'dep_delay' variable. There is a lack of information about a particular flight before it reaches NYC. Instead, we get information about where the flight is going next which through commons sense, would reveal less information about the current condition of the plane and what kind of maintenance it would need and therefore what dep_delay it would have. Out of the methods that we covered in class, I found gradient boosted models to provide the best performance based on having the lowest root mean squared error on the cross validations et. I believe that this makes sense because GBMs are able to capture non linear relationships between the explanatory variables and 'dep_delay' whereas linear regression cannot.

5.3 Future Work

TODo: add dep_delay vs explanatory plots. TODO: remove points that are outliers ie dep_delay > 200 or 300 etc. or remove less than x number of points. then use k-folds cross validation on cross validation set where no points were removed. can repeat k-folds for different seeds. can just try this on my quickest model, i.e. linear regression. should be bowl shape vs RMSE vs. number of points removed. theoretically

I also considered removing based on cook's distance but this took too long to compute.

5 folds with 10 different random seeds

have train, CV and test set 1/3 train, 1/3 CV, 1/3 test 2/3% train, 1/3%CV, wait for prof test set

try lasso regression add more insights into generating dataset.

Conclusion stuff. certain planes tend to fly back and forth and that is why dest is good? interaction terms may be important because they are hard to predict in advance when you schedule flights which leads to delays? add more about shortcomings of approach. cut off too much of tail, reduce predictive power on test dataset which has tail in exchange for better performance on stuff not in tail.

maybe for stretch goal try rank stuff that prof did.

6 Code

6.1 Preparing the programming environment

6.1.1 Loading Libraries

```r
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------ tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts --------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

6.2 Data Preprocessing

6.2.1 Loading the data

```r
library(nycflights13)
library(Hmisc)
```

```
## Loading required package: lattice
```

```
## Loading required package: survival
```

```
## Loading required package: Formula
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:dplyr':
##
##     src, summarize
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

Table 1: A table of the first few rows of the nycflights13 data.

year.x	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	carrier	f
2013	11	7	600	600	0	826	825	1	WN	
2013	10	30	1252	1250	2	1356	1400	-4	AA	
2013	12	18	1723	1715	8	2008	2020	-12	DL	
2013	11	20	2029	2030	-1	2141	2205	-24	WN	
2013	10	21	1620	1625	-5	1818	1831	-13	DL	
2013	11	7	852	900	-8	1139	1157	-18	B6	

```
set.seed(42)
original_data <- read_csv("fltrain.csv.gz")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   carrier = col_character(),
##   tailnum = col_character(),
##   origin = col_character(),
##   dest = col_character(),
##   time_hour = col_datetime(format = ""),
##   name = col_character(),
##   dst = col_character(),
##   tzone = col_character(),
##   type = col_character(),
##   manufacturer = col_character(),
##   model = col_character(),
##   engine = col_character()
## )

## See spec(...) for full column specifications.
```

```
DF <- original_data
```

turning all columns with datatype characters to factors.

```
DF[sapply(DF, is.character)] <- lapply(DF[sapply(DF, is.character)],
                                       as.factor)
DF$flight <- as.factor(DF$flight)
```

```
library(lubridate)
```

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

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

```
DF$sched_arr_time_posix <- as.POSIXct(str_pad(as.character(DF$sched_arr_time), 4, pad="0"),format="%H%M
DF$sched_arr_time_hour <- hour(DF$sched_arr_time_posix)
DF$sched_arr_time_minute <- minute(DF$sched_arr_time_posix)

#num minute is number of minutes since start of day for scheduled arrival time
```

```r
DF$sched_arr_time_num_minute <- 60*DF$sched_arr_time_hour + DF$sched_arr_time_minute

DF$sched_dep_time_posix <- as.POSIXct(str_pad(as.character(DF$sched_dep_time),4 , pad="0"),format="%H%M
DF$sched_dep_time_hour <- hour(DF$sched_dep_time_posix)
DF$sched_dep_time_minute <- minute(DF$sched_dep_time_posix)
#num minute is number of minutes since start of day for scheduled depival time
DF$sched_dep_time_num_minute <- 60*DF$sched_dep_time_hour + DF$sched_dep_time_minute
```

```r
select(original_data, time_hour, sched_dep_time, sched_arr_time, tz, tzone)
```

```
## # A tibble: 200,000 x 5
##    time_hour           sched_dep_time sched_arr_time    tz tzone
##    <dttm>                       <dbl>          <dbl> <dbl> <chr>
##  1 2013-11-07 11:00:00            600            825    -5 America/New_York
##  2 2013-10-30 16:00:00           1250           1400    -5 America/New_York
##  3 2013-12-18 22:00:00           1715           2020    -5 America/New_York
##  4 2013-11-21 01:00:00           2030           2205    -6 America/Chicago
##  5 2013-10-21 20:00:00           1625           1831    -5 America/New_York
##  6 2013-11-07 14:00:00            900           1157    -5 America/New_York
##  7 2013-09-29 19:00:00           1529           1649    -6 America/Chicago
##  8 2013-12-21 20:00:00           1530           1710    -6 America/Chicago
##  9 2013-11-07 21:00:00           1650           1906    -5 America/New_York
## 10 2013-03-31 21:00:00           1700           1821    -5 America/New_York
## # ... with 199,990 more rows
```

```r
select(DF, sched_arr_time, sched_arr_time_hour)
```

```
## # A tibble: 200,000 x 2
##    sched_arr_time sched_arr_time_hour
##             <dbl>               <int>
##  1            825                   8
##  2           1400                  14
##  3           2020                  20
##  4           2205                  22
##  5           1831                  18
##  6           1157                  11
##  7           1649                  16
##  8           1710                  17
##  9           1906                  19
## 10           1821                  18
## # ... with 199,990 more rows
```

```r
DF$sched_air_time <- DF$sched_arr_time_posix - DF$sched_dep_time_posix
drops <- c('sched_arr_time_posix', 'sched_arr_time_hour', 'sched_dep_time_posix', 'sched_dep_time_hour'
DF <- DF[ , !(names(DF) %in% drops)]
```

```r
drops <- c("dep_time", "arr_time", "air_time", "arr_delay", "year.x", 'tailnum')
DF <- DF[ , !(names(DF) %in% drops)]
```

```r
## Remove columns with more than 50% NA
DF <- DF[, -which(colMeans(is.na(DF)) > 0.5)]
```

```r
DF$sched_air_time <- as.numeric(DF$sched_air_time)
library(imputeMissings)
```

```
##
```

```
## Attaching package: 'imputeMissings'

## The following object is masked from 'package:Hmisc':
##
##     impute

## The following object is masked from 'package:dplyr':
##
##     compute
```

```
impute_model <- imputeMissings::compute(DF, method="median/mode")
impute_model
```

```
## $month
## [1] 7
##
## $day
## [1] 16
##
## $dep_delay
## [1] -2
##
## $carrier
## [1] "UA"
##
## $flight
## [1] "15"
##
## $origin
## [1] "EWR"
##
## $dest
## [1] "ATL"
##
## $distance
## [1] 872
##
## $temp
## [1] 57.2
##
## $dewp
## [1] 42.8
##
## $humid
## [1] 57.69
##
## $wind_dir
## [1] 220
##
## $wind_speed
## [1] 10.35702
##
## $precip
## [1] 0
##
## $pressure
## [1] 1017.5
##
## $visib
## [1] 10
##
## $name
## [1] "Hartsfield Jackson Atlanta Intl"
##
## $lat
## [1] 36.09775
##
## $lon
## [1] -83.35339
##
## $alt
## [1] 433
##
## $tz
## [1] -5
##
## $dst
## [1] "A"
##
## $tzone
## [1] "America/New_York"
##
## $year.y
## [1] 2002
##
## $type
## [1] "Fixed wing multi engine"
##
## $manufacturer
## [1] "BOEING"
##
## $model
## [1] "A320-232"
##
## $engines
## [1] 2
##
## $seats
## [1] 149
##
## $engine
## [1] "Turbo-fan"
##
## $sched_arr_time_minute
## [1] 30
##
## $sched_arr_time_num_minute
## [1] 957
##
## $sched_dep_time_minute
```

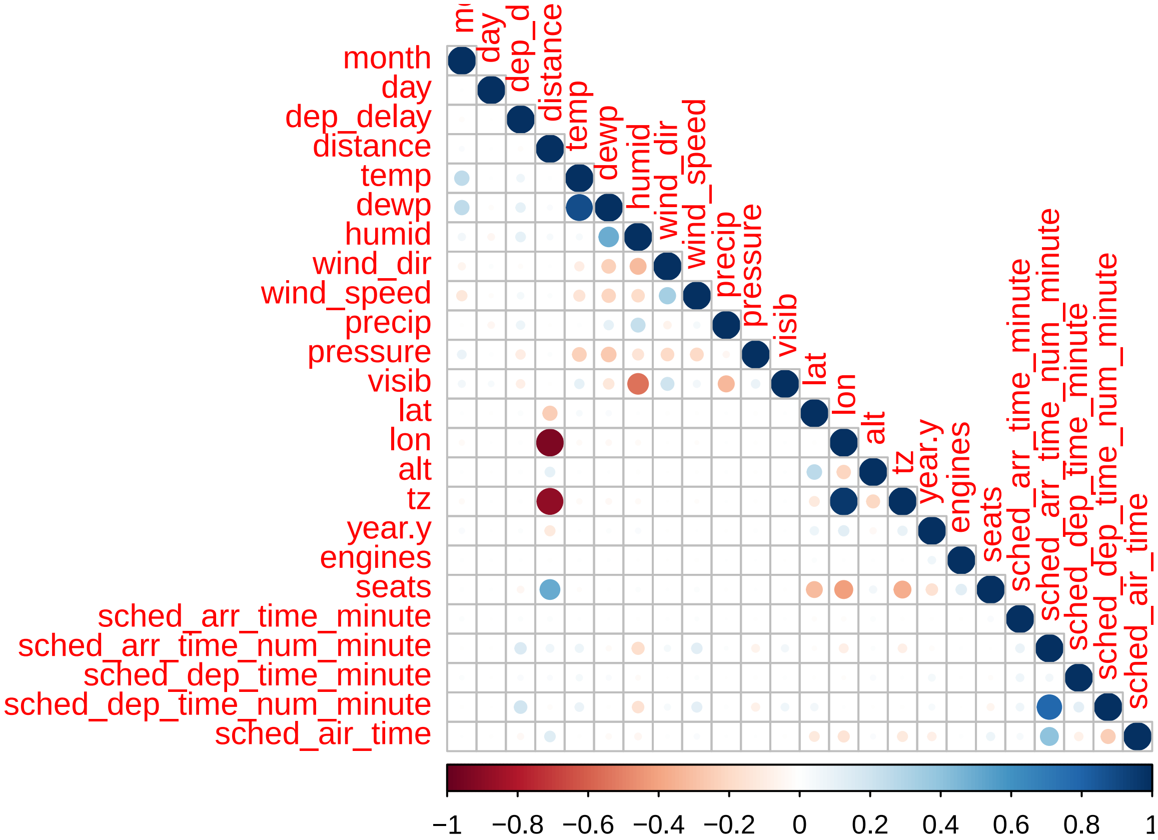


Figure 1: My caption

```
## [1] 29
##
## $sched_dep_time_num_minute
## [1] 839
##
## $sched_air_time
## [1] 139
```

```
DF <- impute(DF, object=impute_model, flag=TRUE)
DF <- DF[!duplicated(as.list(DF))]  #remove all redundant flag columns that are identical to each other
```

```
numeric_only_df <- dplyr::select_if(DF, is.numeric)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

6.3 try features scaling

```
dep_delay_vec <- DF$dep_delay
DF$dep_delay <- NULL
head(DF)
```

```
##   month day carrier flight origin dest distance  temp  dewp humid wind_dir
## 1    11   7      WN   1716    LGA  ATL      762 62.96 55.94 77.83      210
## 2    10  30       AA     178    JFK  BOS      187 59.00 46.94 64.22      240
## 3    12  18       DL    1585    LGA  MCO      950 33.98 17.06 49.51      270
## 4    11  20       WN    3494    EWR  MDW      711 37.04 17.96 45.58       20
## 5    10  21       DL    2231    LGA  DTW      502 62.96 41.00 44.47      160
## 6    11   7       B6      27    EWR  MCO      937 64.40 55.40 77.29      240
##   wind_speed precip pressure visib                               name      lat
## 1   13.80936      0   1011.0    10    Hartsfield Jackson Atlanta Intl 33.63672
## 2    9.20624      0   1024.9    10 General Edward Lawrence Logan Intl 42.36435
## 3   17.26170      0   1019.8    10                       Orlando Intl 28.42939
## 4    5.75390      0   1035.6    10                Chicago Midway Intl 41.78597
## 5   13.80936      0   1016.9    10             Detroit Metro Wayne Co 42.21244
## 6   16.11092      0   1017.5    10                       Orlando Intl 28.42939
##         lon  alt tz dst            tzone year.y                    type
## 1 -84.42807 1026 -5   A America/New_York   2001 Fixed wing multi engine
## 2 -71.00518   19 -5   A America/New_York   2002 Fixed wing multi engine
## 3 -81.30899   96 -5   A America/New_York   2002 Fixed wing multi engine
## 4 -87.75242  620 -6   A  America/Chicago   2006 Fixed wing multi engine
## 5 -83.35339  645 -5   A America/New_York   1992 Fixed wing multi engine
## 6 -81.30899   96 -5   A America/New_York   2006 Fixed wing multi engine
##       manufacturer    model engines seats    engine sched_arr_time_minute
## 1           BOEING  737-7H4       2   140 Turbo-fan                    25
## 2           BOEING A320-232       2   149 Turbo-fan                     0
## 3           AIRBUS A319-114       2   145 Turbo-fan                    20
## 4           BOEING  737-7H4       2   140 Turbo-fan                     5
## 5 AIRBUS INDUSTRIE A320-211       2   182 Turbo-jet                    31
## 6           AIRBUS A320-232       2   200 Turbo-fan                    57
##   sched_arr_time_num_minute sched_dep_time_minute sched_dep_time_num_minute
## 1                       505                     0                       360
## 2                       840                    50                       770
## 3                      1220                    15                      1035
## 4                      1325                    30                      1230
## 5                      1111                    25                       985
## 6                       717                     0                       540
##   sched_air_time dep_delay_flag temp_flag wind_dir_flag wind_speed_flag
## 1            145              0         0             0               0
## 2             70              0         0             0               0
## 3            185              0         0             0               0
## 4             95              0         0             0               0
## 5            126              0         0             0               0
## 6            177              0         0             0               0
##   precip_flag pressure_flag name_flag year.y_flag type_flag
## 1           0             0         0           0         0
## 2           0             0         0           1         1
## 3           0             0         0           0         0
## 4           0             0         0           0         0
## 5           0             0         0           0         0
## 6           0             1         0           0         0
```

```r
library(dplyr)
DF <- DF %>% mutate_if(is.numeric, scale)
head(DF)
```

```
##     month        day carrier flight origin dest   distance       temp
## 1 1.30322 -0.9929373      WN   1716    LGA  ATL -0.3777852  0.3339858
## 2 1.01019  1.6325235      AA    178    JFK  BOS -1.1644742  0.1127815
## 3 1.59625  0.2627179       DL   1585    LGA  MCO -0.1205721 -1.2848272
## 4 1.30322  0.4910188       WN   3494    EWR  MDW -0.4475611 -1.1138966
## 5 1.01019  0.6051693       DL   2231    LGA  DTW -0.7335054  0.3339858
## 6 1.30322 -0.9929373       B6     27    EWR  MCO -0.1383581  0.4144237
##         dewp      humid    wind_dir wind_speed     precip    pressure     visib
## 1  0.7418623  0.9315583  0.07717317  0.4871566 -0.1492223 -0.96830167 0.3664282
## 2  0.2753242  0.2375566  0.36735789 -0.3415806 -0.1492223  1.01596006 0.3664282
## 3 -1.2735821 -0.5125364  0.65754261  1.1087096 -0.1492223  0.28792159 0.3664282
## 4 -1.2269283 -0.7129351 -1.76066338 -0.9631336 -0.1492223  2.54341334 0.3664282
## 5 -0.0325909 -0.7695363 -0.40646802  0.4871566 -0.1492223 -0.12606108 0.3664282
## 6  0.7138700  0.9040226  0.36735789  0.9015253 -0.1492223 -0.04040949 0.3664282
##                                  name        lat       lon         alt
## 1    Hartsfield Jackson Atlanta Intl -0.4207546 0.3298674  0.48364704
## 2 General Edward Lawrence Logan Intl  1.1190951 1.2378347 -0.60619417
## 3                       Orlando Intl -1.3395034 0.5408516 -0.52285973
## 4                Chicago Midway Intl  1.0170501 0.1049977  0.04424731
## 5             Detroit Metro Wayne Co  1.0922942 0.4025621  0.07130394
## 6                       Orlando Intl -1.3395034 0.5408516 -0.52285973
##           tz dst            tzone      year.y                    type
## 1  0.6826595   A America/New_York -0.08500492 Fixed wing multi engine
## 2  0.6826595   A America/New_York  0.08617407 Fixed wing multi engine
## 3  0.6826595   A America/New_York  0.08617407 Fixed wing multi engine
## 4 -0.2514221   A  America/Chicago  0.77089000 Fixed wing multi engine
## 5  0.6826595   A America/New_York -1.62561576 Fixed wing multi engine
## 6  0.6826595   A America/New_York  0.77089000 Fixed wing multi engine
##       manufacturer    model    engines      seats    engine
## 1           BOEING  737-7H4 0.05879311 0.02232546 Turbo-fan
## 2           BOEING A320-232 0.05879311 0.15869100 Turbo-fan
## 3           AIRBUS A319-114 0.05879311 0.09808410 Turbo-fan
## 4           BOEING  737-7H4 0.05879311 0.02232546 Turbo-fan
## 5 AIRBUS INDUSTRIE A320-211 0.05879311 0.65869797 Turbo-jet
## 6           AIRBUS A320-232 0.05879311 0.93142905 Turbo-fan
##   sched_arr_time_minute sched_arr_time_num_minute sched_dep_time_minute
## 1            -0.2348938                -1.4325947           -1.36042229
## 2            -1.6716145                -0.3129587            1.23408583
## 3            -0.5222379                 0.9570761           -0.58206985
## 4            -1.3842703                 1.3080068            0.19628258
## 5             0.1099192                 0.5927766           -0.06316823
## 6             1.6041087                -0.7240489           -1.36042229
##   sched_dep_time_num_minute sched_air_time dep_delay_flag temp_flag
## 1                -1.6236293     0.14883297              0         0
## 2                -0.1673447    -0.24317221              0         0
## 3                 0.7739125     0.35790240              0         0
## 4                 1.4665357    -0.11250381              0         0
## 5                 0.5963168     0.04952499              0         0
## 6                -0.9842849     0.31608852              0         0
##   wind_dir_flag wind_speed_flag precip_flag pressure_flag name_flag year.y_flag
## 1             0               0           0             0         0           0
## 2             0               0           0             0         0           1
## 3             0               0           0             0         0           0
## 4             0               0           0             0         0           0
## 5             0               0           0             0         0           0
## 6             0               0           0             1         0           0
##   type_flag
```

Table 2: proportion of variance explained by each principal component

	x
PC1	0.1352687
PC2	0.1061946
PC3	0.0862791
PC4	0.0688860
PC5	0.0605903
PC6	0.0582950

```
## 1          0
## 2          1
## 3          0
## 4          0
## 5          0
## 6          0
```

```
DF$dep_delay <- dep_delay_vec
```

6.4 Exploratory Data Analysis

```
numeric_DF <- dplyr::select_if(DF, is.numeric) %>%  scale()
```

```
prcomp_res <- prcomp(numeric_DF)
sdev <- prcomp_res$sdev
sdev
```

```
##  [1] 1.801791e+00 1.596455e+00 1.438992e+00 1.285793e+00 1.205889e+00
##  [6] 1.182827e+00 1.096174e+00 1.028253e+00 1.017278e+00 9.998650e-01
## [11] 9.870519e-01 9.598095e-01 9.527413e-01 9.302188e-01 8.722020e-01
## [16] 8.678655e-01 8.004108e-01 7.322880e-01 7.049102e-01 6.392628e-01
## [21] 2.137689e-01 1.418198e-01 5.881124e-02 1.734179e-14
```

6.4.1 all four components at same time

proportion of variance explained by each component

```
pve <- colSums(prcomp_res$x^2)/sum(numeric_DF^2)
```

```
rotation <- as.data.frame(prcomp_res$rotation)
rotation[order(-abs(rotation$PC1)),]
```

```
##                                   PC1          PC2          PC3           PC4
## lon                         0.5342621313  0.008227987 -0.011812828 -0.0071811692
## distance                   -0.5303215501 -0.013672653  0.042091019  0.0006060091
## tz                          0.5152320734  0.003918041 -0.003819370 -0.0093898478
## seats                      -0.3267974610 -0.021806698  0.092397836  0.0075811906
## alt                        -0.1291184773  0.001274092 -0.022430973  0.0179064904
## sched_air_time             -0.1269319210 -0.059262301 -0.046756285 -0.0535120057
## year.y                      0.1124548236  0.025774490 -0.038449639  0.0056654703
## sched_arr_time_num_minute -0.0808327463 -0.139515851 -0.548184284 -0.2068187847
## lat                        0.0722096939  0.032190638 -0.080234329  0.0246355126
## dewp                      -0.0345819273  0.539843580 -0.222885964  0.1989663082
## temp                      -0.0291750632  0.384082292 -0.336092763  0.3735114971
## month                     -0.0237188202  0.182298459 -0.084197143  0.3005556332
## sched_arr_time_minute     -0.0215092361 -0.015025384 -0.089818353 -0.0429715263
## humid                     -0.0203751852  0.453383850  0.144953536 -0.3156006821
## engines                   -0.0127603053  0.004264643  0.001902439  0.0110691777
## wind_speed                -0.0082656281 -0.253349116 -0.146083514 -0.2029687361
## dep_delay                  0.0077853490  0.083641254 -0.226497841 -0.2377245680
## visib                      0.0071744619 -0.280198624 -0.146006623  0.5175332824
## pressure                   0.0051374481 -0.127426965  0.233410657  0.1117576955
## precip                    -0.0032312124  0.177264013  0.017625071 -0.4019773885
## wind_dir                   0.0031602295 -0.294478266 -0.112869407  0.0491076591
## sched_dep_time_minute     -0.0026413644  0.003439200 -0.121374246 -0.0057857207
## day                        0.0004234420 -0.029084254 -0.002485255  0.0805143860
## sched_dep_time_num_minute  0.0003538067 -0.107997556 -0.550807880 -0.1834313382
##                                    PC5          PC6          PC7          PC8
## lon                       -0.141255373  0.021194537  0.014171904 -0.023884973
## distance                  -0.042941576 -0.005942745 -0.122735618  0.043705253
## tz                        -0.207494095  0.030490047  0.001420389 -0.017609684
## seats                     -0.299154868 -0.009581806 -0.206563161 -0.144397469
## alt                        0.466078117 -0.062307363  0.241777658 -0.082760758
## sched_air_time            -0.299112698  0.151635298  0.696202208 -0.179246326
## year.y                     0.174145263  0.012033458 -0.193371095 -0.427597840
## sched_arr_time_num_minute -0.104506952  0.310826773  0.201433129 -0.047918183
## lat                        0.662475235 -0.078116941  0.239929946 -0.075262096
## dewp                      -0.078261453 -0.121386482  0.033430978  0.007641409
## temp                      -0.092120074 -0.132315189  0.049043959  0.010668654
## month                     -0.018610656  0.151704760 -0.001495157 -0.086444244
## sched_arr_time_minute     -0.039500802  0.101328352 -0.031997725 -0.024890953
## humid                      0.003708481 -0.016414325 -0.017906587 -0.005803837
## engines                   -0.044314253  0.009660787 -0.123433154 -0.848006590
## wind_speed                -0.061901823 -0.437801958  0.009361381 -0.047218482
## dep_delay                  0.045376993 -0.018945452 -0.127408457  0.067633220
## visib                     -0.007702427 -0.018742792 -0.026418970  0.024830862
## pressure                   0.116032120  0.563438391 -0.052924000 -0.001194143
## precip                    -0.003734675 -0.050856276  0.034962297 -0.050209716
## wind_dir                  -0.062792217 -0.485144428  0.012771217 -0.047966323
## sched_dep_time_minute      0.092015290  0.054589781 -0.390804984  0.021989590
## day                       -0.001689151  0.020937857 -0.047272003  0.037610315
## sched_dep_time_num_minute  0.092202353  0.227284354 -0.259042906  0.070884810
##                                    PC9         PC10        PC11         PC12
## lon                       -0.022482607  0.032770492 -0.09242396  0.035571265
## distance                   0.020078457 -0.041564282  0.11149965 -0.041739208
## tz                        -0.044272752  0.043751701 -0.10878566  0.038322354
## seats                      0.015630688  0.063366532 -0.24788381  0.086569086
## alt                       -0.138469996  0.146822298 -0.29987639  0.092299979
## sched_air_time            -0.149133954  0.035683038  0.19693563 -0.143550228
## year.y                    -0.034976443 -0.155852739  0.66240241 -0.262765846
## sched_arr_time_num_minute  0.053926504 -0.020110402  0.04077386 -0.025711336
## lat                        0.017805635  0.016011358 -0.03785986  0.012953704
## dewp                       0.001688937  0.004365652 -0.01228182 -0.051931274
## temp                       0.006200971 -0.013446913 -0.02987388 -0.002801481
## month                     -0.120787905 -0.156460256  0.19154435  0.625378579
## sched_arr_time_minute     -0.732058004  0.278259833 -0.14647214 -0.044943774
## humid                     -0.010455120  0.037697965  0.03421562 -0.091955092
## engines                    0.132086910  0.104896992 -0.32454087  0.021879150
## wind_speed                -0.071625458 -0.057666857  0.10354898  0.185290272
## dep_delay                  0.200325467  0.111865346 -0.09362580 -0.081177009
## visib                      0.032108130 -0.048673404 -0.05950128 -0.052108675
## pressure                   0.010837763 -0.093446997  0.03191945  0.236585784
## precip                    -0.076608798 -0.105843585  0.08898224  0.535791873
## wind_dir                  -0.088916746 -0.037067781  0.09755445  0.208042591
## sched_dep_time_minute     -0.502138130  0.043987210  0.09868480 -0.123135660
## day                        0.213140360  0.886240079  0.33176453  0.184965679
## sched_dep_time_num_minute  0.158656717 -0.045621287 -0.09049849  0.070227231
##                                   PC13         PC14         PC15         PC16
## lon                       -0.039446447 -0.011082224 -0.095304541  0.077597524
## distance                   0.039795147  0.002647274  0.039897734 -0.004261933
## tz                        -0.063720131 -0.035598494 -0.136768637  0.126354681
## seats                     -0.027325189  0.020728319 -0.252623268  0.247686911
## alt                       -0.170421919 -0.125274181 -0.455475123  0.483583152
## sched_air_time            -0.164479367 -0.208946245  0.041593996 -0.030363222
## year.y                     0.197515002  0.011913488 -0.161597312  0.354827113
## sched_arr_time_num_minute -0.077917477  0.109167190 -0.088599384 -0.027228430
## lat                        0.035792204  0.086517293  0.199298954 -0.284868167
## dewp                      -0.017049314  0.049682409  0.061863302  0.060756471
## temp                      -0.047848117  0.049638704  0.196084120  0.173608736
## month                      0.118816359 -0.207210166 -0.371210630 -0.334318394
## sched_arr_time_minute      0.556752791  0.125447499  0.111052124 -0.001069467
## humid                      0.040272692  0.026173139 -0.229179766 -0.191945954
## engines                   -0.057545879  0.002619518  0.178322651 -0.213145500
## wind_speed                 0.003326375 -0.006229440 -0.077318196 -0.076182657
## dep_delay                  0.290573141 -0.828553616  0.116709581  0.042342062
## visib                      0.056744264 -0.097207648  0.203736898  0.193394671
## pressure                   0.027199545 -0.128666719  0.184057892  0.108434211
## precip                    -0.102936363  0.077513423  0.479127840  0.404120615
## wind_dir                   0.042744908 -0.080419991 -0.006239399 -0.069929842
## sched_dep_time_minute     -0.670765113 -0.241104191  0.085424899 -0.145499061
## day                       -0.079973980  0.072516223  0.018126851  0.001947757
## sched_dep_time_num_minute  0.028967986  0.258010080 -0.122424720 -0.008303141
##                                    PC17         PC18          PC19         PC20
## lon                       -0.0136295128  0.017750740  0.0968378420 -0.013684012
## distance                   0.0205984456 -0.007328044 -0.1825459698  0.002448852
## tz                        -0.0051770963  0.018809032  0.0334504287 -0.017652697
## seats                     -0.0560405964  0.016440667  0.7255605863  0.013581073
## alt                        0.0190843976  0.036647205 -0.2601368362 -0.018726070
## sched_air_time            -0.0034980154 -0.004840517  0.0992775830 -0.034805534
## year.y                     0.0096125145 -0.009443594  0.0894488451  0.039095520
## sched_arr_time_num_minute  0.0562858424  0.035350187  0.0356674787 -0.080371931
## lat                       -0.0520614544 -0.012599070  0.5277297074 -0.001035736
## dewp                      -0.0942744599  0.231249850 -0.0056935536 -0.133117366
## temp                      -0.1050475638  0.211514725 -0.0068747953  0.258118643
## month                      0.0289369124 -0.245815406 -0.0116573597  0.055270714
## sched_arr_time_minute     -0.0117347846 -0.001681325 -0.0254211838  0.016161871
## humid                      0.0025024154  0.129312624 -0.0006039130 -0.656492439
## engines                    0.0086998591 -0.003776375 -0.2195595169 -0.011263309
## wind_speed                -0.7670490408  0.118367572 -0.0704672685 -0.051948884
## dep_delay                 -0.0005171053 -0.022901277  0.0568514930  0.061563276
## visib                     -0.0735870801 -0.324502693  0.0148202229 -0.645786881
## pressure                  -0.2197533179  0.626620074  0.0007539589 -0.115345022
## precip                     0.1120149351 -0.238971520  0.0196643203 -0.093284192
## wind_dir                   0.5566040091  0.499155917  0.0416506799 -0.142598451
## sched_dep_time_minute      0.0036919766 -0.018413525  0.0629338235 -0.020942773
## day                       -0.0242345214 -0.001266925  0.0008713042 -0.029416910
## sched_dep_time_num_minute  0.0621948131  0.040857801 -0.0295601569 -0.061762457
##                                   PC21          PC22          PC23
## lon                        0.051203785  0.8076315701  2.289417e-03
## distance                  -0.673700342  0.4367623709 -5.904235e-04
## tz                        -0.692935877 -0.3885478079 -2.387247e-03
## seats                      0.035356839 -0.0141846176  3.536570e-05
## alt                        0.004741932  0.0333354457  1.708179e-04
## sched_air_time            -0.011013975  0.0180351813  6.012199e-04
## year.y                     0.005825894 -0.0149597542 -4.767877e-04
## sched_arr_time_num_minute -0.006289146  0.0065552063 -1.225185e-03
## lat                       -0.247687309  0.0630062661 -3.594062e-04
## dewp                       0.003122233  0.0006517912 -7.090823e-01
## temp                      -0.005663154  0.0002352898  6.113164e-01
## month                     -0.001646308  0.0024681356  2.750428e-03
## sched_arr_time_minute      0.004516970  0.0014876945  2.250719e-05
## humid                      0.009389784 -0.0061721256  3.494391e-01
## engines                   -0.006154972  0.0044264994 -3.645510e-05
## wind_speed                 0.001061596 -0.0025257566 -5.755781e-04
## dep_delay                 -0.001497187  0.0024853904 -1.169113e-03
## visib                      0.003848279 -0.0010054753  3.640126e-02
## pressure                   0.002485028 -0.0027882656  1.438635e-03
## precip                    -0.002132669  0.0014480599 -5.757478e-03
## wind_dir                   0.003375411 -0.0014023794 -3.540807e-04
## sched_dep_time_minute      0.017182271  0.0024772274 -8.010616e-04
## day                        0.001530860 -0.0004536717 -2.571908e-04
## sched_dep_time_num_minute  0.000800948 -0.0052895688 -1.710632e-03
##                                    PC24
## lon                        1.073158e-14
## distance                   6.778678e-15
## tz                        -7.737005e-16
## seats                      2.744135e-16
## alt                       -8.633487e-17
## sched_air_time            -4.221606e-01
## year.y                     4.918863e-16
## sched_arr_time_num_minute  6.602011e-01
## lat                        2.167659e-15
## dewp                      -1.348152e-15
## temp                      -8.870884e-16
## month                      2.656342e-16
## sched_arr_time_minute     -1.997126e-15
## humid                      2.401847e-15
## engines                   -1.067216e-16
## wind_speed                -2.446632e-15
## dep_delay                  5.015833e-17
## visib                      5.676978e-16
## pressure                   1.661713e-15
## precip                    -8.702010e-16
```

Table 3: coefficients for each variable on each principal component

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
dewp	-0.0345819	0.5398436	-0.2228860	0.1989663	-0.0782615	-0.1213865	0.0334310	0.0076414
humid	-0.0203752	0.4533838	0.1449535	-0.3156007	0.0037085	-0.0164143	-0.0179066	-0.0058038
temp	-0.0291751	0.3840823	-0.3360928	0.3735115	-0.0921201	-0.1323152	0.0490440	0.0106687
wind_dir	0.0031602	-0.2944783	-0.1128694	0.0491077	-0.0627922	-0.4851444	0.0127712	-0.0479663
visib	0.0071745	-0.2801986	-0.1460066	0.5175333	-0.0077024	-0.0187428	-0.0264190	0.0248309
wind_speed	-0.0082656	-0.2533491	-0.1460835	-0.2029687	-0.0619018	-0.4378020	0.0093614	-0.0472185

```
## wind_dir                     1.569069e-15
## sched_dep_time_minute        1.336574e-15
## day                         -3.738223e-16
## sched_dep_time_num_minute -6.212206e-01
```

```
pca_rotation <- head(rotation[order(-abs(rotation$PC2)),])
```

6.4.2 take out extreme departure delays

```
DF<-DF[DF$dep_delay < 30,]
```

```
set.seed(42)
DF$flight <- NULL
train_index <- sample(1:nrow(DF),size=2*nrow(DF)/3,replace=FALSE)
train_df <- DF[train_index,]
test_df <- DF[-train_index,]
```

```
# pre-allocate space
preallocate_df <- function(n){
  df <- data.frame(model_description = character(n), rmse = numeric(n), stringsAsFactors = FALSE)
  for(i in 1:n){
    df$model_description[i] <- i
    df$rmse[i] <- toString(i)
  }
  df
}
```

7 predicting 0

```
benchmark_df <- data.frame(model_description = character(), rmse = numeric(), stringsAsFactors = FALSE)
rmse = mean((test_df$dep_delay-0)^2) %>% sqrt()
model_description = "predicting 0"
benchmark_df <- rbind(benchmark_df, data.frame(model_description = model_description, rmse=rmse))
```

8 predicting the mean

```
rmse = mean((test_df$dep_delay-mean(train_df$dep_delay))^2)%>% sqrt()
rmse
```

```
## [1] 8.299767
```

```
benchmark_df
```

```
##   model_description    rmse
## 1      predicting 0 8.30571
```

9 predicting the median

```
rmse = mean((test_df$dep_delay-median(train_df$dep_delay))^2)%>% sqrt()
rmse
```

```
## [1] 8.469257
```

```
model_description <- 'predicting the median'
benchmark_df <- rbind(benchmark_df, data.frame(model_description = model_description, rmse=rmse))
```

10 linear regression with dep

```
model <- lm(dep_delay ~ ., data=train_df)
model_without_dep <-  lm(dep_delay ~ .-dest, data=train_df)
anova(model, model_without_dep)
```

```
## Analysis of Variance Table
##
## Model 1: dep_delay ~ month + day + carrier + origin + dest + distance +
##     temp + dewp + humid + wind_dir + wind_speed + precip + pressure +
##     visib + name + lat + lon + alt + tz + dst + tzone + year.y +
##     type + manufacturer + model + engines + seats + engine +
##     sched_arr_time_minute + sched_arr_time_num_minute + sched_dep_time_minute +
##     sched_dep_time_num_minute + sched_air_time + dep_delay_flag +
##     temp_flag + wind_dir_flag + wind_speed_flag + precip_flag +
##     pressure_flag + name_flag + year.y_flag + type_flag
## Model 2: dep_delay ~ (month + day + carrier + origin + dest + distance +
##     temp + dewp + humid + wind_dir + wind_speed + precip + pressure +
##     visib + name + lat + lon + alt + tz + dst + tzone + year.y +
##     type + manufacturer + model + engines + seats + engine +
##     sched_arr_time_minute + sched_arr_time_num_minute + sched_dep_time_minute +
##     sched_dep_time_num_minute + sched_air_time + dep_delay_flag +
##     temp_flag + wind_dir_flag + wind_speed_flag + precip_flag +
##     pressure_flag + name_flag + year.y_flag + type_flag) - dest
##   Res.Df     RSS Df Sum of Sq     F    Pr(>F)
## 1 113488 7240975
## 2 113491 7242239 -3   -1263.7 6.602 0.0001863 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary <- round(summary(model)$coefficients,6)
sorteddf <- summary[order(summary[,ncol(summary)]),]
head(sorteddf)
```

```
##              Estimate Std. Error    t value Pr(>|t|)
## carrierFL    4.615319   0.887510   5.200301        0
## carrierUA    2.440853   0.484369   5.039237        0
## originJFK    1.037568   0.117779   8.809419        0
## wind_speed   0.291278   0.027929  10.429081        0
## precip       0.206814   0.029257   7.068777        0
## pressure    -0.294549   0.027059 -10.885462        0
```

```
head(sorteddf)
```

```
##              Estimate Std. Error    t value Pr(>|t|)
## carrierFL    4.615319   0.887510   5.200301        0
## carrierUA    2.440853   0.484369   5.039237        0
## originJFK    1.037568   0.117779   8.809419        0
## wind_speed   0.291278   0.027929  10.429081        0
## precip       0.206814   0.029257   7.068777        0
## pressure    -0.294549   0.027059 -10.885462        0
```

```
lm_test_df <- test_df

in_test_but_not_train <- setdiff(unique(lm_test_df$model), unique(train_df$model))
lm_test_df <- lm_test_df[ !lm_test_df$model %in% in_test_but_not_train, ]

in_test_but_not_train <- setdiff(unique(lm_test_df$dest), unique(train_df$dest))
lm_test_df <- lm_test_df[ !lm_test_df$dest %in% in_test_but_not_train, ]

preds = predict(model, newdata=lm_test_df)
```

```
## Warning in predict.lm(model, newdata = lm_test_df): prediction from a rank-
## deficient fit may be misleading
```

```
rmse = sqrt(mean((lm_test_df$dep_delay - preds)^2))
rmse
```

```
## [1] 7.989994
```

```
model_description <- 'linear regression'
benchmark_df <- rbind(benchmark_df, data.frame(model_description = model_description, rmse=rmse))
```

11 gbm

```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

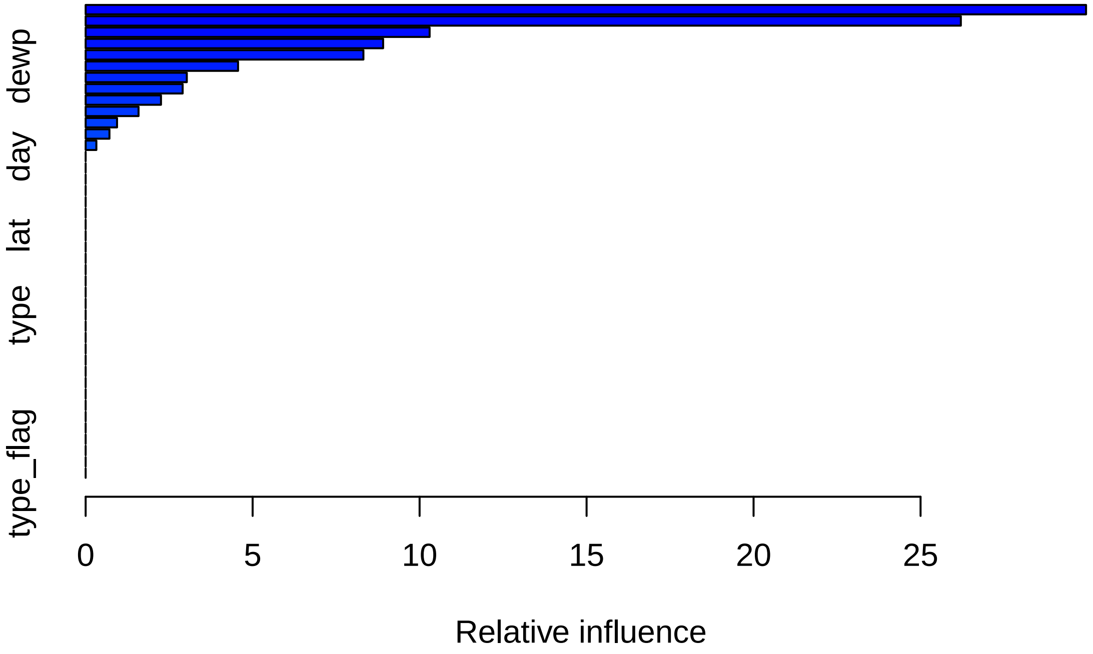
```
train_gbm <- function(filename){
set.seed(42)
model <- gbm(dep_delay ~ ., data=train_df,
              n.trees=100, shrinkage=0.1) # default shrinkage = 0.1
preds = predict(model, newdata=test_df, n.trees=100)
rmse = sqrt(mean((test_df$dep_delay - preds)^2))
```

Table 4: gbm relative influence

	var	rel.inf
sched_dep_time_num_minute	sched_dep_time_num_minute	29.955850
model	model	26.206955
dest	dest	10.299706
carrier	carrier	8.907853
month	month	8.317570
dewp	dewp	4.563640

```
summary(model)
saveRDS(model, filename)
return(model)
}

destfile <- "models/model2.rds"
if (!file.exists(destfile)) {
   train_gbm(destfile)
 }
model <- readRDS(destfile)
```

```
benchmark_df <- rbind(benchmark_df, data.frame(model_description = 'gbm', rmse=7.94458))
```



Here, you can see the relative influence for each variable for gbm.
For a gbm, the improvement in the splitting criterion (which is mean squared error for regression) for a given variable is calculated at each step. The relative influence for a given variable is the average of these improvements over all the trees where the aforementioned variable is used.

```
model <- gbm(dep_delay ~ ., data=train_df,
              n.trees=100, shrinkage=0.01) # default shrinkage = 0.1
```

```
## Distribution not specified, assuming gaussian ...
```

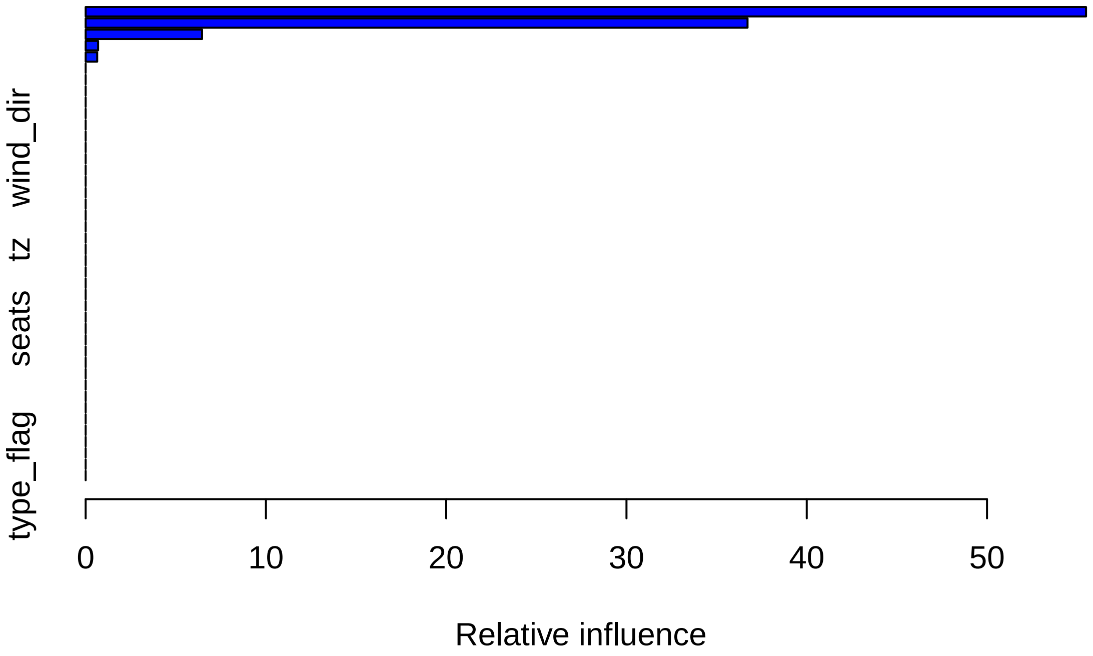
```
preds = predict(model, newdata=test_df, n.trees=1000)
```

```
## Warning in predict.gbm(model, newdata = test_df, n.trees = 1000): Number of
## trees not specified or exceeded number fit so far. Using 100.
```

```
filename <- "models/gbm_shrinkage_0point01_ntrees_100_v1.rds"
saveRDS(model, filename)
rmse = sqrt(mean((test_df$dep_delay - preds)^2))
rmse
```

```
## [1] 8.179232
```

```
summary(model)
```



```
##                                             var    rel.inf
## sched_dep_time_num_minute sched_dep_time_num_minute 55.4932927
## model                                     model 36.7164030
## carrier                                 carrier  6.4586978
## dest                                       dest  0.6914379
## sched_arr_time_num_minute sched_arr_time_num_minute  0.6401686
## month                                     month  0.0000000
## day                                         day  0.0000000
## origin                                   origin  0.0000000
## distance                               distance  0.0000000
## temp                                       temp  0.0000000
## dewp                                       dewp  0.0000000
## humid                                     humid  0.0000000
## wind_dir                               wind_dir  0.0000000
## wind_speed                           wind_speed  0.0000000
## precip                                   precip  0.0000000
## pressure                               pressure  0.0000000
## visib                                     visib  0.0000000
## name                                       name  0.0000000
## lat                                             lat  0.0000000
## lon                                             lon  0.0000000
## alt                                             alt  0.0000000
## tz                                               tz  0.0000000
## dst                                             dst  0.0000000
## tzone                                         tzone  0.0000000
## year.y                                       year.y  0.0000000
## type                                           type  0.0000000
## manufacturer                           manufacturer  0.0000000
## engines                                     engines  0.0000000
## seats                                         seats  0.0000000
## engine                                       engine  0.0000000
## sched_arr_time_minute         sched_arr_time_minute  0.0000000
## sched_dep_time_minute         sched_dep_time_minute  0.0000000
## sched_air_time                       sched_air_time  0.0000000
## dep_delay_flag                       dep_delay_flag  0.0000000
## temp_flag                                 temp_flag  0.0000000
## wind_dir_flag                         wind_dir_flag  0.0000000
## wind_speed_flag                     wind_speed_flag  0.0000000
## precip_flag                             precip_flag  0.0000000
## pressure_flag                         pressure_flag  0.0000000
## name_flag                                 name_flag  0.0000000
## year.y_flag                             year.y_flag  0.0000000
## type_flag                                 type_flag  0.0000000
```

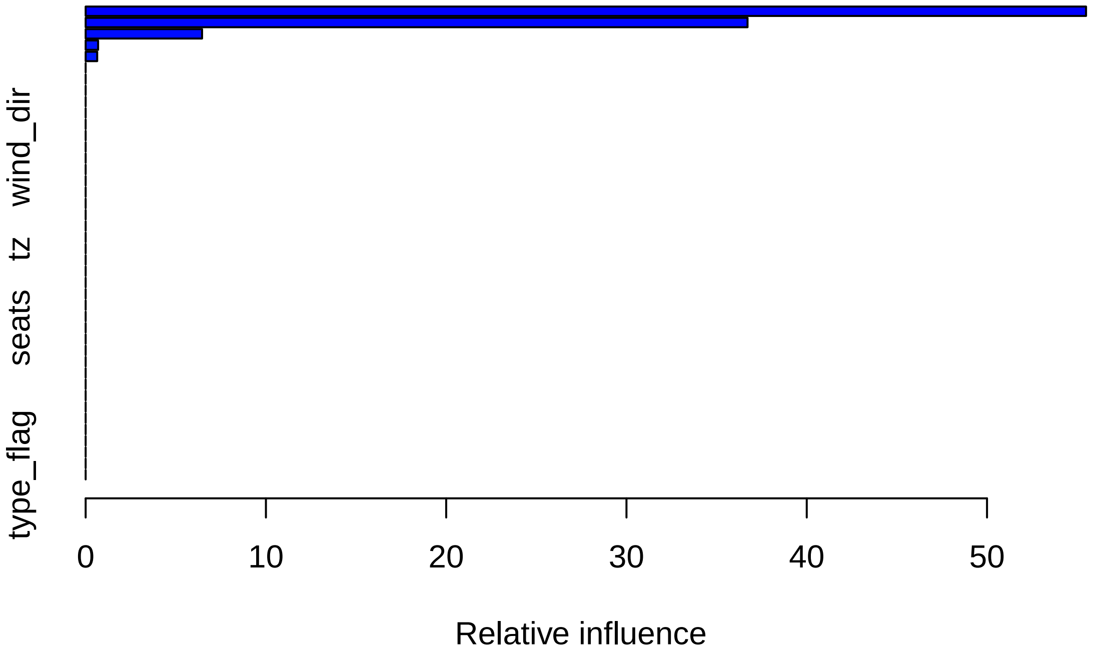
```r
library(gbm)
filename <- "models/gbm_shrinkage_0point01_ntrees_100_v1.rds"
if (!file.exists(filename)) {
  model <- gbm(
    dep_delay ~ .,
    data = train_df,
    n.trees = 100,
    shrinkage = 0.01
  ) # default shrinkage = 0.1
  saveRDS(model, filename)
} else {
  print("reading saved model")
  model <- readRDS(filename)
}
```

```
## [1] "reading saved model"
```

```r
preds = predict(model, newdata = test_df, n.trees = 100)
rmse = sqrt(mean((test_df$dep_delay - preds) ^ 2))
rmse
```

```
## [1] 8.179232
```

```r
summary(model)
```



```
##                                                var    rel.inf
## sched_dep_time_num_minute sched_dep_time_num_minute 55.4932927
## model                                         model 36.7164030
## carrier                                     carrier  6.4586978
## dest                                           dest  0.6914379
## sched_arr_time_num_minute sched_arr_time_num_minute  0.6401686
## month                                         month  0.0000000
## day                                             day  0.0000000
## origin                                       origin  0.0000000
## distance                                   distance  0.0000000
## temp                                           temp  0.0000000
## dewp                                           dewp  0.0000000
## humid                                         humid  0.0000000
## wind_dir                                   wind_dir  0.0000000
## wind_speed                               wind_speed  0.0000000
## precip                                       precip  0.0000000
## pressure                                   pressure  0.0000000
## visib                                         visib  0.0000000
## name                                           name  0.0000000
## lat                                             lat  0.0000000
## lon                                             lon  0.0000000
## alt                                             alt  0.0000000
## tz                                               tz  0.0000000
## dst                                             dst  0.0000000
## tzone                                         tzone  0.0000000
## year.y                                       year.y  0.0000000
## type                                           type  0.0000000
## manufacturer                           manufacturer  0.0000000
## engines                                     engines  0.0000000
## seats                                         seats  0.0000000
## engine                                       engine  0.0000000
## sched_arr_time_minute         sched_arr_time_minute  0.0000000
## sched_dep_time_minute         sched_dep_time_minute  0.0000000
## sched_air_time                       sched_air_time  0.0000000
## dep_delay_flag                       dep_delay_flag  0.0000000
## temp_flag                                 temp_flag  0.0000000
## wind_dir_flag                         wind_dir_flag  0.0000000
## wind_speed_flag                     wind_speed_flag  0.0000000
## precip_flag                             precip_flag  0.0000000
## pressure_flag                         pressure_flag  0.0000000
## name_flag                                 name_flag  0.0000000
## year.y_flag                             year.y_flag  0.0000000
## type_flag                                 type_flag  0.0000000
```

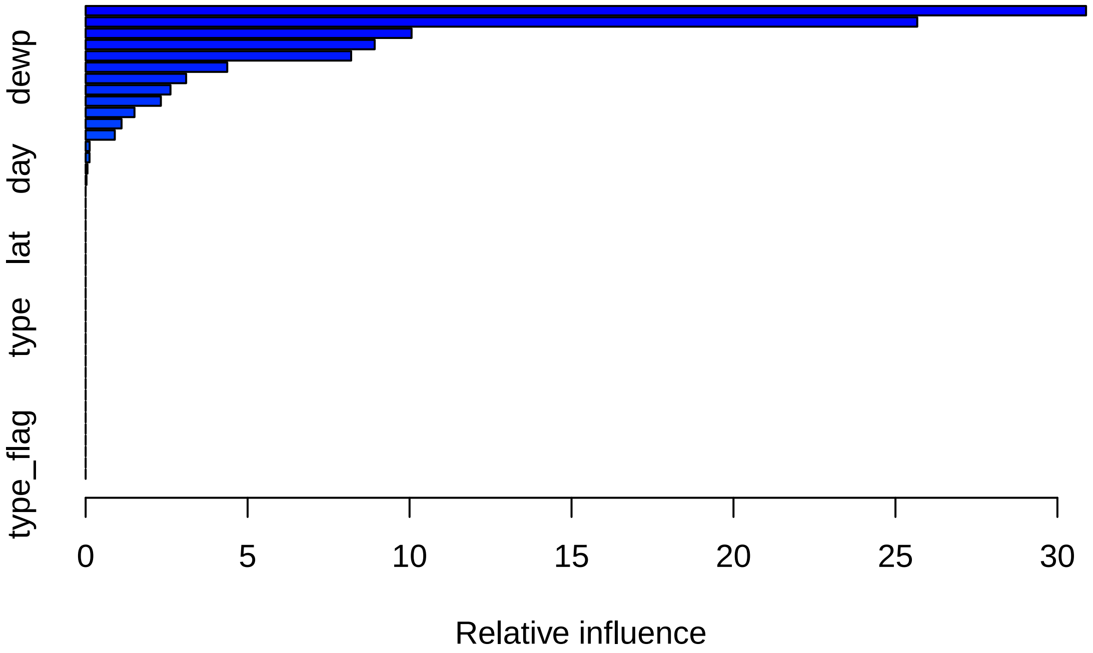
```
rmse = sqrt(mean((test_df$dep_delay - preds)^2))
rmse
```

```
## [1] 8.179232
```

```
set.seed(42)
x <- 2 ^ seq(5, 14, by = 1)
rmse_vec <- numeric(length(x))
count <- 1
for (val in x) {
  filename <-
    paste0("models/gbm_shrinkage_0point001_ntrees_", val)
  filename <- paste0(filename, "_v1.rds")
  if (!file.exists(filename)) {
    hboost <- gbm(
      dep_delay ~ .,
      data = train_df,
      n.trees = val,
      distribution = 'gaussian',
      shrinkage = 0.001
    )
    saveRDS(hboost, filename)
  } else{
    hboost <- readRDS(filename)
  }

  preds = predict(hboost, n.trees = val, newdata = test_df)
  mse = mean((test_df$dep_delay - preds) ^ 2)
  rmse <- sqrt(mse)
  rmse_vec[count] <- rmse
  print(val)
  print(rmse)
  count = count + 1
}
```

```
## [1] 32
## [1] 8.247971
## [1] 64
## [1] 8.211231
## [1] 128
## [1] 8.159652
## [1] 256
## [1] 8.101211
## [1] 512
## [1] 8.038247
## Warning in predict.gbm(hboost, n.trees = val, newdata = test_df): Number of
## trees not specified or exceeded number fit so far. Using 1000.

## [1] 1024
## [1] 7.980282

## Warning in predict.gbm(hboost, n.trees = val, newdata = test_df): Number of
## trees not specified or exceeded number fit so far. Using 1000.

## [1] 2048
## [1] 7.980282

## Warning in predict.gbm(hboost, n.trees = val, newdata = test_df): Number of
## trees not specified or exceeded number fit so far. Using 1000.

## [1] 4096
## [1] 7.980282

## Warning in predict.gbm(hboost, n.trees = val, newdata = test_df): Number of
## trees not specified or exceeded number fit so far. Using 1000.

## [1] 8192
## [1] 7.980282

## Warning in predict.gbm(hboost, n.trees = val, newdata = test_df): Number of
## trees not specified or exceeded number fit so far. Using 1000.

## [1] 16384
## [1] 7.980282
```

```r
summary(hboost)
```

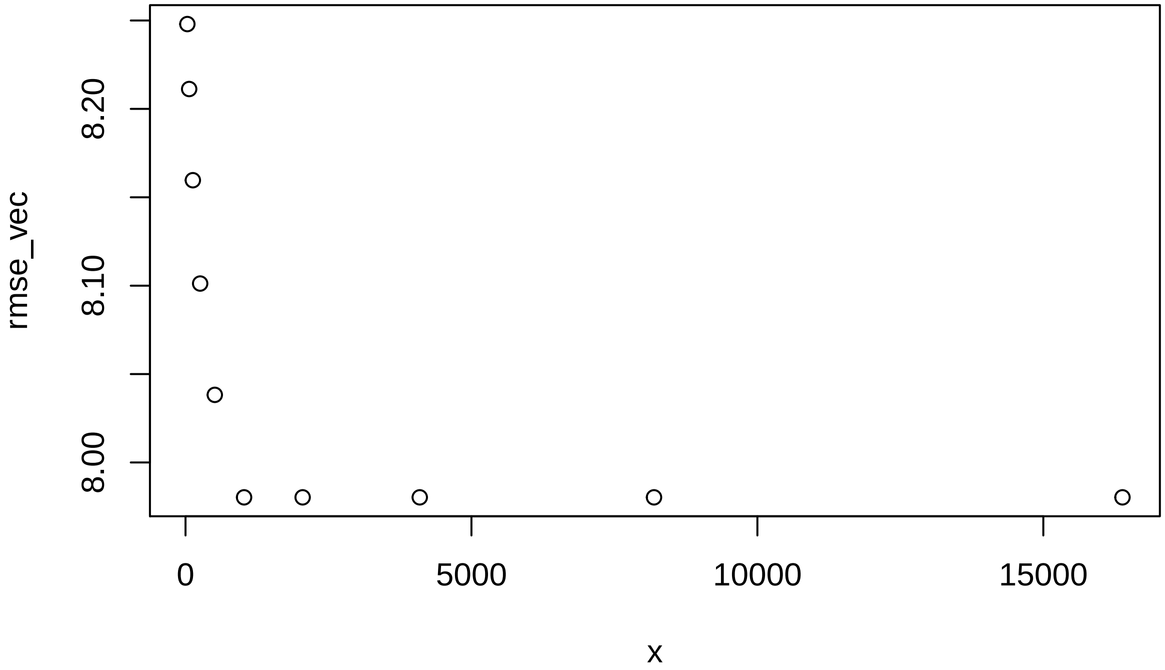


```
##                                                 var     rel.inf
## sched_dep_time_num_minute sched_dep_time_num_minute 30.88424567
## model                                         model 25.67333688
## dest                                           dest 10.06233500
## carrier                                     carrier  8.92374891
## month                                         month  8.19668822
## dewp                                           dewp  4.37183782
## origin                                       origin  3.10232960
## sched_arr_time_num_minute sched_arr_time_num_minute  2.61899718
## precip                                       precip  2.32288426
## pressure                                   pressure  1.50793482
## humid                                         humid  1.10879519
## dep_delay_flag                       dep_delay_flag  0.90034697
## temp                                           temp  0.12324130
## pressure_flag                         pressure_flag  0.11994578
## day                                             day  0.05664389
## sched_dep_time_minute         sched_dep_time_minute  0.02668850
## distance                                   distance  0.00000000
## wind_dir                                   wind_dir  0.00000000
## wind_speed                               wind_speed  0.00000000
## visib                                         visib  0.00000000
## name                                           name  0.00000000
## lat                                             lat  0.00000000
## lon                                             lon  0.00000000
## alt                                             alt  0.00000000
## tz                                               tz  0.00000000
## dst                                             dst  0.00000000
## tzone                                         tzone  0.00000000
## year.y                                       year.y  0.00000000
## type                                           type  0.00000000
## manufacturer                           manufacturer  0.00000000
## engines                                     engines  0.00000000
## seats                                         seats  0.00000000
## engine                                       engine  0.00000000
## sched_arr_time_minute         sched_arr_time_minute  0.00000000
## sched_air_time                       sched_air_time  0.00000000
## temp_flag                                 temp_flag  0.00000000
## wind_dir_flag                         wind_dir_flag  0.00000000
## wind_speed_flag                     wind_speed_flag  0.00000000
## precip_flag                             precip_flag  0.00000000
## name_flag                                 name_flag  0.00000000
## year.y_flag                             year.y_flag  0.00000000
## type_flag                                 type_flag  0.00000000
```

```
write.csv(summary,
          'performance/gbm_shrinkage_0point001_16384trees_summary_v1.csv')

plot(x, rmse_vec)
```

Table 5: benchmark comparing all the models that we tried

model_description	rmse
predicting 0	8.305710
predicting the median	8.469257
linear regression	7.989994
gbm	7.944580



```
num_trees_vs_rmse <- data.frame("num_trees" = x, "rmse" = rmse_vec)
write.csv(
  num_trees_vs_rmse,
  'performance/gbm_shrinkage_0point001_num_trees_vs_rmse_v1.csv'
)
```

References

"Gradient Boosting Machines · UC Business Analytics R Programming Guide." 2019. http://uc-r.github.io/gbm_regression#h2o.