On Hard Drives & Failure

KE 5107 CA 1 Report

Authored by:
Bharat Nagaraju (A0178258N)
Edwin Tam (A0178396J)
Vignesram Andiappan Selvaraj (A0178215A)

Contents

What in Back blazes is this about?

Context & Business

Backblaze is a cloud storage provider for developers and IT departments. They run and manage data centers to make sure there's enough capacity, redundancy, and functional hard drives to read, write, and store data.

As such, it is important that they know the "goodness" of each hard drive – so that they can replace faulty drives early and efficiently.

To achieve that, Backblaze uses SMART statistics of hard drives.

What are SMART statistics?
Also known as Self-Monitoring, Analysis and Reporting Technology, SMART statistics is a monitoring system included in hard drives that reports on various attributes of the state of a given drive.

There are about 70 SMART stats (source) available… some of which can be used to predict the failure rate of a hard drive.

However, hard drives – even those from the same manufacturer – provide inconsistent SMART stats.

What's the Business Problem?
As said earlier, Backblaze's business is based on the "goodness" of their hard drives. They need to know when a hard drive will fail early and efficiently.

Thus, we can ask:

1. Which SMART stat contributes to failure or success rates of a hard drive?
2. What would be a good model to predict hard drive failure?

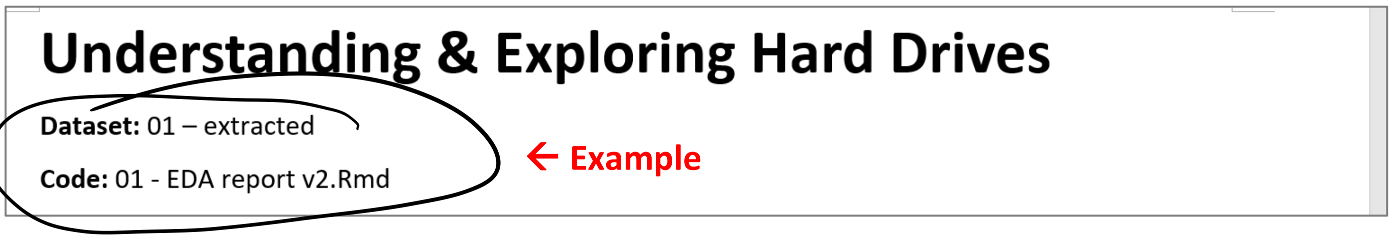
Obviously, we can use the answers to focus on a few SMART stats and models. With these questions in mind, we now look at the available datasets. **Where do we get this data?**
Backblaze logs Hard Drive Test Datasets on their webpage. Each zipped file contains daily snapshots for each of their 40,000 Hard Drives. This also includes hard drive statuses (Failure or OK).

We have selected to use 2018 dataset instead of across the years. Of note, the 2018 dataset contains additional SMART attributes which we will not use in this CA.

How to read this Report
The rest of this report will follow the content page

1. What in Back blazes is this about?
2. Data collection and Pre-processing
3. Understanding & Exploring Hard Drives
4. Preparing the Data
5. Building Models to Predict Good Hard Drives
6. Did It Work (Or Evaluating our Models)?
7. Appendix

At the beginning of each section, we will specify the Dataset & code used. All code and dataset can be found in the aptly named "Code & Dataset" folder.



Understanding & Exploring Hard Drives

Dataset: 01 – extracted

Code: 01 - EDA report v2.Rmd

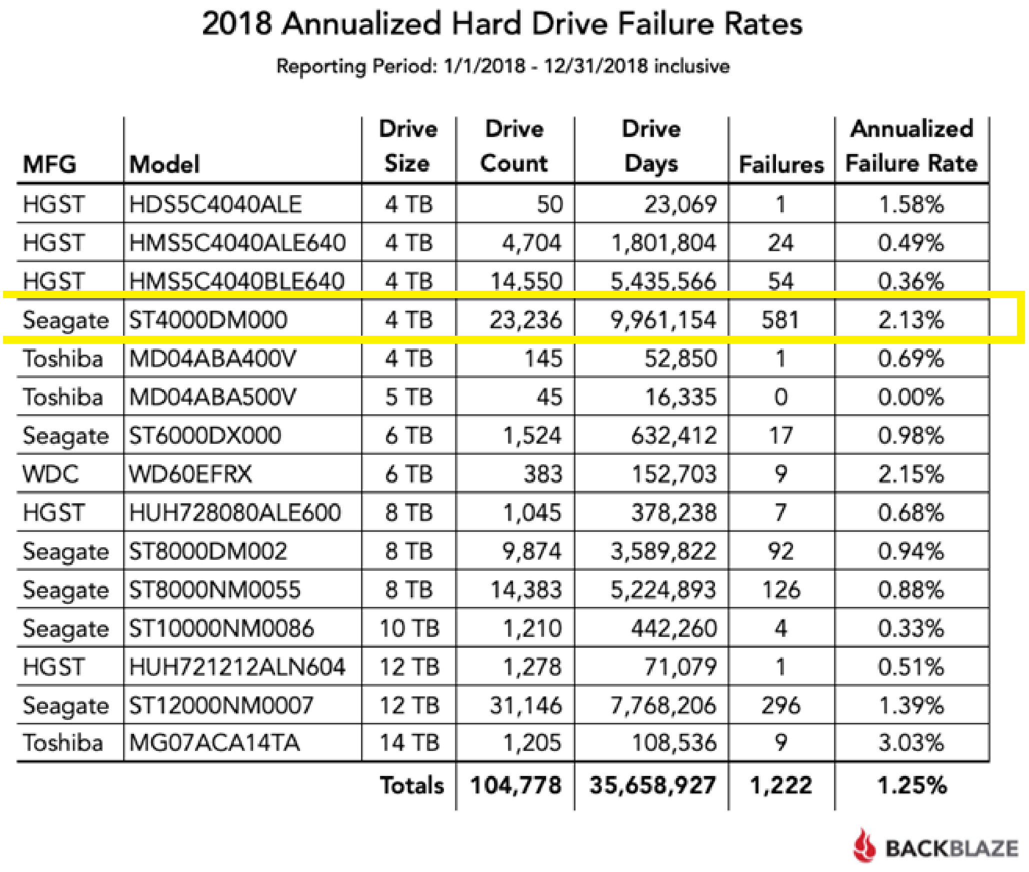
← **Example**

Data Collection and Preprocessing

Code: 00 - MyProgram_Success.scala

Data Collection

Based on the statistics published by Back blaze for the year 2018, We chose to examine and analyze the failure reasons for the hard drive that failed the most. Seagate ST4000DM000 had 581 instances of failure and was the highest among all the hard drives used by Back blaze and will be used in this project for analysis and prediction.



2018 Annualized Hard Drive Failure Rates

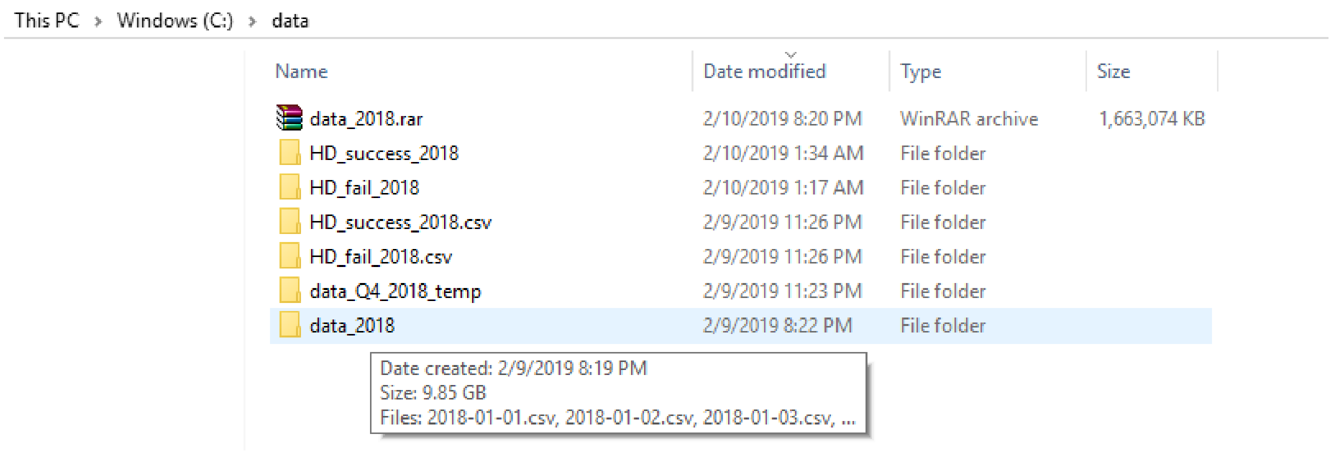
Reporting Period: 1/1/2018 - 12/31/2018 inclusive

MFG	Model	Drive Size	Drive Count	Drive Days	Failures	Annualized Failure Rate
HGST	HDS5C4040ALE	4 TB	50	23,069	1	1.58%
HGST	HMS5C4040ALE640	4 TB	4,704	1,801,804	24	0.49%
HGST	HMS5C4040BLE640	4 TB	14,550	5,435,566	54	0.36%
Seagate	ST4000DM000	4 TB	23,236	9,961,154	581	2.13%
Toshiba	MD04ABA400V	4 TB	145	52,850	1	0.69%
Toshiba	MD04ABA500V	5 TB	45	16,335	0	0.00%
Seagate	ST6000DX000	6 TB	1,524	632,412	17	0.98%
WDC	WD60EFRX	6 TB	383	152,703	9	2.15%
HGST	HUH728080ALE600	8 TB	1,045	378,238	7	0.68%
Seagate	ST8000DM002	8 TB	9,874	3,589,822	92	0.94%
Seagate	ST8000NM0055	8 TB	14,383	5,224,893	126	0.88%
Seagate	ST10000NM0086	10 TB	1,210	442,260	4	0.33%
HGST	HUH721212ALN604	12 TB	1,278	71,079	1	0.51%
Seagate	ST12000NM0007	12 TB	31,146	7,768,206	296	1.39%
Toshiba	MG07ACA14TA	14 TB	1,205	108,536	9	3.03%
		Totals	**104,778**	**35,658,927**	**1,222**	**1.25%**

BACKBLAZE

Data Pre-Processing

Data shared by back blaze is snapshot i.e. hard drives which haven't failed appears in every day s record until they fail, if they fail. However, Failed records doesn't appear from the day they fail and a replacement hard drive (identified by a new serial number) appears from the next day. In order to perform our analysis, we need to extract the data from this huge dataset. After unzipping the archive, the data amounts to a whopping 10 GB.



This PC › Windows (C:) › data

Name	Date modified	Type	Size
data_2018.rar	2/10/2019 8:20 PM	WinRAR archive	1,663,074 KB
HD_success_2018	2/10/2019 1:34 AM	File folder	
HD_fail_2018	2/10/2019 1:17 AM	File folder	
HD_success_2018.csv	2/9/2019 11:26 PM	File folder	
HD_fail_2018.csv	2/9/2019 11:26 PM	File folder	
data_Q4_2018_temp	2/9/2019 11:23 PM	File folder	
data_2018	2/9/2019 8:22 PM	File folder	

Date created: 2/9/2019 8:19 PM
Size: 9.85 GB
Files: 2018-01-01.csv, 2018-01-02.csv, 2018-01-03.csv, ...

This volume of data cannot be processed using simple R or Python packages since they are not designed to scale to such huge volume of data and use in memory processing. We tried to use SparkR and SparklyR , R packages that are provided to connect to a Spark instance for Batch processing data files. Different techniques were employed using both packages – Streaming data connector of SparkR to handle huge number of data files (365 files – each file of minimum size 40 MB) and Batch File processing using SparklyR. These packages were able to only load a maximum of 3 Gig and failed terribly beyond that. We decided to use native Spark using Scala to extract the required data. Using data tables and RDD’s, we managed to extract all the hard drives that are at least one year old and all the failed hard drive data records in order to maintain a balance in the number of success and failure records.

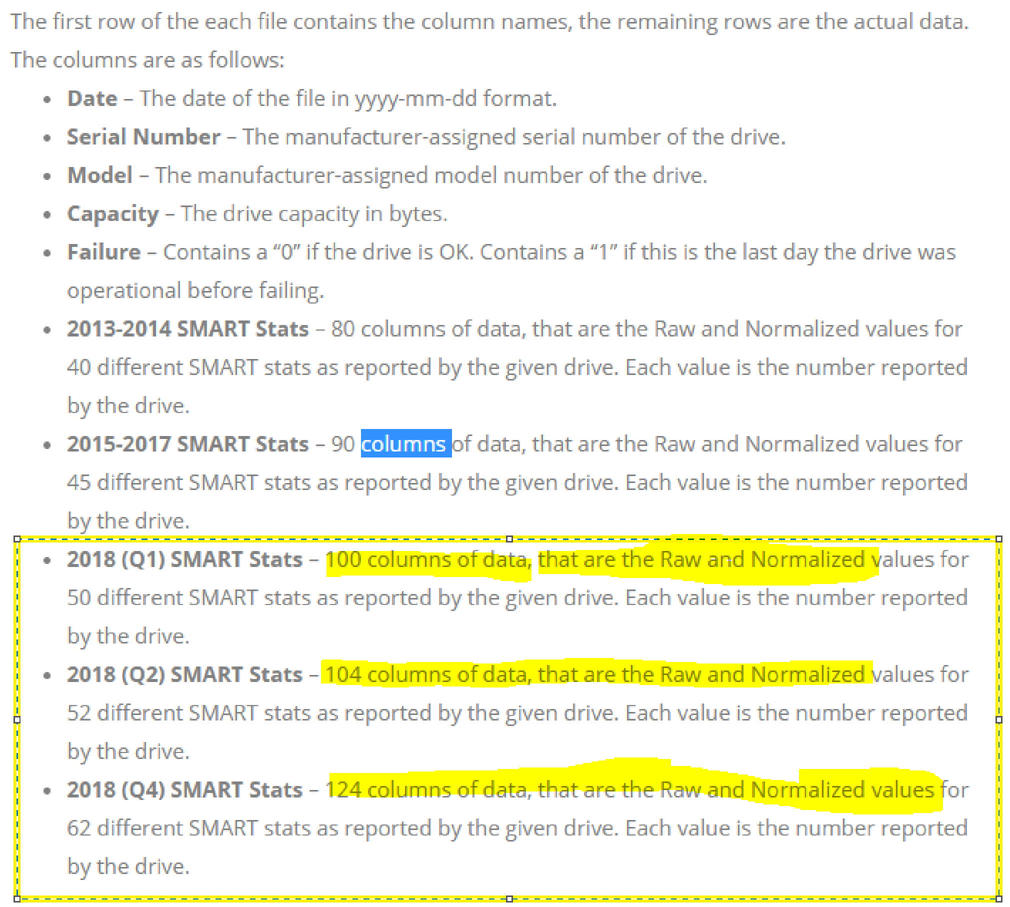
Spark Snippet:

```
val success = spark.sql("select * from (SELECT *, dense_rank()
OVER (PARTITION BY serial_number ORDER BY date DESC) as LatestRec
FROM peopleTemp ) where failure=0 and model='ST4000DM000'
and CAST(smart_9_raw as INT) > 8760 and LatestRec=1")

 success.coalesce(1).write.format("com.databricks.spark.csv")
 .option("header","true").save("file:///C:/data/HD_success_2018")
```

Here smart_9_raw is the life of hard drive. Hard drives whose life is more than 365 days (365*24 = 8760) and which haven't failed are chosen for our analysis.

```
val failures = spark.sql("SELECT * FROM peopleTemp where failure=1 and
model='ST4000DM000'")
failures.coalesce(1).write.format("com.databricks.spark.csv")
.option("header","true").save("file:///C:/data/HD_fail_2018")
```

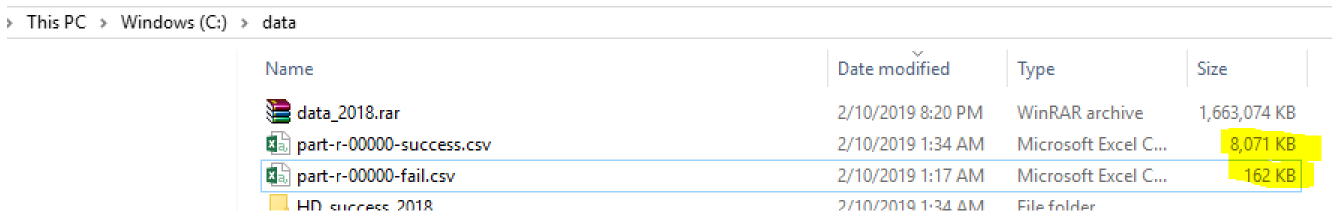
_Back blaze team has constantly evolved the data emitted out of hard drive by adding more measures / fields. Q2 / Q3 has 4 more columns compared to Q1 and Q4 has 20 more columns compared to Q2/Q3. So we are eliminating those additional columns from Q2 , Q3 and Q4 to transformed all of the dataset to one standard i.e. Q1 format.



The first row of the each file contains the column names, the remaining rows are the actual data.
The columns are as follows:

- **Date** – The date of the file in yyyy-mm-dd format.
- **Serial Number** – The manufacturer-assigned serial number of the drive.
- **Model** – The manufacturer-assigned model number of the drive.
- **Capacity** – The drive capacity in bytes.
- **Failure** – Contains a "0" if the drive is OK. Contains a "1" if this is the last day the drive was operational before failing.
- **2013-2014 SMART Stats** – 80 columns of data, that are the Raw and Normalized values for 40 different SMART stats as reported by the given drive. Each value is the number reported by the drive.
- **2015-2017 SMART Stats** – 90 columns of data, that are the Raw and Normalized values for 45 different SMART stats as reported by the given drive. Each value is the number reported by the drive.
- **2018 (Q1) SMART Stats** – 100 columns of data, that are the Raw and Normalized values for 50 different SMART stats as reported by the given drive. Each value is the number reported by the drive.
- **2018 (Q2) SMART Stats** – 104 columns of data, that are the Raw and Normalized values for 52 different SMART stats as reported by the given drive. Each value is the number reported by the drive.
- **2018 (Q4) SMART Stats** – 124 columns of data, that are the Raw and Normalized values for 62 different SMART stats as reported by the given drive. Each value is the number reported by the drive.

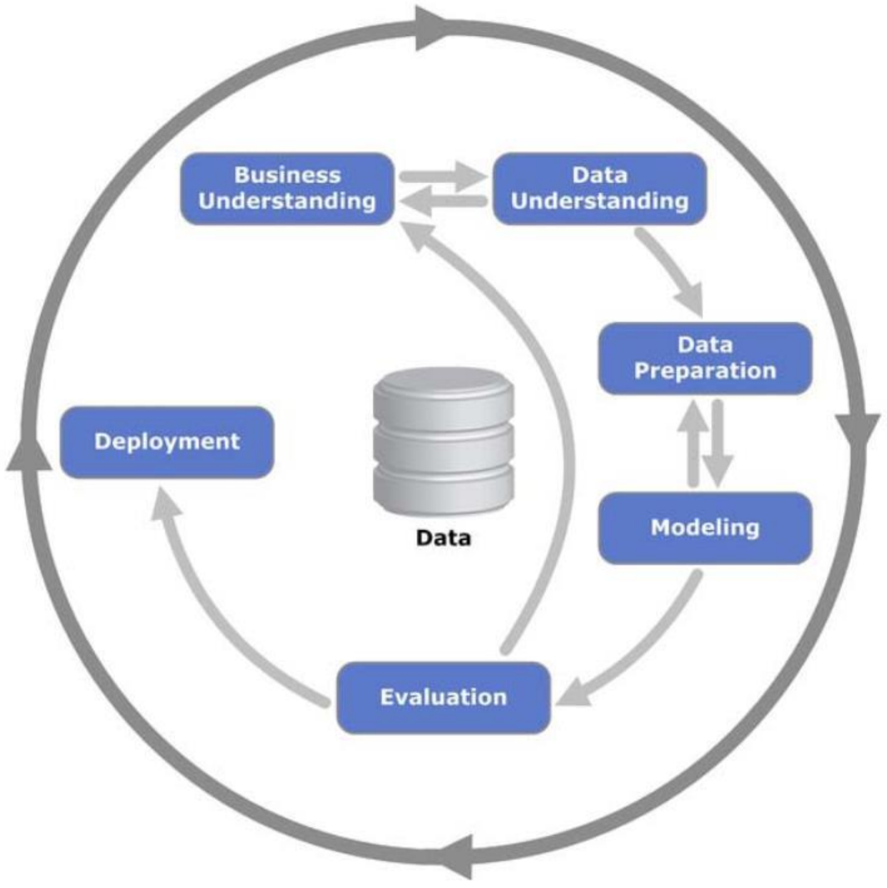
After running the preprocessing scripts, required data has been reduced to 8MB approximately and is ready for EDA / Data cleansing and modelling



› This PC › Windows (C:) › data

Name	Date modified	Type	Size
data_2018.rar	2/10/2019 8:20 PM	WinRAR archive	1,663,074 KB
part-r-00000-success.csv	2/10/2019 1:34 AM	Microsoft Excel C...	8,071 KB
part-r-00000-fail.csv	2/10/2019 1:17 AM	Microsoft Excel C...	162 KB
HD_success_2018	2/10/2019 1:34 AM	File folder	

We will be following the CRISP-DM Model for achieving the business and data mining goals for this project as illustrated below.



If the modelling results are not satisfactory, required data needs to be crunched out as per CRISP DM process.

Understanding & Exploring Hard Drives

Dataset: 01 – extracted

Code: 01 - EDA report v2.Rmd

Overview

We are currently using 2018 data from Back Blaze. These are SMART stats for each hard disk drive in operation and whether it fails or not. The data is mapped across days in a year.

Please note that R Code is interspersed within this section

Short Description of Dataset The dataset is divided into 2 distinct parts:

1. Descriptors

   These include Hard Drive information (e.g. model, serial number), test dates, Failure status etc. Failure Status is the ground truth label for each Hard Drive.
   Columns associated are date, serial_number, model, capacity_bytes, failure

2. SMART Stats

   These are the actual stats for each Hard Drive. Each stat has a raw and normalised version. According to the documentation, there will be missing values as hard drives do not necessarily report all SMART stats. Columns associated are smart_1_normalized, smart_1_raw, smart_2_normalized, smart_2_raw, smart_3_normalized, smart_3_raw, smart_4_normalized, smart_4_raw, smart_5_normalized, smart_5_raw, smart_7_normalized, smart_7_raw, smart_8_normalized, smart_8_raw, smart_9_normalized, smart_9_raw, smart_10_normalized, smart_10_raw, smart_11_normalized, smart_11_raw, smart_12_normalized, smart_12_raw, smart_13_normalized, smart_13_raw, smart_15_normalized, smart_15_raw, smart_22_normalized, smart_22_raw, smart_177_normalized, smart_177_raw, smart_179_normalized, smart_179_raw, smart_181_normalized, smart_181_raw, smart_182_normalized, smart_182_raw, smart_183_normalized, smart_183_raw, smart_184_normalized, smart_184_raw, smart_187_normalized, smart_187_raw, smart_188_normalized, smart_188_raw, smart_189_normalized, smart_189_raw, smart_190_normalized, smart_190_raw, smart_191_normalized, smart_191_raw, smart_192_normalized, smart_192_raw, smart_193_normalized, smart_193_raw, smart_194_normalized, smart_194_raw, smart_195_normalized, smart_195_raw, smart_196_normalized, smart_196_raw, smart_197_normalized, smart_197_raw, smart_198_normalized, smart_198_raw, smart_199_normalized, smart_199_raw, smart_200_normalized, smart_200_raw, smart_201_normalized, smart_201_raw, smart_220_normalized, smart_220_raw, smart_222_normalized, smart_222_raw, smart_223_normalized, smart_223_raw, smart_224_normalized, smart_224_raw, smart_225_normalized, smart_225_raw, smart_226_normalized, smart_226_raw, smart_235_normalized, smart_235_raw, smart_240_normalized, smart_240_raw, smart_241_normalized, smart_241_raw, smart_242_normalized, smart_242_raw, smart_250_normalized, smart_250_raw, smart_251_normalized, smart_251_raw, smart_252_normalized, smart_252_raw, smart_254_normalized, smart_254_raw, smart_255_normalized, smart_255_raw, LatestRec

Exploratory Data Analyses (EDAs) As the dataset is time-bound, we will be looking at how time affects the dataset in addition to the usual EDA techniques.

1. **Counts & Dimensions of Descriptors: To understand volume & "uniqueness" across time**

- Dimension of dataset
- Count of dates, hard drives, failed (1) vs OK (0) hard drives
- Proportion of failed vs OK hard drives

2. **Deep Dive into SMART Stats: To understand how much is missing, what's missing, and outliers**

- Data Summary
- Missing data & When they are missing
- SMART Stats with no variances
- Outliers

3. **Correlations between features: To see if there are similar features**
4. **Next Steps**

The rest of this section will highlight interesting findings and possbile approaches. Please note that R code is embedded within the document.

Counts and Dimensions of Descriptors

In this EDA, we are concerned about the shape, form, and size of the descriptors.

Dataset Dimensions

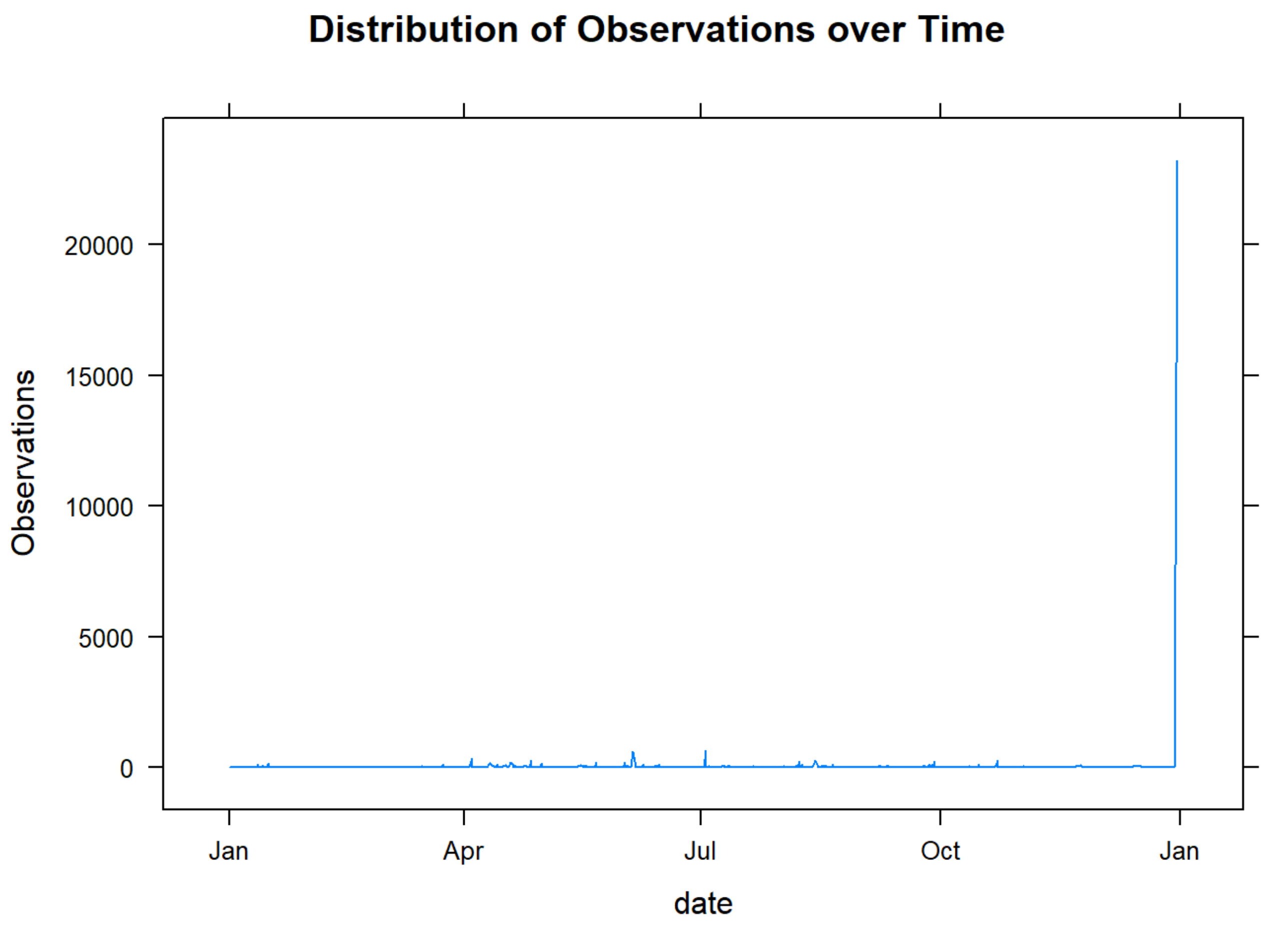
The dataset has these dimensions: 32056, 106

1. **Observations: 32056**
2. **Features: 106**

1. Unique Dates

There are **288** unique dates in 32056 observations. That means some observations might be "bunched" up in time. Let's have a look at how observations are distributed across time

```
temp <- hd %>%
  group_by(date, failure) %>%
  summarize(count=n())
  xyplot(count~date, temp, type="l", auto.key=TRUE, main="Distribution of Observations over Time", ylab="Observations
")
```



It’s quite apparent that there’s an **observation spike near the end of the year**.

2. How many Hard Drives do we have?

There are **32054** Hard Drives in 32056 observations. Almost every observation is matched to a unique hard drive

3. How many failed (1) vs OK (0) Hard Drives are there?

```
count(hd, failure)
```

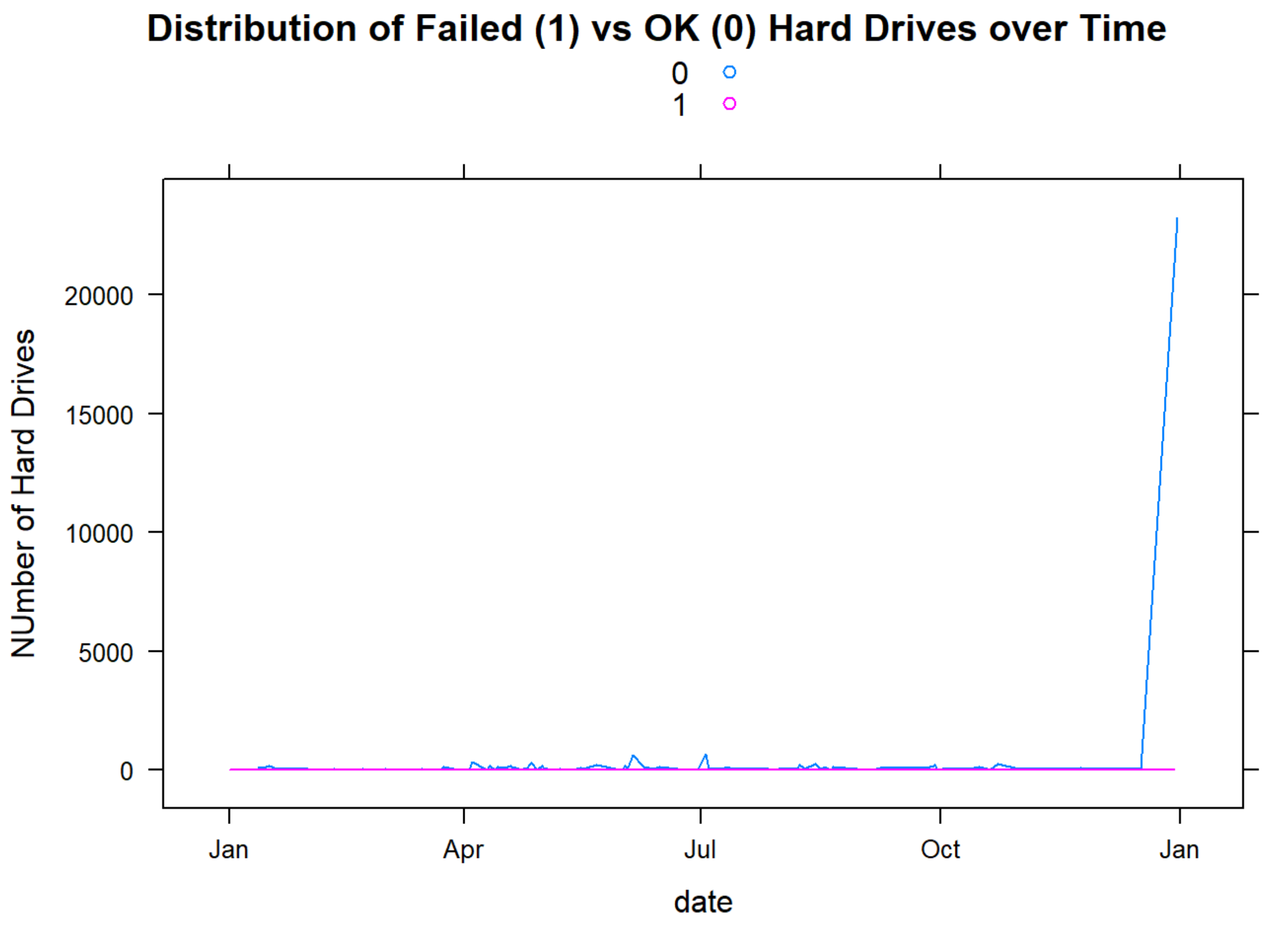
failure	n
<dbl>	**<int>**
0	31475

failure	n
<dbl>	<int>
1	581

As the table shows, we have a disporportionate breakdown of failed (1) vs OK (0) Hard Drives. There are only 1.8124532% failures. This means that we will have to **rebalance the dataset in some way.**

Following on this (dis)proportion, what is the **distribution of failed (1) vs OK (0) hard drives across time?**

```
xyplot(count~date, temp, groups=failure, type="l", auto.key=TRUE, main="Distribution of Failed (1) vs OK (0) Hard Drives over Time", ylab="NUmber of Hard Drives")
```



This is an expected result. We could probably get rid of some OK Hard Drives that are near the end of the year.

Counts & Dimensions Summary

OK (0) Hard Drives tend to “bunch” up towards the end of the year. This creates an imbalance in the dataset proportion between Failed (1) and OK (0) Hard Drives. The dataset will need to be rebalanced before running predictive models later.

Deep Dive into SMART Stats

Now that we know what's in our Descriptors, it's time to look at the actual predictors (in this case, SMART Stats of hard drives).

1. Summary of SMART Stats

```
summary(hd [6:ncol(hd)])
```

See

As this lengthy summary shows, some columns are empty (e.g. "smart_2_normalized", smart_2_raw"); some have just 1 value (e.g."LatestRec","smart_251_normalized"); and what is the difference between normalised and raw data columns?

For the rest of this section, we will look for the amount of missing data, columns with 0 variances, and outliers in columns that remain.

2. How Much Data is Missing?

```
missingHD <- colSums(is.na(hd))/nrow(hd)*100
```

This is best answered as such:

1. **How many columns have no missing data?**

   27 (about 25.4716981%)

2. **How many columns have < 50% missing data?**

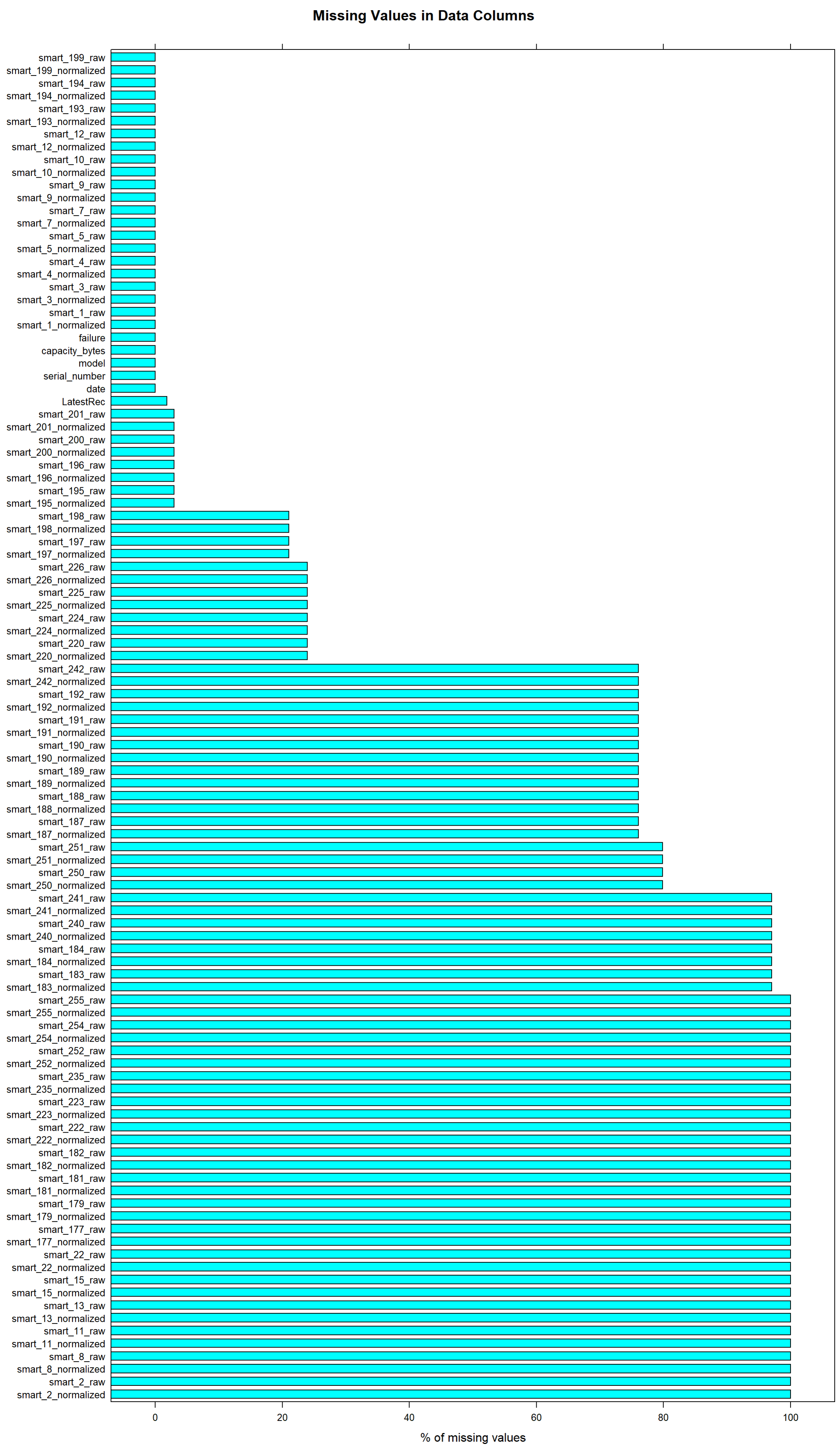
   48 (about 45.2830189%)

3. **How many columns have > 90% missing data?**

   40 (about 37.7358491%)

This leads us to the next question: **Which columns are Missing then?**

```
barchart(sort(missingHD,decreasing = TRUE), xlab = "% of missing values", main="Missing Values in Data Columns")
```


Missing Values in Data Columns
smart_199_raw
smart_199_normalized
smart_194_raw
smart_194_normalized
smart_193_raw
smart_193_normalized
smart_12_raw
smart_12_normalized
smart_10_raw
smart_10_normalized
smart_9_raw
smart_9_normalized
smart_7_raw
smart_7_normalized
smart_5_raw
smart_5_normalized
smart_4_raw
smart_4_normalized
smart_3_raw
smart_3_normalized
smart_1_raw
smart_1_normalized
failure
capacity_bytes
model
serial_number
date
LatestRec
smart_201_raw
smart_201_normalized
smart_200_raw
smart_200_normalized
smart_196_raw
smart_196_normalized
smart_195_raw
smart_195_normalized
smart_198_raw
smart_198_normalized
smart_197_raw
smart_197_normalized
smart_226_raw
smart_226_normalized
smart_225_raw
smart_225_normalized
smart_224_raw
smart_224_normalized
smart_220_raw
smart_220_normalized
smart_242_raw
smart_242_normalized
smart_192_raw
smart_192_normalized
smart_191_raw
smart_191_normalized
smart_190_raw
smart_190_normalized
smart_189_raw
smart_189_normalized
smart_188_raw
smart_188_normalized
smart_187_raw
smart_187_normalized
smart_251_raw
smart_251_normalized
smart_250_raw
smart_250_normalized
smart_241_raw
smart_241_normalized
smart_240_raw
smart_240_normalized
smart_184_raw
smart_184_normalized
smart_183_raw
smart_183_normalized
smart_255_raw
smart_255_normalized
smart_254_raw
smart_254_normalized
smart_252_raw
smart_252_normalized
smart_235_raw
smart_235_normalized
smart_223_raw
smart_223_normalized
smart_222_raw
smart_222_normalized
smart_182_raw
smart_182_normalized
smart_181_raw
smart_181_normalized
smart_179_raw
smart_179_normalized
smart_177_raw
smart_177_normalized
smart_22_raw
smart_22_normalized
smart_15_raw
smart_15_normalized
smart_13_raw
smart_13_normalized
smart_11_raw
smart_11_normalized
smart_8_raw
smart_8_normalized
smart_2_raw
smart_2_normalized
0
20
40
60
80
100
% of missing values

We can remove columns with > 90% missing values without affecting the dataset too much. We should get rid of these columns: smart_2_normalized, smart_2_raw, smart_8_normalized, smart_8_raw, smart_11_normalized, smart_11_raw, smart_13_normalized, smart_13_raw, smart_15_normalized, smart_15_raw, smart_22_normalized, smart_22_raw, smart_177_normalized, smart_177_raw, smart_179_normalized, smart_179_raw, smart_181_normalized, smart_181_raw, smart_182_normalized, smart_182_raw, smart_183_normalized, smart_183_raw, smart_184_normalized, smart_184_raw, smart_222_normalized, smart_222_raw, smart_223_normalized, smart_223_raw, smart_235_normalized, smart_235_raw, smart_240_normalized, smart_240_raw, smart_241_normalized, smart_241_raw, smart_252_normalized, smart_252_raw, smart_254_normalized, smart_254_raw, smart_255_normalized, smart_255_raw

Apart from missing values, what columns contain only 1 value? Columns with constant values shouldn't have an effect on the Hard Drive's failure/OK rate. We can use variance as a measure to identify these columns.

3. Which Columns have only 1 value? Measured by Variance of 0

```
varHD <- apply(hd, 2, var,na.rm=TRUE)

noVar <- varHD[!is.na(varHD)]
```

There are 11 columns with constant values.

Remove these 0 variance columns: capacity_bytes, smart_3_raw, smart_10_normalized, smart_10_raw, smart_226_normalized, smart_240_normalized, smart_241_normalized, smart_242_normalized, smart_250_normalized, smart_251_normalized, LatestRec

Intermission

We now know the columns that have >90% missing values and 0 variance. Before we conduct further EDAs, it would be useful to remove these columns from our dataset. In addition, we are also taking out normalized variables.

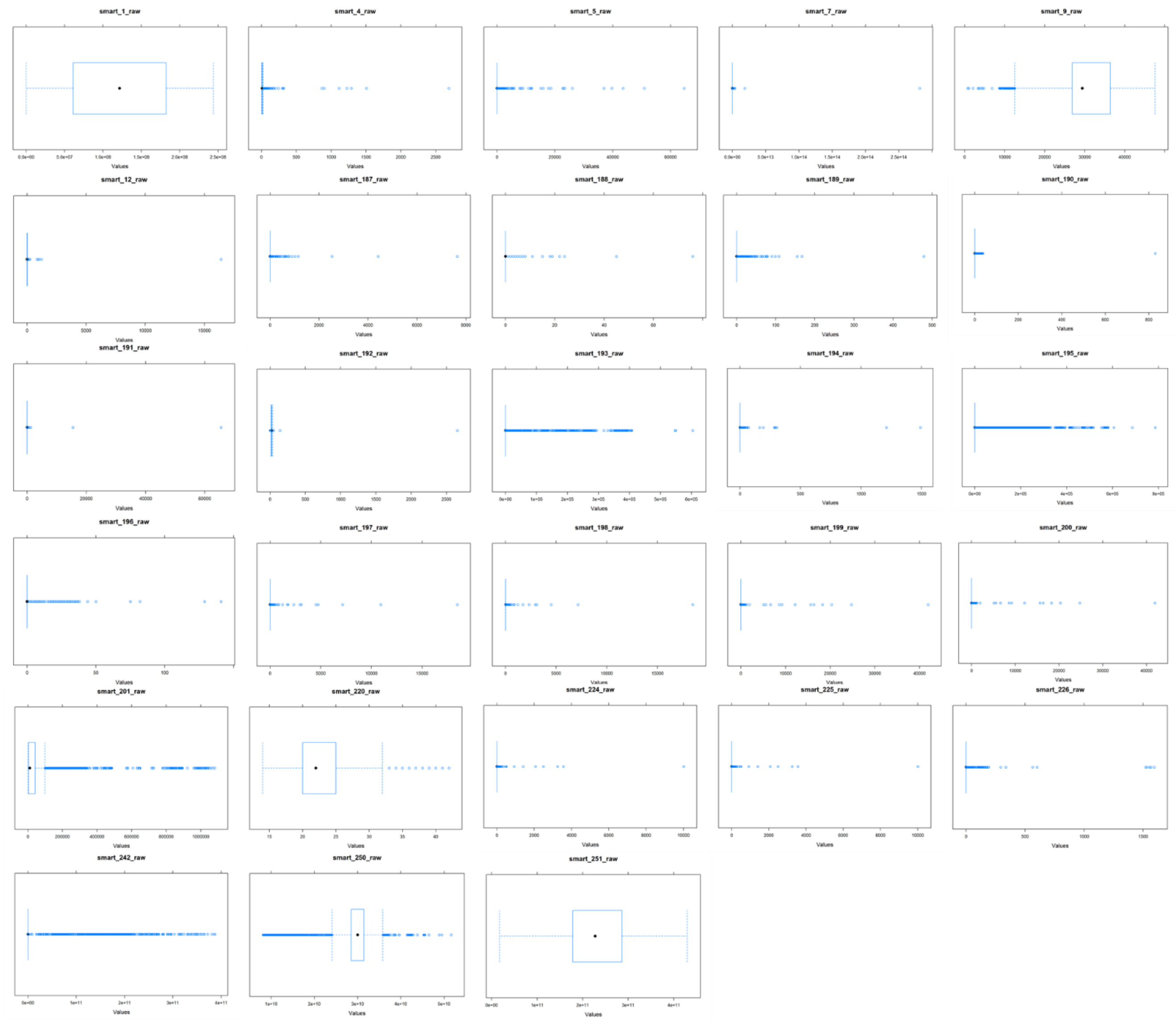
```
# removing columns with > 90% missing values, 0 variance, and -Normalized variables


t <- hd[,missingHD <90]

t <- t[, -which(names(t) %in% names(noVar[noVar==0]))]

t <- t[, -grep("normalized$", colnames(t))]
```

After removal, we are left with 32 columns.

These are the variables that make the cut: date, serial_number, model, failure, smart_1_raw, smart_4_raw, smart_5_raw, smart_7_raw, smart_9_raw, smart_12_raw, smart_187_raw, smart_188_raw, smart_189_raw, smart_190_raw, smart_191_raw, smart_192_raw, smart_193_raw, smart_194_raw, smart_195_raw, smart_196_raw, smart_197_raw, smart_198_raw, smart_199_raw, smart_200_raw, smart_201_raw, smart_220_raw, smart_224_raw, smart_225_raw, smart_226_raw, smart_242_raw, smart_250_raw, smart_251_raw

4. How Many Outliers Are There In Each Column?

```
for (i in 5:32) {
  d1 <- t[,i]
  d1 <- d1[!is.na(d1)]
  n <- colnames(t)[i]
  show(bwplot(d1, xlab="Values", ylab="", main=n))
}
```



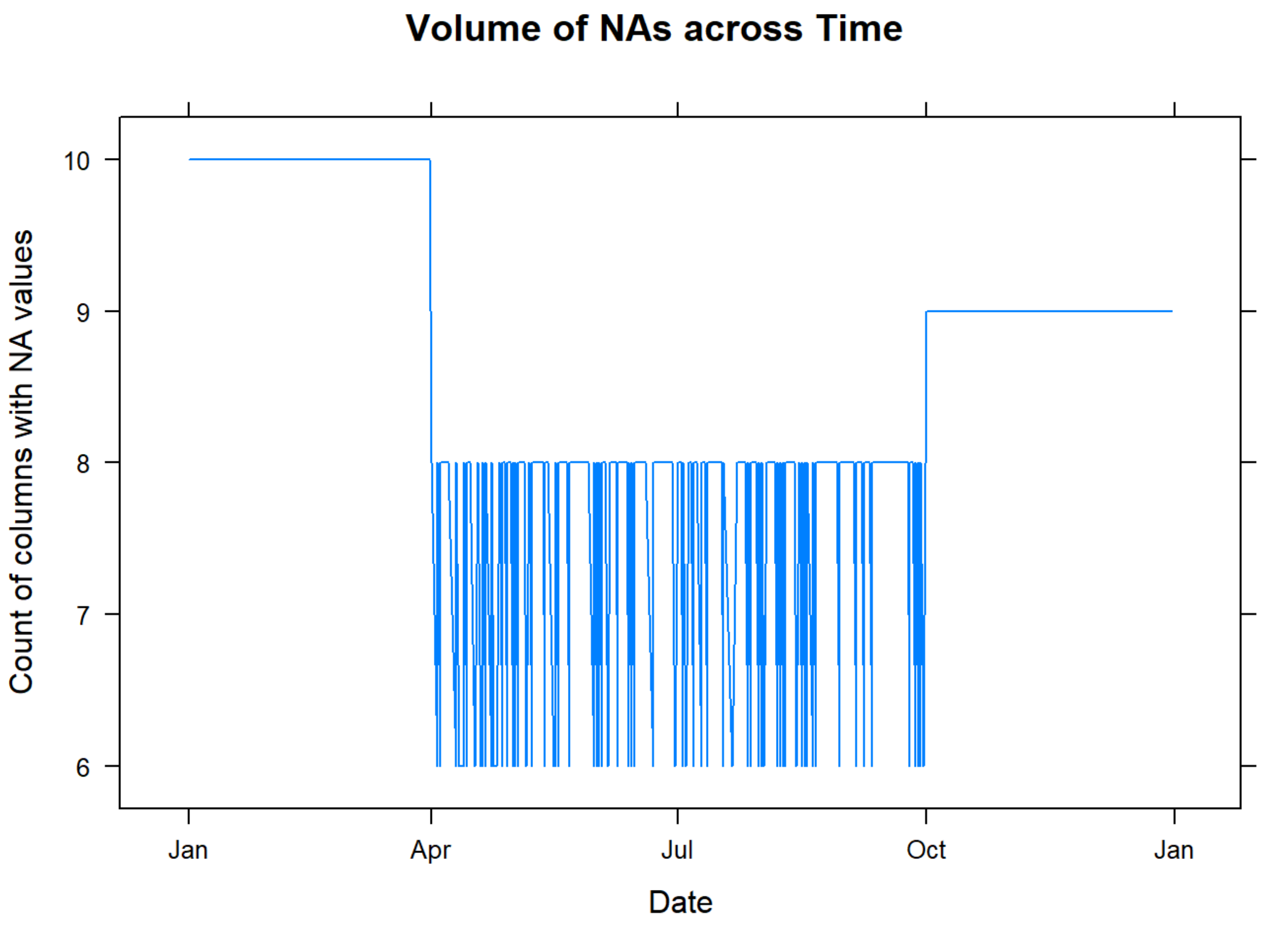
We notice that a variety of attributes are skewed left or right. However we suspect that the skew is due to them having some kind of significance in predicting the output (1 or 0) as SMART Stats are only reported on an ad-hoc basis.

We will investigate further when we build the models.

5. When would Data Go Missing?

This dataset contains time data and, in addition, Hard Drives when queried do not necessarily reveal certain SMART Stats. We'd like to see if a patterns appear when we look at the volume of missing values against time.

```
t$naVal <- rowSums(is.na(t))

xyplot(naVal~date, t, type="l", main="Volume of NAs across Time", auto.key = TRUE, xlab="Date", ylab="Count of columns with NA values")
```



The chart shows that more columns are missing between Jan to Apr. Fewer during Apr to Oct (Q2 & Q3).

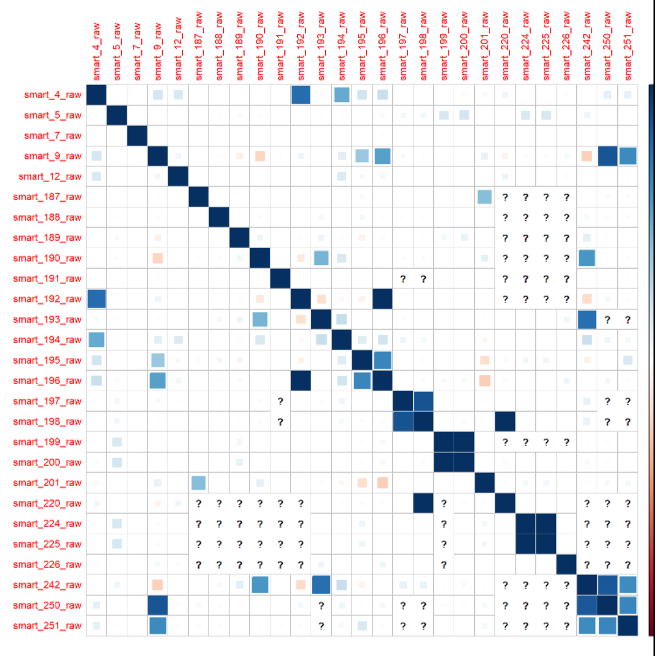
We would consider **using just the Apr to Oct duration for our prediction models, and filling in the missing values using a replacement technique**.

3. Correlations between features

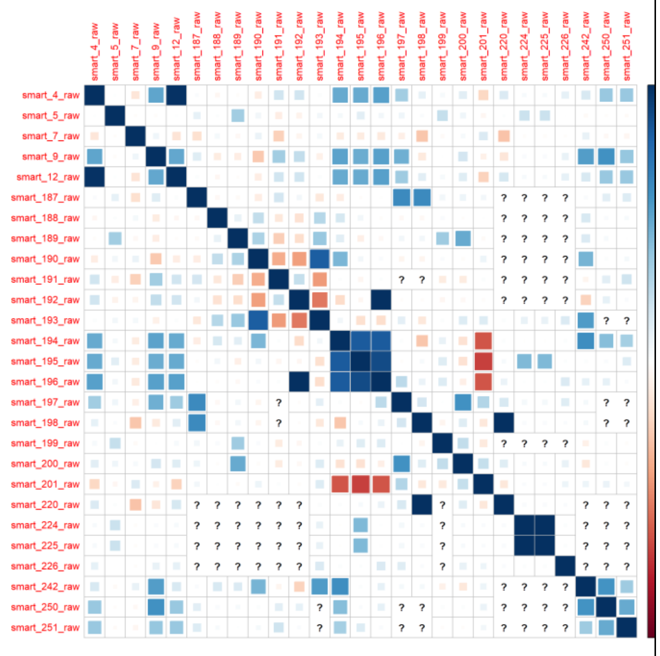
Lastly we investigate the correlations between features. Ideally, the features shouldn't show high correlations (>0.8) between each other. If so, we will have to remove either one.

This correlation was conducted using "pairwise.complete.obs"; otherwise the NAs would simply render the correlations moot.

```
a <- cor(t[6:32], use = "pairwise.complete.obs", method="pearson")
## Warning in cor(t[6:32], use = "pairwise.complete.obs"): the standard
## deviation is zero
corrplot(a, method="square")
a <- cor(t[6:32], use = "pairwise.complete.obs", method="spearman")
corrplot(a, method="square")
```



Pearson



Spearman

The plot shows high correlations between:

1. smart_4_raw | ~~smart_192_raw~~
2. smart_4_raw | ~~smart_194_raw~~
3. smart_9_raw | ~~smart_196_raw~~

4. smart_9_raw | ~~smart_250_raw~~
5. smart_9_raw | ~~smart_251_raw~~
6. smart_192_raw | ~~smart_196_raw~~
7. smart_193_raw | smart_242_raw
8. smart_195_raw | ~~smart_196_raw~~
9. smart_198_raw | ~~smart_197_raw~~
10. smart_199_raw | ~~smart_200_raw~~
11. smart_198_raw | ~~smart_220_raw~~
12. smart_224_raw | ~~smart_225_raw~~
13. smart_242_raw | ~~smart_250_raw~~

Crossed out variables (e.g. ~~smart_225_raw)~~ should be dropped from the dataset. In addition, we will keep the following variables: smart_4_raw; smart_9_raw; smart_193_raw; smart_195_raw; smart_198_raw; smart_199_raw; smart_198_raw; smart_224_raw; smart_242_raw.

Next Steps

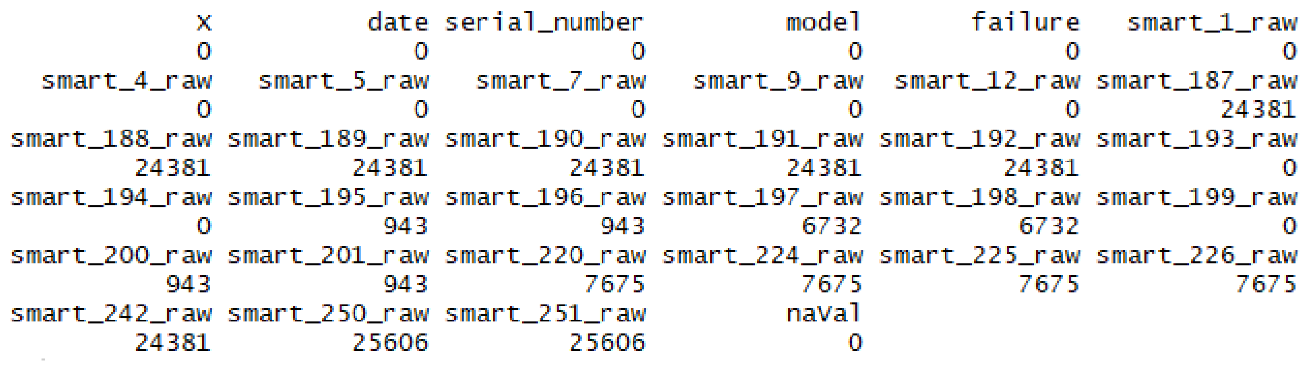
1. Remove columns with > 90% missing values, normalised columns, and columns with 0 variances
2. Remove columns based on high correlation findings
3. Select sub-section of observations that have the most data available
4. Fill in in missing values using averages src: Towards Data Science)

Preparing the Data

This is one of the most important steps in the process of preparing the data as the accuracy and the stability of the model purely depends on the data that we used to train it. In this step the focus will mainly be on to impute the null values in the feature variable set. Also the other important process will be to normalize the feature variables as we have a few features such as smart_1_raw which has values in the ranges of few millions whereas smart_4_raw has in the range of 0 to 100. Hence there is a huge chance that smart_1_raw can dominate the other features during training and making it biased Hence to over come this we need to do normalization.

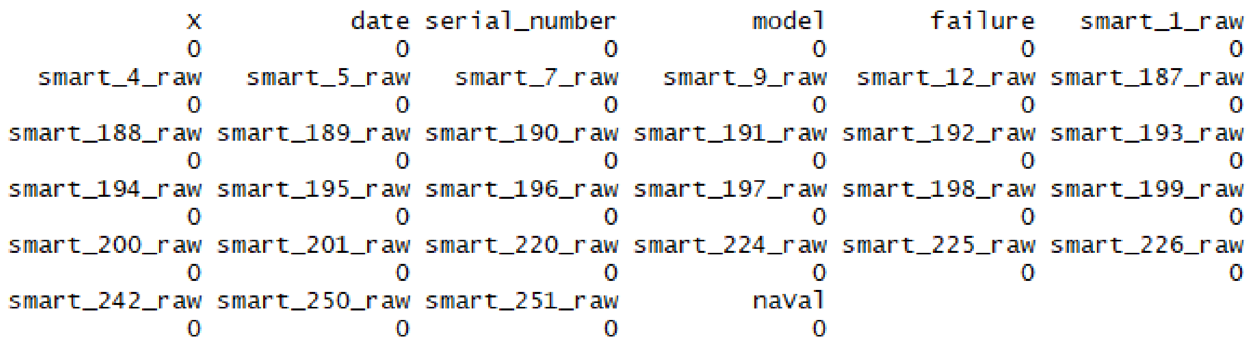
Below are the steps involved to impute null values in the input features.
Find the NAs in each input feature and replace the same with means of the same feature.

Before the logic was applied to remove the NAs.



```
            X          date serial_number         model       failure   smart_1_raw 
            0             0             0             0             0             0 
  smart_4_raw   smart_5_raw   smart_7_raw   smart_9_raw  smart_12_raw smart_187_raw 
            0             0             0             0             0         24381 
smart_188_raw smart_189_raw smart_190_raw smart_191_raw smart_192_raw smart_193_raw 
        24381         24381         24381         24381         24381             0 
smart_194_raw smart_195_raw smart_196_raw smart_197_raw smart_198_raw smart_199_raw 
            0           943           943          6732          6732             0 
smart_200_raw smart_201_raw smart_220_raw smart_224_raw smart_225_raw smart_226_raw 
          943           943          7675          7675          7675          7675 
smart_242_raw smart_250_raw smart_251_raw         naVal 
        24381         25606         25606             0 
```

After the logic was applied to remove the NAs.



```
            X          date serial_number         model       failure   smart_1_raw 
            0             0             0             0             0             0 
  smart_4_raw   smart_5_raw   smart_7_raw   smart_9_raw  smart_12_raw smart_187_raw 
            0             0             0             0             0             0 
smart_188_raw smart_189_raw smart_190_raw smart_191_raw smart_192_raw smart_193_raw 
            0             0             0             0             0             0 
smart_194_raw smart_195_raw smart_196_raw smart_197_raw smart_198_raw smart_199_raw 
            0             0             0             0             0             0 
smart_200_raw smart_201_raw smart_220_raw smart_224_raw smart_225_raw smart_226_raw 
            0             0             0             0             0             0 
smart_242_raw smart_250_raw smart_251_raw         naval 
            0             0             0             0 
```

Step 2:

Then once the null values are imputed we need to normalize the input features. This is done using the min max normalization.

Step 3:

Then finally we need to balance the data set for all the classes. In the dataset the cases of failures is less than 10 percent and if the model set is built without any modifications then the model will be biased towards the Success class. Hence we followed the mixed technique of both down sampling and up sampling. For the success case we down sampled by picking 2500 records and for success case we up sampled from the original data set size of 581 records to 2500.

Building Models to Predict Good Hard Drives

Dataset: 03 – normalized.xlsx

Code: 03 – Modelling.R

Modelling Techniques

The following modeling techniques were used to predict the hard disk failure using the extracted features for our dataset

1. Decision Tree
2. Random Forest
3. Gradient Boosting (Adaptive)
4. Support Vector Machine
5. Logistic Regression
6. Multi-Layer Neural Network

Test Design

While designing the tests that need to be carried out on the model that was built, we decided to carry empirical validation. In this process the dataset was split using the simple split technique. We use multiple split dimensions on each of our different modelling techniques to find the accuracy in every scenario
The split sizes are:

- 70% Training, 15% Validation, 15% Testing
- 60% Training, 20% Validation, 20% Testing

Build Model

The dataset was split into train and test sets based on the test design. For each trial, the records were sampled sampling and trained on each of the selected model. k-fold cross validation was the evaluation criteria before the final test prediction. The same approach was also considered for the final test sets. The sample runs can be traced to the Appendix section.

Assess Model

Depending on the type of output the model creates we assess them differently:
Class output: Models which outputs either of the binary as result i.e.: SVM
Probability output: Provides its confidence for each class i.e.: Random Forest
As assessing a model is an integral part of model building these metrics were taken into consideration:

Confusion Matrix:
It is an N*N matrix, where N denotes no of classes predicted.
From the confusion matrix we derive these sub-metrics:

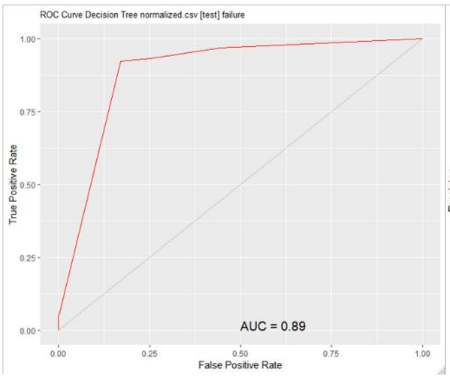
1. Accuracy - % of correct predictions
2. Precision - % of positive cases which were correct
3. Sensitivity/Recall - a portion of actual positive cases which were predicted correctly
4. Specificity - a portion of actual negative cases which were predicted correctly.
5. ROC and Area under the Curve - A plot of TRP vs FPR to showcase the strength of the model.

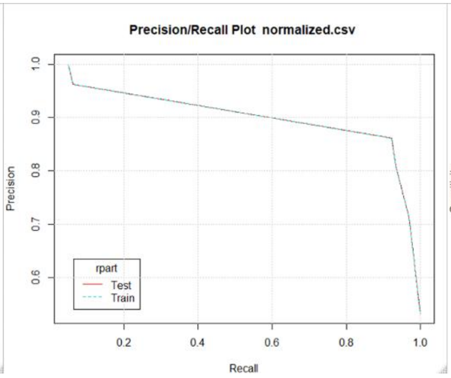
Did It Work (Or Evaluating our Models)?

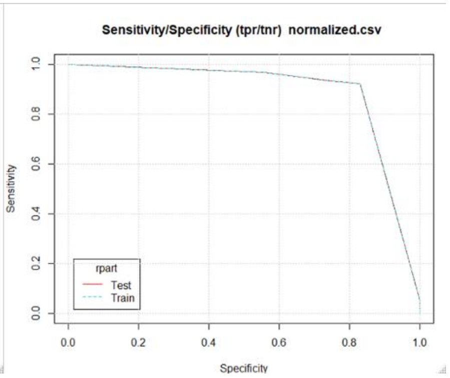
Of all the models that we've tried, the best performing model was a Random Forest Model with 1.8% error rate based on the confusion matrix.

Decision Tree

Overall error: 12%, Averaged class error: 12.3%



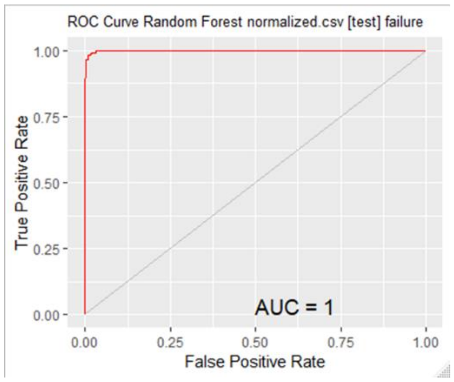


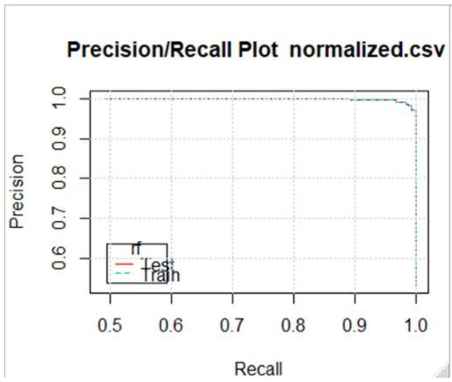


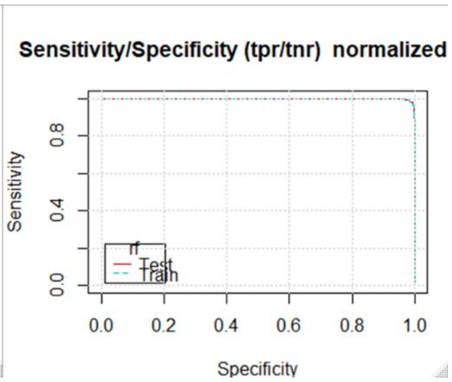
Column1	Predicted		
Actual	0	1	Error
0	304	62	16.9
1	32	385	7.7

Random Forest

Overall error: 1.8%, Averaged class error: 1.85% (best performing model)



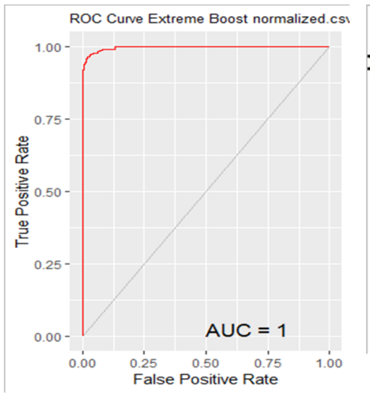


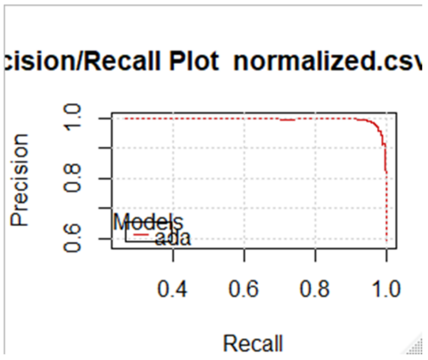


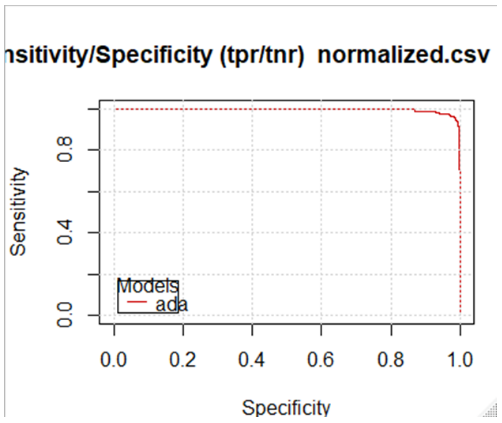
Column1	Predicted		
Actual	**0**	**1**	**Error**
0	**355**	**11**	**3**
1	**3**	**414**	**0.7**

Adaptive Boost

Overall error: 3%, Averaged class error: 2.95%



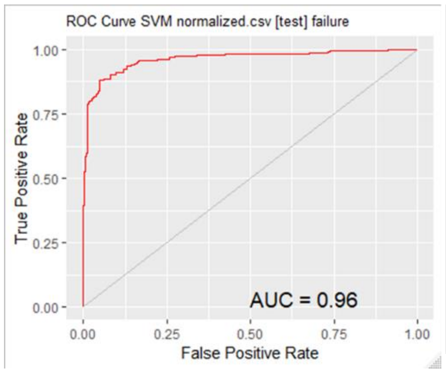


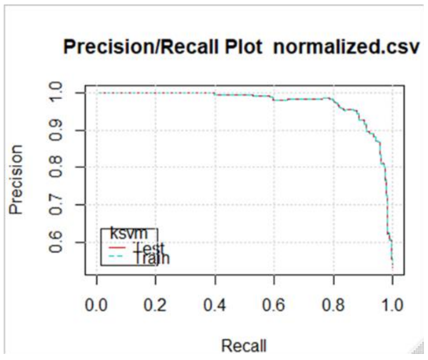


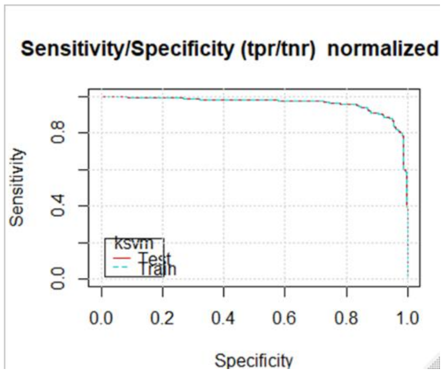
Column1	Predicted		
Actual	0	1	Error
0	317	9	2.8
1	14	442	3.1

SVM

Overall error: 9.2%, Averaged class error: 9.15%



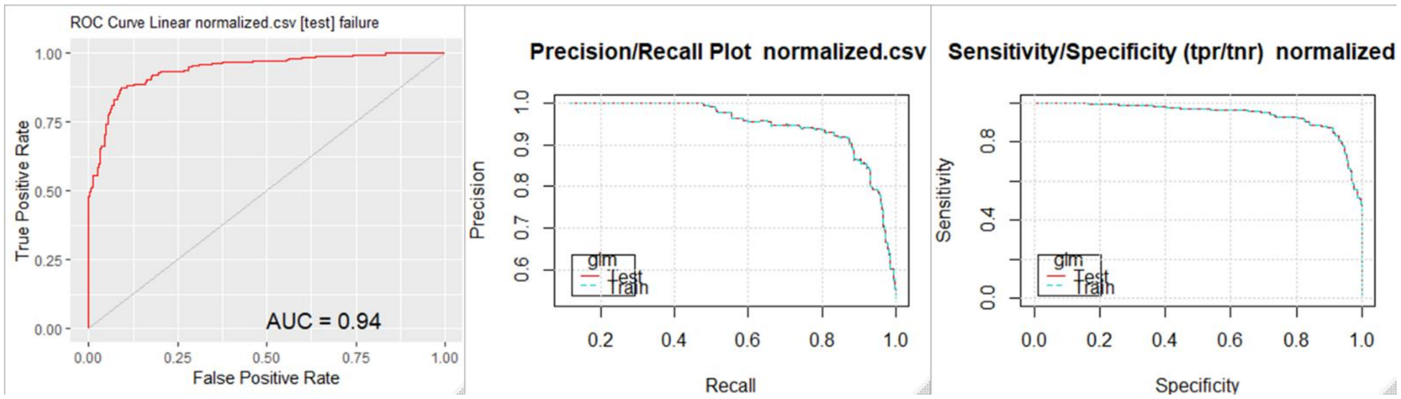




	Predicted		
Actual	0	1	Error
0	336	30	8.2
1	42	375	10.1

Logistic Regression

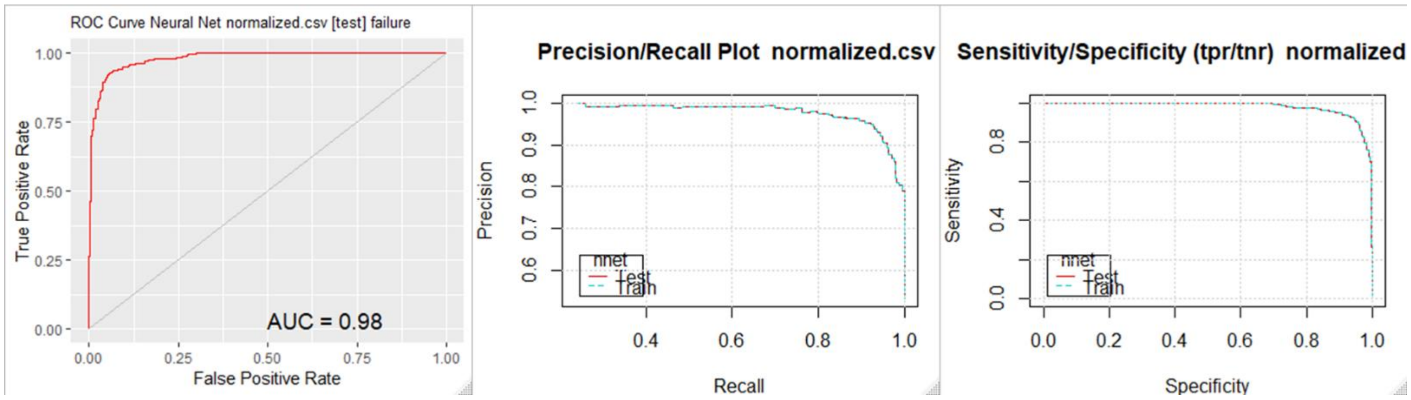
Overall error: 12.8%, Averaged class error: 12.85%



	Predicted		
Actual	0	1	Error
0	314	52	14.2
1	48	369	11.5

Multi-Layer Feed Forward Neural Network

Overall error: 6.8%, Averaged class error: 6.7%



	Predicted		
Actual	0	1	Error
0	344	22	6.0
1	31	386	7.4

EVALUATE RESULTS

The models built using the last day reading was taken as the baseline. Then feature extraction was applied and feature engineering on the temporal aspects of the variables generated new features. These features were used to train different models and evaluate them using the validation and test sets.

Future Enhancements & Conclusion

In this report, we have analyzed the Back blaze hard drive (Seagate - ST4000DM000) failure and used several prediction models for classification. We evaluated the prediction performance among Decision Tree, Support Vector Machine (SVM) – Logistic Regression, Random Forest, Multi-Layer Neural Networks, and Adaptive Gradient Boosting (AdaBoost) models. The next most important task is to introduce these prediction models into the real world. Furthermore, we are now predicting the hard drive device failure on the day of its failure. If we could predict the device failure in advance, then suitable backup action can be taken to avoid the data loss.

Appendix

Feature Dictionary

Feature	Description
smart_1_raw	Raw Read Error Rate
smart_4_raw	Start Stop Count
smart_5_raw	Reallocated Sector Ct
smart_7_raw	Seek Error Rate
smart_9_raw	Power On Hours
smart_12_raw	Power Cycle Count
smart_187_raw	Reported Uncorrect
smart_188_raw	Command Timeout
smart_189_raw	High Fly Writes
smart_190_raw	Airflow Temperature Cel
smart_191_raw	G-Sense Error Rate
smart_192_raw	Power-Off Retract Count
smart_193_raw	Load Cycle Count
smart_194_raw	Temperature Celsius
smart_195_raw	Hardware ECC Recovered
smart_196_raw	Reallocated Event Count
smart_197_raw	Current Pending Sector
smart_198_raw	Offline Uncorrectable
smart_199_raw	UDMA CRC Error Count
smart_200_raw	Multi Zone Error Rate
smart_201_raw	Soft Read Error Rate
smart_220_raw	Disk Shift
smart_224_raw	Load Friction
smart_225_raw	Load/Unload Cycle Count
smart_226_raw	Load 'In'-time
smart_242_raw	Total LBAs Read
smart_250_raw	Read Error Retry Rate
smart_251_raw	Minimum Spares Remaining

Source: Wikipedia