

|  |
| --- |
| On Hard Drives & Failure  KE 5107 CA 1 Report |
|  |
| Authored by:  Raj Baharat  Edwin Tam  Vignesh |

Contents

1. What in Back blazes is this about?
2. Data collection and Pre-processing
3. Understanding & Exploring Hard Drives
4. Preparing the Data
5. Models for prediction
6. Evaluations

# 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](https://www.backblaze.com/blog-smart-stats-2014-8.html)) 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?

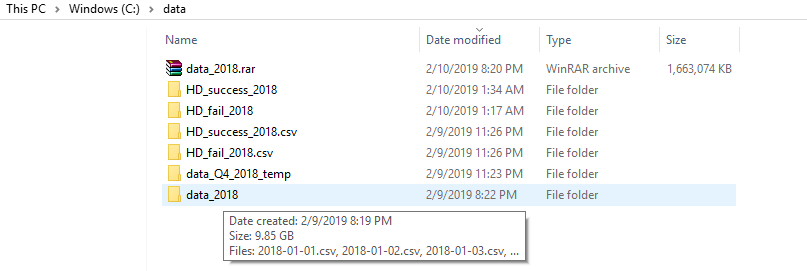
Backblaze logs Hard Drive Test Datasets on their [webpage](https://www.backblaze.com/b2/hard-drive-test-data.html). Each zipped file contains daily snapshots for each of their 40,000 Hard Drives. This also includes hard drive statuses (Failure or OK).

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.

# Data Collection and Preprocessing

|  |
| --- |
| Data Collection Based on the statistics published by [Back blaze](https://www.backblaze.com/blog/hard-drive-stats-for-2018/) 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.   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 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](https://spark.apache.org/docs/latest/sparkr.html) and [SparklyR](https://spark.rstudio.com/) , 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")

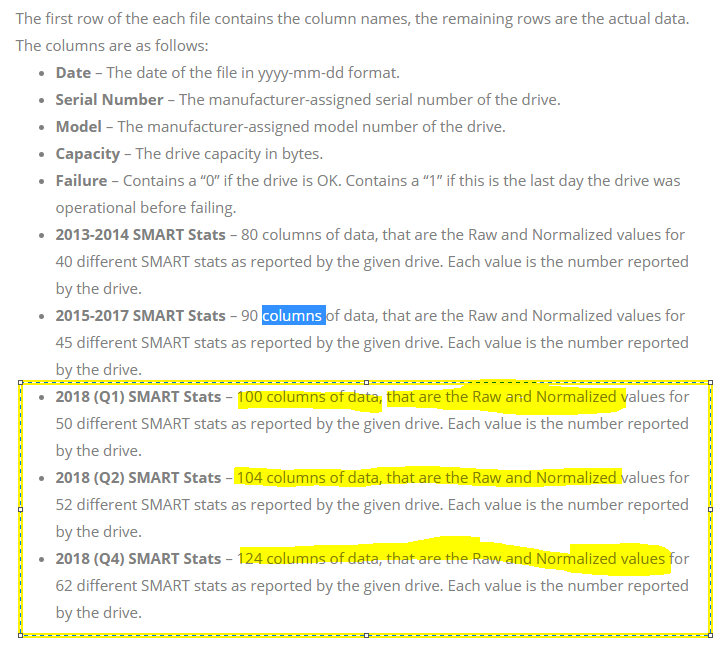
*Here* smart\_9\_raw is the life of hard drive. Hard drives whose life is more than 365 days (365\*24 = 8760) and which haven’t failed are chosen for our analysis.

val failures = spark.sql("SELECT \* FROM peopleTemp where failure=1 and model='ST4000DM000'")

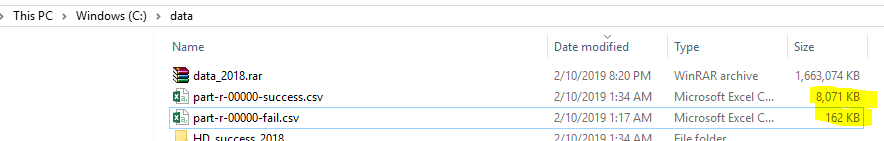
failures.coalesce(1).write.format("com.databricks.spark.csv")

.option("header","true").save("file:///C:/data/HD\_fail\_2018")

Back blaze team has constantly [evolved](https://www.backblaze.com/b2/hard-drive-test-data.html) 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.



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



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

## **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

1. 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

1. 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

1. Correlations between features: To see if there are similar features
2. 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**

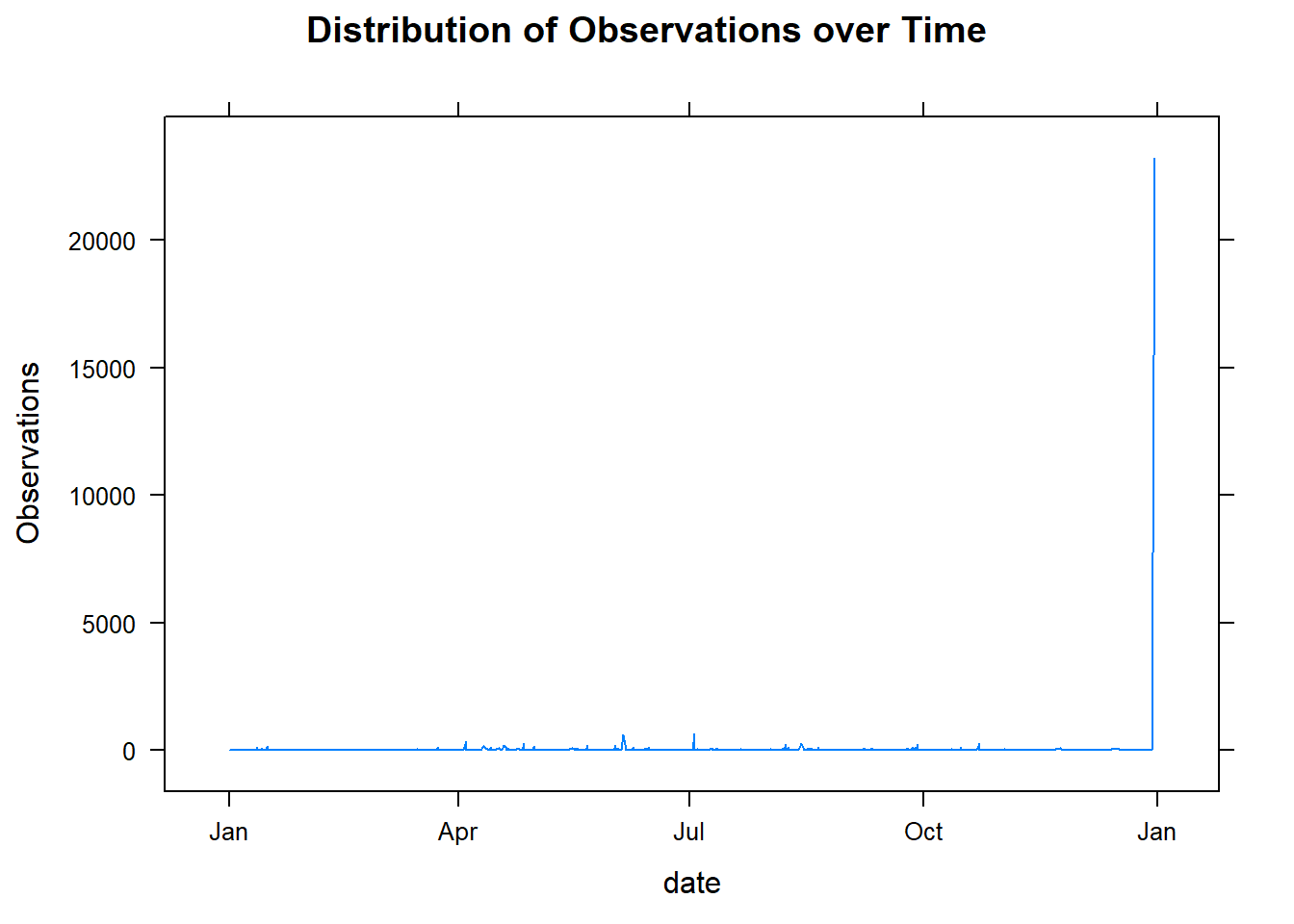
There are **288** unique dates in 32056 observations. That means some observations might be “bunched” up in time. Let’s have a look at how observations are distributed across time

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)

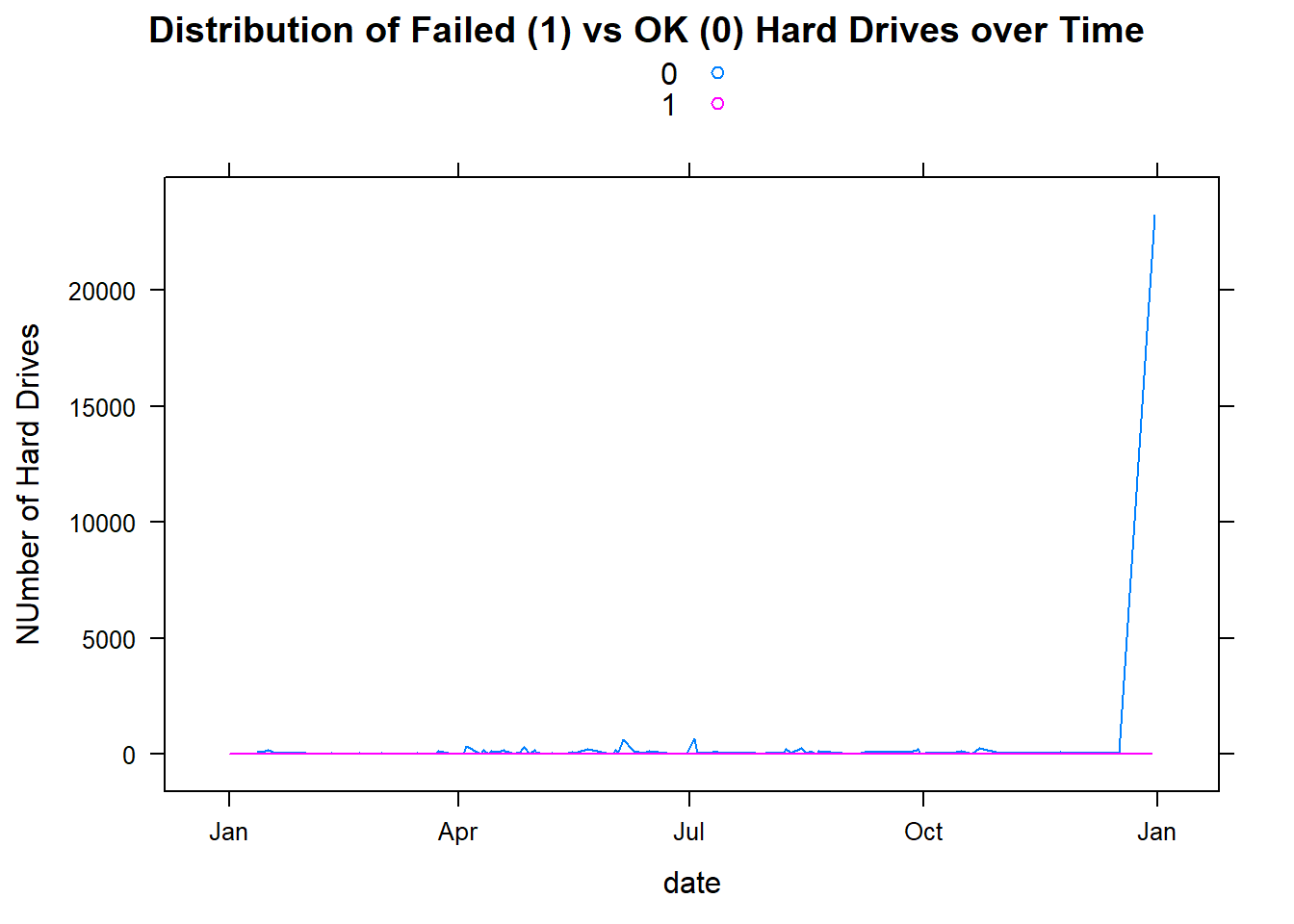
|  |
| --- |
|  |

| failure  <dbl> | n  <int> |
| --- | --- |
| 0 | 31475 |
| 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%)

1. How many columns have < 50% missing data?

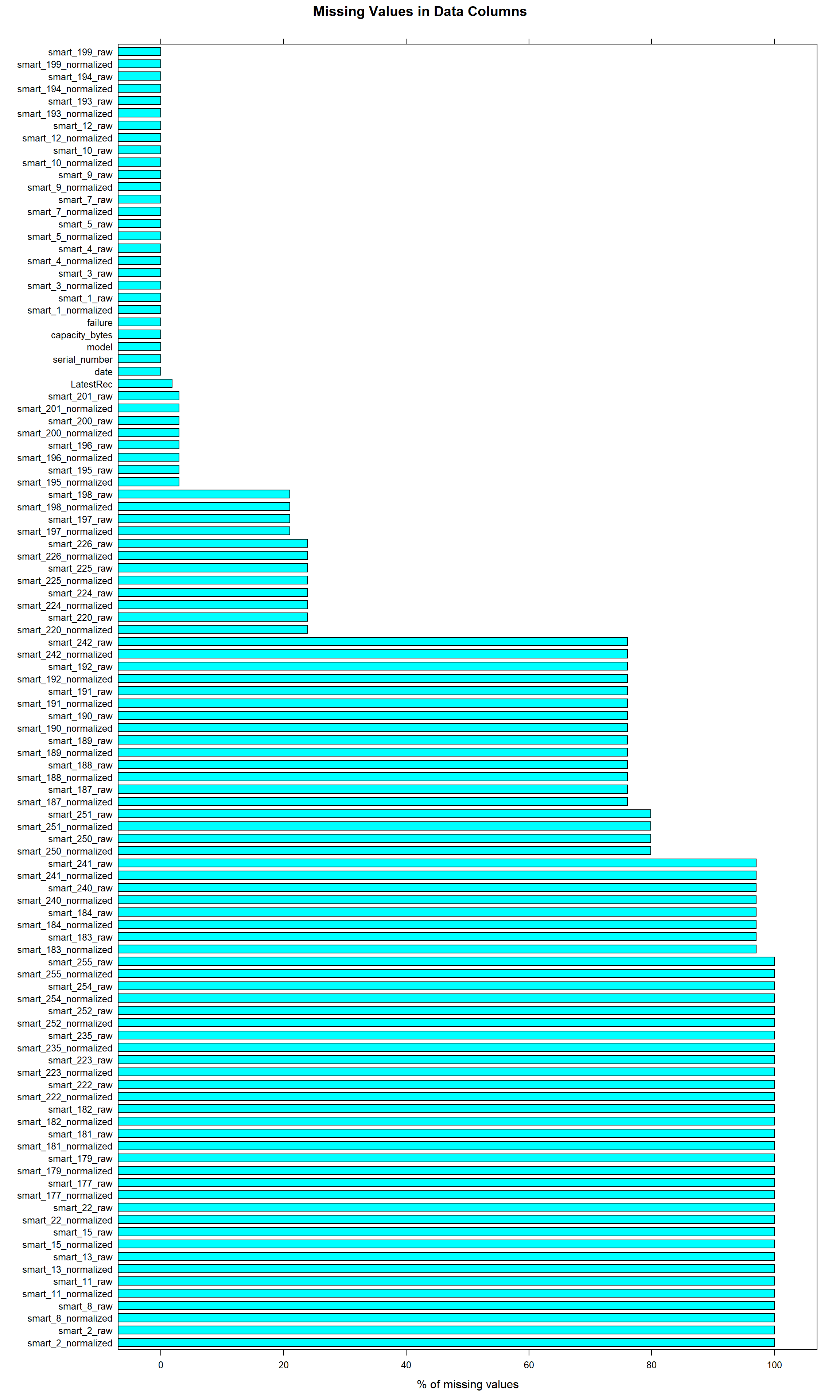
48 (about 45.2830189%)

1. 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")

****

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) {

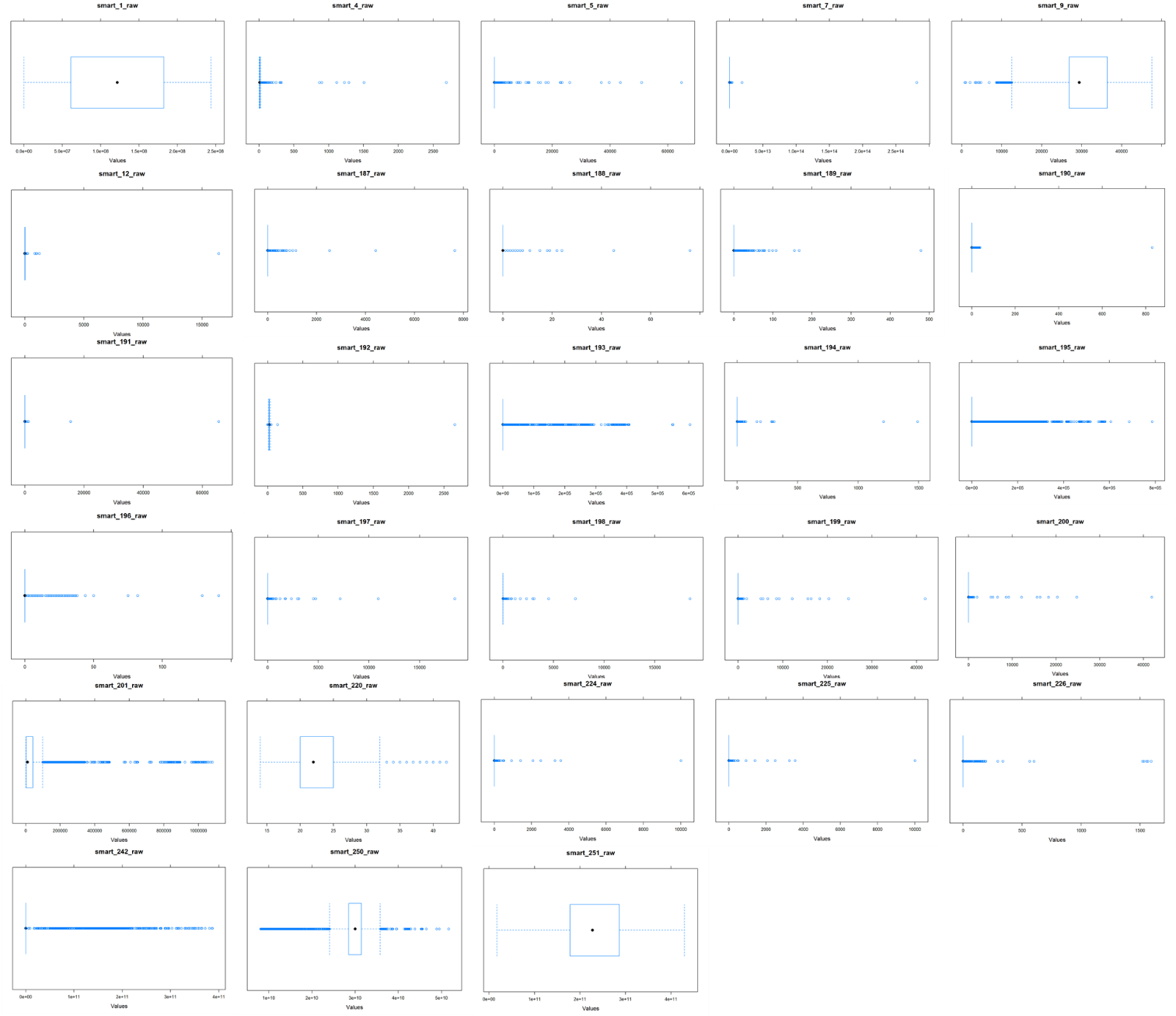
d1 <- t[,i]

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.

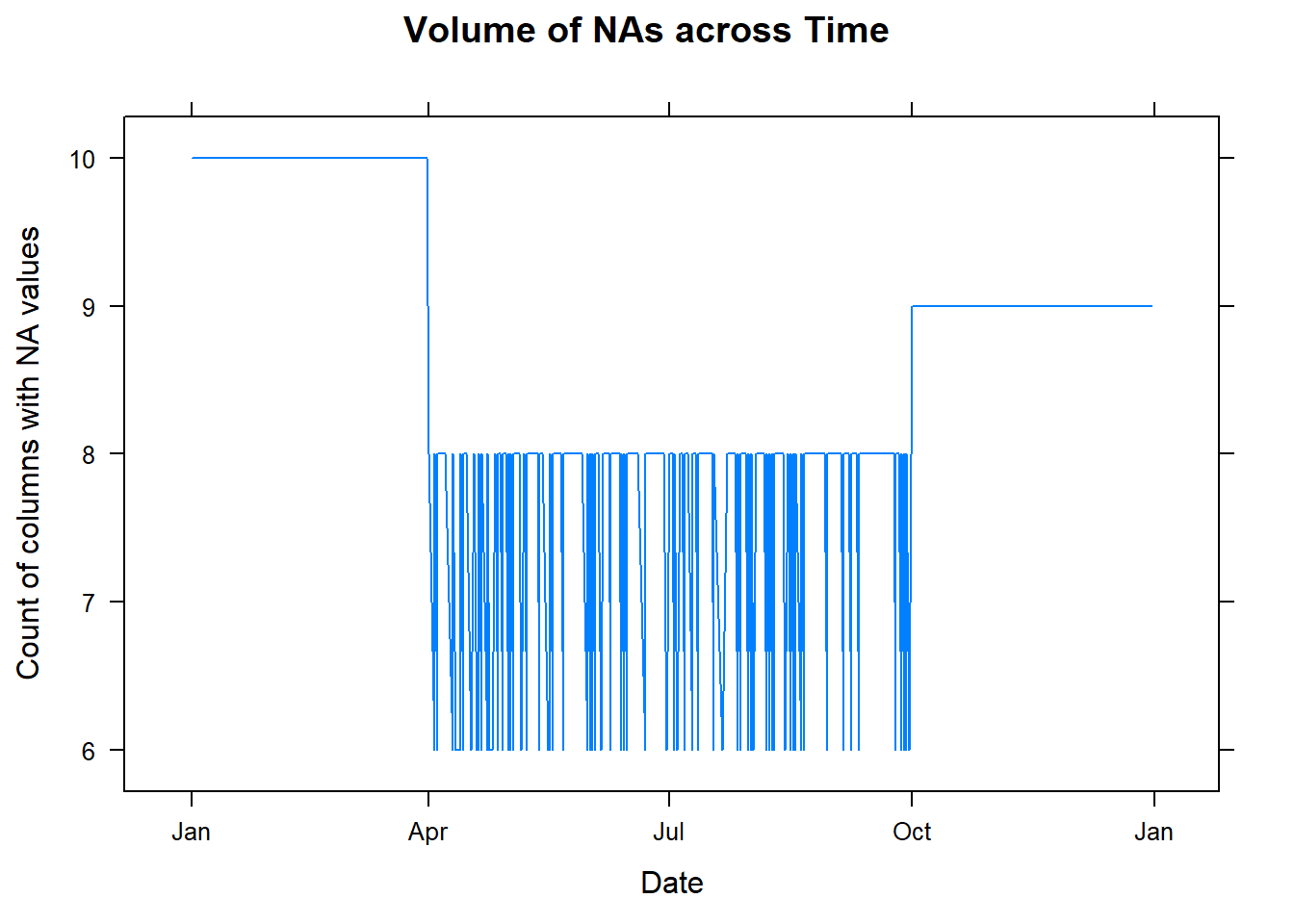
We will investigate further when we build the models.

**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")

|  |  |
| --- | --- |
| **C:\Users\skybe\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\AC\INetCache\Content.MSO\36554359.tmp**  Pearson | C:\Users\skybe\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\AC\INetCache\Content.MSO\8ABA0937.tmp  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

We should 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](https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4))

# Preparing the Data

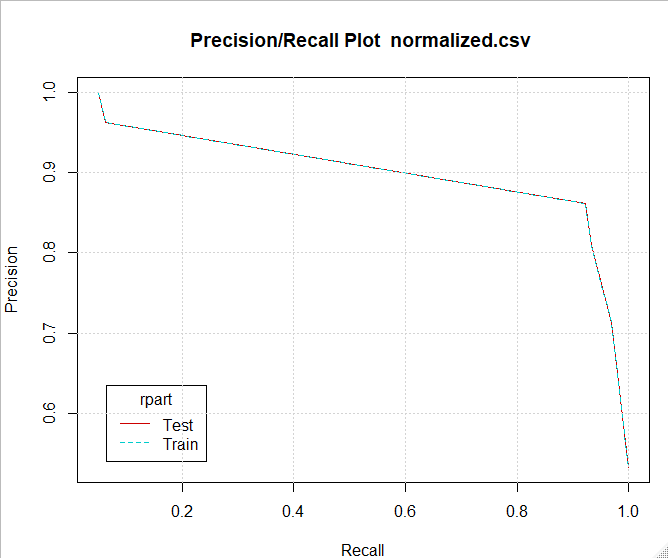
|  |
| --- |
| Subtitle Text Here To get started right away, just tap any placeholder text (such as this) and start typing to replace it with your own.  Want to insert a picture from your files or add a shape, text box, or table? You got it! On the Insert tab of the ribbon, just tap the option you need. |
| *“Find even more easy-to-use tools on the Insert tab, such as to add a hyperlink or insert a comment”* |
| To get started right away, just tap any placeholder text (such as this) and start typing to replace it with your own.  Want to insert a picture from your files or add a shape, text box, or table? You got it! On the Insert tab of the ribbon, just tap the option you need. |

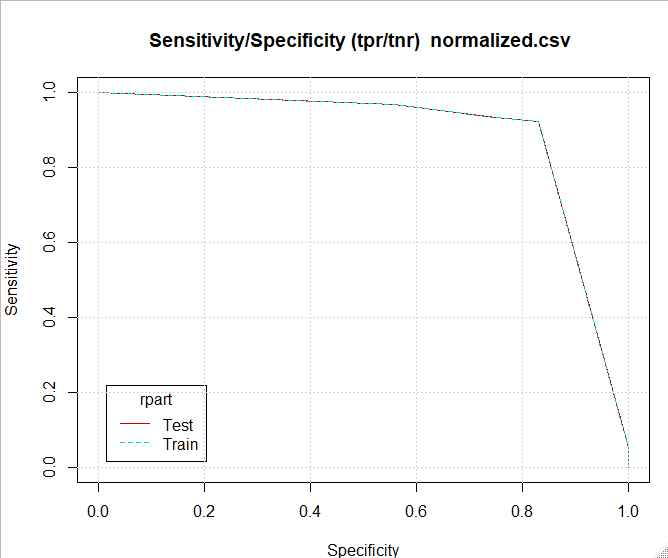
# Building Models to Predict Good Hard Drives

|  |
| --- |
| 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:  Accuracy - % of correct predictions  Precision - % of positive cases which were correct  Sensitivity/Recall - a portion of actual positive cases which were predicted correctly  Specificity - a portion of actual negative cases which were predicted correctly.  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)?

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Results 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.  Decision Tree:   |  |  |  |  | | --- | --- | --- | --- | | Column1 | Predicted |  |  | | Actual | 0 | 1 | Error | | 0 | 304 | 62 | 16.9 | | 1 | 32 | 385 | 7.7 | |
| Overall error: 12%, Averaged class error: 12.3% |

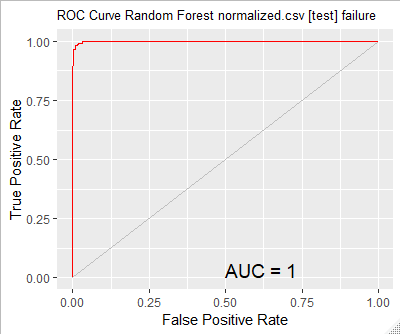


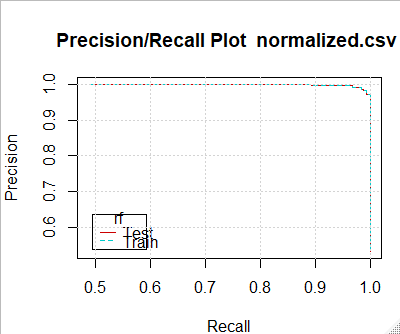


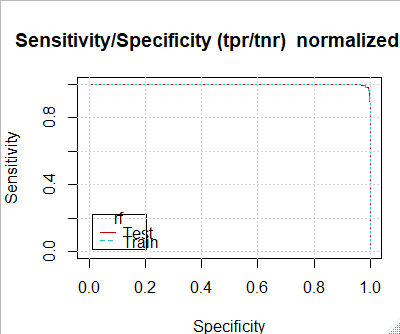
Random Forest:

|  |  |  |  |
| --- | --- | --- | --- |
| Column1 | Predicted |  |  |
| Actual | 0 | 1 | Error |
| 0 | 355 | 11 | 3 |
| 1 | 3 | 414 | 0.7 |

Overall error: 1.8%, Averaged class error: 1.85%



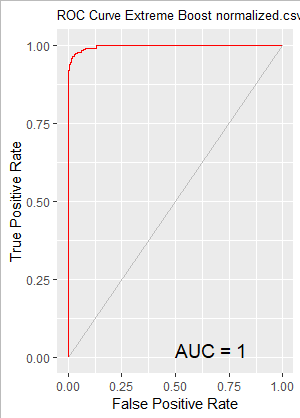


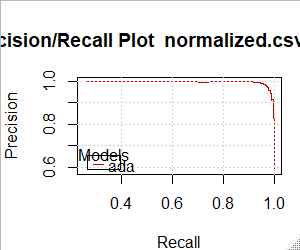
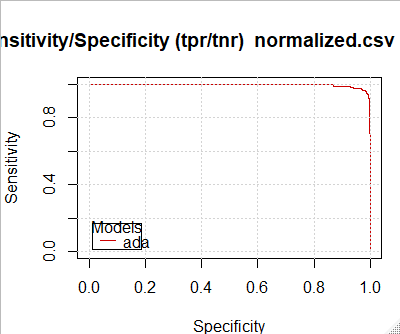


Adaptive Boost:

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

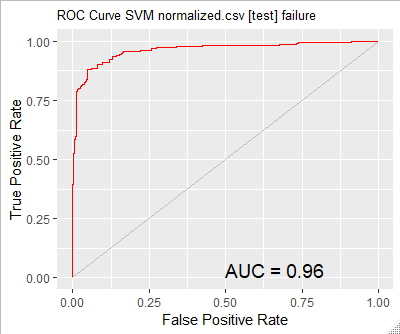
Overall error: 3%, Averaged class error: 2.95%

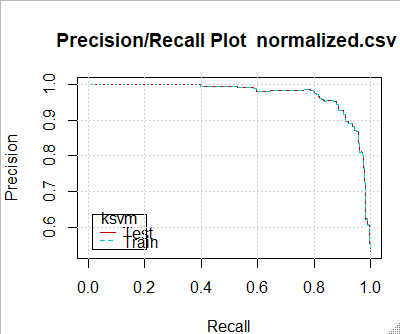


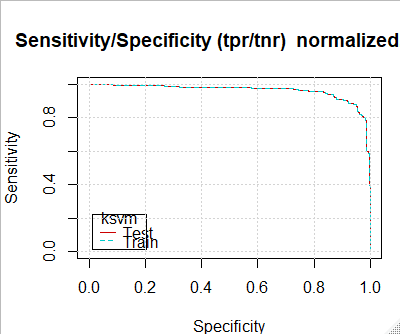
 

SVM:

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

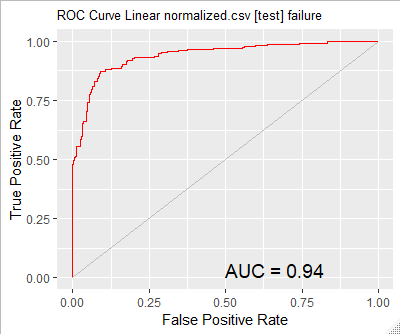
Overall error: 9.2%, Averaged class error: 9.15%  


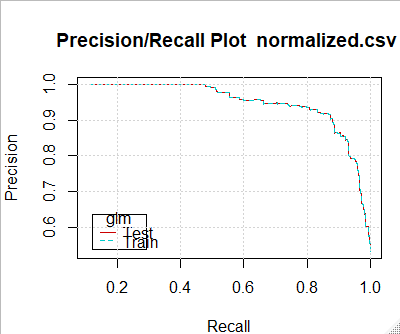


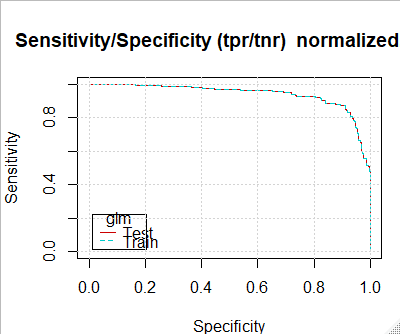


Logistic Regression:

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

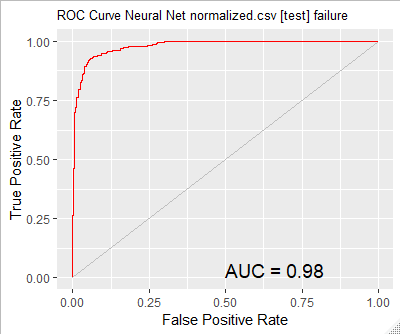
Overall error: 12.8%, Averaged class error: 12.85%

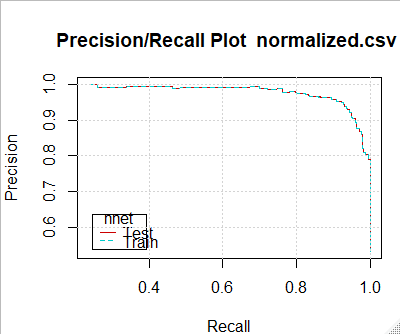


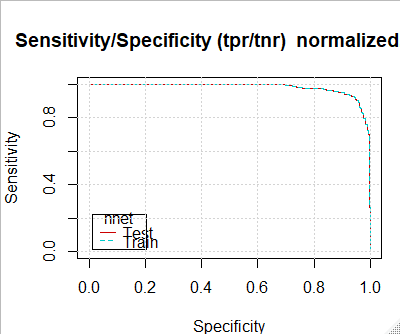


Multi-Layer Feed Forward Neural Network:

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

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





# What in Black Blazes is this about?

|  |
| --- |
| 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.  Random Forest, Ada-Boost and Multi-Layer Feed forward networks yielded good results had achieved accuracy up to 98.8% accuracy  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. |