# NYC\_Taxi\_Ride

*Shailaja\_Kotagiri*

*October 2, 2017*

suppressWarnings(**library**(tidyr))

suppressWarnings(**library**(dplyr))

##

## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':

##

## filter, lag

## The following objects are masked from 'package:base':

##

## intersect, setdiff, setequal, union

suppressWarnings(**library**(MASS))

##

## Attaching package: 'MASS'

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

##

## select

suppressWarnings(**library**(randomForest))

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##

## Attaching package: 'randomForest'

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

##

## combine

suppressWarnings(**library**(xgboost))

##

## Attaching package: 'xgboost'

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

##

## slice

suppressWarnings(**library**(caret))

## Loading required package: lattice

## Loading required package: ggplot2

##

## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':

##

## margin

suppressWarnings(**library**(NMOF))

suppressWarnings(**library**(zipcode))

The objective of this project is to build a model to predict the duration of taxi trips in New York city based on individual trip attributes. Duration of the trip is in seconds.

trainTrip <- read.csv("train.csv",header = T)

dim(trainTrip)

## [1] 1458644 11

glimpse(trainTrip)

## Observations: 1,458,644

## Variables: 11

## $ id <fctr> id2875421, id2377394, id3858529, id3504673...

## $ vendor\_id <int> 2, 1, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2, 2, 2, 2...

## $ pickup\_datetime <fctr> 2016-03-14 17:24:55, 2016-06-12 00:43:35, ...

## $ dropoff\_datetime <fctr> 2016-03-14 17:32:30, 2016-06-12 00:54:38, ...

## $ passenger\_count <int> 1, 1, 1, 1, 1, 6, 4, 1, 1, 1, 1, 4, 2, 1, 1...

## $ pickup\_longitude <dbl> -73.98215, -73.98042, -73.97903, -74.01004,...

## $ pickup\_latitude <dbl> 40.76794, 40.73856, 40.76394, 40.71997, 40....

## $ dropoff\_longitude <dbl> -73.96463, -73.99948, -74.00533, -74.01227,...

## $ dropoff\_latitude <dbl> 40.76560, 40.73115, 40.71009, 40.70672, 40....

## $ store\_and\_fwd\_flag <fctr> N, N, N, N, N, N, N, N, N, N, N, N, N, N, ...

## $ trip\_duration <int> 455, 663, 2124, 429, 435, 443, 341, 1551, 2...

1. Feature Engineering:

Find the observations with the same longitute and lattitude for both pick-up and drop-off locations. While these may be the obervations where the passengers may have gone for a ride around the block, these do not contribute to taxi ride duration prediction accuracy. Therefore, discarding these obseervations.

*## Observations with pickup lat-long similar to dropoff lat-long.*

dim(trainTrip[trainTrip$pickup\_latitude==trainTrip$dropoff\_latitude &

trainTrip$pickup\_longitude==trainTrip$dropoff\_longitude,])

## [1] 5897 11

*## Remove such anomalous observations.*

trainTrip <- trainTrip[!(trainTrip$pickup\_latitude==trainTrip$dropoff\_latitude &

trainTrip$pickup\_longitude==trainTrip$dropoff\_longitude),]

Convert date variables datetime data type. For ease of comparision, the following code converts all dates to EST. It also extracts pick hour, day and month.

*# Convert pickup and dropoff datetimes into POSIXct*

trainTrip$pickup\_datetime <- as.POSIXct(trainTrip$pickup\_datetime,format = "%Y-%m-%d %H:%M:%S",usetz = F,tz="EST")

trainTrip$dropoff\_datetime <- as.POSIXct(trainTrip$dropoff\_datetime,format = "%Y-%m-%d %H:%M:%S",usetz = F,tz="EST")

*# Get pickup hour*

trainTrip$PickupHour <- as.factor(format(trainTrip$pickup\_datetime, "%H"))

*# Get pickup weekday*

trainTrip$PickupWeekday <- weekdays(trainTrip$pickup\_datetime )

*# Get pickup month*

trainTrip$PickupMonth <- as.factor(format(trainTrip$pickup\_datetime, "%m"))

Trip duration should be positively correlated with distance between pickup and drop off points. Apart from using longitude and latitude parameters as they are, one can also use distance between the points. But due to the roughly spherical shape of the earth, distance between two points specified by longitude and latitude can be calculated using Haversine formula (in kilo meaters). The follwing calculates Haversine distance between pickup and dropoff points. The formula requires points to be converted to radian from degrees.

*# Function to convert degrees to radians*

deg2rad <- **function**(deg){

**return**(deg\*pi/180)

}

*# Get distance based on Haversine*

getDistance <- **function**(intDistance){

*# Radius of the earth at the equator*

R <- 6371

**return**(R \* 2\* atan2(intDistance^0.5,(1-intDistance)^0.5))

}

intermediateVal <- sin((deg2rad(trainTrip$dropoff\_latitude)-deg2rad(trainTrip$pickup\_latitude))/2)^2 +

(cos(deg2rad(trainTrip$pickup\_latitude)) \* cos(deg2rad(trainTrip$dropoff\_latitude)) \*

sin((deg2rad(trainTrip$dropoff\_longitude)-deg2rad(trainTrip$pickup\_longitude))/2)^2)

trainTrip$distance <- getDistance(intermediateVal)

However, Haversine formula gives distance analogous to euclidean distance rather than manhattan distance. But manhattan distance is more appropriate for this use case. The following code attempts to calculate manhattan distance using the same Haversine formula twice, once in direction of longitude and the second time in the direction of longitude. Then sums the two parts to arrive at manhattan distance.

*# Calculate distance using latitude alone*

lattitudinalDistance <- sin(deg2rad(trainTrip$dropoff\_latitude - trainTrip$pickup\_latitude)/2)^2

*# Calculate distance using longitude alone*

longitudinalDistance <- cos(deg2rad(trainTrip$pickup\_latitude)) \* cos(deg2rad(trainTrip$dropoff\_latitude)) \*

sin(deg2rad(trainTrip$dropoff\_longitude - trainTrip$pickup\_longitude)/2)^2

*# Sum the above to quantities to get manhattan distance*

trainTrip$ManhattanDistance <- getDistance(lattitudinalDistance) + getDistance(longitudinalDistance)

2. Data Cleaning

Examine the distribution of the response variable (trip duration): The histogram shows only a single bin with values and a long tail. Theis suggests the existance of outliers. Boxplot gives a more detailed view of the quartiles and outliers in the response variable,where as summary gives the numeric values of the quartiles. Boxplot clearly shows the apparent outliers which are beyond 1,000,000 seconds (more than 11.5 days of taxi ride!). These must be removed to be able to visualize the rest of the trip durations.

*# Histogram of trip duration*

hist(trainTrip$trip\_duration)

*# Box plot*

boxplot(trainTrip$trip\_duration)

*# Summary shows the*

summary(trainTrip$trip\_duration)

## Min. 1st Qu. Median Mean 3rd Qu. Max.

## 1 398 664 961 1076 3526000

*# dim(trainTrip[(trainTrip$trip\_duration > 1000000),])*

*# Clearly, these are outliers. Remove the obvious outliers to get a closer look.*

trainTrip <- trainTrip[!(trainTrip$trip\_duration > 1000000),]

The rest of the trip durations still have a large number of outliers. But identification of outliers needs another variable to be able to systamatically identify them. Distance is a very good measure to identify utliers in trip durations. One of the ways of identifying the outliers is using copula.

*# Box plot*

boxplot(trainTrip$trip\_duration)

Copula describes dependence between random variables. Empirical copula can be identified by the multivariate distribution of the rank transformed random varible marginal distributions. The emipical copula is then compared to a theoritical copula to identify the outliers. The follosing code plots the rank distribution of manhattan distance and trip duration. Ranks are normalized by the saample length to limit the range of ranks to 0-1.

*# Find sample size*

nsample <- nrow(trainTrip)

*# plot the empirical copula*

plot(base::rank(trainTrip$ManhattanDistance,ties.method="first")/nsample,

base::rank(trainTrip$trip\_duration,ties.method="first")/nsample,

xlab = "Distance rank",

ylab = "Trip duration rank",

main = "Empirical copula of manhattan distance and trip duration")

The resultant empirical does not represent any theoritial copulae I know (Gaussian, Gumbel, Frank or Clayton). Therefore, copula cannot be used to identify outliers. However, the plot points at the following interesting observations: 1. The thick back line of observations on the left side of the plot shows that there are many observations among which the trip duration for small distances vary greatly.

1. The bottom-right part of the plot indicates the presence of observations where it took relaatively short time to travel long distances.

The following code attempts to plot empirical copula using euclidean distance. This copula cannot be used, either.

*# Find sample size*

nsample <- nrow(trainTrip)

*# plot the empirical copula with euclidean distance*

plot(base::rank(trainTrip$distance,ties.method="first")/nsample,

base::rank(trainTrip$trip\_duration,ties.method="first")/nsample,

xlab = "Distance rank",

ylab = "Trip duration rank",

main = "Empirical copula of euclidean distance and trip duration")

Since distance and time are suppossed to be positively correlated, but the following scatter plot between distance and trip duration doesn’t show such pattern due to the outliers in distance.

Linear regression depends on the correlation structure of the response and independent variable. It may be helpful in identifying the outliers. The following code also fits a linear regression model to distance and trip duration variables.

plot(trainTrip$distance,

trainTrip$trip\_duration,

xlab = "Distance",

ylab = "Trip durationdepend",

main = "Trip duration Vs Distance")

*# Linear model using euclidean distance*

lmEuclideanDistance <- lm(trip\_duration~distance,trainTrip)

summary(lmEuclideanDistance)

##

## Call:

## lm(formula = trip\_duration ~ distance, data = trainTrip)

##

## Residuals:

## Min 1Q Median 3Q Max

## -138891 -379 -211 45 85793

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 567.1965 3.3403 169.8 <2e-16 \*\*\*

## distance 112.1083 0.6056 185.1 <2e-16 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 3138 on 1452741 degrees of freedom

## Multiple R-squared: 0.02305, Adjusted R-squared: 0.02304

## F-statistic: 3.427e+04 on 1 and 1452741 DF, p-value: < 2.2e-16

*# Linear model using euclidean distance*

lmMahnattanDistance <- lm(trip\_duration~ManhattanDistance,trainTrip)

summary(lmMahnattanDistance)

##

## Call:

## lm(formula = trip\_duration ~ ManhattanDistance, data = trainTrip)

##

## Residuals:

## Min 1Q Median 3Q Max

## -111791 -384 -212 50 85782

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 577.7161 3.3155 174.2 <2e-16 \*\*\*

## ManhattanDistance 84.3925 0.4596 183.6 <2e-16 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 3139 on 1452741 degrees of freedom

## Multiple R-squared: 0.02269, Adjusted R-squared: 0.02269

## F-statistic: 3.372e+04 on 1 and 1452741 DF, p-value: < 2.2e-16

par(mfrow =c(1,2))

qqnorm(lmEuclideanDistance$residuals)

qqline(lmEuclideanDistance$residuals)

qqnorm(lmMahnattanDistance$residuals)

qqline(lmMahnattanDistance$residuals)

Both regression equations show distances to be significant in estimating trip durations, but they result in very low r-squared (2%) due to the presence of outliers. QQplots show very heavy right tails.

In order to identify and remove outliers, now we need to chose one distance over the other. The following code identifies outliers according to both models and creates two datasets by removing outlying observations according to each model. Outlier are those observation with residuals that fall beyond 3 standard deviations.

*# calculate standard deviation of residuals of model with manhattan distance*

sdResMD <- sd(lmMahnattanDistance$residuals)

*# Find the outlier, which are beyond 3 standard deviations.*

manhattanDistanceOutlierIndex <- which((lmMahnattanDistance$residuals)/sdResMD < -3 |

(lmMahnattanDistance$residuals)/sdResMD > 3, arr.ind = T)

*# Remove outliers*

trainTripManhattanDistance <- trainTrip[-manhattanDistanceOutlierIndex,]

*# Number of outliers*

length(manhattanDistanceOutlierIndex)

## [1] 2117

*# calculate standard deviation of residuals of model with euclidean distance*

sdResED <- sd(lmEuclideanDistance$residuals)

*# Find the outlier, which are beyond 3 standard deviations.*

euclideanDistanceOutlierIndex <- which((lmEuclideanDistance$residuals)/sdResED < -3 |

(lmEuclideanDistance$residuals)/sdResED > 3, arr.ind = T)

*# remove outliers*

trainTripEuclideanDistance <- trainTrip[-euclideanDistanceOutlierIndex,]

*# Number of outliers*

length(euclideanDistanceOutlierIndex)

## [1] 2118

*#Removing the main trainTrip data frame from memory, since it will no longer be used.*

rm(trainTrip)

Fit the linear regression to check if the fit has improved.

*# Linear model without outliers*

lmwoOutliersManhattan <- lm(trip\_duration~ManhattanDistance,trainTripManhattanDistance)

*# Summary of linear model without outliers*

summary(lmwoOutliersManhattan)

##

## Call:

## lm(formula = trip\_duration ~ ManhattanDistance, data = trainTripManhattanDistance)

##

## Residuals:

## Min 1Q Median 3Q Max

## -9830.2 -250.5 -86.0 164.2 9194.0

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 414.73782 0.46935 883.6 <2e-16 \*\*\*

## ManhattanDistance 95.00312 0.06816 1393.8 <2e-16 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 430.9 on 1450624 degrees of freedom

## Multiple R-squared: 0.5725, Adjusted R-squared: 0.5725

## F-statistic: 1.943e+06 on 1 and 1450624 DF, p-value: < 2.2e-16

*# Linear model without outliers*

lmwoOutliersEuclidean <- lm(trip\_duration~distance,trainTripEuclideanDistance)

*# Summary of linear model without outliers*

summary(lmwoOutliersEuclidean)

##

## Call:

## lm(formula = trip\_duration ~ distance, data = trainTripEuclideanDistance)

##

## Residuals:

## Min 1Q Median 3Q Max

## -9220.2 -242.5 -83.7 159.7 9088.1

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 395.89419 0.46425 852.8 <2e-16 \*\*\*

## distance 128.23730 0.08852 1448.7 <2e-16 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 421.2 on 1450623 degrees of freedom

## Multiple R-squared: 0.5913, Adjusted R-squared: 0.5913

## F-statistic: 2.099e+06 on 1 and 1450623 DF, p-value: < 2.2e-16

par(mfrow =c(1,2))

*# Residual analysis*

qqnorm(lmwoOutliersManhattan$residuals)

qqline(lmwoOutliersManhattan$residuals)

*# Residual analysis*

qqnorm(lmwoOutliersEuclidean$residuals)

qqline(lmwoOutliersEuclidean$residuals)

3. Model Building

Removal of outliers has dramatically improved the fit from 2% r-squared to 59% r-squared in case of both the models. Residual QQplot still shows outliers in the form of heavy tails. However, euclidean distance model results in a slightly better r-squared (59% compared to 57% using manhattan distance). Therefore, the further analysis will consider euclidean distance.

*# Remove the manhattanDistance dataset since it will no longer be used.*

rm(trainTripManhattanDistance)

*# A function to evaluate the fit of a theritical distribution to the trip duration.*

evaluateDistributionFit <- **function**(randomVar, distribution, theoriticalDistFunction){

*#set seed to ensure repeatability*

set.seed(0)

*# Call fitdistr method to estimate distribution parameters*

fitParams <- fitdistr(randomVar, distribution)

*# Length of random variable vector*

size = length(randomVar)

*# Extract individual parameters with lapply*

listParams <- lapply(c(size,fitParams$estimate), **function**(x){x})

*# Perform ks.test on the random variable and theoritical distribution of your choice with*

*# the given distribution parameters.*

fit <- ks.test(randomVar, do.call(match.fun(theoriticalDistFunction),listParams))

*# Return the p-value*

**return**(fit$p.value)

}

exponentialPVal <- evaluateDistributionFit(trainTripEuclideanDistance$trip\_duration,"exponential","rexp")

## Warning in ks.test(randomVar, do.call(match.fun(theoriticalDistFunction), :

## p-value will be approximate in the presence of ties

gammaPVal <- evaluateDistributionFit(trainTripEuclideanDistance$trip\_duration,"gamma","rgamma")

## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced

## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced

## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced

## Warning in ks.test(randomVar, do.call(match.fun(theoriticalDistFunction), :

## p-value will be approximate in the presence of ties

poisPVal <- evaluateDistributionFit(trainTripEuclideanDistance$trip\_duration,"Poisson","rpois")

## Warning in ks.test(randomVar, do.call(match.fun(theoriticalDistFunction), :

## p-value will be approximate in the presence of ties

nbPVal <- evaluateDistributionFit(trainTripEuclideanDistance$trip\_duration,"negative binomial","rnbinom")

## Warning in ks.test(randomVar, do.call(match.fun(theoriticalDistFunction), :

## p-value will be approximate in the presence of ties

*# Distribution p-values*

(c(exponentialPVal=exponentialPVal,gammaPVal=gammaPVal,poisPVal=poisPVal,nbPVal=nbPVal))

## exponentialPVal gammaPVal poisPVal nbPVal

## 0 0 0 0

None of the distribution p-values is > 0.05. Therefore, trip duration doesn’t fit any of the distributions.

set.seed(1000)

completeIndex <- sample(c(1:nrow(trainTripEuclideanDistance)), 10000)

sampleTrain <- trainTripEuclideanDistance[completeIndex,

c("distance","vendor\_id","passenger\_count",

"store\_and\_fwd\_flag","PickupHour","PickupWeekday",

"id",

"trip\_duration","pickup\_latitude","pickup\_longitude",

"dropoff\_latitude","dropoff\_longitude")]

customRMSLE <- **function**(data, lev = NULL, model = "rf"){

data$pred <- ifelse((1+data$pred) <=0,10^-16,data$pred)

RMSLE\_val <- (sum((log(1+data$pred)-log(data$obs+1))^2)/length(data$pred))^0.5

**return**(c(RMSLE = RMSLE\_val))

}

set.seed(1000)

trCtrl <- trainControl(method = "cv", number = 10,

summaryFunction = customRMSLE)

*## Linear model*

lmTrain <- train(trip\_duration~distance+vendor\_id+passenger\_count+store\_and\_fwd\_flag+PickupHour+PickupWeekday,

data = sampleTrain,

method = "lm",

maximize = FALSE,

trControl = trCtrl)

## Warning in train.default(x, y, weights = w, ...): The metric "RMSE" was not

## in the result set. RMSLE will be used instead.

lmTrain$results

## intercept RMSLE RMSLESD

## 1 TRUE 0.5315042 0.03031915

set.seed(1000)

trCtrl <- trainControl(method = "cv", number = 10,

savePredictions = T,

summaryFunction = customRMSLE)

tuneGrd <- data.frame( mtry = c(3,4))

*## randomforest*

rfTrain <- train(trip\_duration~distance+vendor\_id+passenger\_count+store\_and\_fwd\_flag+PickupHour+PickupWeekday,

data = sampleTrain,

method = "rf",

maximize = FALSE,

trControl = trCtrl,

tuneGrid = tuneGrd,

ntree = 400)

## Warning in train.default(x, y, weights = w, ...): The metric "RMSE" was not

## in the result set. RMSLE will be used instead.

rfTrain$results

## mtry RMSLE RMSLESD

## 1 3 0.6033207 0.03569777

## 2 4 0.5558138 0.03449627

paramGrid <- expand.grid(nrounds = c(150,200,250), max\_depth = c(4,5), eta = c(0.2,0.3,0.4), gamma = 0,colsample\_bytree=1,

min\_child\_weight=1,subsample=1)

set.seed(1000)

trCtrl <- trainControl(method = "cv", number = 5,

savePredictions = T,

summaryFunction = customRMSLE)

*## Boosted trees*

xgbTrain <- train(trip\_duration~distance+vendor\_id+passenger\_count+store\_and\_fwd\_flag+PickupHour+PickupWeekday,

data = sampleTrain,

method = "xgbTree",

maximize = FALSE,

trControl = trCtrl,

tuneGrid = paramGrid)

## Loading required package: plyr

## -------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.

## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:

## library(plyr); library(dplyr)

## -------------------------------------------------------------------------

##

## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':

##

## arrange, count, desc, failwith, id, mutate, rename, summarise,

## summarize

## Warning in train.default(x, y, weights = w, ...): The metric "RMSE" was not

## in the result set. RMSLE will be used instead.

xgbTrain$results

## eta max\_depth gamma colsample\_bytree min\_child\_weight subsample nrounds

## 1 0.2 4 0 1 1 1 150

## 7 0.3 4 0 1 1 1 150

## 13 0.4 4 0 1 1 1 150

## 4 0.2 5 0 1 1 1 150

## 10 0.3 5 0 1 1 1 150

## 16 0.4 5 0 1 1 1 150

## 2 0.2 4 0 1 1 1 200

## 8 0.3 4 0 1 1 1 200

## 14 0.4 4 0 1 1 1 200

## 5 0.2 5 0 1 1 1 200

## 11 0.3 5 0 1 1 1 200

## 17 0.4 5 0 1 1 1 200

## 3 0.2 4 0 1 1 1 250

## 9 0.3 4 0 1 1 1 250

## 15 0.4 4 0 1 1 1 250

## 6 0.2 5 0 1 1 1 250

## 12 0.3 5 0 1 1 1 250

## 18 0.4 5 0 1 1 1 250

## RMSLE RMSLESD

## 1 0.4660437 0.01752168

## 7 0.4678639 0.01866259

## 13 0.4836579 0.02426312

## 4 0.4723342 0.01596459

## 10 0.4763532 0.01808730

## 16 0.4856554 0.01421869

## 2 0.4672856 0.02340275

## 8 0.4708247 0.01705957

## 14 0.4908173 0.02341181

## 5 0.4731278 0.01720715

## 11 0.4805771 0.01933679

## 17 0.4950567 0.01119193

## 3 0.4722090 0.02104228

## 9 0.4754138 0.01487000

## 15 0.4948829 0.01858898

## 6 0.4785146 0.01466246

## 12 0.4857843 0.01719000

## 18 0.5051036 0.01242471