#Uploading libraries\

names(train)

library(ggplot2)

library(data.table)

library(leaps)

library(caret)

library(reshape2)

library(ggplot2)

#Uploading Files(

setwd("C:/Users/mvkum/Desktop")

sfcrime<-read.csv("train.csv", stringsAsFactors=FALSE)

#Developed function to extract features.

make\_vars\_date <- function(sfcrime) {

sfcrime$Years = strftime(strptime(sfcrime$Dates,

"%Y-%m-%d %H:%M:%S"),"%Y")

sfcrime$Month = strftime(strptime(sfcrime$Dates,

"%Y-%m-%d %H:%M:%S"),"%m")

sfcrime$DayOfMonth = strftime(strptime(sfcrime$Dates,

"%Y-%m-%d %H:%M:%S"),"%d")

sfcrime$Hour = strftime(strptime(sfcrime$Dates,

"%Y-%m-%d %H:%M:%S"),"%H")

sfcrime$YearsMo = paste( sfcrime$Years, sfcrime$Month ,

sep = "-" )

sfcrime$DayOfWeek = factor(sfcrime$DayOfWeek,

levels=c("Monday","Tuesday",

"Wednesday","Thursday",

"Friday","Saturday","Sunday"),

ordered=TRUE)

sfcrime$weekday = "Weekday"

sfcrime$weekday[sfcrime$DayOfWeek== "Saturday" |

sfcrime$DayOfWeek== "Sunday" |

sfcrime$DayOfWeek== "Friday" ] = "Weekend"

addr\_spl = strsplit(as.character(sfcrime$Address),"/")

sfcrime$AddressType = "Non-Intersection"

ind\_l = vector()

ind\_inxn = sapply(1:dim(sfcrime)[1],

function(x) length(addr\_spl[[x]]) == 2)

sfcrime$AddressType[ ind\_inxn ]="Intersection"

return(sfcrime)

}

train<-make\_vars\_date(sfcrime)

#Study the data

#Pie Chart

g <- make\_ring(39)

values <- lapply(1:39, function(train) sample(1:39,3))

if (interactive()) {

plot(g, vertex.shape="pie", vertex.pie=values,

vertex.pie.color=list(heat.colors(10)),

vertex.size=seq(10,30,length=39), vertex.label=train$Category)

}

#Decision Tree

# Making features appropriate for model(Ran each time if I required to make changes)

train$Category <- factor(train$Category)

train$Month <- as.numeric(train$Month)

train$Years <- as.numeric(train$Years)

train$Month <- as.numeric(train$Month)

train$Hour <- as.numeric(train$Hour)

train$adtype <- ifelse(train$AddressType == "Intersection", 1, 0)

str(train)

library(rpart)

train <- train

tree <- rpart(

Category ~ X + Y + Month + Years ,

data = train,

method = "class",

control = rpart.control(minsplit = 200, cp = 0)

)

rpart.plot::rpart.plot(tree)

library(rpart)

predicted2 <- as.data.frame(predicted2)

test <- read.csv("test.csv")

test <- read.csv("test.csv")

test <- make\_vars\_date(test)

test$Month <- as.numeric(test$Month)

test$Years <- as.numeric(test$Years)

test$Month <- as.numeric(test$Month)

test$Hour <- as.numeric(test$Hour)

predicted <- predict(object = tree, newdata = test)

final <- data.frame(Id = test$Id , predicted)

colnames(final) <- c("Id", levels(train$Category))

summary(final)

summary(predicted)

predicted2 <- predicted

predicted2 <- as.data.frame(predicted2)

predicted2[, "max"] <- apply(predicted2[, 1:39], 1, max)

Random Forest Accuracy

> fit.xgbTree <- train(Category~., data=trainData, method='xgbTree',

+ objective = 'multi:softprob',

+ trControl=control,metric = "logLoss", booster = 'gbtree',

+ objective = 'multi:softprob', eta = 1.0, gamma = 0,

+ max\_depth = 6, min\_child\_weigth = 1, max\_delta\_step = 1)

Something is wrong; all the Accuracy metric values are missing:

Accuracy Kappa

Min. : NA Min. : NA

1st Qu.: NA 1st Qu.: NA

Median : NA Median : NA

Mean :NaN Mean :NaN

3rd Qu.: NA 3rd Qu.: NA

Max. : NA Max. : NA

NA's :108 NA's :108

Multiple logLoss function

mlogloss <- function( testSet, output, targetVar ) {

# testSet - the validation set /test set

# output - resulted data frame having predicted values for each of class

# targetVar - name of the target attribute/variable/column

log\_sum <- 0

N = nrow(testSet)

for( i in 1:N ) {

curPred = output[i, testSet[i, targetVar]]

if(curPred == 0) curPred = 0.00000000001

log\_sum <- log\_sum + log( curPred )

}

mclass\_log\_loss = -1 \* log\_sum/N

#return

mclass\_log\_loss

}

# Define model.

model = Category ~ DayOfWeek + PdDistrict + X+ Y + Hour

# Set seed for reproducibility.

set.seed(1)

# Create random forest.

rfor = randomForest(model,

data = train,

ntree = 10,

importance = T)

# View feature importance Plot.

varImpPlot(rfor) #Caret Package

#One Accuracy and Another Logloss

train\_pred = data.table(predict(rfor,

newdata = train,

type = 'response'))

test\_pred = data.table(predict(rfor,

newdata = test,

type = 'prob'))

#####

##Training set accuracy

rfor #

# Add training set predictions to 'test'.

train$pred = train\_pred$V1

# View training accuracy.

print('Training Set Accuracy')

table(train$category\_predict == train$pred)

prop.table(table(train$category\_predict == train$pred))

XGBOOST

#################################################

#################load libraryd package

(library("dplyr"))

(library("readr"))

(library("reshape2"))

(library("lubridate"))

(library('ROCR'))

(library('caret'))

(library('nnet'))

(library('Rtsne'))

(library('xgboost'))

(library('corrplot'))

currentDate = Sys.Date()

#########Set the file dir

##read file

train = read\_csv("train.csv") %>% select( Dates, Category, DayOfWeek, PdDistrict, X, Y ) %>% sample\_n(100000)

##lets get levels of category

y.lvl = as.factor(train$Category)

num.class = length(unique((y.lvl)))

rcolnames = as.character(sort(unique(y.lvl)))

y = as.numeric(y.lvl)

test = read\_csv("test.csv") %>% select(Id, Dates, DayOfWeek, PdDistrict, X, Y )

train$Id = 123456789

test$Category = "dummy"

train = rbind(train, test)

##lets add month

train$month = month(train$Dates)

train$weeknum = week(train$Dates)

train$years = year(train$Dates)

##lets convert cretain coloum to factor

train$Category = as.factor(train$Category)

train$PdDistrict = as.factor(train$PdDistrict)

train$DayOfWeek = as.factor(train$DayOfWeek)

##lets do pca in X and Y

pca = preProcess(train[, c("X", "Y")], method=c( "center", "scale", "pca"))

pc = predict(pca, train[, c("X", "Y")])

##lets replace x and y with pca

train$X = pc$PC1

train$Y = pc$PC2

##lets convvert to factor

train2 = transform(train,

DayOfWeek = as.numeric(as.factor(DayOfWeek)),

PdDistrict = as.numeric(as.factor(PdDistrict)),

month = as.numeric(as.factor(month)),

weeknum = as.numeric(as.factor(weeknum)),

years = as.numeric(as.factor(years)))

##lets sperate data into test and train

test = train2 %>% filter(Category=="dummy")

train = train2 %>% filter(Id==123456789)

## do some anlysis on train

train = train %>% select(-Id, -Dates, -Category)

test = test %>% select( -Dates, -Category)

##correlation plot

corrplot.mixed(cor(train), lower="circle", upper="color",

tl.pos="lt", diag="n", order="hclust", hclust.method="complete")

##lets prepare data

# convert data to matrix

train.matrix = as.matrix(train)

mode(train.matrix) = "numeric"

test.matrix = as.matrix(test)

mode(test.matrix) = "numeric"

# convert outcome from factor to numeric matrix

# xgboost takes multi-labels in [0, numOfClass)

y = as.matrix(y-1)

# xgboost parameters

param <- list("objective" = "multi:softprob", # multiclass classification

"num\_class" = num.class, # number of classes

"eval\_metric" = "merror", # evaluation metric

"nthread" = 8, # number of threads to be used

"max\_depth" = 6, # maximum depth of tree

"eta" = 0.3, # step size shrinkage

"gamma" = 0, # minimum loss reduction

"subsample" = 1, # part of data instances to grow tree

"colsample\_bytree" = 1, # subsample ratio of columns when constructing each tree

"min\_child\_weight" = 1

# minimum sum of instance weight needed in a child

)

# set random seed, for reproducibility

set.seed(1234)

# k-fold cross validation, with timing

nround.cv = 100

system.time( modelxg.cv <- xgb.cv(param=param, data=train.matrix, label=y,

nfold=4, nrounds=nround.cv, prediction=TRUE, verbose=1) )

tail(modelxg.cv$evaluation\_log)

min.merror.idx = which.min(modelxg.cv$evaluation\_log[, test\_merror\_mean])

> min.merror.idx

[1] 92

# minimum merror

> modelxg.cv$evaluation\_log[min.merror.idx,]

iter train\_merror\_mean train\_merror\_std test\_merror\_mean test\_merror\_std

1: 92 0.6258432 0.001850702 0.7424998 0.003011229

library(data.table)

library(lubridate)

get

traindat<-fread("train.csv",data.table=FALSE)

testDF<-fread("test.csv",data.table=FALSE)

#setting data.table=FALSE so that traindat and testDF are data frames instead of data tables

traindat$Dates<-fast\_strptime(traindat$Dates, format="%Y-%m-%d %H:%M:%S", tz="UTC")

traindat$Day<-day(traindat$Dates)

traindat$Month<-month(traindat$Dates)

traindat$Year<-year(traindat$Dates)

traindat$Hour<-hour(traindat$Dates)

traindat$Minute<-minute(traindat$Dates)

traindat$Second<-second(traindat$Dates)

#got idea from https://brittlab.uwaterloo.ca/2015/11/01/KaggleSFcrime/

traindat$Night<-ifelse(traindat$Hour > 22 | traindat$Hour < 6,1,0)

traindat$Intersection<-grepl("/", traindat$Address)

traindat$Intersection<-plyr::mapvalues(traindat$Intersection,from=c("TRUE","FALSE"),to=c(1,0))

traindat\_subset<-traindat[,names(traindat)[-c(1,3,6,7,10,14,15)]]

#can't try PCA since we have categorial variables

#remove these columns because the 3rd column Descript and

#6th column Resolution are not in the test set

#the 7th column Address cannot be quantified. Maybe convert it to Zipcode?

#Can also use X and Y instead of the 7th column

testDF$Dates<-fast\_strptime(testDF$Dates, format="%Y-%m-%d %H:%M:%S", tz="UTC")

testDF$Day<-day(testDF$Dates)

testDF$Month<-month(testDF$Dates)

testDF$Year<-year(testDF$Dates)

testDF$Hour<-hour(testDF$Dates)

testDF$Minute<-minute(testDF$Dates)

testDF$Second<-second(testDF$Dates)

testDF$Night<-ifelse(testDF$Hour > 22 | testDF$Hour < 6,1,0)

testDF$Intersection<-grepl("/", testDF$Address)

testDF$Intersection<-plyr::mapvalues(testDF$Intersection,from=c("TRUE","FALSE"),to=c(1,0))

testDF\_subset<-testDF[,names(testDF)[-c(1,2,5,8)]]

#convert to sparse matrix

index <- sample(1:nrow(traindat), 600000)

traindat\_subsetCV<-traindat\_subset[index,]

categoryMatrix<-data.frame(with(traindat\_subsetCV,model.matrix(~Category+0)))

names(categoryMatrix)<-sort(unique(traindat$Category))

traindat\_subsetCV<-cbind(categoryMatrix,traindat\_subsetCV)

library(caret)

library(Metrics)

library(gbm)

library(xgboost)

library(doMC)

registerDoMC(4)

set.seed(999)

#need PdDistrct and DayOfWeek to be converted to Factor

#model.matrix then converts the factors into dummy variables, that is

#Monday = (1,0,0,...), Tuesday = (0,1,0,0,..)

m <- model.matrix(

~ PdDistrict + DayOfWeek + X + Y +Night+Intersection, data = traindat\_subsetCV

)

traindat\_subsetCV$Category<-factor(traindat\_subsetCV$Category)

num.class=length(levels(traindat\_subsetCV$Category))

levels(traindat\_subsetCV$Category)=1:num.class

ynum = as.matrix(as.integer(traindat\_subsetCV$Category)-1)

param <- list("objective" = "multi:softprob",

"eval\_metric" = "mlogloss", "nthread" = 4,

"num\_class" = num.class, "max\_depth" = 16, # maximum depth of tree

"eta" = 0.3) # step size shrinkage

modelxg.cv = xgb.cv(param=param, data = m, label = ynum,

nfold = 3, nrounds = 20) #nrounds = max number of iterations

#Of the nfold subsamples, a single subsample is retained as the validation data for testing

#the model, and the remaining nfold - 1 subsamples are used as training data.

#The cross-validation process is then repeated nrounds times, with each of the nfold subsamples

#used exactly once as the validation data

#locate iteration with lowest logloss score on validation set

min.merror.idx = which.min(modelxg.cv$test\_[, test\_mlogloss\_mean])

min.merror.idx

modelxg.cv$dt[min.merror.idx,]

#best CV was 16th iteration with cv nfolds=3 and logloss.mean=2.56

#now fit all training set, instead of just CV, onto boosting

modelxg <- xgboost(param=param, data=sparse.mat.tr, label=ynum, nrounds=min.merror.idx, verbose=0)

modelxg <- xgboost(param=param, data=sparse.mat.tr, label=ynum, nrounds=16, verbose=TRUE)

#nrounds=50, eta=.1

#eta=.01 produces training error too large, due to underfitting perhaps

#results in logloss of 2.43 on Kaggle dashboard page

testDF\_subset1<-subset(testDF, select = c("X", "Y", "DayOfWeek","PdDistrict","Intersection","Night"))

matMod.actualtest<-sparse.model.matrix(~as.factor(PdDistrict)+X+Y+DayOfWeek+

Intersection+Night, data=testDF\_subset1)

pred <- predict(modelxg, matMod.actualtest)

Multinomial logistic regression model

> library(nnet)

> setwd("C:/Users/mvkum/Desktop")

> train <- read.csv("train.csv")

> test <- read.csv("test.csv")

> train.df <- data.frame(Category = train$Category, DayOfWeek = train$DayOfWeek,

+ PdDistrict = train$PdDistrict)

> test.df <- data.frame(DayOfWeek = test$DayOfWeek, PdDistrict = test$PdDistrict)

> # Create a new column with the hour of the incident

> train.df$Hour <- sapply(train$Dates, function(x) as.integer(strftime(x, format = "%H")))

> test.df$Hour <- sapply(test$Dates, function(x) as.integer(strftime(x, format = "%H")))

> # Remove the original dataframes

> rm(train)

> rm(test)

> # Multinomial log-linear model using the day of the week, the district of the crime

> # and the hour of the incident as the predictors.

> multinom.model <- multinom(Category ~ DayOfWeek + PdDistrict + Hour, data = train.df,

+ maxit = 500)