Stats 652 project Name: Jaideep Misra Student Number:301368653

Flight delay prediction

Introduction

This project deals with the objective of predicting departure delays of flights in the USA which is an everyday problem and could come in handy for saving valuable time. To take into consideration various factors which are suspected to contribute to delaying, the data has been constructed from flights departing from NYC, meteorological data for each airport, specification of the aeroplane and airport location. Since flights are often delayed, this analysis can be useful to get insights such as what season is the worst to fly out of NYC or which airport has the most delays and on an average what are the departure delay for a given day.

Data

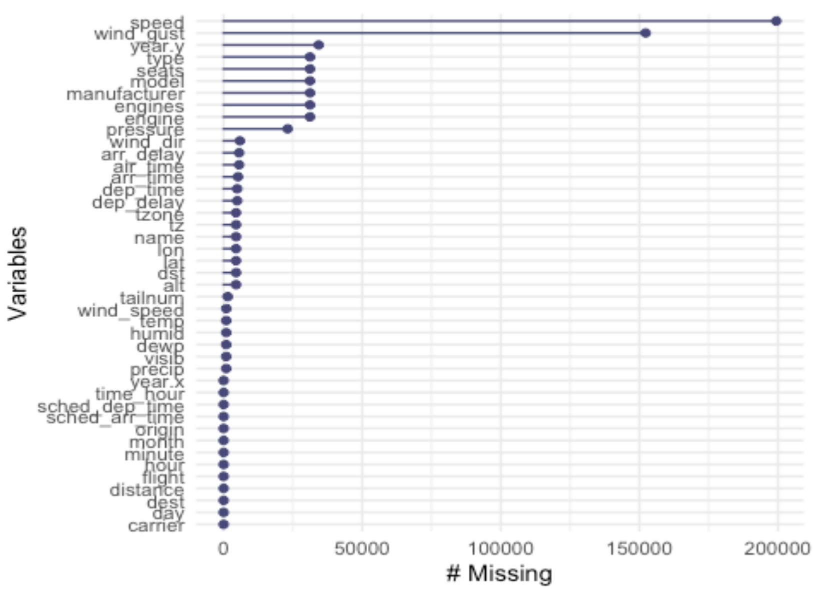
The data nycflights13 is from 3 New York city airports JFK, LGA and EWR in 2013 which contains 336,776 flights in total. This data has been combined with the following 4 datasets to get more factors responsible for delays in departures:

flights: Contains 336776 records and 19 features.
weather: Contains hourly meteorological data for the 3 airports i.e. LGA, JFK and EWR. Having a total 26115 records with 15
airports: This dataset contains the airport location information and has a total of 1458 records with 8 features. planes: Has information about all the aircraft specifications such as seats, engines etc.

Methods

1) EDA

After combining the data, we first check the number of missing values in our data as this can be a problem while generating our predictive models. For this I used the “ggplot2” library and generated the following visiualisation:

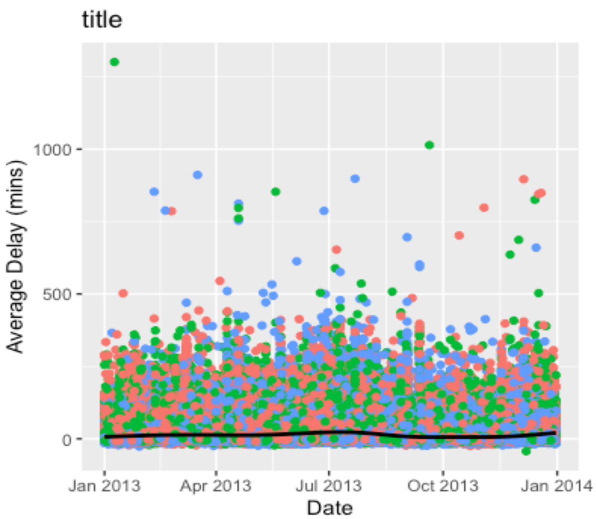


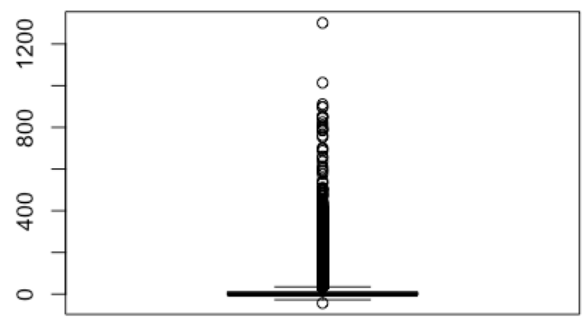
From the basic rule of thumb, we discard any feature having more than 5% i.e.10,000 records of missing values. Therefore, we drop the top 10 variables in the plot above. Next we impute the missing values in the weather data with the mean under the assumption that these values are missing completely at random. Thus, we can either remove these rows or impute these values. I chose to impute these values.

After imputing we remove any rows with NA in other features such as dep_delay which we can't predict as it is our target variable.

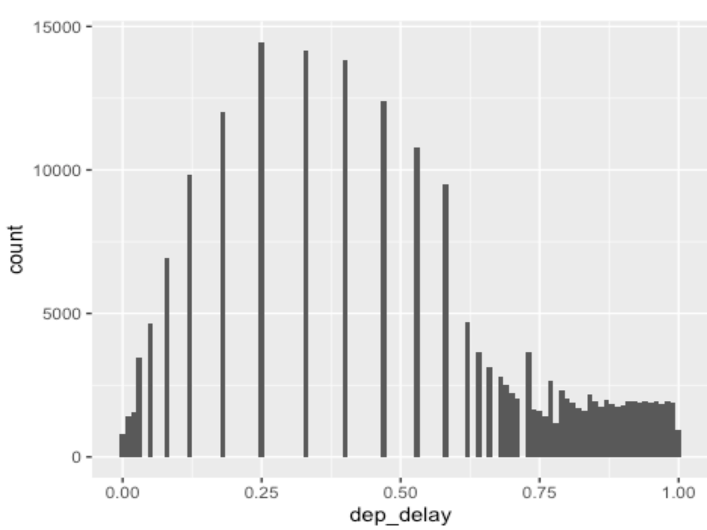
2) Feature Engineering

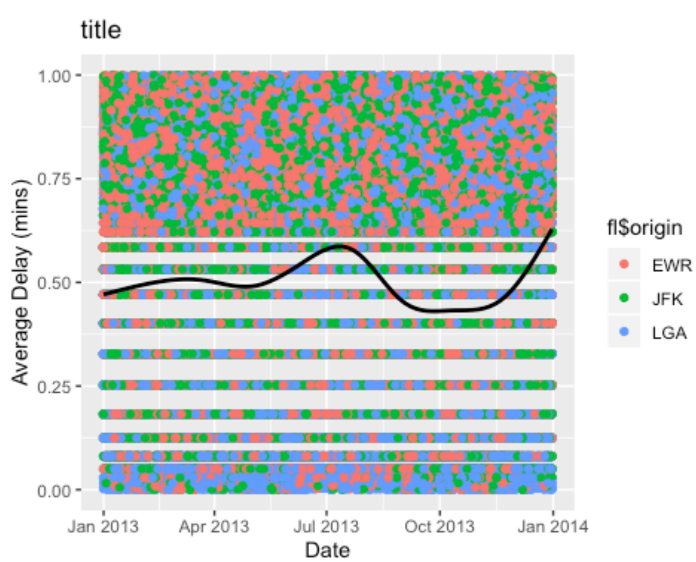
Departure Delay vs Seasons



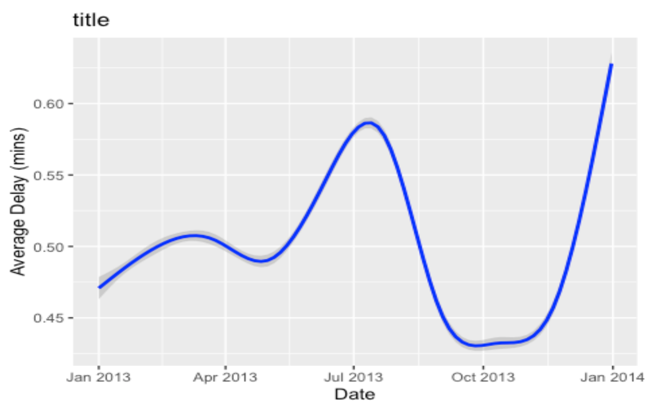


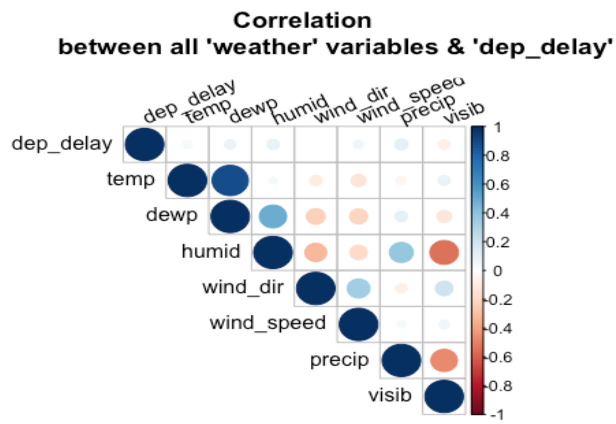
From the plot it can be seen that we can't see any variation in dep_delay across seasons let's have a closer look at dep_delay.
For this I did a box plot to see how the values range in our target variable dep_delay.
As it can be seen that majority of the values lie outside the box and thus making the distribution right skewed with some outliers. As outliers can impact the training of our model, we are going to map these values to quantiles of the standard normal (like grading departure delays on a "curve"), or mapping to ranks. we do the same plot again to see the variation of dep_delay.





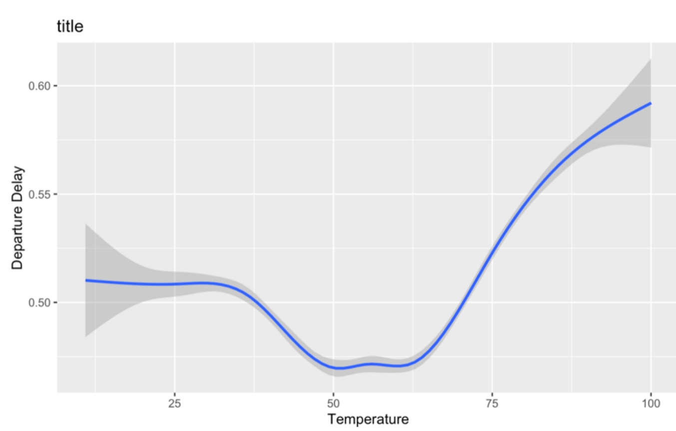
On zooming we can see that there is a clear seasonal trend in the data showing that there are huge delays in flights during the summer season and Holiday season (December)and thus we can introduce a new feature for seasons/quarters.

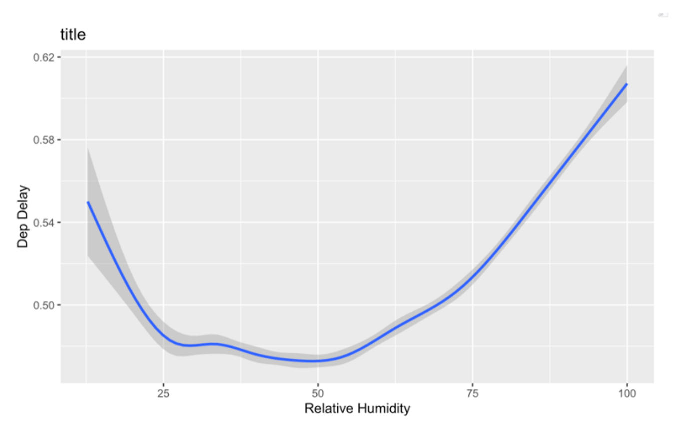




Departure Delay vs Weather

It can be seen that when there is high humidity i.e. in summer there is high delays in departure. Similar observation can be verified from the temperature graph. This is I tune with the seasonality graphs i.e. in summer there will be high temperature and occasional rains would cause high humidity leading to high delays.





3) Prediction Methods

For the prediction I tried 5 different models:

1) Linear Regression: Dropped certain features which were highly correlated such as dewp and temp and other non-relevant features based on p-value
2) Ridge Regression: Trained with best lambda value 0.006737716
3) Lasso Regression: Trained with best lambda value 0.006737716
4) GAM
5) GBM

Out of these GBM performed the best, with Lasso and Linear Regression performing the worst.

GBM model used:

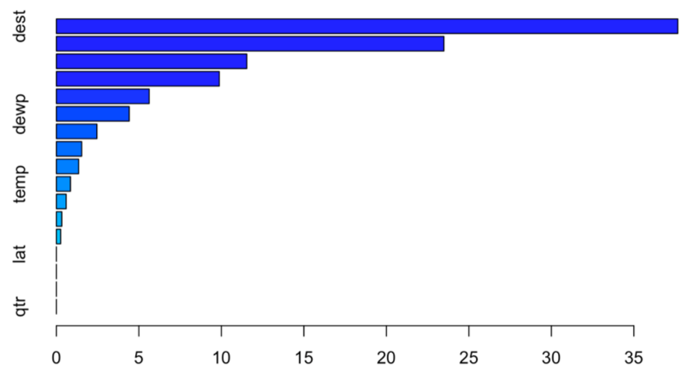
```
library(gbm)
dep_date_numeric <- as.numeric(fl_tr$dep_date)
dep_date_numeric <- dep_date_numeric - mean(dep_date_numeric)
fl_tr_tem <- mutate(fl_tr,dep_date = dep_date_numeric)
gbm_fit <-gbm(dep_delay ~ .,data=fl_tr_tem,distribution="gaussian",
               n.trees = 1000, shrinkage = 0.2)
summary(gbm_fit)
```

I reduced the shrinkage value from 0.001 to 0.2 which provided better results.

Results

GBM gave the minimum MSE on the holdout set giving 0.06939837

	var <fctr>	rel.inf <dbl>
dest	dest	37.6644921
sched_dep_time	sched_dep_time	23.4873864
carrier	carrier	11.5425397
dep_date	dep_date	9.8769189
humid	humid	5.6268465
dewp	dewp	4.4172575
precip	precip	2.4590047
origin	origin	1.5350780
wind_speed	wind_speed	1.3499554
temp	temp	0.8547841



Prediction Method	Validation MSE	Test MSE
Linear Regression	0.0725943	0.07268724
Ridge Regression	0.07214063	0.07226757
Lasso Regression	0.07327871	0.07341612
GAM	0.07023355	0.07038877
GBM	0.06932153	0.06939837

Conclusions and Discussion

From the results obtained from the GBM model it is evident that the top 3 factors which contribute to departure delay are destination, Scheduled departure time and the carrier.

Thus, if someone wanted to avoid taking delayed flights in 2013, one should have selected flights during winter which depart in the morning and should avoid carriers EV (Atlantic Southeast Airlines), WN (Southwest Airlines) and destinations such as **St Louis** Lambert International **Airport, Detroit Metropolitan Airport.**

- **Future work**

Improving accuracy of the model by implementing XGBoost model on the same features.

```r
knitr::opts_chunk$set(echo = TRUE,warning=FALSE,message=FALSE)
#R version 3.6.1 (2019-07-05)
#packages
used:"tidyverse","nycflights13","dplyr","ggplot2","naniar","lubridate","corrp
lot","zoo","gam","gbm","glmnet".
#Estimated Knit time: 2 mins

#1 Libraries
library(tidyverse)
library(nycflights13)
library(dplyr)
library(ggplot2)
library(naniar)
library(lubridate)
library(corrplot)
library(zoo)
library(gam)
library(gbm)
library(glmnet)

#2 Input Files
fltrain <- read_csv("fltrain.csv.gz") # Train set
fltest <- read_csv("fltest.csv.gz") # Holdout test set
#3 Feature Engineering and EDA
#For converting categorical variable to factors in train and test set
fl <- fltrain
fl_test = fltest
for(i in 1:ncol(fl)) {
  if(typeof(fl[[i]]) == "character") {
    fl[[i]] <- factor(fl[[i]])
  }
}
for(i in 1:ncol(fl_test)) {
  if(typeof(fl_test[[i]]) == "character") {
    fl_test[[i]] <- factor(fl_test[[i]])
  }
}
#visualising the amount of missing values in train and test set
gg_miss_var(fl)
gg_miss_var(fl_test)
#Imputing missing values in weather data
df = fl %>%
  mutate(wind_speed = ifelse(is.na(wind_speed), mean(wind_speed, na.rm =
TRUE), wind_speed),
         wind_dir = ifelse(is.na(wind_dir), mean(wind_dir, na.rm = TRUE),
wind_dir),
         humid = ifelse(is.na(humid), mean(humid, na.rm = TRUE), humid),
         dewp = ifelse(is.na(dewp), mean(dewp, na.rm = TRUE), dewp),
         temp = ifelse(is.na(temp), mean(temp, na.rm = TRUE), temp),
         precip = ifelse(is.na(precip), mean(precip, na.rm = TRUE), precip),
```

```
        visib = ifelse(is.na(visib), mean(visib, na.rm = TRUE), visib))

df_test = fl_test %>%
  mutate(wind_speed = ifelse(is.na(wind_speed), mean(wind_speed, na.rm =
TRUE), wind_speed),
         wind_dir = ifelse(is.na(wind_dir), mean(wind_dir, na.rm = TRUE),
wind_dir),
         humid = ifelse(is.na(humid), mean(humid, na.rm = TRUE), humid),
         dewp = ifelse(is.na(dewp), mean(dewp, na.rm = TRUE), dewp),
         temp = ifelse(is.na(temp), mean(temp, na.rm = TRUE), temp),
         precip = ifelse(is.na(precip), mean(precip, na.rm = TRUE), precip),
         visib = ifelse(is.na(visib), mean(visib, na.rm = TRUE), visib))
#dropping features with more than 5% NA values
drops <-
c("speed","seats","engines","year.y","pressure","wind_gust","manufacturer","t
ype","model","engine")
df =df[ , !(names(df) %in% drops)]
df_test =df_test[ , !(names(df_test) %in% drops)]
#Creating date column from year, month, day and dropping irrelevant features
df <- df %>%
  mutate(dep_date = make_date(year.x,month,day)) %>%
  select(-year.x,-month,-day,-dep_time,-arr_time,-arr_delay,
         -sched_arr_time,-tailnum,-flight,-name,-air_time,
         -hour,-minute,-time_hour,-tz,-dst,-tzone) %>%
  mutate(precip = as.numeric(precip>0))

df_test <- df_test %>%
  mutate(dep_date = make_date(df_test$year.x,df_test$month,df_test$day)) %>%
  select(-year.x,-month,-day,-dep_time,-arr_time,-arr_delay,
         -sched_arr_time,-tailnum,-flight,-name,-air_time,
         -hour,-minute,-time_hour,-tz,-dst,-tzone) %>%
  mutate(precip = as.numeric(precip>0))
# Dropping rows having NA values after preprocessing
fl <- na.omit(df)
fl_test <- na.omit(df_test)
summary(fl)
dim(fl)
fl_test

# Dep_delay seasonality
plt <- ggplot(fl, aes(x = fl$dep_date, y = fl$dep_delay, title = "Seasonality
Trends"))

plt + geom_point(aes(color = fl$origin)) + xlab("Date") + ylab("Departure
Delay")
plt + geom_point(aes(color = fl$origin)) + xlab("Date") + ylab("Departure
Delay") + geom_smooth(color = "Black")
#Dep_delay target analysis
boxplot(fl$dep_delay)
```

```
#Converting to rank to overcome outliers and right skewed data
#fl <- fl %>% mutate(dep_delay = qqnorm(dep_delay)$x)
den <- nrow(fl)+1
fl <- fl %>% mutate(dep_delay = rank(dep_delay)/den)
 ggplot(fl,aes(x=dep_delay)) + geom_histogram(binwidth=.01)
den_te <- nrow(fl_test)+1
fl_test <- fl_test %>% mutate(dep_delay = rank(dep_delay)/den_te)
 ggplot(fl_test,aes(x=dep_delay)) + geom_histogram(binwidth=.01)
fl_test
# Departure delay seasonality after normalising dep_delay
plt <- ggplot(fl, aes(x = fl$dep_date, y = fl$dep_delay, title = "Seasonality
Trends"))

plt + geom_point(aes(color = fl$origin)) + xlab("Date") + ylab("Average Delay
(mins)")
plt + geom_point(aes(color = fl$origin)) + xlab("Date") + ylab("Average Delay
(mins)") + geom_smooth(color = "Black")
#zoomed in plot of seasonality
plt + xlab("Date") + ylab("Average Delay (mins)") + geom_smooth(color =
"Blue")
# checking for collinearity between dep_delay and weather data
library(corrplot)
cor_data <- select(fl, dep_delay, temp, dewp, humid,
                   wind_dir, wind_speed, precip, visib)
corrplot(cor(na.omit(cor_data)), method = "circle", type = "upper",
         tl.srt = 25, tl.col = "Black", tl.cex = 1, title = "Correlation
         between all 'weather' variables & 'dep_delay'", mar =c(0, 0, 4, 0) +
0.1)

# Plotting a smoother for Departure delay v/s Relative Humidity
g <- ggplot(cor_data, aes(y = dep_delay, x =humid ,title = "Departure delay
v/s Relative Humidity"))
g + geom_smooth() + ylab("Dep Delay") + xlab("Relative Humidity")

#Plotting a smoother for Departure delay v/s Temperature
g <- ggplot(cor_data, aes(y = dep_delay, x = temp ,
                          title = "Departure delay v/s Temperature"))
g + geom_smooth() + ylab("Departure Delay") +
  xlab("Temperature")
#Analysis of Dep_delay with origin, carrier & destination.
Q3 <- function(x) { quantile(x,probs=.75) }
fl %>% group_by(carrier) %>%
  summarize(n=n(),med_d = median(dep_delay),Q3_d = Q3(dep_delay), max_d =
max(dep_delay)) %>%
  arrange(desc(Q3_d)) %>% head(10)
fl %>% group_by(origin,carrier) %>%
  summarize(n=n(),med_d = median(dep_delay),Q3_d = Q3(dep_delay), max_d =
max(dep_delay)) %>%
  arrange(desc(Q3_d)) %>% head(10)
fl %>% group_by(dest,carrier) %>%
```

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

ggplot(fl,aes(x=fl$sched_dep_time,y=fl$dep_delay)) + geom_point(alpha=0.01) +
geom_smooth()
# delays increase throughout the day
ggplot(fl,aes(x=fl$distance,y=fl$dep_delay)) + geom_point(alpha=0.01) +
geom_smooth()
ggplot(fl,aes(x=log(fl$distance),y=fl$dep_delay)) + geom_point(alpha=0.01) +
geom_smooth()
# increases with distance -- use log distance
fl <- mutate(fl,logdistance = log(distance)) %>% select(-distance)
fl_test <- mutate(fl_test,logdistance = log(distance)) %>% select(-distance)
ggplot(fl,aes(x=fl$temp,y=fl$dep_delay)) + geom_point(alpha=0.01) +
geom_smooth()
# delays when too hot or too cold
ggplot(fl,aes(x=fl$dewp,y=fl$dep_delay)) + geom_point(alpha=0.01) +
geom_smooth()
# similar to temp
# Etc.
# Replace alt with log(alt)
fl <- mutate(fl,logalt = log(alt)) %>% select(-alt)
fl_test <- mutate(fl_test,logalt = log(alt)) %>% select(-alt)
#Adding additional feature to group data into quarters
fl$qtr <- as.yearqtr(fl$dep_date, format = "%Y-%m-%d")
fl_test$qtr <- as.yearqtr(fl_test$dep_date, format = "%Y-%m-%d")
fl$qtr = factor(fl$qtr)
fl_test$qtr = factor(fl_test$qtr)
## Split training set in two for tuning
set.seed(123)
tr_size <- ceiling(2*nrow(fl)/3)
train <- sample(1:nrow(fl),size=tr_size)
fl_tr <- fl[train,]  #Train set
fl_ve <- fl[-train,] # Validation set

# baseline to compare learning methods to:
var_dd <- var(fl_ve$dep_delay)
var_dd
#4 Learning Methods

#GAM
form <- formula(dep_delay ~ s(dep_date) + s(sched_dep_time) + carrier +
origin + s(logdistance) +
                  s(temp) + s(dewp) + s(humid) + s(wind_dir) + s(wind_speed)
+ precip + s(visib) + qtr)
gam_fit <- gam(form, data=fl_tr,family=gaussian)
summary(gam_fit)
plot(gam_fit,se=TRUE)
```

```r
#GAM prediction validation set
gam_pred_val <- predict(gam_fit,newdata=fl_ve)
mse_gam <- mean((fl_ve$dep_delay-gam_pred_val)^2)
mse_gam
abs(mse_gam - var_dd)/var_dd
# GAM prediction test set
#gam_fit$xlevels$dest <- union(gam_fit$xlevels$dest, levels(fl_test$dest))
drops <- c("dest")
fl_test_cl =fl_test[ , !(names(fl_test) %in% drops)]
gam_pred_test <- predict(gam_fit,newdata=fl_test_cl)
mse_gam <- mean((fl_test$dep_delay-gam_pred_test)^2)
mse_gam
abs(mse_gam - var_dd)/var_dd

# GBM
dep_date_numeric <- as.numeric(fl_tr$dep_date)
dep_date_numeric <- dep_date_numeric - mean(dep_date_numeric)
fl_tr_tem <- mutate(fl_tr,dep_date = dep_date_numeric)
gbm_fit <-gbm(dep_delay ~ .,data=fl_tr_tem,distribution="gaussian",
              n.trees = 1000, shrinkage = 0.2)
summary(gbm_fit)
#
dep_date_numeric <- as.numeric(fl_ve$dep_date)
dep_date_numeric <- dep_date_numeric - mean(dep_date_numeric)
fl_ve_tem <- mutate(fl_ve,dep_date = dep_date_numeric)
# GBM prediction Validation set
gbm_pred <- predict(gbm_fit,newdata=fl_ve_tem,n.trees = 1000)
mse_gbm <- mean((fl_ve$dep_delay-gbm_pred)^2)
mse_gbm
abs(mse_gbm - var_dd)/var_dd

#
dep_date_numeric <- as.numeric(fl_test$dep_date)
dep_date_numeric <- dep_date_numeric - mean(dep_date_numeric)
fl_test_tem <- mutate(fl_test,dep_date = dep_date_numeric)
# GBM prediction Test set
gbm_pred <- predict(gbm_fit,newdata=fl_test_tem,n.trees = 1000)
mse_gbm <- mean((fl_test$dep_delay-gbm_pred)^2)
mse_gbm
abs(mse_gbm - var_dd)/var_dd
# Linear Regression
lm.fit=lm(dep_delay~.,fl_tr)
summary(lm.fit)
# Dropping irrelevant features
drops <- c("lat","lon","logalt","dewp","dest")
fl_tr_LR =fl_tr[ , !(names(fl_tr) %in% drops)]
fl_ve_LR =fl_ve[ , !(names(fl_ve) %in% drops)]
lm.fit=lm(dep_delay~.,fl_tr_LR)
# Linear Regression prediction on Validation set
```

```
lr_pred =predict(lm.fit,newdata=fl_ve)
mse_lr <- mean((fl_ve$dep_delay-lr_pred)^2)
mse_lr
abs(mse_lr - var_dd)/var_dd
# Linear Regression prediction on Test set
lr_pred =predict(lm.fit,newdata=fl_test_cl)
mse_lr <- mean((fl_test_cl$dep_delay-lr_pred)^2)
mse_lr
abs(mse_lr - var_dd)/var_dd
#Ridge Regression
x=model.matrix(fl_tr$dep_delay~.,fl_tr)[,-1]
y=fl_tr$dep_delay
x_valid =model.matrix(fl_ve$dep_delay~.,fl_ve)[,-1]
y_valid = fl_ve$dep_delay
x_test =model.matrix(fl_test$dep_delay~.,fl_test)[,-1]
y_test = fl_test$dep_delay
cv.out=cv.glmnet(x,y,alpha=0) # defualt 10 fold cv
plot(cv.out)
bestlam=cv.out$lambda.min
bestlam
ridge.mod=glmnet(x,y,alpha=0,lambda=bestlam, thresh =1e-12)

#Ridge Regression prediction on validation set
ridge.pred=predict(ridge.mod,s=4,newx=x_valid)
mean((ridge.pred-y_valid)^2)
#Ridge Regression prediction on Test set
ridge.pred=predict(ridge.mod,s=4,newx=x_test)
mean((ridge.pred-y_test)^2)
# Lasso
cv.out=cv.glmnet(x,y,alpha=0) # defualt 10 fold cv
plot(cv.out)
bestlam=cv.out$lambda.min
bestlam
ridge.mod=glmnet(x,y,alpha=1,lambda=bestlam, thresh =1e-12)

#Lasso prediction on validation set
ridge.pred=predict(ridge.mod,s=4,newx=x_valid)
mean((ridge.pred-y_valid)^2)

#Lasso prediction on Test set
ridge.pred=predict(ridge.mod,s=4,newx=x_test)
mean((ridge.pred-y_test)^2)
```