NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

MACHINE LEARNING TECHNIQUES FOR DEVELOPMENT OF A CONDITION-BASED MAINTENANCE PROGRAM FOR NAVAL PROPULSION PLANTS

by

Eric A. Therrio

December 2018

Thesis Advisor: Monique P. Fargues
Co-Advisor: Roberto Cristi

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2018	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE MACHINE LEARNING TECHNIQUES FOR DEVELOPMENT OF A CONDITION-BASED MAINTENANCE PROGRAM FOR NAVAL PROPULSION PLANTS	**5. FUNDING NUMBERS**
6. AUTHOR(S) Eric A. Therrio	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

In this thesis, we investigate a specific type of machine learning (ML) algorithm, specifically a support vector machine (SVM) regressor, as the foundation behind a condition-based maintenance (CBM) program for the major components affecting a naval propulsion system (NPS). This program is designed to specifically monitor the degradation of the ship's engines, the propeller, and the hull. Simulated data generated in previous work by modeling a combined diesel electric and gas NPS is applied to design the SVM and optimize its hyperparameter values—insensitivity, penalty parameter, and kernel spread. Our results show that an optimally tuned and trained SVM algorithm can make predictions with error rates below 0.5%. Results also show our SVM algorithm outperforms the SVM algorithm discussed in previous work. In this work, we established a good base for developing a CBM program for the U.S. Navy.

14. SUBJECT TERMS machine learning, support vector machine, Gaussian kernel, condition-based maintenance	**15. NUMBER OF PAGES** 61
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

MACHINE LEARNING TECHNIQUES FOR DEVELOPMENT OF A CONDITION-BASED MAINTENANCE PROGRAM FOR NAVAL PROPULSION PLANTS

Eric A. Therrio
Lieutenant, United States Navy
BSEE, University of Arizona, 2013

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2018

Approved by: Monique P. Fargues
Advisor

Roberto Cristi
Co-Advisor

Clark Robertson
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In this thesis, we investigate a specific type of machine learning (ML) algorithm, specifically a support vector machine (SVM) regressor, as the foundation behind a condition-based maintenance (CBM) program for the major components affecting a naval propulsion system (NPS). This program is designed to specifically monitor the degradation of the ship's engines, the propeller, and the hull. Simulated data generated in previous work by modeling a combined diesel electric and gas NPS is applied to design the SVM and optimize its hyperparameter values—insensitivity, penalty parameter, and kernel spread. Our results show that an optimally tuned and trained SVM algorithm can make predictions with error rates below 0.5%. Results also show our SVM algorithm outperforms the SVM algorithm discussed in previous work. In this work, we established a good base for developing a CBM program for the U.S. Navy.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

CBM	condition-based maintenance
CBM+	condition-based maintenance plus
CODLAG	combined diesel electric and gas
COGAG	combined gas and gas
CPP	controllable pitch propeller
FRCSW	Fleet Readiness Center Southwest
GE	General Electric
GT	gas turbine
GTC	gas turbine compressor
HLL	hull
HP	high power
LP	low power
MAPE	mean absolute percentage error
ML	machine learning
MRG	main reduction gear
SVM	support vector machine
NPS	naval propulsion system
PRP	propeller
RCM	reliability-centered maintenance
RPM	rotations per minute
RPS	rotations per second
TIC	turbine injection control

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

This work is dedicated to my mother, Debra Therrio, for always supporting and pushing me to attain my goals. I would also like to thank my amazing girlfriend, Brooke Devereaux, for always keeping me going and putting a smile on my face.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. OBJECTIVE

Operational requirements of U.S. Navy ships have steadily increased over the past fifteen years, while the Navy ranks have shrunk by 20 percent [1]. Budget cuts and sequestration have led the Navy to operate and maintain a leaner fleet on a smaller budget. Even with an increase in defense spending for fiscal year 2019, ship maintenance is an important factor in safety, operational readiness, and total system cost. One of the major systems is a ship's propulsion system for which the engines, propellers, and hull state are important factors. Each of these major systems requires dedicated in-port periods or even dry dock availability, which incurs major costs and takes away from operational availability. In this thesis, we attempt to take advantage of modern data collection capability to develop a machine learning (ML) algorithm, specifically a support vector machine (SVM) regressor, to implement a condition-based maintenance (CBM) program for the major components affecting a naval propulsion system (NPS). Specifically, we target the degradation of the compressor and turbines of the main engines, the propeller, and the hull. The final objective is to produce an algorithm that can consider a multitude of sensor data to accurately determine when one of these major subsystems needs servicing.

B. MOTIVATION

The Navy has two CBM programs which both rely on a third concept, reliability-centered maintenance (RCM). OPNAVINST 4790.16B, the Naval Instruction governing CBM, defines "condition-based maintenance" and "condition-based maintenance plus" [2]. Condition-based maintenance is defined in part as "a maintenance strategy derived from analysis, using DoD approved RCM principles. CBM includes maintenance processes and capabilities derived from real or near-real time assessments obtained from embedded sensors and external tests and measurements using either portable equipment or actual inspection" [2]. The second program is condition-based maintenance plus (CBM+) defined as "the application and integration of appropriate processes, technologies, and knowledge-based capabilities to achieve the target availability, reliability, and operation and support

costs of DoD systems and components across their life cycle. At its core, CBM+ is maintenance performed based on evidence of need" [2]. RCM is defined as "a method for determining maintenance requirements based on the analysis of the likely functional failures of components, equipment, subsystems, or systems having a significant impact on safety, operations, and life cycle cost" [3]. In summary, RCM is the process that establishes the conditions in which a CBM or CBM+ program is built. The main difference between a CBM and a CBM+ program is that in CBM, the conditions are decided from current sensor readings, whereas CBM+ applies "technologies and knowledge-based capabilities" [2] such as emerging ML techniques.

Currently, the General Electric (GE) LM2500 engine type is used for many of the U.S. Navy's surface combatants to include all cruisers and destroyers as well as surface ships from many other foreign nations. The LM2500 engines undergo service at the Navy's Fleet Readiness Center Southwest (FRCSW) based on a CBM program. Currently, the known decay state of the gas turbine (GT) engine is not actually known, and decisions to service engines are based on direct sensor readings of current running condition [4].

Cipollini, Oneto, Coraddu, Murphy, and Anguita developed "a real-data validated complex numerical simulator" [5] of a navy frigate characterized by a combined diesel electric and gas (CODLAG) propulsion plant that was developed in MATLAB using the Simulink toolbox. This simulated model was run to generate a dataset [6] used to compare the performances of several ML techniques [7]. In this work, we design an SVM in MATLAB and apply it to this specific dataset [6] to investigate whether a CBM+ program can be implemented.

Due to the need for navy ships to be at a fully operational status, many maintenance programs are designed on a conservative schedule to minimize risk of a breakdown. Although GT engines are on a CBM program, the actual degradation state is not known before the engine is removed from the ship. The same can be said about the degradation states of the hull and propellers. Direct measurements of the state of degradation of these major subsystems are not possible; however, a lot of sensor data is available that makes this problem an excellent choice for ML.

C. THESIS CHAPTER BREAKDOWN

We organize the rest of the thesis in the following manner. In Chapter II, we present the background on the dataset we use and some basic background on ML algorithms and SVMs. The procedure described in Chapter III includes the analysis of the dataset, the SVM algorithm training setup in MATLAB, and the procedure for obtaining the optimum hyperparameter settings. Results of the training optimization are presented in Chapter IV. Finally, our conclusions and recommendations for future work are presented in Chapter V.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

A. SHIP PROPULSION MODELS

One of the greatest obstacles to developing a ML algorithm is gathering and classifying data with which to train the algorithm. Previous work [5] created a simulation to model the sensor inputs and degradation outputs of a GT engine, specifically a GE LM2500 engine on a naval frigate. Coraddu, Cipollini, Oneto, and Anguita then improved on that initial model to include additional inputs affecting an NPS and added degradation outputs for the propeller and hull [7]. This improved simulation generated the dataset [6] which was used in [7] to test more sophisticated ML algorithms. In this thesis, we use this specific dataset [6] to develop a regression SVM model.

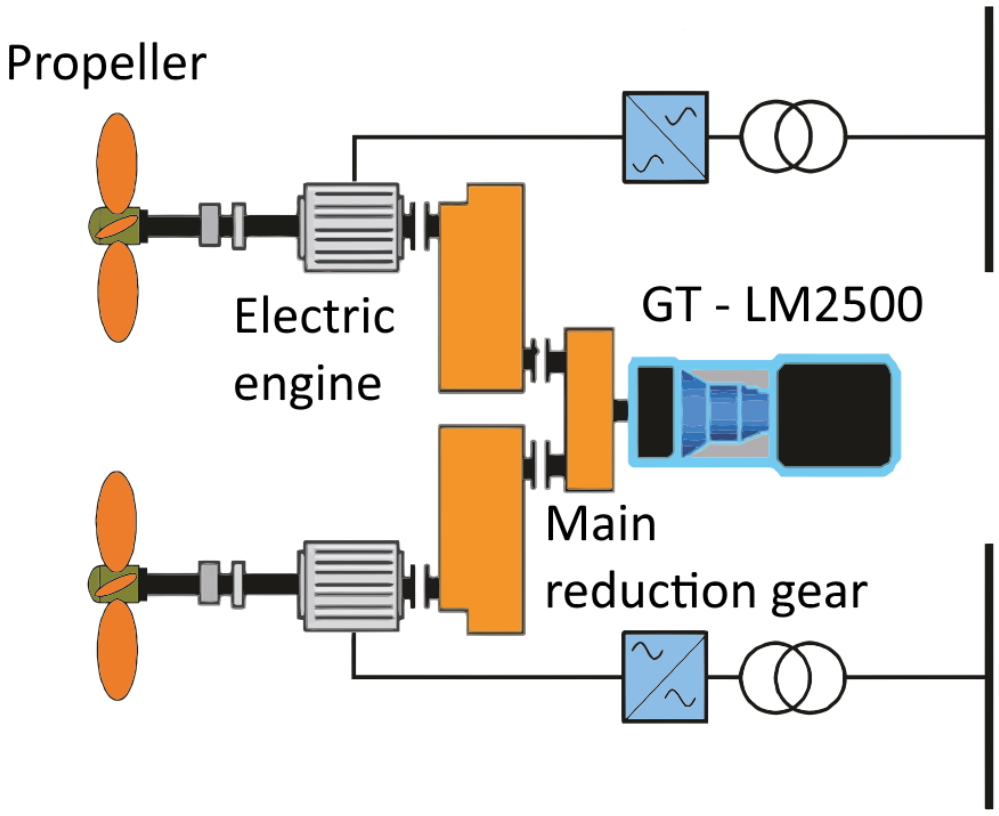

Figure 1. CODLAG NPS. Adapted from [7].

The model used to generate [6] was developed in MATLAB Simulink and developed, tested, and verified over many years of research [5], [7]–[10]. This model is characterized by a combined diesel electric and gas (CODLAG) NPS and is illustrated in Figure 1 [5], [7]. The generated dataset is comprised of 589,223 observations with 25 features and five decay-state coefficients for each observation.

In this model, multiple components and subsystems of an NPS are simulated to model the performance decay of the GT, GT compressor (GTC), hull (HLL), and propeller (PRP). The GTC degradation was simulated by adjusting the airflow rate through the compressor and recorded as the degradation coefficient kM_C [5]. The degradation of the GT component represents the air flow rate through the engine and is recorded as the degradation coefficient kM_T [5]. The HLL experiences degradation due to corrosion and fouling of the hull and is recorded as the coefficient kH [5]. The PRP degradation represents the roughness in the blade surface and is recorded by increasing the torque coefficient kK_q and reducing the thrust coefficient kK_t [5]. Since the two PRP coefficients are related linearly by $1-kK_t = kK_q - 1$, only kK_t is analyzed [5]. The decaying variables kK_t, kH, kM_C, and kM_T vary in a range that is expected to be seen in a period of two years of operation [5]. The range of decay is at a "degree of precision sufficient to have a good granularity of representation" [5]. Each decay range is represented by 15 states, inclusive. These variables and their decay value ranges are listed in Table 1.

Table 1. Output decay values. Adapted from: [5].

#	**Outputs**	**Range**
1	Propeller Thrust decay state coefficient	$kK_t \in [0.9,1.0]$
2	Hull decay state coefficient	$kH \in [1.0,1.2]$
3	GTC decay state coefficient	$kM_c \in [0.95,1.0]$
4	GT Turbine decay state coefficient	$kM_t \in [0.975,1.0]$

The features of each observation are represented as sensor data from the ship and the NPS. All the sensor data present in this dataset is readily available for collection on the frigates on which the simulated model is based [5]. These features are listed in Table 2.

Table 2. Dataset feature list. Source: [5].

#	**Feature name**	**Units**
1	Lever (lp)	[#]
2	Speed	[knots]
3	Gas Turbine shaft torque (GTT)	[kN m]
4	Gas Turbine Speed (GT rpm)	[rpm]
5	Controllable Pitch Propeller Thrust stbd (CPP T stbd)	[N]
6	Controllable Pitch Propeller Thrust port (CPP T port)	[N]
7	Shaft Torque port (Q port)	[kN]
8	Shaft rpm port (rpm port)	[rpm]
9	Shaft Torque stbd (Q stdb)	[kN]
10	Shaft rpm stbd (rpm stbd)	[rpm]
11	HP Turbine exit temperature (T48)	[C]
12	Gas Generator speed (GG rpm)	[rpm]
13	Fuel flow (mf)	[kg/s]
14	ABB Tic control signal (ABB Tic)	[%]
15	GT Compressor outlet air pressure (P2)	[bar]
16	GT Compressor outlet air temperature (T2)	[C]
17	External Pressure (Pext)	[bar]
18	HP Turbine exit pressure (P48)	[bar]
19	TCS tic control signal (TCS tic)	[%]
20	Thrust coefficient stbd (Kt stbd)	[]
21	Propeller rps stbd (rps prop stbd)	[rps]
22	Thrust coefficient port (Kt port)	[]
23	Propeller rps port (rps prop port)	[rps]
24	Propeller Torque port (Q prop port)	[Nm]
25	Propeller Torque stbd (Q prop stbd)	[Nm]

A diagram of the GT components is shown in Figure 2. The lever position (feature 1) controls the ship's speed (feature 2) and is the only user input into this system. TCS and ABB turbine injection control (TIC) (features 14 and 19), relate to fuel in that TIC "is the control signal (i.e., percent of fuel flow) for the propulsion engine" [7] (TCS and ABB not

defined in [5], [7]) and in turn have an effect on fuel flow (feature 13). In an LM2500 engine, air first enters the GT at the compressor, where the air is pressurized (feature 15) and heats up (feature 16). Fuel is added to the compressed air (feature 13), and the combustion creates even higher pressure that flows through the high power (HP) turbine where pressure (feature 18) and temperature (feature 11) are recorded. The HP turbine powers (rotates) the gas generator shaft, measured as rotations per minute (RPM) (feature 12). The gases leaving the HP turbine flow into the low power (LP) turbine where torque (feature 3) and speed (feature 4) are monitored. The LP turbine connects to the ship's main reduction gear (MRG) where power is delivered to the two shafts.

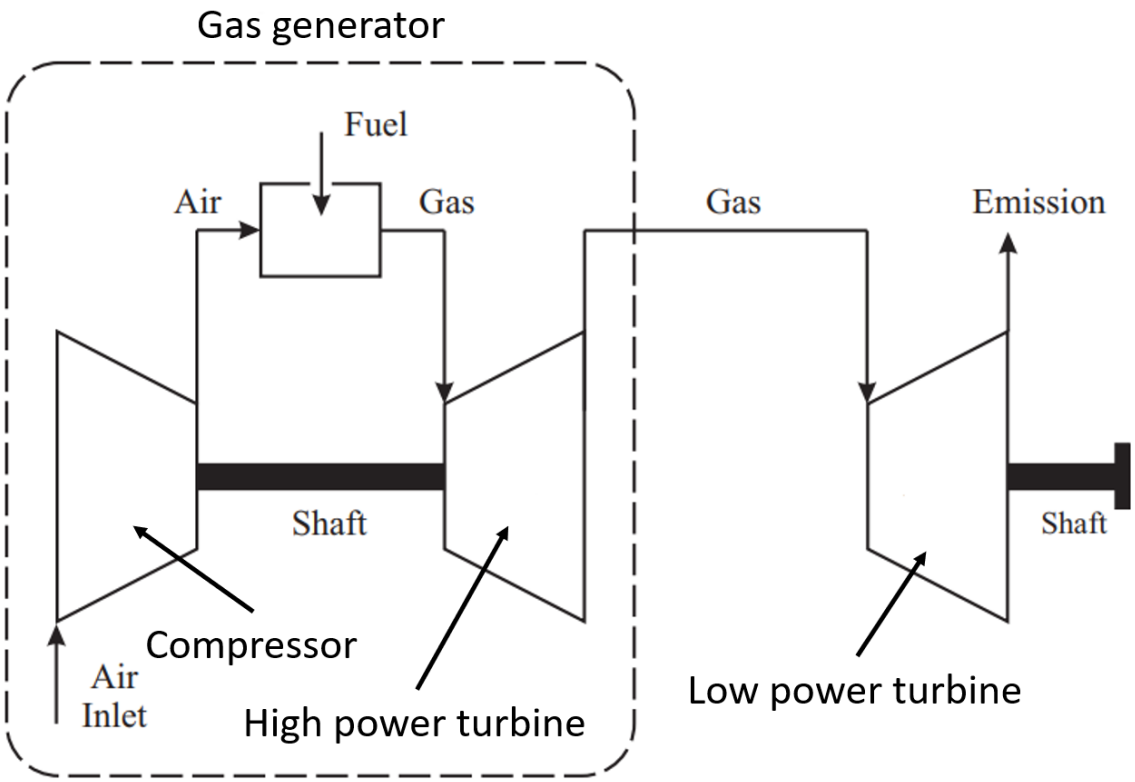

Figure 2. GT diagram. Adapted from [7].

The main shafts leaving the MRG run from the engine rooms to the stern of the ship where they connect to each controllable pitch propeller (CPP). Features 7 through 10 are measurements from the port and starboard main shafts. The port and starboard CPPs'

thrust, torque, and rotations per second (RPS) are recorded in features 5, 6, 21, 23, 24, and 25. Features 17, 20, and 22 are used in developing the ML model but are not clearly defined in reference [5].

B. MACHINE LEARNING

One of the earliest pioneers of ML is Arthur Samuel who, while working for IBM in 1959, developed a computer program that bested him in checkers. He defined ML as a "field of study that gives computers the ability to learn without being explicitly programmed" [11]. This old definition still applies, although a more recent and specific definition came from Dr. Tom Mitchell, the Machine Learning Department Head at Carnegie Mellon University, who in 2006 said, "To be more precise, we say that a machine learns with respect to a particular task T, performance metric P, and type of experience E, if the system reliably improves its performance P at task T, following experience E." [12] Daniel Faggella, the founder of the website TechEmergence.com consulted several sources, including university researchers in the field, on the definition of ML; "Machine Learning is the science of getting computers to learn and act like humans do, and improve their learning over time in autonomous fashion, by feeding them data and information in the form of observations and real-world interactions" [13].

ML can be broken down by the style of learning and the type of problem to solve. Two major ML styles are supervised and unsupervised learning; two common ML problems are regression and classification, both of which fall into the supervised learning style. In supervised learning, the ML "algorithm is given training data which contains the "correct answer" for each example" [11]. In this case, the ML algorithm learns from experiences where the data is already labeled for the task. For example, for a loan application scenario, the input may be an applicant's credit history and the associated label whether the applicant qualifies for a loan or not. In unsupervised learning "The algorithm looks for structure in the training data, like finding which examples are similar to each other, and then groups them in clusters" [11]. In this case, the ML algorithm is fed the same data and multiple applicants' credit histories but without any loan qualification data, and the algorithm attempts to group the individual histories based on the data. In this case,

further research may be needed to determine the validity of those groups because the ML algorithm does not provide that information. In the ML classification problem, “the answer to be learned is one of finitely many possible values” [11]. In our previous example, the “is qualified” or “is not qualified” for a home loan are two discrete answers. The second ML problem, regression, is “where the answer to be learned is a continuous value” [11]. Instead of the label being qualified or not qualified for a home loan, the label to each credit history would be their corresponding FICO credit score.

The terms observations, features, labels, training, predictions, and performance are used consistently in this thesis. Observations relate to Mitchell’s definition as the experience E and, in the credit example, each individual credit history is one observation [12]. The known correct answers, or labels, are the factual data that correlate to each observation, the data showing FICO scores or status of the loan qualification. The task T is the ML algorithm training on the labeled data of each observation. This can range from hundreds to hundreds of thousands of observations. Prediction is the ML algorithm output from new data after it has gone through the training phase. The performance metric P is the error of the ML algorithm when tested with new data. The last term we use is features, which are the individual elements that make up one observation. In the credit example, features may include total available credit, outstanding debt, late payments, number of credit accounts, etc.

The dataset [6] we use for this thesis has already been labeled with degradation coefficients. These responses were generated via a simulated model and are, therefore, finite; however, this is a regression problem because the responses represent a continuous range. Cipollini, Oneto, Coraddu, Murphy, and Anguita, in previous work [5], tested multiple supervised ML algorithms such as SVM, kernelized regularized least squares, shallow and deep neural networks, and others. SVMs are known for their ability with multivariate functions and being highly accurate as universal approximators [14]. These characteristics make them a good choice for modeling a complex, nonlinear, unknown process like that considered in this thesis [14]. The complexity and nonlinearity of this dataset and the need for accurate predictions are the reasons behind the SVM approach focus presented in this thesis.

C. SUPPORT VECTOR MACHINES

SVMs belong to a class of supervised ML methods that nonlinearly transform the training data input into a higher dimensional feature space and create a mapping function that predicts the output of new data for either classification or regression problems [15], [16]. One of the main strengths of SVMs lies in their ability to map non-linearly separable input features into a higher dimensional space where transformed features become linearly separable [15]. One of the main advantages of the SVM is that transformed features are never explicitly computed. Instead, transformed features appear only as inner products in the optimization process using the kernel function K defined as

$$K\left(\vec{x}_i, \vec{x}_j\right) = \Phi^T\left(\vec{x}_i\right)\Phi\left(\vec{x}_j\right). \tag{1}$$

In (1), Φ is the mapping function that transforms the input data $\vec{x}_i$ into a different and often higher dimension. The variable $\vec{x}_j$ is a vector that the kernel function compares with $\vec{x}_i$. Not having to conduct the mapping explicitly to conduct operations is known as the kernel trick [14]. A selection of popular kernels is provided in Table 3.

Table 3. Common kernels. Adapted from: [14].

Kernel Functions	Type of Classifier
$K\left(\vec{x}_i, \vec{x}_j\right) = \vec{x}_i^T \vec{x}_j$	Linear, dot product
$K\left(\vec{x}_i, \vec{x}_j\right) = \left[\left(\vec{x}_i^T \vec{x}_j\right) + 1\right]^d$	Complete polynomial of degree d
$K\left(\vec{x}_i, \vec{x}_j\right) = \exp\left(-\left\|\vec{x}_i - \vec{x}_j\right\|^2 \Big/ 2\sigma^2\right)$	Gaussian
$K\left(\vec{x}_i, \vec{x}_j\right) = \tanh\left[\left(\vec{x}_i^T \vec{x}_j\right) + b\right]$	Sigmoid

With nonlinear data transformed into a feature space where transformed features are linear, we can work with linear methods for classification SVMs. In a classification scenario, SVMs create a decision boundary between separable data. This boundary is referred to as the hyperplane [14]. With any set of separable data, there are multiple

decision boundaries that can be created. The optimal boundary is the one with the largest margins between the two sets of data, as illustrated in Figure 3 [14]. In the plot on the left, the margins are narrow and future data is more likely to be misclassified. In the plot on the right, the margins are as wide as can be; it is observed that this is a more accurate representation of separating the data than the plot on the left. During training, the SVM will select the decision boundary with the widest margins [14].

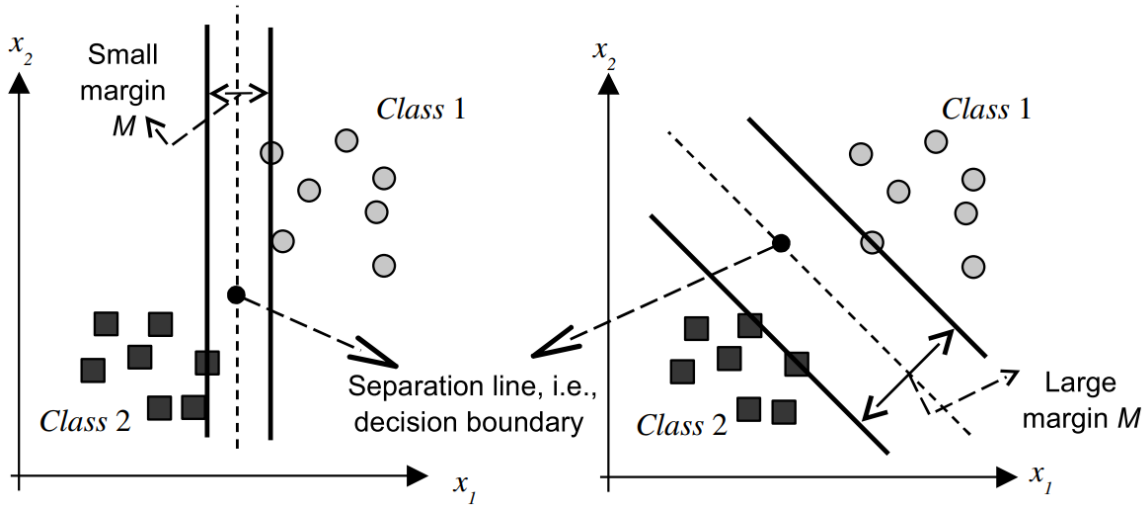

Figure 3. Different SVM decision boundaries. Source [14].

Not all data is perfectly separable. The previous example is known as a hard margin classifier, where the decision boundary and the margins perfectly separate the two classes [14]. A soft margin classifier allows for some data outliers to fall within the margins or even cross the decisions boundary [14], as shown in Figure 4. The size of the soft margin is controlled by the penalty parameter C [14]. Increasing C leads to a smaller margin and fewer misclassifications of training data and vice versa [14]. The tradeoff is that the decision boundary tends to overfit the data and can lead to the SVM being unable to generalize new data well when a small margin is selected [14].

Regression SVMs (SVRs) use similar concepts of margin. SVRs, however, are designed to find a best fit for the data instead of best separation between data. The decision boundary hyperplane becomes a prediction function hyperplane for regression problems [14]. A new parameter, called the insensitivity ε, is introduced to control how accurately

the prediction hyperplane follows the data [14]. The insensitivity creates what is known as an ε-tube, i.e., a tube around the hyperplane in which data points are not penalized nor do they affect the shape of the hyperplane. The SVR also allows for outlier data outside the ε-tube. The distance of this data from the edges of the ε-tube to a data point is the distance ξ and is penalized by the penalty parameter C [14]. Note that the same variable name C is used in both the classification and regression algorithms for differing uses. Further use of the penalty parameter C in this thesis is in reference to the regression parameter unless specified otherwise. The variables ε, C, and σ (for the Gaussian kernel) are the hyperparameters we modify to optimize the regression SVM model in this thesis.

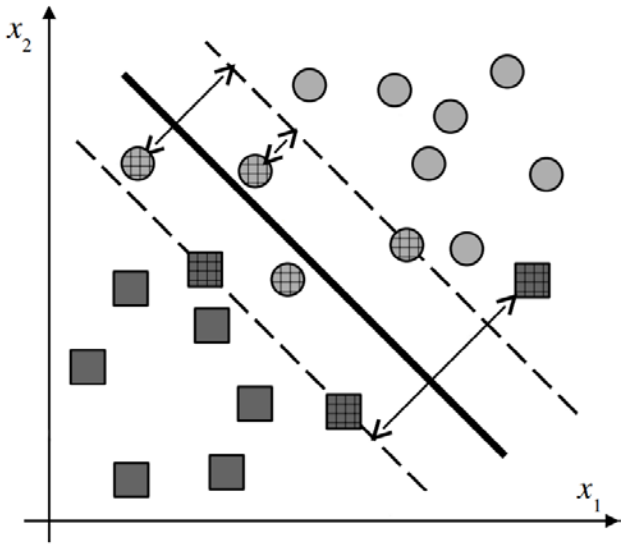

Figure 4. Soft margin SVM example. Adapted from [14].

The optimization process of an SVR that forms a final prediction hyperplane involves several complex steps. In this thesis, we do not modify the standard methods in which an SVR forms a prediction hyperplane and do not cover the details of that process. Specific details of the SVM and SVR processes can be found in [14] and [16].

We express the dataset as

$$D = \{(\vec{x_1}, y_1), (\vec{x_2}, y_2), \ldots, (\vec{x_m}, y_m)\}, \tag{2}$$

where D is the input data, $\vec{x_i}$ is an n-dimensional real vector where $\vec{x_i}$ represents a single observation, and n represents each feature of that observation. The variable y_i represents the corresponding label, for a total of m observations [6], [14], [16]. The dataset D can be split into a training, validation, and testing data subsets such that $\{D_{train}, D_{validate}, D_{test}\} \in D$. In binary classification problems, the label values are usually either 1 and 0 or 1 and -1, while in regression problems, the label is defined in a continuous range [15].

An example of a two-dimensional regression SVM is illustrated in Figure 5. The solid green line represents the hyperplane developed by the SVM, and the various boxes represent data points from the training dataset [14]. The purple outlined boxes are points within the ε-tube, the solid blue boxes are data points on the border of the tube; these form the free support vectors. The red solid boxes outside the tube, a distance measured as ξ, form the bounded support vectors [14]. The free and bounded support vectors are used in forming the weight vector $\vec{w}$ which is used to shape the predicted hyperplane [14].

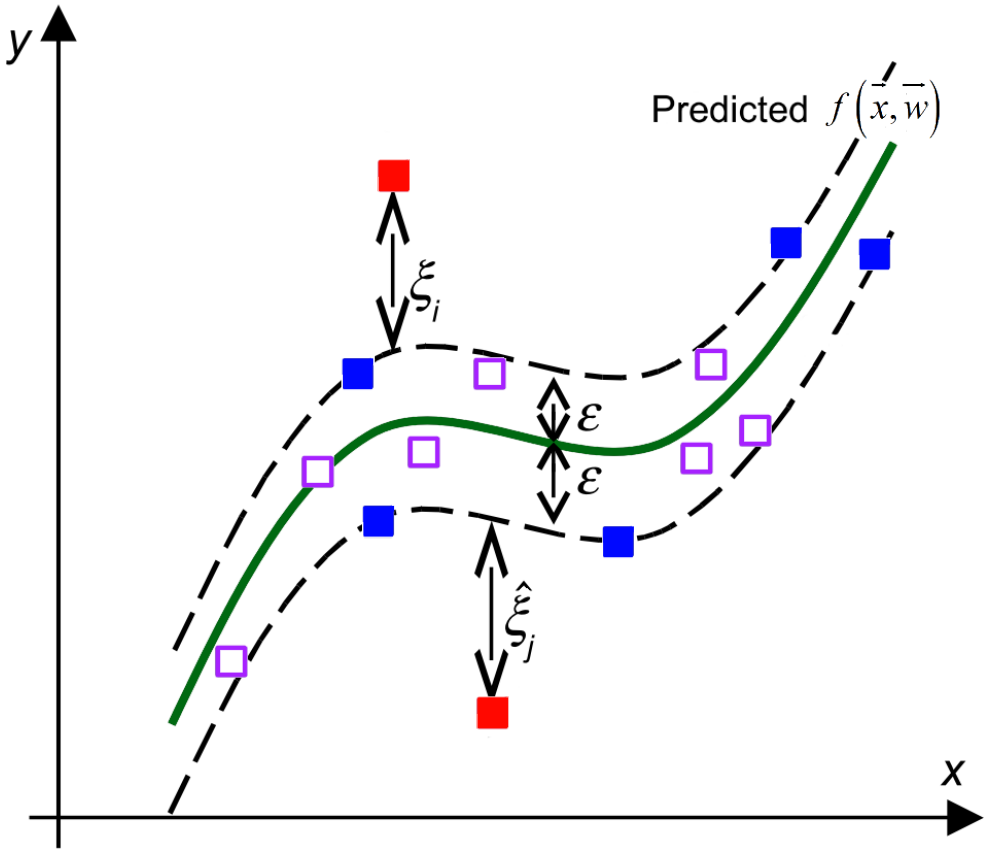

Figure 5. An SVM hyperplane and its margins. Adapted from [14].

THIS PAGE INTENTIONALLY LEFT BLANK

III. PROCEDURE

In this section we analyze the dataset and then plan out the procedure for selecting the optimum hyperparameters for our dataset. We then train the algorithm with final hyperparameters across a range of training set sizes. We hope to prove the ability of an SVR to accurately predict the decay states of the major components of an NPS.

A. DATASET ANALYSIS

The first step in creating an ML algorithm is data analysis. During this step, we familiarize ourselves with the data, look at correlations in the data, and identify redundant data. Because the lever position and speed create dramatic changes between the different features, this analysis is conducted at each lever position separately.

During our analysis we identified some replication of data, specifically in the data that was represented in features that had both port and starboard representation. Of the duplicated data, the replications of the port side were removed and the data simplified to one feature: CPP thrust (features 5 and 6), shaft torque (features 7 and 9), shaft RPM (features 8 and 10), thrust coefficient (features 20 and 22), propeller RPS (features 21 and 23), and propeller torque (features 24 and 25). The features lever position and speed are also directly linearly correlated. The speed and lever position are related by $speed = 3lp.$ For this reason, we removed the speed feature and only used the lever position in our training. As previously mentioned in Chapter II, the decay state coefficients for PRP thrust and PRP torque are linearly related by $1 - kK_t = kK_q - 1;$ thus, the PRP torque decay state coefficient was removed from the dataset. The final reduced dataset is represented in Table 3.

That last step in data analysis is to normalize the data. This normalization step can be accomplished using MATLAB's regression SVM training function; however, since the data is used in many training evolutions, we chose to do it separately. Normalization was conducted for each feature by removing its mean and dividing by its standard deviation.

Table 4. Reduced feature set used for training SVM.

#	Feature name	Abbreviation	Units
1	Lever position	(lp)	[#]
2	Fuel flow	(ff)	[kg/s]
3	ABB TIC signal	(abb_tic)	[%]
4	TCS TIC signal	(tcs_tic)	[%]
5	External pressure	(ext_p)	[bar]
6	GT compressor outlet air pressure	(gtc_p)	[bar]
7	GT compressor outlet air temperature	(gtc_temp)	[C]
8	HP turbine exit pressure	(hpt_p)	[bar]
9	HP turbine exit temperature	(hpt_temp)	[C]
10	Gas Generator torque	(gtc_t)	[kN m]
11	Gas Generator speed	(gg_rpm)	[rpm]
12	Power turbine speed	(pt_rpm)	[rpm]
13	Shaft rpm	(shaft_rpm)	[rpm]
14	Shaft Torque	(shaft_t)	[kN]
15	CPP RPM	(cpp_rpm)	[rpm]
16	CPP torque	(cpp_t)	[Nm]
17	CPP thrust	(cpp_T)	[N]
18	Thrust coefficient	(T_coef)	[]

B. TRAINING AND MODEL SELECTION

MATLAB provides several functions for training various ML algorithms including regression SVMs. For training, we used the function *fitrsvm* to create a regression SVM model. This model was then used with the function *predict* to return predictions for a set of observations $\vec{x}$. MATLAB allows for various changes to the training process using name-value pair arguments. The following name-value pair arguments were used for adjusting the hyperparameters during the training process of the regression SVM: the penalty factor C with *"BoxConstraint"*, the selection of the kernel with *"KernelFunction"*, and the insensitivity ε with *"Epsilon"*. MATLAB's Gaussian kernel is defined as $K\left(\overrightarrow{x_j},\overrightarrow{x_k}\right)=\exp\left(-\left\|\overrightarrow{x_j}-\overrightarrow{x_k}\right\|^2\right)$ [17]. Note that MATLAB's Gaussian kernel definition is different from that provided in Table 2; MATLAB's kernel omits the factor of $2\sigma^2$ which does not provide the ability to modify the kernel parameter. MATLAB does

allow the use of custom kernels to be programmed. Since σ is a hyperparameter we optimize in this thesis, we programed our own Gaussian kernel defined as

$$K\left(\overrightarrow{x_j}, \overrightarrow{x_k}\right) = \exp\left(-\left\|\overrightarrow{x_j} - \overrightarrow{x_k}\right\|^2 \Big/ 2\sigma^2\right) \tag{3}$$

that allows for the adjustment of σ. To enable the automatic adjustment of the parameter σ in our custom kernel function, another MATLAB script was coded to open the file containing the kernel function and change the text on the line containing the sigma value.

When developing a ML algorithm, a performance metric must be chosen to measure the algorithm's performance. For regression problems, two metrics are commonly used to measure loss, the absolute loss and squared loss [7] defined, respectively, as

$$\text{Absolute loss: } \ell_1\left(f\left(\vec{x}\right), y\right) = \left|f(\vec{x}) - y\right| \tag{4}$$

and

$$\text{Squared loss: } \ell_2\left(f\left(\vec{x}\right), y\right) = \left(f(\vec{x}) - y\right)^2. \tag{5}$$

In (4) and (5), y is the labeled value of the data, and $f\left(\vec{x}\right)$ is the predicted output. In [7], the authors chose to measure the loss by mean absolute percentage error (MAPE) defined as

$$MAPE = \frac{100}{m}\sum_{k=1}^{m}\left|\frac{y_k - f\left(\overrightarrow{x_k}\right)}{y_k}\right|. \tag{6}$$

In (6), m is the number of observations predicted y_k, and $f\left(\overrightarrow{x_k}\right)$ are the labeled and predicted values, respectively, corresponding to the k^{th} observation. We use the same performance metric so we can compare results with previous work [7].

1. Training and Validating

In ML, it is common to split a dataset into subsets for training and validation. A trained model will naturally perform well when making predictions using the same data set

with which it was trained. Having a separate validation subset provides us with the ability to test the model on "new" data on which it was not trained, providing a more valuable performance metric. The validation subset is used to optimize the hyperparameters.

When limited training data is available, it is common to split the data between training and validation with common ratios around 7/10 and 3/10, respectively. This type of validation is known as holdout validation [18], [19]. This dataset is large enough that we do not need to holdout observations for validation. To ensure a consistent and accurate performance metric, all validation steps were conducted with 1,000 observations, $n_{validate} = 1000.$

The selection of data used to train each model can have dramatic influence in the performance of that model. If optimal data were selected by chance, the model could have abnormally good performance. The reverse could be true as well. For the performance of each model to be statistically significant, and in order to minimize the effect of abnormalities, we replicated each training and predicting session 30 times. Each of the training procedures was conducted with a different random permutation of the entire dataset before separating into the subsets for training and validating.

2. Selecting Optimum Hyperparameters

The SVR discussed in Chapter II has three hyperparameters that can be adjusted to modify the training process of a regression SVM. They are the penalty factor C, the insensitive ε, and the hyperparameter for the Gaussian kernel σ. Optimizing these hyperparameters is known as model selection.

Tuning the hyperparameters requires training an SVR model at various values for each hyperparameter and analyzing the training and validation performance results. For this process, the training subset is kept constant throughout. The hyperparameter is varied across a range of acceptable values, and the performance, in this case the MAPE, is plotted for the training and validation subsets.

A hypothetical example is shown in Figure 6. In this figure, the red curve P_t and the blue curve P_v are the performance of the training and validation subsets, respectively.

The region on the left, where P_t and P_v are larger and similar, is known as a high bias problem [19]. This behavior occurs when the model does not generalize well and is known as underfitting [19]. The region on the right, where $P_t << P_v$, is known as a high variance problem [19]. This behavior occurs when the model fits the training data extremely well but does not generalize well to new data and is known as overfitting [19]. The optimal setting for a given hyperparameter is when both P_t and P_v are small and before P_v starts to diverge from P_t.

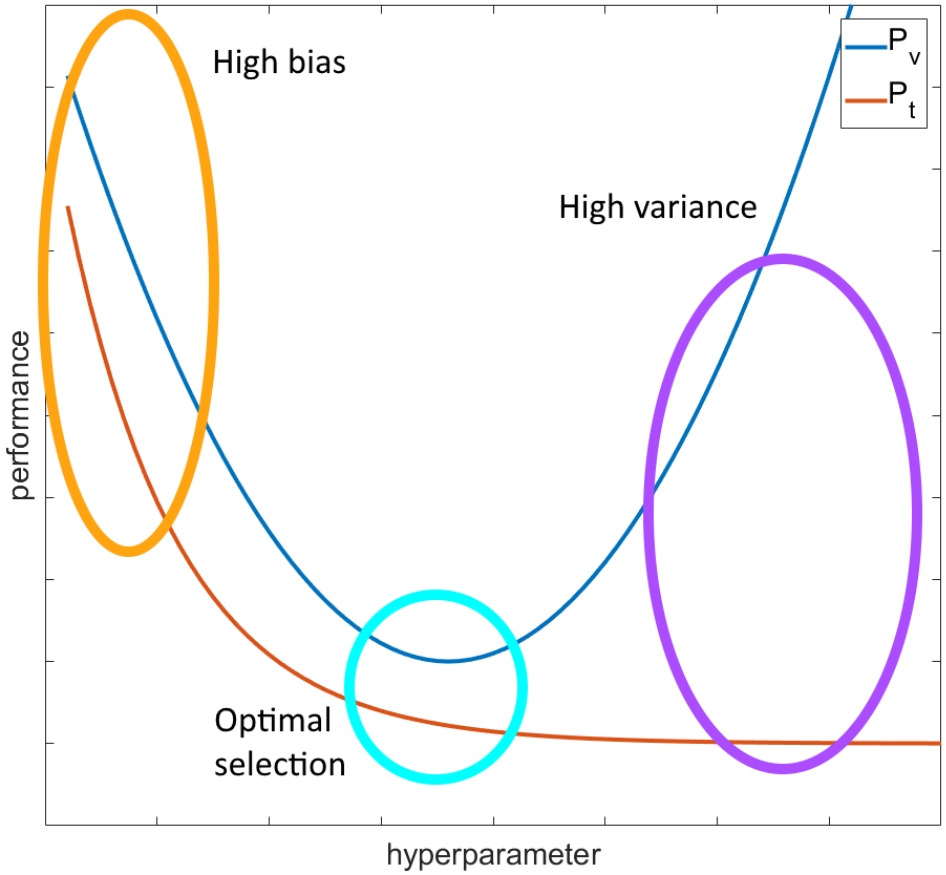

Figure 6. Bias and variance

We optimize the hyperparameters for our model in two phases. In the first phase, we optimize each hyperparameter individually with the other two hyperparameters set at MATLAB's default settings. We use the validation method described previously for each

hyperparameter with the training observations $n_{train} = 1000$. Since MATLAB's Gaussian kernel does not have the $2\sigma^2$ factor, when varying ε and C, we set $\sigma = 1/\sqrt{2}$ as the default setting. The creation of the regression SVM model is based on all three interdependent hyperparameters. For this reason, in the second phase we vary all three hyperparameter and compare each model's performance. After executing the second phase, we decided to further tighten the parameters and conducted an additional trial for increased resolution. The two trials of the second phase are labeled A and B, respectively. The initial hyperparameter optimization from the first phase provided us with an optimal range to train and validate on.

Table 5. Hyperparameter values

First Phase	
Parameter	range
ε	$10^{-6+\frac{n}{4}}, \quad n = 0,1,\ldots,29$
σ	$10^{-3+\frac{n}{4}}, \quad n = 0,1,\ldots,25$
C	$10^{-6+\frac{n}{4}}, \quad n = 0,1,\ldots,37$

Second Phase (A)	
Parameter	range
ε	$10^{-4+\frac{n}{2}}, \quad n = 0,1,\ldots,7$
σ	$10^{-2+\frac{n}{2}}, \quad n = 0,1,\ldots,7$
C	$10^{-3+\frac{n}{2}}, \quad n = 0,1,\ldots,11$

Second Phase (B)	
Parameter	range
ε	$10^{-4+\frac{n}{4}}, \quad n = 0,1,\ldots,7$
σ	$10^{-1+\frac{n}{8}}, \quad n = 0,1,\ldots,13$
C	$10^{-1+\frac{n}{4}}, \quad n = 0,1,\ldots,13$

3. Final Testing

The final testing was conducted with the optimum hyperparameter values obtained after the validation phase. In order to compare performance metrics with [7], we used the same initial training subset sizes as considered in [5]. The sizes of the training subsets considered were

$$n_{train} \in \{10, 24, 55, 130, 307, 722, 1700, 4000\}. \quad (7)$$

All results were obtained by averaging 30 training and prediction sessions for our performance metric. We selected a test subset size $n_{test} = 1000$.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESULTS AND ANALYSIS

In this chapter, we explore the findings of our work. We first conducted independent hyperparameter training of ε, σ, and C. Using results obtained in that phase, we selected a narrow range over which to vary all three each hyperparameters at the same time. Next, using the best hyperparameter values derived in phase 2, we trained the SVR algorithm across multiple training set sizes. Results show that different training set sizes lead to different hyperparameter values. We briefly discuss these findings at the end of this chapter.

A. VALIDATION RESULTS

The first phase involved selecting one hyperparameter and varying it while setting the other two hyperparameters to MATLAB's default values. We then compared the performance of the generated algorithm on the dataset used for training and on a separate dataset used for validation to find the optimum value for the selected hyperparameter. Because the hyperparameters are interdependent, we selected a narrow range that was used later when varying all three hyperparameters.

Training and validation performances obtained by varying the hyperparameter σ are presented in Figure 7. Results show the optimal value for σ is around 0.3. In addition, results show that larger errors in both the training and validating datasets are obtained for values of σ significantly above 0.3. For values of σ significantly below 0.3, the training error reaches a minimum while the validation error increases, which signifies the algorithm is overfitting. For the second phase we initially use the range $10^{-2} \leq \sigma \leq 10$.

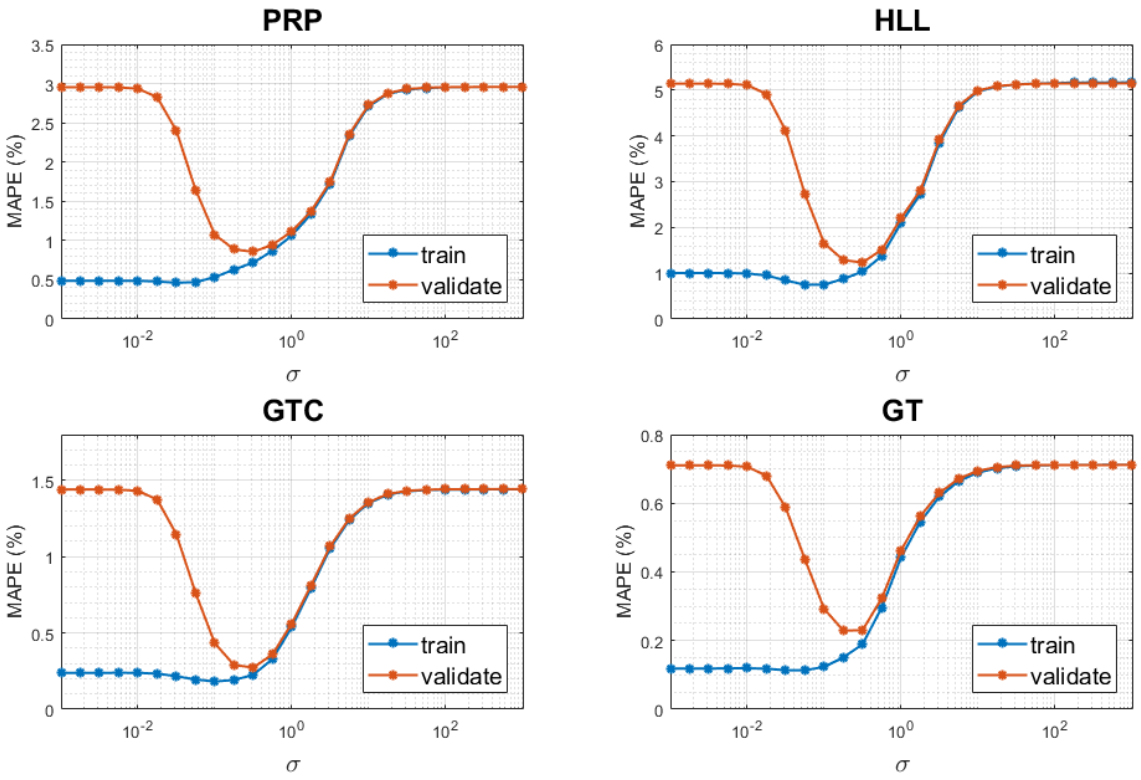

Figure 7. Validation for σ

Training and validation performances obtained when varying the hyperparameter ε are illustrated in Figure 8. In this figure, we see that $\varepsilon > 10^{-1}$ produces relatively larger error values. For $\varepsilon < 10^{-3}$ the error is near its minimum value. It is observed that further decreasing ε does not create an overfitting situation and provides no additional improvement to the algorithm. For the second phase we initially use the range $10^{-4} \le \varepsilon \le 10^{-1}$.

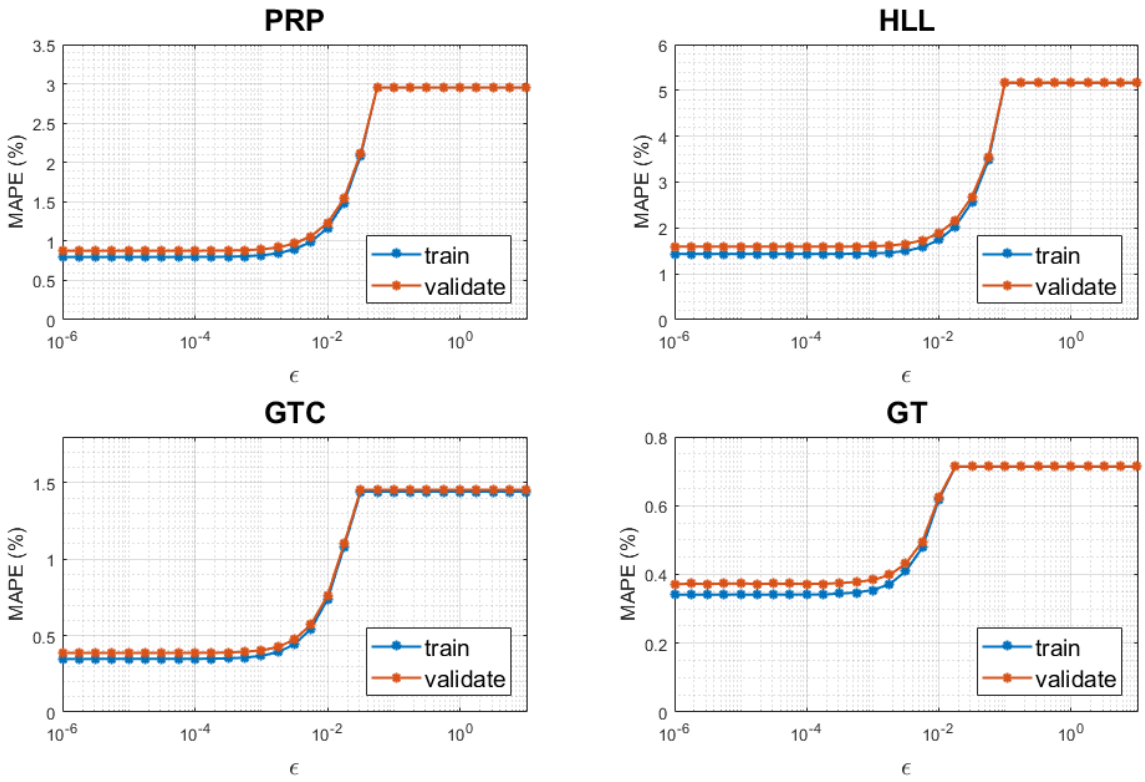

Figure 8. Validation for ε

Training and validation performances obtained when varying the hyperparameter C are illustrated in Figure 9. In this figure, it is shown that for $C<10^{-3}$ the error is relatively high. As C gets larger both the training and validation error values reduce in unison. Error values eventually reach a minimum for $C>10^{2}$. For the second phase we initially use the range $10^{-2} \le C \le 10^{2}$.

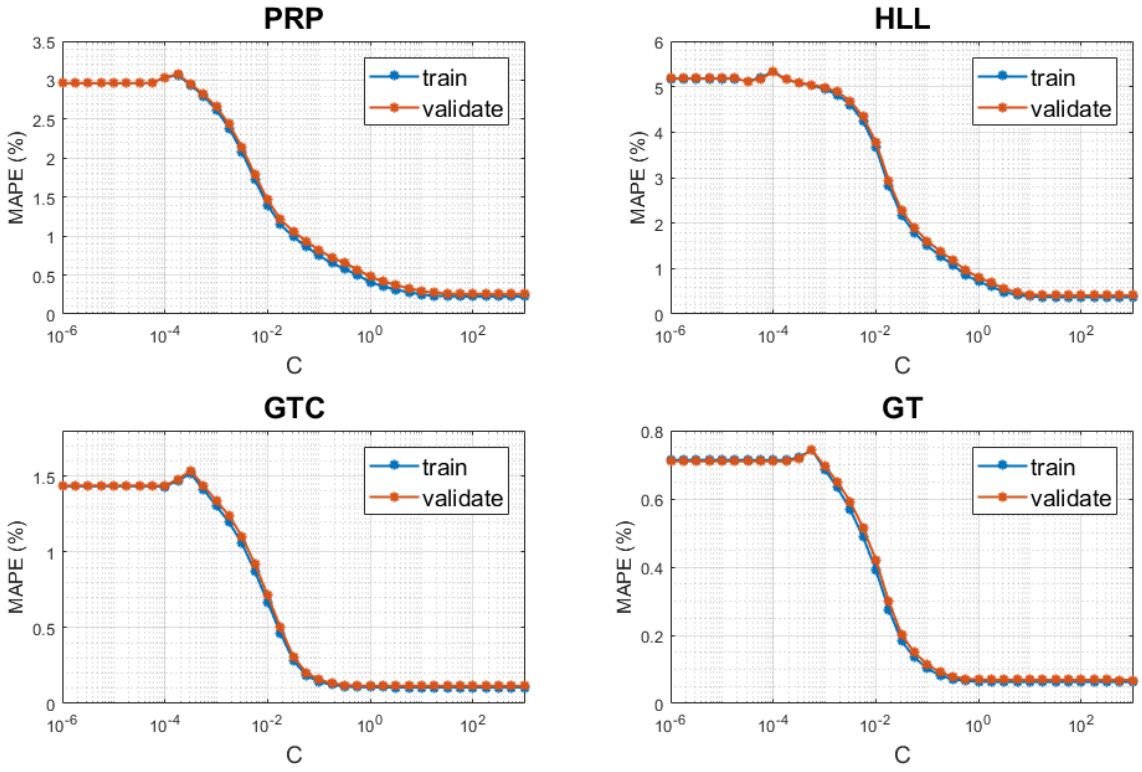

Figure 9. Validation for C

In Figure 10, we see the performance results obtained by varying all three hyperparameters ε, σ, and C for the HLL output. Small error values are shown in red, and large error values are shown in blue. Results illustrated in Figure 10 can be used to narrow the range of values for the hyperparameters to $10^{-1} \le C \le 10^{2}$, $10^{-0.5} \le \sigma \le 10^{0.5}$, and $10^{-4} \le \varepsilon \le 10^{-2.5}$, effectively zooming in to the optimal range. The error performances obtained over these restricted ranges of hyperparameter values are shown in Figure 11.

From Figures 8 and 9, we know that for ε and C the error converges to a minimum value. In Figure 7, we see that varying σ creates an absolute minimum with increasing error as σ moves away from that optimum value.

We used a finer resolution by setting $\sigma = 10^{-1+\frac{n}{8}}$, $n = 0,1,\ldots,13$ to identify its optimal value. These results are illustrated in Figure 12. From here, we can again "zoom

in" on a range to identify the optimum value for σ. This process is repeated for all four output values PRP, HLL, GTC, and GT.

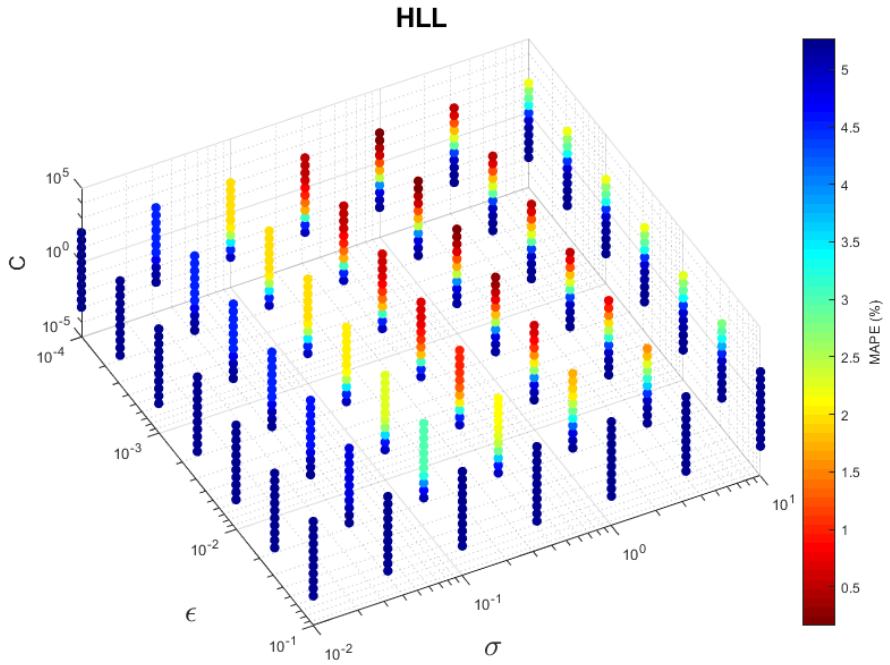

Figure 10. MAPE when varying hyperparameters ε, σ, and C.

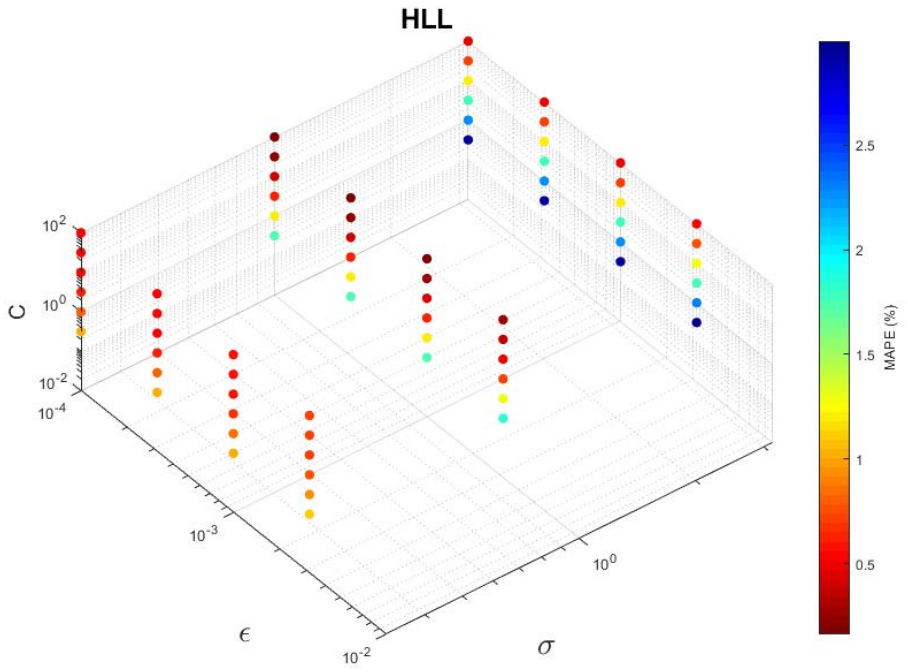

Figure 11. MAPE when varying hyperparameters ε, σ, and C at a selected range.

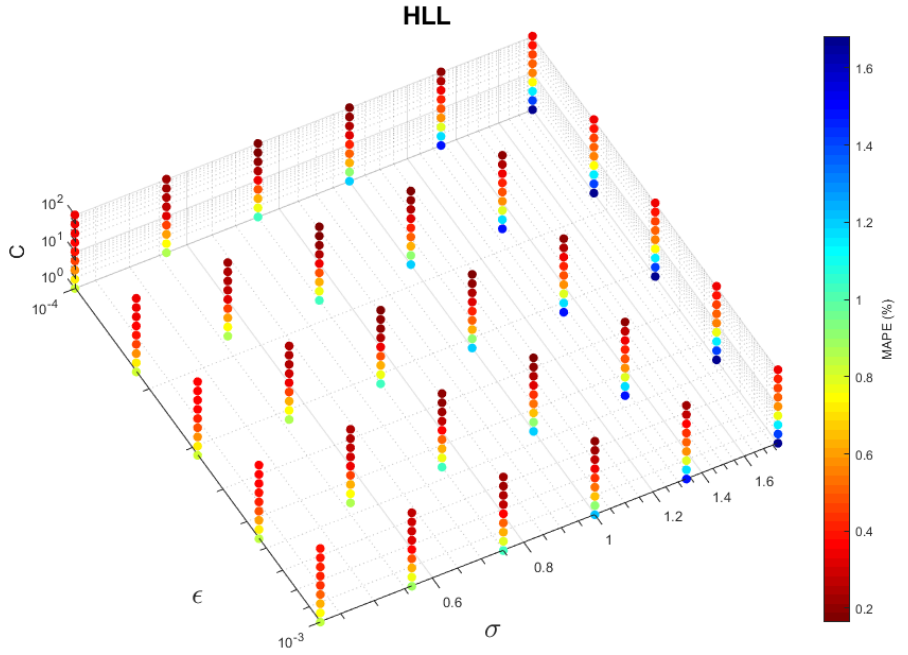

Figure 12. MAPE when varying hyperparameters of ε, σ, and C, with increase resolution.

The estimated optimum values for all the hyperparameters with respect to each output are listed in Table 6. Results show both PRP and HLL had the same optimal σ, which seems logical as both parameters model similar effects on the NPS. The same behavior is observed for GTC and GT as they are also very similar in function.

Table 6. Optimum hyperparameter values

	PRP	**HLL**	**GTC**	**GT**
ε	10^{-4}	10^{-4}	10^{-4}	10^{-4}
σ	1	1	2.3714	2.3714
C	100	100	100	100

As we analyzed ε and C for each output, the difference in error was negligible at the extreme values ε and C. For example, when we hold the other two hyperparameters constant and epsilon lies in the range $10^{-4} \le \varepsilon \le 10^{-3}$, the difference in error remains below 0.1%. This result agrees with the information shown in Figures 8 and 9 where we see that the error reaches a minimum value and no longer decreases. The largest error spread was found when σ varied between $10^{-1} \le \sigma \le 10$, which is observed in Figure 10 and agrees with the shape of Figure 7.

B. FINAL RESULTS

Next, we trained the algorithm with varying training set sizes using the identified optimal hyperparameter values. These different training set sizes were selected to match the training set sizes considered in previous work [5] for accurate comparison. Note that in [5], they only reported their results up to training set of size 377. In Figure 13 we see the comparison of our optimized SVR algorithm along with the SVR and the deep neural network (DNN) algorithms from [5].

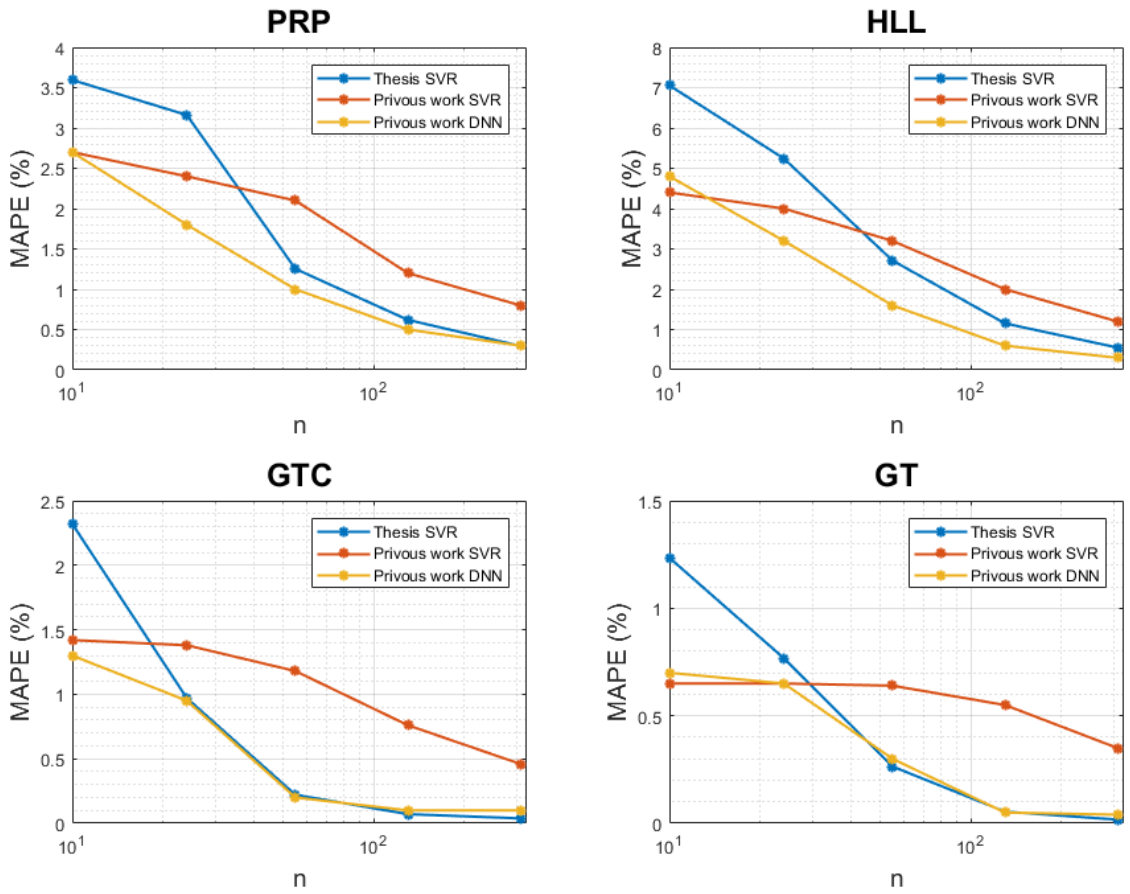

Figure 13. Performance compared to previous work. Adapted from [5].

Our optimized SVR algorithm quickly outperforms the SVR considered in previous work [5] and even matches the best algorithm they tested, the DNN algorithm. In Figure 14, the mean of 30 training sessions is plotted with the standard deviation represented with margin bars. As illustrated in Figure 14, as training set size increases, the algorithm performs exceptionally well. As training set size exceeds 100, PRP error is less than 1% and HLL error is just over 1%. Both GTC and GT error are below 0.5% before this point and begin to converge to 0.1%.

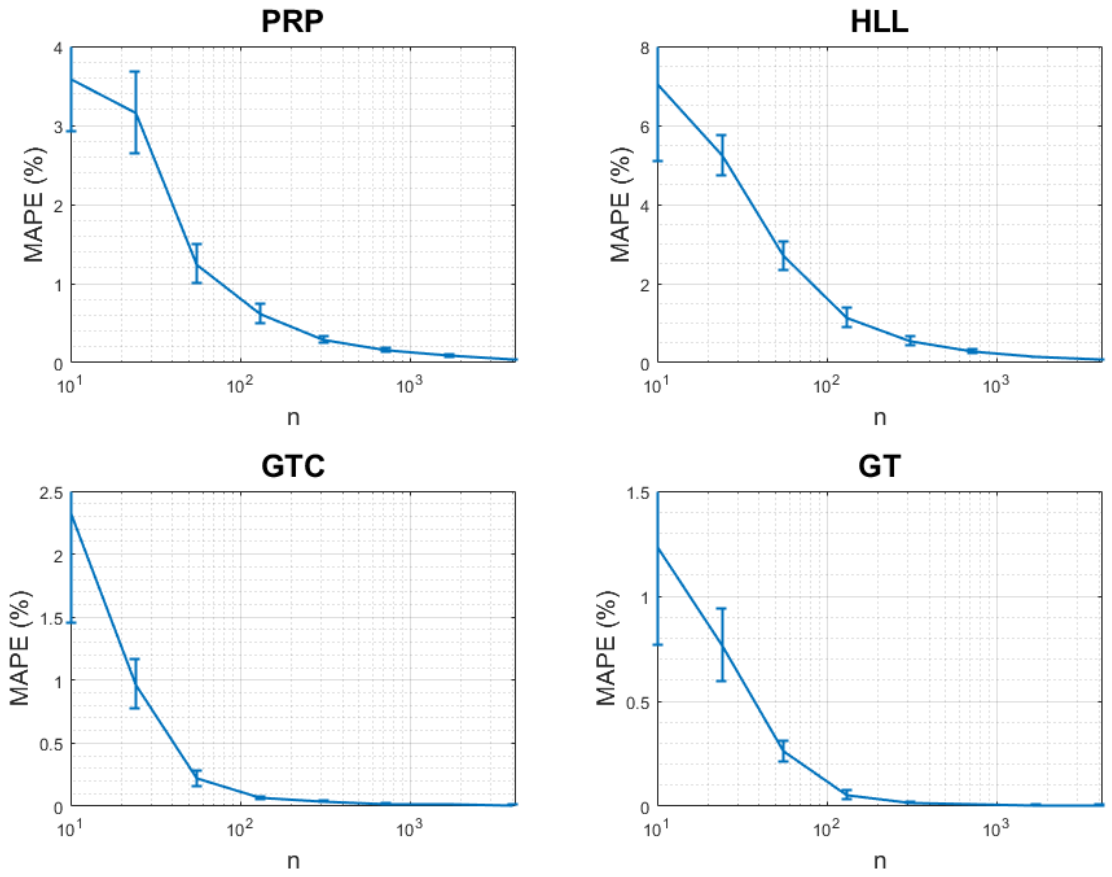

Figure 14. Averaged performance with standard deviation vs. training set size.

From Figure 14, we see that HLL is the output value most difficult to predict by the SVR algorithm followed by PRP. The SVR algorithm does a good job predicting both GTC and GT values with GT being slightly better. These results agree with the results in [5]. Such behavior is possibly due to the makeup of the dataset, as many of the features are related to the gas turbine condition. In contrast, there are few features that are related to the RPR and even fewer with the HLL.

If we look back at Figure 13, we notice that the results from [5] are better than our results at the extremely small training set sizes of $n_{train} = 10$ and $n_{train} = 24$. With the dataset exceeding 500,000 observations, we conducted our hyperparameter optimization with a training size $n_{train} = 1000$. Additional simulation experimentation showed that different hyperparameter values can produce better performance at smaller training sets but do not always excel with larger training sets, as is illustrated in Figure 15. The red line represents

the performance of the SVR algorithm trained with hyperparameters that perform well at small training sets, and the blue line represents the performance of the SVR with hyperparameters listed in Table 6. In Figure 13, we see that our SVR results outperform those obtained in [5] at larger training size configurations. The procedure conducted in this thesis can be followed to improve performance for small training set sizes.

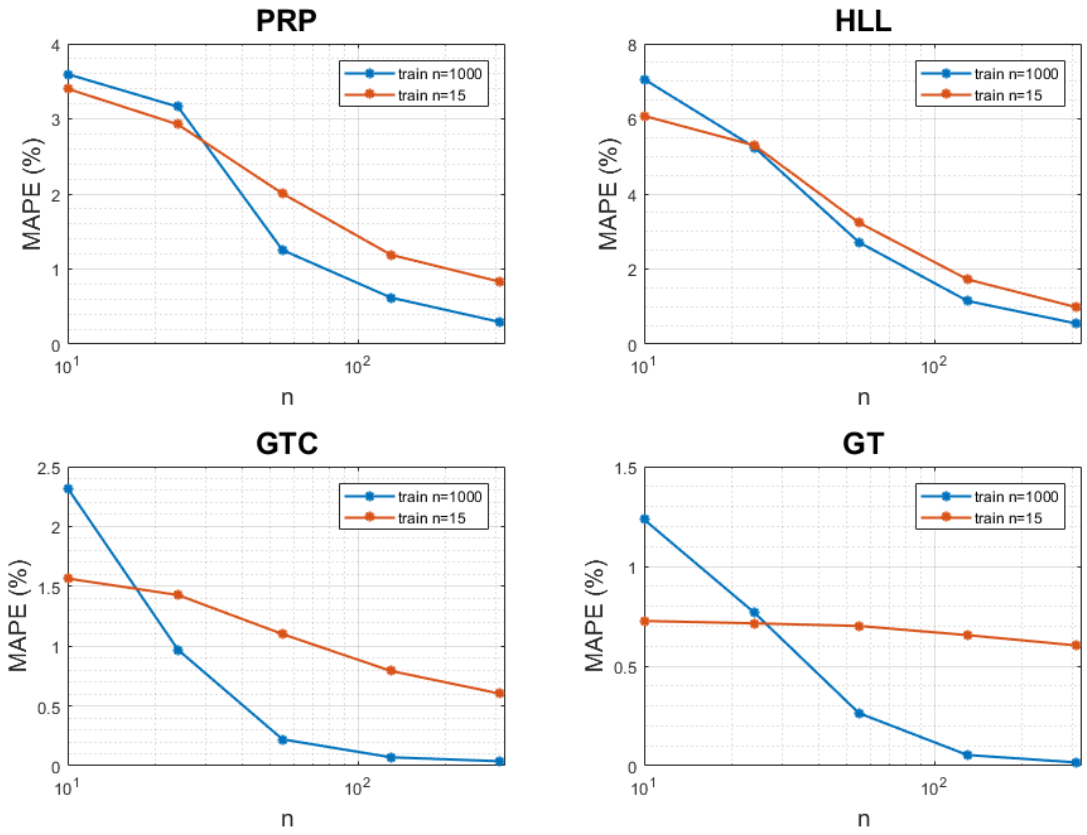

Figure 15. Performance comparison of two SVR algorithms for small and large training set sizes.

V. CONCLUSIONS AND FUTURE WORK

A. CONCLUSIONS

Through an analytical approach and computer simulations, we have shown in this research that an SVR algorithm can make accurate predictions of decay state of major components of an NPS providing the ability to develop a CBM program. With larger training sets we have shown that these predictions can be incredibly accurate, with error rates below 0.5% on all components PRP, HLL, CTC, and GT.

In this work, we analyzed the dataset [6] used in previous work [5] to train and test several different ML algorithms. The dataset had been developed from several years' worth of efforts to create a "real-data validated simulator" of a frigate NPS characterized by a CODLAG propulsion plant [5]. Through our analysis, we found that the dataset represented expected values of sensor readings on a real ship. We found high levels of correlation where we expected there to be, e.g., increase in fuel leads to increase in temperature. We also found that there was some replicated data in that several features were represented with identical port and starboard values. Following data analysis, we trained our SVR algorithm with broad hyperparameter settings to get a general idea of the range that affected our prediction model performances. With those ranges established, we created a test setup that varied all three hyperparameters to identify the optimum setting for each decaying variable PRP, HLL, GTC, and GT. With the optimum values selected for each hyperparameter, we were able to match or exceed the performance recorded in [5]. Finally, we showed that an optimally tuned and trained SVR algorithm can make predictions with error rates less than 0.5%. This evidence shows that an SVR algorithm is a viable option to implement a CBM+ program for the major components of an NPS provided appropriate data is available for its development.

B. FUTURE WORK

Though the initial research objective was met, there are still challenges and recommendations to be researched and implemented before these findings can be applied to ships in the U.S. Navy. First and foremost is that the dataset used in this thesis is modeled

on a CODLAG propulsion plant which the U.S. Navy does not use. Many of the U.S. Navy's surface ships have a combined gas and gas (COGAG) propulsion plant as illustrated in Figure 16; thus, a different simulated model needs to be developed to reflect the conditions of the U.S. Navy or actual data needs to be obtained to make accurate predictions. Additionally, in a COGAG lineup, there could be variations in the port and starboard features (i.e., shaft rpm, shaft torque, etc.) which would make them relevant in this model. In [5] it is mentioned that the ship model uses CPP but the pitch angle is not a feature in the dataset. The U.S. Navy uses controllable reversable pitch (CRP) propellers which are similar but can reverse the direction of thrust for astern propulsion, and the pitch angle could still prove important. U.S. Navy ships also use GT engines for their electric generators. The Rolls-Royce AG9140 type is a GT engine used in many of the U.S. Navy's surface vessels. An additional model and algorithm can be developed for these GT engines as well.

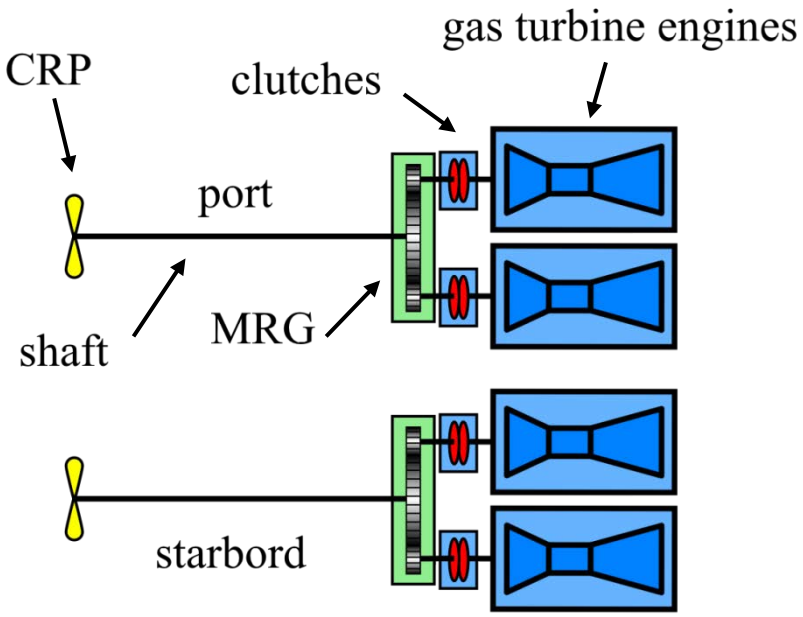

Figure 16. A common U.S. Navy COGAG NPS. Adapted from [20].

GT casualties are an infrequent issue but are possible. Another model can be created for anomaly detection to identify situations in which GTs are likely to experience a casualty. This investigation would likely have to be developed from a set of real data prior to actual GT casualties. Finally, future researchers can reach out to FRCSW, where all the engine overhauls are conducted on the GE LM2500 engine type, for real data they may have collected over the years. Access to such data would be useful in validating a new simulated model as well as be used for anomaly detection ML algorithms in the case of GT casualty prediction.

THIS PAGE INTENTIONALLY LEFT BLANK

SUPPLEMENTAL. MATLAB SCRIPTS AND FUNCTIONS

Included in this section is a list of all supplemental files that are available upon request. To obtain a copy of these files, contact the Dudley Knox Library located on the campus of the Naval Postgraduate School in Monterey, California. The scripts m_removeData.m and m_normalizeData.m removed the excess data and then normalized the data, respectively. The script m_loadData.m is used to load the saved data after it has been normalized for training purposes. The three procedure scripts follow the procedure described in Chapter III. The function f_gaussian1.m is the custom Gaussian kernel and f_editKernelFile.m is the function used to edit the kernel parameter. The function f_SVMR_train_Gaussian.m trains the SVR algorithm, f_SVMR_validate.m conducts the validation sessions, and f_SVMR_test.m conducts the final trainings and testing.

- m_loadData.m
- m_removeData.m
- m_normalizeData.m
- procedure_phaseOne.m
- procedure_phaseTwo.m
- procedure_finalTest.m
- f_gaussian1.m
- f_editKernelFile.m
- f_SVMR_train_Gaussian.m
- f_SVMR_validate.m
- f_SVMR_test.m

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] G. Ziezulewicz, “Navy’s 7th Fleet no stranger to high ops tempo,” *Navy Times*, Aug 21, 2017. [Online]. Available: https://www.navytimes.com/news/your-navy/2017/08/21/navys-7th-fleet-no-stranger-to-high-ops-tempo/

[2] *Condition-Based Maintenance and Condition-Based Maintenance Plus Policy*, OPNAV Instruction 4790.16B, Chief of Naval Operations, Washington, DC, USA, 2015. [Online]. Available: https://doni.documentservices.dla.mil/Directives/04000%20Logistical%20Support%20and%20Services/04-700%20General%20Maintenance%20and%20Construction%20Support/4790.16B.pdf

[3] *Reliability-Centered Maintenance (RCM) Process*, MIL-STD-3034A Department of Defense, Washington, DC, USA, 2014.

[4] Naval Air Systems Command, “FRCSW LM2500 engine program surpasses 40 years of service,” May 29, 2018 [Online]. Available: http://www.navair.navy.mil/index.cfm?fuseaction=home.NAVAIRNewsStory&id=6833

[5] F. Cipollini, L. Oneto, A. Coraddu, A. J. Murphy, and D. Anguita, “Condition-based maintenance of naval propulsion systems with supervised data analysis,” *Ocean Eng.*, vol. 149, pp. 268–278, Feb. 2018. [Online]. doi: https://doi.org/10.1016/j.oceaneng.2017.12.002

[6] A. Coraddu, F. Cipollini, L. Oneto, and D. Anguita, Hull, Propeller and gas turbine efficiency decay: Data analysis with minimal feedback, 2018. [Online]. Available: https://sites.google.com/view/cbm/home

[7] A. Coraddu, L. Oneto, A. Ghio, S. Savio, D. Anguita, and M. Figari, “Machine learning approaches for improving condition-based maintenance of naval propulsion plants,” *Proc. Inst. Mech. Eng. Part M J. Eng. Marit. Environ.*, vol. 230, no. 1, pp. 136–153, Feb. 2016. [Online]. doi: https://doi.org/10.1177/1475090214540874

[8] M. Altosole, U. Campora, M. Martelli, and M. Figari, “Performance decay analysis of a marine gas turbine propulsion system,” *J. Ship Res.*, vol. 58, no. 3, pp. 117–129, Sep. 2014. [Online]. doi: https://doi.org/10.5957/JOSR.58.3.130037

[9] U. Campora and M. Figari, “Numerical simulation of ship propulsion transients and full-scale validation,” *Proc. Inst. Mech. Eng. Part M J. Eng. Marit. Environ.*, vol. 217, no. 1, pp. 41–52, 2003. [Online]. doi: https://doi.org/10.1243/147509003321623130

[10] M. Altosole, G. Benvenuto, M. Figari, and U. Campora, “Real-time simulation of a COGAG naval ship propulsion system,” *Proc. Inst. Mech. Eng. Part M J. Eng. Marit. Environ.*, vol. 223, no. 1, pp. 47–62, 2009. [Online]. doi: https://doi.org/10.1243/14750902JEME121

[11] J. F. Puget, “What Is Machine Learning?,” IBM, May 18, 2016. [Online]. Available: https://www.ibm.com/developerworks/community/blogs/jfp/entry/What_Is_Machine_Learning.

[12] T. M. Mitchell, “The Discipline of Machine Learning,” unpublished. [Online]. Available: http://www.cs.cmu.edu/~tom/pubs/MachineLearning.pdf

[13] D. Faggella, “What is Machine Learning?,” TechEmergence, Sep 19, 2016. [Online]. Available: https://www.techemergence.com/what-is-machine-learning/.

[14] T. M. Huang, V. Kecman, and I. Kopriva, *Kernel Based Algorithms for Mining Huge Data Sets Supervised, Semi-supervised, and Unsupervised Learning*. The Netherlands: Springer, 2006.

[15] H. Yu, “Support Vector Machine,” *Encyclopedia of Database Systems*. Accessed October 30, 2018. [Online] doi: https://doi.org/10.1007/978-0-387-39940-9_557

[16] N. Cristianini and J. Shawe-Taylor, *Support Vector Machines and other kernel-based learning methods*. Cambridge, United Kingdom: Cambridge University Press, 2000.

[17] MathWorks, “Fit a support vector machine regression model—MATLAB fitrsvm,” Accessed October 15, 2018. [Online]. Available: https://www.mathworks.com/help/stats/fitrsvm.html

[18] J. Watt, R. Borhani, and A. Katsaggelos, *Machine Learning Refined: Foundations, Algorithms, and Applications*. New York, NY, USA: Cambridge University Press, 2016.

[19] “Evaluating a learning algorithm ,” class notes for Machine Learning, Stanford University, Stanford, CA, USA, Winter 2018. [Online]. Available: https://www.coursera.org/learn/machine-learning/home/info

[20] “Combined gas and gas,” *Wikipedia*. Apr 4, 2018. [Online]. Available https://en.wikipedia.org/wiki/Combined_gas_and_gas

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California