

Team Presentation 2: Predicting NYC Taxi Fares

Christopher Carlevato, Kate Luo, David Tabert, Ian Wesley McDaniel

4/16/2020

Read in and inspect data:

Fread is fast and convenient way to read in large CSV files.

Data Source: <https://www.kaggle.com/c/new-york-city-taxi-fare-prediction/data>

```
set.seed(1)
NY_Train <- fread("train.csv", nrows = 3000000) #Read 3M rows of the trainig data
NY_Test <- read.csv("test.csv")
summary(NY_Train)
```

```
##           key           fare_amount      pickup_datetime
## Length:3000000      Min.   : -62.00      Length:3000000
## Class :character    1st Qu.:   6.00      Class :character
## Mode  :character    Median :   8.50      Mode  :character
##                               Mean  :  11.34
##                               3rd Qu.:  12.50
##                               Max.   :1273.31
##
## pickup_longitude  pickup_latitude  dropoff_longitude
## Min.   : -3426.61  Min.   : -3488.08  Min.   : -3408.43
## 1st Qu.: -73.99   1st Qu.:  40.73   1st Qu.: -73.99
## Median : -73.98   Median :  40.75   Median : -73.98
## Mean    : -72.51   Mean    :  39.92   Mean    : -72.51
## 3rd Qu.: -73.97   3rd Qu.:  40.77   3rd Qu.: -73.96
## Max.    : 3439.43  Max.    : 2912.47  Max.    : 3457.62
##                               NA's   :23
##
## dropoff_latitude  passenger_count
## Min.   : -3488.08  Min.   : 0.000
## 1st Qu.:  40.73   1st Qu.: 1.000
## Median :  40.75   Median : 1.000
## Mean    :  39.92   Mean    : 1.685
## 3rd Qu.:  40.77   3rd Qu.: 2.000
## Max.    : 3345.92  Max.    :208.000
## NA's    :23
```

```
summary(NY_Test)
```

```
##           key           pickup_datetime
## 2009-01-01 11:04:24.0000001: 1 2011-12-13 22:00:00 UTC: 270
## 2009-01-01 11:04:24.0000002: 1 2013-09-25 22:00:00 UTC: 251
## 2009-01-01 11:04:24.0000003: 1 2012-11-20 21:54:00 UTC: 246
## 2009-01-02 17:45:40.0000001: 1 2014-07-21 18:19:00 UTC: 243
## 2009-01-02 17:45:40.0000002: 1 2010-08-27 18:45:00 UTC: 235
## 2009-01-02 17:45:40.0000003: 1 2011-06-01 07:37:00 UTC: 227
## (Other) :9908 (Other) :8442
## pickup_longitude pickup_latitude dropoff_longitude dropoff_latitude
## Min.   : -74.25  Min.   :40.57  Min.   : -74.26  Min.   :40.57
## 1st Qu.: -73.99  1st Qu.:40.74  1st Qu.: -73.99  1st Qu.:40.74
## Median : -73.98  Median :40.75  Median : -73.98  Median :40.75
## Mean    : -73.97  Mean    :40.75  Mean    : -73.97  Mean    :40.75
## 3rd Qu.: -73.97  3rd Qu.:40.77  3rd Qu.: -73.96  3rd Qu.:40.77
## Max.    : -72.99  Max.    :41.71  Max.    : -72.99  Max.    :41.70
##
## passenger_count
## Min.   :1.000
## 1st Qu.:1.000
## Median :1.000
## Mean    :1.671
## 3rd Qu.:2.000
## Max.    :6.000
##
```

Data cleansing:

```
#Remove key column:
NY_Train <- NY_Train %>% select(-key)
NY_Test <- NY_Test %>% select(-key)

#Create place holder column in NY_test for fare_amount:
NY_Test$fare_amount <- NA

#Drop NAs in train data:
sum(is.na(NY_Train))
```

```
## [1] 46
```

```
NY_Train <- na.omit(NY_Train)
#Filter out unreasonable price data:
NY_Train<- filter(NY_Train, fare_amount > 0, fare_amount < 700)
#Filter our unreaonable passenger data:
NY_Train%>%
  filter(passenger_count==0)%>%
  nrow()
```

```
## [1] 10610
```

```
NY_Train <- filter(NY_Train, passenger_count >0, passenger_count <= 10)

#Filter out geo data that is not in NYC boroughs:
NY_Train<-NY_Train%>%
  filter(pickup_longitude > -80 & pickup_longitude < -70) %>%
  filter(pickup_latitude > 35 & pickup_latitude < 45) %>%
  filter(dropoff_longitude > -80 & dropoff_longitude < -70) %>%
  filter(dropoff_latitude > 35 & dropoff_latitude < 45)
```

Feature engineering:

```
#Use package lubridate to extract various time components:
combined<-data.frame(rbind(NY_Train, NY_Test))

combined<-combined%>%
  mutate(
    pickup_datetime = ymd_hms(pickup_datetime),
    year = as.factor(year(pickup_datetime)),
    month = as.factor(month(pickup_datetime)),
    day = as.numeric(day(pickup_datetime)),
    dayOfWeek = as.factor(wday(pickup_datetime)),
    hour = as.numeric(hour(pickup_datetime)),
    timeOfDay = as.factor(ifelse(hour >= 3 & hour < 9,
                                "Morning", ifelse(hour >= 9 & hour < 14, "Mid-Day",
                                ifelse(hour >= 14 & hour < 18, "Evening", "Night")))))
  )%>%
  select(-pickup_datetime)

#Picking out distance to exact locations:
#jfk
jfk_lat<-40.6413
jfk_long<--73.7781
jfk<-c(jfk_long, jfk_lat)
#newark
nwk_lat<-40.6895
nwk_long<--74.1745
nwk<-c(nwk_long, nwk_lat)
#laguardia
lag_lat<-40.779
lag_long<--73.8740
lag<-c(lag_long, lag_lat)
# #MSG
# msg_lat<-40.7505
```

```

# msg_long<--73.9934
# msg<-c(msg_long, msg_lat)
# #times square
# ts_lat<-40.7589
# ts_long<--73.9851
# ts<-c(ts_long, ts_lat)
# #freedom tower
# freedom_lat<-40.7127
# freedom_long<--74.0134
# freedom<-c(freedom_long, freedom_lat)
# #empire state building
# esb_lat<-40.7484
# esb_long<--73.9857
# esb<-c(esb_long, esb_lat)
#grand central
grand_lat<-40.7527
grand_long<--73.9772
grand<-c(grand_long, grand_lat)
#bronx
bronx_lat <- (40.837048 * pi)/180
bronx_long <- (-73.865433 * pi)/180
bronx<-c(bronx_long, bronx_lat)
nyc<-c(-74.0063889, 40.7141667)

combined<-combined%>%
  mutate(
    dist = distHaversine(cbind(pickup_longitude, pickup_latitude), cbind(dropoff_longitude, dropoff_latitude)
), r = 6371),
    to_jfk = distHaversine(cbind(pickup_longitude, pickup_latitude), jfk, r = 6371) + distHaversine(cbind(dropoff_longitude, dropoff_latitude), jfk, r = 6371),
    to_nkw = distHaversine(cbind(pickup_longitude, pickup_latitude), nwk, r = 6371) + distHaversine(cbind(dropoff_longitude, dropoff_latitude), nwk, r = 6371),
    to_lag = distHaversine(cbind(pickup_longitude, pickup_latitude), lag, r = 6371) + distHaversine(cbind(dropoff_longitude, dropoff_latitude), lag, r = 6371),
    #to_msg = distHaversine(cbind(pickup_longitude, pickup_latitude), msg, r = 6371) + distHaversine(cbind(dropoff_longitude, dropoff_latitude), msg, r = 6371),
    #to_ts = distHaversine(cbind(pickup_longitude, pickup_latitude), ts, r = 6371) + distHaversine(cbind(dropoff_longitude, dropoff_latitude), ts, r = 6371),
    #to_freedom = distHaversine(cbind(pickup_longitude, pickup_latitude), freedom, r = 6371) + distHaversine(cbind(dropoff_longitude, dropoff_latitude), freedom, r = 6371),
    to_grand = distHaversine(cbind(pickup_longitude, pickup_latitude), grand, r = 6371) + distHaversine(cbind(dropoff_longitude, dropoff_latitude), grand, r = 6371),
    to_bronx = distHaversine(cbind(pickup_longitude, pickup_latitude), bronx, r = 6371) + distHaversine(cbind(dropoff_longitude, dropoff_latitude), bronx, r = 6371)
    #to_nyc = distHaversine(cbind(pickup_longitude, pickup_latitude), nyc, r = 6371) + distHaversine(cbind(dropoff_longitude, dropoff_latitude), nyc, r = 6371)
  )

#Now resplit train and test:
NY_Test <- combined[is.na(combined$fare_amount), ]
NY_Train <- combined[!is.na(combined$fare_amount), ]

```

Further Explore and Visualize Data:

```

print(paste("The mean fair price in the training data set is $", round(mean(NY_Train$fare_amount),2), ".", sep=""))

```

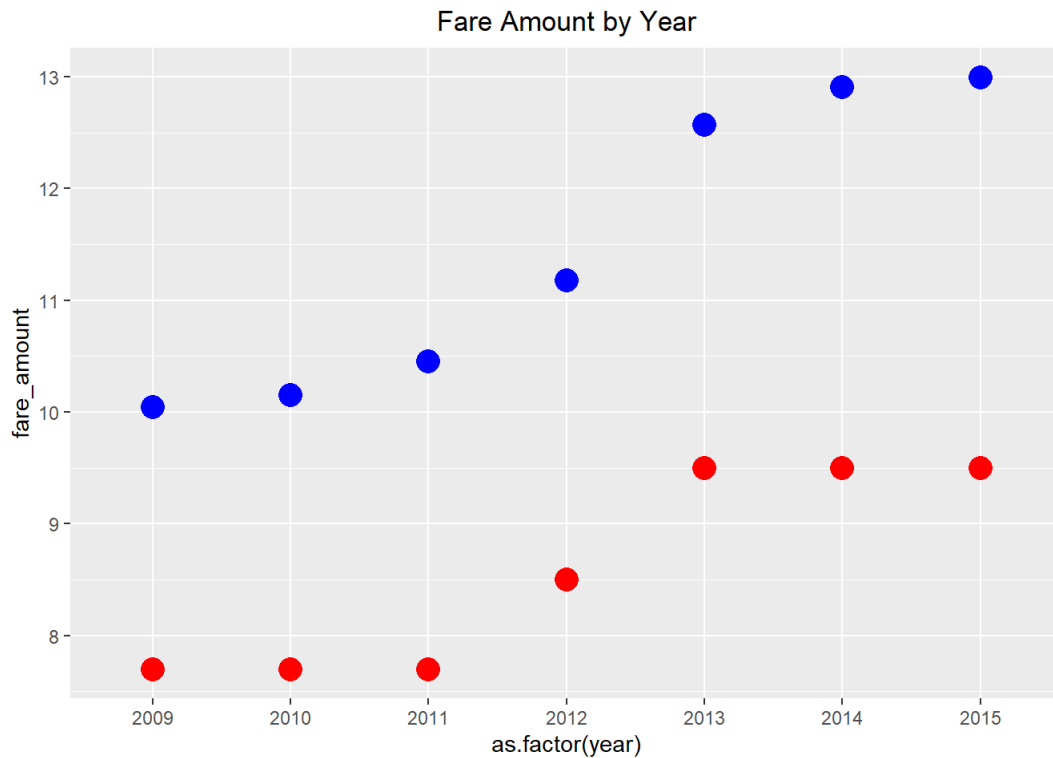
```

## [1] "The mean fair price in the training data set is $11.34."

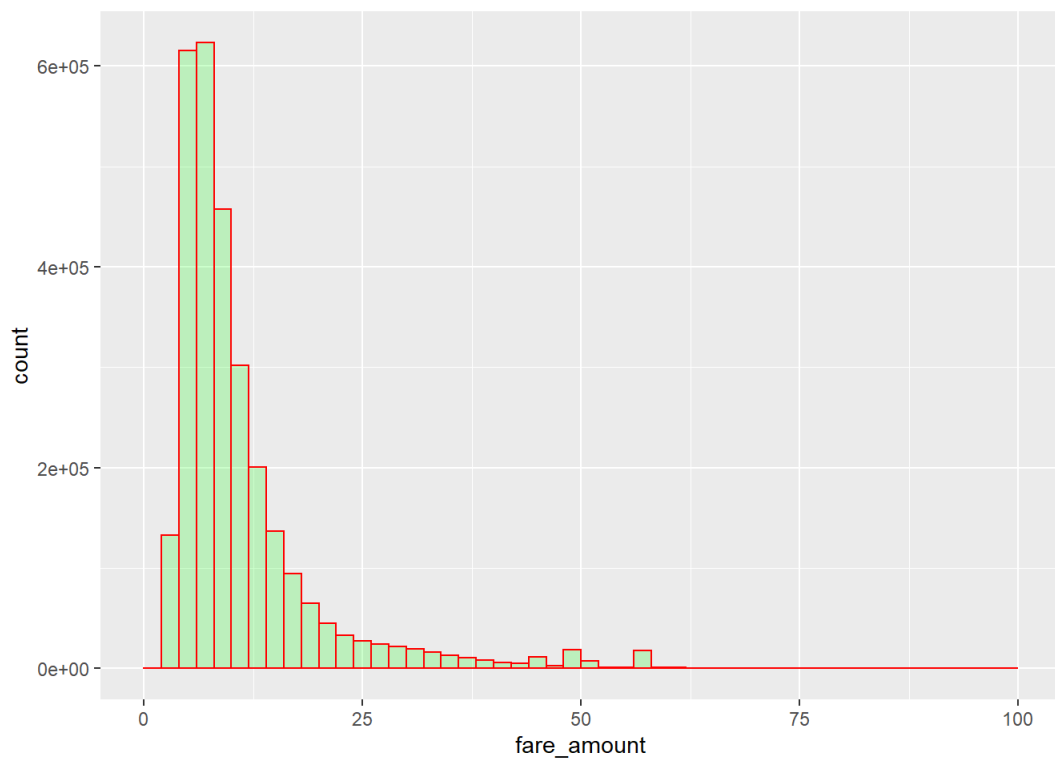
```

```
#ggplot variables:
m<-geom_point(stat = "summary", fun.y = "mean", col = "blue", size = 5)
med<-geom_point(stat = "summary", fun.y = "median", col = "red", size = 5)
manLegend<-scale_color_manual(name = "Summary Stat", values = c("mean"="blue", "median"="red"))

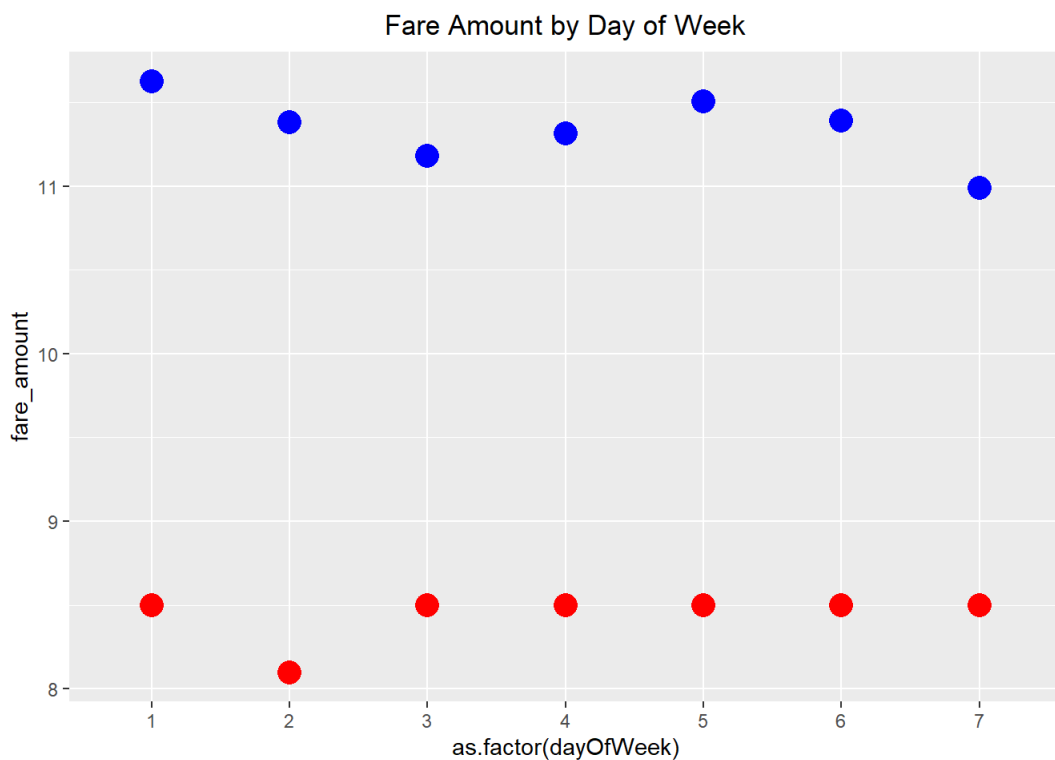
#Year and price:
ggplot(NY_Train, aes(as.factor(year), fare_amount))+
  m+
  med+
  manLegend+
  ggtitle("Fare Amount by Year")+
  theme(plot.title = element_text(hjust = .5), legend.position = "bottom")
```



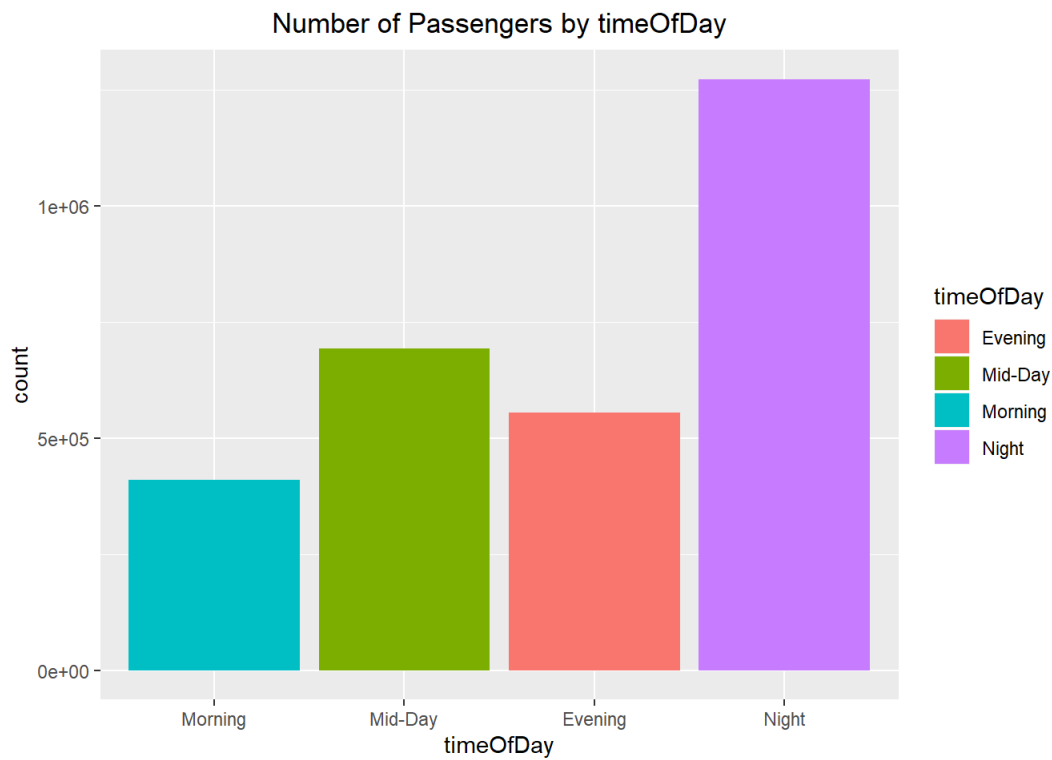
```
#Distribution of prices:
ggplot(NY_Train, aes(fare_amount)) +
  geom_histogram(breaks=seq(0, 100, by=2),
    col="red",
    fill="green",
    alpha=.2)
```



```
#Fare amount by day of week:
ggplot(NY_Train, aes(as.factor(dayOfWeek), fare_amount))+
  m+
  med+
  manLegend+
  ggtitle("Fare Amount by Day of Week")+
  theme(plot.title = element_text(hjust = .5), legend.position = "bottom")
```



```
#Passengers by time of day:
ggplot(NY_Train, aes(timeOfDay, fill = timeOfDay))+
  geom_bar(stat = "count", aes(y = ..count..))+
  scale_x_discrete(limits=c("Morning", "Mid-Day", "Evening", "Night"))+
  ggtitle("Number of Passengers by timeOfDay")+
  theme(plot.title = element_text(hjust = .5))
```



XGBoost Model:

```
#Create a validation set:
size = floor(.8*nrow(NY_Train))

xx <-sample(1:nrow(NY_Train), size)
NY_Valid<-NY_Train[-xx,]
NY_Train<-NY_Train[xx,]

#Convert to matrices for XGBoost algorithm to work:
dvalid <- xgb.DMatrix(data = data.matrix(NY_Valid[,-1]), label = NY_Valid[,1])
dtrain <- xgb.DMatrix(data = data.matrix(NY_Train[,-1]), label = NY_Train[,1])
dtest<-xgb.DMatrix(data = data.matrix(NY_Test[,-1]))

p <- list(objective = "reg:linear",
          eval_metric = "rmse",
          max_depth = 10,
          eta = .02,
          subsample=1,
          colsample_bytree=0.8,
          num_boost_round=250,
          nrounds = 500)

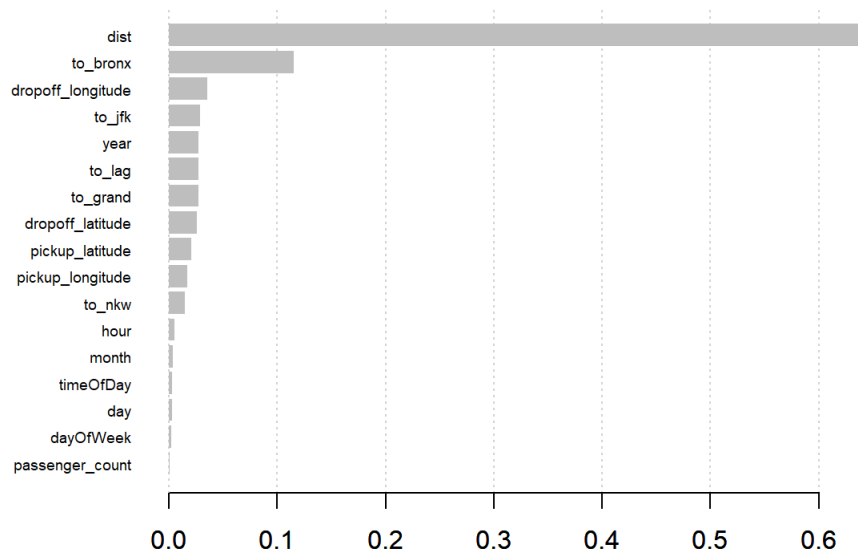
#Train the model:
set.seed(1)
m_xgb <- xgb.train(p, dtrain, p$nrounds, list(val=dvalid), print_every_n = 10, early_stopping_rounds = 10)
```

```
## [1] val-rmse:14.272313
## Will train until val_rmse hasn't improved in 10 rounds.
##
## [11] val-rmse:11.897102
## [21] val-rmse:10.014572
## [31] val-rmse:8.529146
## [41] val-rmse:7.323254
## [51] val-rmse:6.387753
## [61] val-rmse:5.671983
## [71] val-rmse:5.128957
## [81] val-rmse:4.726066
## [91] val-rmse:4.431731
## [101] val-rmse:4.214577
## [111] val-rmse:4.060880
## [121] val-rmse:3.948243
## [131] val-rmse:3.866209
## [141] val-rmse:3.806600
## [151] val-rmse:3.760950
## [161] val-rmse:3.728875
## [171] val-rmse:3.703937
## [181] val-rmse:3.684858
## [191] val-rmse:3.668841
## [201] val-rmse:3.656088
## [211] val-rmse:3.645313
## [221] val-rmse:3.635564
## [231] val-rmse:3.627590
## [241] val-rmse:3.620423
## [251] val-rmse:3.614422
## [261] val-rmse:3.609952
## [271] val-rmse:3.606364
## [281] val-rmse:3.602234
## [291] val-rmse:3.598963
## [301] val-rmse:3.595448
## [311] val-rmse:3.592509
## [321] val-rmse:3.589816
## [331] val-rmse:3.587926
## [341] val-rmse:3.585603
## [351] val-rmse:3.583902
## [361] val-rmse:3.581295
## [371] val-rmse:3.579829
## [381] val-rmse:3.577835
## [391] val-rmse:3.576200
## [401] val-rmse:3.574358
## [411] val-rmse:3.571254
## [421] val-rmse:3.570027
## [431] val-rmse:3.568401
## [441] val-rmse:3.566739
## [451] val-rmse:3.564548
## [461] val-rmse:3.562817
## [471] val-rmse:3.561215
## [481] val-rmse:3.558859
## [491] val-rmse:3.558043
## [500] val-rmse:3.556642
```

```
#Feature importance:
(impt_matrix <- xgb.importance(colnames(dtrain), model = m_xgb))
```

##	Feature	Gain	Cover	Frequency
## 1:	dist	0.6438705640	0.20147209	0.08438685
## 2:	to_bronx	0.1153522954	0.06509696	0.05877893
## 3:	dropoff_longitude	0.0354258208	0.07717865	0.08995730
## 4:	to_jfk	0.0284823425	0.09076591	0.08376058
## 5:	year	0.0275103059	0.06460437	0.03521163
## 6:	to_lag	0.0273045825	0.03412743	0.05801783
## 7:	to_grand	0.0270446162	0.03699437	0.04613979
## 8:	dropoff_latitude	0.0256484103	0.06541405	0.08771593
## 9:	pickup_latitude	0.0204772809	0.04436401	0.09842235
## 10:	pickup_longitude	0.0167353853	0.06843292	0.12761405
## 11:	to_nkw	0.0148450739	0.03728096	0.05194397
## 12:	hour	0.0051391290	0.07044823	0.05218069
## 13:	month	0.0038145099	0.04992147	0.03543936
## 14:	timeOfDay	0.0027381620	0.01177675	0.01624991
## 15:	day	0.0025599695	0.01555296	0.03447449
## 16:	dayOfWeek	0.0022509792	0.04456362	0.02780733
## 17:	passenger_count	0.0008005726	0.02200526	0.01189902

```
xgb.plot.importance(impt_matrix)
```



```
#Make predictions:
preds <- predict(m_xgb, dtest)

#Create CSV file with predictions to submit to Kaggle to obtain RMSE:
read.csv("sample_submission.csv") %>%
  mutate(fare_amount = preds) %>%
  write.csv("team12_sub.csv", row.names = F)

#Test RMSE: Obtained by submitting to Kaggle
RMSE <- 3.04896
print(paste("The RMSE for our model is $", round(RMSE,2), ".", sep=""))
```

```
## [1] "The RMSE for our model is $3.05."
```

```
print(paste("This means that, on average, the model misestimated the fare price by $", round(RMSE,2), ".", sep=""))
```

```
## [1] "This means that, on average, the model misestimated the fare price by $3.05."
```