On high level, do the following steps

1. Read the h5 file.
2. Removing the columns (variables) which are having missing values >15%.
3. Removing Rows containing NA values,as we have still good number of observation/or impute missing values
4. Dropping ID and Timestamp
5. Checking for Correlation/outlier between columns, so as to determine whether dimensions reduction is required or not
6. Saving the final dataset for model building

**Problem Definition**

A technology company dedicated to finding value in the world’s data. Since its founding in 2001, the company has built an innovative platform that combines extraordinary computing power, vast amounts of information, and advanced data science to produce breakthroughs in investment management, insurance and related fields. Economic opportunity depends on the ability to deliver singularly accurate forecasts in a world of uncertainty. By accurately predicting financial movements, you will learn about scientifically-driven approaches to unlocking significant predictive capability. The company is excited to find predictive value and gain a better understanding of the skills offered by the global data science crowd.

## Load Libraries

rhdf5 - To read h5 data file

plotly - For data visualisation

tidyr - For data manipulation

dplyr - For data manipulation

glmnet- For regularization

data.table - For faster data reading

ggplot2 - For data visualisation

corrplot - For data visualisation

Hmisc- - For data imputation

gam - For non linear modeling

sparklyR - For Spark

xgboost - For Boosting

mlr -

ModelMetrics - For model performance

caret - For modeling

#List the objects within the file to find the data group you want to read:  
  
setwd("Path/BIg Data/Assignment")

filePath = paste(getwd(),"/train.h5",sep="")  
h5ls(filePath)

## group name otype dclass dim  
## 0 / train H5I\_GROUP   
## 1 /train axis0 H5I\_DATASET STRING 111  
## 2 /train axis1 H5I\_DATASET INTEGER 1710756  
## 3 /train block0\_items H5I\_DATASET STRING 2  
## 4 /train block0\_values H5I\_DATASET INTEGER 2 x 1710756  
## 5 /train block1\_items H5I\_DATASET STRING 109  
## 6 /train block1\_values H5I\_DATASET FLOAT 109 x 1710756

#Read the HDF5 data:

data <- h5read(file=filePath, name="train")

# Checking the structure of the data

str(data)

Input file type : h5

So data was read to memory using rhdf5 package of <http://bioconductor.org/biocLite.R>

block0item <- h5read(file=filePath, name="train/block0\_items")  
block0\_values <- h5read(file=filePath, name="train/block0\_values")  
block0\_values = t(block0\_values)  
colnames(block0\_values) <- block0item  
train <- data.table(block0\_values)  
fwrite(train, "D:/BIg Data/Assignment/bloc0.csv" )  
rm(list = "block0item", "block0\_values", "train" )  
  
block1item <- h5read(file=filePath, name="train/block1\_items")  
block1\_values <- h5read(file=filePath, name="train/block1\_values")  
block1\_values <- t(block1\_values)  
colnames(block1\_values) = block1item  
train <- data.table(block1\_values)  
fwrite(train, "D:/BIg Data/Assignment/bloc1.csv" )

Removing Unnecessary variables from the memory

rm(list = "block1item", "block1\_values", "train" )

Reading it using datatable for faster computation in matrix  
block0 <- fread("D:/ BIg Data/Assignment/bloc0.csv")  
block1 <- fread("D:/ BIg Data/Assignment/bloc1.csv")

data <- cbind(block0, block1)

nrow(data)

## [1] 1710756

mydata <- data.table(data)  
rm(data)

Number of rows in df: 1710756, Number of columns in data: 111

H5close()  
rm(list = "block0", "block1")  
gc()

*#Displaying first few rows of data to check data*

head(mydata)

**Data Pre Processing :Exploration and Cleaning**

names(mydata)

## [1] "id" "timestamp" "derived\_0" "derived\_1"

## [5] "derived\_2" "derived\_3" "derived\_4" "fundamental\_0"

## [9] "fundamental\_1" "fundamental\_2" "fundamental\_3" "fundamental\_5"

## [13] "fundamental\_6" "fundamental\_7" "fundamental\_8" "fundamental\_9"

## [17] "fundamental\_10" "fundamental\_11" "fundamental\_12" "fundamental\_13"

## [21] "fundamental\_14" "fundamental\_15" "fundamental\_16" "fundamental\_17"

## [25] "fundamental\_18" "fundamental\_19" "fundamental\_20" "fundamental\_21"

## [29] "fundamental\_22" "fundamental\_23" "fundamental\_24" "fundamental\_25"

## [33] "fundamental\_26" "fundamental\_27" "fundamental\_28" "fundamental\_29"

## [37] "fundamental\_30" "fundamental\_31" "fundamental\_32" "fundamental\_33"

## [41] "fundamental\_34" "fundamental\_35" "fundamental\_36" "fundamental\_37"

## [45] "fundamental\_38" "fundamental\_39" "fundamental\_40" "fundamental\_41"

## [49] "fundamental\_42" "fundamental\_43" "fundamental\_44" "fundamental\_45"

## [53] "fundamental\_46" "fundamental\_47" "fundamental\_48" "fundamental\_49"

## [57] "fundamental\_50" "fundamental\_51" "fundamental\_52" "fundamental\_53"

## [61] "fundamental\_54" "fundamental\_55" "fundamental\_56" "fundamental\_57"

## [65] "fundamental\_58" "fundamental\_59" "fundamental\_60" "fundamental\_61"

## [69] "fundamental\_62" "fundamental\_63" "technical\_0" "technical\_1"

## [73] "technical\_2" "technical\_3" "technical\_5" "technical\_6"

## [77] "technical\_7" "technical\_9" "technical\_10" "technical\_11"

## [81] "technical\_12" "technical\_13" "technical\_14" "technical\_16"

## [85] "technical\_17" "technical\_18" "technical\_19" "technical\_20"

## [89] "technical\_21" "technical\_22" "technical\_24" "technical\_25"

## [93] "technical\_27" "technical\_28" "technical\_29" "technical\_30"

## [97] "technical\_31" "technical\_32" "technical\_33" "technical\_34"

## [101] "technical\_35" "technical\_36" "technical\_37" "technical\_38"

## [105] "technical\_39" "technical\_40" "technical\_41" "technical\_42"

## [109] "technical\_43" "technical\_44" "y"

*#Cheking the Structure and missing values in data*

str(mydata)

## Classes 'data.table' and 'data.frame': 1710756 obs. of 111 variables:

## $ id : num 10 11 12 25 26 27 31 38 39 40 ...

## $ timestamp : num 0 0 0 0 0 0 0 0 0 0 ...

## $ derived\_0 : num 0.3703 0.0148 -0.0106 NaN 0.1767 ...

## $ derived\_1 : num -0.00632 -0.03806 -0.05058 NaN -0.02528 ...

## $ derived\_2 : num 0.2228 -0.0174 3.3796 NaN -0.0577 ...

## $ derived\_3 : num -0.213 0.3207 -0.1575 NaN 0.0151 ...

## $ derived\_4 : num 0.7293 -0.0341 -0.0686 NaN 0.1809 ...

## $ fundamental\_0 : num -0.33563 0.00441 -0.15594 0.17849 0.13944 ...

## $ fundamental\_1 : num 0.113 0.114 1.219 NaN -0.126 ...

## $ fundamental\_2 : num 1.62124 -0.21018 -0.76452 -0.00726 -0.01871 ...

## $ fundamental\_3 : num -0.1794 0.2163 NaN -0.0979 0.1964 ...

## $ fundamental\_5 : num NaN 0.0967 NaN NaN NaN ...

## $ fundamental\_6 : num -0.0721 0.082 -0.0514 NaN -0.1639 ...

## $ fundamental\_7 : num 0.2492 -0.2244 -0.2583 -0.0937 -0.0198 ...

## $ fundamental\_8 : num 0.0244 -0.0855 -0.1221 -0.027 -0.0357 ...

## $ fundamental\_9 : num -0.1279 0.0248 -0.121 NaN 0.1129 ...

## $ fundamental\_10: num NaN -0.0623 -0.0583 -0.0499 0.1042 ...

## $ fundamental\_11: num 1.4127 -0.2022 -0.8995 0.0199 -0.1678 ...

## $ fundamental\_12: num -0.02958 1.74669 -0.02213 0.00936 -0.02599 ...

## $ fundamental\_13: num 1.265 -0.188 -0.079 0.23 -0.228 ...

## $ fundamental\_14: num -0.0557 -0.0347 -0.0311 0.0182 0.1396 ...

## $ fundamental\_15: num 1.5923 -0.1352 -0.1006 0.0506 -0.0756 ...

## $ fundamental\_16: num -0.285 0.306 -0.341 0.169 -0.142 ...

## $ fundamental\_17: num -0.21289 0.02759 -0.00752 -0.44493 -0.03366 ...

## $ fundamental\_18: num 0.4042 -0.2087 0.078 -0.3102 0.0218 ...

Etc…

summary(mydata)

#Result

Missing Value Treatment(Imputation)

MissingValue <- sort(colMeans(is.na(mydata)))  
MissingValue

## id timestamp technical\_22 technical\_34 y   
## 0.000000 0.000000 0.000000 0.000000 0.000000   
## technical\_7 technical\_19 technical\_21 technical\_27 technical\_35   
## 0.001379 0.001379 0.001379 0.001379 0.001379   
## technical\_36 technical\_40 technical\_13 technical\_20 technical\_30   
## 0.001379 0.001379 0.002875 0.002875 0.002875   
## technical\_2 technical\_6 technical\_11 technical\_17 technical\_43   
## 0.003011 0.003011 0.003011 0.003011 0.003383   
## technical\_14 technical\_33 fundamental\_33 fundamental\_18 fundamental\_36   
## 0.009247 0.009387 0.010547 0.011998 0.011998   
## fundamental\_48 fundamental\_45 fundamental\_59 fundamental\_42 fundamental\_0   
## 0.011998 0.012794 0.012794 0.016023 0.016416   
## fundamental\_53 technical\_0 technical\_9 technical\_12 technical\_32   
## 0.016416 0.017882 0.017882 0.017882 0.017882   
## technical\_37 technical\_38 technical\_16 technical\_18 technical\_39   
## 0.017882 0.017882 0.018713 0.018733 0.018733   
## technical\_42 fundamental\_7 fundamental\_41 technical\_41 fundamental\_19   
## 0.018733 0.020224 0.021752 0.033496 0.034897   
## fundamental\_21 technical\_29 derived\_0 derived\_1 technical\_24   
## 0.035006 0.036057 0.046549 0.051746 0.054937   
## fundamental\_17 fundamental\_10 fundamental\_62 technical\_3 fundamental\_12   
## 0.067846 0.071079 0.071079 0.076351 0.077694   
## fundamental\_20 fundamental\_32 fundamental\_25 derived\_3 fundamental\_58

## 0.077694 0.077918 0.085039 0.090204 0.097361   
## technical\_1 technical\_10 fundamental\_52 technical\_5 technical\_31   
## 0.099104 0.102199 0.105811 0.120260 0.143661   
## technical\_25 fundamental\_27 fundamental\_40 technical\_44 technical\_28   
## 0.163412 0.178534 0.178613 0.186791 0.207683   
## fundamental\_29 fundamental\_13 fundamental\_15 fundamental\_16 fundamental\_30   
## 0.217814 0.218038 0.218038 0.218038 0.218038   
## fundamental\_50 fundamental\_44 fundamental\_43 fundamental\_60 fundamental\_14   
## 0.218054 0.218224 0.218239 0.218834 0.218974   
## fundamental\_37 fundamental\_46 fundamental\_23 fundamental\_2 fundamental\_11   
## 0.218974 0.218974 0.219230 0.225322 0.225322   
## fundamental\_55 fundamental\_56 fundamental\_8 fundamental\_39 fundamental\_54   
## 0.225322 0.225322 0.230840 0.233887 0.237017   
## derived\_2 derived\_4 fundamental\_63 fundamental\_35 fundamental\_34   
## 0.244608 0.248812 0.250635 0.264892 0.270276   
## fundamental\_51 fundamental\_47 fundamental\_3 fundamental\_31 fundamental\_22   
## 0.276523 0.280938 0.286555 0.286555 0.336843   
## fundamental\_49 fundamental\_9 fundamental\_26 fundamental\_57 fundamental\_24   
## 0.336843 0.393554 0.397596 0.397596 0.400225   
## fundamental\_61 fundamental\_1 fundamental\_28 fundamental\_6 fundamental\_38   
## 0.405992 0.409715 0.417087 0.422723 0.462220   
## fundamental\_5   
## 0.557617

Remove any feature missing more than 15%

mydata <- mydata[, c("technical\_44","technical\_31", "technical\_28","technical\_25",  
 "technical\_5","technical\_1","technical\_3","technical\_24"):=NULL]

etc..

#mydata$id = NULL  
#mydata$timestamp = NULL

Checking Number of missing variables and doing Missing Values Imputation

table(is.na(mydata))

##   
## FALSE TRUE   
## 66551763 7448237

nomiss <- data.frame(lapply(mydata,function(x) {  
 if(is.numeric(x)) ifelse(is.na(x),median(x,na.rm=T),x) else x}))  
  
table(is.na(nomiss))

##   
## FALSE   
## 74000000

mydata <- data.table(nomiss)  
rm(nomiss)

b) Outlier Treatment and Variable Trasformation

#Max of Y = 0.09349781  
#Min of Y = 0.08609413  
  
max(mydata$y)

## [1] 0.09349781

min(mydata$y)

## [1] -0.08609413

mydata <- mydata[y < 0.093497 & y > -0.086094,]

c ) Important Variable Identification

#------------Find out top 10 predictors which have highest co-relation with response Y

as.matrix(head(sort(abs(apply(mydata, 2 ,function(x) { cor(mydata$y,x)})), decreasing = TRUE),11))

## [,1]  
## y 1.000000000  
## technical\_20 0.020045191  
## technical\_40 0.015913560  
## technical\_7 0.013059272  
## technical\_30 0.012890756  
## fundamental\_21 0.008138511  
## fundamental\_53 0.007797596  
## fundamental\_54 0.007312744  
etc..

#Break Data into Train and Test  
set.seed(100)  
dt = sort(sample(nrow(mydata), nrow(mydata)\*.7))  
train<-mydata[dt,]  
test<-mydata[-dt,]  
train.y <- train$y  
train$y <- NULL  
test.y <- test$y  
test$y <- NULL  
rm(mydata)

*#Correlation between derived variables*

correlations <- cor(train[,c(3:7)],use="pairwise.complete.obs",method="pearson")

**library**(corrplot)

corrplot(correlations, method="color", order="hclust")

*# checking for fundamental and technical features*

correlations <- cor(train[,c(8:70)],use="pairwise.complete.obs",method="pearson")

**#Result**

#Do input data visualization in several ways , mainly against timestamp/Y

*#The target variable, y, appears normally distributed around 0, but with peaks at the start and end. As it is financial data, it may represent price movements. However, we don't yet know time time period for price movements.*

ggplot(train, aes(x = y)) + geom\_histogram() + theme\_minimal()

#Plot

#Do some multivariate plotting also like

*#Fundamental Variables*

plot\_ly(data = train[train$id == 30, ],

x = ~timestamp,

y = ~fundamental\_0,

type = "scatter",

mode = "lines",

name = "fundamental\_0") %>%

add\_trace(y = ~fundamental\_1,

mode = "lines",

name = "fundamental\_1") %>%

add\_trace(y = ~fundamental\_2,

mode = "lines",

name = "fundamental\_2") %>%

add\_trace(y = ~fundamental\_3,

mode = "lines",

name = "fundamental\_3")

## Note down your observation from EDA and plotting like

1- Output of Y is normally distributed. We know that prices are not normally distributed but percentage gain over period of time is proven to be normally distributed. Therefore we can speculate Y to be either percentage gain or any such relative measure

2. Almost flat histogram of timestamp suggest that observation time period for all the parameters for a particular ID is common. In other terms observation for all the IDS is being taken for almost similar time period.

Etc..

**Predictive Model Building and Model prediction**

a) Linear Regression

sc <- spark\_connect(master = "local")

#copy data

train <- cbind(train, train.y)

test <- cbind(test, test.y)

train.x <- copy\_to(sc, train, "traindata", overwrite = TRUE)

test.x <- copy\_to(sc, test, "testdata", overwrite = TRUE)

spark.liner.fit <- train.x %>%

ml\_linear\_regression(train\_y~. )

## \* No rows dropped by 'na.omit' call

summary(spark.liner.fit)

# Check VIF

library(car)

b) Linear Regression With Regularization

spark.ridge.fit <- train.x %>%

ml\_linear\_regression(train\_y~., alpha = 0 ,lambda = lambda.opt.ridge)

## \* No rows dropped by 'na.omit' call

summary(spark.ridge.fit)

spark.elastic.fit <- train.x %>%

ml\_linear\_regression(train\_y~., alpha = .2 ,lambda = lambda.opt.elastic)

## \* No rows dropped by 'na.omit' call

summary(spark.elastic.fit)

fit.lasso <- glmnet(as.matrix(train), train.y, family="gaussian", alpha=1)  
fit.ridge <- glmnet(as.matrix(train), train.y, family="gaussian", alpha=0)  
fit.elnet <- glmnet(as.matrix(train), train.y, family="gaussian", alpha=.5)  
# 10-fold Cross validation for each alpha = 0, 0.1, ... , 0.9, 1.0  
# (For plots on Right)  
for (i in 0:10) {  
 assign(paste("fit", i, sep=""), cv.glmnet(as.matrix(train),train.y,   
 type.measure="mse",   
 alpha=i/10,  
 family="gaussian"))  
}  
  
# Plot solution paths:  
par(mfrow=c(3,2))  
# For plotting options, type '?plot.glmnet' in R console  
plot(fit.lasso, xvar="lambda")  
plot(fit10, main="LASSO")  
  
plot(fit.ridge, xvar="lambda")  
plot(fit0, main="Ridge")  
  
plot(fit.elnet, xvar="lambda")  
plot(fit5, main="Elastic Net")

yhat0 <- predict(fit0, s=fit0$lambda.min, newx= as.matrix(test))  
yhat1 <- predict(fit1, s=fit1$lambda.min, newx=as.matrix(test))  
yhat2 <- predict(fit2, s=fit2$lambda.min, newx=as.matrix(test)) #best case in elastic net  
yhat3 <- predict(fit3, s=fit3$lambda.min, newx=as.matrix(test))  
yhat4 <- predict(fit4, s=fit4$lambda.min, newx=as.matrix(test))  
yhat5 <- predict(fit5, s=fit5$lambda.min, newx=as.matrix(test))  
yhat6 <- predict(fit6, s=fit6$lambda.min, newx=as.matrix(test))  
yhat7 <- predict(fit7, s=fit7$lambda.min, newx=as.matrix(test))  
yhat8 <- predict(fit8, s=fit8$lambda.min, newx=as.matrix(test))  
yhat9 <- predict(fit9, s=fit9$lambda.min, newx=as.matrix(test))  
yhat10 <- predict(fit10, s=fit10$lambda.min, newx=as.matrix(test))  
  
mse0 <- mean((test.y - yhat0)^2) #ridge  
mse1 <- mean((test.y - yhat1)^2)  
mse2 <- mean((test.y - yhat2)^2)  
mse3 <- mean((test.y - yhat3)^2)  
mse4 <- mean((test.y - yhat4)^2)  
mse5 <- mean((test.y - yhat5)^2)  
mse6 <- mean((test.y - yhat6)^2)  
mse7 <- mean((test.y - yhat7)^2)  
mse8 <- mean((test.y - yhat8)^2)  
mse9 <- mean((test.y - yhat9)^2)  
mse10 <- mean((test.y - yhat10)^2) #lasso

mse0 #0.0006342551 rigde is min

Interpret the model results with visualization

c) Boosting

setDT(train)

setDT(test)

table(is.na(train))

##

## FALSE

## 5060120

table(is.na(test))

##

## FALSE

## 2168718

#Check if any variable is categorical. For using xgboost all variables should be numeric.This is done

#by one hot encoding . model.matrix is used for same.

labels <- train$train.y

ts\_label <- test$test.y

train.new <- model.matrix(~.+0,data = train[,-c("train.y"),with=F])

test.new <- model.matrix(~.+0,data = test[,-c("test.y"),with=F])

#Preparing matrix

dtrain <- xgb.DMatrix(data = train.new,label = labels)

dtest <- xgb.DMatrix(data = test.new,label=ts\_label)

#Model with default parameters

params = list( booster = "gbtree",

objective ="reg:linear",

eta=0.3,

gamma=0,

max\_depth=6,

min\_child\_weight=1,

subsample=1,

colsample\_bytree=1

)

#optimum number of trees : nrounds = 18

print(xgbcv)

#first default - model training

xgb1 <- xgb.train(

params = params,

data = dtrain,

nrounds = 18,

watchlist = list(val=dtest,train=dtrain),

print\_every\_n = 10,

early\_stopping\_rounds = 10,

maximize = F

#eval\_metric = "rmse"

)

#Prediction with test set

xgbpred <- predict(xgb1,dtest)

print(head(xgbpred))

library(ModelMetrics)

##

## Attaching package: 'ModelMetrics'

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

##

## auc

rmse(ts\_label, xgbpred)

## [1] 0.02505072

#Feature Importance

mat <- xgb.importance(feature\_names = colnames(train.new),model = xgb1)

print(mat)

#Calculate best number of trees using in built cross validation

xgbcv <- xgb.cv(params = params,

data = dtrain,

nrounds = 100,

nfold = 5,

showsd = T,

stratified = T,

print\_every\_n = 10,

early\_stopping\_rounds = 20,

maximize = F)

Interpret the model results with visualization

d) Non Linear Regression

#Create ols regression model first and we will compare non linear one with this model  
gam.ols= gam(train.y~ fundamental\_13 + fundamental\_43 +   
 technical\_20 + technical\_21, data = train)  
  
  
#GAM using smoothing cubic spline  
gam.smooth =gam(train.y~s(fundamental\_13) + s(fundamental\_43) +   
 s(technical\_20) + s(technical\_21), data = train)  
  
  
par(mfrow=c(1,4))   
plot.gam(gam.smooth , se=TRUE , col="red")

summary (gam.smooth)

Multivariate Adaptive Regression Splines(MARS) can also be used for non-linear regression models –

## Multivariate Adaptive Regression Splines

library(earth)

Interpret the model results with visualization

Analyze Results

Y is continuous , so it makes sense to compare R square and root mean square error for all the models.  This can change as for the taken set of independent variables chosen . From this particular example, we can see that non linear model provides least RMSE value.

|  |  |  |
| --- | --- | --- |
| Type of Model | R Squared | RMSE |
| Linear Regression | 0.0009605 | 0.02482 |
| LASSO | 0.002882 | 0.02483 |
| RIDGE | 0.003182 | 0.02483 |
| ELASTIC NET | 0.002727 | 0.02483 |
| XGBOOSTING | 0.002183908 | 0.02485649 |
| Non Linear |  | 0.01887384 |