COMPARISON OF MACHINE LEARNING TECHNIQUES TO PREDICT MISSING CYANOBACTERIA DATA AND TROPHIC STATES OF LAKES

by

PRIYANKA LUTHRA

(Under the Direction of Lakshmish Ramaswamy)

ABSTRACT

Cyanobacteria are harmful blue green algae that deplete oxygen level of lakes and release toxins that adversely affect aquatic and human life. Hence, cyanobacteria monitoring is crucial. Scientists use satellite data to track cyanobacteria concentration in lakes. However, cloud cover and fog hinder satellite data collection, thereby creating a need to forecast the missing data values.

We investigate machine learning approaches on 10 years of satellite data of 99 lakes in South East US. We formulate the missing data problem as a classification problem, and we compare performance of various classifiers which utilize historical data of other lakes for classification. In addition, we conduct a spatio-temporal analysis wherein we leverage matrix factorization techniques to predict missing data. We achieve 88.9% accuracy with Random Forests for 5% data missing from target lake, and observe that Random Forest and k Nearest Neighbors are highly effective to combat missing data problem.

INDEX WORDS: Machine learning, Classification, Missing data, Cyanobacteria, Non-negative Matrix Factorization, kNN, Random Forests

COMPARISON OF MACHINE LEARNING TECHNIQUES TO PREDICT MISSING CYANOBACTERIA DATA AND TROPHIC STATES OF LAKES

by

PRIYANKA LUTHRA

B.TECH., JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, INDIA, 2015

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2017

COMPARISON OF MACHINE LEARNING TECHNIQUES TO PREDICT MISSING CYANOBACTERIA DATA AND TROPHIC STATES OF LAKES

by

PRIYANKA LUTHRA

DEDICATION

This one for you, mom.

ACKNOWLEDGEMENTS

I would like to thank Dr. Lakshmish Ramaswamy for his continued guidance and support without whom this work would not be possible. I would like to thank my committee members Dr. Rasheed and Dr. Bhandarkar who supported me throughout and introduced me to state of the art machine learning techniques and their applicability in various domains like computer vision. I would like to express my gratitude to Dr. Quinn for the projects which encouraged me to challenge myself and improve my learning curve. I would like to thank Dr. Mishra for his geography based insights at the research meetings. I would also like to thank my parents for their constant encouragement and support. I would like to thank all my professors and friends who supported me throughout my journey and believed in me. This research has been partially funded by the National Science Foundation (NSF) under grant number CCF-1442672. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author, and do not necessarily reflect the views of the NSF.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Cyanobacteria are harmful blue green photosynthetic algae which bloom in lakes. The name cyanobacteria is derived from the term “cyan” as its color resembles the color cyan, which is blue-green in appearance. Cyanobacteria thrive in lakes with slow moving warm water, which are exposed to plenty of sunlight and are rich in nutrients like Nitrogen and Phosphorus. Floods and farm run-off can intensify cyanobacteria growth, resulting in a blanket of scum referred to as CyanoHABs (Cyanobacteria Harmful algal blooms) that block sunlight from reaching plants below water surface, thereby degrading aquatic habitats and causing massive fish kills, bringing about both environmental and economic damage. They also consume the oxygen in water bodies thereby jeopardizing the survival of freshwater plants and animals.

Moreover, some cyanobacteria are responsible for production of potent cyanotoxins [1] [2] during the bloom season. In the United States, these blooms are prevalent in summer and early fall, although the Centers for Disease Control and Prevention reports that they can occur any time of year [3]. According to the Environment Protection Agency, the most common cyanotoxin producing varieties in US are Microcystis, Anabaena and Oscillatoria [2].

Cyanotoxin exposure can occur via inhalation, ingestion of contaminated water, direct contact or ingestion during water based recreational activities. This can lead to a myriad of health issues in humans like skin diseases, nausea, fever, headache, gastroenteritis, respiratory arrest and allergic response. Cyanotoxins can be classified into three categories based on their effects -neurotoxins (affect the nervous system), hepatotoxins (affect the liver), and dermatoxins (affect the skin) [4]. Life-threatening liver damage may occur in people exposed to cyanobacteria through contaminated dialysis water. In 1996, 116 patients were exposed to cyanotoxins at a renal dialysis clinic in Brazil by cyanotoxin contaminated intravenous dialysis procedure and experienced headache, nausea, blurred vision. One Hundred patients affected by cyanotoxins developed acute liver failure out of which 76 died [5]. Recent studies have associated cyanotoxins to neurodegenerative diseases like ALS, Alzheimer and Dementia in humans [6]. In addition, cyanobacteria blooms killed approximately 100 dogs in 13 states between 2002 and 2012, according to a 2013 CDC-led study in the journal Toxins [7]. Apart from health concerns, CyanoHABs may disrupt aquatic food webs by killing fish, birds and zooplanktons thus decreasing biodiversity and encouraging a new species selection. They also restrict recreational activities like swimming, fishing and pet-related activities which results in huge economical loss. Additionally, they pose problems for households that obtain drinking water from lakes and reservoirs. Hence, it is crucial to develop an accurate economical monitoring system to limit the adverse impacts of the bloom on aquatic, animal and human life.

Satellites are ideal for cyanobacteria monitoring as they are capable of observing remote environments, concealed features and events imperceptible by human eyesight. They cover large patches of area at once and this makes data collection economical for scientists who analyze ecological trends from the dataset such as monitoring amount of cyanobacteria in lakes or tracking a lake’s trophic state. Some of the other ecological monitoring techniques include aerial photography, in-situ sampling and sensor based approaches. While the above techniques are highly accurate and provide high scale resolution, they also require significant cost, knowledge and manpower to build and deploy sensors in each lake location. Furthermore, to leverage the in-situ sampling technique, visiting each lake periodically to monitor changes in algal bloom is necessary. Hence, these techniques aren't scalable. Moreover, adverse environmental conditions can affect the data collection done by the above techniques. In contrast, satellites prove to be an economical scalable solution as there is no cost associated with location specific deployment and they cover large areas of interest periodically. Furthermore, no prior authorization needs to be obtained while using the datasets.

A major shortcoming of satellite data collection is the presence of cloud cover and fog which obstructs the images of the lake from being seen clearly and hinders data collection by scientists. Hence, a significant amount of data is missing. In this context, a question that needs to be investigated is whether various machine learning approaches are effective in computing the missing data and how these approaches compare to each other. To address the above question, we analyzed 10 years of cyanobacteria data of 99 lakes spread over South-east US, collected by MERIS satellite during the bloom season (May – August).

1.1 Thesis Contributions:

The main contributions of this work include:

1. We formulate missing data problem of lakes as a supervised classification problem which takes historical and current data of other lakes into consideration while assigning a trophic state class to the target lake.

2. We compare different classification algorithms and evaluate them based on prediction error metrics to classify the lakes into trophic states.

3. We leverage a novel non-negative matrix factorization technique on simulated missing data which exploits spatio-temporal correlations between lakes to forecast missing data.

The rest of the thesis is structured as follows: Section 2 discusses the background of this project and related work, followed by the formal problem definition in Section 3. Section 4 involves a comprehensive review of various classification methodologies leveraged in this study to predict the missing data. This is followed by section 5 which provides details of dataset and gives a comparative evaluation of various classifiers followed by results of this work. The last section provides conclusion and future work.

CHAPTER 2

BACKGROUND

2.1 Cyanobacteria and Trophic States:

Cyanobacteria blooms in lakes predominantly during May to August. It forms a scum on water surface which blocks oxygen for aquatic plants thereby negatively impacting biodiversity. It is also related to a myriad of health issues as discussed in introduction section. A lake becomes susceptible to cyanobacteria growth if it undergoes eutrophication. Hence, it is crucial to determine a lake's trophic state.

Eutrophication, also referred as Hypertrophication, is the enrichment of a water body by overabundance of nutrients. This process promotes harmful algal and phytoplankton growth and increases the biomass load which in turn reduces the oxygen levels of the lake. Decomposition of algae consumes oxygen, which reduces the concentration of dissolved oxygen and this process is referred as Biochemical oxygen demand (BOD) [8]. Insufficient oxygen levels lead to fatalities of fish and marine life thereby adversely impacting biodiversity. The primary factors responsible for eutrophication include improper disposal of phosphorus and nitrogen rich detergents, fertilizers and sewage into water bodies. A lake can be naturally eutrophic if it is situated in nutrient rich region. High phosphorus levels in lake have been linked to eutrophication [9]. Ecological effects of eutrophication include adverse effects to biodiversity, decrease in water quality, growth of toxic species of algae and fish fatalities.

Based on the level of eutrophication due to amount of chlorophyll-a, phosphorus content and secchi depth [10] a lake is generally classified into one of the four categories - *Oligotrophic*, *Mesotrophic*, *Eutrophic* or *Hyperoligotrophic* (*Hypereutrophic)*. For our experiments, we are majorly concerned about the three classes – oligotrophic, eutrophic and hypereutrophic. Table 1 demonstrates how the chlorophyll-a values are mapped to trophic classes [11].

Table 1: Relationship between chlorophyll-a and trophic states

Chlorophyll-a concentration (mg m^{-3})	Lake's Trophic State
56 - 155+	Hypereutrophic
20-56	Eutrophic
0-20	Oligotrophic

Large deep lakes with low nutrient content, which are relatively clear and have low algae production fall into oligotrophic category. These lakes have high levels of oxygen and support various species of fish which require well oxygenated water. They are referred to as unproductive lakes due to insufficient nutrient and plant growth. Generally, such lakes are found in cold areas having resistant igneous rocks (especially granitic bedrock) [12] [13].

Lakes or ponds with high amounts of nitrogen and phosphorus, and an abundance of aquatic life and algal blooms fall into the eutrophic category. These high productivity lakes tend to have inadequate oxygen levels and are covered with murky scum like algal blooms, which adversely impact fishes and other marine life. These lakes have high chances of having cyanobacteria blooms due to high amounts of nutrients like nitrogen and phosphorus.

Extremely nutrient rich lakes with low transparency, characterized by presence of frequent algal blooms fall in Hypereutrophic class. Hypereutrophic lakes contain more than 40 microgram/liter of chlorophyll and more than 100gram/liter of phosphorus [13]. These lakes are most susceptible to frequent cyanobacteria HAB blooms and have dead zones below the surface due to a severe lack of oxygen.

An oligotrophic lake can convert into a hypereutrophic lake due to improper disposal of phosphorus and nitrogen rich detergents, fertilizers and sewage into water bodies [13].

2.2 CyanoTracker Project:

This work is a part of CyanoTracker project [14] at University of Georgia. In the CyanoTracker project, an early warning system which tracks cyanobacteria blooms across US is under implementation. This project involves development of a multi cloud framework that integrates crowd sourced social media observations, remote sensing measurements via both satellite and sensors, and multimedia data. Figure 1 represents the high-level architecture of CyanoTracker with three major components, namely, text analytics engine, satellite and sensor analytics engine and the image analytics engine.

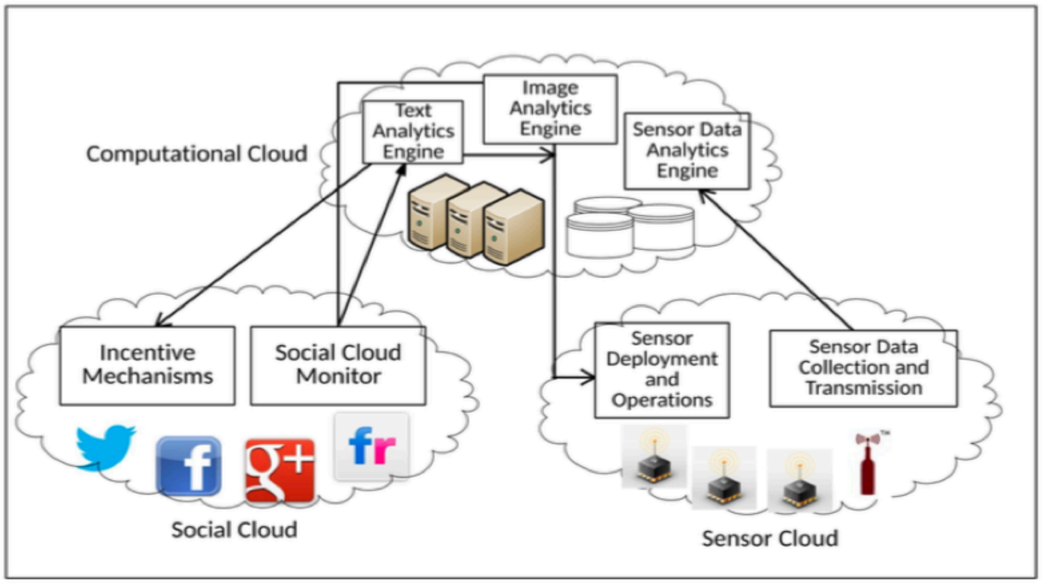


Figure 1: CyanoTracker Project [11]

The text analytics engine aggregates data about cyanobacteria blooms from social media platforms like twitter with the news and journal articles published about the HABs. This crowd sourcing based analytics engine leverages machine learning and natural language processing to locate high algal bloom regions and understand the health and ecological impacts of algal blooms in the US.

CyanoTracker android app is another platform which aims to collect crowdsourced image data. A user can upload images of algal blooms present in lakes in their vicinity and the blooms get represented on the project website. This app is incentivized to encourage user participation.

Sensor and satellite data analytics engine forms the core of the project. Sensors have been deployed at four lakes in Georgia. These Raspberry Pi based sensors use Oceanoptics hyperspectral sensor to collect algal spectra from lakes wirelessly. They are energy efficient and economical and in the future, can be deployed at other lakes. The satellite data powers the image analytics engine as well as the trophic state detection system. The satellite images and google earth images are aggregated and analyzed to predict whether an algal bloom is cyanobacteria or not. The satellite datasets are also used to predict the trophic states and cyanobacteria concentration of lakes within South East US.

Results from all of the above engines, are aggregated and visualized on the Cyanotracker website, which in turn publishes cyanobacteria occurrence warnings to the social media platforms.

2.3 Related Work:

Missing data problem spans across various domains. Data incompleteness problem may arise due to sensor errors, human errors, incompatible data formats while storing data in a database, restricted access to data and sensor failures. Data quality is a chief concern in machine learning as we need data to gather insights for knowledge discovery. Hence, there is a need to predict missing data while being careful not to introduce bias.

Missing data randomness is divided into three categories by Little etal. - Missing completely at random, Missing at Random and Not missing at random [15]. To gather insights from a dataset, missing data can be handled in following ways. The easiest method is to ignore missing data by discarding all the instances containing missing data values or discarding attributes which have a lot of missing values. Another alternative is to estimate parameters using Maximum Likelihood Estimation techniques. Third technique referred to as Imputation consists of exploiting data of other instances to fill the missing values.

Some of the widely used imputation techniques include: Case substitution, Mean or mode imputation, Hot deck or Cold deck technique and Predictive modelling.

Case substitution technique replaces an instance with missing value with a similar complete instance from the dataset. Mean or mode imputation substitute the missing value by the mean or mode of all known attribute values. Hot deck and cold deck techniques use clustering methods to fill missing data values. Predictive modelling technique builds a predictive model which considers the missing attribute value as target attribute and uses remaining attributes as inputs to the model. It exploits the correlation between attributes to fill missing data [16].

Predictive modelling using machine learning and matrix factorization approaches have been leveraged across different domains for solving missing data problem. Nanni etal. leverage expectation maximization alongwith SVM based ensembles for missing data prediction in medical context [17]. Fernández-Delgado etal. compare 179 machine learning approaches on UCI datasets [18]. Matrix factorization techniques like SVD have been used to predict missing data but their downside is that the resulting matrix has negative entries, making result interpretation difficult [19]. Recently, techniques like Non-negative matrix factorization(NMF) have been used for data incompleteness problem as they construct non-negative part based representation of the original matrix. Dmitry Tsarev etal. used NMF to estimate missing data of employee's access needs for computer security [20]. Yi, Xiuwen etal. [21] used matrix factorization techniques like NMF to predict missing geo-sensory data for Beijing air quality monitoring.

In context of cyanobacteria, Hollister etal. used random forest approach to estimate a lake's trophic state [22]. Boddula etal. [11] used dynamic time warping with clustering and regression to estimate algal severity and multilayer perceptron technique to predict a lake's trophic state. No existing work to our knowledge, has provided a comprehensive comparison of various machine learning techniques for trophic state prediction. Hence, we explore various machine learning techniques for missing data imputation in context of cyanobacteria prediction and leverage matrix factorization techniques to predict algal severity.

CHAPTER 3

OVERVIEW

Cyanobacteria blooms are harmful algal blooms that produce cyanotoxins which is harmful for human life as well as biodiversity. Satellite monitoring is a predominant method to monitor the CyanoHABs. A major shortcoming of this method is its inability to capture algal data in the presence of cloud cover. This leads to significant amount of missing data values. Hence, there is a need to address this problem. This chapter provides a formal definition of the data incompleteness problem and some simple solutions to address it. It also outlines the shortcomings of using the naïve approaches for missing data estimation in context of cyanobacteria problem.

3.1 Problem Formulation – Trophic State:

We obtain satellite data for cyanobacteria and trophic state monitoring. On an average, in 10 years of data collected for experiments, the lakes were cloud covered 2/3rd of times thus inhibiting timely monitoring of cyanobacteria blooms and various lake related monitoring programs [11]. A question that needs to be explored is whether machine learning approaches can overcome missing data problem by exploiting inter-lake correlations. To answer this question, we model trophic state of lake i at timestamp t as follows:

$$TS_{i,t} \in [O, E, H]$$

Lake i at timestamp t can belong to any one of the trophic states – Oligotrophic(O), Eutrophic(E) or Hypereutrophic (H). Time series data for Lake i with missing data can be denoted as follows:

$$TS_{i,t} = \{TS_{i,1}, TS_{i,2}, NA, \dots TS_{i,m}, \dots NA, TS_{i,n}\}$$

The problem is to estimate the missing trophic state values of a lake i at timestamp t denoted by NA.

3.1.1 Simple Solutions:

A simple solution is to estimate trophic state at missing timestamp t by assigning the mode of the all observations at similar timestamp last year or assigning mode of the trophic states which were available at timestamp t-1. Another simple solution is to use the same lake's historical data to predict the missing data.

However, the above suggested solutions are naïve solutions as bloom conditions can change radically due to changing environmental conditions. For instance, heavy rains, rapid increase in temperature, improper sewage disposal can significantly impact bloom conditions. Hence temporal algorithms based only on historical data do not provide good results in this context. [11]

To resolve the data incompleteness problem and predict trophic state of a lake, we model it as a supervised learning classification task. For classification, we need to provide training samples to classifier. Training set contains instance space X and label space Y such that ith training instance in X at a given time t assuming target lake is jth lake:

$$\langle x_i , y_i \rangle$$

where $x_i = \langle TS_{1,t,}, TS_{2,t}, \ldots TS_{j-1,t} \rangle$ and $y_i = \langle TS_{j,t} \rangle$.

To predict the trophic label of target lake at a given timestamp say lake j, a target function which models pattern between input x and label y needs to be learned, such that

$$y = f(x)$$

where $y \in Y$, Y is the set of trophic states [O, E, H] and $x \in X$. The target function is unknown and classifiers try to emerge at a hypothesis function g(x) which closely approximates f(x) and outputs a trophic state label for any given instance $x_i \in X$. We leverage various classifiers and compare their performance in this work for resolving the data incompleteness problem as discussed in detail in next section.

3.2 Problem Formulation – Algal Severity:

Cyanobacteria use chlorophyll-a for photosynthesis. Hence, chlorophyll-a concentration can determine amount of cyanobacteria activity in a lake. To estimate algal severity, chlorophyll-a concentration of a lake is determined by using Normalized Difference Chlorophyll Index(NDCI) algorithm [23]. NDCI values vary from -1 (absence) to +1 (severe) which indicate amount of chlorophyll-a, thereby indicating presence and absence of algal blooms. Depending on chlorophyll-a concentration of a lake, bloom severity is categorized in three classes- Low, Medium and High as depicted in Table 2. [11] [23]

Table 2: Relationship between NDCI values, Chlorophyll-a and Bloom Severity

NDCI range	Chlorophyll-a concentration (mg m^{-3})	Algal Severity
-1 to 0	<16	Low Bloom
0 to 0.4	16-50	Medium Bloom
0.4 to 1	>50	High Bloom

Boddula etal [11] formulate the problem of algal severity as follows. For a Lake i, at timestamp t, the algal severity of lake (percentage of lake pixels) in terms of low, medium and high bloom concentration can be denoted as:

$$L^{k,algal} = \{L^k_{Low}, L^k_{Med}, L^k_{High}\} \text{ [11]}$$

Representing missing data as NA, time series of lake L^k is represented as follows:

$$L^{k,ts} = \{L^{k,algal}_1, L^{k,algal}_{,2}, NA^{k,algal}_3, \ldots, NA^{k,algal}_{t-1}, L^{k,algal}_t\}$$

A simple solution to estimate algal concentration using historical observations can be

$$L^{k,algal}_{no\_data,t} = \sum_{i=1}^{lag}(w_i * L^k_i)$$

where w_i represents the weight assigned to a lake at a given lag i. This constitutes a naïve solution as well due to rapidly changing bloom conditions as discussed above.

Hence, there is a need to develop a spatio-temporal solution which exploits inter-lake and intra-lake correlations to forecast algal bloom severity and predict trophic states of lake for effective cyanobacteria monitoring. We use Non-negative matrix factorization technique to solve the above problem.

CHAPTER 4

MACHINE LEARNING APPROACHES

In previous section, we formulate missing trophic state problem as a supervised classification problem. In this chapter, we discuss various machine learning approaches explored in this study for trophic state prediction. We also discuss the spatio-temporal non-negative matrix factorization approach which we use to estimate the algal severity in detail.

4.1 Logistic Regression:

Logistic regression is a supervised classification method developed by statistician David Cox in 1958 [24] [25]. Also known as *Logit Regression* or *Logit Model*, it is a statistical model used to estimate the probability of an event occurring based on one or more predictors(features). It assumes that the dependent variable is categorical. When the dependent/target variable can take only two values, we use Binary Logistic regression. If the target variable has multiple categorical values, Multinomial Logistic regression is used.

Logistic Regression uses the logistic function to find a model that fits the data points. Logistic function is based on concept of odd ratios to calculate probabilities. Given a probability p that input x belongs to the class y=1, odds are computed as follows:

$$odds = \frac{p(y = 1/x)}{1 - p(y = 1/x\ )}$$

The natural logarithm of the odds ratio [26] is taken to compute logit function:

$$logit(P(x)) = \ln\left(\frac{p(y=1/x)}{1-p(y=1/x\,)}\right)$$

The value of logit function reaches infinity as p approaches 1 and reaches negative infinity as p approaches 0. Probabilities are mapped to real numbers using logit function, as opposed to logistic function which takes real numbers as input values and maps them to values in the range [0, 1]. The logistic function gives an 'S' shaped curve to model the data. The curve is restricted between 0 and 1. The logistic function is the inverse of the logit function [24]. It predicts the conditional probability that a certain sample belongs to a class.

Logistic regression assumes that logit of the probability p is linear with respect to x, so the logit can be defined as:

$$logit(P(x)) = w_o + w_1x_1 + w_2x_2 + .. + w_mx_m = w^Tx$$

where $x_1 \ldots\ldots x_m$are features, $w_1 \ldots\ldots.. w_m$are the weights/regression coefficients associated with the corresponding features. The regression coefficients are usually estimated using maximum likelihood estimation. Using the two equations together we obtain the logistic function:

$$P(y = 1/x\,) = \sigma(w^Tx) = \frac{1}{1 + e^{-(w^Tx)}}$$

This final equation is the logistic curve for Logistic regression. It models the non-linear relationship between x (training instance) and y (target value) with an 'S'-like curve for the probabilities that event occurs. The graph of Logistic function is shown in Figure 2.

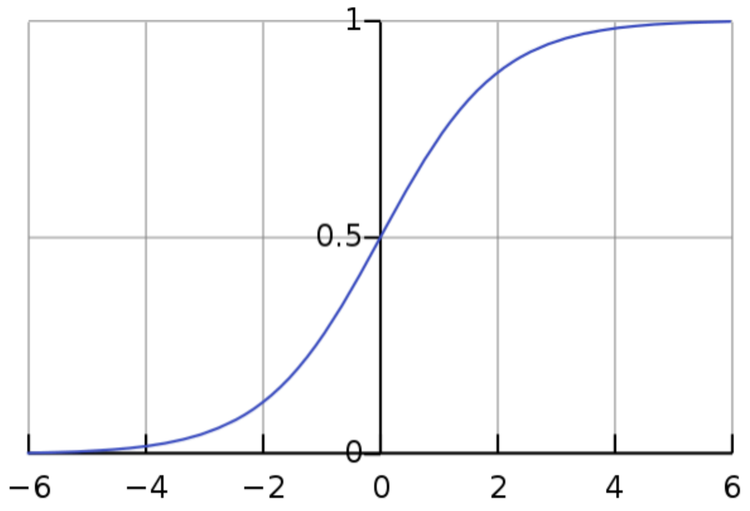


Figure 2: Standard logistic function [24]

The logistic regression models are categorized based on the number of target classes. They use sigmoid function for binary classification and softmax function for multinomial classification to predict target class. Logistic regression assumes that the target variable is binary. In our application, the target variable takes three values, hence we use One-vs-Rest classification strategy. In One-vs-Rest logistic regression (OVR) a separate model is trained for each class predicted whether an observation is that class or not (thus making it a binary classification problem), which assumes that each binary classification problem is independent.

4.2 Support Vector Machines (SVM):

Support Vector Machine(SVM) is a supervised classification technique which finds the hyperplane that best divides a dataset into 2 classes. It selects the Maximum margin hyperplane which gives maximum separation between the nearest training point of either class and hyperplane (this distance is called margin). Support vectors are the coordinates of instances closest to maximum margin hyperplane. They can be used to specify the margin, thereby making SVMs more memory efficient as the need to store the entire dataset is redundant. Each class has atleast one support vector. The distance between support vectors of a class and maximum margin hyperplane is the same. Figure 3 illustrates hyperplane and maximum margin for SVM.

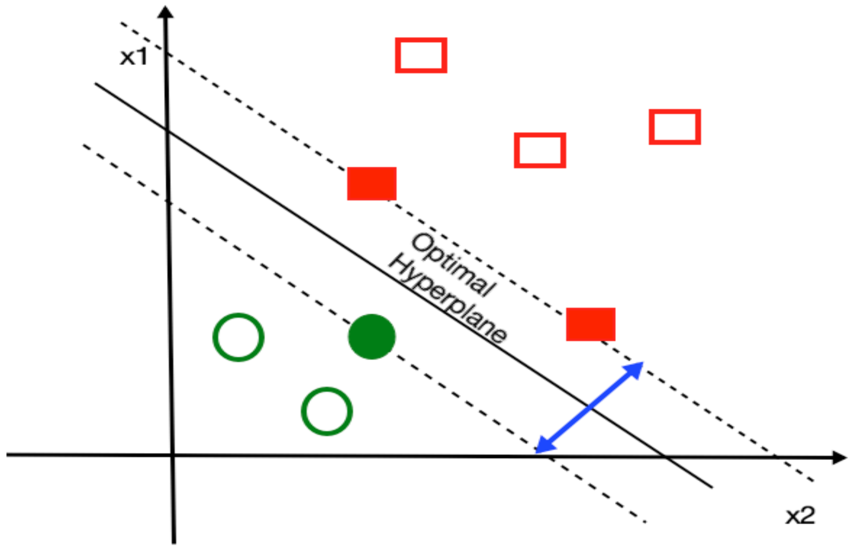


Figure 3: Hyperplane and Maximum margin in SVM. Maximum margin hyperplane is depicted in blue. Optimal hyperplane is the solid line. The filled boxes/circles are support vectors.

Hyperplanes with higher margins have less chance of misclassification and hence SVMs are robust to outliers. For linearly separable data, hyperplanes can be determined using quadratic optimization.

For non-linear classification problems, wherein a linear decision boundary cannot separate the classes in the original feature space, SVM maps the dataset to higher dimensions where it exhibits linear separability [27]. Training set is mapped to a higher dimension feature space F using non-linear function $\varphi : X \rightarrow F$, such that the discrimination function is defined as [28]:

$$f(x) = < w.\varphi(x) > +b$$

On projecting the decision boundary back to original feature space, a non-linear decision boundary is obtained. This mapping to higher dimension leads to increase in degrees of freedom while computing dot product (large number of coefficients to be computed), thereby increasing time complexity. It can also result in overfitting if number of coefficients is larger than number of training samples.

To address this issue, SVM utilizes *kernel trick*, which uses a kernel function which implicitly computes dot product in the original feature space to map the data in a higher dimension feature space before the non-linear mapping is performed. This does not utilize any additional memory and has limited impact on the computation time thereby resolving the time complexity issue. A function needs to satisfy Mercer's theorem to be a kernel function such that [28] [29]:

$$k(x,z) = < \varphi(x).\varphi(z) >$$

where $\phi(x)$ belongs to the Hilbert space. Different kernel functions like linear kernel, polynomial kernel, radial basis function or sigmoid can be used to map dataset in expanded feature space efficiently.

In addition, SVMs are highly unlikely to overfit as the maximum margin hyperplane is relatively stable and changing any training instances apart from support vectors will not affect it. Moreover, SVMs can be generalized to cases where training data is not separable by placing an upper bound on the coefficient alpha.

In case of multi-class classification using SVM, two strategies can be employed -One vs rest strategy and One vs One approach. Both approaches learn different number of classifiers and hence output different decision boundaries. In One vs all strategy, one classifier per class is trained by assuming the labels of that class as positive and rest of the labels as negative. If the total number of distinct classes is N, then N classifiers have to be learnt. The disadvantage of this approach is that the training samples used for building the N classifiers can be highly unbalanced.

In contrast, in the latter approach, a separate classifier is trained for each unique pair of labels. This leads to N(N−1)/2 classifiers. This is much less sensitive to the problems of imbalanced datasets but is much more computationally expensive. We follow the former approach for our experiments.

4.3 Random Forests:

Random forest is an ensemble classification method in which several decision trees are constructed to classify an unseen observation. Each tree, which is constructed, assigns a class to the new observation and the class decided by the majority of the trees is selected as the class of the new observation. This is illustrated in Figure 4.

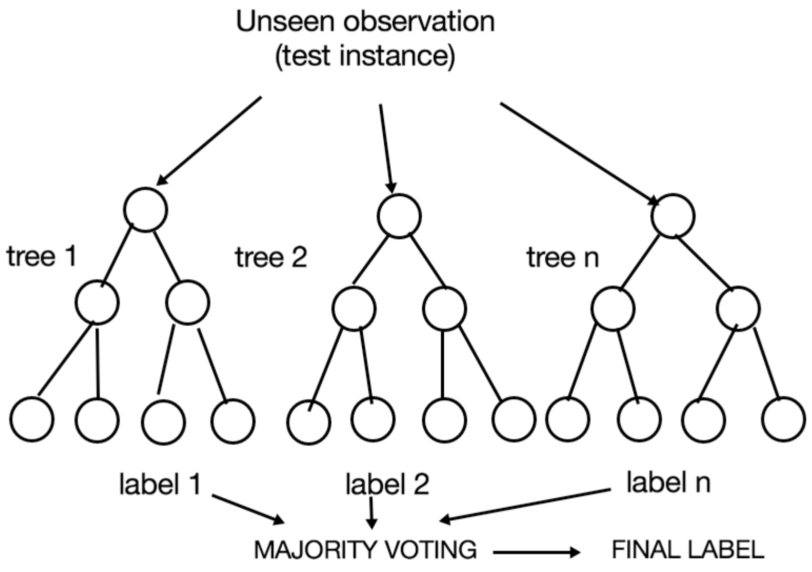


Figure 4: Majority Voting for assigning label to new instance in Random Forest

This method involves two stages of randomization – *Random Subspace* method and *Bootstrap* method. The former involves randomizing the selection of input features taken into consideration while building each tree. If the original dataset has M features, then a constant integer (constant for entire forest) m<<M is selected such that m variables are randomly chosen from M [30]. The tree is split on the best split from the resulting m variables rather than choosing the best split on the entire features. This makes random forests computationally efficient and highly scalable as the computational efficiency is now a function of m instead of all of the features M.

Deciding the value of m requires a tradeoff between the individual tree's classification accuracy and inter-tree correlation. Inter-tree correlation is directly proportional to the random forest's error rate but inversely proportional to individual tree's classification accuracy. Traditionally, the optimal value of m is square root of M (total no of features) [30] [31].

The latter method of introducing randomization is called Bootstrap method. If the original dataset has M training instances, M training samples are sampled from it with replacement, which form the training set (bootstrap sample) for the current tree. When the training set for the current tree is drawn by sampling with replacement, about 33% of the samples are left out of the bootstrap sample [30]. This is referred to as out-of-bag data. This out-of-bag data gives an unbiased classification error and helps estimate the importance of variables for classification.

The error rate of random forests is estimated using Out of Bag (OOB) Error. Each tree in the forest is created using a different bootstrap sample as training set, from which 33% of observations are left out. To estimate the OOB error, each instance of the original dataset is classified using the majority voting by the trees that did not include this instance in their training samples. The proportion of times that class obtained is not equal to the true class of that observation, averaged over all the instances in training set, is the OOB error rate. This error metric makes the need of cross validation to get unbiased estimate of test error redundant and ensures that random forests do not overfit. OOB error metric is considered to be accurate as using a separate test set of the same size as the original dataset [30].

The random forest implementation used for this study differs slightly from the original paper as it aggregates classifier results by averaging their probabilistic prediction, instead of using the majority vote method [32].

5.4 k Nearest Neighbors:

k Nearest Neighbor (kNN) is a supervised non-parametric classification method. The non-parametric nature of kNN implies that it does not presume any hypothesis about underlying data distributions. It is also an instance based/lazy learning method which stores training instances instead of creating a model from the training data.

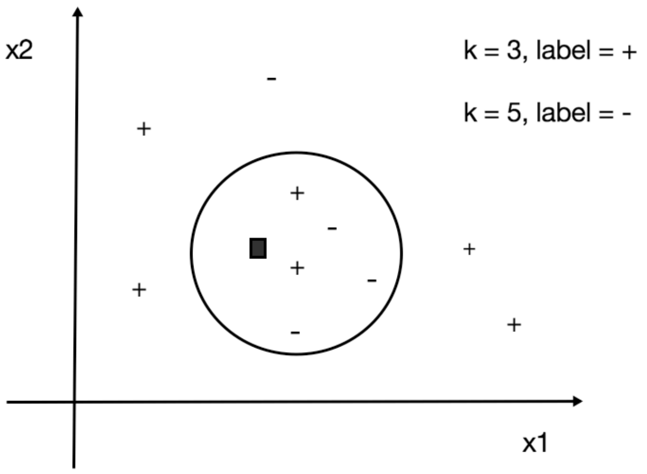


Figure 5: Assigning class to the new instance(square) in kNN with different k values.

To classify a new unseen observation, kNN finds k most similar instances from training set. In basic version of kNN, each of the k nearest observations from training set cast a vote for the class of new observation and a class is assigned by majority voting assuming uniform weight of each vote. Figure 5 illustrates how label is assigned by majority voting on varying k.

To assign more weight to classes voted by closer/more similar neighbors, distance based weighting can be used. Euclidean distance is generally used but other distance metrics like Manhattan or Hamming can also be used for computation. Similarity is defined according to a distance metric between two data points. The Euclidean distance is computed as follows:

$$d(x, x') = \sqrt[2]{(x_1 - x'_1)^2 + \cdots + (x_n - x'_n)^2}$$

Here, x' is the new instance to be classified and x is a labelled instance in the training set. The distance of unseen observation is computed from each instance in training set. After which weighted votes of k nearest neighbors are used to assign a class to the new observation.

Choosing value of k, the hyperparameter is crucial. Generally, k is odd to prevent a tie. Large values of k lead to smooth decision boundary and suppress the effect of noise/outliers due to lower variance and higher bias. On the other hand, a small k makes the overall distribution of data obscure for the classifier and leads to jagged decision boundary due to lower bias and higher variance [33] [34].

In our experiments, we choose k by using cross validation and hyperparameter grid search which is explained in the experiments section. kNN has many shortcomings as it follows a "Minimal training but expensive testing" principle [34]. The cost of classifying an unseen observation during testing phase is high as all computations take place during testing. In addition, kNN has a high memory cost as it stores the entire training set for classification which further makes it unfit for dynamic web mining applications.

4.5 Multilayer Perceptron (MLP):

Multilayer perceptron(MLP) is a multilayer feed forward neural network used for supervised classification problems. It has three types of computational layers: input layer, hidden layer and output layer. Each layer is made of neurons and neurons in one layer are directly connected to the neurons in the next layer. Random weights are initialized for all connections in the network. The number of neurons in output layer is equivalent to number of target classes. Figure 6 shows the architecture of a multilayer perceptron with one hidden layer.

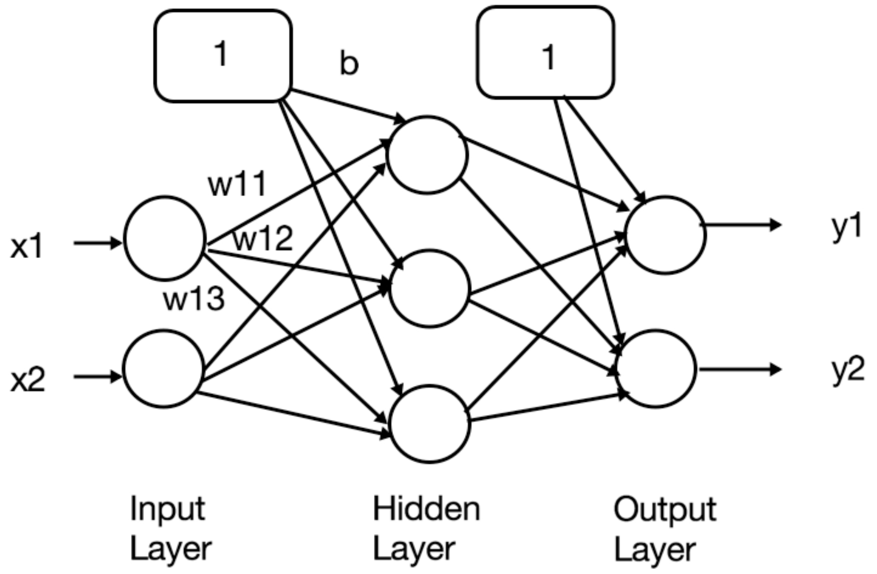


Figure 6: Multilayer Perceptron architecture with one hidden layer comprising 3 neurons. b represents bias in the figure.

MLPs are a network of simple perceptrons with a sigmoid activation function. Rosenblatt proposed simple perceptron model in 1958 [35]. The simple perceptron computes a single *output* from *inputs* by forming a linear combination according to its input *weights* and then applying nonlinear activation on the resulting sum to give output. MLP was developed to overcome the shortcoming of simple perceptron as it can classify data that is not linearly separable. In addition, it differs from simple perceptron as it uses a sigmoid activation function instead of a linear activation and have many stacked perceptron layers.

Moreover, a true perceptron performs binary classification while MLP can perform multi class classification and regression. MLPs can use a variety of activation functions like sigmoid, tanh etc. Generally sigmoid function is used as it is easily differentiable and can be backpropogated easily.

Inputs to the neural network should be numerical which can be achieved using label or one hot encoding discussed in next section. In addition, inputs must be scaled in a way such that each column has zero mean and standard deviation as 1. They can also be normalized by rescaling the inputs between 0 and 1.

MLPs are trained using gradient descent approach. Inputs are fed to input layer neurons which then get transferred to hidden layer neurons after computing weighted sum and applying activation function to compute hidden layer output. The hidden layer output is treated as inputs for neurons in output layer. The output is computed by taking a weighted sum of hidden layer outputs and applying activation. Hence, it is a feed forward network without any cycles or loops. This is referred to as forward pass of the neural network.

The output of the network is compared to the target output to compute error. The network learns and updates weights using backpropagation. The backpropagation algorithm is a supervised learning approach which utilizes gradient of loss function(error) to adjust the weights of the neurons. Its referred to as backpropagation [36] as error calculated at output layer is propagated backwards to hidden layers and input layer. This enables the network to update weights via Stochastic Gradient Descent (SGD) [37]. Generally stochastic gradient descent is preferred as one row of data is exposed to the network at a time as input instead of feeding the entire dataset to the network and then updating weights. This process is repeated for all the instances in training set and once all the instances are fed to the network, one epoch gets completed. An MLP can be trained for many epochs.

Online learning/Stochastic gradient descent refers to updating weights after calculating error for each training instance. On the other hand, weights can be adjusted after error is calculated for the entire training set resulting in batch learning which is stable. SGD is computationally faster than batch Gradient Descent (GD) as only one training sample or a subset of them are fed to the network before updating weights. Moreover, fewer number of training instances need to be stored in memory. It also converges faster and minimizes the error approximately to the minimum error achieved by GD. [37] It starts oscillating once it reaches the optimal minima. The amount by which weights are adjusted is controlled by a user defined hyperparameter called learning rate or step size.

Generally, we want our network to make large changes to weights in the initial epochs so that it converges faster and small changes in the later epochs once the network begins to converge, to fine tune the network. This can be controlled using Learning Rate Decay parameter, which decreases the learning rate as number of epochs increases.
MLP with many hidden layers have a non-convex loss function which implies that they have more than one local optimum. To ensure that the neural network is not stuck at a local optimum, another parameter called momentum, allows weights to change in the same direction even if the error does not decrease. MLP uses alpha parameter for regularization (L2 regularization) term to avoid overfitting by penalizing weights with large magnitudes.

Once the training process is over, the test set instances are classified by being passed as inputs through forward pass of the MLP. The network topology and final weights can be stored to classify future unseen instances. MLP requires tuning a number of hyperparameters such as the number of hidden neurons, layers, epochs and learning rate so that it can reach the optimum result.

4.6 Non-negative Matrix Factorization:

Non-negative matrix factorization (NMF) is a matrix factorization technique which assumes that the data and its components contain non-negative values. [38] It involves decomposition of a given (u x v) matrix V, into component rank-k matrix W (u x k) and H (k x v) such that

$$V \approx WH$$

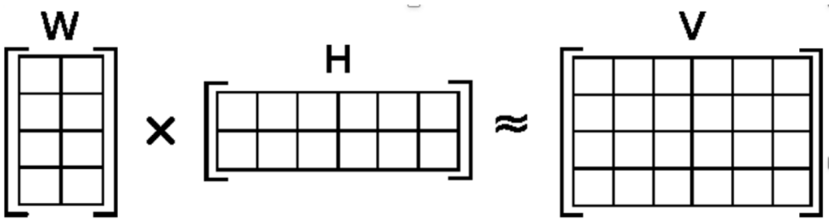


Figure 7: Non-negative matrix factorization [39]

Generally, the value of k is chosen to be less than minimum of m and n. Figure 7 shows the matrix decomposition. Each of the n columns of V can be built from k columns of W, thereby making it the basis vector. H is known as coefficient matrix as columns of H give weights associated with each column of W (basis vector).

It optimizes the distance d between X and the matrix product WH. Generally, the distance function used is the squared Frobenius norm which is based on L2 norm: [39] [40]

$$\min \| V - WH \|_{Fro}^{2}$$

such that W, H ≥ 0.

Data is compressed implicitly by NMF as component matrix are much smaller than the original matrix. Some of the applications of NMF include image recognition, text classification, recommendation systems. It has also been used to predict missing data values [21]. One approach is to initially insert 0s for missing data entries, then perform NMF, producing W and H. We then compute WH as our estimate of V, and now have estimates for the missing entries. Historically, NMF algorithms used the following methodologies – Alternating Least Square [41], Multiplicative update rules [40] [42], Gradient Descent [43] [44].

The advantage of NMF based missing data imputation, compared to other model based methods, is that it takes into account all the complete entries when imputing a single missing entry, which means it can capture both spatial and temporal dependencies among entries whereas a general missing value imputation algorithm usually models missing data by capturing either spatial or temporal dependencies.

The chief hyperparameter which needs to be tuned for NMF is rank k. This is done by randomly deleting some entries from original matrix V and then imputing these values by varying k. The value of k that gives minimum error is optimal as it provides correct decomposition to recover missing entries.

CHAPTER 5

COMPARITIVE STUDY

5.1 Dataset:

The satellite dataset is collected from European Space Agency(ESA) website. We use data collected by Boddula etal. [11] which contains 10 years of MERIS data from 2002-2011 for 5 states in South East US- Alabama, Georgia, Florida, South Carolina and Tennessee, after which it was decommissioned. As early summer and late Fall generally see high level of Cyanoblooms in US lakes, the data was collected from April to August.

MERIS has a spatial resolution of 300mx300m. Hence, lakes covering minimum area of 2.2 square km were selected so that minimum 20 pixels can be obtained from them. Studying 20 pixels of satellite image of a lake is equivalent to collecting 20 in-situ water samples from a lake [11].

Total data of 99 lakes was collected and analyzed in this study. The data was extracted from ESA website which provides MERIS satellite images in two resolutions – Full Resolution images(300mx300m) and Reduced resolution images(1.2kmx1.2km). We can also obtain different levels of MERIS data (0-3). Level 2 data was selected as it takes into account atmospheric correction. Upper limit of cloud cover was set to 50% and images covering region of interest were downloaded [11]. For each pixel, MERIS raster data provides information about 15 reflectance and 19 auxiliary bands. ArcPy was used to extract algal 2 band to generate a mask for water body. This ensured that we obtain the *Normalized Difference Chlorophyll Index*(NDCI) [14] data of lakes and it does not encompass NDCI data of land. We obtained reflectance band 7 (665 nm) and band 9(708.75 nm) by applying water mask. *Chlorophyll-a* is found in all cyanobacteria species as they utilize it for photosynthesis. As earlier discussed, amount of cyanobacteria bloom in a lake is strongly correlated with Chlorophyll-a concentration of a lake. NDCI algorithm is used to assess the amount of chlorophyll-a. It is suitable to be used with satellites providing multi spectral bands data to monitor presence of CyanoHABs in water. Finally, the NDCI algorithm is applied which helps to classify bloom severity into 3 classes – Low, medium and High as depicted in Table 2 [23].

Shapefiles for hydrography data for the five states were downloaded from National Hydrography Dataset (USGS). These shape files enabled us to extract data of 99 lakes within the selected states. Frequency distribution of pixels in three ordinal classes was analyzed for each of the lake. Thus, the resulting dataset consisted of 3 different pixel counts for each severity class of each lake. Table 3 provides details about number of lakes within the five states along with minimum and maximum pixel count viewed from MERIS satellite images [11].

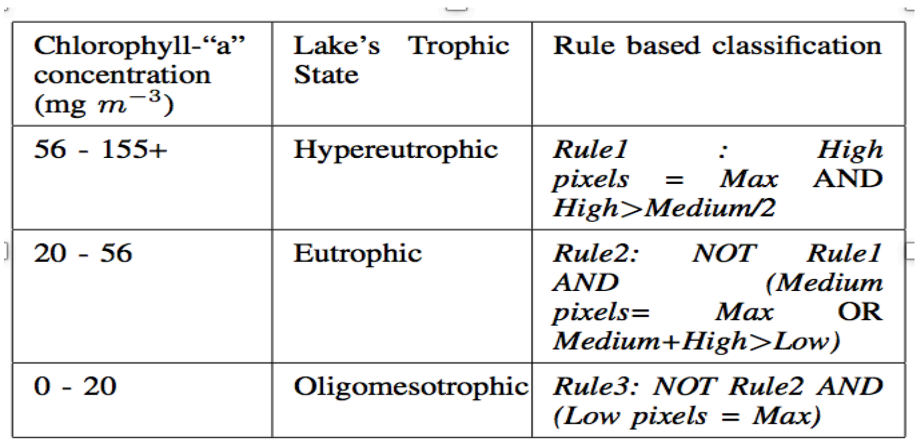
Table 3: Dataset summary with pixel distribution

State	No: of Lakes	Max/ Min pixels	Avg pixels
Alabama	20	3033/143	999
Florida	35	14506/139	879
Georgia and Tennessee	30	3033/128	859
South Carolina	14	3882/135	1221

We can obtain maximum 10 MERIS images for a lake per month. But due to complete or partial cloud cover, we extracted maximum number of visible pixels every month and assigned its pixel distribution to respective class for the month. Then this dataset was converted to a bi weekly dataset by averaging the pixel distribution of every pair of two consecutive months. We obtained the percentage distribution of pixels in each severity class by dividing the lake data by maximum visible pixels in 10-year time period for a lake. Hence, our dataset is a biweekly spatio-temporal data with percentage of algal bloom in three severity classes for each of the 99 lakes. This dataset is analyzed for the NMF experiment.

As discussed in the previous section, the trophic state of a lake is highly correlated to its chlorophyll-a values. As shown in Table 4, a rule based approach is suggested by the domain scientist to assign trophic states to lakes based on the Low, Medium and High severity pixel count [11]. Thus, our trophic file dataset contains monthly time-series classification of 99 lakes in one of the three states viz. Hypereutrophic, Eutrophic or Oligotrophic for 10 years.

Table 4: Trophic state classification rules [11]



Chlorophyll-"a" concentration (mg m^{-3})	Lake's Trophic State	Rule based classification
56 - 155+	Hypereutrophic	*Rule1 : High pixels = Max* AND *High>Medium/2*
20 - 56	Eutrophic	*Rule2: NOT Rule1 AND (Medium pixels= Max* OR *Medium+High>Low)*
0 - 20	Oligomesotrophic	*Rule3: NOT Rule2 AND (Low pixels = Max)*

5.2 Trophic State Prediction Comparison

5.2.1 Single Lake Data Missing Trophic State Prediction Methodology:

In this experiment, we assume that at a given time, data of any one random lake will be missing while data of other lakes will be present. We transform the categorical labels in trophic file dataset to numerical values so that it can be fed to classifier. Numerical encoding is necessary as classification algorithms and libraries used cannot handle features in string format [45]. Two approaches can be leveraged to map classes to numerical values namely Label encoding and One Hot Encoding. The Label encoding approach directly maps the categories to ordinal numerical values. One hot encoding approach provides binary mappings for the categories and does not retain ordinality [46] [47]. chief drawback of the latter approach is that storage size increases exponentially for high dimensional data with lot of non-numerical categories, thereby it is generally followed by a dimensionality reduction technique. As the latter approach does not preserve ordinality between categories and requires large storage space, it is unsuitable for this application. Hence, we use Label Encoding approach to map classes as follows- Oligotrophic as 0, Eutrophic as 1 and Hypereutrophic as 2.

After encoding the dataset, x% of lakes are assumed to be target lakes where a percentage of data is missing. In our experiments, we choose x as 5. We simulate missing data by withholding 5,10,15,20% of a lake's data from classifier which is equivalent to hiding 0.05,0.1,0.15,0.2% of entire dataset. We achieve this by splitting the dataset into training and testing sets. We ensure not to withhold consecutive data entries as our file contains monthly data values and withholding 5% of a lake's data is equivalent to hiding a year worth of observations, which greatly deteriorates classification accuracy. This happens because weather conditions can drastically change within a year and trophic state is dependent on a variety of phenomenon like amount of rainfall, weather conditions, farm run offs, industrial or agricultural activities in vicinity. An oligotrophic lake can convert to a hypereutrophic lake within a year and hence, our classifier is unable to predict such patterns.

We leverage various machine learning models like Logistic Regression, Support Vector Machines, Random Forests, K Nearest Neighbors and Multilayer Perceptron for supervised classification. These models have been discussed in detail in the previous section. We fit our models on training set, and evaluate its performance on unseen test set. We use the test dataset only once in order to avoid overfitting while computing the prediction-error metrics. Overfitting occurs when a model learns the detail and noise in the training data as concepts which adversely impacts its ability to classify new data as the model is unable to generalize on unseen instances. This results in high prediction-error on test set. Thus, to avoid overfitting, we use 10-fold cross-validation for model creation [48].

We utilize 10-fold cross validation to tune hyperparameters of our classifiers. Hyperparameters are the parameters of our estimators that are not directly learned from the training data but need to be optimized separately to improve classifier performance and achieve good generalization. We use GridSearch, which evaluates all possible parameter combination for a model and retains the best hyperparameters.

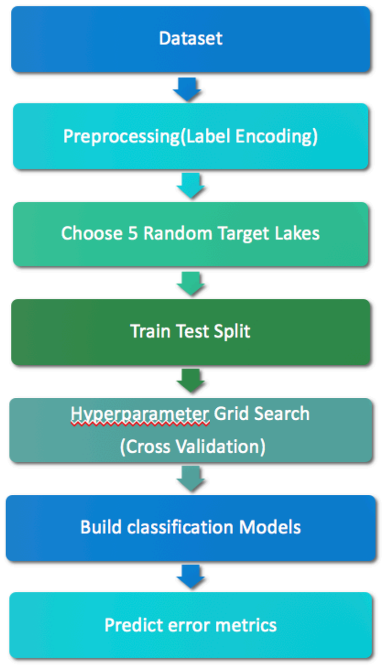

Dataset
Preprocessing(Label Encoding)
Choose 5 Random Target Lakes
Train Test Split
Hyperparameter Grid Search (Cross Validation)
Build classification Models
Predict error metrics

Figure 8: Trophic state classification pipeline

We illustrate our classification pipeline in Figure 8. We repeat the above process 100 times and report the average prediction accuracy of the classifiers in Table 5. Accuracy is percentage of correct predictions out of the total predictions [49].

$$\text{Accuracy} = \frac{No\ of\ Correct\ Prediction}{No\ of\ All\ predictions} * 100$$

Accuracy is insufficient to determine a classifier's performance as it is susceptible to Accuracy Paradox [50] as it presumes equal weight for all kind of errors. Hence, we also evaluate confusion matrix, average precision and recall metrics to compare the performance of our classifiers.

Precision or positive predictive value is the fraction of relevant instances among the retrieved instances, while recall is the fraction of relevant instances that have been retrieved over the total number of relevant instances [49] [51].

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

5.2.2 Large Scale Data Missing Trophic State Prediction Methodology:

In our previous experiment, we assume that data of any one random lake is missing while data of every other lake is present. We experiment on 5% of data of a given target lake which amounts to 0.05% (similarly 10,15 and 20% amount to 0.1%,0.15% and 0.2%) of entire dataset. Hence, there is a need to investigate how the classifiers perform when large amount of data is simultaneously missing from more than one lake.

To address this issue, we perform another experiment on our trophic file dataset. After the label encoding step, we simulate missing data by removing x% of data randomly from our entire dataset. We vary missing data percentage between 2% and 10% for our entire dataset. For each lake as target, we use a simple approach to substitute missing data of all other lakes, except for the target lake. We replace the missing values of other lakes by their mode. This is done as classifiers cannot handle missing data values.

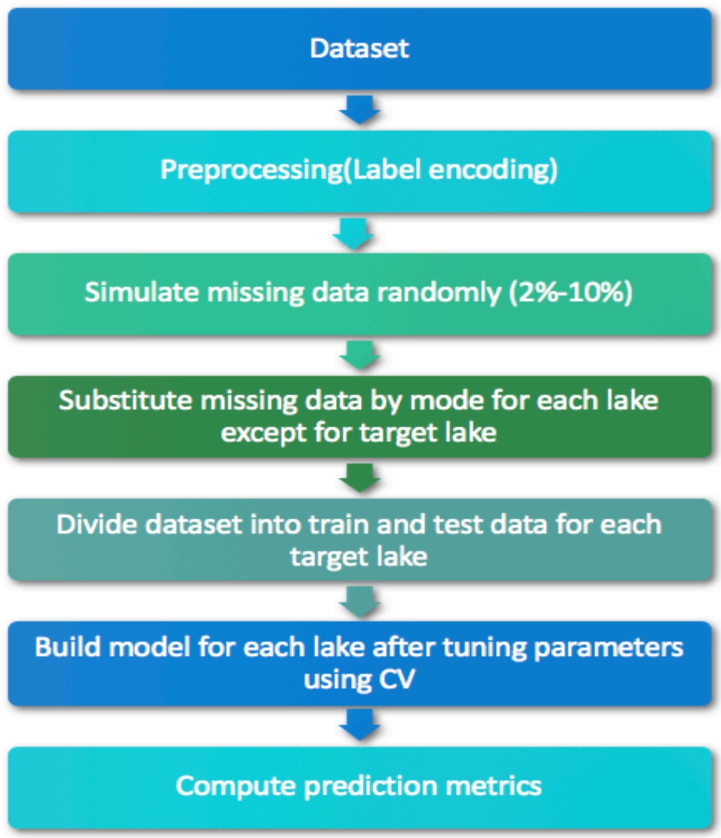

Dataset
Preprocessing(Label encoding)
Simulate missing data randomly (2%-10%)
Substitute missing data by mode for each lake except for target lake
Divide dataset into train and test data for each target lake
Build model for each lake after tuning parameters using CV
Compute prediction metrics

Figure 9: Trophic Dataset- Large scale data missing classification pipeline

We use instances which do not contain missing data for target lake, as our training set for the lake and rest of the instances are considered as test set. We then leverage machine learning algorithms discussed in previous section and build models with tuned parameters for each lake as target. After building the models, the missing data of each lake is predicted using the model built for the lake. We compare the predictions to ground truth and report the accuracy, precision and recall for two trials.

5.2.3 Single Lake Data Missing Prediction Results:

We evaluate and compare the classifiers based on their average accuracy. Table 5 and Figure 10 shows the trend of change in percentage of average accuracy for each classifier with change in percentage of missing data. As discussed earlier, the missing data percentage shown in the graph corresponds to missing data percentage of a single lake. This refers to the fact that 5% missing data of a single lake implies 0.05% of data is missing from entire dataset. Similarly, for this experiment, 10%,15% and 20% missing data of a lake corresponds to 0.1%,0.15% and 0.2% of data missing from entire dataset respectively. From Figure 10, we observe that Random Forest outperforms other machine learning approaches for trophic state classification. We hypothesize that Random forest works well as it performs implicit feature selection and provides pretty good feature relevance in addition to reducing bias [30] [53] [18]. The data does not need to come from a specific distribution due to the nonparametric nature of random forests. In addition, they work well with large numbers of predictors [52]. Moreover, random forest predictions are based upon a consensus of many models and not just a single model which may leads to better results. We observe that kNN, SVM and MLP techniques also perform well for the classification problem.

In addition, we note that Logistic Regression performs the worst. We hypothesize that Logistic Regression does not perform well because it is a generalized linear model and hence it finds linear decision surfaces and doesn't perform well for non-linear problems. In addition, it doesn't perform well in presence of high dimensional input feature space. Moreover, we hypothesize accuracy of Multilayer perceptron is low as it requires large amount of data for training and can get stuck in local minima.

Table 5: Average accuracy results for trophic state prediction. The average accuracy after performing 100 experiments on 5 random target lakes is reported.

METHODS	5%(0.05%)	10% (0.1%)	15%(0.15%)	20%(0.2%)
LR	82.83	76.2	76.22	74.18
SVM	83.77	78	76.1	76.13
RF	88.93	81.65	80.14	80.08
KNN	87.4	79.2	79.28	77.75
MLP	84.03	75.7	75.12	74.33

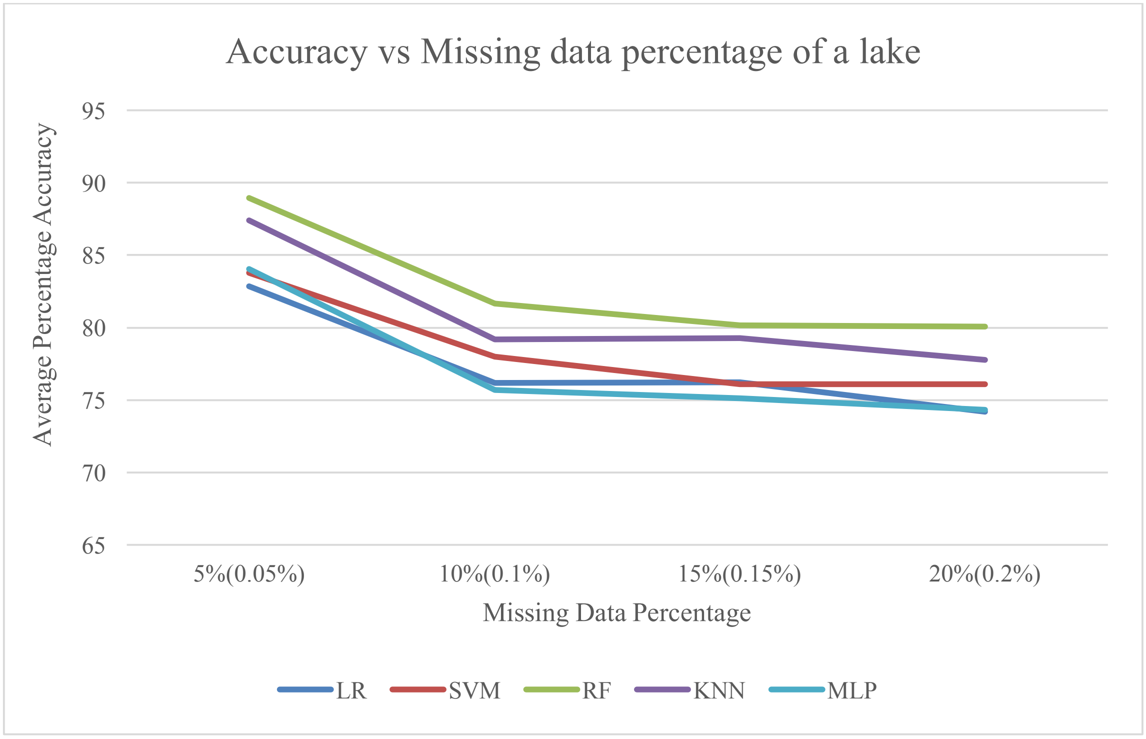


Figure 10: Classifier average percentage accuracy vs percentage of missing data of a lake. Some of the lines are overlapping.

As discussed in last section, accuracy is not sufficient to compare classifier performance due to Accuracy Paradox. Hence, we compare the classifiers based on precision and recall metrics (Table 6 and Table 7).

Figure 11 and Figure 12 demonstrate the average precision and recall of various classifiers at 5% missing lake data (equivalent to 0.05% missing data for entire dataset) for each trophic class. We note that kNN and random forest have high precision for all three classes. In addition, all the classifiers have higher precision for Oligotrophic and Hypereutrophic class in comparison to Eutrophic class. Logistic regression and Multilayer perceptron despite having low accuracy give high precision and recall for Hypereutrophic class when 5% of lake data (0.05% of total data) is missing. We also observe that random forest has high recall for all three classes.

To study the trends of precision and recall for the trophic states with change in percentage of missing data, we plot precision vs missing data and recall vs missing data curves. Figure 13,14,15 depict the change in classifier precision with change in percentage of missing data for Oligotrophic class, Eutrophic class and Hypereutrophic class respectively. Figure 16,17,18 demonstrate change in classifier recall with change in percentage of missing data for the three classes. From the above figures, we observe that Random forest and kNN have the best precision and recall for all three classes despite varying the missing data percentage. This aligns with our accuracy results, making Random Forest and kNN most suitable for the application.

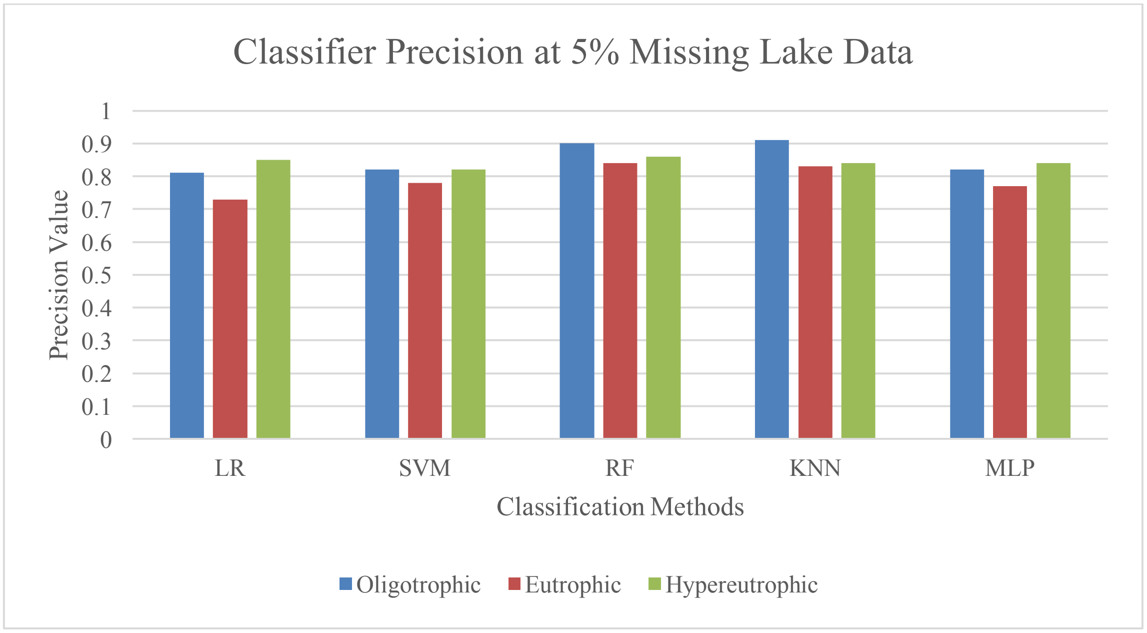


Figure 11: Average classifier Precision for three trophic states at 5% missing lake data (0.05% of entire dataset)

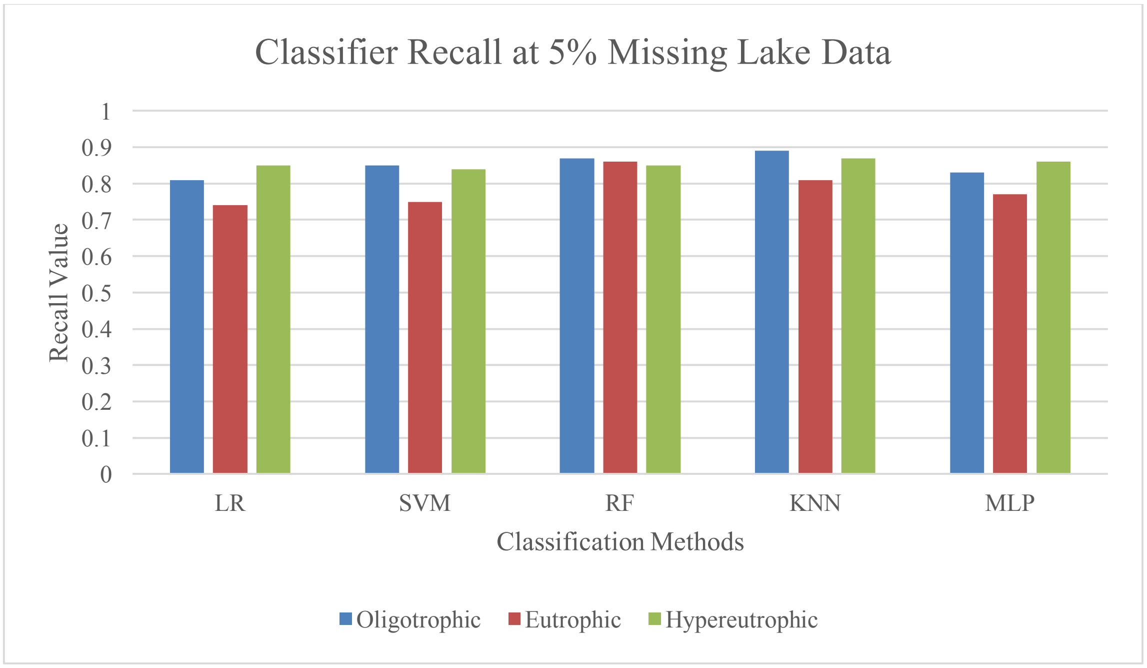


Figure 12: Classifier Recall for three trophic states at 5% missing data (0.05% of entire dataset)

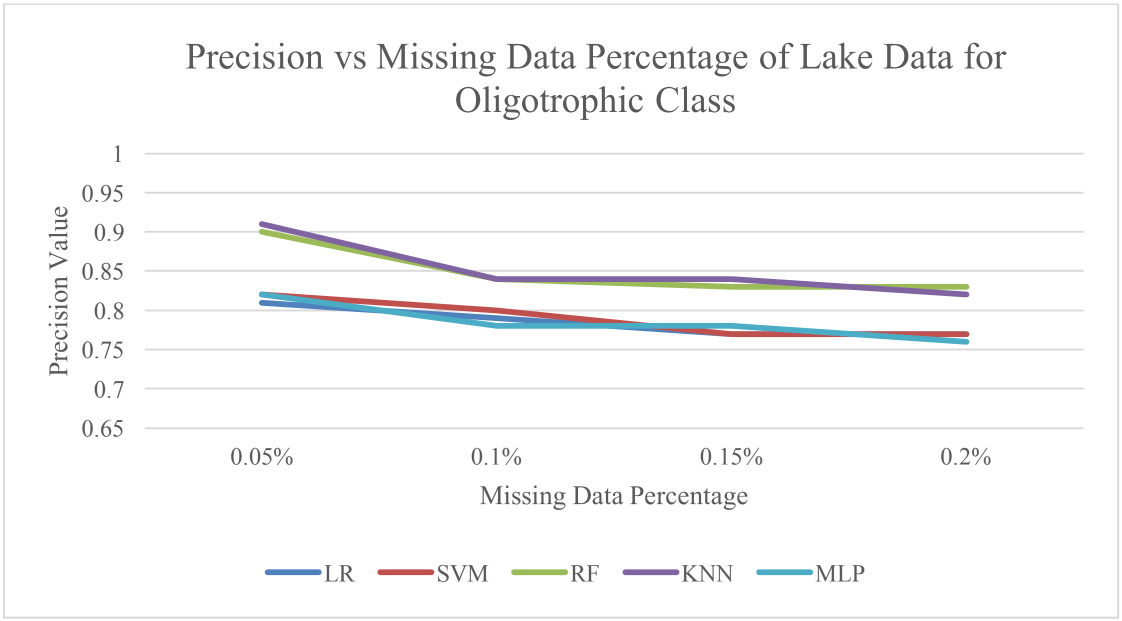


Figure 13: Classifier precision vs missing data percentage for Oligotrophic class. Some lines are overlapping and hence not visible.

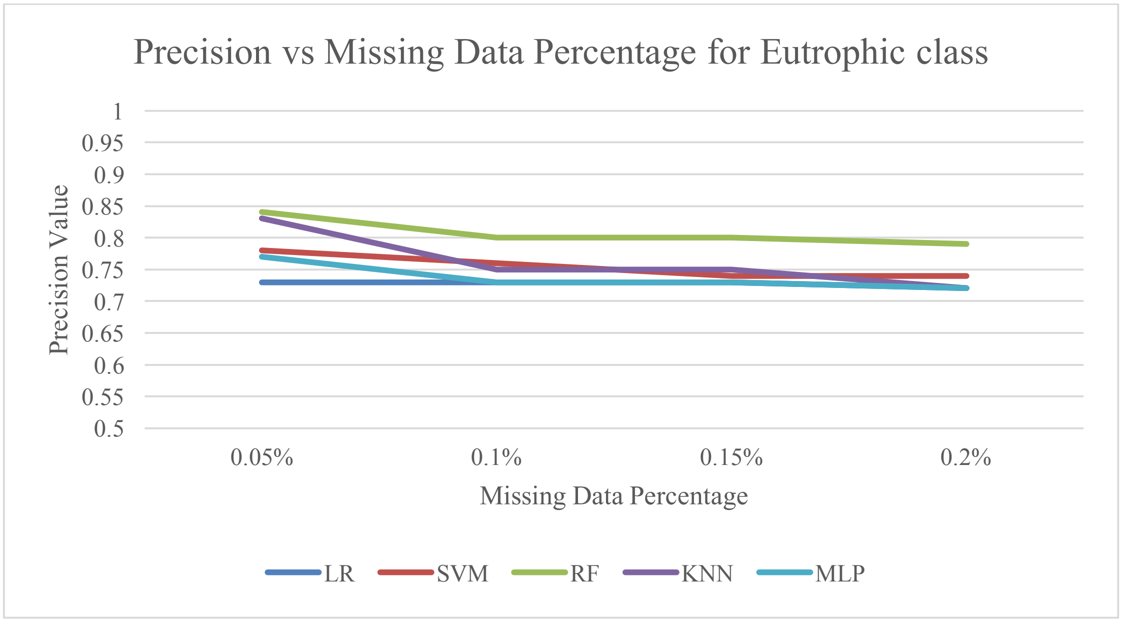


Figure 14: Classifier precision vs missing data percentage for Eutrophic class

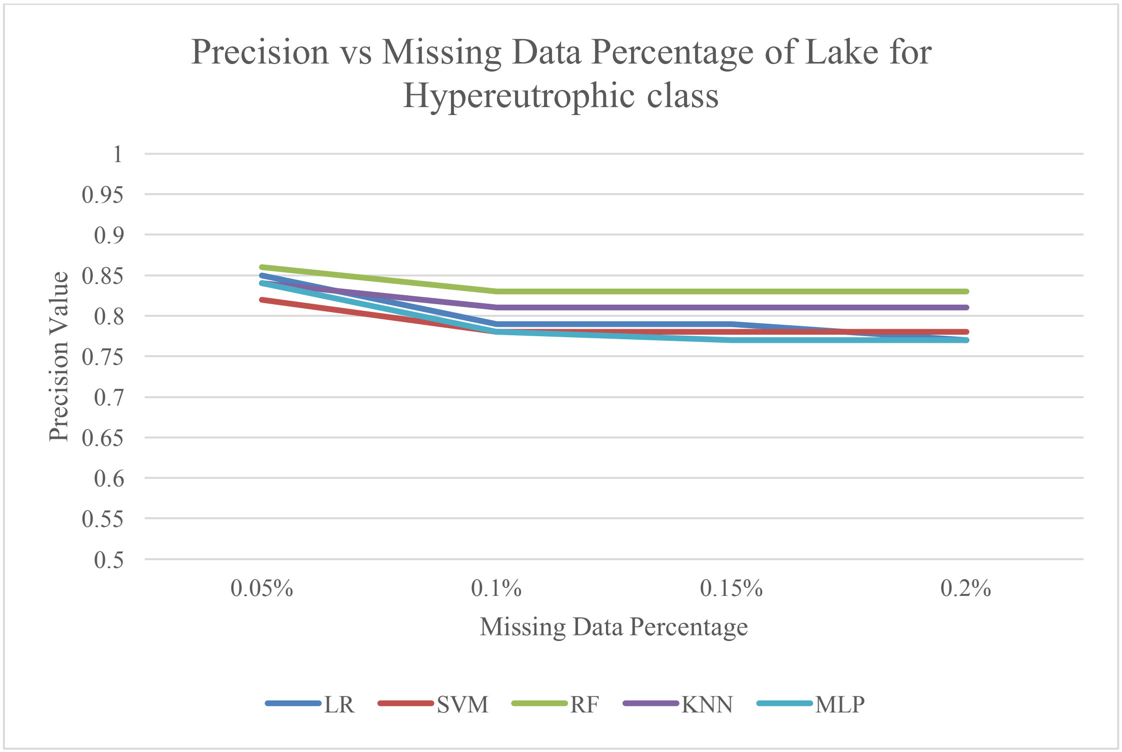


Figure 15: Classifier precision vs missing data percentage for Hypereutrophic class

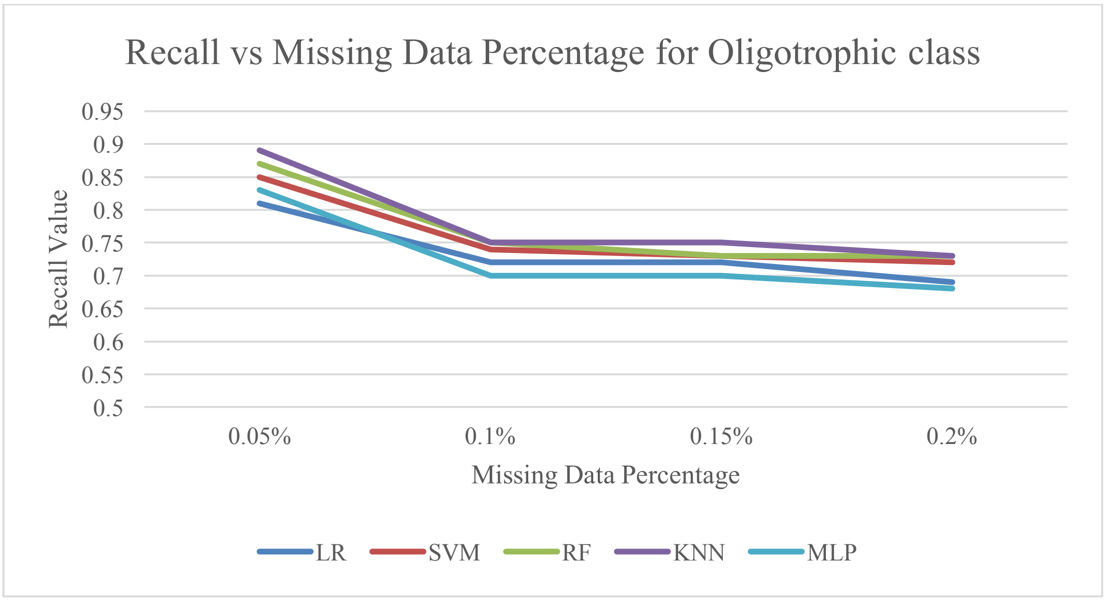


Figure 16: Classifier recall vs missing data percentage for Oligotrophic class

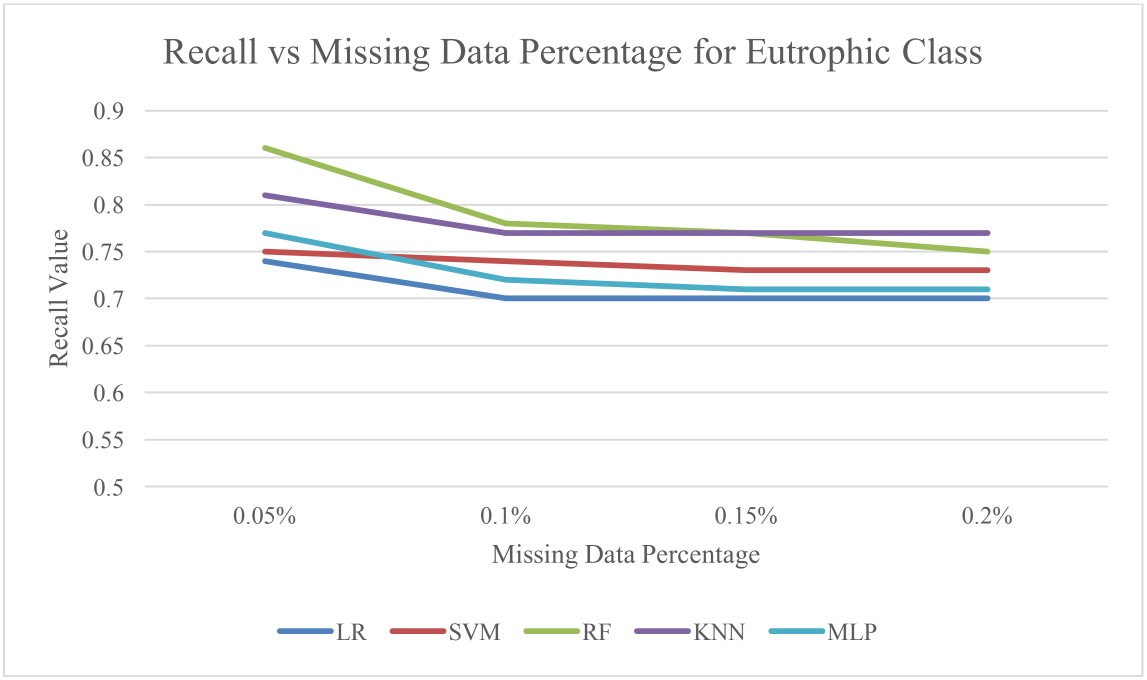


Figure 17: Classifier recall vs missing data percentage for Eutrophic class

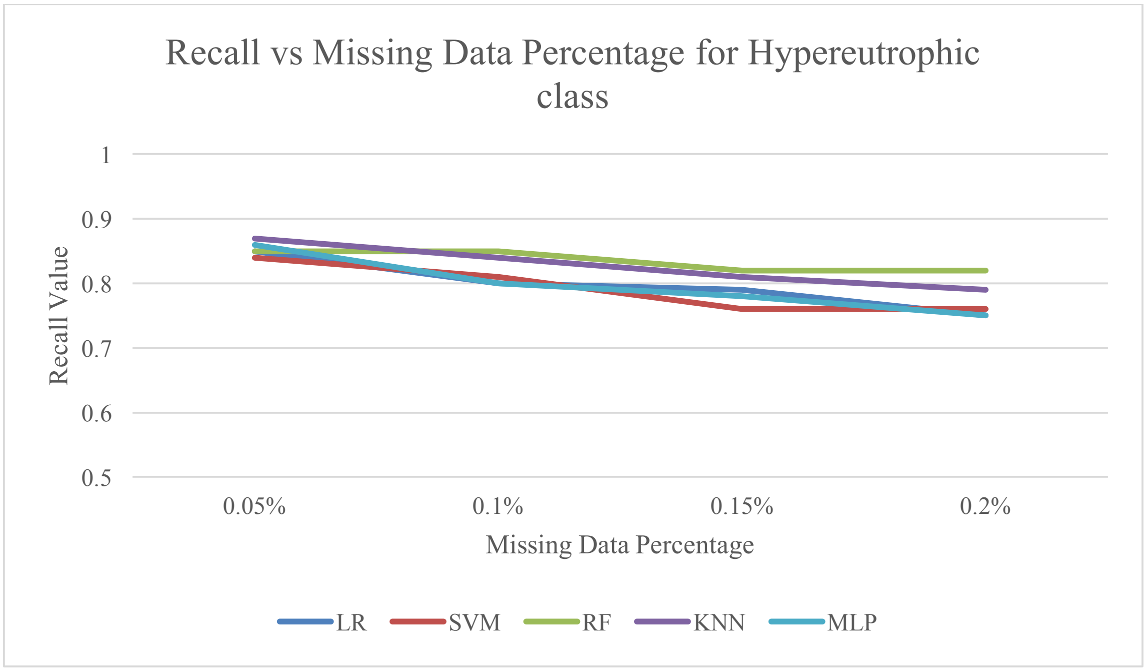


Figure 18: Classifier recall vs missing data percentage for Hypereutrophic class

5.2.4 Large Scale Data Missing Prediction Results:

We compare the classifier accuracy in Table 8 and Figure 19. We can observe that accuracy decreases with increase in percentage of missing data. In addition, we note that kNN performs the best, closely followed by Random Forest and SVM. Logistic Regression and MLP perform comparatively worse. We hypothesize kNN performs best as it is non–parametric and robust to noisy training data as the labels are assigned based on distance of a test instance from its neighbors. In addition, kNN is known to have performed better than multilayer perceptron and self-organizing maps for missing data imputation in medical domain [54]. We hypothesize Logistic Regression doesn't perform well as it obtains linear boundaries between classes and cannot perform well for non-linear problems. In addition, multilayer perceptron doesn't perform as well as other techniques because they require a large amount of data to train. Furthermore, multilayer perceptron is susceptible to getting stuck in local minima.

We compare classifiers based on precision and recall metrics (Table 9 and 10). Figure 20 and Figure 21 demonstrate the average precision and recall of various classifiers at 10% missing data for each trophic class. We observe that precision for Oligotrophic class is high for all classifiers, we hypothesize the cause of this to be more Oligotrophic observations in our dataset. We observe that kNN gives high precision and recall for all three classes closely followed by random forest which aligns with our accuracy results.

To study the trends of precision and recall for the trophic states with change in percentage of missing data, we plot precision vs missing data and recall vs missing data curves. Figures 22, 23, 24 depict the change in classifier precision with change in percentage of missing data for Oligotrophic class, Eutrophic class and Hypereutrophic class respectively. Figures 25, 26, 27 demonstrate change in classifier recall with change in percentage of missing data for the three classes. From the above figures, we observe that kNN and random forest outperform other algorithms and Logistic Regression performs the worst.

METHODS	2%	4%	6%	8%	10%
LR	78.77	77.11	74.89	71.79	69.72
SVM	81.05	79.84	76.1	74.18	72.22
RF	86.07	83.49	82.93	81.46	78.42
KNN	86.76	85.42	85.05	84.41	82.88
MLP	79.68	76.99	74.81	72.53	70.77

Table 8: Average accuracy results for large scale data missing trophic state prediction.

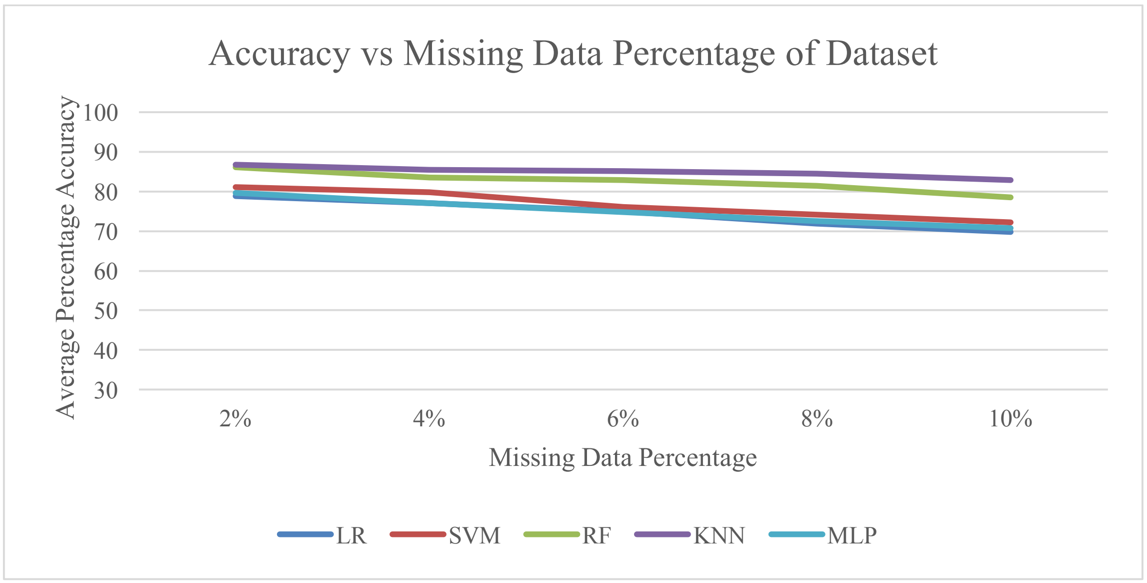


Figure 19: Average accuracy of classifier vs missing data percentage of entire dataset. Some lines are overlapping. Results reported after averaging accuracy of two trials.

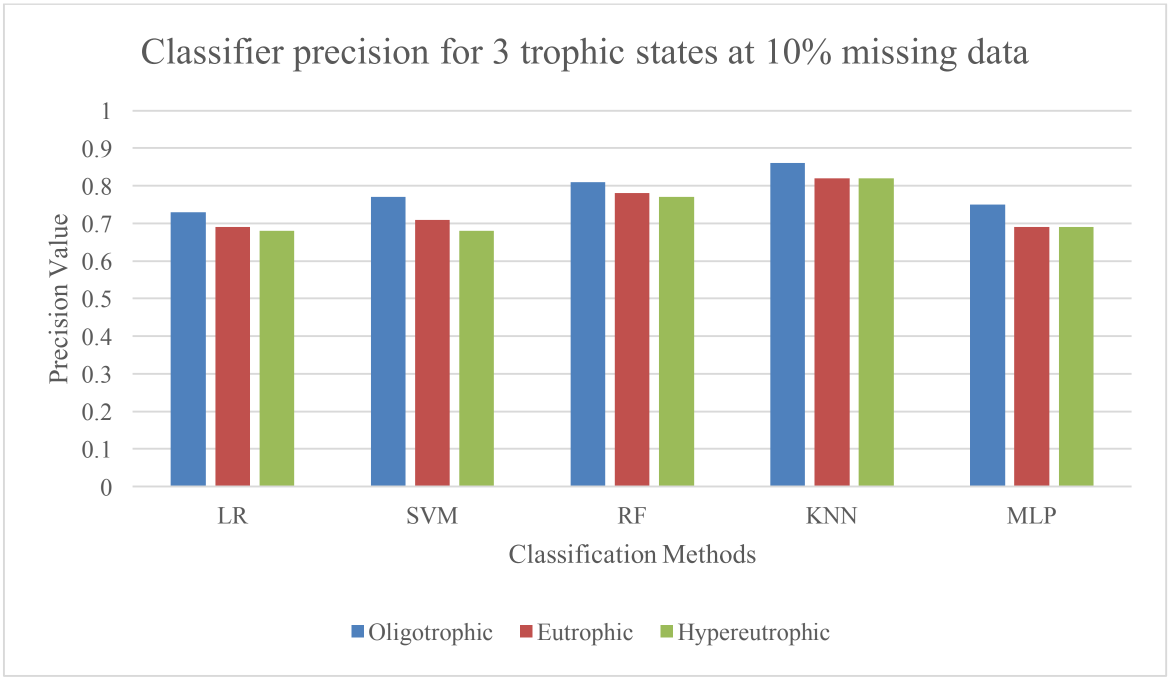


Figure 20: Classifier Precision for three trophic states at 10% missing data

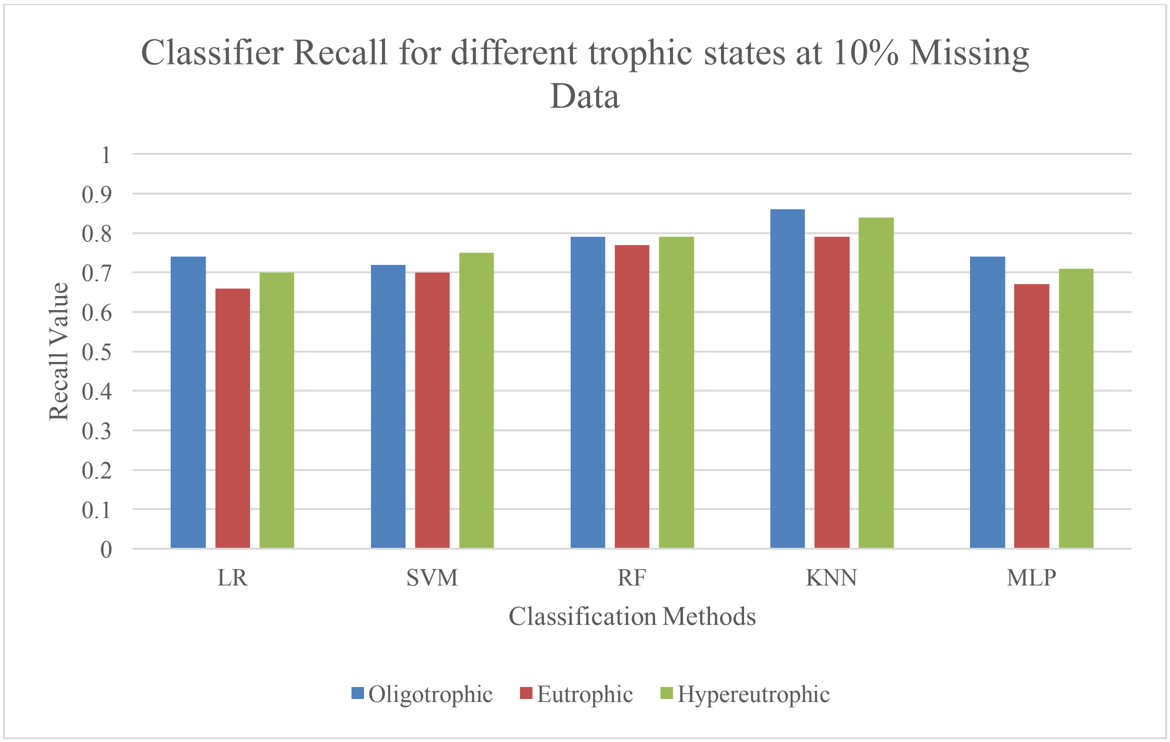


Figure 21: Classifier Recall for three trophic states at 10% missing lake da

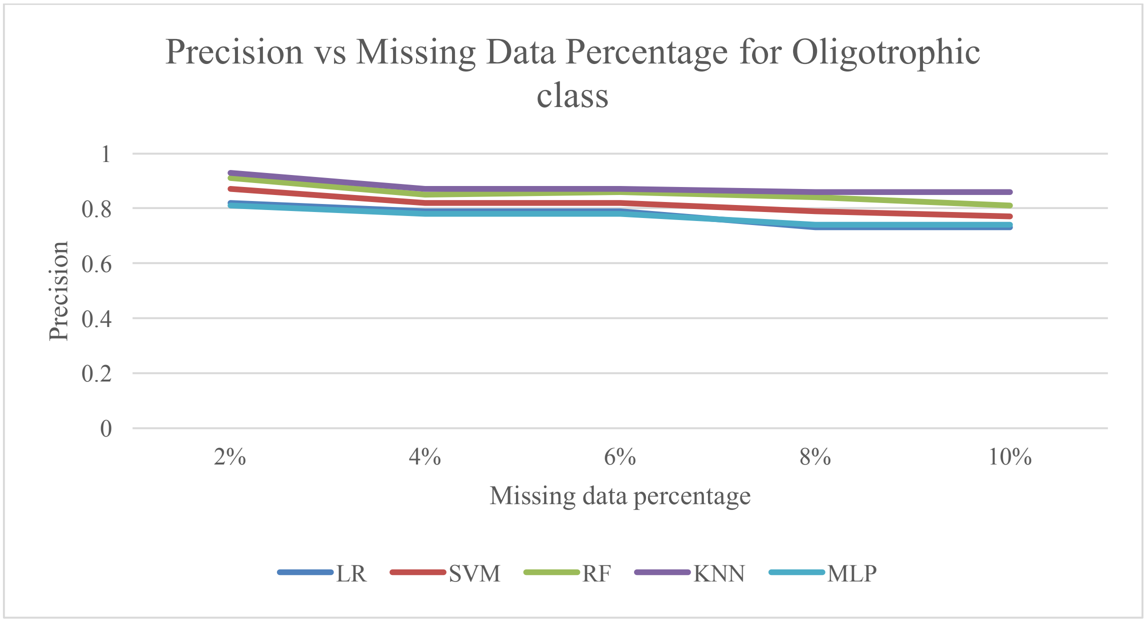


Figure 22: Classifier precision vs missing data percentage for Oligotrophic class. LR and MLP are almost overlapping.

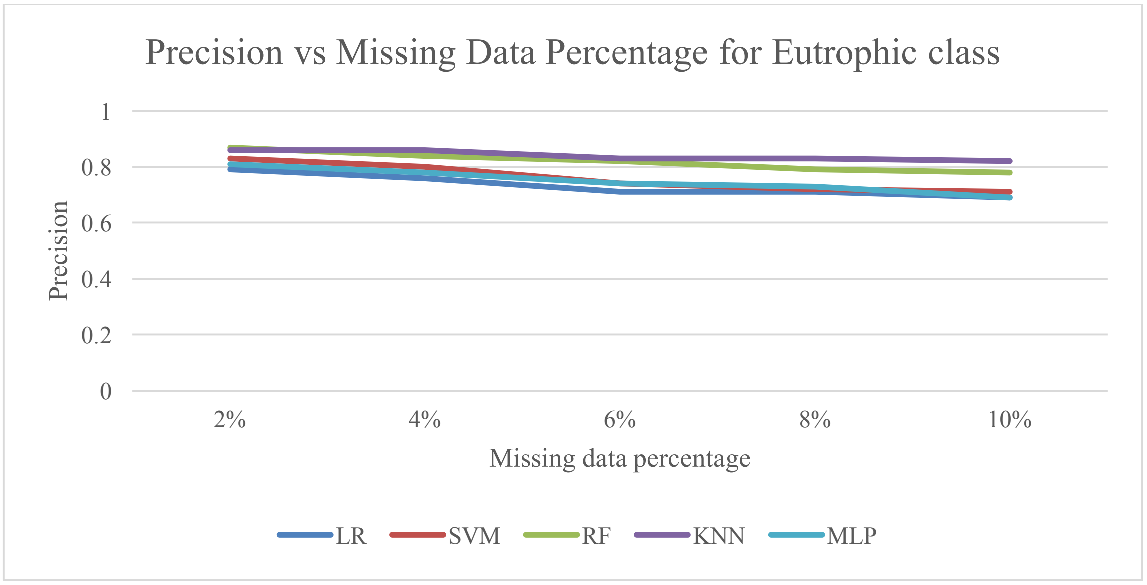


Figure 23: Classifier precision vs missing data percentage for Eutrophic class. Some lines are overlapping.

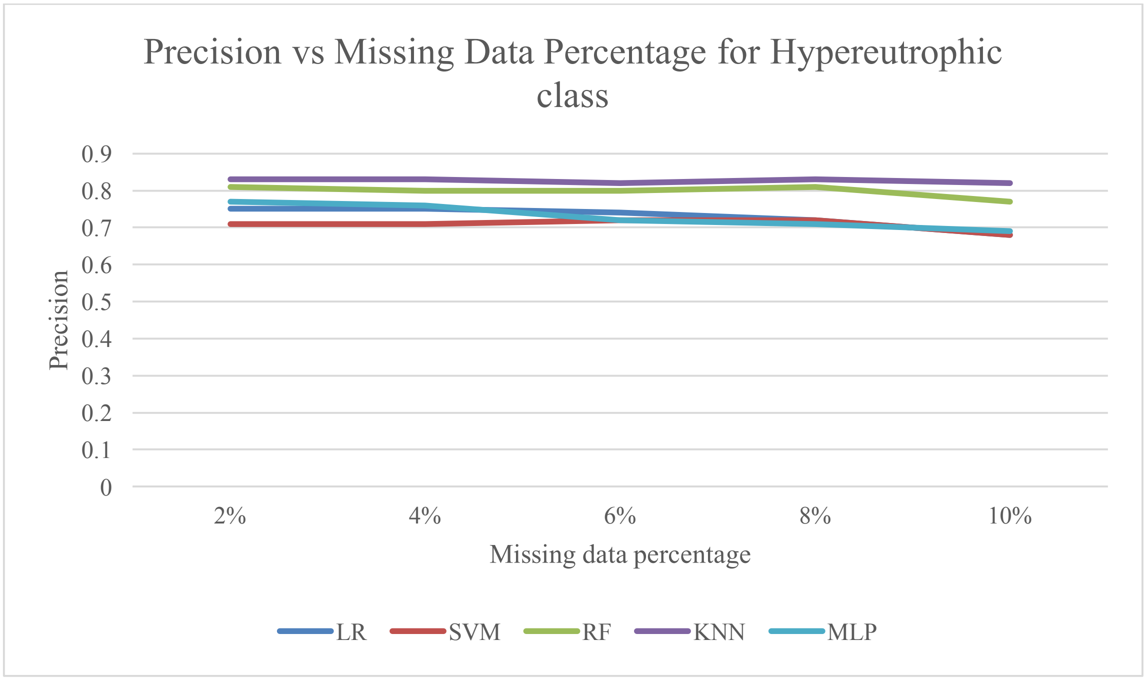


Figure 24: Classifier precision vs missing data percentage for Hypereutrophic class

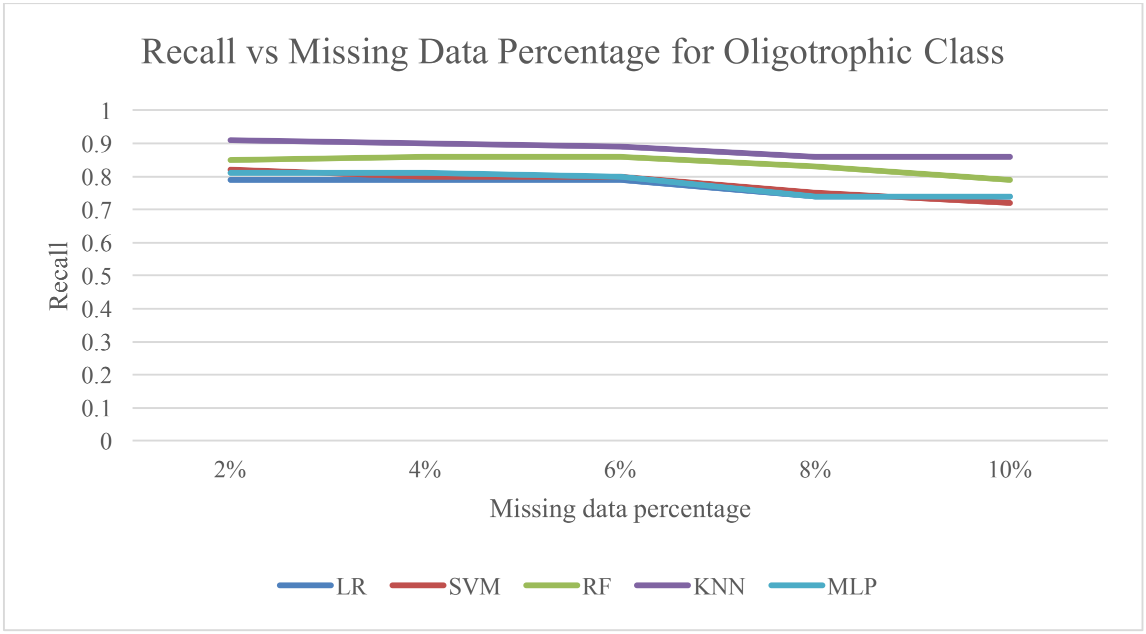


Figure 25: Classifier recall vs missing data percentage for Oligotrophic class

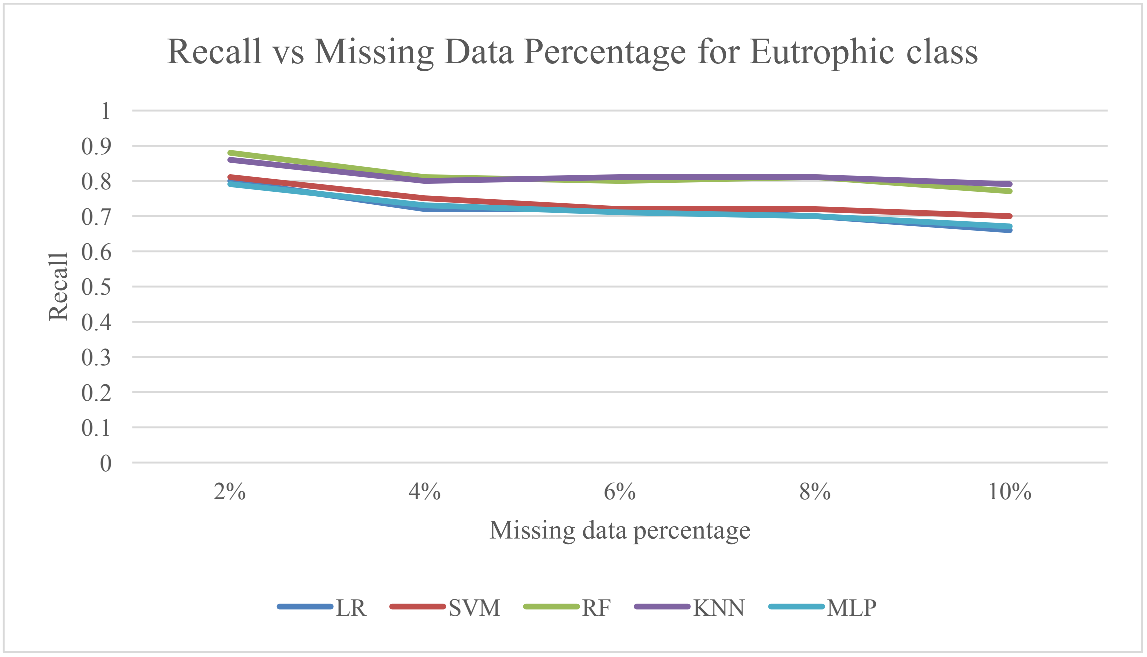


Figure 26: Classifier recall vs missing data percentage for Eutrophic class. Some lines are overlapping.

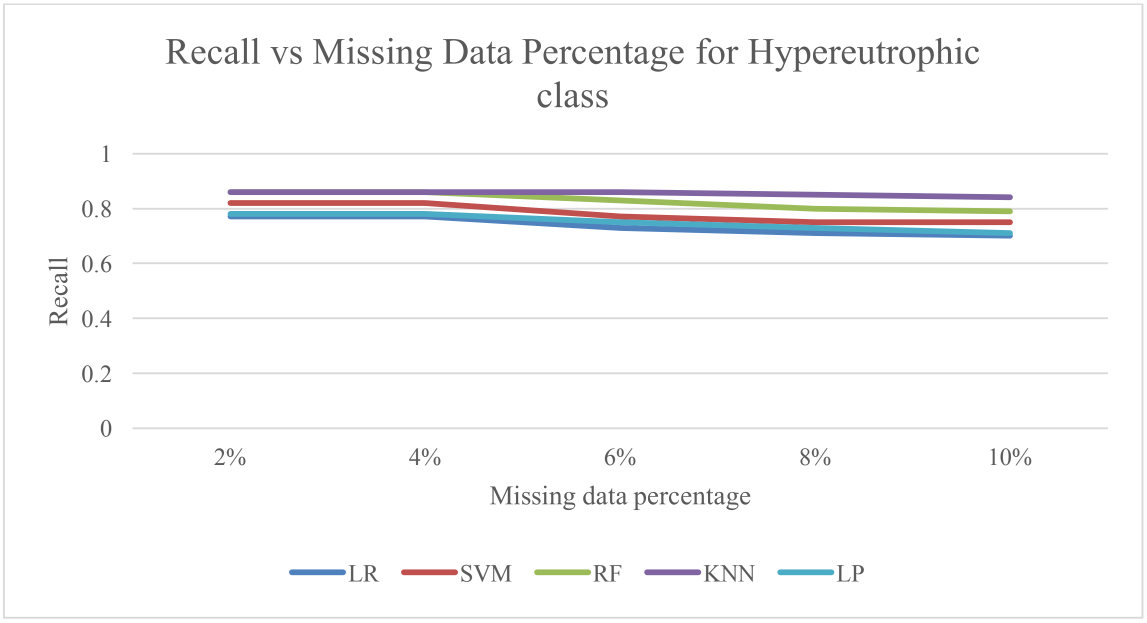


Figure 27: Classifier recall vs missing data percentage for Hypereutrophic class

5.3 Algal Severity Comparison:

5.3.1 Experimental Methodology:

We conducted an additional study to predict algal severity of lakes by exploiting both spatial and temporal data of other lakes. For this study, we use the dataset containing biweekly spatio-temporal data with percentage of algal bloom in three severity classes for each of the 99 lakes as discussed in dataset section.

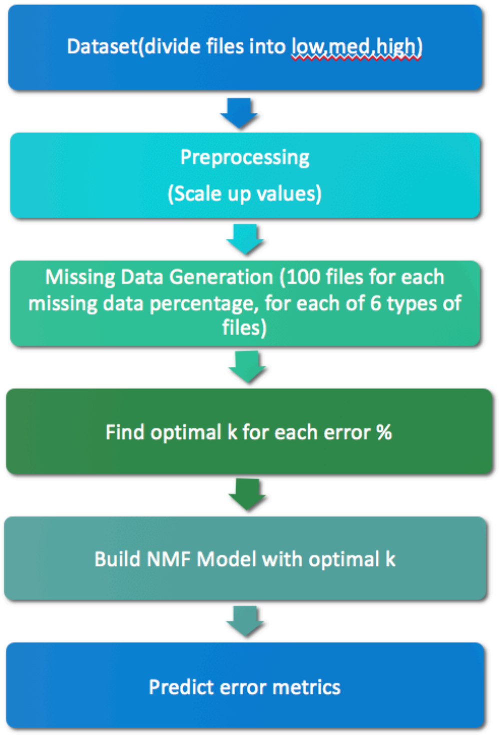

Dataset(divide files into low,med,high)
Preprocessing
(Scale up values)
Missing Data Generation (100 files for each missing data percentage, for each of 6 types of files)
Find optimal k for each error %
Build NMF Model with optimal k
Predict error metrics

Figure 28: NMF Pipeline

Before feeding our file to NMF model, we scale up the values in file by 1 as 0 is considered as missing value by NMF. Then, we divide the file into three different files containing low, medium and high percentage of pixels respectively. This results in 4 files – one with the original dataset and the others with low, medium and high pixel percentage. After generating four different files, we simulate y% of missing data by withholding random index values from all four files which we treat as matrix. For our experiments, we vary percentage of missing data between 5% to 20%.

We tackle the entire lake data missing problem by withholding all entries for a given lake at a given timestamp – low, med and high from original dataset. We call this "complete lake missing" problem. In addition, we tackle consecutive data missing problem by withholding consecutive data entries from the original dataset. We call it the "consecutive missing" problem. This results in six different kinds of files with missing data entries namely – low, medium, high, original, complete and consecutive.

For each of the six files, we vary missing data percentage between 5 to 20%. We perform 100 experiments by randomly generating missing data in order to tune k. We vary k from 1 to 100 and calculate average mean squared error over the 100 experiments for a given missing data percentage. We choose the k values, which give minimum average mean squared error at a given missing data percentage. For obtaining optimal k at a given missing data percentage, we average all the k values obtained above for each file.

Once we have our tuned hyperparameter k, we fit an NMF model with optimal k for each missing value percentage over 100 experiments. We calculate the prediction error metrics like Mean absolute error, Mean Squared error, R square error and Explained Variance score. Figure 28 shows the nmf pipeline.

Let y' be predicted value of ith instance, y be the corresponding true value and n be the number of instances, then the error metrics are computed as follows: [51]

$$MAE(y, y') = \frac{1}{n}\sum_{i=0}^{n-1} |y_i - y'_i|$$

$$RMSE(y, y') = sqrt\left(\frac{1}{n}\sum_{i=0}^{n-1} (y_i - y'_i)^2\right)$$

$$R^2(y, y') = 1 - \frac{\sum_{i=0}^{n-1}(y_i - y'_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2}$$

$$\text{exp}\, variance\ score\ (y, y') = 1 - \frac{Var(y - \hat{y})}{Var(y)}$$

5.3.2 Results:

We evaluate the results of our NMF model on different files by comparing their performance as percentage of missing data increases (Table 11). Figure 29 illustrates the effect of increasing missing data on RMSE values. Lower the RMSE value, better is the model's performance. We can observe from the plot that the least RMSE values while varying amount of missing data are given by original and complete file. The least value of RMSE achieved on these files is 13.59 at 5% missing data (Figure 29).

We also observe that when considered independently (when we split the original file to low, medium and high) the low, medium and high files perform worse than when they are considered together in original files. In addition, taking file with only high values in consideration is comparatively better than taking low or medium files independently for prediction.

We consider a naïve solution of taking average values of lake as predictions for missing data for a lake and compare our model to this naïve solution for the original file. Our model performs better as can be seen in Figure 30.

We can observe how R square values change with change in amount of missing data in Figure 31. The highest R square value of 0.82 is achieved when 5% of data is missing and prediction are made on original and complete file. R square value decreases from 0.82 to 0.65 when missing data is increased from 5% to 20% for the original file. When R square value is calculated for consecutive file (which contains consecutive simulated missing entries), we observe that R2 value decreases from 0.69 to 0.27. We hypothesize that our model performs worse on consecutive file than original file, because it's a biweekly dataset and missing consecutive entries means that atleast a month of data for a lake is missing. In a month, the cyanobacteria concentration of lakes can vary greatly due to surrounding environmental factors.

Average mean absolute error follows similar pattern to RMSE metric (Figure 32). The average mean absolute errors are least for predictions on original and complete file. We also plot average explained variance score vs percentage of missing data in Figure 33. Best value of explained variance score in an ideal situation is 1. Our model has an explained variance score of 0.86 for original file at 5% missing data which reduces with increase in percentage of missing data.

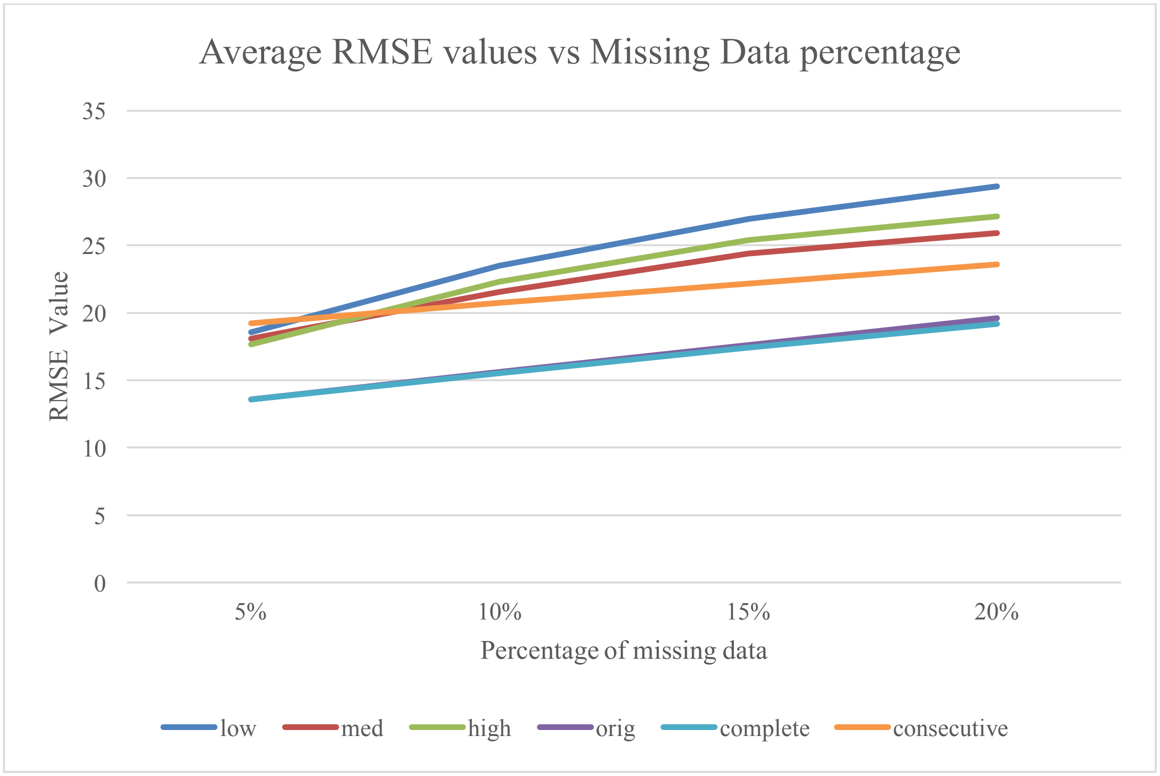


Figure 29: Average RMSE values vs missing data percentage. Yellow and blue lines are overlapping in the figure.

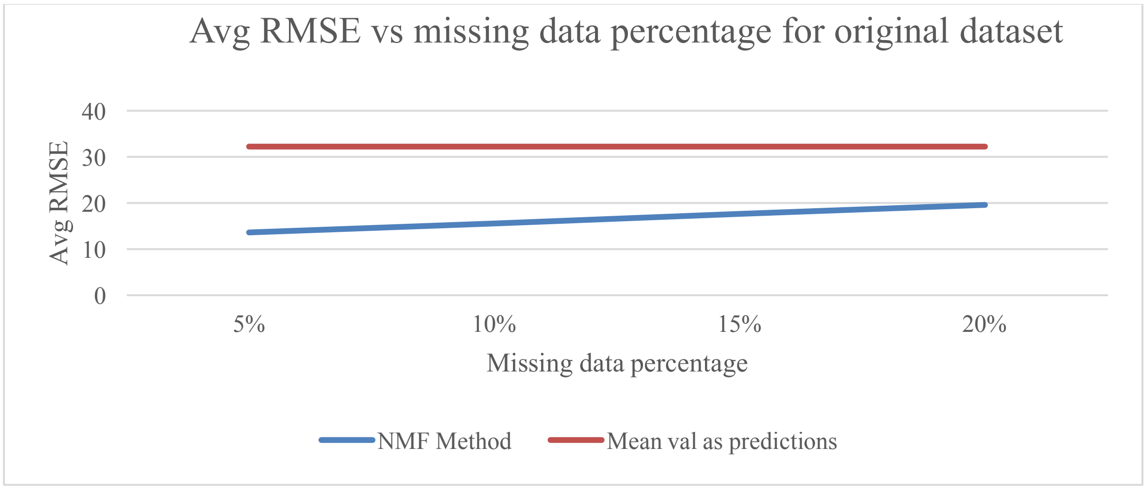


Figure 30: Average RMSE values vs missing data percentage for original file. Comparison of NMF method with naïve solution.

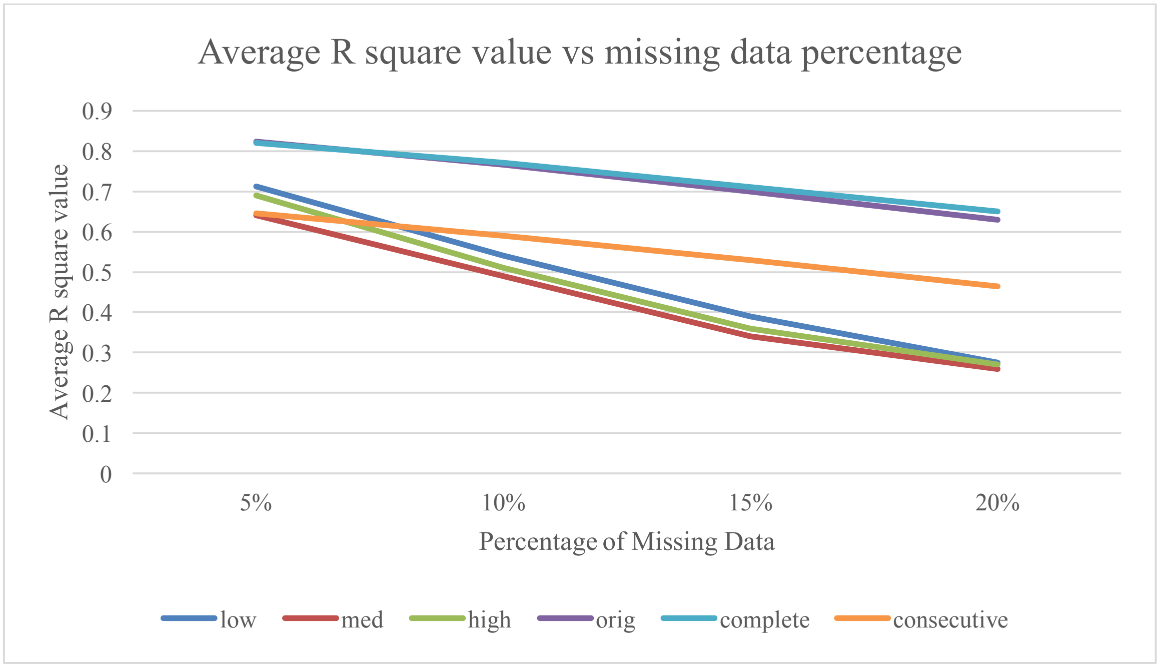


Figure 31: Average R square values vs missing data percentage. Yellow and blue lines almost are overlapping in the figure

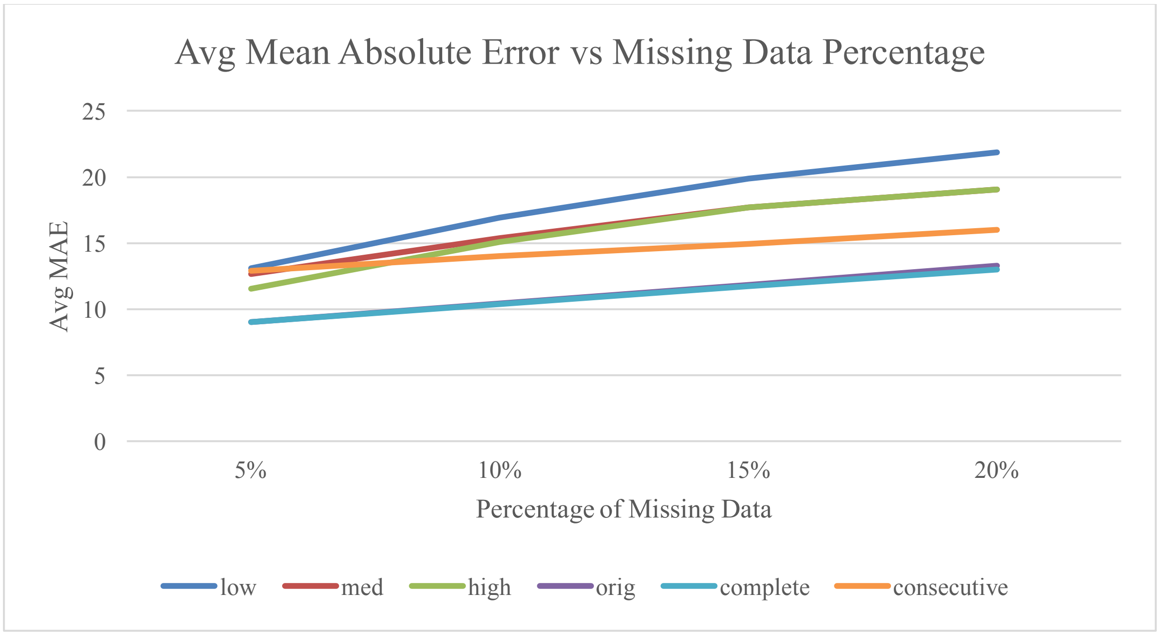


Figure 32: Average MAE values vs missing data percentage. Yellow and blue lines almost are overlapping in the figure.

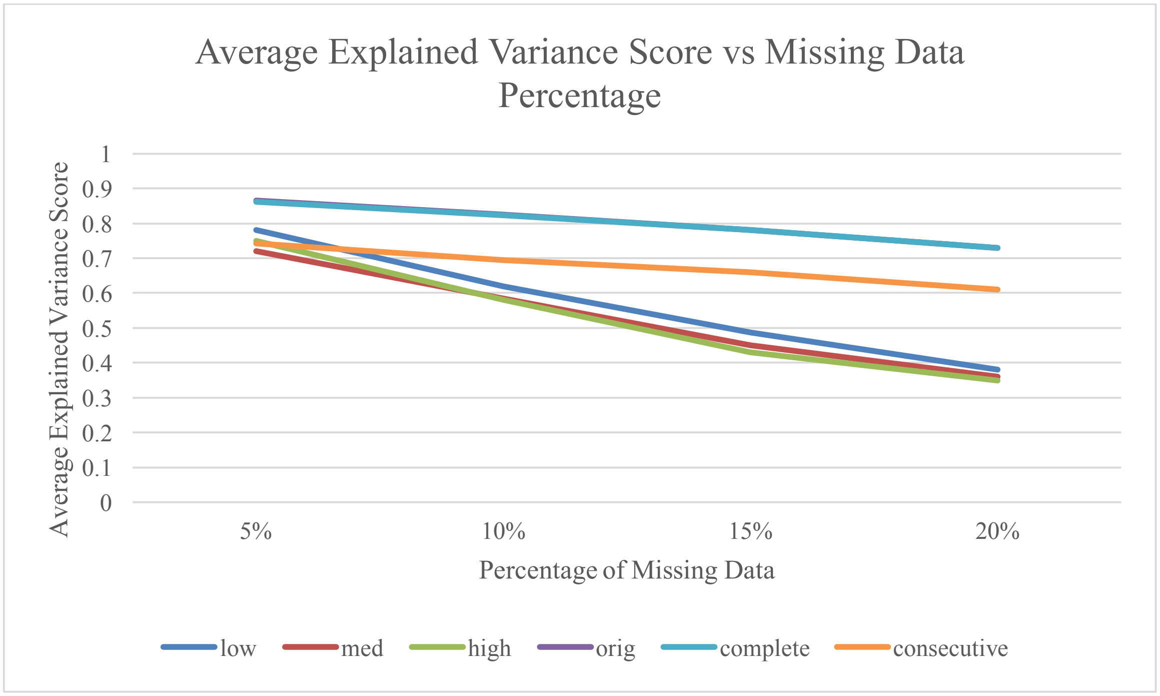


Figure 33: Average Explained Variance Score values vs missing data percentage. Yellow and blue lines almost are overlapping in the figure.

CHAPTER 6

CONCLUSIONS

Data incompleteness problem spans various domains due to a variety of causes like missing values, inconsistencies in data as well as data aggregation from a variety of sources [11]. Human induced errors or sensor failures due to technical or environmental issues also contribute to the missing data problem.

To solve missing data problem in the context of Cyanobacteria monitoring via satellites, we model the problem as a supervised classification problem. The spatial data of other lakes is exploited to predict the missing data of a particular lake at a given timestamp. We conclude that a lake scientist can leverage machine learning approaches like Random Forest and k Nearest Neighbors to classify data into different trophic states for cyanobacteria prediction. In addition, we demonstrate that spatio-temporal approaches like Non-negative matrix factorization give a good estimate of missing cyanobacteria data and algal severity for lakes.

REFERENCES

[1] W. W. Carmichael, W. R. Evans, Q. Q. Yin, P. Bell and E. Moczydlowski, "Evidence for paralytic shellfish poisons in the freshwater cyanobacterium Lyngbya wollei (Farlow ex Gomont) comb. nov.," *Applied and Environmental Microbiology,* vol. 63, no. 8, pp. 3104-3110, 1997.

[2] Y. G. Yanni and W. W. Carmichael, Screening of cyanobacteria isolated from soil, rice fields and water resources of Nile river delta for production of cyanotoxins, 1998, pp. 493-494.

[3] CDC, "CS258158 National Center for Environmental Health Division of Environmental Health and Health Hazards Cyanobacteria Blooms FAQs," 01 June 2017. [Online]. Available: https://www.cdc.gov/habs/pdf/cyanobacteria_faq.pdf. [Accessed 17 Oct 2017].

[4] EPA, "Cyanobacteria and Cyanotoxins: Information for Drinking Water Systems," 1 September 2014. [Online]. Available: https://www.epa.gov/sites/production/files/2014-08/documents/cyanobacteria_factsheet.pdf. [Accessed 17 October 2017].

[5] S. M. Azevedo, W. W. Carmichael, E. M. Jochimsen, K. L. Rinehart, S. Lau, G. R. Shaw and G. K. Eaglesham, "Human intoxication by microcystins during renal dialysis treatment in Caruaru—Brazil," *Toxicology,* vol. 181, pp. 441-446, 2002.

[6] L. Takser, N. Benachour, B. Husk, H. Cabana and D. Gris, "Cyanotoxins at low doses induce apoptosis and inflammatory effects in murine brain cells: potential implications for neurodegenerative diseases," *Toxicology Reports,* vol. 3, pp. 180-189, 2016.

[7] L. C. Backer, J. H. Landsberg, M. Miller, K. Keel and T. K. Taylor, "Canine cyanotoxin poisonings in the United States (1920s–2012): Review of suspected and confirmed cases from three data sources.," *Toxins,* vol. 5, no. 9, pp. 1597-1628, 2013.

[8] M. R. Penn, J. J. Pauer and J. R. Mihelcic, Environmental and ecological chemistry - Biochemical Oxygen Demand, vol. II, 2004.

[9] M. N. Khan and F. Mohammad, Eutrophication: challenges and solutions. In Eutrophication: Causes, Consequences and Control, Springer, 2014, pp. 1-15.

[10] F. S. Watanabe, E. Alcântara, T. W. Rodrigues, N. N. Imai, C. C. Barbosa and L. H. Rotta, "Estimation of Chlorophyll-a Concentration and the Trophic State of the Barra Bonita Hydroelectric Reservoir Using OLI/Landsat-8 Images," *International journal of environmental research and public health,* vol. 12, no. 9, pp. 10391-10417, 2015.

[11] V. Boddula, L. Ramaswamy and D. Mishra, "A Spatio-Temporal Mining Approach for Enhancing Satellite Data Availability: A Case Study on Blue Green Algae," *2017 IEEE International Congress on Big Data (BigData Congress),* pp. 216-223, 2017.

[12] N. R. Kevern, D. L. King and R. Ring, "Lake classification systems – Part 1. Three Rivers, MI: Michigan Lake and Stream Association," The Michigan Riparian February, 1996.

[13] "Trophic state index," 16 October 2016. [Online]. Available: https://en.wikipedia.org/wiki/Trophic_state_index. [Accessed 17 October 2017].

[14] "CyanoTracker," [Online]. Available: http://www.cyanotracker.uga.edu/. [Accessed 17 October 2017].

[15] R. J. A. Little and D. Rubin, Statistical analysis with missing data, New York: John Wiley & Sons, 1986.

[16] G. E. Batista and M. C. Monard, "An analysis of four missing data treatment methods for supervised learning," *Applied artificial intelligence,* vol. 17, no. 5-6, pp. 519-533, 2003.

[17] L. Nanni, A. Lumini and S. Brahnam, "A classifier ensemble approach for the missing feature problem," *Artificial intelligence in medicine,* vol. 55, no. 1, pp. 37-50, 2012.

[18] M. Fernández-Delgado, E. Cernadas, S. Barro and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems," *Journal of machine learning research,* vol. 15, no. 1, pp. 3133-3181, 2014.

[19] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein and R. B. Altman, "Missing value estimation methods for DNA microarrays," *Bioinformatics,* vol. 17, no. 6, pp. 520-525, 2001.

[20] D. Tsarev, R. Kurynin, M. Petrovskiy and I. Mashechkin, "Applying non-negative matrix factorization methods to discover user's resource access patterns for computer security tasks," in *Hybrid Intelligent Systems (HIS), 2014 14th International Conference on Pages 43-48*, 2014.

[21] X. Yi, Y. Zheng, J. Zhang and T. Li, "ST-MVL: filling missing values in geo-sensory time series data," in *IJCAI'16 Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence Pages 2704-2710* , 2016.

[22] J. W. Hollister, W. B. Milstead and B. J. Kreakie, "Modeling lake trophic state: a random forest approach.," *Ecosphere,* vol. 7, no. 3, 2016.

[23] S. Mishra and D. R. Mishra, "Normalized difference chlorophyll index: A novel model for remote estimation of chlorophyll-a concentration in turbid productive waters," *Remote Sensing of Environment,* vol. 117, pp. 394-406, 2012.

[24] "Logistic Regression," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Logistic_regression. [Accessed 17 October 2017].

[25] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society. Series B (Methodological),* pp. 215-242, 1958.

[26] D. G. Kleinbaum and M. Klein, "Analysis of matched data using logistic regression," in *Logistic Regression*, Springer New York, 2010, pp. 389-428.

[27] "Introduction to Support Vector Machines," OpenCV, [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html. [Accessed October 2017].

[28] R. Berwick, "An Idiot's guide to Support vector machines (SVMs)," [Online]. Available: http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf. [Accessed October 2017].

[29] M. Awad and R. Khanna, "Support Vector Machines for Classification," in *Efficient Learning Machines-Theories, Concepts, and Applications for Engineers and System Designers*, Apress, 2015, pp. 39-66.

[30] L. Brieman, "Random Forests," *Machine Learning,* vol. 45, no. 1, pp. 5-32, 2001.

[31] V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan and B. P. Feuston, "Random forest: a classification and regression tool for compound classification and QSAR modeling," *Journal of chemical information and computer sciences,* vol. 43, no. 6, pp. 1947-1958, 2003.

[32] "Ensemble methods," Scikit Learn, [Online]. Available: http://scikit-learn.org/stable/modules/ensemble.html#forest. [Accessed October 2017].

[33] S. Fortmann-Roe, "Understanding the bias-variance tradeoff," 2012. [Online]. Available: http://scott.fortmann-roe.com/docs/BiasVariance.html. [Accessed October 2017].

[34] K. Zakka, "A Complete Guide to K-Nearest-Neighbors with Applications in Python and R," [Online]. Available: https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/.

[35] F. Rosenblatt, "The perceptron, a perceiving and recognizing automaton Project Para.," Cornell Aeronautical Laboratory, 1957. [Online].

[36] E. M. Johansson, F. U. Dowla and D. M. Goodman, "Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method," *International Journal of Neural Systems,* vol. 2, no. 04, pp. 291-301, 1991.

[37] S. Ruder, "An overview of gradient descent optimization algorithms," 2016. [Online]. Available: arXiv preprint arXiv:1609.04747.

[38] D. Donoho and V. Stodden, "When does non-negative matrix factorization give a correct decomposition into parts?" in *Advances in neural information processing systems*, 2004.

[39] Y. X. Wang and Y. J. Zhang, "Nonnegative matrix factorization: A comprehensive review," *IEEE Transactions on Knowledge and Data Engineering,* vol. 25, no. 6, pp. 1336-1353, 2013.

[40] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, 2001.

[41] P. Paatero and U. Tapper, "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics,* vol. 5, no. 2, pp. 111-126, 1994.

[42] P. O. Hoyer, "Non-negative sparse coding," in *Neural Networks for Signal Processing. Proceedings of the 2002 12th IEEE Workshop on Pages 557-565*, 2002.

[43] P.O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *Journal of machine learning research,* vol. 5, no. Nov, pp. 1457-1469, 2004.

[44] V. P. Pauca, F. Shahnaz, M. W. Berry and R. J. Plemmons, "Text mining using non-negative matrix factorizations," in *Proceedings of the 2004 SIAM International Conference on Data Mining On Pages 452-456*, 2004.

[45] " Preprocessing and Label Encoding", Scikit learn, [Online]. Available: http://scikit-learn.org/stable/modules/preprocessing.html. [Accessed October 2017].

[46] C. Moffitt, "Guide to encoding categorical variables in python," Feb 2017. [Online]. Available: http://pbpython.com/categorical-encoding.html. [Accessed October 2017].

[47] "Feature transformation with ensembles of trees," Scikit learn, [Online]. Available: http://scikit-learn.org/stable/auto_examples/ensemble/plot_feature_transformation.html#sphx-glr-auto-examples-ensemble-plot-feature-transformation-py. [Accessed October 2017].

[48] G. C. Cawley and N. L. Talbot, "On over-fitting in model selection and subsequent selection bias in performance evaluation.," *Journal of Machine Learning Research,* vol. 11, no. Jul, pp. 2079-2107, 2010.

[49] K. Markham, "Simple guide to confusion matrix terminology," June 2016. [Online]. Available: http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/. [Accessed October 2017].

[50] F. J. Valverde-Albacete and C. Peláez-Moreno, "100% classification accuracy considered harmful: The normalized information transfer factor explains the accuracy paradox.," *PloS one,* vol. 9, no. 1, p. e84217, 2014.

[51] "Model evaluation: quantifying the quality of predictions," Scikit learn, [Online]. Available: http://scikit-learn.org/stable/modules/model_evaluation.html. [Accessed October 2017].

[52] D. R. Cutler, T. C. Edwards, K. H. Beard, A. Cutler, K. T. Hess, J. Gibson and J. J. Lawler, "Random forests for classification in ecology," *Ecology,* vol. 88, no. 11, pp. 2783-2792, 2007.

[53] J. Peters, B. De Baets, N. E. Verhoest, R. Samson, S. Degroeve, P. De Becker and W. Huybrechts, "Random forests as a tool for ecohydrological distribution modelling," *ecological modelling,* vol. 207, no. 2, pp. 304-318, 2007.

[54] J. M. Jerez, I. Molina, P. J. García-Laencina, E. Alba, N. Ribelles, M. Martín and L. Franco, " Missing data imputation using statistical and machine learning methods in a real breast cancer problem," *Artificial intelligence in medicine,* vol. 50, no. 2, pp. 105-115, 2010.

APPENDICES

A.1. Trophic state error metrics table – Experiment 1:

Table 6: Precision of classifiers at different missing data percentage for the 3 classes

Precision for 5% missing data	Oligotrophic	Eutrophic	Hypereutrophic
LR	0.81	0.73	0.85
SVM	0.82	0.78	0.82
RF	0.9	0.84	0.86
KNN	0.91	0.83	0.84
MLP	0.82	0.77	0.84

Precision for 10% missing data	Oligotrophic	Eutrophic	Hypereutrophic
LR	0.79	0.73	0.79
SVM	0.8	0.76	0.78
RF	0.84	0.8	0.83
KNN	0.84	0.75	0.81
MLP	0.78	0.73	0.78

Precision for 15% missing data	Oligotrophic	Eutrophic	Hypereutrophic
LR	0.77	0.73	0.79
SVM	0.77	0.74	0.78
RF	0.83	0.8	0.83
KNN	0.84	0.75	0.81
MLP	0.78	0.73	0.77

Precision for 20% missing data	Oligotrophic	Eutrophic	Hypereutrophic
LR	0.77	0.72	0.77
SVM	0.77	0.74	0.78
RF	0.83	0.79	0.83
KNN	0.82	0.72	0.81
MLP	0.76	0.72	0.77

Table 7: Classifier recall at different missing data percentage for the three classes

Recall for 5% missing data	Oligotrophic	Eutrophic	Hypereutrophic
LR	0.81	0.74	0.85
SVM	0.85	0.75	0.84
RF	0.87	0.86	0.85
KNN	0.89	0.81	0.87
MLP	0.83	0.77	0.86

Recall for 10% missing data	Oligotrophic	Eutrophic	Hypereutrophic
LR	0.72	0.70	0.80
SVM	0.74	0.74	0.81
RF	0.75	0.78	0.85
KNN	0.75	0.77	0.84
MLP	0.70	0.72	0.80

Recall for 15% missing data	Oligotrophic	Eutrophic	Hypereutrophic
LR	0.72	0.70	0.79
SVM	0.73	0.73	0.76
RF	0.73	0.77	0.82
KNN	0.75	0.77	0.81
MLP	0.70	0.71	0.78

Recall for 20% missing data	Oligotrophic	Eutrophic	Hypereutrophic
LR	0.69	0.70	0.75
SVM	0.72	0.73	0.76
RF	0.73	0.75	0.82
KNN	0.73	0.77	0.79
MLP	0.68	0.71	0.75

A.2. Trophic state error metrics table – Experiment 2:

Table 9: Precision of classifiers at different missing data percentage for the three classes

Precision for 2% missing data	Oligotrophic	Eutrophic	Hypereutrophic
LR	0.82	0.79	0.75
SVM	0.87	0.83	0.71
RF	0.91	0.87	0.81
KNN	0.93	0.86	0.83
MLP	0.81	0.81	0.77

Precision for 4% missing data	Oligotrophic	Eutrophic	Hypereutrophic
LR	0.79	0.76	0.75
SVM	0.82	0.8	0.71
RF	0.85	0.84	0.8
KNN	0.87	0.86	0.83
MLP	0.78	0.78	0.76

Precision for 6% missing data	Oligotrophic	Eutrophic	Hypereutrophic
LR	0.79	0.71	0.74
SVM	0.82	0.74	0.72
RF	0.86	0.82	0.8
KNN	0.87	0.83	0.82
MLP	0.78	0.74	0.72

Precision for 8% missing data	Oligotrophic	Eutrophic	Hypereutrophic
LR	0.73	0.71	0.72
SVM	0.79	0.72	0.72
RF	0.84	0.79	0.81
KNN	0.86	0.83	0.83
MLP	0.74	0.73	0.71

Precision for 10% missing data	Oligotrophic	Eutrophic	Hypereutrophic
LR	0.73	0.69	0.68
SVM	0.77	0.71	0.68
RF	0.81	0.78	0.77
KNN	0.86	0.82	0.82
MLP	0.74	0.69	0.69

Table 10: Classifier recall at different missing data percentage for the three classes

Recall for 2% missing data	Oligotrophic	Eutrophic	Hypereutrophic
LR	0.79	0.8	0.77
SVM	0.82	0.81	0.82
RF	0.85	0.88	0.86
KNN	0.91	0.86	0.86
MLP	0.81	0.79	0.78

Recall for 4% missing data	Oligotrophic	Eutrophic	Hypereutrophic
LR	0.79	0.72	0.77
SVM	0.8	0.75	0.82
RF	0.86	0.81	0.86
KNN	0.9	0.8	0.86
MLP	0.81	0.73	0.78

Recall for 6% missing data	Oligotrophic	Eutrophic	Hypereutrophic
LR	0.79	0.72	0.73
SVM	0.8	0.72	0.77
RF	0.86	0.80	0.83
KNN	0.89	0.81	0.86
MLP	0.80	0.71	0.75

Recall for 8% missing data	Oligotrophic	Eutrophic	Hypereutrophic
LR	0.74	0.70	0.71
SVM	0.75	0.72	0.75
RF	0.83	0.81	0.80
KNN	0.86	0.81	0.85
MLP	0.74	0.70	0.73

Recall for 10% missing data	Oligotrophic	Eutrophic	Hypereutrophic
LR	0.74	0.66	0.70
SVM	0.72	0.70	0.75
RF	0.79	0.77	0.79
KNN	0.86	0.79	0.84
MLP	0.74	0.67	0.71

B. **Algal Severity Error Metric Tables**

Table 11: Error metrics for NMF on different files

Metric for low file analysis	5%	10%	15%	20%
Average R square	0.712	0.54	0.39	0.275
Average RMSE	18.56	23.48	26.96	29.4
Average MAE	13.09	16.93	19.87	21.88
Average explained variance score	0.78	0.62	0.487	0.38

Metric for med file analysis	5%	10%	15%	20%
Average R square	0.64	0.49	0.34	0.26
Average RMSE	18.07	21.53	24.4	25.92
Average MAE	12.64	15.39	17.72	19.07
Average explained variance score	0.72	0.585	0.45	0.36

Metric for high file analysis	5%	10%	15%	20%
Average R square	0.69	0.51	0.36	0.27
Average RMSE	17.65	22.29	25.4	27.16
Average MAE	11.55	15.06	17.7	19.06
Average explained variance score	0.75	0.58	0.43	0.35

Metric for original file analysis	5%	10%	15%	20%
Average R square	0.8227	0.766	0.7	0.63
Average RMSE	13.59	15.63	17.61	19.61
Average MAE	9.01	10.44	11.84	13.28
Average explained variance score	0.8654	0.825	0.78	0.73

Metric for complete file analysis	5%	10%	15%	20%
Average R square	0.82	0.77	0.71	0.65
Average RMSE	13.597	15.51	17.41	19.18
Average MAE	9.019	10.36	11.73	12.99
Average explained variance score	0.8611	0.823	0.78	0.73

Metric for consecutive file analysis	5%	10%	15%	20%
Average R square	0.646	0.59	0.53	0.465
Average RMSE	19.21	20.76	22.19	23.6
Average MAE	12.88	14	14.93	16
Average explained variance score	0.743	0.695	0.66	0.61