# Model Building

Tejas Bawaskar

## Continued from data exploratory / data cleaning file

```
cl <- makeCluster(detectCores(), type='PSOCK')
registerDoParallel(cl)
```

## Input Clean Data

```
clean_datetime <- df[,-c(2,3,15,19)]
#drop/pickup datetime, tip amt, payment type

# Selecting the top variables from the list by using p-value.
summary(lm(Tip_percent ~. , data = clean_datetime[sample(nrow(clean_datetime) , 100000),]))

##
## Call:
## lm(formula = Tip_percent ~ ., data = clean_datetime[sample(nrow(clean_datetime),
##     1e+05), ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -208.705   -0.898    0.849    2.021   81.818
##
## Coefficients:
##                          Estimate Std. Error  t value Pr(>|t|)
## (Intercept)            -2.641e+02  3.334e+01   -7.920 2.39e-15 ***
## VendorID               -7.348e-02  3.467e-02   -2.119  0.03408 *
## Store_and_fwd_flag      1.577e-01  2.051e-01    0.769  0.44208
## RateCodeID             -6.600e-01  1.606e-01   -4.109 3.98e-05 ***
## Pickup_longitude       -3.604e+00  4.690e-01   -7.684 1.56e-14 ***
## Pickup_latitude        -2.066e+00  4.161e-01   -4.964 6.91e-07 ***
## Dropoff_longitude      -4.229e+00  3.723e-01  -11.358  < 2e-16 ***
```

```
## Dropoff_latitude      -5.309e+00  4.190e-01  -12.670  < 2e-16 ***
## Passenger_count        2.406e-02  1.356e-02    1.774  0.07604 .
## Trip_distance         -7.524e-02  1.546e-02   -4.867 1.13e-06 ***
## Fare_amount           -3.292e+00  1.080e-02 -304.659  < 2e-16 ***
## Extra                 -3.424e+00  4.150e-02  -82.514  < 2e-16 ***
## MTA_tax               -4.178e+00  1.279e+00   -3.266  0.00109 **
## Tolls_amount          -3.271e+00  1.694e-02 -193.124  < 2e-16 ***
## improvement_surcharge -3.849e+00  1.922e+00   -2.003  0.04522 *
## Total_amount           2.866e+00  7.423e-03  386.096  < 2e-16 ***
## Trip_type              6.341e-01  8.761e-01    0.724  0.46922
## trip_duration         -7.147e-04  4.711e-05  -15.171  < 2e-16 ***
## hour                   6.014e-03  2.120e-03    2.837  0.00456 **
## weekday_num            2.186e-03  6.861e-03    0.319  0.74999
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.423 on 99980 degrees of freedom
## Multiple R-squared:  0.6099, Adjusted R-squared:  0.6098
## F-statistic:  8227 on 19 and 99980 DF,  p-value: < 2.2e-16
```

```r
# check for multicollinearity
vif(lm(Tip_percent ~.,data=clean_datetime))
```

```
##              VendorID    Store_and_fwd_flag            RateCodeID
##              1.033153              1.017930             11.058885
##      Pickup_longitude       Pickup_latitude     Dropoff_longitude
##              1.745318              2.783382              1.745057
##      Dropoff_latitude       Passenger_count         Trip_distance
##              2.692893              1.010608             14.545188
##           Fare_amount                 Extra               MTA_tax
##             57.578044              1.147736             12.385551
##          Tolls_amount improvement_surcharge          Total_amount
##              1.598381             11.172918             35.453936
##             Trip_type         trip_duration                  hour
##             20.627401              5.828800              1.094788
```

```r
##           weekday_num
##              1.007344

# A vif value above 1 indicates the predictors are slightly correlated. A vif between 5 and 10 indicates hi
gh correlation that maybe problematic. And anything above 10, it can be concluded that the regression coeff
icients aren't correct/poorly estimated. To solve it standardizing the continuous predictors can be used, i
f not, we would have to rmove the highly correlated variables.


# From the variables we can notice that Trip distance, fare amount, mta tax, trip type and improvement surc
harge have high vif's. This is due to the fact that these variables are highly correlated and can be seen f
rom the correlation plot.

# Mean absolute error
MAE <- function(actual, predict){
  error <- abs(actual - predict)
  return(formatC(mean(error),digits=2, format="f"))
}

#Root Mean squared error
RMSE <- function(actual, predict){
  ans = sqrt(mean((actual - predict)^2))
  return(formatC(ans,digits=2, format="f"))
}

# Accuracy
accuracy <- function(actual, predict){
    error = abs(actual - predict)
    num = length(error[error <= 1])
    den = length(error)
    acc = 100*num/den
  return(paste(formatC(acc,digits=2, format="f") ,'%',sep=''))
}
```

From this we notice that the best variables that have p-values less than 0.05

## Creating training, validation & testing subsets

Since there are nearly 678000 observations in this dataset, it would make sense to randomly subset a sample of these observations for our model so that we can run it on a memory of 4GB.

```
set.seed(789)

#using just a sample of this to run the various models

spec = c(train = .4, test = .3, validate = .3)

g = sample(cut(
  seq(nrow(clean_datetime)),
  nrow(clean_datetime)*cumsum(c(0,spec)),
  labels = names(spec)
))

res = split(clean_datetime, g)

clean_datetime.test <- res$test[sample(nrow(res$test), 10000), ]
clean_datetime.train <- res$train[sample(nrow(res$train), 10000), ]
clean_datetime.val <- res$validate[sample(nrow(res$validate), 10000), ]
```

## Construction of related models and tuning them subsequently

### Linear Model

```
set.seed(789)

clean_datetime <- df[,-c(2,3,19)]

nums <- sapply(clean_datetime, is.numeric)
num.df <- clean_datetime[ , nums]
norm.df <- normalize(num.df, method = 'range', range = c(0,1))
```

```
clean_datetime_1 <- norm.df

clean_datetime.test <- clean_datetime[sample(nrow(clean_datetime_1), 10000), ]
clean_datetime.train <- clean_datetime[sample(nrow(clean_datetime_1), 10000), ]
clean_datetime.val <- clean_datetime[sample(nrow(clean_datetime_1), 2000), ]

#Model 1 Linear Regression
# To solve the multicollinearity issue, standardizing the continuous predictors can be used, but after tryi
ng it (standardizing/normalizing) the vif's values were unaffected. While building the linear model, we wou
ld have to subset variables that are highly related.

lm.model <- lm(Tip_percent ~ Dropoff_longitude + Pickup_longitude + Pickup_latitude + Total_amount + Trip_t
ype + trip_duration, data = clean_datetime.train)

#plot(lm.model)
summary(lm.model)

##
## Call:
## lm(formula = Tip_percent ~ Dropoff_longitude + Pickup_longitude +
##     Pickup_latitude + Total_amount + Trip_type + trip_duration,
##     data = clean_datetime.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -27.794  -2.821   1.715   3.730  55.209
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -1.219e+03  1.558e+02  -7.827 5.51e-15 ***
## Dropoff_longitude -1.570e+01  1.739e+00  -9.029  < 2e-16 ***
## Pickup_longitude  -7.538e+00  2.261e+00  -3.334  0.00086 ***
## Pickup_latitude   -1.167e+01  1.269e+00  -9.200  < 2e-16 ***
## Total_amount       3.142e-01  1.104e-02  28.453  < 2e-16 ***
## Trip_type         -9.660e+00  8.944e-01 -10.801  < 2e-16 ***
## trip_duration     -5.380e-03  1.898e-04 -28.353  < 2e-16 ***
```

```r
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.74 on 9993 degrees of freedom
## Multiple R-squared:  0.1035, Adjusted R-squared:  0.103
## F-statistic: 192.3 on 6 and 9993 DF,  p-value: < 2.2e-16

# All vif' are below 5
vif(lm.model)

## Dropoff_longitude  Pickup_longitude   Pickup_latitude      Total_amount
##          1.678263          1.747785          1.095485          3.703159
##         Trip_type     trip_duration
##          1.032770          3.656015

# this is the best model for understanding relationships between variables.

# train data
lm.predict <- predict(lm.model,clean_datetime.train,se.fit = TRUE, interval = "confidence",level = 0.95)
# Absolute Mean error of lm model
lm.mae.train <- MAE(clean_datetime.train$Tip_percent,lm.predict$fit)
#RMSE of lm model
lm.mse.train <- RMSE(clean_datetime.train$Tip_percent,lm.predict$fit)
#Accuracy of the model
lm.acc.train <- accuracy(clean_datetime.train$Tip_percent,lm.predict$fit)

# Validation data
lm.predict <- predict(lm.model,clean_datetime.val,se.fit = TRUE, interval = "confidence",level = 0.95)
# Absolute Mean error of lm model
lm.mae.val <- MAE(clean_datetime.val$Tip_percent,lm.predict$fit)
#RMSE of lm model
lm.mse.val <- RMSE(clean_datetime.val$Tip_percent,lm.predict$fit)
#Accuracy of the model
lm.acc.val <- accuracy(clean_datetime.val$Tip_percent,lm.predict$fit)

# test data
```

```r
lm.predict <- predict(lm.model,clean_datetime.test,se.fit = TRUE, interval = "confidence",level = 0.95)
# Absolute Mean error of lm model
lm.mae.test <- MAE(clean_datetime.test$Tip_percent,lm.predict$fit)
#RMSE of lm model
lm.mse.test <- RMSE(clean_datetime.test$Tip_percent,lm.predict$fit)
#Accuracy of the model
lm.acc.test <- accuracy(clean_datetime.test$Tip_percent,lm.predict$fit)



values_s <- data.frame(c('MAE','RMSE','Accuracy')
                     ,c(lm.mae.train,lm.mse.train,lm.acc.train)
                     ,c(lm.mae.val,lm.mse.val,lm.acc.val)
                     ,c(lm.mae.test,lm.mse.test,lm.acc.test))

colnames(values_s) <- c('measure','Train','Validation','Test')

lm.all.model <- lm(Tip_percent~., data = clean_datetime_1[sample(nrow(clean_datetime_1) , 100000),])

summary(lm.all.model)

##
## Call:
## lm(formula = Tip_percent ~ ., data = clean_datetime_1[sample(nrow(clean_datetime_1),
##     1e+05), ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.5509 -0.0108  0.0105  0.0229  0.4848
##
## Coefficients:
##                         Estimate Std. Error  t value Pr(>|t|)
## (Intercept)            1.791e-01  1.031e-02   17.375  < 2e-16 ***
## VendorID              -1.501e-03  4.042e-04   -3.713 0.000205 ***
## Store_and_fwd_flag    -3.344e-03  2.394e-03   -1.397 0.162488
## RateCodeID            -3.781e-03  7.155e-03   -0.528 0.597209
```

```
## Pickup_longitude      -2.664e-02  2.838e-03   -9.388  < 2e-16 ***
## Pickup_latitude       -8.402e-03  1.855e-03   -4.530 5.89e-06 ***
## Dropoff_longitude     -2.580e-02  2.388e-03  -10.804  < 2e-16 ***
## Dropoff_latitude      -2.404e-02  2.037e-03  -11.801  < 2e-16 ***
## Passenger_count        3.019e-03  7.860e-04    3.841 0.000122 ***
## Trip_distance          1.962e-02  1.107e-02    1.773 0.076269 .
## Fare_amount           -1.157e+01  5.511e-02 -209.885  < 2e-16 ***
## Extra                 -3.053e-02  4.805e-04  -63.543  < 2e-16 ***
## MTA_tax                5.858e-03  7.381e-03    0.794 0.427380
## Tolls_amount          -1.651e+00  1.142e-02 -144.573  < 2e-16 ***
## improvement_surcharge -1.544e-03  7.298e-03   -0.212 0.832384
## Total_amount           1.204e+01  3.727e-02  323.001  < 2e-16 ***
## Trip_type              1.724e-02  1.043e-02    1.653 0.098240 .
## trip_duration         -8.998e-02  9.584e-03   -9.388  < 2e-16 ***
## hour                   2.477e-03  5.694e-04    4.350 1.36e-05 ***
## weekday_num            3.931e-05  4.794e-04    0.082 0.034644 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05153 on 99980 degrees of freedom
## Multiple R-squared:  0.5243, Adjusted R-squared:  0.5242
## F-statistic:  5799 on 19 and 99980 DF,  p-value: < 2.2e-16
```

```r
# train data
lm.all.predict <- predict(lm.all.model,clean_datetime.train,se.fit = TRUE, interval = "confidence",level =
0.95)
# Absolute Mean error of lm model
lm.all.mae.train <- MAE(clean_datetime.train$Tip_percent,lm.all.predict$fit)
#RMSE of lm model
lm.all.mse.train <- RMSE(clean_datetime.train$Tip_percent,lm.all.predict$fit)
#Accuracy of the model
lm.all.acc.train <- accuracy(clean_datetime.train$Tip_percent,lm.all.predict$fit)


# Validation data
lm.all.predict <- predict(lm.all.model,clean_datetime.val,se.fit = TRUE, interval = "confidence",level = 0.
```

```r
95)
# Absolute Mean error of lm model
lm.all.mae.val <- MAE(clean_datetime.val$Tip_percent,lm.all.predict$fit)
#RMSE of lm model
lm.all.mse.val <- RMSE(clean_datetime.val$Tip_percent,lm.all.predict$fit)
#Accuracy of the model
lm.all.acc.val <- accuracy(clean_datetime.val$Tip_percent,lm.all.predict$fit)

# test data
lm.all.predict <- predict(lm.all.model,clean_datetime.test,se.fit = TRUE, interval = "confidence",level = 0
.95)
# Absolute Mean error of lm model
lm.all.mae.test <- MAE(clean_datetime.test$Tip_percent,lm.all.predict$fit)
#RMSE of lm model
lm.all.mse.test <- RMSE(clean_datetime.test$Tip_percent,lm.all.predict$fit)
#Accuracy of the model
lm.all.acc.test <- accuracy(clean_datetime.test$Tip_percent,lm.all.predict$fit)


values <- data.frame(c('MAE','RMSE','Accuracy')
                     ,c(lm.all.mae.train,lm.all.mse.train,lm.all.acc.train)
                     ,c(lm.all.mae.val,lm.all.mse.val,lm.all.acc.val)
                     ,c(lm.all.mae.test,lm.all.mse.test,lm.all.acc.test))

colnames(values) <- c('measure','Train','Validation','Test')

# Linear Model with all variables
values_s

##    measure  Train Validation   Test
## 1      MAE   5.06       5.33   5.04
## 2     RMSE   6.74       7.14   6.73
## 3 Accuracy 11.11%     10.12% 11.68%
```

## Lasso regression

```
clean_datetime.train <- scale(clean_datetime.train)
clean_datetime.val <- scale(clean_datetime.val)
clean_datetime.test <- scale(clean_datetime.test)

lm.model <- lm(Tip_percent ~ (Store_and_fwd_flag) + poly(Pickup_longitude,Pickup_latitude,  Dropoff_longitu
de,Dropoff_latitude,degree = 3) + Extra + Trip_type + MTA_tax + Tolls_amount + poly(trip_duration, degree =
16) + log(Trip_distance) + Fare_amount, data = clean_datetime.train)

l1 <- as.matrix(clean_datetime)
x1 <- poly(clean_datetime$trip_duration, degree = 16)

l1 <- cbind(l1,x1[,-1])

g = sample(cut(
  seq(nrow(clean_datetime)),
  nrow(clean_datetime)*cumsum(c(0,spec)),
  labels = names(spec)
))

res = split(clean_datetime, g)

l1.test <- l1[sample(nrow(l1), 10000), ]
l1.train <- l1[sample(nrow(l1), 10000), ]
l1.val <- l1[sample(nrow(l1), 10000), ]

lasso.mod <- glmnet(l1.train[,-20],l1.train[,20], lambda = 5)

# train data
lm.predict <- predict(lasso.mod, newx = as.matrix(l1.train))
# Absolute Mean error of lm model
l1.mae.train <- MAE(l1.train[,20],lm.predict)
#RMSE of lm model
l1.mse.train <- RMSE(l1.train[,20],lm.predict)
#Accuracy of the model
```

```r
l1.acc.train <- accuracy(l1.train[,20],lm.predict)

# Validation data
l1.predict <- predict(lasso.mod, newx = as.matrix(l1.val))
# Absolute Mean error of lm model
l1.mae.val <- MAE(l1.val[,20],lm.predict)
#RMSE of lm model
l1.mse.val <- RMSE(l1.val[,20],lm.predict)
#Accuracy of the model
l1.acc.val <- accuracy(l1.val[,20],lm.predict)

# test data
l1.predict <- predict(lasso.mod, newx = as.matrix(l1.test))
# Absolute Mean error of lm model
l1.mae.test <- MAE(l1.test[,20],lm.predict)
#RMSE of lm model
l1.mse.test <- RMSE(l1.test[,20],lm.predict)
#Accuracy of the model
l1.acc.test <- accuracy(l1.test[,20],lm.predict)

values <- data.frame(c('MAE','RMSE','Accuracy')
                     ,c(lm.mae.train,lm.mse.train,lm.acc.train)
                     ,c(lm.mae.val,lm.mse.val,lm.acc.val)
                     ,c(lm.mae.test,lm.mse.test,lm.acc.test))

colnames(values) <- c('measure','Train','Validation','Test')

values

##    measure  Train Validation   Test
## 1      MAE   1.54       1.54   1.54
## 2     RMSE   2.00       2.01   2.03
## 3 Accuracy 55.73%     56.06% 57.15%
```

## SVM Model

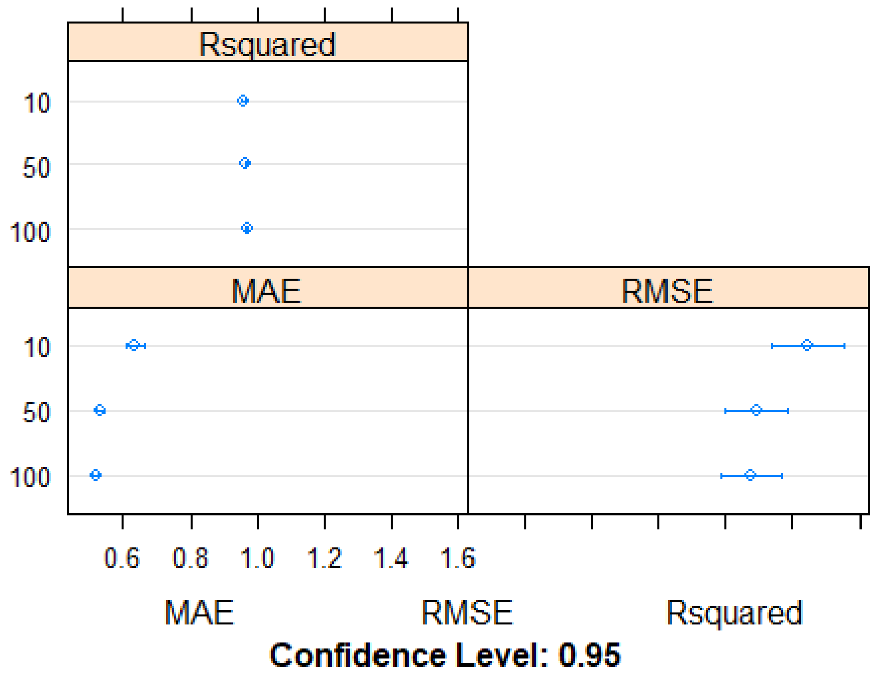
```
# Model 2 SVM
# Using grid search technique, we can find the optimal cost and gamma values
obj <- tune(svm, Tip_percent~VendorID + Pickup_longitude + Pickup_latitude +
    Dropoff_longitude + Dropoff_latitude + Passenger_count + weekday_num +
    Trip_distance + Fare_amount + Extra + MTA_tax + RateCodeID +
    Tolls_amount, data = clean_datetime.train,
            validation.x = clean_datetime.val,
              ranges = list(gamma = 2^(-1:1), cost = 2^(2:4)),
              tunecontrol = tune.control(sampling = "fix")
             )

obj

##
## Parameter tuning of 'svm':
##
## - sampling method: fixed training/validation set
##
## - best parameters:
##  gamma cost
##    0.5    4
##
## - best performance: 60.61326

# Model tuned
# Since we have high multicollinearity among our features, we will be using the linear kernel. Also selecti
ng the features that have p-values less than 0.05
svm.model <- svm(Tip_percent ~ VendorID + Pickup_longitude + Pickup_latitude +
    Dropoff_longitude + Dropoff_latitude + Passenger_count + RateCodeID +
    Trip_distance + Fare_amount + Extra + MTA_tax + weekday_num +
    Tolls_amount, kernel="linear", cost=4, gamma=0.5, clean_datetime.train)

# Use the predictions on the data
svm.predict <- predict(svm.model, clean_datetime.test)
```

```r
#Absolute Mean Error of SVM
svm.mae <- MAE(clean_datetime.test$Tip_percent,svm.predict)

#RMSE of SVM model
svm.mse <- RMSE(clean_datetime.test$Tip_percent,svm.predict)

#Accuracy of the model
svm.acc <- accuracy(clean_datetime.test$Tip_percent,svm.predict)
```

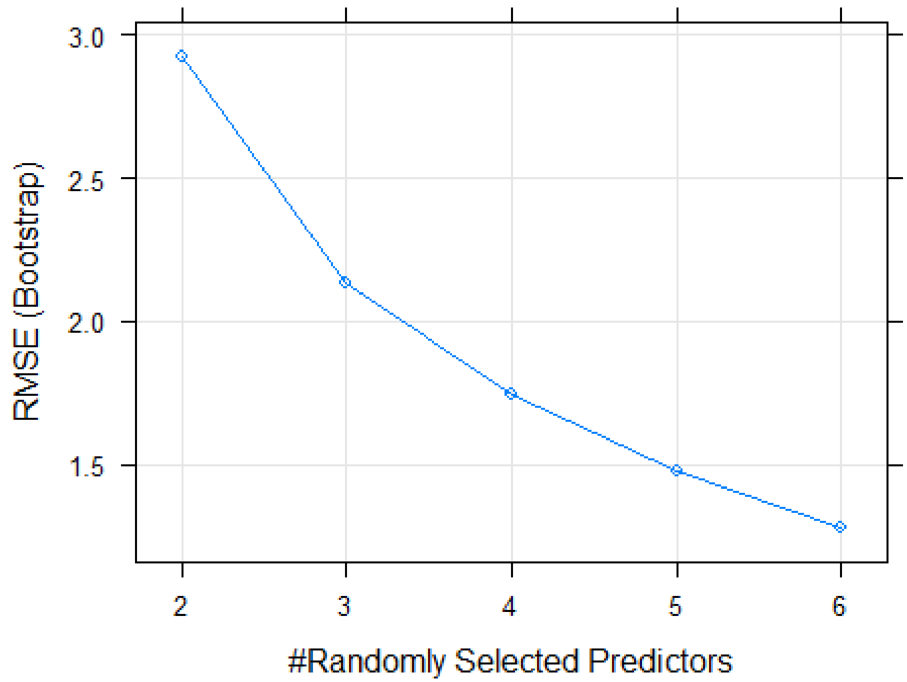
### Construction of stacked ensemble model (Random Forest)

```r
# Random forest

# Grid/Manual Search
control <- trainControl(search="grid")
x = floor(sqrt(ncol(clean_datetime)))
tunegrid <- expand.grid(.mtry=c(x-2):(x+2))
modellist <- list()
for (ntree in c(10,50,100)) {
    set.seed(21)
    fit <- train(Tip_percent ~ Pickup_longitude + Pickup_latitude + Dropoff_longitude + Dropoff_latitude + 
Passenger_count + Trip_distance + Fare_amount + Extra + MTA_tax + Tolls_amount + weekday_num + hour + Airpo
rt_ride + Speed + trip_duration
                 , data=clean_datetime.train
                 , method="rf"
                 , metric='RMSE'
                 , tuneGrid=tunegrid
                 , trControl=control
                 , ntree=ntree)

    key <- toString(ntree)
    modellist[[key]] <- fit
}

# compare results
results <- resamples(modellist)
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: 10, 50, 100
## Number of resamples: 25
##
## MAE
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## 10  0.5322117 0.5772466 0.6252153 0.6337742 0.6770370 0.7757087    0
## 50  0.4752654 0.5068742 0.5248541 0.5302574 0.5558008 0.5768740    0
## 100 0.4571882 0.5033352 0.5152869 0.5159461 0.5331733 0.5576718    0
##
## RMSE
##          Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## 10  1.0115919 1.217182 1.435603 1.446571 1.604056 1.877732    0
## 50  0.9318611 1.120927 1.277347 1.292449 1.443461 1.694966    0
## 100 0.9288438 1.103927 1.278733 1.276651 1.468736 1.661056    0
##
## Rsquared
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## 10  0.9334967 0.9529644 0.9607755 0.9587378 0.9702034 0.9787146    0
## 50  0.9496130 0.9627093 0.9696975 0.9680444 0.9755177 0.9828141    0
## 100 0.9516583 0.9611407 0.9696948 0.9690145 0.9771272 0.9830156    0

dotplot(results)
```


Rsquared
MAE
RMSE
10
50
100
0.6
0.8
1.0
1.2
1.4
1.6
MAE
RMSE
Rsquared
Confidence Level: 0.95

```
plot(fit)
```




*Ntree = 500 gives the least amount of error which is desirable. The number of variables selected is 6, which i believe is the best combination of mtry and ntree.*

```
rf.model <- randomForest(Tip_percent ~ Pickup_longitude + Pickup_latitude + Dropoff_longitude + Dropoff_lat
itude + Passenger_count + Trip_distance + Fare_amount + Extra + MTA_tax + Tolls_amount + weekday_num + hour
+ Airport_ride + Speed + trip_duration
                      , data = clean_datetime.train
                      , ntree = 500
                      , mtry = 6
                      , replace = TRUE
                      , nodesize = 5)

#Predict the outcome
```

```
rf.predict <- predict(rf.model,clean_datetime.test)

#Absolute Mean Error of random forest model
rf.mae <- MAE(clean_datetime.test$Tip_percent,rf.predict)

#RMSE of rforest model
rf.mse <- RMSE(clean_datetime.test$Tip_percent,rf.predict)

#Accuracy of the model
rf.acc <- accuracy(clean_datetime.test$Tip_percent,rf.predict)
```

## Comparing the MAE, RMSE and Accuracy

```
compare.model<-data.frame(name = c("Linear model (All variables)","SVM","Lasso Regression","RandomForest")
      ,MAE = c(lm.all.mae.test, svm.mae, l1.mae.test, rf.mae)
      ,RMSE = c(lm.all.mse.test, svm.mse, l1.mse.test, rf.mse)
      ,Accuracy = c(lm.all.acc.test, svm.acc, l1.acc.test, rf.acc))

compare.model

##                           name  MAE RMSE Accuracy
## 1 Linear model (All variables) 5.04 6.73   11.68%
## 2                          SVM 2.65 2.96   46.25%
## 3             Lasso Regression 1.54 2.03   57.15%
## 4                 RandomForest 0.34 0.85   91.98%
```

## Results

**1. Trips originating from airports John F Kennedy (JFK), Westchester and Newark (EWR) has better rewards in terms of tips.**

**2. Pickups and drop offs to the south east will lead to higher tip percentage.**

**3. Shorter the trip the better in terms of trip duration (time).**

**4. Morning 5 am is a good time to earn higher tips on days like Friday, Monday, Thursday, Tuesday and Wednesday.**

**5. At night, after 9 pm on days like Saturday, Tuesday and Wednesday are the best days to earn higher tips.**

**6. If one looks to save time and wants the best return on its distance travelled, they should work through the weekdays (Wednesday, Thursday); else, if the driver has time and wants to earn a bit more, it would be smarter to work on weekends.**

**7. Vendor VeriFone Inc has incorrectly recorded fares at times. This should be further investigated, to avoid loss and errors in data.**

**8. Among all the models, Random forest is the best model giving an accuracy of ~93%.**