Great Lakes PGPBABI -Hyderabad

Big Data Analytics– Assignment

Data Extraction, Visualization & Model Building

Group - 2

Submission By

**Chaitanya N**

**Rajesh Kumar Pandey**

**Tejanath Reddy**

**Karnam Uday**

**Lakshmi Divya Vegiraju**

**Problem statement:** 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.

Data: train.h5.zipView in a new window

Tasks (the grading will be based on the following tasks):

Make use of the following techniques for prediction:

application of linear regression

application of non-linear regression

application of LASSO and elastic net regression

application of XGBoost model

interpretation of models

Model result visualization

Input Data Visualization

**Approach:** The team followed the basic steps needed for exploratory data analysis, cleaning, transforming and visualization. The file was big and R was returning error while processing the error. Even after using (RHDF5) package in R, the processing was not happening as it would need more than 8 GB processor for simple exercise like Transform or CBIND.

The team took whatever learning they have from R and moved ahead with try to solve the problem using python.

The python code used in Spyder(Anaconda) is shared below along with visualizations.

As python was new to us, we referenced materials from other place to plug the code together.

**About the dataset:**

This dataset contains anonymized features pertaining to a time-varying value for a financial instrument. Each instrument has an id. Time is represented by the 'timestamp' feature and the variable to predict is 'y'.  It is Hadoop based format (.h5) and intended to reduce 2.5GB of excel data to just 0.5 GB data in H5 format.

#######################Code and Inferences ######################

########### Code from Spider Anaconda python console ##############

# Load necessary library for processing the train.h5 file ###############

import numpy as np

import pandas as pd

import matplotlib.pyplot as matplt

import seaborn as sns

from sklearn import linear\_model as lm

from sklearn.model\_selection import train\_test\_split

%matplotlib inline

# set maximum no. of columns to 111 (figured out in R using rhdf5 package)

pd.set\_option('display.max\_columns', 111)

# Read the data frame in the assignment file

with pd.HDFStore("C:/GLIM/BDA/Group Assignment/train.h5", "r") as train:

data\_frame = train.get("train")

####################Input Data Visualization ####################

# The top rows are shown to check variables and nature of data in the file

data\_frame.head()

data\_frame.shape

data\_frame.columns.summary

**Output:**

<bound method Index.summary of Index([u'id', u'timestamp', u'derived\_0', u'derived\_1', u'derived\_2',

u'derived\_3', u'derived\_4', u'fundamental\_0', u'fundamental\_1',

u'fundamental\_2', ...

u'technical\_36', u'technical\_37', u'technical\_38', u'technical\_39', u'technical\_40', u'technical\_41', u'technical\_42', u'technical\_43',

u'technical\_44', u'y'],

dtype='object', length=111)>

## # Inferences

#There are 111 columns in the dataset. The complete list is:

# id column - 1

# timestamp column - 1

# columns with prefix 'derived' - 5

# columns with prefix 'fundamental' - 63 - 'fundamental\_4' is missing.

# columns with prefix 'technical' - 40 - technical\_4, technical\_8,

# technical\_15, technical\_23, technical\_26 are missing.

# target variable named 'y' - 1

# The distribution of data in each of these columns

data\_frame.describe()

**Output: (Showing only few from the output)**

id timestamp derived\_0 derived\_1 derived\_2 \

count 1.710756e+06 1.710756e+06 1.637797e+06 1.629727e+06 1.312105e+06

mean 1.093858e+03 9.456257e+02 -4.536046e+00 7.729436e+11 -3.320328e-01

std 6.308563e+02 5.195685e+02 2.497382e+02 7.620606e+13 6.519810e+01

min 0.000000e+00 0.000000e+00 -2.017497e+04 -7.375435e-02 -9.848880e+03

25% 5.500000e+02 5.040000e+02 -1.449710e-01 -2.956479e-02 -5.967524e-02

50% 1.098000e+03 9.560000e+02 -8.368272e-04 5.523058e-03 2.109505e-02

75% 1.657000e+03 1.401000e+03 1.199108e-01 1.078554e-01 1.952209e-01

max 2.158000e+03 1.812000e+03 3.252527e+03 1.068448e+16 3.823001e+03

derived\_3 derived\_4 fundamental\_0 fundamental\_1 \

count 1.561285e+06 1.304298e+06 1.686809e+06 1.031686e+06

mean -5.046012e-01 1.801661e+01 -2.040938e-02 -5.703754e+08

std 1.020749e+02 9.258360e+02 2.494859e-01 7.502322e+10

min -3.434176e+04 -8.551914e+03 -2.344957e+00 -1.043737e+13

25% -1.655826e-01 -1.057050e-01 -1.996543e-01 -1.960470e-01

50% 2.475614e-03 1.175234e-02 -4.064488e-02 -7.395084e-03

75% 3.037236e-01 1.556464e-01 1.303819e-01 1.832071e-01

max 1.239737e+03 6.785965e+04 1.378195e+00 5.203165e+02 …...

technical\_39 technical\_40 technical\_41 technical\_42 technical\_43 \

count 1.690740e+06 1.708520e+06 1.666567e+06 1.690755e+06 1.706070e+06

mean -7.287001e-02 4.908321e-02 5.236218e-03 -1.699966e-02 -9.735299e-01

std 2.235729e-01 3.102316e-01 1.133733e-01 2.116284e-01 9.605551e-01

min -1.000000e+00 -5.250904e-01 -4.449529e-01 -1.000000e+00 -2.000000e+00

25% -2.203252e-05 -1.521701e-01 -7.377038e-02 -3.887695e-15 -2.000000e+00

50% -1.591224e-16 -1.476793e-02 9.782702e-05 0.000000e+00 -6.597540e-01

75% 0.000000e+00 1.772415e-01 7.855728e-02 0.000000e+00 -5.188884e-08

max 0.000000e+00 1.569265e+00 6.844833e-01 1.000000e+00 0.000000e+00

technical\_44 y

count 1.473977e+06 1.710756e+06

mean 3.881475e-04 2.217509e-04

std 3.011983e-02 2.240643e-02

min -1.265686e-01 -8.609413e-02

25% -1.998819e-02 -9.561389e-03

50% 1.117279e-05 -1.570681e-04

75% 2.047074e-02 9.520990e-03

max 1.435858e-01 9.349781e-02

# Count the number of missing values in each of the columns.

labels = []

values = []

for col in data\_frame.columns:

labels.append(col)

values.append(data\_frame[col].isnull().sum())

print(col, values[-1])

**Output: (Showing only few from the output)**

(u'id', 0)

(u'timestamp', 0)

(u'derived\_0', 72959)

(u'derived\_1', 81029)

(u'derived\_2', 398651)

(u'fundamental\_0', 23947)

(u'fundamental\_1', 679070)

(u'fundamental\_2', 368840)

(u'technical\_22', 0)

(u'technical\_24', 71146)

(u'technical\_25', 208056)

(u'technical\_27', 2420)

(u'technical\_28', 262916)

(u'technical\_29', 61615)

(u'technical\_30', 4764)

(u'technical\_31', 182678)

(u'technical\_32', 19165)

(u'technical\_33', 14535)

(u'technical\_34', 0)

(u'technical\_44', 236779)

(u'y', 0)

## # Inferences

# There are NaN values in all input columns except for the two columns :

# technical\_22 and technical\_34.

ind = np.arange(len(labels))

width = 0.9

fig, ax = matplt.subplots(figsize=(12,50))

rects = ax.barh(ind, np.array(values), color='y')

ax.set\_yticks(ind+((width)/2.))

ax.set\_yticklabels(labels, rotation='horizontal')

ax.set\_xlabel("Count of Missing Values")

ax.set\_title("Number of Missing Values in each column")

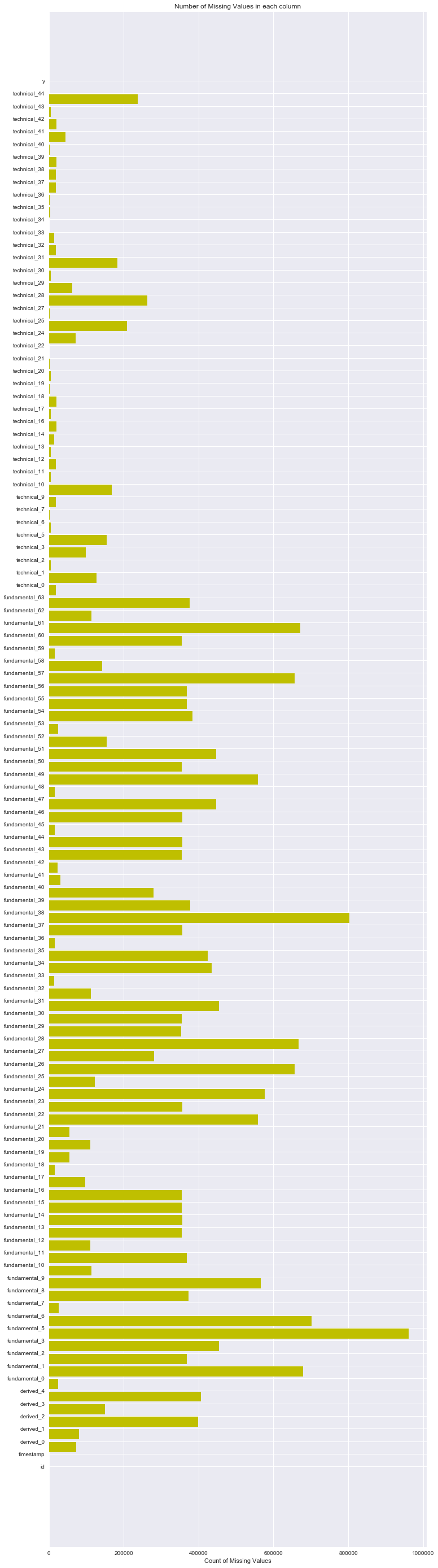
matplt.show()

**Output:**

(The graph is shown below and inferences are listed here)

# Maximum number of missing values are in Fundamental\_5 followed by fundamental\_38.

# Number of missing values are more in Fundamental\_% then Technical\_% fields.



#Let us split the data into tran and test dataset

train, test = train\_test\_split(data\_frame, test\_size = 0.2)

# Let us fix the mising values before moving ahead with analysis

mean\_values = data\_frame.mean(axis=0)

data\_frame.fillna(mean\_values, inplace=True)

data\_frame.head()

**Output (only few results shown below) :**

id timestamp derived\_0 derived\_1 derived\_2 derived\_3 derived\_4 \

0 10 0 0.370326 -6.316399e-03 0.222831 -0.213030 0.729277

1 11 0 0.014765 -3.806422e-02 -0.017425 0.320652 -0.034134

2 12 0 -0.010622 -5.057707e-02 3.379575 -0.157525 -0.068550

3 25 0 -4.536046 7.729436e+11 -0.332033 -0.504601 18.016613

4 26 0 0.176693 -2.528418e-02 -0.057680 0.015100 0.180894

fundamental\_0 fundamental\_1 fundamental\_2 fundamental\_3 fundamental\_5 \

0 -0.335633 1.132921e-01 1.621238 -0.179404 0.77524

1 0.004413 1.142851e-01 -0.210185 0.216281 0.09675

2 -0.155937 1.219439e+00 -0.764516 0.027802 0.77524

3 0.178495 -5.703754e+08 -0.007262 -0.097903 0.77524

4 0.139445 -1.256869e-01 -0.018707 0.196391 0.77524 …..

technical\_39 technical\_40 technical\_41 technical\_42 technical\_43 \

0 -0.07287 -0.414776 0.005236 -0.017 -2.0

1 -0.07287 -0.273607 0.005236 -0.017 -2.0

2 -0.07287 -0.175710 0.005236 -0.017 -2.0

3 -0.07287 -0.211506 0.005236 -0.017 -2.0

4 -0.07287 -0.001957 0.005236 -0.017 0.0

technical\_44 y

0 0.000388 -0.011753

1 0.000388 -0.001240

2 0.000388 -0.020940

3 0.000388 -0.015959

4 0.000388 -0.007338

# Now move ahead with numerical observations for which we will try to look at the correlation

# coefficients of each variables

x\_cols = [col for col in data\_frame.columns if col not in ['id','timestamp','y']]

labels = []

values = []

for col in x\_cols:

labels.append(col)

values.append(np.corrcoef(data\_frame[col].values, data\_frame.y.values)[0,1])

ind = np.arange(len(labels))

width = 0.9

fig, ax = matplt.subplots(figsize=(12,40))

rects = ax.barh(ind, np.array(values), color='y')

ax.set\_yticks(ind+((width)/2.))

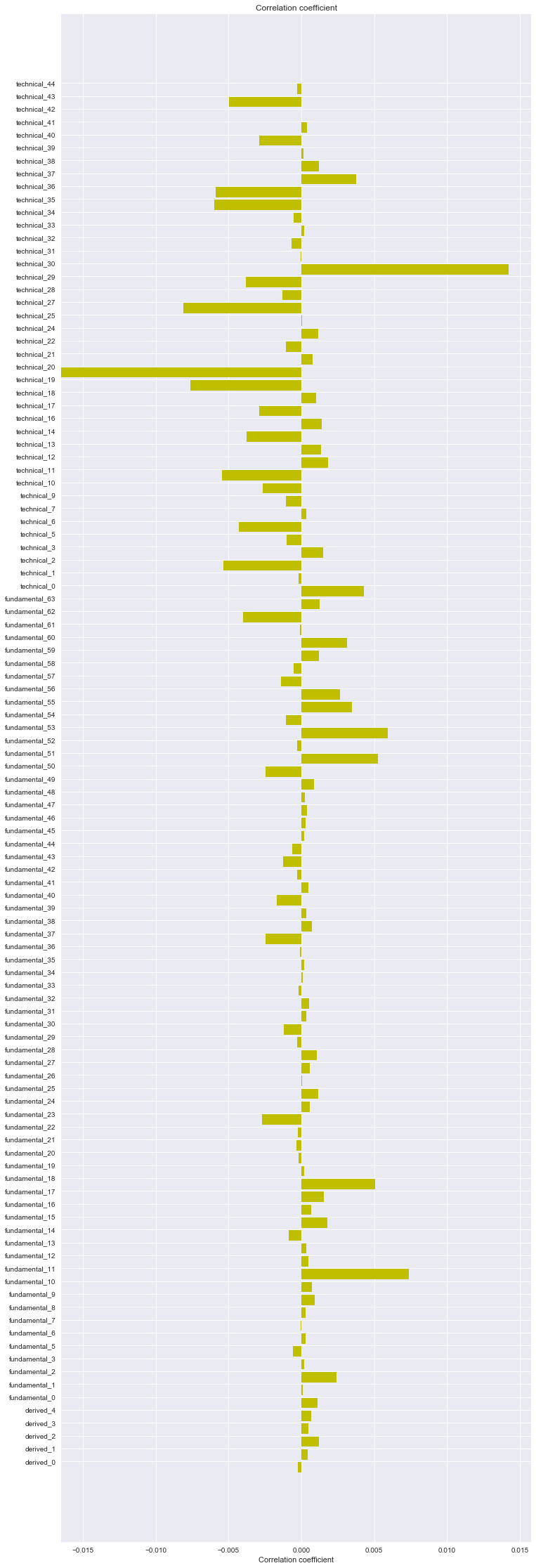
ax.set\_yticklabels(labels, rotation='horizontal')

ax.set\_xlabel("Correlation coefficient")

ax.set\_title("Correlation coefficient")

matplt.show()

**Output:**



**Inferences:**

# The correlation coefficient values are very low and the maximum value is around 0.016

# (in both positive and negative) as seen from the plot below.

# We will try to look into the top 4 variables from the plot below and do some more analysis on

# them alone.

# The fields are:

# technical\_30

# technical\_20

# technical\_19

# fundamental\_11

# Find the correlation coefficient in between top 4 variables and draw the heatmap using seaborn

cols\_to\_use = ['technical\_30', 'technical\_20', 'fundamental\_11', 'technical\_19']

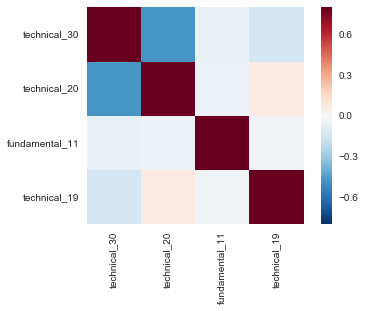
temp\_df = data\_frame[cols\_to\_use]

corrmat = temp\_df.corr(method='spearman')

f, ax = matplt.subplots(figsize=(8, 8))

sns.heatmap(corrmat, vmax=.8, square=True)

matplt.show()



#Clearly, there are some negative correlation between 'technical\_30' and 'technical\_20'.