Citi Bike Trip

Feiyi Ding

```r
dat_full <- fread("JC-202010-citibike-tripdata.csv")

# randomly sample 25% of the dataset
set.seed(1)
dat <- dat_full[sample(nrow(dat_full), 0.25*nrow(dat_full)), ]
```

1. Introduction

```r
head(dat)
```

```
##    tripduration                starttime                 stoptime
## 1:         1593 2020-10-17 15:25:04.2190 2020-10-17 15:51:37.5260
## 2:         5730 2020-10-24 10:30:56.0220 2020-10-24 12:06:26.5180
## 3:         1114 2020-10-04 19:28:54.8810 2020-10-04 19:47:29.3490
## 4:          927 2020-10-26 18:02:55.8320 2020-10-26 18:18:23.8110
## 5:          762 2020-10-13 06:41:39.2510 2020-10-13 06:54:21.9490
## 6:          300 2020-10-25 17:03:22.9740 2020-10-25 17:08:23.8480
##    start station id start station name start station latitude
## 1:             3202       Newport PATH               40.72722
## 2:             3269    Brunswick & 6th               40.72601
## 3:             3199       Newport Pkwy               40.72874
## 4:             3185          City Hall               40.71773
## 5:             3214   Essex Light Rail               40.71277
## 6:             3275     Columbus Drive               40.71836
##    start station longitude end station id           end station name
## 1:               -74.03376           3185                  City Hall
## 2:               -74.05039           3203              Hamilton Park
## 3:               -74.03211           3207                 Oakland Ave
## 4:               -74.04385           3792 Columbus Dr at Exchange Pl
## 5:               -74.03649           3203              Hamilton Park
## 6:               -74.03891           3638              Washington St
##    end station latitude end station longitude bikeid   usertype birth year
## 1:             40.71773             -74.04385  42212   Customer       1969
## 2:             40.72760             -74.04425  42358 Subscriber       1988
## 3:             40.73760             -74.05248  42545   Customer       1994
```

```
## 4:             40.71687             -74.03281  41369 Subscriber       1970
## 5:             40.72760             -74.04425  47054 Subscriber       1990
## 6:             40.72429             -74.03548  42609 Subscriber       1990
##    gender
## 1:      0
## 2:      1
## 3:      1
## 4:      1
## 5:      1
## 6:      2
```

```r
str(dat)
```

```
## Classes 'data.table' and 'data.frame':   7521 obs. of  15 variables:
##  $ tripduration           : int  1593 5730 1114 927 762 300 978 378 326 957 ...
##  $ starttime              : chr  "2020-10-17 15:25:04.2190" "2020-10-24 10:30:56.0220" "2020
##  $ stoptime               : chr  "2020-10-17 15:51:37.5260" "2020-10-24 12:06:26.5180" "2020
##  $ start station id       : int  3202 3269 3199 3185 3214 3275 3280 3269 3199 3272 ...
##  $ start station name     : chr  "Newport PATH" "Brunswick & 6th" "Newport Pkwy" "City Hall"
##  $ start station latitude : num  40.7 40.7 40.7 40.7 40.7 ...
##  $ start station longitude: num  -74 -74.1 -74 -74 -74 ...
##  $ end station id         : int  3185 3203 3207 3792 3203 3638 3681 3273 3202 3278 ...
##  $ end station name       : chr  "City Hall" "Hamilton Park" "Oakland Ave" "Columbus Dr at E
##  $ end station latitude   : num  40.7 40.7 40.7 40.7 40.7 ...
##  $ end station longitude  : num  -74 -74 -74.1 -74 -74 ...
##  $ bikeid                 : int  42212 42358 42545 41369 47054 42609 44343 44540 42446 47239
##  $ usertype               : chr  "Customer" "Subscriber" "Customer" "Subscriber" ...
##  $ birth year             : int  1969 1988 1994 1970 1990 1990 1978 1990 1969 1974 ...
##  $ gender                 : int  0 1 1 1 1 2 1 1 0 2 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Citi Bike can be seen everywhere in New York City. Many people choose to use it either for transportation or a bicycle trip to explore the city. This project focuses on building machine learning models to predict the trip duration of Citi Bike. The data set is obtained from Citi Bike's website.

The data set contains trip data of Citi Bike in October 2020. In the original data set, there are over thirty thousand data entries. Apart from the trip duration, the data also includes information about the time (start/end time and date), station (start/end name, id, latitude/longitude), bike (id), and user (type, gender, year of birth). When building the models, only 25% of randomly selected data are used.

The models built in this project can predict the trip duration with given information. All the information needed is available at the beginning of the trip. The prediction can tell the user how long their trip will be if they enter the destination. This can be a new feature for the Citi Bike app. Users who use Citi Bike as transportation will find it useful. If this new feature is introduced in the app, people can plan their trip ahead of time[1].

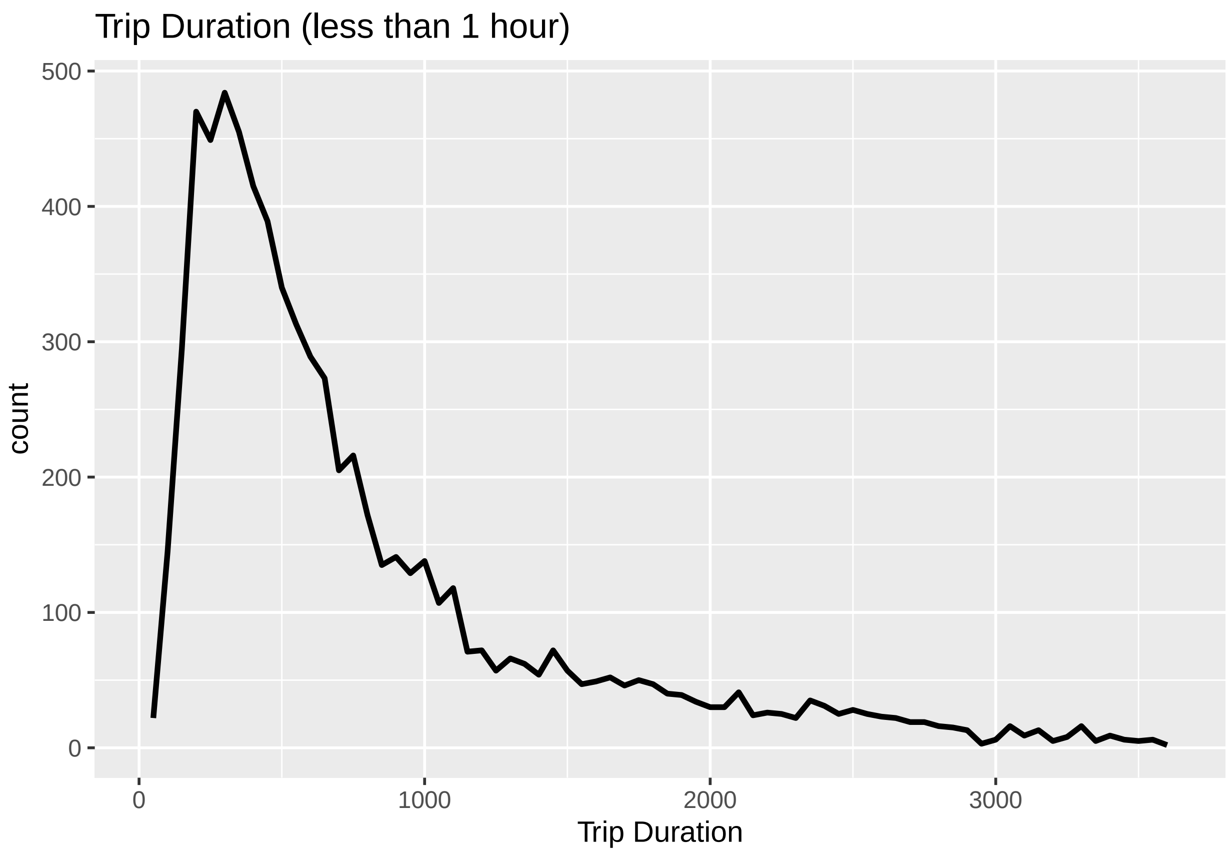
2. Exploratory Analysis

2.1 Trip duration

```
summary(dat$tripduration)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##      61     327     573    1414    1099 1579726
```

```
ggplot(dat[dat$tripduration < 3600,],aes(x=tripduration)) +
  geom_line(stat="bin", binwidth=50,size=1) +
  labs(title="Trip Duration (less than 1 hour)", x="Trip Duration")
```

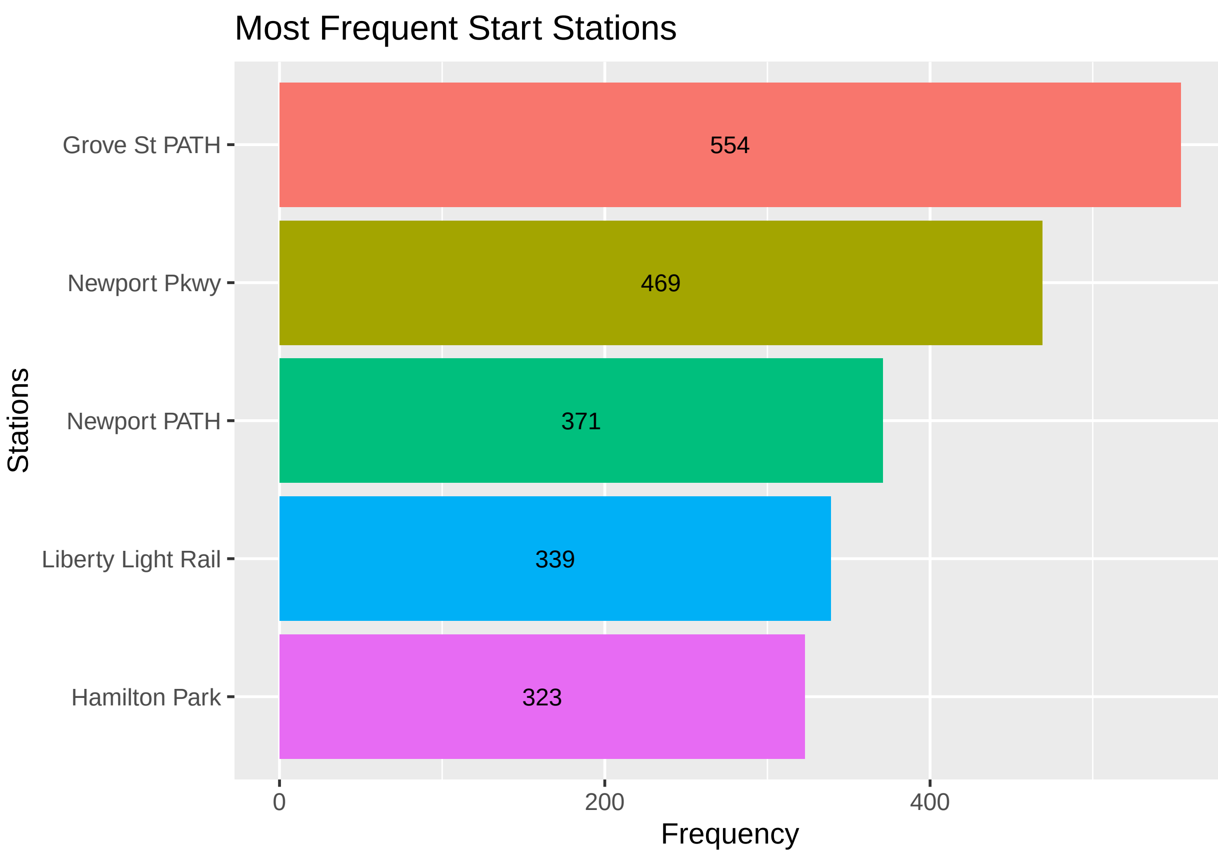


Most trips finished within 1099 seconds. There are some extreme data such as 1579726 secconds (more than 18 days), which should be removed as outlier.

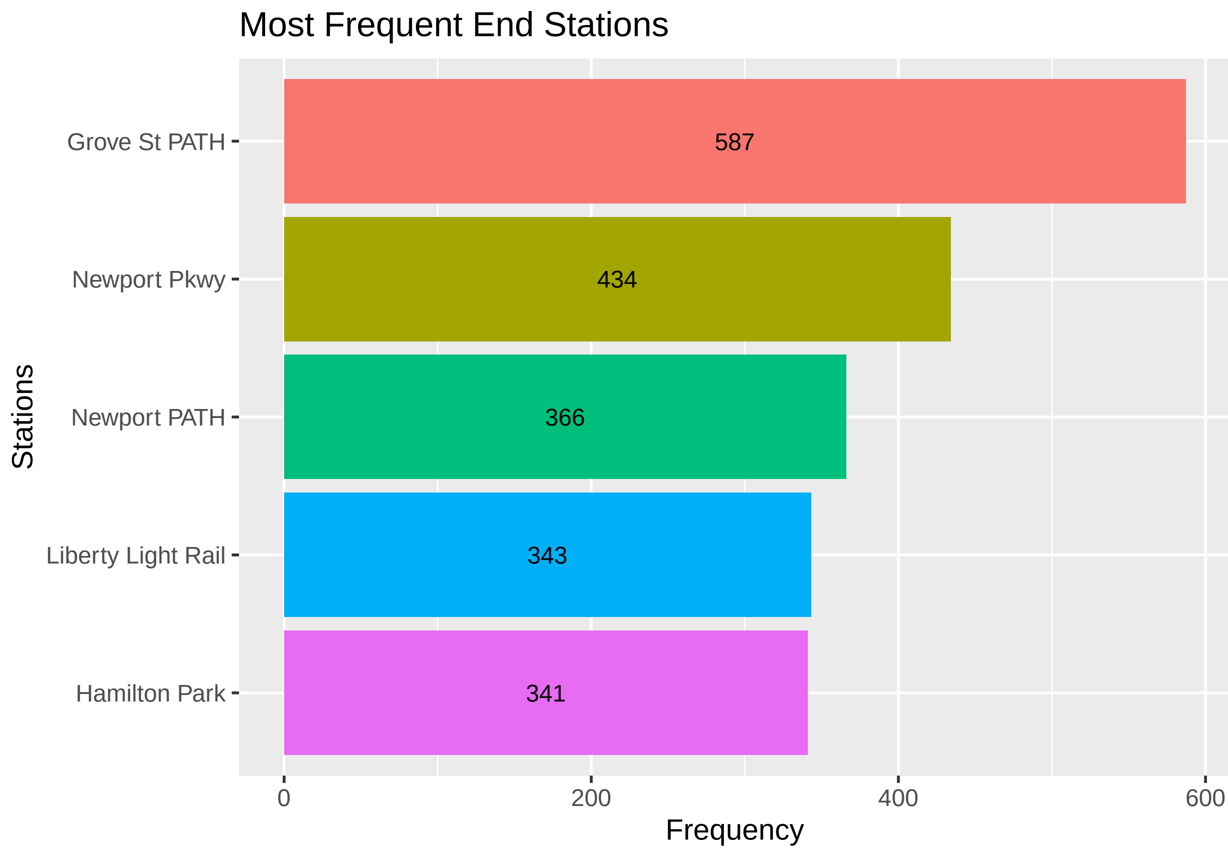
2.2 Popular stations and routes

```
top_10_start_station <- as.data.frame(sort(table(dat$`start station name`),decreasing=TRUE)[1:5
ggplot(top_10_start_station,aes(x = reorder(Var1,Freq), y = Freq)) +
```

```r
  geom_col(aes(fill=Var1))+
  coord_flip() +
  theme(legend.position = "none") +
  labs(title = "Most Frequent Start Stations", y = "Frequency", x = "Stations") +
  geom_text(aes(label= Freq), size = 3, position = position_stack(vjust = 0.5))
```



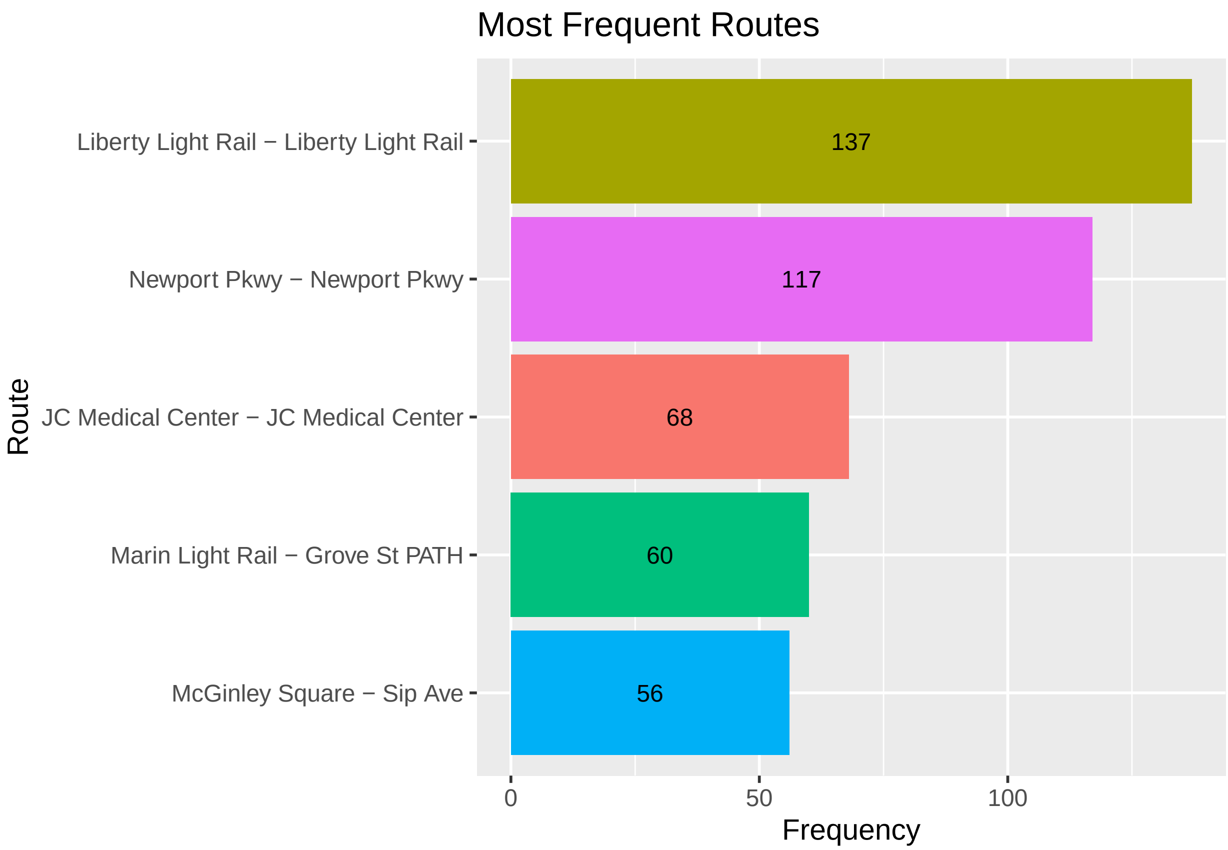

```r
top_10_end_station <- as.data.frame(sort(table(dat$`end station name`),decreasing=TRUE)[1:5])
ggplot(top_10_end_station,aes(x = reorder(Var1,Freq), y = Freq)) +
  geom_col(aes(fill=Var1))+
  coord_flip() +
  theme(legend.position = "none") +
  labs(title = "Most Frequent End Stations", y = "Frequency", x = "Stations") +
  geom_text(aes(label= Freq), size = 3, position = position_stack(vjust = 0.5))
```




```
route <- dat %>%
  group_by(`start station name`, `end station name`) %>%
  summarise(Freq = n())
```

```
## `summarise()` regrouping output by 'start station name' (override with `.groups` argument)
```

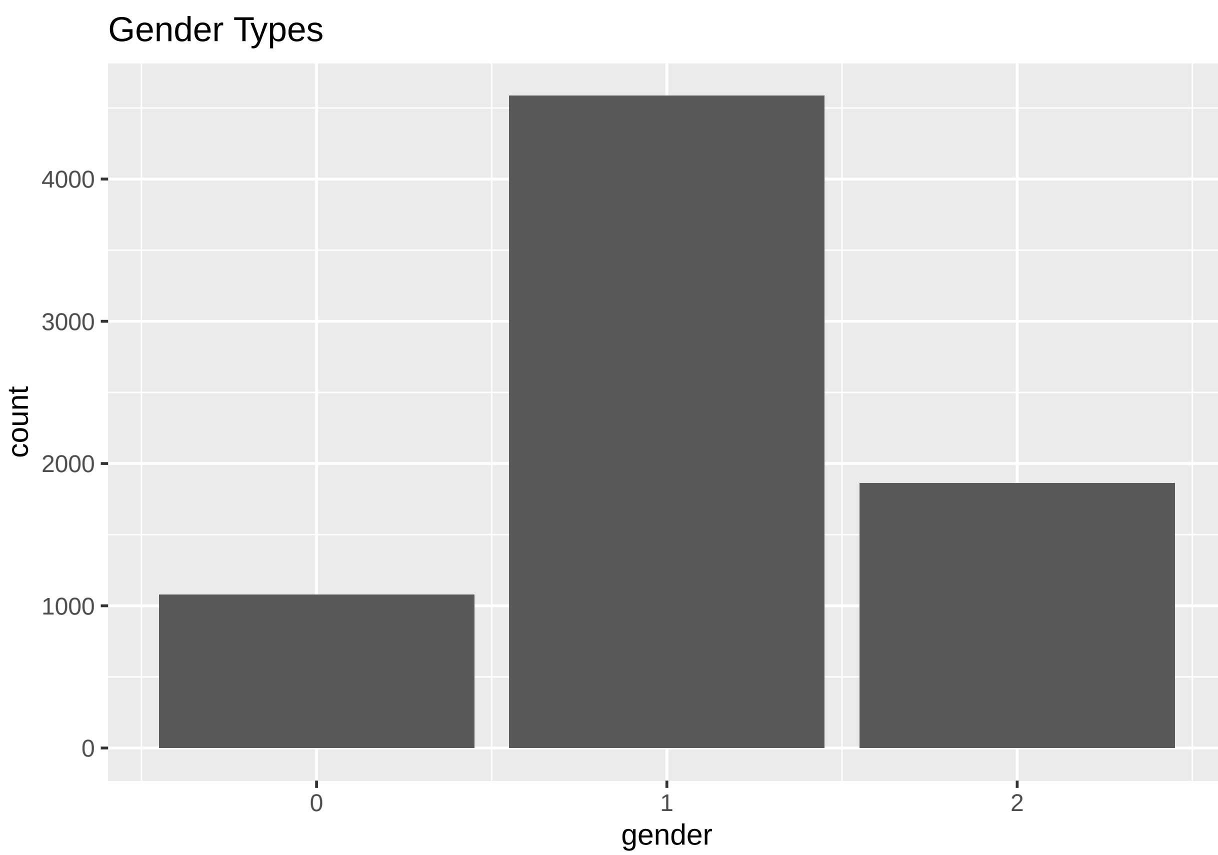
```
route$routes <- paste(route$`start station name`,"-",route$`end station name`)
route <- route[order(route$Freq, decreasing = T),]

ggplot(route[1:5, ],aes(x = reorder(routes,Freq), y = Freq)) +
  geom_col(aes(fill=routes))+
  coord_flip() +
  theme(legend.position = "none") +
  labs(title = "Most Frequent Routes", y = "Frequency", x = "Route") +
  geom_text(aes(label= Freq), size = 3, position = position_stack(vjust = 0.5))
```



Grove St PATH, Newport Pkwy, Newport PATH, Liberty Light Rail, and Hamilton Park are top 5 most popular stations for both start stations and end stations. Three out of five most popular routes have the same start station and end stations.

2.3 Gender and age

```
ggplot(dat, aes(x=gender))+
  geom_bar(aes(fill=gender)) +
  labs(title = "Gender Types")
```

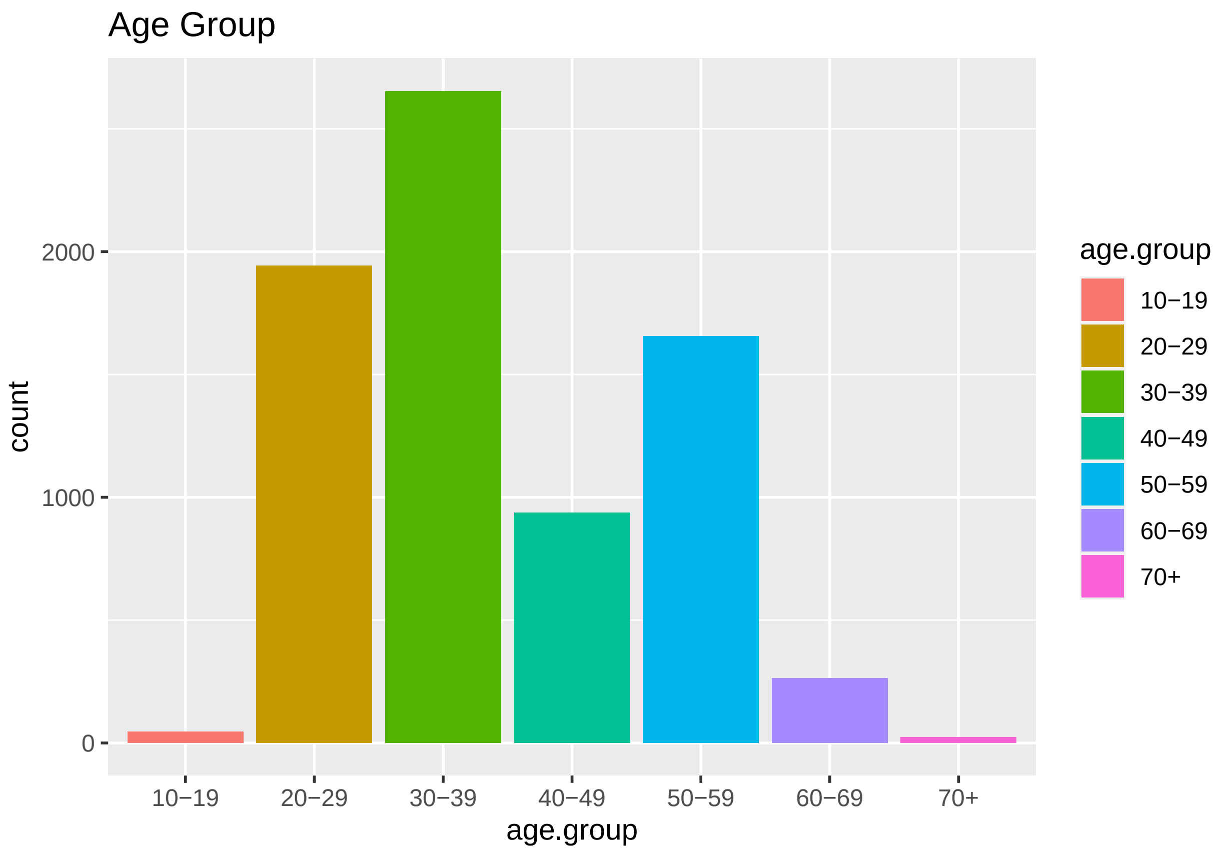


0=unknown; 1=male; 2=female

```
dat$age = as.numeric(2020 - dat$`birth year`)
summary(dat$age)
```

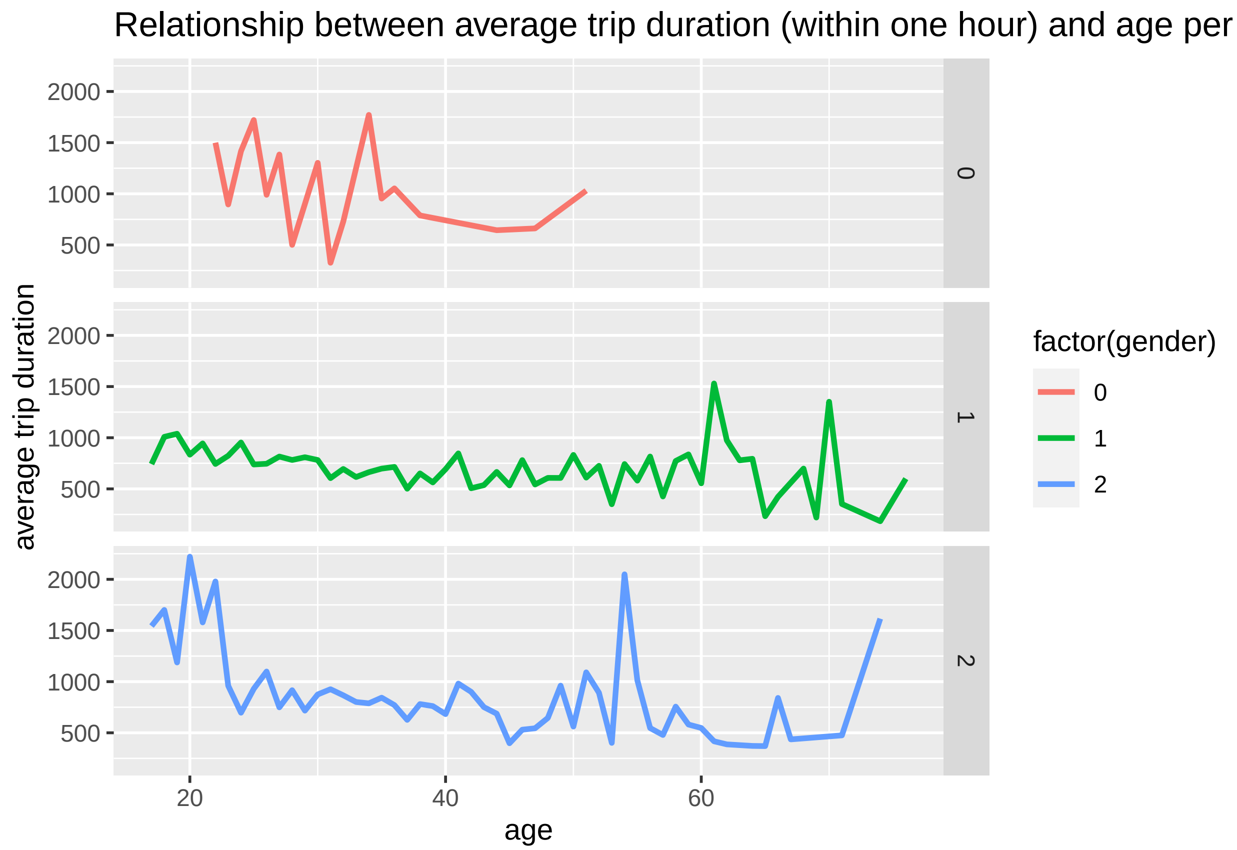
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   17.00   29.00   35.00   37.95   50.00   76.00
```

```
age.group <- c(paste(seq(0, 60, by = 10), seq(0 + 10 - 1, 70 - 1, by = 10),
                sep = "-"), paste(70, "+", sep = ""))
age.group = age.group[-1]
dat$age.group <- cut(dat$age, breaks = c(seq(10, 70, by = 10), Inf), labels = age.group, right
ggplot(dat, aes(x=age.group))+
  geom_bar(aes(fill=age.group)) +
  labs(title = "Age Group")
```




```
ggplot(dat[dat$tripduration < 3600], aes(x=age,y=tripduration ,colour=factor(gender))) +
  stat_summary(fun.y="mean", geom="line", size=1) + facet_grid(gender ~ .) +
  labs(title = "Relationship between average trip duration (within one hour) and age per gende
```

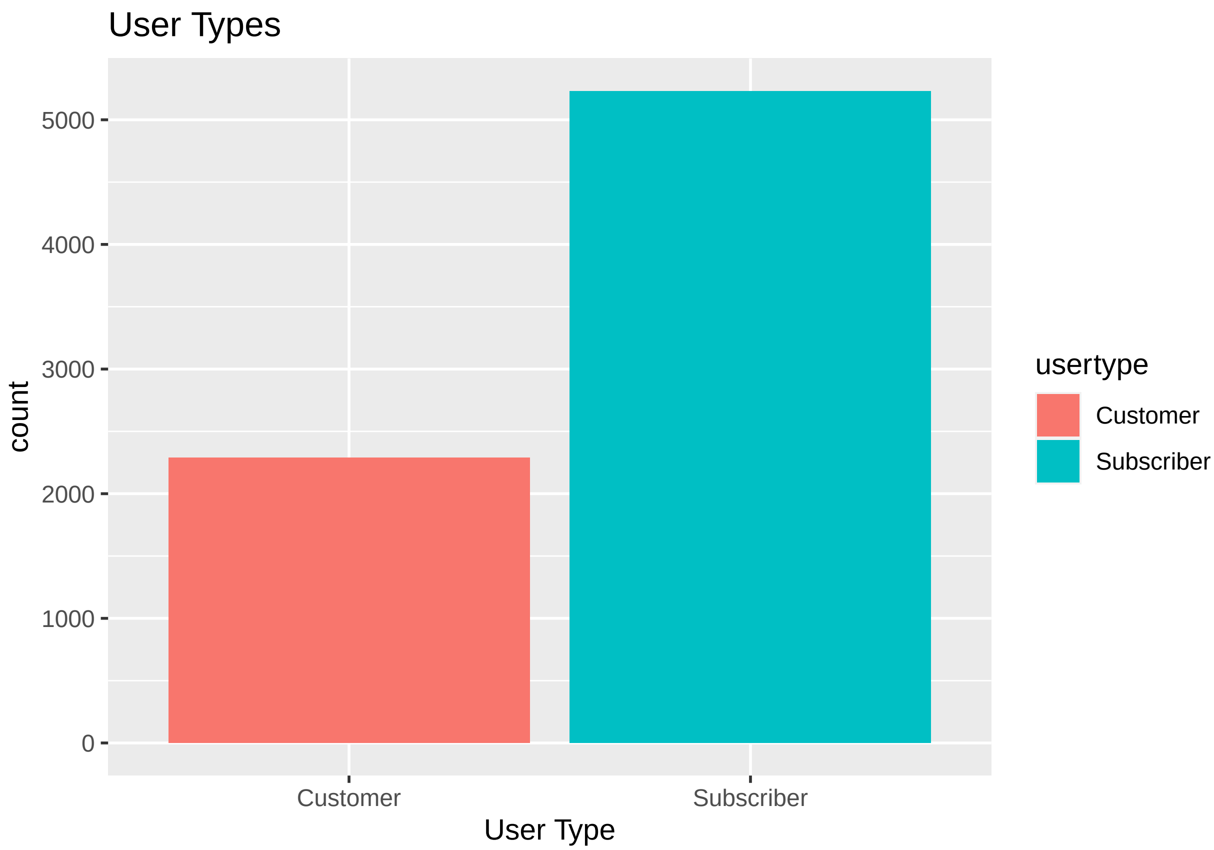
```
## Warning: `fun.y` is deprecated. Use `fun` instead.
```



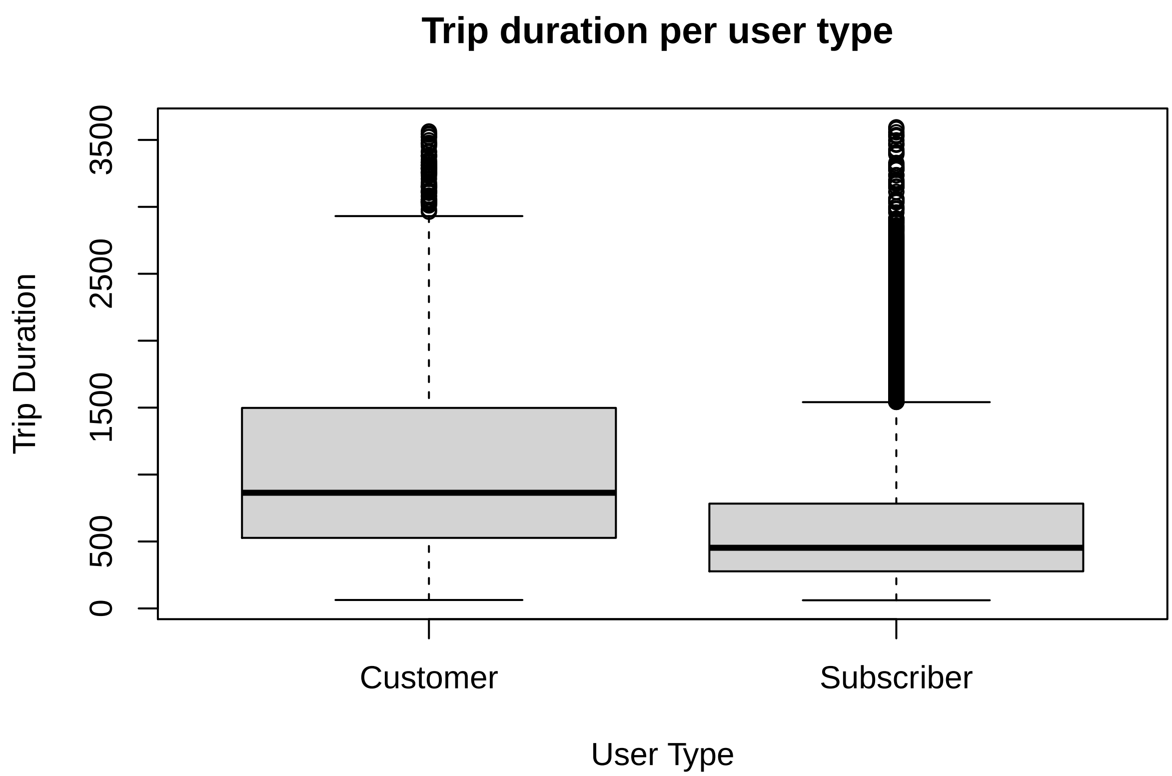
The majority of user are male.
Users in their 20s and 30s account for the largest proportion.
Exclude gender unknown users, yong and mid-age female users have longer average trip duration that male in their age. While for user who is over 60 years old, male users have longer average trip duration than female.

2.3 User Type

```
ggplot(dat, aes(x=usertype))+
  geom_bar(aes(fill=usertype)) +
  labs(title = "User Types", x= "User Type")
```




```
boxplot(tripduration~usertype, data=dat[dat$tripduration<3600,],main = "Trip duration per user
        xlab = "User Type", ylab = "Trip Duration")
```



The majority of user are subscribers (annual). However, customers (24-hour pass or 3-day pass user) tends to have a longer trip duration.

3. Data Cleaning & Imputation

```
sapply(dat, function(x) sum(is.na(x)))
```

```
##             tripduration               starttime                stoptime
##                        0                       0                       0
##         start station id      start station name  start station latitude
##                        0                       0                       0
## start station longitude          end station id        end station name
##                        0                       0                       0
##     end station latitude   end station longitude                  bikeid
##                        0                       0                       0
##                usertype              birth year                  gender
##                        0                       0                       0
##                      age               age.group
##                        0                       0
```

There's no na in the data set

```
dat$starttime = ymd_hms(dat$starttime,tz=Sys.timezone())
dat$stoptime = ymd_hms(dat$stoptime,tz=Sys.timezone())
breaks <- hour(hm("00:00", "6:00", "12:00", "18:00", "23:59"))
labels <- c("Night", "Morning", "Afternoon", "Evening")
dat$time.of.day <- cut(x=hour(dat$starttime), breaks = breaks, labels = labels, include.lowest=
```

Add morning/afternoon/evening label column to the data set.
The age column and age.group column have been added during exploratory data analysis.

```
citi <- dat[dat$tripduration <= 7200, ]
```

Remove outliers, whose trip duration time is unrealistic.

```
citi <- as.data.frame(citi)
citi <- citi[, -3]
```

Remove stop time as it contains the information for duration.

```
citi$gender <- factor(citi$gender)
colnames(citi) <- c("tripduration", "starttime", "start_station_id", "start_station_name", "sta
```

```
# backward selection
start_mod = lm(tripduration~.,data=citi)
empty_mod = lm(tripduration~1,data=citi)
full_mod = lm(tripduration~.,data=citi)
backwardStepwise = step(start_mod,
                        scope=list(upper=full_mod,lower=empty_mod),
                        direction='backward')
```

The variables selected by backward selection are `starttime`, `start station name`, `end station name`, `bikeid`, `usertype`, `birth year`, `gender`, `time.of.day`.

4. Machine Learning Models

```
set.seed(1)
split = sample.split(citi ,SplitRatio = 0.7)
train = citi[split, ]
test = citi[!split, ]
test <- test[test$end_station_name != "Broadway & W 49 St", ] #fix trouble in predict
```

The 5 machine learning models chosen for this project are linear regression, decision tree, random forest, boosting, and neural network. Since the project aims at predicting the trip duration (numeric), so I chose algorithms that works well in building supervised regression models[2]. I also take the running time into consideration. Considering I only worked on a small proportion of the entire Citi Bike data set, if the model takes too long to run, it may fail to provide value in the real world.

Model 1 linear regression

Linear regression is a linear approach to modeling the relationship between dependent independent variables. There are four assumptions in linear regression: 1) there exists a linear relationship between the independent (x) and dependent (y) variables; 2) independence assumption: the residuals are independent; 3) homoscedasticity assumption: the residuals have constant variance at every level of x; 4) normality assumption: the residuals of the model are normally distributed. Based on the assumptions, we can know that linear regression has some limitation: limited to linear relationship, sensitive to outliers, and linear regression only consider the relationship between the mean of independent variables and the value of dependent variable. Since trip duration is a numeric variable, I think it is a good choice that starts with a basic model like linear regression.

```
trControl=trainControl(method = "repeatedcv",
  number = 5,
  repeats = 5)
set.seed(1)
cv.lm = train(tripduration ~ starttime + start_station_id + end_station_id +
                          bikeid + usertype + birth_year + gender + time_of_day,
                 data=train, method="lm", trControl=trControl)
cv.lm$results
```

```
##   intercept     RMSE  Rsquared      MAE   RMSESD RsquaredSD    MAESD
## 1      TRUE 968.8939 0.1141469 629.1153 39.97476 0.01631392 14.84614
```

```
lm = lm(tripduration ~ starttime + start_station_id + end_station_id +
          bikeid + usertype + birth_year + gender + time_of_day,
        data=train)
pred.lm = predict(lm,newdata=test)
rmse.lm = RMSE(test$tripduration, pred.lm)
rmse.lm
```

```
## [1] 965.712
```

```
mae.lm = MAE(test$tripduration, pred.lm)
mae.lm
```

```
## [1] 623.7424
```

```
r2.lm = R2_Score(y_pred = pred.lm, test$tripduration)
r2.lm
```

```
## [1] 0.1376047
```

Model 2 decision tree

Decision tree can be used for both regression and classification models. It is a fast model with interpretable predictions. So I chose decision tree. Since decision prefers categorical variable, one assumption when building the tree is that if the values are continuous, then they will be discretized before building the model. The biggest limitations of decision tree is overfitting. If the tree is too complex, it may have poor performance on the testing data.

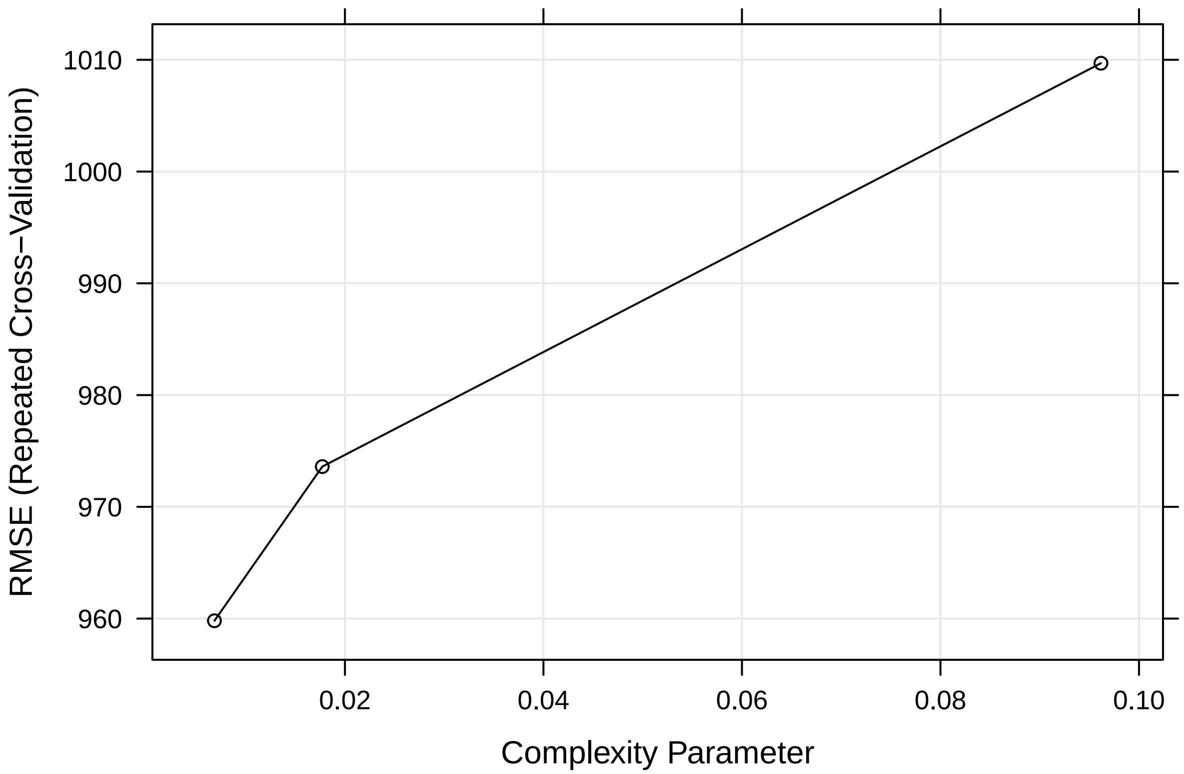
```
trControl=trainControl(method = "repeatedcv",
  number = 5,
  repeats = 5)
set.seed(1)
cv.tree = train(tripduration ~ starttime + start_station_id + end_station_id +
                          bikeid + usertype + birth_year + gender + time_of_day,
                 data=train, method="rpart", trControl=trControl)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
cv.tree$results
```

```
##            cp      RMSE   Rsquared      MAE   RMSESD RsquaredSD    MAESD
## 1 0.006855427  959.8066 0.13206104 621.7299 42.55877 0.02336769 15.94622
## 2 0.017721056  973.5992 0.10546783 632.7534 37.05757 0.01524554 12.73550
## 3 0.096169206 1009.6933 0.08352041 663.6535 46.95127 0.00714976 26.90107
```

```
trellis.par.set(caretTheme())
plot(cv.tree)
```




```
tree = rpart(tripduration ~ starttime + start_station_id + end_station_id +
               bikeid + usertype + birth_year + gender + time_of_day,
             data=train,
             cp = 0.006855427)
pred.tree = predict(tree,newdata=test)
rmse.tree = RMSE(test$tripduration, pred.tree)
rmse.tree
```

```
## [1] 949.5435
```

```
mae.tree = MAE(test$tripduration, pred.tree)
mae.tree
```

```
## [1] 616.6461
```

```
r2.tree = R2(pred = pred.tree, obs = test$tripduration)
r2.tree
```

```
## [1] 0.1703725
```

Model 3 random forest

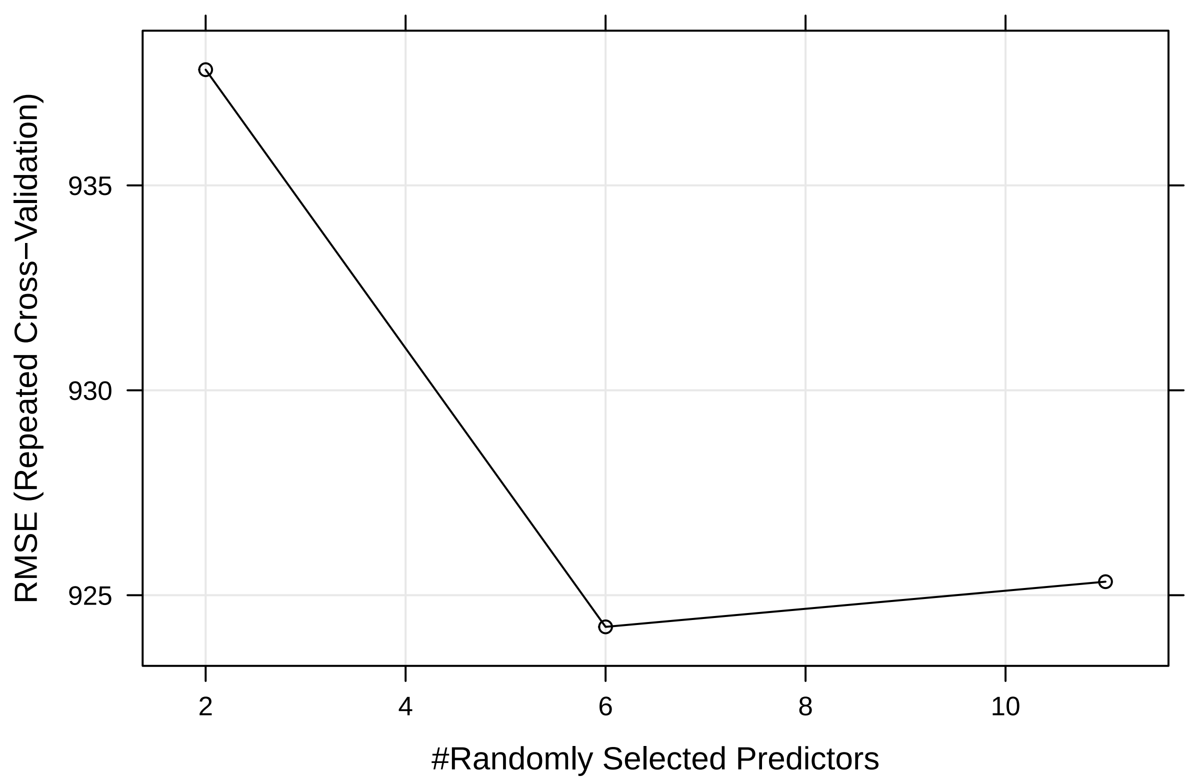
Random Forest is a ensemble learning method. It is consisted of many decision trees. Usually, random forest has better predictions than decision tree. So, I chose it to see the result. Random

Forest doesn't have distributional assumptions as the model can handle all types of data. The limitation of random forest is that the model is biased in favor of predictors with more levels. Besides, random forest also has overfitting issue.

```
trControl=trainControl(method = "repeatedcv",
  number = 5,
  repeats = 5)
set.seed(1)
cvForest = train(tripduration ~ starttime + start_station_id + end_station_id +
                           bikeid + usertype + birth_year + gender + time_of_day,
                 data=train, method="rf",ntree=200,trControl=trControl)
cvForest$results
```

```
##   mtry     RMSE  Rsquared      MAE   RMSESD RsquaredSD    MAESD
## 1    2 937.8233 0.1759481 603.4575 38.87560 0.02347218 13.47915
## 2    6 924.2277 0.1988185 594.2371 37.77378 0.02513750 13.50772
## 3   11 925.3297 0.2015721 593.5854 37.40510 0.02578340 14.71236
```

```
trellis.par.set(caretTheme())
plot(cvForest)
```




```
rf = randomForest(tripduration ~ starttime + start_station_id + end_station_id +
          bikeid + usertype + birth_year + gender + time_of_day,
        data=train,
```

```
        mtry = 6,
        ntree=200)
pred.rf = predict(rf,newdata=test)
rmse.rf = RMSE(test$tripduration, pred.rf)
rmse.rf
```

```
## [1] 878.6543
```

```
mae.rf = MAE(test$tripduration, pred.rf)
mae.rf
```

```
## [1] 555.6693
```

```
r2.rf = R2(pred = pred.rf, obs = test$tripduration)
r2.rf
```

```
## [1] 0.2873685
```

Model 4 boosting

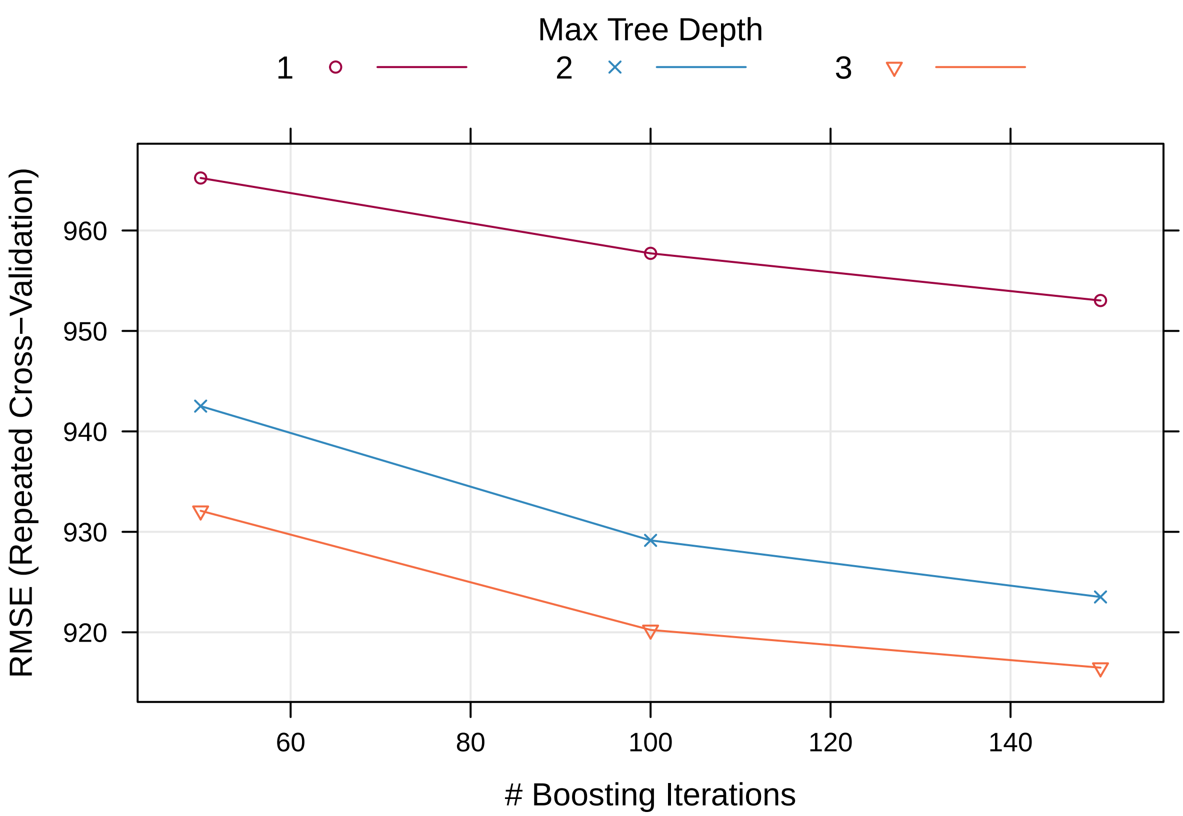
Boosting is an ensemble meta-algorithm with the advantage of reducing bias and variance. The assumption for boosting model is that observations should be independent. The limitation is that boosting is very sensitive to outliers and hard to scale up. I chose boosting to compare its performance to random forest, another ensemble algorithm. The run time for boosting is shorter but the prediction are worse.

```
fitControl <- trainControl(
  method = "repeatedcv",
  number = 5,
  repeats = 5)

set.seed(1)
gbmFit <- train(tripduration ~ starttime + start_station_id + end_station_id +
                  bikeid + usertype + birth_year + gender + time_of_day,
                 data = train,
                 method = "gbm",
                 trControl = fitControl,
                 verbose = FALSE)

gbmFit$results
```

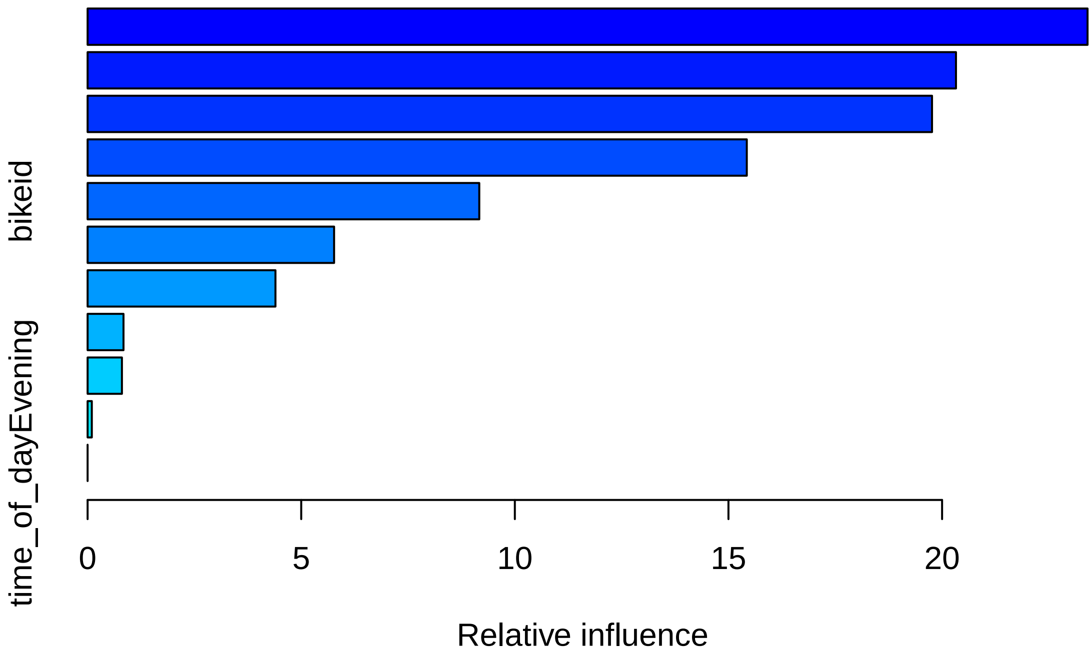
```
##   shrinkage interaction.depth n.minobsinnode n.trees     RMSE  Rsquared
## 1       0.1                 1             10      50 965.2280 0.1238184
## 4       0.1                 2             10      50 942.5147 0.1658373
## 7       0.1                 3             10      50 932.0960 0.1836360
```

```
## 2         0.1                   1              10      100  957.7273  0.1359893
## 5         0.1                   2              10      100  929.1515  0.1867834
## 8         0.1                   3              10      100  920.2322  0.2022740
## 3         0.1                   1              10      150  953.0311  0.1439137
## 6         0.1                   2              10      150  923.5137  0.1961531
## 9         0.1                   3              10      150  916.4772  0.2087204
##          MAE    RMSESD  RsquaredSD     MAESD
## 1 625.4352 40.73687 0.01682541 14.93929
## 4 604.8344 40.72698 0.02340154 16.12523
## 7 596.2574 39.50485 0.02547041 14.22608
## 2 617.9759 40.36613 0.01802728 15.07298
## 5 590.9658 39.87451 0.02411950 15.41699
## 8 584.0474 38.51381 0.02694168 14.39791
## 3 612.5796 40.30452 0.01930023 15.35876
## 6 584.6631 39.50808 0.02543636 15.52391
## 9 580.1349 38.18314 0.02808796 14.63609
```

```r
trellis.par.set(caretTheme())
plot(gbmFit)
```




```r
summary(gbmFit)
```




```
##                                             var     rel.inf
## usertypeSubscriber     usertypeSubscriber 23.40513771
## end_station_id             end_station_id 20.32516310
## start_station_id         start_station_id 19.76478384
## starttime                       starttime 15.42914191
## bikeid                             bikeid  9.16770773
## birth_year                     birth_year  5.76956586
## time_of_dayAfternoon time_of_dayAfternoon  4.39673505
## gender2                           gender2  0.84008900
## gender1                           gender1  0.80303396
## time_of_dayMorning     time_of_dayMorning  0.09864182
## time_of_dayEvening     time_of_dayEvening  0.00000000
```

```r
pred.gbm <- predict(gbmFit, newdata = test)
rmse.gbm = RMSE(test$tripduration, pred.gbm)
rmse.gbm
```

```
## [1] 894.2113
```

```r
mae.gbm = MAE(test$tripduration, pred.gbm)
mae.gbm
```

```
## [1] 560.6499
```

```r
r2.gbm = R2(pred = pred.gbm, obs = test$tripduration)
r2.gbm
```

```
## [1] 0.2675938
```

Model 5 neural net

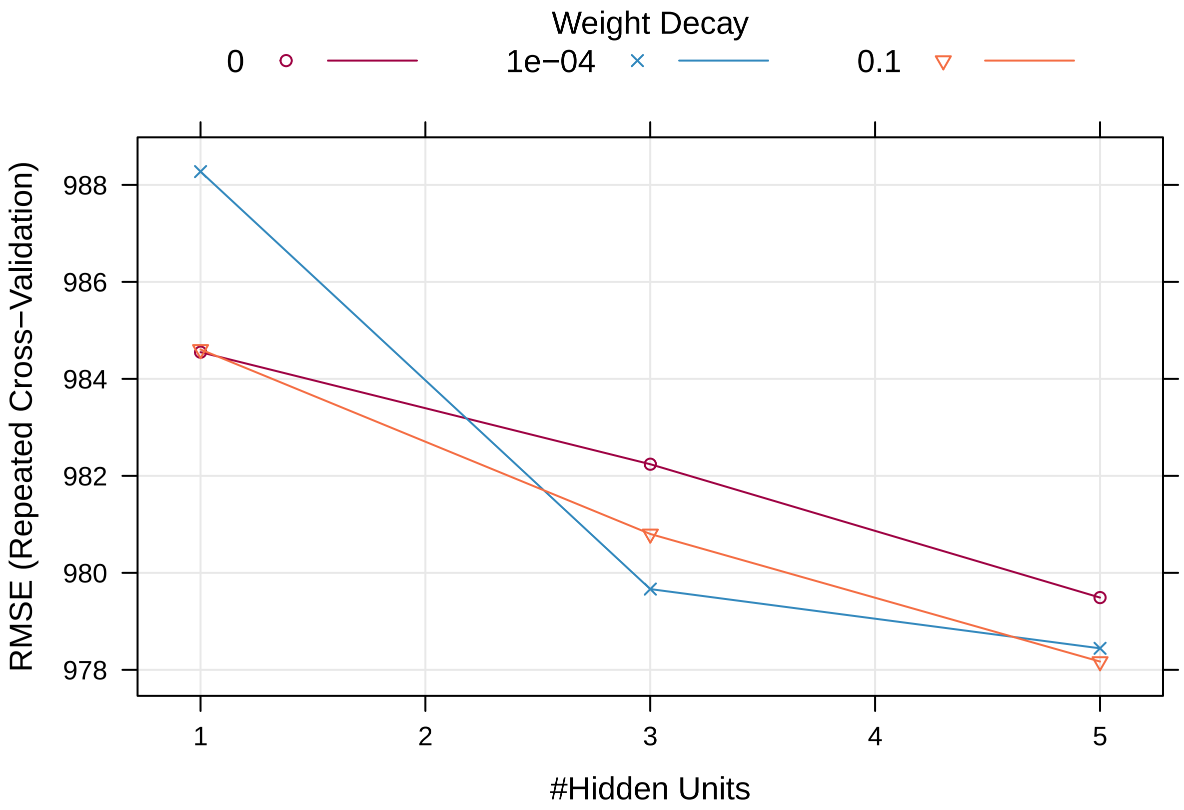
Neural network is a series of algorithms that tries to mimic how human brain operates and uses it to underlying relationships in the data set. it doesn't have assumptions for the data. One biggest limitation of neural network is its black box nature. It is very hard to understand how the neural network get this specify prediction.

```
control <- trainControl(method="repeatedcv", number=5, repeats=5)



cv.nn <- train(tripduration ~ starttime + start_station_id + end_station_id +
                 bikeid + usertype + birth_year + gender + time_of_day,
               data=train, method="nnet", trControl = control, preProc=c("center", "scale"), li
```

```
cv.nn$results
```

```
##   size decay     RMSE   Rsquared      MAE   RMSESD RsquaredSD    MAESD
## 1    1 0e+00 984.5505 0.08520263 641.7270 41.19779 0.02324914 16.21693
## 2    1 1e-04 988.2747 0.08128351 643.8376 40.91027 0.02594469 18.46997
## 3    1 1e-01 984.6070 0.08541592 640.8429 43.03448 0.02450803 17.82828
## 4    3 0e+00 982.2406 0.09067864 639.7099 46.28022 0.02174345 19.20807
## 5    3 1e-04 979.6653 0.09475315 637.6606 40.01842 0.02533011 15.76152
## 6    3 1e-01 980.8023 0.09331477 638.9568 39.43830 0.01958153 16.12237
## 7    5 0e+00 979.4908 0.09662748 637.0009 41.83545 0.01653692 17.85910
## 8    5 1e-04 978.4441 0.09810411 636.7123 41.47139 0.01791934 17.93634
## 9    5 1e-01 978.1710 0.09889546 636.9130 42.40159 0.02119429 17.31284
```

```
trellis.par.set(caretTheme())
plot(cv.nn)
```




```
pred.nn <- predict(cv.nn, newdata = test)
rmse.nn = RMSE(test$tripduration, pred.nn)
rmse.nn
```

```
## [1] 969.0434
```

```
mae.nn = MAE(test$tripduration, pred.nn)
mae.nn
```

```
## [1] 628.249
```

```
r2.nn = R2(pred = pred.nn, obs = test$tripduration)
r2.nn
```

```
## [1] 0.1334445
```

ensemble model

```
control_stacking <- trainControl(method="repeatedcv", number=5, repeats=3, savePredictions=TRUE

algorithms_to_use <- list(lm = caretModelSpec(method = 'lm'),
```

```r
                         rpart = caretModelSpec(method = 'rpart'),
                         rf = caretModelSpec(method = 'rf', ntree=200),
                         gbm = caretModelSpec(method = 'gbm'),
                         nnet = caretModelSpec(method="nnet", preProc=c("center", "scale"), linou

set.seed(1)
stacked_models <- caretList(tripduration ~ starttime + start_station_id + end_station_id +
                                bikeid + usertype + birth_year + gender + time_of_day,
                      data = test, trControl=control_stacking, tuneList=algorithms_to_use)
```

```r
ensemble <- caretEnsemble(
  stacked_models,
  metric="RMSE",
  trControl=trainControl(
    number=5
  ))
summary(ensemble)
```

```
## The following models were ensembled: lm, rpart, rf, gbm, nnet
## They were weighted:
## -4.5391 -0.0331 0.0168 0.5159 0.4952 -0.0169
## The resulting RMSE is: 913.0787
## The fit for each individual model on the RMSE is:
##  method     RMSE   RMSESD
##      lm 966.7705 34.63691
##   rpart 957.1345 35.19631
##      rf 923.1540 33.79255
##     gbm 923.1730 32.70335
##    nnet 977.0812 36.12355
```

```r
predictTest = predict(ensemble, newdata = test)

rmse.ensemble = RMSE(test$tripduration, predictTest)
rmse.ensemble
```

```
## [1] 640.467
```

```r
mae.ensemble = MAE(test$tripduration, predictTest)
mae.ensemble
```

```
## [1] 395.4881
```

```r
r2.ensemble = R2(predictTest, test$tripduration)
r2.ensemble
```

```
## [1] 0.7145112
```

5. Results

```
res <- data.frame(model = c('lm', 'rpart', 'rf', 'gbm', 'nnet', 'ensemble'),
                  rmse = c(rmse.lm, rmse.tree, rmse.rf, rmse.gbm, rmse.nn, rmse.ensemble),
                  mae = c(mae.lm, mae.tree, mae.rf, mae.gbm, mae.nn, mae.ensemble),
                  r2 = c(r2.lm, r2.tree, r2.rf, r2.gbm, r2.nn, r2.ensemble))
res
```

```
##      model     rmse      mae        r2
## 1       lm 965.7120 623.7424 0.1376047
## 2    rpart 949.5435 616.6461 0.1703725
## 3       rf 878.6543 555.6693 0.2873685
## 4      gbm 894.2113 560.6499 0.2675938
## 5     nnet 969.0434 628.2490 0.1334445
## 6 ensemble 640.4670 395.4881 0.7145112
```

6. Discussion and Next Steps

The ensemble model no doubt has the best performance. Among the 5 models I built, the random forest model has the best performance while the neural network has the worst performance. Since the linear correlation between tripduration and other variable is not very strong, so the prediction of linear regression is not very good. Random forest is the ensemble model of decision tree, so it will have a better prediction. As for boosting and neural network, I think to improve their performance, more outliers should be removed and some parameters should be selected manually instead of only rely on cross validation. Besides, when tuning the parameters, more methods can be used instead of just cross validation may be able to generate better model.

7. References

[1] Shah, V. (2018, May 25). Citi Bike 2017 Analysis. Retrieved December 10, 2020, from https://towardsdatascience.com/citi-bike-2017-analysis-efd298e6c22c
[2] An easy guide to choose the right Machine Learning algorithm. (n.d.). Retrieved December 10, 2020, from https://www.kdnuggets.com/2020/05/guide-choose-right-machine-learning-algorithm.html