Capstone Project Proposal: Hard drive fail probability

by Darius Murawski

Domain Background

Hard drives are used to save the data from the operating system and different applications that are running on the server. The average price for a gigabyte is dropping and the demand for more space on the servers is growing. This results in a higher total number of hard drives running.

Fails on a hard drive can be recovered using different techniques like raid settings or mirroring the data in different data centers. But the broken hard drive have to be replaced sooner or later with a new one, to make sure more fails on associated hard drives don't break the data consistency. For each broken hard drive, somebody have to drive to the storage system, look it up in the storage system and replace it. This procedure have to be done each time one drive fails in the worst case and each time generating maintenance costs for the operating company. Each drive have some values that are intended to show the health state of a hard drive. They are called S.M.A.R.T. and are manufacturer specific.

Looking up in papers and articles, this kind of problem is referenced as predictive maintenance (PdM): Before waiting that something breaks, we replace the appropriate part in a regular maintenance to make sure that the entire system is able to continue running as expected.

I found the following related papers and articles:

Authors	Topic / Title	Document
Julia Scavicchio	Definition "Predictive Maintenance" (PdM)"	Link
Jennifer Ho	Overview of industries, using Algorithms to reduce their machine downtime with further links	Link
Taylor Short	What type of sensors can be used for predictive maintenance	Link
Gian Antonio Susto, Andrea Schirru, Simone Pampuri, Seán McLoone, Alessandro Beghi	Machine Learning for Predictive Maintenance: A Multiple Classifier Approach	Link

Dr. Miguel A. Sanz Bobi, Maria Cruz García, Javier del Pico-Aznar	SIMAP: Intelligent System for Predictive Maintenance: Application to the health condition monitoring of a windturbine gearbox	Link
Hongfei Li, Dhaivat Parikh, Qing He, Buyue Qian, Zhiguo Li Dongping Fang, Arun Hampapur	Improving rail network velocity: A machine learning approach to predictive maintenance	Link
Eduardo Pinheiro, Wolf-Dietrich Weber and Luiz Andre Barroso, Google	Failure Trends in a Large Disk Drive Population	Link or Link
Various	General Introduction into S.M.A.R.T.	Wikipedia

Problem Statement

This costs can be reduced by replacing more drives than just the broken one by the maintenance people, as they only have to went to the storage system once and not several times. But what drives should they replace? In this Capstone project I want to generate a fail probability for each of the drives running in the storage system. The drives with a predicted fail should be replaced beforehand to reduce the maintenance costs in mid-range for the company operating the storage system.

The described problem can be seen as classification problem, where the allowed labels are only 0, for “hard drive is running” and 1 for “hard drive failed”. When generating predictions, 0 represents “hard drive will run further” and 1 for “hard drive will (soon) fail and should be replaced.

Datasets and Inputs

The dataset using for this capstone project can be downloaded from the backblaze. They provide several huge csv files with the S.M.A.R.T. values for each hard drive they are operating, including the values of hard drives that failed just before the fail. This data is also historically splitted. Additionally of creating a test and training set, further evaluation can be done by training only for a given year and than comparing the generated predictions for data that follows in the next time line. For example train and validate on data of 2013 and run a further evaluation on the data of 2014.

As the Data is really huge (4,5 GB Compressed, 23,7 GB uncompressed), it's also challenging to work with the data. As my device can handle upto 32GB of ram, I have to do smart separation of the data to be able to train models on the data. I also have the possibility to run more memory intensive operations on virtual machines in the cloud if required.

The Input Features are the hard drive model, and the returned S.M.A.R.T. values of the drive at the timestamp represented by "date". The data is highly unbalanced, as only about 1.84% drives in the reporting period between April 2003 and March 2018 failed (see: Link).

The data can be obtained from backblaze directly using the following link: https://www.backblaze.com/b2/hard-drive-test-data.html - the provided data is huge, so I will not include it in the repository, instead just link it. I will start with the files from Q4 2017 and Q1 2018. The Structure of the data is very nice explained on the linked page. I will just cite it here:

- 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 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.

As the data from 2018 have a different set of S.M.A.R.T. values, I have to check if this data can be used. If not, I will stay with the data from only Q3 and Q4 of 2017. I would split the data based for each hard drive model, to not load all the data into the RAM, so each prediction will be based for

Solution Statement

I want to train a model that returns a probability or a boolean value (for example: replace, Yes), given by the provided features, that returns a prediction for a hard drive fail. Drives with a high probability should be replaced by the maintenance team operating the storage system of the company.

Benchmark Model

This model is reproducible, as the training data is static and not rely on other sources. Googles results are as follows (link provided above, summary see wikipedia link):

- 60 days after finding uncorrectable errors (S.M.A.R.T. Value 198), the drive had 39 times higher change to fail

- First errors in S.M.A.R.T. 196 (Relocation) and 5 (offline Relocation) are strongly correlated to higher probabilities of failure
- 56% of the drives failed without recording any count in the four string S.M.A.R.T. Warnings (Scan Errors, Relocation Count, offline Relocation and probiatinal Count)
- 36% Failed without any S.M.A.R.T. error at all

This lets me create the benchmark model as following:
The total amount of fails is 1.84% for all drives. This 1.84% can now be splitted into three groups, provided using the google paper:

Fails without errors: 1.84 * 0.36 = 0.6624 %
Fails with any smart warning: 1.84 * 0.56 = 1.0304 %
Fails that could be predicted: 1.84 * (1 - 0.36 - 0.56) = 0.8 %

The Benchmark model, using random choice, predict with a change of 0.8% that a drive will fail. Each model have different chance to fail, this means that the 0.8 % is the average probability over every model in the dataset.

Evaluation Metric

I will use F Score as evaluation metric together with the accuracy, as this data is heavily unbalanced. I will split the data into test, train and validation.

Project Design

1. Downloading the Dataset using python
2. Unpacking the data
3. Create a csv for each model
4. Remove Models that not failed at all
5. Normalize the raw data to have every smart value per model in their range. If raw data is binary, remap it to own columns.
6. Calculating runtime of each drive given by timestamp
7. Split data into train (0.8), test (0.1), and validation set (0.1). Make sure that in each part "failed" drives are present.
8. Feature extraction for each model
9. Train a decision tree classifier on the data.
   a. Depending on the performance, try different algorithms with and without a neuronal network and challenge them against each other (for example Xgboost and LightGBM). Drive running: 0, Drive Failed: 1
10. Compare the fails with the fail probability generated by the model