Optimizing Data for XGBOOST

Code ▾

This is both a personal guide & my approach to the EY Data Science Challenge 2019:

- Pre-processing data
- Dectecting outliers & cleaning data
- Importance of normalisation & balancing in datasets
- Clustering methods
- Tunning using bayesian optimization & xgb.cv
- Training a model
- Evalutation & correlations
- Submissions

Context: Students will have access to a data file data_train.csv that contains the anonymized geolocation data of multiple mobile devices in the City of Atlanta (US) for 11 working days in October 2018. The devices' ID resets every 24 hours; therefore, you will not be able to trace the same device across different days. Therefore, every device ID represents a 1-day journey. Each journey is formed by several trajectories. A trajectory is defined as the route of a moving person in a straight line with an entry and an exit point.

Hide

```
head(citytrainframe) #I've already changed the id, time, & exit columns on excel.
```

	hash	id	T…	T…	T…	T…	time_diff	x_entry	y_e
	<chr>	<int>	<int>	<int>	<int>	<int>	<int>	<dbl>	<
1	0000a8602cf2def930488dee7cdad104_1	0	7	4	7	8	4	3751014	-1909
2	0000a8602cf2def930488dee7cdad104_1	1	7	20	7	25	5	3743937	-1932
3	0000a8602cf2def930488dee7cdad104_1	2	7	53	8	3	10	3744868	-1929
4	0000a8602cf2def930488dee7cdad104_1	3	8	17	8	37	20	3744880	-1929
5	0000a8602cf2def930488dee7cdad104_1	4	14	38	14	38	0	3744909	-1928
6	0000a8602cf2def930488dee7cdad104_1	5	15	2	15	18	16	3744945	-1928

6 rows | 1-10 of 12 columns

PRE-PROCESSING

What hidden data can we extract?

- After quick look of the data set provided on excell, i noticed that each hash *"0467278736f0b4abf9ea7fc75ae634ac_29"* ends in the date. Hence i was able to create two new additional pieces of data; Date & Day of the week.
- I also noticed that each trajectory *"traj_0467278736f0b4abf9ea7fc75ae634ac_29_8"* ended in the number of the trajectory and was able to seperate this into a numeric only column.
- I seperated time *"8:16:37"* into hour, minutes & time_diff
- I decided to remove the velocity columns as they had too many NA values and would be of no help.

- Distance can be calculated from the two (x1,y1) & (x2,y2) coordinates, which also means velocity can be calculated. You will noticed that i removed this data later. I will explain towards the training section.
- Lastly i removed the x_exit & y_exit of every trajectory at hour 15 & 16 so that both the test and train sets are void of bias. BALANCE!

Can you add more data?

- YES! In traffic engineering, roads use something called a k-factor to determine traffic flow. Luckily Georgia Atlanta has this informattion publically avaliable on their GDOT API, however, our data does not have longitudes & latitudes which means unfortunetly i could not sync this data.
- Weather data could have also been added, however, there was no snow in the month of October 2018 and the rain was less than 0.1mm.
- Possibly by turning the data into a widedata set by hash!

Hide

```
## LIBRARIES
suppressPackageStartupMessages({ library(tidyverse) # Clean & Visual
library(data.table) })# Clean & Read

setwd("C:/Users/ssoma/Desktop/EY Challenge")
city_test <- fread(file = "keras_test3.csv")  #Using fread for faster read times as it was a
 large dataset
city_train <- fread(file = "keras_train3.csv")

# Creating Dataframe
citytrainframe <- data.frame(city_train, stringsAsFactors = TRUE)
citytestframe <- data.frame(city_test, stringsAsFactors = TRUE)
citytrainframe <- citytrainframe[,-c(1)]
citytestframe <- citytestframe[,-c(1)]


# Combining Train and Test Data to perform analysis on.
kclusterdata <- rbind(citytrainframe,citytestframe)

# Seperating The date from Hash
kclusterdata <- kclusterdata %>% separate(hash, into = c("hash",  "date"), sep = "_")
kclusterdata$date <- as.numeric(kclusterdata$date)
kclusterdata$hash <- as.factor(kclusterdata$hash)
kclusterdata$hash <- as.numeric(kclusterdata$hash)

# Creating a Day column from the Date Column
kclusterdata$day <- cut(kclusterdata$date, breaks = c(-Inf, 1,3,5,9,11,15,19,23,25,29,31), la
bels = c(1,3,5,2,4,1,5,2,4,1,3))
kclusterdata <- kclusterdata[,c(1,14,2:13)]
kclusterdata$day <- as.numeric(kclusterdata$day)

summary(kclusterdata)
```

```
      hash              day              date              id               TENH              TENM
TEXH             TEXM           time_diff
 Min.   :     1   Min.   :1.000   Min.   : 1.00   Min.   : 0.000   Min.   : 0.00   Min.   :
0.00   Min.   : 0.00   Min.   : 0.00   Min.   :  0.000
 1st Qu.: 41966   1st Qu.:1.000   1st Qu.: 5.00   1st Qu.: 2.000   1st Qu.: 8.00   1st Qu.:1
4.00   1st Qu.: 8.00   1st Qu.:14.00   1st Qu.:  0.000
 Median : 84010   Median :3.000   Median :15.00   Median : 4.000   Median :11.00   Median :2
9.00   Median :11.00   Median :29.00   Median :  0.000
 Mean   : 83909   Mean   :2.846   Mean   :16.34   Mean   : 5.706   Mean   :10.43   Mean   :2
9.24   Mean   :10.53   Mean   :29.28   Mean   :  5.779
 3rd Qu.:125844   3rd Qu.:4.000   3rd Qu.:25.00   3rd Qu.: 8.000   3rd Qu.:14.00   3rd Qu.:4
4.00   3rd Qu.:14.00   3rd Qu.:44.00   3rd Qu.:  8.000
 Max.   :167578   Max.   :5.000   Max.   :31.00   Max.   :66.000   Max.   :16.00   Max.   :5
9.00   Max.   :16.00   Max.   :59.00   Max.   :409.000

    x_entry            y_entry               x_exit            y_exit               target
 Min.   :3741027   Min.   :-19382915   Min.   :3740998   Min.   :-19376883   Min.   :0.000
 1st Qu.:3755179   1st Qu.:-19274682   1st Qu.:3755443   1st Qu.:-19271796   1st Qu.:0.000
 Median :3760079   Median :-19230184   Median :3760075   Median :-19230377   Median :0.000
 Mean   :3760406   Mean   :-19221509   Mean   :3760416   Mean   :-19221874   Mean   :0.299
 3rd Qu.:3767500   3rd Qu.:-19169903   3rd Qu.:3767361   3rd Qu.:-19172540   3rd Qu.:1.000
 Max.   :3777099   Max.   :-19042657   Max.   :3777055   Max.   :-19045740   Max.   :1.000
                                       NA's   :167578    NA's   :167578      NA's   :16461
```

Hide

```
kclusterdata
```

hash <dbl>	day <dbl>	date <dbl>	id <int>	TENH <int>	TENM <int>	TEXH <int>	TEXM <int>	time_diff <int>	x_entry <dbl>
1	1	1	0	7	4	7	8	4	3751014
1	1	1	1	7	20	7	25	5	3743937
1	1	1	2	7	53	8	3	10	3744868
1	1	1	3	8	17	8	37	20	3744880
1	1	1	4	14	38	14	38	0	3744909
1	1	1	5	15	2	15	18	16	3744945
2	4	9	0	14	29	14	29	0	3749450
2	4	9	1	14	39	14	39	0	3749090
2	4	9	2	14	50	14	50	0	3749042
2	4	9	3	15	0	15	29	29	3749088

1-10 of 1,017,199 rows | 1-10 of 14 columns Previous **1** 2 3 4 5 6 ... 100 Next

Hide

```
NA
```

OUTLIERS

Univariate Outlier Detection Methods:

- **Tukey's method of outlier detection:** Tukey's rule says that the outliers are values more than 1.5 times the interquartile range from the quartiles — either below Q1 − 1.5IQR, or above Q3 + 1.5IQR.
- **z-score:** Using the outliers package. "z" calculates normal scores (differences between each value and the mean divided by sd).

Multivariate Outlier Detection Methods:

- **Bivariate box plots and scatter plots:** same as above but with more variables.
- **The Mahalanobis distance:** Using the MVN package.

Using trail and error i applied all the above to various aspects of the data to determine outliers. I found Mahalanobis distance to give the best results.

Hide

```
#library(MVN)
# Visualisation
## mvn(data = kclusterdata, multivariateOutlierMethod = "quan", showOutliers = TRUE)
## mvn(data = kclusterdata, multivariateOutlierMethod = "quan", multivariatePlot = "contour")

# Finding Mahalanobis Distance with the data with no NA values!
MAH1 <- kclusterdata[,c(1:6)]
MAH2 <- mahalanobis(MAH1, colMeans(MAH1), cov(MAH1))
kclusterdata$MAH <- round(MAH2, 5)

# Assigning 1 and 0 to data which i think is valid
kclusterdata$outlier_maha <- 0
kclusterdata$outlier_maha[kclusterdata$MAH > 10.5 ] <- 1 # 10.5 was decided via trial and err
or

#no_outliers <- kclusterdata[!(kclusterdata$outlier_maha== 1),]

head(kclusterdata)
```

	hash <dbl>	day <dbl>	date <dbl>	id <int>	TENH <int>	TENM <int>	TEXH <int>	TEXM <int>	time_diff <int>
1	1	1	1	0	7	4	7	8	4
2	1	1	1	1	7	20	7	25	5
3	1	1	1	2	7	53	8	3	10
4	1	1	1	3	8	17	8	37	20
5	1	1	1	4	14	38	14	38	0
6	1	1	1	5	15	2	15	18	16

6 rows | 1-10 of 16 columns

Hide

```
NA
```

Handling Outliers:

Excluding: df[is.na(df),] <- NULL

Imputing: Look up guides on mice, missForest, impute, missRanger. This approach may not always be best, sometimes its best to leave values as Na’s. Do a test of imputed data vs non-imputed to see what works for you.

Capping: NA

In this project because i assigned 1 and 0s to the Maha values therefore i had no need to remove any data. I did however, attempt to use missRanger and Mice to impute the values in a seperate trial where i combined the orginial train and test data and imputed the NA values in test but, the results were poor.

Feature extraction:

Principal Component Analysis (PCA): prcomp, caret, mlr are useful packages for this. preProcess(df, method = “pca”, thresh = 0.90) – cumulative explained variance of 90%. preProcess(df, method = “pca”, pcaComp = *number_of_dimensions_if_known*)

I did attempt some PCA, however, this showed no improvement in results and hence didn’t end up using it.

NORMALISATION

Pick the right nromalisation method for your data.

Centering and scaling: use the scale() function. For mean centering use (center = TRUE, scale = FALSE). For scaling (center = FALSE, scale = TRUE). For RMS (center = FALSE, scale = apply(df, 2, sd, na.rm = TRUE))

z score standardisation: using the scale() with the center = TRUE, scale = TRUE

Min- Max Normalisation: Feel free to use this in combination with lapply() or pipes. minmaxnormalise <- function(x) {(x-min(x,na.rm = TRUE))/(max(x, na.rm =TRUE)-min(x, na.rm = TRUE))}

Data transformation for skewed data.

Box-Cox Transformation: The caret, MASS, forecast, geoR, EnvStats, and AIS packages can all perform BCT. I personally like Forecast as it has functions to find the best lamba parameter.

Data Transformation via Mathematical Operations: Use log10(), log()

Binning/ Discretisation

Equal width (distance) binning: Use the infotheo package and the discretize() function

Equal depth (frequency) binning: discretize() function with disc = “equalfreq”

I used all the above methods when it came to normalising the data thorugh trial and error. I found that min-max, centering and scaling were the best options.

Hide

```
options(na.action='na.pass')
minmaxnormalise <- function(x) {(x-min(x,na.rm = TRUE))/(max(x, na.rm =TRUE)-min(x, na.rm = T
RUE))}

kclusterdata$xEnNorm <- kclusterdata$x_entry %>% minmaxnormalise()
kclusterdata$yEnNorm <- kclusterdata$y_entry %>% minmaxnormalise()
kclusterdata$timeNorm <- kclusterdata$time_diff %>% minmaxnormalise()

head(kclusterdata)
```

	hash <dbl>	day <dbl>	date <dbl>	id <int>	TENH <int>	TENM <int>	TEXH <int>	TEXM <int>	time_diff <int>
1	1	1	1	0	7	4	7	8	4
2	1	1	1	1	7	20	7	25	5
3	1	1	1	2	7	53	8	3	10
4	1	1	1	3	8	17	8	37	20
5	1	1	1	4	14	38	14	38	0
6	1	1	1	5	15	2	15	18	16

6 rows | 1-10 of 19 columns

Hide

```
NA
```

CLUSTERING

Pick the right clustering method for your data.

K-means Clustering:

Hierarchical clustering:

Expectation Maximization Clustering:

Fuzzy clustering:

Density-based clustering:

Model-based clustering:

Partitioning methods:

Source & read more (https://www.datanovia.com/en/blog/types-of-clustering-methods-overview-and-quick-start-r-code/)

A detailed explanation of these clustering methods are in the link above. This section of the project was by far my favourite. Clustering methods are so facinating especially when you can visualise how they work.

Of all the clustering methods k-means and EM Clustering seem to do the best job. The visual printed below doesnt show much detail, but in the high quality saved file you can see that that the clusters have a bias towards the x-axis. Would love to know more about how others have applied clustering.

theme_black() credit (https://jonlefcheck.net/2013/03/11/black-theme-for-ggplot2-2/)

Hide

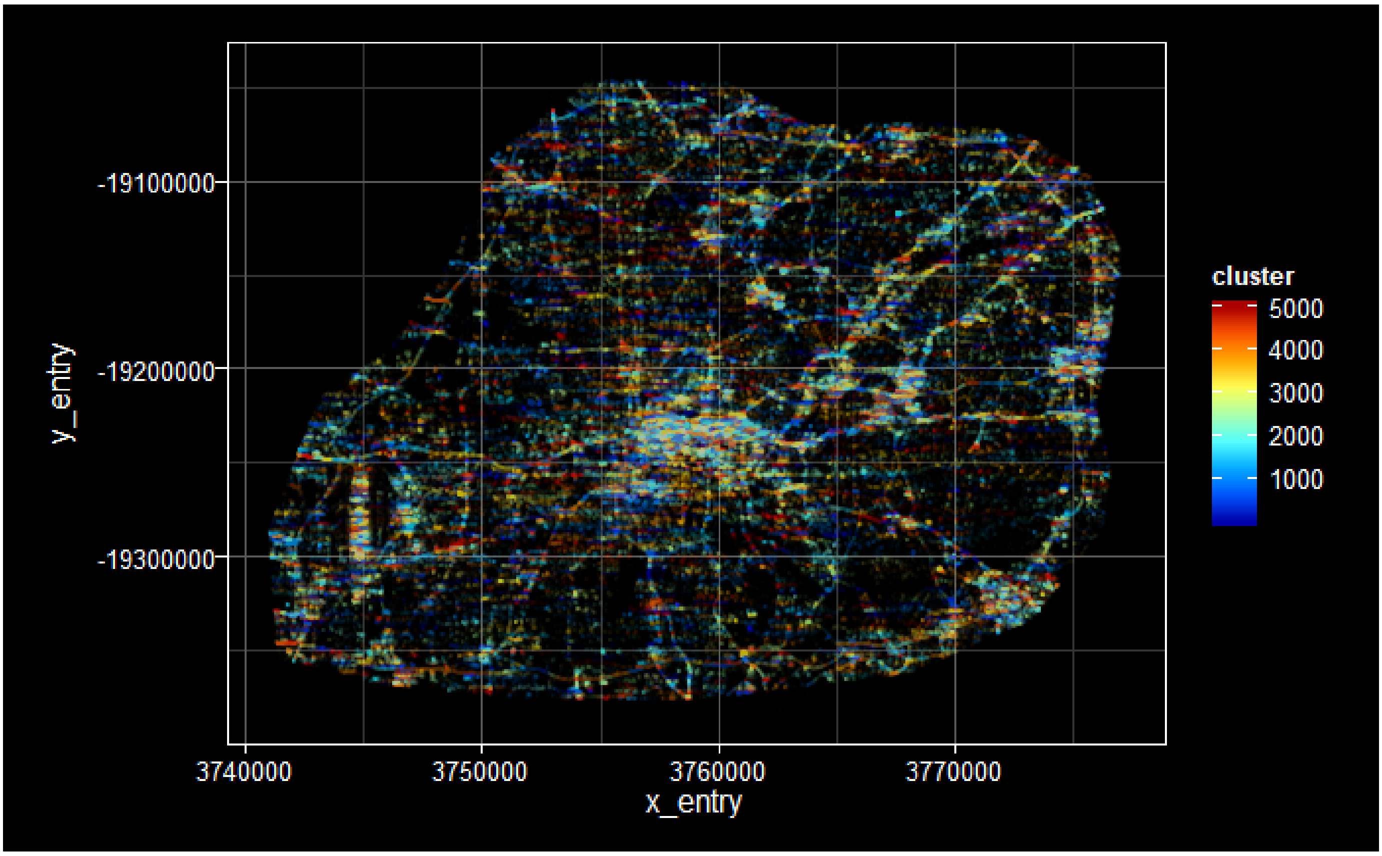
```
# Applying Clustering to the entry points
clustertrain <- kmeans(kclusterdata[,10:11], 5000, iter.max=40)
kclusterdata$cluster <- as.numeric(clustertrain$cluster)
summary(kclusterdata)
```

```
      hash              day              date              id               TENH             TENM
TEXH             TEXM          time_diff
 Min.   :     1   Min.   :1.000   Min.   : 1.00   Min.   : 0.000   Min.   : 0.00   Min.   :
0.00   Min.   : 0.00   Min.   : 0.00   Min.   :  0.000
 1st Qu.: 41966   1st Qu.:1.000   1st Qu.: 5.00   1st Qu.: 2.000   1st Qu.: 8.00   1st Qu.:1
4.00   1st Qu.: 8.00   1st Qu.:14.00   1st Qu.:  0.000
 Median : 84010   Median :3.000   Median :15.00   Median : 4.000   Median :11.00   Median :2
9.00   Median :11.00   Median :29.00   Median :  0.000
 Mean   : 83909   Mean   :2.846   Mean   :16.34   Mean   : 5.706   Mean   :10.43   Mean   :2
9.24   Mean   :10.53   Mean   :29.28   Mean   :  5.779
 3rd Qu.:125844   3rd Qu.:4.000   3rd Qu.:25.00   3rd Qu.: 8.000   3rd Qu.:14.00   3rd Qu.:4
4.00   3rd Qu.:14.00   3rd Qu.:44.00   3rd Qu.:  8.000
 Max.   :167578   Max.   :5.000   Max.   :31.00   Max.   :66.000   Max.   :16.00   Max.   :5
9.00   Max.   :16.00   Max.   :59.00   Max.   :409.000

    x_entry           y_entry              x_exit            y_exit               target
MAH            outlier_maha       xEnNorm          yEnNorm
 Min.   :3741027   Min.   :-19382915   Min.   :3740998   Min.   :-19376883   Min.   :0.000
Min.   :  0.1065   Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:3755179   1st Qu.:-19274682   1st Qu.:3755443   1st Qu.:-19271796   1st Qu.:0.000
1st Qu.:  4.0991   1st Qu.:0.00000   1st Qu.:0.3923   1st Qu.:0.3181
 Median :3760079   Median :-19230184   Median :3760075   Median :-19230377   Median :0.000
Median :  5.4787   Median :0.00000   Median :0.5281   Median :0.4489
 Mean   :3760406   Mean   :-19221509   Mean   :3760416   Mean   :-19221874   Mean   :0.299
Mean   :  6.0000   Mean   :0.06221   Mean   :0.5372   Mean   :0.4744
 3rd Qu.:3767500   3rd Qu.:-19169903   3rd Qu.:3767361   3rd Qu.:-19172540   3rd Qu.:1.000
3rd Qu.:  7.1222   3rd Qu.:0.00000   3rd Qu.:0.7339   3rd Qu.:0.6260
 Max.   :3777099   Max.   :-19042657   Max.   :3777055   Max.   :-19045740   Max.   :1.000
Max.   :133.3027   Max.   :1.00000   Max.   :1.0000   Max.   :1.0000
                                       NA's   :167578    NA's   :167578      NA's   :16461
    timeNorm          cluster
 Min.   :0.00000   Min.   :   1
 1st Qu.:0.00000   1st Qu.:1250
 Median :0.00000   Median :2510
 Mean   :0.01413   Mean   :2504
 3rd Qu.:0.01956   3rd Qu.:3763
 Max.   :1.00000   Max.   :5000
```

Hide

```
library(gridExtra)
# Black theme
theme_black = function(base_size = 12, base_family = "") {

  theme_grey(base_size = base_size, base_family = base_family) %+replace%

    theme(
      # Specify axis options
      axis.line = element_blank(),
      axis.text.x = element_text(size = base_size*0.8, color = "white", lineheight = 0.9),
      axis.text.y = element_text(size = base_size*0.8, color = "white", lineheight = 0.9),
      axis.ticks = element_line(color = "white", size  =  0.2),
      axis.title.x = element_text(size = base_size, color = "white", margin = margin(0, 10, 0
, 0)),
      axis.title.y = element_text(size = base_size, color = "white", angle = 90, margin = mar
gin(0, 10, 0, 0)),
      axis.ticks.length = unit(0.3, "lines"),
      # Specify legend options
      legend.background = element_rect(color = NA, fill = "black"),
      legend.key = element_rect(color = "white",  fill = "black"),
      legend.key.size = unit(1.2, "lines"),
      legend.key.height = NULL,
      legend.key.width = NULL,
      legend.text = element_text(size = base_size*0.8, color = "white"),
      legend.title = element_text(size = base_size*0.8, face = "bold", hjust = 0, color = "wh
ite"),
      legend.position = "right",
      legend.text.align = NULL,
      legend.title.align = NULL,
      legend.direction = "vertical",
      legend.box = NULL,
      # Specify panel options
      panel.background = element_rect(fill = "black", color  =  NA),
      panel.border = element_rect(fill = NA, color = "white"),
      panel.grid.major = element_line(color = "grey35"),
      panel.grid.minor = element_line(color = "grey20"),
      panel.spacing = unit(0.5, "lines"),
      # Specify facetting options
      strip.background = element_rect(fill = "grey30", color = "grey10"),
      strip.text.x = element_text(size = base_size*0.8, color = "white"),
      strip.text.y = element_text(size = base_size*0.8, color = "white",angle = -90),
      # Specify plot options
      plot.background = element_rect(color = "black", fill = "black"),
      plot.title = element_text(size = base_size*1.2, color = "white"),
      plot.margin = unit(rep(1, 4), "lines")

    )

}
library(colorRamps)
#Visualisation of the Clusters
ggplot(kclusterdata) + theme_black() +
 aes(x_entry, y_entry, color = cluster) +
 geom_point(alpha = 0.008, size = 0.0001) +
 scale_color_gradientn(colours=matlab.like(5000))
  #scale_colour_viridis(option = "C") # from the viridis package
```

```
ggsave("kcluster5000black.jpg", units="in", width=14, height=10, dpi=1500)
```



Hide

```
NA
NA
```

TUNING METHOD: BAYESIAN OPTIMIZATION

There are many tunning methods i've tried for this project.Bayesian Optimization with mlrMBO turned out to be the most efficient and accurate.

mlrMBO

[5.1] Bischl B, Richter J, Bossek J, Horn D, Thomas J, Lang M (2017)._mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions_. (http://arxiv.org/abs/1703.03373)

[5.2] Code sampled from Simon Coulombe (https://www.simoncoulombe.com/2019/01/bayesian/)

Hide

```
# print a summary with run
run
```

```
Recommended parameters:
eta=0.00102; gamma=2.38; max_depth=1; min_child_weight=2; subsample=0.404; colsample_bytree=
0.359
Objective: y = 0.707

Optimization path
11 + 10 entries in total, displaying last 10 (or less):
```

	eta <dbl>	gamma <dbl>	max_de... <int>	min_child_weight <int>	subsam... <dbl>	colsample_bytree <dbl>	
12	0.001664730	3.412954	5	4	0.3871472	0.5752815	0.70
13	0.001771330	3.397143	2	8	0.8515646	0.5873521	0.70
14	0.002565296	2.954818	6	10	0.2317204	0.2867190	0.70
15	0.005011954	3.287664	7	8	0.8498699	0.4858637	0.70
16	0.001008299	4.071173	1	5	0.2019193	0.9493196	0.70
17	0.001024414	2.377319	1	2	0.4039261	0.3591744	0.70
18	0.001707212	4.983310	1	1	0.2024028	0.3363391	0.70
19	0.001059178	4.119342	1	2	0.5103523	0.7686407	0.70
20	0.001012248	4.139225	1	7	0.2023848	0.6222008	0.70
21	0.001099040	4.057689	1	4	0.3737862	0.3326670	0.70

1-10 of 10 rows | 1-10 of 19 columns

Hide

```
# return  best model hyperparameters using
run$x
```

```
$eta
[1] 0.001024414

$gamma
[1] 2.377319

$max_depth
[1] 1

$min_child_weight
[1] 2

$subsample
[1] 0.4039261

$colsample_bytree
[1] 0.3591744
```

Hide

```
# return best log likelihood using
run$y
```

```
[1] 0.7073641
```

Hide

```
# return all results using
run$opt.path$env$path
```

eta <dbl>	gamma <dbl>	max_de... <int>	min_child_weight <int>	subsam... <dbl>	colsample_bytree <dbl>	
0.010000000	0.00000000	6	3	0.8000000	0.8000000	0.70
0.029653675	2.05908861	6	6	0.6746934	0.7706236	0.69
0.023476226	4.40334366	10	8	0.5701985	0.4679024	0.69
0.042117124	4.92683403	4	1	0.8041056	0.3855289	0.69
0.033275426	0.07700364	5	4	0.7322089	0.6985217	0.69
0.001317531	3.05007313	9	10	0.8501541	0.9457132	0.70
0.048554996	1.00709307	2	5	0.3517939	0.5348991	0.69
0.037636391	1.86700871	8	9	0.3628718	0.2511837	0.69
0.007812635	0.54903425	7	2	0.9457862	0.3572049	0.70
0.013350092	3.82396167	1	3	0.2494433	0.6396891	0.70

1-10 of 21 rows Previous **1** 2 3 Next

TRAINING A MODEL

Using the best parameters from above.

Hide

```
start.time <- Sys.time()
bst_model <- xgb.train(
  data = dtrain,
  nrounds = 100,
  objective = "binary:logistic",
  booster = "gbtree",
  eval_metric = "error",
 # watchlist = watchlist,
  nfolds = 10,
  eta = 0.001099040 ,
  max.depth = 1  ,
  min_child_weight= 4,
  gamma = 4.057689, #makes it more conservation, to avoid overfitting
  subsample = 0.3737862, #lower values prevents overfitting
  colsample_bytree = 0.3326670  ,
  missing = NA,
  seed = 333)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
```

```
Time difference of 17.62415 secs
```

EVALUATION

Hide

```
# Training & test error plot
e <- data.frame(bst_model$evaluation_log)
#plot(e$iter, e$train_mlogloss, col = 'blue')
#lines(e$iter, e$test_mlogloss, col = 'red')

# Feature importance
imp <- xgb.importance(colnames(dtrain), model = bst_model)
print(imp)
```

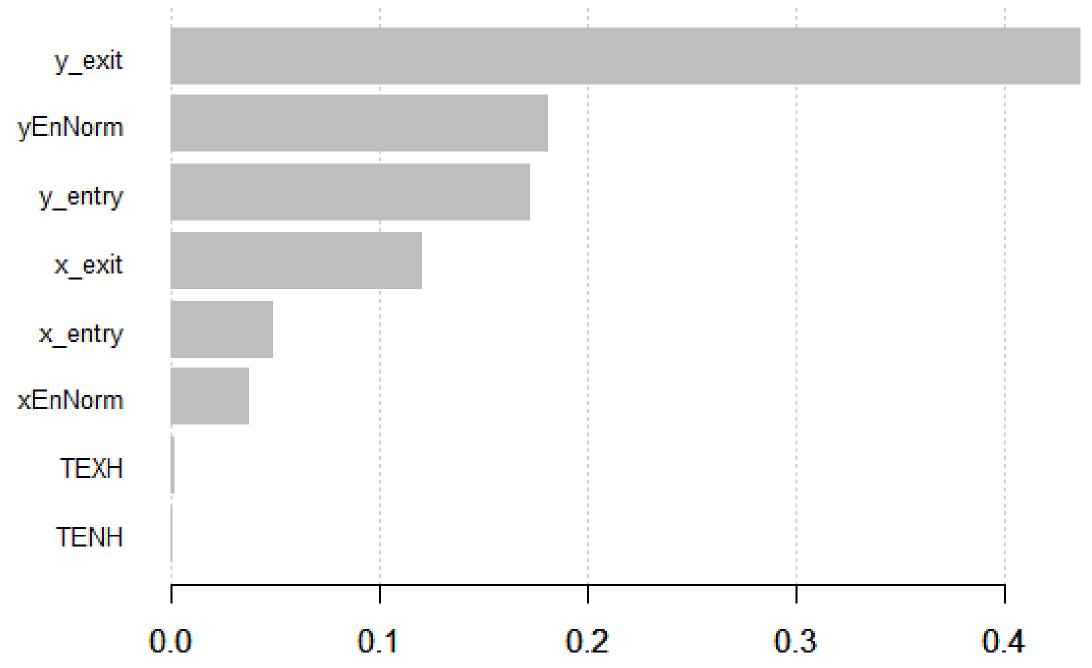
Feature <chr>	Gain <dbl>	Cover <dbl>	Frequency <dbl>
y_exit	4.370654e-01	0.3189249290	0.3198
yEnNorm	1.808232e-01	0.1626566431	0.1509
y_entry	1.725442e-01	0.1535352901	0.1421
x_exit	1.202370e-01	0.1588621476	0.1735
x_entry	4.905289e-02	0.0817891288	0.0858
xEnNorm	3.757046e-02	0.0624352387	0.0653
TEXH	1.563266e-03	0.0251088152	0.0266
TENH	6.677610e-04	0.0154207922	0.0159
id	2.104585e-04	0.0082479186	0.0061

Feature <chr>	Gain <dbl>	Cover <dbl>	Frequency <dbl>
time_diff	1.412149e-04	0.0053504343	0.0066

1-10 of 14 rows Previous **1** 2 Next

Hide

```
xgb.plot.importance (importance_matrix = imp[1:8]) #top 8
```



SUBMISSION

Hide

```
head(citytestframe)
```

	hash <dbl>	day <dbl>	date <dbl>	id <int>	TENH <int>	TENM <int>	TEXH <int>	TEXM <int>	time_diff <int>
814263	8	2	31	0	11	43	11	50	7
814264	8	2	31	2	12	21	12	21	0
814265	8	2	31	3	12	34	13	14	40
814266	8	2	31	4	13	25	13	43	18
814267	8	2	31	5	15	3	15	10	7
814268	11	5	25	0	8	8	8	20	12

6 rows | 1-10 of 20 columns

Hide

```
btestm <- sparse.model.matrix(target~.-1,citytestframe)
btest_label <- citytestframe[,"target"]
btest_matrix <- xgb.DMatrix(data = (btestm), label = btest_label)
target <- predict(bst_model, newdata = btest_matrix)
sub1 <- read.csv("data_test.csv")
sub2 <- cbind(sub1[,c(3,11)],target)
sub3 <- sub2[(is.na(sub2$x_exit)),]
write.csv(sub3[,c(1,3)], file = "xgbpredNEW7.csv")
sub3[,c(1,3)]
```

	trajectory_id <fctr>	target <dbl>
5	traj_00032f51796fd5437b238e3a9823d13d_31_5	0.11506353
8	traj_000479418b5561ab694a2870cc04fd43_25_10	0.21788977
11	traj_000506a39775e5bca661ac80e3f466eb_29_5	0.62713480
14	traj_0005401ceddaf27a9b7f0d42ef1fbe95_1_4	0.25364435
18	traj_00063a4f6c12e1e4de7d876580620667_3_4	0.19175792
24	traj_0006535be25bb52dd06983447880c964_5_12	0.11517836
28	traj_0006f84bb33ec929d1cda7686f861d0a_31_3	0.62509215
32	traj_00093ae562586aed0e053b8431e8ace4_23_10	0.18646917
35	traj_000c739e444a70e1804d757a0580caaa_31_3	0.62505734
40	traj_000d479078af08618bddc7f09082b8c3_11_6	0.26399857

1-10 of 33,515 rows Previous **1** 2 3 4 5 6 ... 100 Next

Hide

```
NA
```

APPENDIX

TUNING METHOD: NESTED LOOP

I used this method over night just to make sure that Bayesian didn't already give me the best results. This method is obviously very time consuming but useful if you know what range you're looking for.

Hide

```
data <- (citytrainframe)
train <- data[1:20]
options(na.action='na.pass')

# binarize all factors
library(caret)
library(Metrics)
dmy <- dummyVars(" ~ .", data = train)
pTrsf <- data.frame(predict(dmy, newdata = train))
################################################################################

# what we're trying to predict adults that make more than 50k
outcomeName <- c('target')
# list of features
predictors <- names(pTrsf)[!names(pTrsf) %in% outcomeName]

# take first 10% of the data only! Depending on your computing power
trainPortion <- floor(nrow(pTrsf)*0.1)

trainSet <- pTrsf[ 1:floor(trainPortion/2),]
testSet <- pTrsf[(floor(trainPortion/2)+1):trainPortion,]

#add eta, gamma & subsample
start.time <- Sys.time()
smallestError <- 100
for (fold in seq(6,32,1)) { #7,15,1
        for (round in seq(1,1000,1)) {  # 20,50,1
          for (etax in seq(0.001,0.25,0.001)) { #0.00,0.3, 0.02
                # train
                bst <- xgboost(data = as.matrix(trainSet[,predictors]),
                               label = trainSet[,outcomeName],
                               max.depth=8, nround=4,
                               objective = "binary:logistic", booster = "gbtree",
                               eval_metric = "error", eta = 0.038,
                               missing = NA, # max.depth = 10,
                               min_child_weight= 8, nfold = fold,
                               gamma = 0.0472, #makes it more conservation, to avoid overfitt
ing
                               subsample = 0.698, #lower values prevents overfitting
                               colsample_bytree = 0.6298,
                               seed = 333, verbose=0)

                gc()

                # predict
                predictions <- predict(bst, as.matrix(testSet[,predictors]), outputmargin=TRU
E)
                err <- rmse(as.numeric(testSet[,outcomeName]), as.numeric(predictions))

                if (err < smallestError) {
                        smallestError = err
                        print(paste(fold,err))
                }

          }
        }
}
```

```
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
```

WIDE DATA TRAINING

This method of turning the dataset into a wide dataset by hash proved to show no signficant improvement. However, i believe the reason for this is due to the amount of NA values in the data set.

hash <dbl>	day <dbl>	date <dbl>	A <int>	B <int>	C <int>	D <int>	E <int>	F <int>	G <int>
1	1	1	1	7	7	7	8	14	15
1	1	1	2	7	7	7	8	14	15
1	1	1	3	7	7	7	8	14	15
1	1	1	1	7	7	7	8	14	15
1	1	1	1	7	7	7	8	14	15
1	1	1	1	7	7	7	8	14	15
2	4	9	1	14	14	14	15	NA	NA
2	4	9	2	14	14	14	15	NA	NA
2	4	9	3	14	14	14	15	NA	NA
2	4	9	1	14	14	14	15	NA	NA

1-10 of 20 rows | 1-10 of 292 columns Previous **1** 2 Next