Proactive Prediction of Hard Disk Drive Failure

Wendy Li (liwendy) - CS221, Ivan Suarez (isuarezr) - CS221, Juan Camacho (jcamach2) - CS229

Introduction

In data center environments, hard disk drive (HDD) failures are a rare but costly occurrence. Therefore, HDD vendors are highly motivated to reduce the rate of failures as a cost saving measure. But currently, HDD manufacturers use Self-Monitoring and Reporting Technology (SMART) attributes collected during normal operations to predict failures. SMART attributes represent HDD health statistics such as the number of scan errors, reallocation counts and probational counts of a HDD. If a certain attribute considered critical to HDD health goes above its threshold value, the HDD is marked as likely to fail [Pinheiro].

Our project focuses on applying machine learning to improve prediction accuracy over baseline heuristics in hard disk drives. The goal of our project is twofold: 1) to achieve a higher recall, precision and accuracy than our baseline implementation modeled off of current enterprise failure detection models. 2) to analyze which of our subset of machine learning models is best suited towards predicting failure of HDDs. We analyze three different algorithms: Logistic Regression, Naive Bayes and Random Forest, to see which has the highest accuracy, recall and precision when predicting HDD failures.

Related Work

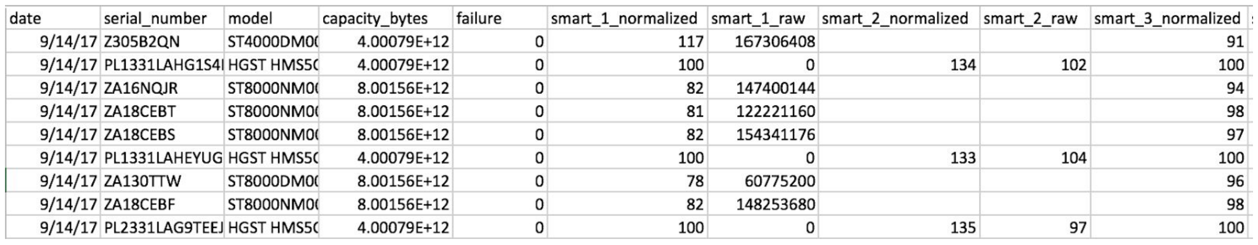
Pinheiro et al analyzed failure trends in a large disk drive population of over one hundred thousand enterprise HDDs at a Google data center. They found that specific SMART parameters (scan errors, reallocation counts, offline reallocation counts, and probational counts) had a large impact on failure probability. Most importantly, a large fraction of failed drives showed no signs of failure in all of its monitored SMART features, making it unlikely to achieve an accurate predictive failure model that can be built based on SMART signals alone [Pinheiro]. Similarly, BackBlaze analyzed the correlation rates between its HDD failures and SMART attributes and found that SMART 5, 187, 188, 197, and 198 had the highest rates of correlation to HDD failure. These SMART attributes are also related to scan error, reallocation count and probational counts [Klein].

Pitakrat et al evaluated 21 machine learning (ML) algorithms for predicting HDD failure. Pitakrat et al found that of the 21 ML algorithms tested, a Random Forest algorithm produced the highest area under a ROC Curve (AUC), while a Nearest Neighbor classifier had the highest F1-score.

Hughes et al also studied machine learning methods for predicting failures in HDDs. They analyzed the performance of Support Vector Machines (SVM), rank-sum and multi instance Naive Bayes. SVMs achieved the highest performance with a 50.6% detection and 0% false alarm rate [Hughes et al].

Dataset

We used the BackBlaze Dataset located on Backblaze.com. Backblaze is an online backup and cloud storage provider that open sources public information on 86,529 of its enterprise hard disks. Backblaze provides daily snapshot reports of its HDDs' SMART attributes and health from the 2013-2017 period. These HDDs are sourced from a multitude of HDD vendors, including: Seagate, Hitachi, HGST, Western Digital , Toshiba and Samsung and are sized from 2TB to 8TB [Beach].



date	serial_number	model	capacity_bytes	failure	smart_1_normalized	smart_1_raw	smart_2_normalized	smart_2_raw	smart_3_normalized
9/14/17	Z305B2QN	ST4000DM0(	4.00079E+12	0	117	167306408			91
9/14/17	PL1331LAHG1S4I	HGST HMS5(	4.00079E+12	0	100	0	134	102	100
9/14/17	ZA16NQJR	ST8000NM0(	8.00156E+12	0	82	147400144			94
9/14/17	ZA18CEBT	ST8000NM0(	8.00156E+12	0	81	122221160			98
9/14/17	ZA18CEBS	ST8000NM0(	8.00156E+12	0	82	154341176			97
9/14/17	PL1331LAHEYUG	HGST HMS5(	4.00079E+12	0	100	0	133	104	100
9/14/17	ZA130TTW	ST8000DM0(	8.00156E+12	0	78	60775200			96
9/14/17	ZA18CEBF	ST8000NM0(	8.00156E+12	0	82	148253680			98
9/14/17	PL2331LAG9TEEJ	HGST HMS5(	4.00079E+12	0	100	0	135	97	100

Figure 1. A snapshot of BackBlaze data sets. The first row contains the feature header and all following rows contain data collected.

As seen in Figure 1, each csv file contains 90 variables which represent SMART features in raw and normalized form as well as a the date the data was collected, serial number of the HDD, model type, total capacity in bytes and failure status. Once a HDD's failure status goes high, it has failed and is subsequently removed from all future data lists. When a new HDD is added to the datacenter, it appears in the csv file on the day it was added. Since each vendor has its own method of collecting SMART attributes, not all HDDs contain the same set of SMART attributes. This can be seen in the sparse smart_2_normalized and smart_2_raw columns. In the cases where the SMART attribute of a disk is empty and it is a feature being used for prediction, we zero-fill the data point.

To achieve consistent results across models, all models were trained on 9 months of data from Q1 2017 to Q3 2017. All models then used the Q1 2016 dataset for validation and finally the Q2 2016 data set for testing

Feature Selection

Ninety variables and millions of data points not only take an extensive amount of time to train and test on, but can also lead to overfitting. To reduce the computational workload and improve the performance of our models, we chose to select only the most relevant features and avoid features with a large amount of unfilled data points.

Due to limited RAM on our personal computers, we limited our feature size to at most 10 SMART attributes. We chose to keep BackBlaze's original five features SMART 5, 187, 188, 197, and 198 because they were selected by BackBlaze for their high correlation to HDD failure. We also included 5 additional features: SMART 9, 193, 194, 241, 242. These SMART attributes were chosen based on an analysis by El-Shimi where he reported the associated weights of each SMART feature in his random forest model [El-Shimi]. Aside from the SMART 5,187, 188, 197 and 198, SMART 9, 193, 194, 241 and 242 had the highest weight values reported. Below, we have defined what each SMART attribute represents.

Logistic Regression used SMART 193, 194, 241, 197 and 9, Naive Bayes used SMART 5,187, 188, 197, and 198, Random Forest used SMART 5,187,188,197, 198, 9, 193, 194, 241 and 242 as its input features.

Preprocessing

The main challenge of our project was handling a highly imbalanced data set with a low rate of failed data points. Raw SMART values also varied wildly from 0 to values as high as 200,000. Below are some of the preprocessing steps taken to address these challenges and improve our model performance.

1. Choose raw over normalized SMART Data points
    a. Choosing both data sets is redundant as they both represent the same data points. How SMART attributes are normalized is not clearly defined and well understood, so we chose to go with raw data and manually normalize data points if needed.
2. Failure status smoothing/ backtracking
    a. For every hard disk drive that fails, we backtrack on their previous n days of data and replace their failure status with a 1 rather than the original 0. For the purposes of the project, we set n = 10.
3. Filter out all HDD models besides Seagate models
    a. SMART attributes are unique to each vendor and the way SMART attributes are measured between each vendor is not guaranteed to be consistent.
    b. Although filtering out all other vendors in both our training and testing sets means we aren't optimizing our model's prediction accuracy for other HDD vendors, the same methods we implemented here can be implemented on any set of HDD models.
4. Balance out the data set
    a. Failure rates on BackBlaze HDDs are around 0.5%. Even when performing failure status smoothing, there are still a disproportionate amount of data points marked as non-failure vs. failure by a large margin. To help further counteract the imbalanced data set issue, after smoothing, we create a balanced data set where we include all failure data points and downsample operational data points to an equal number of non-failure data points, selected randomly.

Approach and Methods

Baseline

Our baseline analysis mimics what BackBlaze currently implements in its failure prediction system. We analyze five SMART attributes (SMART 5, 187, 188, 197, 198) and predict a HDD will fail if any of these critical raw SMART attributes are greater than 0.

Our goal is therefore to maintain as high of a TPR with a maximum TPR equal to the baseline analysis, and to focus on reducing the FPR to 0.

Logistic Regression

Logistic Regression is one of the basic tools for performing binary classification. One of the assumptions made in order for Logistic Regression to potentially perform well is that the data is linear. This means that the score we obtain from Logistic Regression is affected proportionally to changes in the feature values in a linear fashion. The tool mainly served as a second baseline in some sense as it was our first attempt at the classification problem beyond implementing the simple baseline. We also employed L2 regularization.

We attempted several preprocessing methods on the smart attribute values in order to improve our prediction rates. They are the following:

1. No feature preprocessing - We fed in the raw SMART attributes as is into the model, only to observe a TPR of 0.
2. Convert to binary - Raw values are converted into binary values, where the corresponding binary value is of 1 represents all raw values greater than 0 and 0 otherwise.
3. Logarithmic - We transform the given raw value, x, into floor(log(x + 1)) in order to better distinguish zero values from non-zero values.

Naive Bayes

The Naive Bayes classifier model makes the assumption that the value of a feature is conditionally independent of the value of another feature given some class label. Among the different techniques used for building Naive Bayes models, we chose Multinomial Naive Bayes, which assumes that the probability of a feature value given some class label is sampled from a multinomial distribution. For regularization, we use Laplace smoothing.

Although the conditional independence assumption may not necessarily hold for all feature pairs (e.g. “unstable” sector count and uncorrectable sector count are well correlated with each other), it’s not a weak assumption for other feature pairs used in our training model (e.g. command timeout and reported uncorrectable errors). Also, despite the large variance in some features, some value vectors for certain features are more-or-less already bucketized. Thus, counting approach of Naive Bayes for computing probabilities is appropriate for our training set.

Our initial approach to tuning our Naive Bayes model involved doing further bucketization of our feature values. In order words, the codomain of feature values was further reduced. One special case of this method was to make the values binary (either 0 if the original value was 0 or some constant if the original value wais greater than 0). However, this resulted in a significantly worse true positive rate than the approach of simply using the raw values unmodified. The most likely reason for it was that we lose some information about the actual distribution of each feature, resulting in higher bias.

Random Forest

Random forest is an ensemble tool which takes a subset of observations and a subset of variables to build a group of decision trees. It builds multiple such decision trees and amalgamate them together to get a more accurate and stable prediction [Chen].

We chose to use a random forest model as it was robust to noise, potentially caused by poorly correlated SMART features. Since Random Forests are designed to reduce the overall error rate, it tends to focus on improving the prediction accuracy of a majority class which can hurt the prediction accuracy of our minority failed hard disk class [Chen et al].

In tuning our Random Forest model we implemented the following additional processes to our dataset and feature selection.

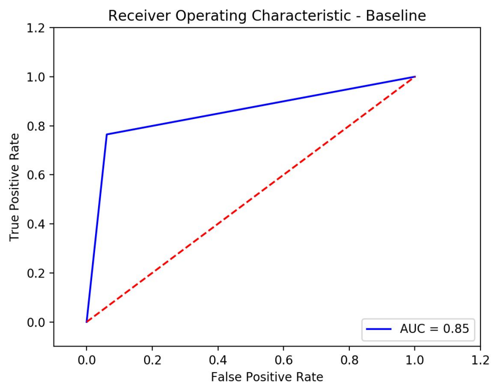
1. Use raw values instead of normalization - normalization has no impact on the performance of a decision tree, therefore for a Random forest implementation, normalization should also have little to no impact as well.
2. Increase feature count to 10 - Because Random Forest was more immune to noise compared to other models such as Naive Bayes, we increased the feature count from 5 to 10 features.
3. Low max depth - since our features had low correlation to failure and were very noisy, we lowered our max depth to 1 to best handle the noisiness of our dataset.

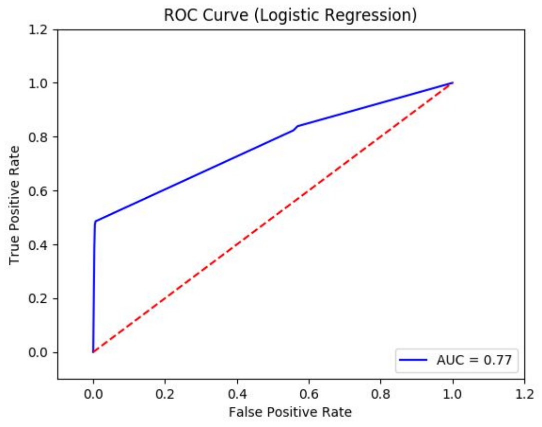
Results

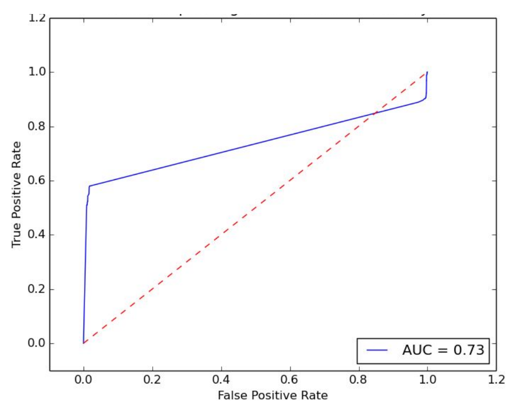
The Q2 2016 dataset we tested our models on contained 40,918 Seagate HDD models. Of these HDDs, 289 failed in this time period and of the 289, 68 had SMART 5,187,188, 197 and 198 attributes that were all zero. These 68 data points were therefore indistinguishable from any of the 38,135 operational HDDs that also had all zero SMART attribute values. Results are below:

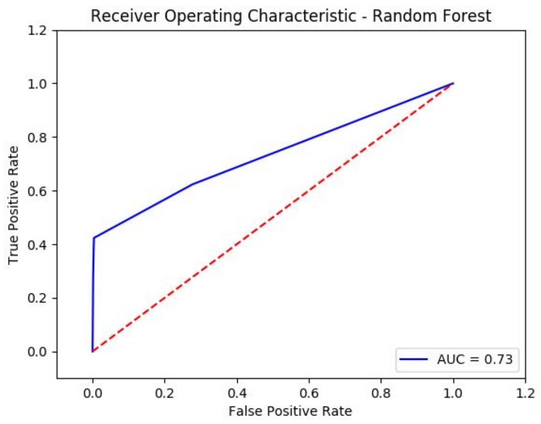
	Equation	Baseline	Logistic Regression	Naive Bayes	Random Forest
Precision	TP/(TP+FP)	0.08139963168	0.4093406593	0.2861788618	0.408839779
Recall	TP/(TP+FN)	0.7647058824	0.5155709343	0.6089965398	0.5121107266
Accuracy	(TP+TN)/(TP+TN+FP+FN)	0.9373869691	0.9913241116	0.9865096046	0.9913241116
F1- score	**2*TP/(2TP + FP+FN)**	**0.1471371505**	**0.4563552833**	**0.389380531**	**0.4546850998**

Below we have the ROC curves for each model:









Discussion

After performing error analysis on our false negative values and false positive values, we saw that the vast majority of our false negative values have features that are all 0. These hard disk drives showed no sign of failure among our given feature set and are virtually impossible to predict accurately without including more features. We therefore set our goal false negative value to 68 (baseline false negative value) rather 0 as we cannot predict HDDs with no failure warnings correctly without significantly increasing the number of false positives

We see from the results that Logistic Regression performs best in terms of F1 score. One potential reason may be that, as baseline suggests, when one of the feature values is 1 or greater, the HDD is now much more likely to fail. As opposed to baseline which predicts failure upon seeing a non-zero feature value, logistic regression gives different weights to different features, so seeing one non-zero value may not be enough to warrant a failure label. Finally, if it were the case that perhaps a failure label is warranted, upon training, the Logistic Regression model would probably be able to figure that out.

An interesting observation we address is the ROC curve of the Naive Bayes classifier. At one point, when raising the threshold of predicting failure high enough, the Naive Bayes classifier actually performs worse than random as suggested by its ROC curve. One explanation is that when an HDD does have at least one non-zero feature value, it is now much more likely to fail. Building upon the observation, there are cases when an HDD may have many non-zero feature values and yet ends up not failing. So these kinds of data points are exactly what may cause the classifier to perform worse than random.

Conclusion

In conclusion we were able to achieve higher accuracy and precision but not recall compared to baseline. It was expected that we could not achieve a higher recall value than the baseline, because the baseline predicts all disks with any sign of failure to fail. If the datacenter operator or vendor prioritized a high true positive rate and therefore recall, then going with our naive baseline approach is ultimately best. If recall and precision is equally important though, then implementing a Logistic Regression model will achieve the highest F1-score and best optimize for high recall and precision. Our analysis shows that while we cannot achieve near 100% prediction accuracy using machine learning with the current data we have available for HDDs, we can improve our prediction accuracy over the baseline approach. Machine learning algorithms are capable of providing more accurate predictions of HDD failures, with readily available data, than what is currently implemented in today's industry.

For future work, we would consider acquiring more powerful computational resources on which we can run our machine learning models on a much larger training set. Using better computing platforms optimized for these kinds of workloads could also help us run bigger experiments for feature selection, where we add other features into the training set and evaluate how it affects the accuracy and precision of the model. In addition to that, we would consider using other machine learning models such as neural networks, which, unlike Naive-Bayes classifiers, would take into account correlation between feature and other non-linearities in the data.

References

Beach, Brian. “Hard Drive SMART Stats.” *Backblaze Blog | Cloud Storage & Cloud Backup*, 27 Jan. 2015, www.backblaze.com/blog/hard-drive-smart-stats/.

Botezatu, Mirela Madalina, et al. "Predicting Disk Replacement towards Reliable Data Centers." *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016.

Chen, Chao, et al. “Using Random Forest to Learn Imbalanced Data.” *University of California, Berkeley* (2004).

El-Shimi, Ahmed. “Predicting Storage Failures.” VAULT-Linux Storage and File Systems Conference. VAULT-Linux Storage and File Systems Conference, 22 Mar. 2017, Cambridge.

Hamerly, Greg, and Charles Elkan. "Bayesian approaches to failure prediction for disk drives." *ICML*. Vol. 1. 2001.

Hughes, Gordon F., et al. "Improved disk-drive failure warnings." *IEEE Transactions on Reliability* 51.3 (2002): 350-357.

Klein, Andy. “What SMART Hard Disk Errors Actually Tell Us.” *Backblaze Blog | Cloud Storage & Cloud Backup*, 6 Oct. 2016, www.backblaze.com/blog/what-smart-stats-indicate-hard-drive-failures/.

Murray, Joseph F., Gordon F. Hughes, and Kenneth Kreutz-Delgado. "Machine learning methods for predicting failures in hard drives: A multiple-instance application." *Journal of Machine Learning Research* 6.May (2005): 783-816.

Pinheiro, Eduardo, Wolf-Dietrich Weber, and Luiz André Barroso. "Failure Trends in a Large Disk Drive Population." *FAST*. Vol. 7. No. 1. 2007.

Pitakrat, Teerat, André van Hoorn, and Lars Grunske. "A comparison of machine learning algorithms for proactive hard disk drive failure detection." *Proceedings of the 4th international ACM Sigsoft symposium on Architecting critical systems*. ACM, 2013.

Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

“Hard Drive Data and Stats.” *Backblaze*. https://www.backblaze.com/b2/hard-drive-test-data.html

Contributions

All of us discussed how to approach this problem and what Machine Learning methods to try. We also did our own research individually and compiled the list of references that we have presented above. Each of us also worked on turning each of the models separately. Contributions by each member are more specifically detailed below:

Wendy Li: worked on the Random Forests model. Also initially came up with the outline for the report and poster, and suggested how to do ROC curves.

Ivan Suarez: worked on the Logistic Regression model. Also wrote up some very useful scripts for importing csv files into Python data structures, and suggested to use a balanced training set.

Juan Camacho: worked on Naive Bayes model. Also worked on doing the baseline script, and suggested to do up sample the failure labels (up to 10 days before hard disk fails)

We also worked on designing the poster and on writing up the report together. Each of us read and reviewed both the posters and the report.