UNIVERSITY OF CALIFORNIA

Los Angeles

Hard Drive Failure Prediction for Large Scale Storage System

A thesis submitted in partial satisfaction

of the requirements for the degree

Master of Science in Statistics

by

Xiaohong Huang

2017

ABSTRACT OF THE THESIS

Hard Drive Failure Prediction for Large Scale Storage System

by

Xiaohong Huang
Master of Science in Statistics
University of California, Los Angeles, 2017
Professor Ying Nian Wu, Chair

Data centers use large numbers of hard drives as data storage devices and it is an increasing challenge to maintain the reliability of the storage system as the number of the hard drives increases exponentially. Manual monitoring does not seem to be efficient for large scale storage systems. Typically, the distributions of healthy hard drives and failed hard drives are highly imbalance. In addition, the size of the training data is large for large scale storage systems. The existence of such challenges makes the hard drive failure prediction problem interesting. In this thesis, several classification models are applied to the hard drive S.M.A.R.T. data from 34,970 hard drives for failure prediction, and the results are compared. Based on the analysis, XGBoost provides the best overall prediction result and it is able to process the data efficiently.

The thesis of Xiaohong Huang is approved.

Hongquan Xu

Qing Zhou

Ying Nian Wu, Committee Chair

University of California, Los Angeles

2017

To my parents . . .

who provide love and support throughout my life

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

It is common for data centers to use hard drives as data storage devices. As the number of hard drives used gets larger, it becomes more difficult to maintain the reliability for the storage systems. For large companies that heavily rely on data, it might use more than 10,000 hard drives and it will be challenging to manually monitor so many hard drives. When hard drives fail, it might lead to loss of data and it could be a serious problem for the users. Even though multiple copies of data can be stored in the system as backup, it might increase the cost at the same time. For monitoring purpose, it is a common approach in the industry to use thresholds on certain metrics. However, thresholds might need to be updated periodically depending on the business needs and it can cause many false alarms if the thresholds are not set up correctly. In addition, the distributions of the attributes might be different for different models of hard drives, and it seems to be challenging to maintain so many thresholds manually in a large data center. Therefore, it seems useful if a model can be developed to predict the hard drive failure and then the operators might use the prediction result to improve the system reliability and reduce the cost.

There are some challenges for such failure prediction. First, to reduce the cost, having low false positive rate seems to be important. Here false positive means a healthy hard drive is classified as unhealthy. If there are only a small percentage of hard drives that are unhealthy and the model detects a much higher percentage of unhealthy hard drives, it might result in replacing more healthy hard drives than without using the detection model. It will defeat the purpose of failure prediction if it cannot reduce the labor and the cost of replacing new disks. Second, hard drive failure is a rare event in general, so the data is very likely to be unbalanced. Not having a balanced data set is very likely to affect the performance of the classifiers as there might not be enough pattern to separate the two classes and the model might be biased toward the majority class. Third, with so many hard drives in the data center, the training dataset is usually large. Therefore, the computational cost might be an issue when training the models.

Some researches have been done for hard drive failure prediction. Murray *et al.* [12] apply the SVM, multiple instance Naive Bayes and non-parametric statistical methods (rank-sum test and reverse arrangements test) on a balanced dataset with 369 hard drives. The best detection rate is 50.6% using SVM, but it takes 17,893 minutes to perform the grid search for parameter selection. Zhao *et al.* [14] use a Hidden Markov Model based approach on a dataset with 277 hard drives, and the best detection rate is 52% at 0% false alarm. However, there are less than 90,000 observations and it takes 3848.4 seconds to train the data. To explore the issue of unbalanced data in hard drive failure prediction, Zhu *et al.* [15] use a dataset with around 1.9% failed hard drives and there are 23,395 hard drives in total. They only use subset of the majority class and all of the minority class to construct the training set and testing set. Using SVM with RBF kernel, the best result is detection rate of 80.0% with 0.3% of good drives that are mis-classified as failed. Even though many methods have been implemented to solve the failure prediction problem, there seems to be potential for improvement. Therefore, several classifiers are applied to the failure prediction problem in this thesis, with the hope that the classifiers will be computationally fast for real world large dataset along with good prediction performance.

The organization of the thesis is as follows. Chapter 2 provides some background information of S.M.A.R.T. data and the descriptions of the dataset. Chapter 3 provides some preliminary analysis and description of the preprocessing steps. Five classification models that are used in the thesis are briefly described in chapter 4. Chapter 5 discusses how the result is evaluated. In addition, results from different models are further analyzed and compared.

CHAPTER 2

Hard Drive S.M.A.R.T. Data

Every disk drive has the S.M.A.R.T. (Self-Monitoring, Analysis and Reporting Technology) monitoring system, which reports various attributes related to the hard drive reliability. The raw value and normalized value are provided for each S.M.A.R.T. attribute. The way of calculating the normalized value is vendor specific and it might not be provided to the users. Since the normalized data is hard to interpret, only the raw value data is used for training models.

The dataset is collected from the data center of an online backup service provider called Backblaze [9]. The data is from 10/01/2016 to 12/31/2016. Each row represents a daily snapshot of one operational hard drive. The snapshot is from the S.M.A.R.T. data. Data from multiple brands and models of hard drives are observed, and there are around 50 different models of hard drives in the dataset. To avoid the differences between different models of hard drive, only the hard drives of model number ST4000DM000 are used for the analysis. This model number is chosen because it corresponds to the highest number of hard drives used in the data center. Each hard drive has a different serial number and each has a capacity of 4 terabytes. There are 34,970 hard drives in total, which consists of 34,736 good hard drives and 234 failed hard drives. The ratio between failed hard drives and healthy hard drives is around 0.67%, whose ratio is even smaller than the dataset used by Zhu *et al.* [15]. There are 3,196,552 daily observations in the dataset with no missing values.

After removing attributes with zero variance, there are 23 attributes left. The detailed descriptions of the attributes are in Appendix A.1. The response variable is the attribute called "failure", which has value of 1 if the observation is from a disk drive that is replaced during the observed period and value of -1 if the observation is from a healthy disk drive.

Most healthy hard drives have 92 observations unless it is used as a replacement during the observed interval. For hard drives that need to be replaced, the shortest length can be 1 while the longest length can be 91. Figure 2.1 shows the distribution of the duration of failed hard drives during the observed period.

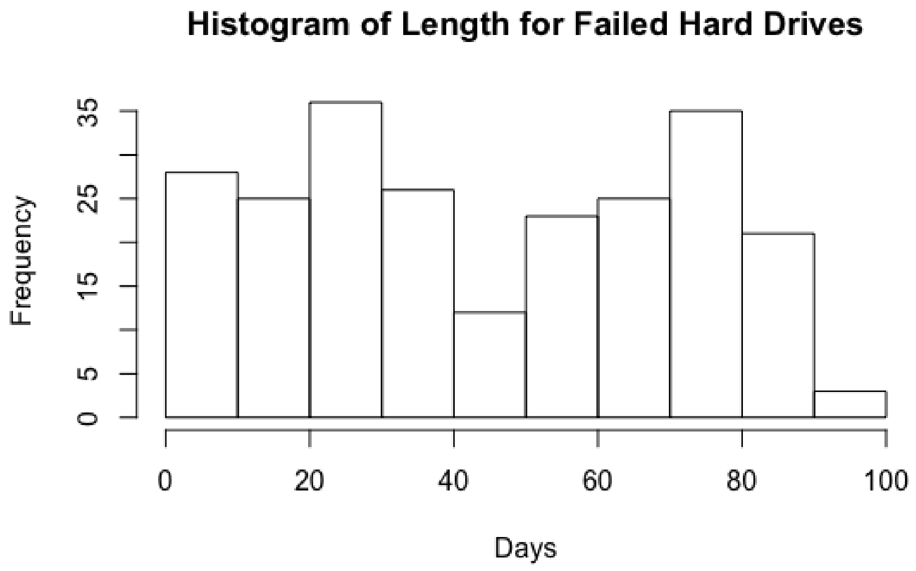

Figure 2.1: Histogram for The Length of Failed Hard Drives

CHAPTER 3

Data Exploration and Preprocessing

Figure 3.1 shows the correlation matrix for the S.M.A.R.T. attributes. Darker color in the graph means higher correlations. It shows some attributes are highly correlated with each other. Therefore, S.M.A.R.T. attributes 4, 9, 190, 198 are removed because they have correlation higher than 0.9. After removing the redundant features , there are 17 S.M.A.R.T. attributes left.

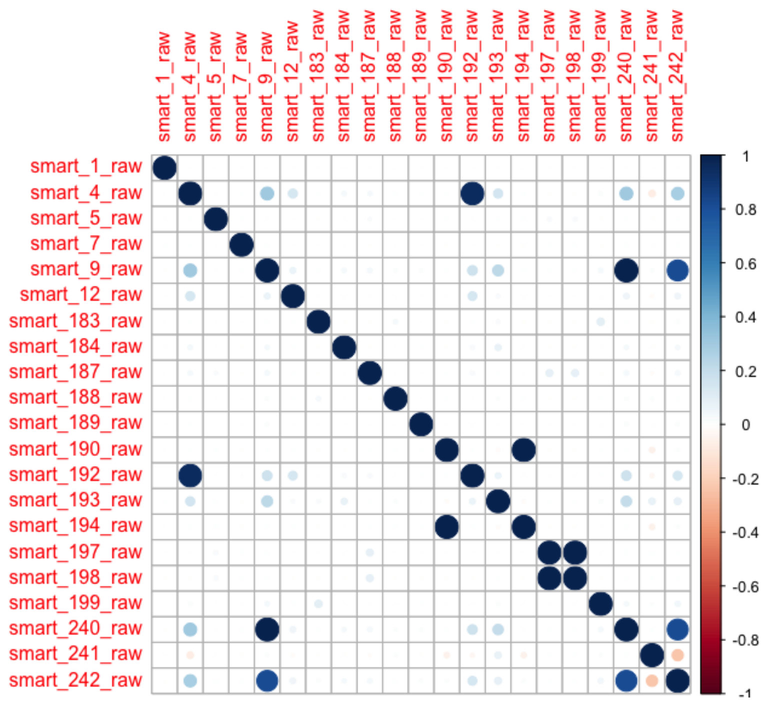

Figure 3.1: Correlation Matrix Between Attributes

According to the description of the data provider [10], most of the S.M.A.R.T. data is cumulative in nature. In addition, the raw data is associated with time and converting to the differenced data might help remove the trend. Therefore, the raw data is grouped by the serial number and it is converted into the difference between each snapshot and its previous value for each hard drive.

Next, it might be interesting to check whether there is a single feature that can separate the healthy hard drives and the failed hard drives. The density plots below show the 4 attributes that are used by the data provider [8] to monitor the hard drive failure. Since there are lots of zero values for both classes in the attributes and it affects the scale of the graphs, values of zeroes are removed when constructing the plots. Figure 3.2 and figure 3.3 are the density plots for attribute 187 and 197. Both plots show that failed hard drives are more likely to have larger range of values while the healthy hard drives are are more likely to be centered around zero with smaller variance. Figure 3.4 and figure 3.5 are the density plots for attribute 5 and 188, which show relatively small differences between healthy hard drives and failed hard drives.

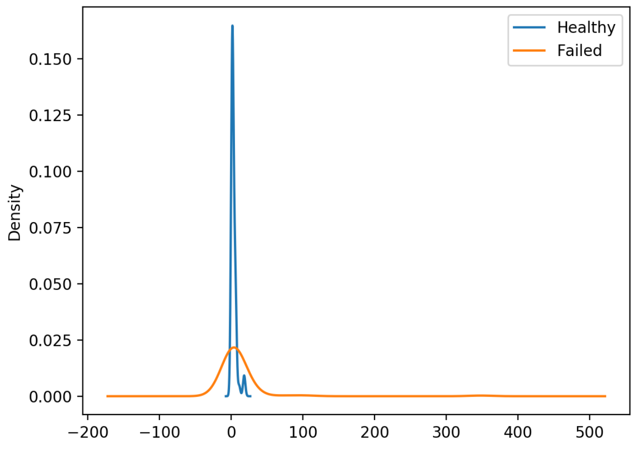

Figure 3.2: Density Plot for SMART Attribute 187

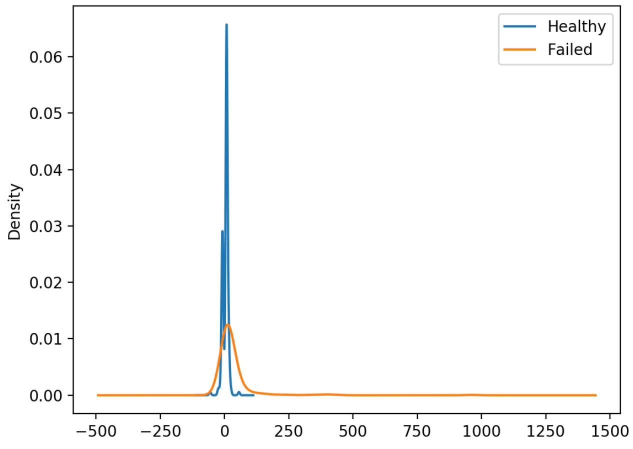

Figure 3.3: Density Plot for SMART Attribute 197

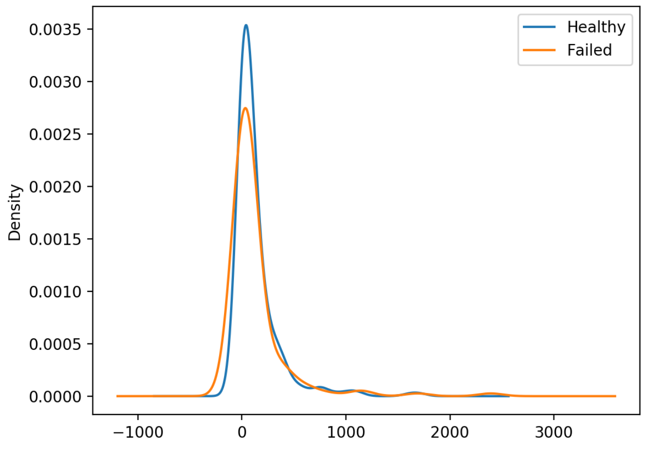

Figure 3.4: Density Plot for SMART Attribute 5

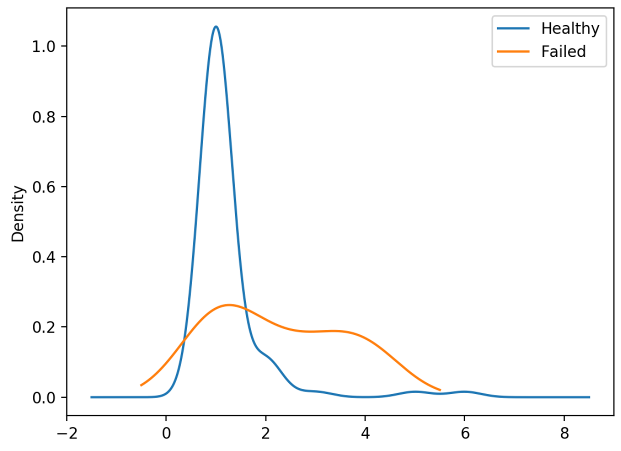

Figure 3.5: Density Plot for SMART Attribute 188

Figure 3.6 shows various plots related to the Principal Component Analysis. The plot on the top left shows the proportion of variance explained for each principal component, and it seems the proportion of variance explained for each principal component is less than 20%. The plot on the top right shows the cumulative proportion of variance explained. The two plots at the bottom are the scree plots. It looks like the number of principal component is over 15 in order to have 90% of the total variance explained, and there are only 17 features in the dataset. PCA only tries to maximize the variance of the projection and preserve the original variance as much as possible. Thus, the analysis might suggest that the variables have very weak correlation, so it might be difficult for the PCA to capture large share of variance in very few dimensional subspace. Since the number of principal components needed is not small, it might be better not to transform the dataset. The 17 S.M.A.R.T. attributes will be used to train the models.

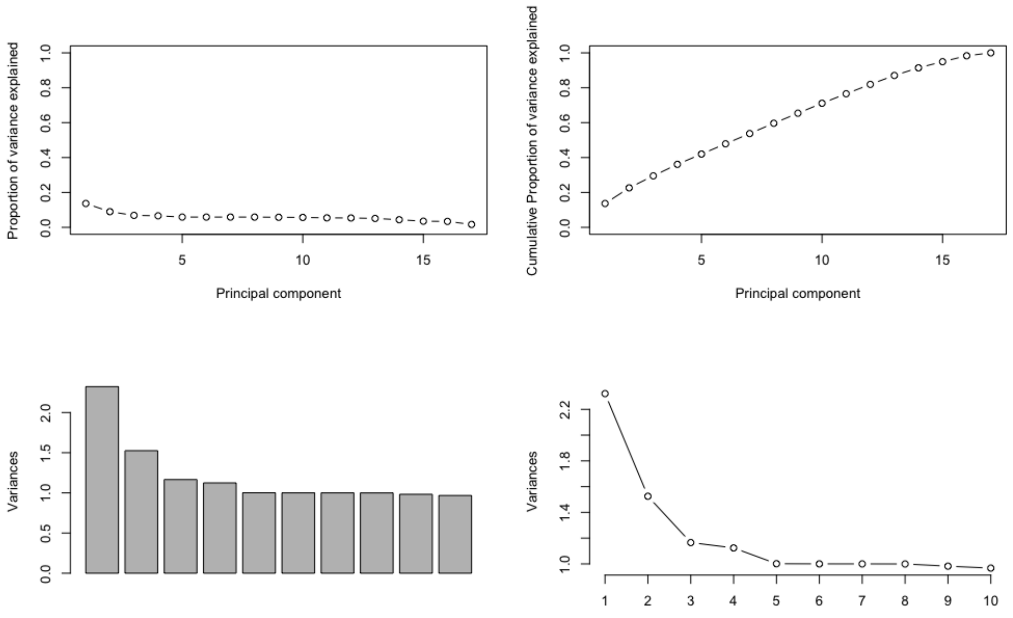

Figure 3.6: PCA Related Plots

The data from 10/01/2016 to 11/30/2016 is used as the training set and data in December is used as the testing set. As shown in figure 2.1, it might take several days for the hard drives to fail and it might not show any patterns of failure at the beginning. Similar to the approach used by Zhu *et al.* [15], only the last n observations for failed hard drives and all the observations for healthy hard drives are used to train the model. Here n is chosen to be 10.

There are 2,389,721 observations in the training set and 764,270 observations in the testing set. Since the units between each attribute are not all the same, normalization is performed on the training set first. Using the mean and standard deviation from the training set, normalization is performed on the testing set as well.

CHAPTER 4

Models

4.1 Quadratic Discriminant Analysis

QDA assumes each class density is multivariate normal:

$$P(X_i|Y_i = g) = N(X_i|\mu_g, \Sigma_g) = ((2\pi)^P|\Sigma_g|)^{-1/2}\exp[-\frac{1}{2}(X_i - \mu_g)^\top\Sigma_g^{-1}(X_i - \mu_g)]$$

The mean and covariance matrix for each class are assumed to be different from each other. It classifies Y_i by :

$$\begin{aligned}\arg\max P(Y_i = g|X_i) &= \arg\max P(g)P(X_i|g)\\ &= \arg\max_{g\in\mathcal{G}}[log(P(g)) - \frac{1}{2}log((2\pi)^P|\Sigma_g|) - \frac{1}{2}(X_i - \mu_g)^\top\Sigma_g^{-1}(X_i - \mu_g)]\end{aligned}$$

The prior P(Y= g) can be estimated by the proportion of instances in each class. QDA has parabolic decision boundary.

4.2 SVM Using Stochastic Gradient Descent

According to the documentation of LIBSVM [3], the primal optimization problem of C-SVM problem can be written as

$$\begin{aligned}\min_{w,b,\xi}\quad & \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C^+\sum_{y_i=1}\xi_i + C^-\sum_{y_i=-1}\xi_i\\ \text{subject to } & y_i(\mathbf{w}^\top\phi(\mathbf{x}) + b) \geq 1 - \xi_i\\ & \xi_i \geq 0, i = 1...l\end{aligned}$$

where C^+ and C^- are regularization parameters for positive and negative classes. By writing the objective function this way, SVM can handle the unbalanced data by assigning different weight for C^+ and C^-. For minority class, more weight can be put on the corresponding regularization parameter and the classifier will be forced to pay more attention to the minority class. However, the standard implementations for SVM might not be very efficient for large data set if the implementation loads all the data into memory at once. Therefore, the Stochastic Gradient Descent(SGD) is used to train the SVM in the analysis as it seems to be able to handle large scale data efficiently.

According to Bottou's paper [2],the stochastic gradient descent algorithm estimates the gradient of the empirical risk on the basis of a single randomly picked example $z_t = (x_t, y_t)$:

$$w_{t+1} = w_t - \gamma_t \bigtriangledown_w Q(z_t, w_t).$$

Here $Q(z_t, w_t) = \ell(f_w(x_t), y_t)$ represents the loss function between the prediction and the ouput and $\bigtriangledown_w$ represents the first derivative with respect to w. By setting $Q(z, w) = \lambda ||w||_2^2 + max\{0, 1 - yw^\top \phi(x)\}$ in the updating rule, this is equivalent to the linear SVM.

A number of epochs over the whole training dataset are made until the algorithm converges. $\{w_t, t = 1, ...n\}$ is a stochastic process that only depends on the randomly picked example z_t at each iteration. The stochastic gradient descent might be more efficient than the standard gradient descent for large dataset as it only calculates the gradient for one sample at each step, while the standard gradient descent computes the gradient using all the data.

4.3 Neural Network

Figure 4.1 is a simple illustration of a neural network with one hidden layer. The leftmost layer is the input layer, x_1, x_2, x_3 are the input data and b represents the bias. Except for the input layer, each node is a "neuron", which maps the weighted inputs to the output through an activation function. Let $(x, y) = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)})...(x^{(n)}, y^{(n)})\}$ denotes the input with $x^{(i)}$ denotes vectors in $\mathbb{R}^3$ and $y^{(i)} \in \{0, 1\}$. The activation function can be written as $h_{W,b}(x) = f(W^\top x) = f(\sum_{i=1}^3 W_i x_i + b)$. Some common choices of $f(\cdot)$ are the sigmoid function $= \frac{1}{1+\exp(-x)}$ and the tanh function $= \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$. Notice that every unit in the input layer is connected to the units in the hidden layer except for the bias. This means the output value of unit i in the hidden layer is given by $f(\sum_{i=1}^{n} W_i x_i + b)$.

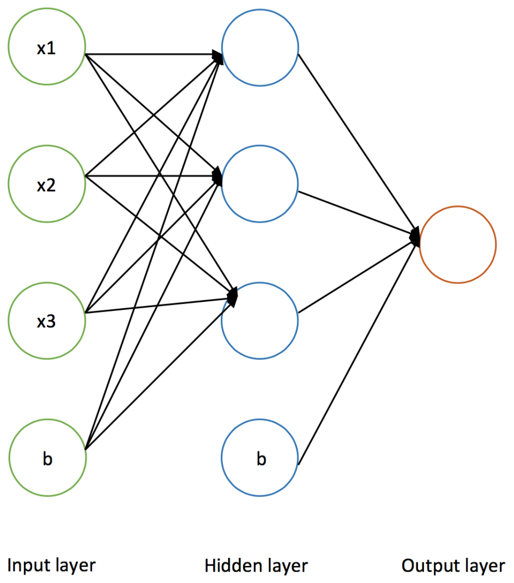

Figure 4.1: Neural Network Model

Here the neural network is trained using Backpropagation. The steps for Backpropagation Neural Network are as follows:

1. Perform a feedforward pass. It means calculating the activations $f(W^\top x)$ for all the layers up to the output layer.

2. Perform Backpropagation. It will first calculate the error signal for all units. Using the errors calculated, the gradients of the errors with respect to output layer weight are calculated by applying the chain rule. Then it calculates the gradients of the errors for the weights of the lower layers using the chain rule. The stochastic gradient descent can be applied here to find the weight and bias that will minimize the cost function.

To accelerate the convergence, a mini-batch gradient descent can be used instead. Mini-batch gradient descent is a compromise between the standard gradient descent and the stochastic gradient descent. Instead of using a single sample to compute the gradient, it uses data of batch size k to compute the gradient at each iteration.

4.4 AdaBoost

Let the output be denoted as $y_i \in \{-1, 1\}$ and let x_i be a vector, i = 1...n. A weak classifier $g(x)$ will map the input to one of the values in {-1, 1}. Suppose there are M weak classifiers $g_1(x), g_2(x)....g_M(x)$, then a strong classifier can be constructed by combining them through a weighted majority vote:

$$G(x) = \text{sign}(\alpha_1 g_1(x) + \alpha_2 g_2(x) + ... + \alpha_M g_M(x))$$

The objective is to find g_i and α_i to minimize the empirical error $Err(G) = \frac{1}{n}\sum_{i=1}^{n} 1(G(x_i) \neq y_i)$. In each iteration, the weights $w_1, w_2...w_n$ are applied to each observation and the weights will be updated. Observations that are misclassified in previous iteration will have higher weights, while those that are correctly classified will have lower weights. Algorithm 1 shows the steps for the computation.

Algorithm 1: Algorithm of Adaboost [5]

```
1 Initialize the weight uniformly $w_i = 1/n$, i = 1...$n$, $\sum_i w_i = 1$.
2 For t= 1... $M$:
    (a) Compute the weighted error $\epsilon_t$ for each weak classifier $g$:
        $$\epsilon_t = \sum_{i=1}^{n} w_i 1(y_i \neq g(x_i))$$
    Choose a weak classifier $g_t$ which has the least weighted error.
    (b) Get the weight $\alpha_t$ for the new classifier $g_t$:
        $$\alpha_t = \frac{1}{2} log \frac{1 - \epsilon_t}{\epsilon_t}$$
    (c) Update the weight of the training data:
        $$w_i \leftarrow \frac{1}{Z_t} w_i * \exp(-y_i \alpha_t g_t(x_i))$$
    Here $Z_t$ is the normalization function.
```

As shown in algorithm 1, the weights for the training data change in each iteration and the classification is performed on the reweighted data space, which makes Adaboost a proper classifier for the data imbalance problem. In SVM, the classifier is forced to pay more attention to the minority class by assigning more weight to the regularization parameter of minority class. In Adaboost, the classifier is forced to pay more attention to the misclassified points by updating the sample weights. In this sense, Adaboost and SVM are both solving the class imbalance problem though cost-sensitive learning.

4.5 XGBoost

XGBoost stands for "Extreme Gradient Boosting". Similar to other boosting algorithms, gradient boosting involves an ensemble of weak classifiers. XGBoost is derived from the gradient boosting algorithm proposed by Friedman [6]. Gradient boosting basically constructs m weak classifiers and the m+1 model F_{m+1}(x) is based on the previous m models through $F_m(x) = F_{m-1}(x) + \rho_m h_m(x)$. Here ρ is the weight that minimizes the loss function between y_i and $F_{m-1}(x) + \rho_m h_m(x)$. $h_m(x)$ is the new weak classifier that is fit based on the negative gradient of $F_{m-1}(x)$.

Here we only briefly review the math in gradient tree boosting algorithm even though XGBoost also supports linear regression as weak classifier. Gradient tree boosting algorithm uses the regression trees as weak learners and tree based models might be more suitable to catch the non-linear relationship than linear regression. The objective in XGBoost [4] is

$$L(\phi) = \sum_i \ell(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \text{ where } \Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2$$

T is the number of leaves in the tree. f_k is a regression tree with $f_k \in \mathcal{F}, \mathcal{F} = \{f(x) = w_q(x)\}$. w represents the score on the leaves and q is the structure of the tree. ℓ is a loss function between prediction and the output. To minimize L, at the t-th iteration, a new weak classifier f_t is added to the (t-1) prediction.

$$L(\phi) = \sum_i \ell(\hat{y}_i^{(t-1)} + f_t(x_i), y_i) + \sum_k \Omega(f_k)$$

After using the second order taylor expansion and removing the constant term, the objective becomes

$$\sum_{i=1}^{m}[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \Omega(f_t)$$

with g_i and h_i be the first and second derivative of $\ell(y_i, \hat{y_i}^{(t-1)})$. Define $I_j = \{i|q(x_i) = j\}$ to be the set of leaf j. The objective can be further simplified as:

$$\hat{L} = \sum_{i=1}^{n}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \gamma T$$

Taking the first derivative with respect to w, the estimated weight can be calculated by:

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda}$$

Putting w_j^* back to $\hat{L}$, the new $\hat{L}$ can be used as the scoring function to evaluate the tree with structure q.

CHAPTER 5

Analysis

5.1 Performance Measure

For classification task, accuracy is a common metric to use in general. However, when the dataset is unbalanced, accuracy might not be a good metric to assess the model performance. The definition of accuracy is $\frac{\text{TP} + \text{TN}}{\sum \text{Total Population}}$.The TP and TN are from the confusion matrix:

	Predicted Positive	Predicted Negative
Actual Positive	True Positive(TP)	False Negative(FN)
Actual Negative	False Positive(FP)	True Negative(TN)

Table 5.1: Example of Confusion Matrix

For the hard drive dataset that is used in the analysis, even if the model classifies every observation as from healthy hard drives, the accuracy will be over 90% because the healthy class is over 90% of the total population. For classification task focusing on predicting the minority class, accuracy does not seem to be very useful for evaluating the predictive ability of a model. Therefore, the precision, recall, F1-score and false positive rate are used to measure the performance instead. Precision measures how many of the samples are indeed positive among the samples that are classified as being positive, while recall measures how many positive samples have been correctly identified. F1-score can be considered as the weighted average of precision and recall. False positive rate measures how many of the negative samples are classified as being positive. The definition of these measures are provided below:

$$\begin{aligned}
\text{Precision} &= \frac{\text{TP}}{\text{TP + FP}} \\
\text{Recall} &= \frac{\text{TP}}{\text{TP + FN}} \\
\text{F1-score} &= 2\frac{\text{precision * recall}}{\text{precision + recall}} \\
\text{False positive rate} &= \frac{\text{FP}}{\text{FP + TN}}
\end{aligned}$$

5.2 Result Comparison

To predict whether a hard drive is going to fail, the predicted label for each row is matched with the serial number of the testing data in the original order. A hard drive is classified as unhealthy if any of its snapshot is classified as unhealthy. The number reported in the confusion matrix below is the number of hard drives, not the number of observations in the testing dataset.

The parameters used in the models are chosen by the grid search with a 5-fold cross validation for each set of parameters. The set of parameters with the lowest F1-score will be chosen. The models are trained on a 2.8 GHz Intel Core i5 Macbook Pro with 8 GB memory using python.

5.2.1 QDA Result

The priors are P(Y = -1) = 0.9962 and P(Y =1) = 0.0037 based on the proportion of instances in each class in the training dataset. Training QDA is relatively fast as there is no parameter tuning required.

	Predicted Healthy	Predicted Failed
Actual Healthy	31483	3251
Actual Failed	23	40

Table 5.2: Confusion Matrix using QDA

5.2.2 SVM Result

To handle the unbalanced data, more weights are put on the regularization parameter for the positive class. Here the class weight is 80 for the positive class. The best parameters are 20 iterations over the whole training dataset for the stochastic gradient descent and the regularization related parameter $\alpha = 0.015$. The learning rate used is the default rate in the implementation. The support vector is defined as $y_i(w^\top x_i + b) \leq 1$ for soft margin SVM. There are 354,867 support vectors, which is around 14.85% of the total training data points. Even though the percentage of support vectors is not very high, the actual number of support vectors is not small. This might suggest that there is some patterns in the data, but the boundary between class might not be very clear. It might explain why the precision is not very high for SVM. Also, we are using linear SVM here since stochastic gradient descent only supports linear SVM. Other non-linear kernels such as RBF might produce better predictive performance, but the training time will be over one day, which is not very ideal for large amount of data.

	Predicted Healthy	Predicted Failed
Actual Healthy	34650	84
Actual Failed	25	38

Table 5.3: Confusion Matrix using SVM

5.2.3 Neural Network Result

Here the response is converted to $y_i \in \{0, 1\}$ to match the notation. The parameters that are tuned are shown in table 5.4. The loss function used is the Cross-Entropy, which is given by

$$Loss(\hat{y}, y, W) = -y \ln \hat{y} - (1 - y) \ln (1 - \hat{y}) + \alpha ||W||_2^2$$

The activation function is the tanh function. Notice that there can be lots of combinations for the number of hidden layers and the number of units in each layer, and it is impossible to explore all of them. Even though adding more hidden layers might improve the prediction result slightly, having too many hidden layers will make overfitting more likely to happen. Also, having a larger number of neurons in the hidden layers might improve the prediction result, but it will increase the computation time a lot with the potential of overfitting. Therefore, a simpler model is preferred here unless the more complex model increases the prediction ability significantly.

Parameter	Value	Parameter	Value
learning rate	0.015	units in hidden layers	(100, 50)
α	0.001	batch size for mini-batch gradient descent	50

Table 5.4: Parameters Chosen for Neural Network

	Predicted Healthy	Predicted Failed
Actual Healthy	34701	33
Actual Failed	29	34

Table 5.5: Confusion Matrix using Neural Network

5.2.4 AdaBoost Result

The number of weak classifier in Adaboost is chosen to be 8 from the grid search.

	Predicted Healthy	Predicted Failed
Actual Healthy	34701	33
Actual Failed	31	32

Table 5.6: Confusion Matrix using Adaboost

To see how well Adaboost separates the observations in the testing set, we can use the decision score $\sum_i \alpha_i g_i(x)$ that is described in Chapter 4. The distribution of the decision score for each class is shown in figure 5.1. Samples with positive decision scores are classified as failed, while samples with negative scores are classified as healthy. The histogram is grouped by the true class label in the testing set. It seems there is some overlapping between the failed and healthy classes, but there are also some positive scores that can be easily separated from the negative class.

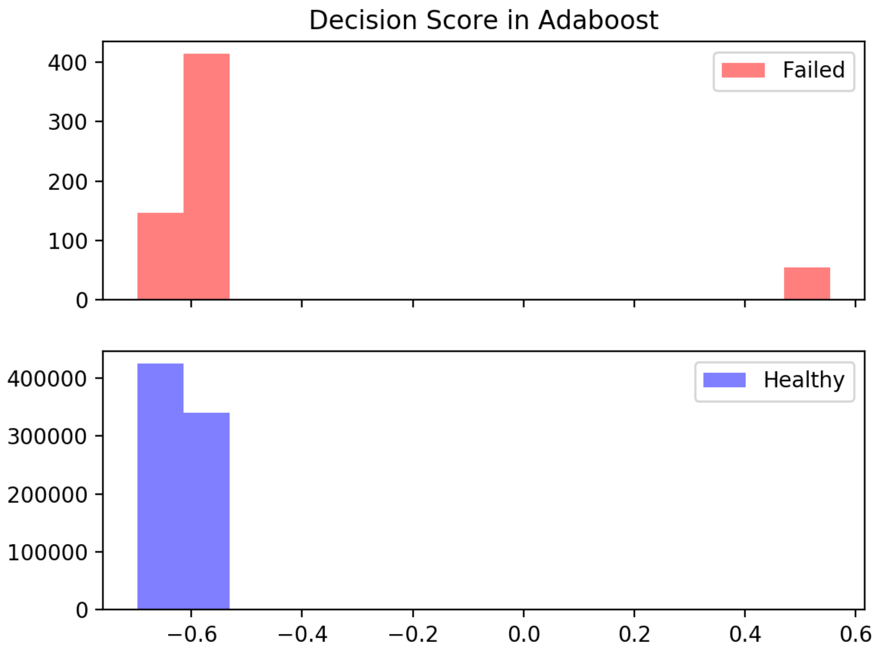

Figure 5.1: Decision Scores in Adaboost

5.2.5 XGBoost Result

The set of best parameters is shown in table 5.7. Here α and λ are the L1 and L2 regularization term for the leaf weight. Adding more weight to the positive class will help balance the positive and negative class weights.

Parameter	Value	Parameter	Value
learning rate	0.15	number of estimators	40
λ	1	α	1
tree depth	3	class weight for positive class	20

Table 5.7: Parameters Chosen for XGBoost

	Predicted Healthy	Predicted Failed
Actual Healthy	34678	56
Actual Failed	25	38

Table 5.8: Confusion Matrix using XGBoost

5.2.6 Ensemble by Majority Vote

After training the above models, an ensemble can be made by taking the majority vote of the prediction result. A hard drive is classified as failed if more than 2 classifiers predict the hard drive is going to fail. However, the majority vote classifier does not seem to improve the result a lot.

	Predicted Healthy	Predicted Failed
Actual Healthy	34680	54
Actual Failed	25	38

Table 5.9: Confusion Matrix using Majority Vote

5.2.7 Comparison

As shown in table 5.10 , QDA is the fastest while neural network has the longest total training time. Even though XGBoost has many parameters that are required to be tuned as well, the training time is still much less than neural network. The average training time seems to be acceptable for XGBoost given that there are over 2 million observations. On the other hand, Adaboost seems to be easier to tune as it does not have as many parameters as XGBoost.

Model	Number of points in Grid Search	Total Training Time	Average Training Time	Testing Time
QDA	N/A	5.99	N/A	1.09
SVM	24	1484.88	61.87	0.11
Neural Network	48	31413.6	654.45	2.98
Adaboost	9	1588.91	176.54	0.54
XGBoost	48	9276.13	193.25	1.00

Table 5.10: Run Time for Training and Testing (in seconds)

Model	Precision	Recall	F1-Score	False Positive Rate
QDA	1.22 %	63.49%	2.38%	9.36 %
SVM	31.15%	60.32%	41.08%	0.24%
Neural Network	50.75%	53.97%	52.31 %	0.10%
Adaboost	49.23 %	50.79%	50.00%	0.09%
XGBoost	40.42%	60.31%	48.40%	0.16%
Majority Vote	41.30%	60.31%	49.03%	0.16%

Table 5.11: Performance Metrics Comparison

Table 5.11 shows there seems to be some trade-off between precision and recall. QDA is an extreme case in the sense that it has the highest recall, but it has the lowest precision. However, it is not surprising to see that QDA has very poor predictive ability as the class density might not be normally distributed, and it will violate the assumption of QDA. The other models do not have many assumptions on the distribution of the dataset, which might be one of the reasons that the other models are performing better than QDA. XGBoost seems to perform better than the other methods when looking at the Recall and the F1-score. On the other hand, even though XGBoost is fast in general, it requires more efforts on parameter tuning and the process to perform the grid search to find the best set of parameters are usually long. SVM also seems to be a good classifier for the failure prediction task as it has relatively short computation time and fewer parameters to tune at 60.32% recall with only slightly higher false positive rate.

CHAPTER 6

Conclusion

In this thesis, several classification algorithms are applied to the S.M.A.R.T. hard drive dataset from a real world data center and failure prediction for the hard drives is performed. Since the dataset is highly imbalance, a good classifier for such failure detection problem should be able to deal with the imbalance. In addition, it seems better if the model does not have too many assumptions on the distribution of the dataset. The computation time, recall and false positive rate are some important factors to consider when comparing the models. It seems XGBoost provides the best prediction ability with 60.31% recall (or detection rate as used in other papers) and 0.16% false positive rate. In addition, its average computation time is fast for over 2 million rows of training data. Given that the false positive rate is low, the result shows XGBoost is able to provide useful prediction in reality.

For future improvement, it might be useful if we can develop a system that incorporates the differences between different models of hard drives. In addition, it might be interesting to identify which attributes are causing the abnormalities for the failed hard drives.

APPENDIX A

Descriptions of Attributes

Table A.1: S.M.A.R.T. attribute list (ATA) [7]

	Name	Description
smart_1_raw	Read Error Rate	Errors occurred while reading raw data from a disk .
smart_4_raw	Start Stop Count	Count of start/stop cycles of spindle
smart_5_raw	Reallocated Sector Count	Count of sectors moved to the spare area
smart_7_raw	Seek Error Rate	Rate of positioning errors of the read/write heads
smart_9_raw	Power-On Hours Count	Total time the drive is powered on
smart_12_raw	Power Cycle Count	Number of complete power on/off cycles
smart_183_raw	Runtime Bad Block	Vendor specific
smart_184_raw	End-to-End error	Vendor specific
smart_187_raw	Reported Uncorrect	Vendor specific
smart_188_raw	Command Timeout	Vendor specific
smart_189_raw	High Fly Writes	Vendor specific
smart_190_raw	Airflow Tempearture	The temperature of the air inside the hard disk housing.
smart_192_raw	Power Off Retract Count	Count of power off cycles

smart_193_raw	Load Cycle Count	Count of load cycles
smart_194_raw	Temperature	The temperature inside the hard disk housing.
smart_197_raw	Current Pending Sector	Count of unstable sectors
smart_198_raw	Off-Line Uncorrectable Sector Count	Count of uncorrectable errors when reading/writing
smart_199_raw	UDMA CRC Error Count	Count of errors during data transfer between disk and host
smart_240_raw	Head Flying Hours	Time spent during the positioning of the drive heads.
smart_241_raw	Total LBAs Written	Total count of LBAs written
smart_242_raw	Total LBAs Read	Total count of LBAs read
date		Time of the snapshot
failure		Binary indicator variable for the condition of hard drive

REFERENCES

[1] Léon Bottou. Learning with large datasets. *Tutorial of NIPS*, 2007.

[2] Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

[3] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[4] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[5] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[6] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[7] Hard Disk Sentinel. S.M.A.R.T. attribute list (ata). `http://www.hdsentinel.com/smart/smartattr.php`, May 2017.

[8] Backblaze Inc. Hard Drive SMART Stats. `https://www.backblaze.com/blog/hard-drive-smart-stats//`, May 2017.

[9] Backblaze Inc. Hard Drive Test Data. `https://www.backblaze.com/b2/hard-drive-test-data.html`, May 2017.

[10] Backblaze Inc. What SMART Stats Tell Us About Hard Drives. `https://www.backblaze.com/blog/what-smart-stats-indicate-hard-drive-failures/`, May 2017.

[11] Stephen Marsland. *Machine learning: an algorithmic perspective.* CRC press, 2015.

[12] Joseph F Murray, 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):783–816, 2005.

[13] Andrew Ng, Jiquan Ngiam, Chuan Yu Foo, Yifan Mai, and Caroline Suen. UFLDL Tutorial. `http://deeplearning.stanford.edu/wiki/index.php/UFLDL_Tutorial`, May 2017.

[14] Ying Zhao, Xiang Liu, Siqing Gan, and Weimin Zheng. Predicting disk failures with hmm-and hsmm-based approaches. In *Industrial Conference on Data Mining*, pages 390–404. Springer, 2010.

[15] Bingpeng Zhu, Gang Wang, Xiaoguang Liu, Dianming Hu, Sheng Lin, and Jingwei Ma. Proactive drive failure prediction for large scale storage systems. In *Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on*, pages 1–5. IEEE, 2013.