Predictive Maintenance in a Railway Scenario using One-Class Support Vector Machines

Master-Arbeit
Sandesh Nair

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik

Fachgebiet Knowledge Engineering
Prof. Dr. Johannes Fürnkranz

Predictive Maintenance in a Railway Scenario using One-Class Support Vector Machines
Master-Arbeit
Eingereicht von Sandesh Nair
Matriculation number : 2796574
Tag der Einreichung: 4. October 2016

Gutachter: Prof. Dr. Johannes Fürnkranz
Betreuer: Sebastian Kauschke

Technische Universität Darmstadt
Fachbereich Informatik

Fachgebiet Knowledge Engineering
Prof. Dr. Johannes Fürnkranz

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Master-Arbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in dieser oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Die schriftliche Fassung stimmt mit der elektronischen Fassung überein.

Darmstadt, den 4. October 2016 Sandesh Nair

1 Abstract

In almost any commercial service domain, to be competitive the best and most reliable service must be offered to attract customers. Commercial trains are no exception to this, with reliability being a crucial issue. A locomotive failure is also not just restricted to a single train but the whole network is directly affected. Besides, fixing the component failure later on has added cost overheads of its own. To avoid all this a predictive maintenance approach must be adopted, so that problems can be identified before hand, hence minimizing the risk of actual failures. The goal of this thesis is to predict component failures, specifically power converter failure. For this, we train one-class svm models with multiple parametrisations exclusively on non-failure cases. These models are then validated by testing on a hold-out set to determine the accuracy. If good results are achieved then this approach can be further enhanced to study other types of component failures, with the final aim being developing a prototype which can be used on a real time basis. The experiments show encouraging results, with up to 25.5 % failures predicted correctly with keeping a non-failure prediction rate higher than 90 %.

Contents

List of Figures

List of Tables

2 Introduction

Maintenance of trains is presently done by performing routine periodic checks. In these checks, components of the train are manually inspected and necessary parts replacement is done. Replacements are based on parts failures, general wear and tear, standard maintenance procedures and guidelines, various checklists based on past experiences and component history. Once it is completed, the train is then termed fit for operation.

Unfortunately these checks alone are not sufficient. Some of the components may already be in a deteriorated state and go unnoticed during the inspection. In some cases, for instance, in between two checks there may be unforeseen issues which may result in failures in the immediate future. To handle these failures, corrective maintenance (which is reactive in nature) must be immediately applied to restore the trains' operability.

However, failure of a train component can cause a range of issues. In most cases they result in operational delays which may in turn yield financial losses and loss of potential business in the future. Customers may also lose confidence in the service being offered and in some worst cases a failure can even be catastrophic resulting in loss of life. To avoid all such circumstances a predictive maintenance process must be adopted.

Predictive maintenance will not just help in avoiding failures and unnecessary layovers, but also it has further benefits like, better operational efficiency, reduction in overall maintenance cost and better planning and management of resources and materials available in hand. To achieve this we can use the diagnostics data(events, errors and warnings, status changes etc.) generated by the train‘s components in order to predict an upcoming failure. These predictions can then be further used to make preventive repairs and replacements .

So, predicting a failure is the first key step towards predictive maintenance. But failure cannot be always linked to a specific component as the train has many components and numerous things can go wrong. Many components are interconnected, one component failure can result in multiple other failures. Besides a faulty component, there can be other environmental factors which can equally contribute or act as a catalyst to elevate the problem. Although all of these interconnected and environmental factors are quite relevant, for the sake of simplicity, we can focus on individual components for now and then probably build up the entire picture later on.

Machine learning strategies can be employed to learn component failure patterns by analyzing a combination of diagnostics data of various components based on failure recordings. This learned model can be then tested on other known failure and non-failure cases. Based on whether they were classified correctly or not we can determine the accuracy of the machine learning algorithm in context of that specific problem case. If the accuracy meets the acceptable operational standards, we can then design a prototype to make real time component failure predictions.

In this thesis one such problem case of the power converter failure(PCF) was selected. A power converter is a train component which converts high voltage electricity from the running power cables which can then be used for running the electric motors engines and other train components. Historical diagnostic data of each train is segregated out and organized in a form of a train tour. With a "tour" being of at least 100 kms covered at a stretch and a minimum time duration of 2 hours from powering on the main engine to switching it off. Tours containing power converters failures are separated out. These are termed as "Incident Tours", and the other tours where the functioning was normal is termed as "Non-Incident Tours". Also in the paper, we refer to failures as "Incidents" and non-failures as "Non-Incidents".

For predictive maintenance, we train the one class support vector machine(SVM) exclusively only on non-incident tours. Then this trained model is applied on a variety of tours to understand the accuracy of the learning algorithm. Although the goal is to accurately predict both incident and non-incident cases, any false alarms which indicate failure cannot be tolerated. This is because if a failure is reported, then the train is taken out of service and sent to the workshop, and if nothing is found, besides the wastage of time and money even the users confidence in the system is lost.

So if any incident is predicted, it must be absolutely true. With respect to our experiments we set a threshold of 90% or above accuracy in non-incident prediction. If it passes this threshold, only then to consider the corresponding experiment's incident prediction rate. We had data of total of 340 tours to work on, with 40 of them being incident tours in which power converter failures occurs. In this thesis, we modeled the support vector machine with 43335 combinations of different input data and parameter settings. By running all these experiments we achieved a maximum incident detection rate of 25.5%.

The ground work of this thesis, which includes basics of machine learning and understanding of support vector machines in general and one-class SVM in particular are all explained in following chapters. To avoid data source being accessed directly, we designed a web service using REST, details of which are explained in chapter 8. The detailed experimental setup, along with important one class parameters considerations, is explained later in chapter 9 & 10. The grounds on which threshold and accuracy parameters were established along with the findings of the experiments is described in chapter 11. We finally wind up this report with concluding remarks.

3 Related Work

3.1 One class SVM in other fault detection scenarios

One-class support vector machines an application in machine fault detection and classification [1]: This paper presents the use of one-class SVM for machine fault detection and classification of electro-mechanical machinery using vibration measurements. The objective of this paper is to compare one-class SVM and Artificial Neural Networks(ANN) particularly Multilayer perceptron(MLP). They have taken the benchmark dataset which contains 209 instances. Every instance has two types of information. The first information is about the target value and attributes. The second information is about machine condition classification which can be grouped into 6 classes(Faults). The purpose of one-class SVM in this paper is to construct a model that should suggest the target value in the testing dataset by using only the given attributes. ANN has been regarded as suitable algorithm for this kind of application because of the non-linear properties. By using ANN, they achieved 99.7 % success on the training set and 97.8 % with the test set. When compared to the performance of one-class SVM with kernels (linear, sigmoid, and polynomial), MLP gave relatively good results. However with RBF kernel it consistently gave better performance than ANN.

Fault detection and diagnosis in process data using one-class support vector machines [2] : In this paper, one class SVM and SVM-Recursive feature elimination(SVM-RFE) method was used for fault detection and process data diagnosis. They used the Tennessee Eastman process[3] which is considered to be a benchmark simulation problem in chemical engineering. The plant-wide control structure [4] is used to simulate data. This results were compared with Principal Component Analysis(PCA) and Dynamic Principal Component Analysis(DPCA). One-class SVM algorithm outperformed PCA and DPCA in both diagnosis of data and fault detection. The major drawback of both PCA and DPCA are linearity assumption. On the other had, the efficiency of one class SVM comes by using kernel technique for non-linear classified data.

Power Wind Mill Fault Detection via one-class ν-SVM Vibration Signal Analysis[5] : Vibration analysis is one of the most effective methods on high speed rotating equipment for the early detection of fault thereby avoiding fatal failures. The system design includes alarm detection and diagnoses of failure in four stages that is detection, location, evaluation, and prognosis all of which is to be done in real time. The method should be able to provide continuous likeliness of normality in new captures and should have a minimum number of adjustable parameters. For all these requirements to match, one class SVM is used. With the basic SVM, there is no automatic method, but with one class SVM with RBF kernel and its parameters eases this task to some extend. The results of one class svm are compared to that of Auto-associative Neural Networks(ANN). The ANN obtains good results without noise, however with little noise there is a delay in fault detection. On the other hand one-class SVM has higher accuracy and is more robust under noisy situations.

3.2 Project - Techlok

German national railway company "Deutsche Bahn Schenker Rail" is quite actively working on this idea of failure predictions and preventive maintenance. They have an ongoing project named as "Techlok" which is researching in this direction. All the data required for our work which includes diagnostic data (train components data), failure records including data from work shops part replacement details are provided by Deutsche Bahn. Further than that, we could gain valuable insights by experts knowledge that was provided by DB Schenker Rail employees.

The Knowledge engineering department at Technical university Darmstadt is also been working closely with Deutsche Bahn since the last three years in this regard. Sebastian Kauschke diploma thesis[6] and work done in the following years by the department are the principle ground work for this master thesis. The main key aspects of the work done is briefly explained here in the section below.

3.3 Prediction of Failures based on Diagnostic-Code Frequency

In 2014 a paper was published[7] by knowledge engineering group which highlights the initial progress made in this topic. The approach uses the frequency of certain types of log entries(diagnostic codes) occurrence in a time based window as a measure to identify inconsistencies in the train's behavior and predict failures. The principle hypothesis made was that just before a failure occurs there is an increased occurrence of certain specific diagnostic-codes. Based on this hypothesis using machine learning a technique to predict failures was created.

The classifying algorithms JRip, J48, SMO and RandomForest, were chosen and 5-fold cross-validation was used to verify the results. Precision and recall values of the warning class are the main performance indicators. The main parameters considered in the experiment were days taken into account(v) and number of days before the failure that are labeled as failure(w). The following five experiments were conducted,

- Predicting motor control failure
- Predicting guiding system failure
- Predicting antenna of guiding system failure
- Predicting all three failure types on a combination of data
- Predicting failures using an extended timeframes(Increase of the v and w values)

The results clearly indicate that prediction of failures is indeed possible, however the desired goal of high accuracy is still far. Results show that combined classification as done in experiment 4 is also possible, however performance is much more better when each problem is dealt separately. The data used for this research had huge gaps in the recorded time span which significantly impacted the performance.

3.4 Efforts to improve prediction

Prediction accuracy is highly dependent on the data quality. In 2015, information from workshop repairs and failure reports(hotline data) were used in conjunction with diagnostic data to precisely distinguish different kinds of failures[8]. In consultation with domain experts two specific failure cases; guiding system failure and main air compressor failure were selected based on the frequency of occurrence and the urgency in the need for it to be addressed. Based on these two failures cases, more accurate diagnostic data instances were separated out, yet it cannot be directly used for classification as information can be misleading. So ground truth which can be used to create good labels for classification were reconstructed based on the following criteria.

1. Day of failure.
2. Information on when the train was in the workshop afterwards.
3. List of double repairs of the same component within a few days.
4. List of layovers that were considered unnecessary.

The classifying algorithms JRip, BayesNet and RandomForest were chosen. In some cases the attribute selection methods such as ReliefF, CFSSubset were applied. Two initial experiments were conducted,

1. The prediction experiment
   Goal is to classify instances from the warning period.
2. The unnecessary layover experiment
   Determine whether classifier is able to distinguish instances that belong to a layover with a repair (normal), or to an unnecessary layover

Unfortunately, results of the initial experiments were not encouraging. None of the classifiers were able to predict the upcoming failures(prediction experiment). Neither could they distinguish correct and wrong failures(unnecessary layover experiment). However overall progress was made, a process to determine the ground truth for different failure types was established. Also useful features from the diagnostic data were extracted.

3.5 Motivation

It is quite clear a lot of efforts were put into this idea previously, however the results are still not favorable. There is still a long way to go. A prototype which works in all cases is the goal. As seen above working on individual components have yielded better results. We will also focus on a specific component problem. The work done in this thesis will be just one of the pieces of this jigsaw puzzle.

The approach also is similar to the papers mentioned earlier, however the applied machine learning strategy and the following analysis on the data differ. We are going to train the data using one class support vector machines, which was never used previously. The idea is to model a SVM with different parameters combinations and try to analyze failure patterns.

The data source of this thesis remains the same, the one provided by Deutsche Bahn. Although in this thesis we will focus only on diagnostic data. Also, type of failure we will be analyzing has changed, we will now try to predict power converter failures. In the next chapter we will look into the data source in detail.

4 Data Sources

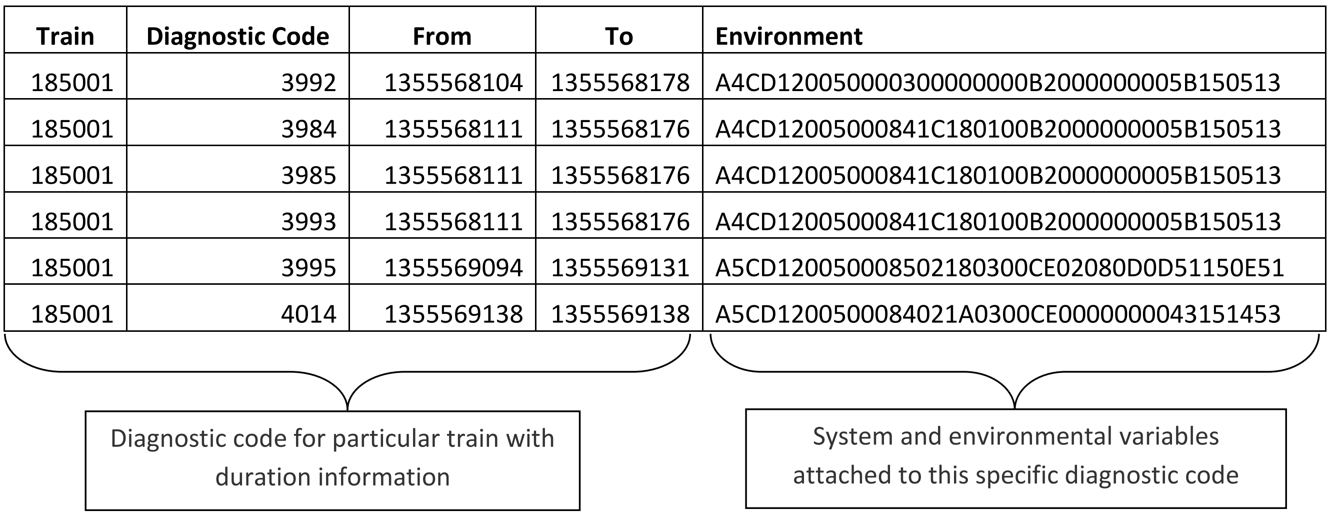
The data source used in this thesis is the same as stated in the previously referenced papers. However, the problem case differs, we deal with only power converter failures.

4.1 Diagnostic Data

Various on-board systems of a train generate continuous logs comprising of events, status changes, errors and other notifications. This is called as diagnostics data. The logging procedure was developed for some different purposes; back then predictive maintenance was not anticipated. The storage capacity on the train is also limited and these logs are then manually collected by a mechanic in the workshop when the train is in for maintenance. Each diagnostic message has a diagnostic-code attached to it, which indicate the particular system and subsystem this code belongs to, timestamps of first and last occurrence, as well as the environment data.

4.1.1 Assigned system variables

The figure below shows a small sample of diagnostic data instances for a specific train. The time range indicates the start and end time of that specific code. Most codes arise only once hence have no time range, while some can occur for a longer period even lasting for days, like code 8703 which indicates the switching on train main engine will last until the train is switched off.



Train	Diagnostic Code	From	To	Environment
185001	3992	1355568104	1355568178	A4CD120050000300000000B2000000005B150513
185001	3984	1355568111	1355568176	A4CD12005000841C180100B2000000005B150513
185001	3985	1355568111	1355568176	A4CD12005000841C180100B2000000005B150513
185001	3993	1355568111	1355568176	A4CD12005000841C180100B2000000005B150513
185001	3995	1355569094	1355569131	A5CD12005000850218030 0CE02080D0D51150E51
185001	4014	1355569138	1355569138	A5CD1200500084021A0300CE0000000043151453

Diagnostic code for particular train with duration information
System and environmental variables attached to this specific diagnostic code

Figure 4.1: Sample diagnostic code with attached values

Each instance of diagnostic data has also a set of system variables attached to it which are encoded as a string. Based on the type of diagnostic code only specific variables which are relevant to the code are encoded. These hold a wide range of values like temperature or air pressure, simple boolean indicators, system states of specific components etc.

4.2 Data Preparation

We select only those diagnostic data instances which are related to the power converter unit. This is done by sorting the data by specific diagnostic message code which are specific to power converter.

4.2.1 Tours

Diagnostic data of each train is separated and organized in a form of a train tour. With a tour starting with main engine powering on and ending with engine switching off. At least 100 kms must be covered at a stretch and a minimum time duration of a tour is 2 hours.

4.2.2 Incident Tours and Non Incident Tours

Tour in which power converter failure takes place is termed as incident tour. Tours in which there are no imminent failures are termed as non incident tour.

4.3 Converting Diagnostic Data to Instances

Diagnostic data for a specific time frame can be easily extracted using the from and to timestamps. Further a total of 888 diagnostic codes which are related to power converter unit are used to extract tours. In consultation with domain experts 11 relevant system variables were also selected. Each diagnostic message is then represented by a boolean flag which indicates whether that specific code was active or inactive for any specific time instance. Status vectors for each tour are generated in for each relevant point in time, where relevant is described as a point where the status vector changes in contrast to the one before. Using fixed intervals might result in multiple identical instances as status might not change rapidly hence we use arbitrary intervals instead. Preparing data this way, multiple instances are created per tour and each instance has fixed number of features. In all we created 900 features per instance. Table below shows an example of conversion of status vectors to binary instances.

	Code							
	C_1	C_2	C_3	**....**	C_{n-1}	C_n	$Temp_1$	$Temp_2$
Instance 1	0	1	0	....	0	0	?	?
Instance 2	1	0	0	....	0	1	12.5	14
Instance 3	0	1	1	....	0	0	?	?
Instance 4	0	1	0	....	1	0	?	?

Table 4.1: Example of status vectors as binary instances

5 Basics of Machine Learning

Machine learning is a subset of modern computing that focuses primarily on devising algorithms that are fundamental for predicting useful patterns from huge pool of existing data as well as large volume of data that is getting generated every second[9].

5.1 Basic Terminologies

We have listed here few of the key definitions and terms required for understanding this thesis report. For better understanding and more details you can refer to[10].

5.1.1 Classification

Inputs(instances) are organized into multiple classes based on some grouping strategy mostly based on feature attributes. The task is then to learn this model of classification so that it can be later on applied to unknown inputs which can be then classified into one of those learned classes.For example by analyzing blood samples of people suffering from a certain disease we can build a model which can be later on used to classify other test samples.

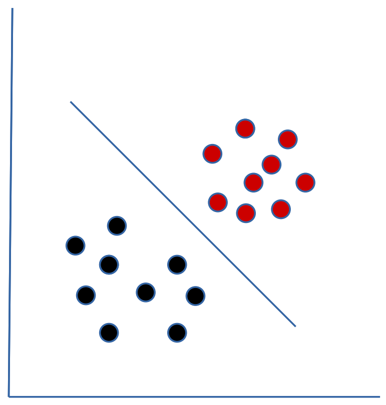


Figure 5.1: Classification

In the figure above we can see red and black dots which represents two distinct classes. An ideal classifier will be able to separate each class instance accurately. Classification is used in areas like text categorization for instance in case of filtering of spam emails, detection of fraud, optical character recognition, segmentation of markets with respect to product and pricing, bio-informatics etc.

5.1.2 Clustering

Clustering is a widely used style of machine learning which helps the process of forming a group of abstract objects within classes of synonymous objects. In other words, when there is no specified class, clustering is used to group items that seem to naturally belong together. It is a very common step in the domain of data analysis. It basically assigns an observation sets into various subsets (which are called clusters) so that the observations are synonymous in some sense. For example the task of grouping similar animal pictures. In the case of classification, information about actual animals that were depicted are known through labels. In clustering, which animals are depicted is completely unknown. By analyzing the set of pictures the clustering algorithm will group similar photos together.

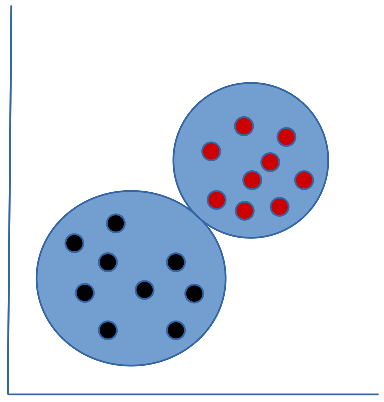


Figure 5.2: Clustering

In the figure above, we can see two clusters which are formed for two classes denoted by red and black dots. One of the most common and widely used methods of clustering is k-means which is an algorithm that is used in classifying objects depending on attributes into k-amount of groups. Clustering is used in areas like data mining, pattern recognition , image analysis and image processing, weather report analysis done by meteorological department etc.

5.1.3 Association

Association learning helps in finding out various interesting connections between elements of data and the sequence of behaviors that lead to some correlated results. It is usually used to analyze and observe trends in transactional data. In most cases, the order of items within a transaction or across transactions is mostly ignored while learning.

Retail industry is one of the biggest examples where the association rule is utilized. Nowadays almost all purchases done by a customer is easily documented. Big retail chains use these records by executing a so called "market basket analysis" of the customers purchases to identify which products go along, and also obtain a brief idea of the customers shopping patterns, interests and current trends.

5.1.4 Regression

Regression analysis is a statistical tool for the exploring possible relationships between variables. More specifically, investigators try to understand how the typical value of the dependent variable changes with respect to changes in any one or multiple independent variables. To explore such issues, the investigator assembles data on the underlying variables of interest and employs regression to estimate the quantitative effect of the causal variables upon the variable that they influence. The investigator also typically assesses the "statistical significance" [11] of the estimated relationships, that is, the degree of confidence that the true relationship is close to the estimated relationship.

Some classic examples of regression problem analysis are related to the field of economic statistics, such as increase in price upon increase in demand or shortage of supply, changes in the money supply upon fluctuations in rate of inflation.

5.2 Types of learning

Machine learning is predominantly used today as a significant tool for recognizing patterns of data which can help in developing automated models. In essence, the main types of ML are; supervised, semi-supervised and unsupervised learning

5.2.1 Supervised Learning

In supervised type of ML, algorithms are basically trained with the utilization of completely labeled data. This labeling of training data is done by a ’teacher’. The desired output is already known. The goal is then to learn the mapping between the input and the corresponding output, so that this knowledge can be applied for other unknown cases(test cases).

Various processes like regression, classification, prediction and gradient are utilized for the purpose of predicting the label values on additional unlabelled data. Majorly in cases where the historical data are good predictors of future outcomes, supervised machine learning is mainly utilized. One of the most common application of supervised machine learning is found in banks and financial institutions.

One such example is fraudulent credit cards, banks and the financial institutions execute data analysis on their customer database for predicting whether the credit card transactions will be fraudulent or not[12]. They do this by learning the customers usual behavior, if certain unusual activities which are not consistent with this customers profile are performed it will automatically raise alarms.

5.2.2 Semi-supervised Learning

The semi-supervised type of machine learning is another dimension of machine learning wherein proposal for solution is given when there is an amalgamation of large volume of unlabelled data with small amount of labeled data. The unlabeled data cost less than labeled data and less effort is engaged for acquiring them. The process of semi-supervised machine learning is effective when the cost associated with the labeling is comparatively higher to allow for fully labeled training process[13].

The motivation behind semi-supervised machine learning is that of making the process of data analysis faster, better as well as cheaper. In varied real world applications, the process of acquiring huge volume of unlabeled data is relatively easy. As for instance, documents can be easily downloaded from the internet; various images can be extracted from the surveillance cameras and speech can be extracted from various broadcasts. However, standing at this position, the corresponding labels for the prediction task namely sentiment orientation, intrusion detection; phonetic transcript often incorporates sluggish human annotation and circumscribes highly expensive laboratory experiments. Thus, this constraint in the labeling bottleneck gives rise to scarce labeled data and thereby produces surplus unlabeled data. In the recent times, semi-supervised learning has found vehement application in fields like cognitive psychology as a computational model. Research evidences reveal that human beings are quite capable and good at combining labeled and unlabeled data for facilitating the learning process[14]. By using semi supervised learning, many data scientists have observed considerable improvement in learning accuracy.

5.2.3 Unsupervised Learning

Training data is not labeled, leaving the machine on its own to discover patterns in the input and deduce the output. In case of unsupervised ML, the probability of getting success in the new predictive models depends on explanatory factors like patterns, structures and relationship of the incoming data sets. One of the most common and practiced algorithm of unsupervised learning is cluster analysis that is targeted towards finding out the hidden patterns within the data sets. The cluster analysis includes examples like socioeconomic tiers, psychographic data, social network graphs, purchasing patterns and so on. The other approach of unsupervised ML is using a reward system wherein positive and negative rewards are given for providing valuable feedback to the predictive models when it is successful[15].

In a traditional sense, search engine giants like Google has huge volumes of labeled data as for instance, whenever someone asks a query in the voice search and then tap on that result, it implicitly signifies that the answers which are returned from the query is right. Thus, the task of labeling is achieved. Now, the crux of supervised ML types is that it takes into account the necessary algorithms that are trained with the help of using labeled data and the system is not given the right answer. The mechanism is such aligned that the algorithm can figure out what is being shown in it. The prime objective is to crunch the raw data given and thereby figure out the structure within it.

Majorly the mechanics of unsupervised learning works very well when it is treated with transactional data. As for instance, unsupervised can help in the process of identifying customer segments with alike characteristics who can be subsequently treated synonymously in varied promotional and marketing campaigns. The unsupervised technique can also find out major attributes that can segregate different customers from each other. In unsupervised learning, widely used techniques are self-organization maps, k-means clustering, nearest neighbor mapping and singular value decomposition. The underlying mechanism of these algorithms help in segmenting text topics, recommending items and thereby helps in identifying ways to treat the outliers which are grey areas within the data.

5.3 Evaluation

Evaluating how good the learning algorithm performed is the key to making real progress in data mining and it can be done using the following techniques.

5.3.1 Training and Testing

This is the most basic approach in which the classifier is modeled using a specific training set, then its performance is evaluated on an independent test set. It is important to note that test data is not used in any manner to model the classifier. For better results, both the training and test data must be good representative samples of the underlying problem. In other words, each class in the complete data set should be represented in the right proportion in both the training and testing sets. However in certain cases test data instances might be distinct in nature from the training data, for example in case of fraudulent credit cards, training can be done exclusively on normal customer operations ignoring all fraudulent test instances, while testing should done on combination of both.

- Holdout
  The easiest approach for creating the test and training sets is the holdout method. This method reserves a certain amount of data for training and uses the remainder for testing. Usually in most cases the ratios are two-third for training and one-third for testing.

- Stratification
  By using the holdout method there is no guarantee that both the sets individually are good representatives of all classes present in the complete data set. If certain class is skipped in the training set, then classifier learned using such data would hardly be able to classify any instances of that class in the test set. Instead, to avoid such circumstances, random sampling must be done in such a way that it each class is properly represented in both training and test sets. This procedure is called stratification.

5.3.2 Cross Validation

Training and testing can be quite an easy choice and with stratification results will definitely improve, however it might not be sufficient especially when learning large amount of data with many features. A very efficient alternative is cross validation in which the user decides on fixed number of folds according to his use case. For example if he selects five folds, Then the data is split into five equal partitions. Each partition is then used in turn for testing and the remaining all are combined and used for training. That is, use one-fifth of the data for testing and four-fifth for training and repeat the procedure five times so that in the end, every instance has been used four times for training and exactly once for testing. This is called fivefold cross-validation, and if stratification is adopted then it is called stratified fivefold cross-validation.

5.4 Metrics

The results of evaluation can be measured using the metrics explained below. Users must specifically choose and focus on some metrics that become crucial for solving their problem case the most. In the two-class case with classes yes and no, a single prediction has the four different possible outcomes as shown in table below.

		Predicted Class	
		Yes	**No**
Actual Class	**Yes**	True Positive	False Negative
	No	False Positive	True Negative

Table 5.1: Outcomes of two-class prediction

The true positives (TP) and true negatives (TN) are correct classifications. While a false positive (FP) is when the outcome is incorrectly predicted positive(as yes) when it is actually negative(no). A false negative(FN) is when the outcome is incorrectly predicted as negative(as no) when it is actually positive(yes).

5.4.1 True positive rate

True positive rate is also known as "Sensitivity" or "Recall". It measures the proportion of positives that are correctly identified as such. In our example, true positive rate indicates how often does the classifier predict yes, when actual value is yes.

$$True\ positive\ rate\ (TPR) = \frac{Number\ of\ true\ positives\ (TP)}{Total\ number\ of\ positives}$$

$$True\ positive\ rate\ (TPR) = \frac{Number\ of\ true\ positives\ (TP)}{Number\ of\ true\ positives\ (TP) + Number\ of\ false\ negatives\ (FN)}$$

5.4.2 True negative rate

True negative rate is also known as "Specificity". It measures the proportion of negatives that are correctly identified as such. In our example, true negative rate indicates how often does the classifier predict no, when actual value is no.

$$True\ negative\ rate\ (TNR) = \frac{Number\ of\ true\ negatives\ (TN)}{Total\ number\ of\ negatives}$$

$$True\ negative\ rate\ (TNR) = \frac{Number\ of\ true\ negatives\ (TN)}{Number\ of\ true\ negatives\ (TN) + Number\ of\ false\ positives\ (FP)}$$

5.4.3 Positive predicative value

If the classifier predicts positive(yes), how often it is correct. Positive predicative value is also called as "Precision".

$$Positive\ predicative\ value = \frac{Number\ of\ true\ postives\ (TP)}{Number\ of\ true\ postives\ (TP) + Number\ of\ false\ postives\ (FP)}$$

5.4.4 Negative predicative value

If the classifier predicts negative(no) , how often it is correct.

$$Negative\ predicative\ value = \frac{Number\ of\ true\ negatives\ (TN)}{Number\ of\ true\ negatives\ (TN) + Number\ of\ false\ negatives\ (FN)}$$

5.4.5 F-measure

F-measure is the harmonic mean between recall and precision.

$$F - measure = 2 \cdot \frac{Precision\ .\ recall}{Precision + recall}$$

5.4.6 Accuracy

Accuracy is also referred as "Overall success rate". It measures how often the classifier is correct.

$$Accuracy = \frac{Number\ of\ true\ postives\ (TP) + Number\ of\ true\ negatives\ (TN)}{Total\ (N)}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

5.4.7 Misclassification rate

Misclassification rate is also referred as "Overall failure rate". It measures how often the classifier is wrong.

$$Misclassification\ Rate = \frac{Number\ of\ false\ postives\ (FP) + Number\ of\ false\ negatives\ (FN)}{Total\ (N)}$$

$$Misclassification\ Rate = \frac{FP + FN}{TP + TN + FP + FN}$$

5.4.8 Matthews correlation coefficient

Matthews correlation coefficient(MCC)[16] was introduced in 1975 by Brian W. Matthews. It is used to a evaluate the performance of binary classifications. MCC measures the correlation between all the predicted values with the actual values. It returns a value between -1 and +1. Where a correlation of C = 1 indicates perfect agreement, C = 0 is expected for a prediction no better than random, and C = -1 indicates total disagreement between prediction and observation.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)\ (TP + FN)\ (TN + FP)\ (TN + FN)}}$$

5.5 WEKA

Weka stands for "Waikato Environment for Knowledge Analysis"[17]. It is developed by University of Waikato, New Zealand and is an open source suite of machine learning software written in Java issued under the GNU General Public License. WEKA basically comprises of a wide collection of machine learning algorithms along with useful tools for the purpose of data pre-processing, segmentation, clustering, regression, association as well as for devising rules and visualization. The WEKA platform is also instrumental for the purpose of devising new machine learning schemas. The WEKA machine learning algorithms can be fully implemented within the Java programming and it has the capability of running on almost every modern computing platform like Windows, Mac OS X and Linux.

Within the broad trajectory of Java programming language, WEKA can be contemplated as a collection of algorithms for machine learning that are instrumental for accomplishing data mining tasks primarily. Data mining is one of the most essential practices for the purpose of manipulating and analysing huge datasets for the purpose of identifying and discovering the hidden patterns and thereby helps in inferring business intelligence. Business intelligence is one of the most essential aspects of tapping useful information and then deciphering valuable business acumen from it.

The functioning of WEKA platform runs in a very sequential and logical manner. In data preprocessing, selection and transformation are done after which data is prepared. Subsequently data is trained after which it is subjected to mining. Then patterns are identified and developed into varied models which can be later on subjected to evaluation and verification. There are mainly three graphical user interfaces namely "The Explorer", "The Experimenter" and the "Knowledge Flow". "The Explorer" is basically fragmented into pre-process data, building classifiers, clustering data, finding associations, selecting attributes and thereby helps in the process of data visualization. "The Experimenter" is utilized in the process of comparing the performance of various learning schemas. The "Knowledge Flow" is basically a Java-Beans-based interface that is plugged for setting up as well as running machine learning experiments.

The Explorer interface is composed of a wide array of panels that give access to all major components of the workbench.

- The panel of pre-process has the facilities of data import from a database or file for the purpose of pre-processing and utilization of filtering algorithm. The filters are used in the process of transforming the data i.e. transforming the numeric attributes into discrete ones and then make it possible for deleting the instances as well as the attributes as per the specific criteria provided.

- The Classify panel helps in enabling the user towards applying classification as well as the running regression algorithms.

- The associate panel helps in providing access to the rules of association and helps in identifying relationships between several attributes within the datasets.

- The cluster panel helps in providing access to the techniques of clustering within the WEKA platform. There is a simultaneous implementation of the expectation maximization logic for the purpose of learning an amalgamation of normal distributions.

- The Select attributes panel helps in providing algorithms that helps in the process of identifying the majority of the predictive attributes within the dataset.

- The Visualize panel basically helps in showing the scatter plot matrix wherein individual scatter plots can be selected and then enlarged as well as analysing for the purpose of varied selection operators [18].

6 Support Vector Machines

Support vector machines (SVMs)[19] was first introduced in COLT-92 [20] are a set of supervised learning methods used for classification, regression analysis and outliers detection. SVM algorithm is called as non-probabilistic binary linear classifier by building a model that assigns the new objects or examples to one or the other category. It is a supervised learning classifier that analyses data using the regression analysis for decision boundary defined by a separating hyper plane. SVM constructs this hyper plane which separates the set of different class members. The best hyperplane for an SVM means the one with the largest margin between the two classes. That is why it is also called as widest street approach. It is very diverse and used in many fields like machine learning, statistics, neural networks, optimization, statistics.It has nice properties like convex and nonlinear.

6.1 Linear SVM

If data is linearly separable, then we can use the Linear SVM which directly classifies data by finding the best hyperplane that separates all data points of one class from those of the other class. Margin means the maximal width of the slab parallel to the hyperplane that has no interior data points.

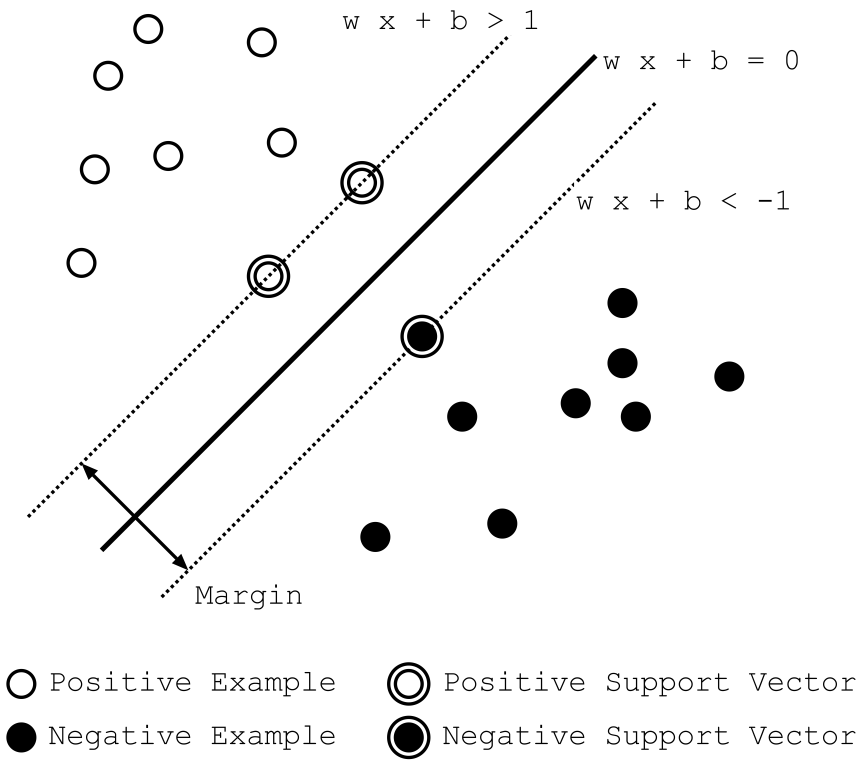


Figure 6.1: Linear SVM
[21]

Let us consider we have a training data set of n points of the form

$$(\vec{x_1}, y_1), \ldots, (\vec{x_n}, y_n)$$

In this y_i is the class which the point x_i belongs to. y_i can be 1 or -1 . Each x_i is a P-dimensional real vector. Then we want to find the maximum margin hyper-plane which divides the group of points into two classes for which y_i =1 or y_i =-1, so that the distance between the nearest point x_i and the hyper-plane from one of the group is maximized.

The hyperplane can be written as ,

$$\vec{w}.\vec{x} - b = 0$$

With all x_i points. w_i= normal vector to the plane.
The parameter $\frac{b}{||\vec{w}||}$ determines the offset of the hyperplane from the origin along the normal vector $||w||$.

For separating two classes of data, we can select two parallel hyperplanes so that the distance between them is as large as possible. This we can do, if the training data is linearly separable. The region bounded between both hyperplanes is called as Margin. The maximum margin hyperplane lies in the center of them. The parallel hyperplanes can be written as,

$$\vec{w}.\vec{x} - b = 0 \text{ and } \vec{w}.\vec{x} - b = -1$$

The distance between the two hyperplanes is $\frac{2}{||\vec{w}||}$. So the distance is inversely proportional to w. For maximizing the distance, we need to minimize the $||\vec{w}||$.

We need to prevent the data points from falling into margin. For that we need to add the following constraint,

$$\vec{w}.\vec{x_i} - b >= 1, \text{if } y_i = 1 \text{ or } \vec{w}.\vec{x_i} - b <= -1, \text{if } y_i = \text{-}1$$

The above constraints state that each point must stay on the correct side of the margin. The above formula can be rewritten as,

$$y_i\ (\vec{w}.\vec{x_i} - b) >= 1, \text{for all } 1 <= i <= n$$

We can solve optimization problem from all the above formulas put together,

$$Minimize||w|| \text{ subject to } y_i(\vec{w}.\vec{x_i} - b) >= 1, \text{for } i = 1, \ldots, n$$

The w and b solve the optimization problem to determine the classifier,

$$\vec{x} \mapsto sgn(\vec{w}.\vec{x_i} - b)$$

From above geometrical calculation, we can say that maximum margin hyperplane is determined by the x_i which are very near to the hyperplane. Those x_i are called as support vectors.

6.2 Non-Linear SVM

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. If data is not linearly separable in original feature x_1 an additional deterministic features could be added. For example going for two features x_1 and x_2 instead of x_1 where $x_2 = x_1^2$ and the data lie on this curve. Then in this new higher dimensional space, the data is more likely to be linearly separable.

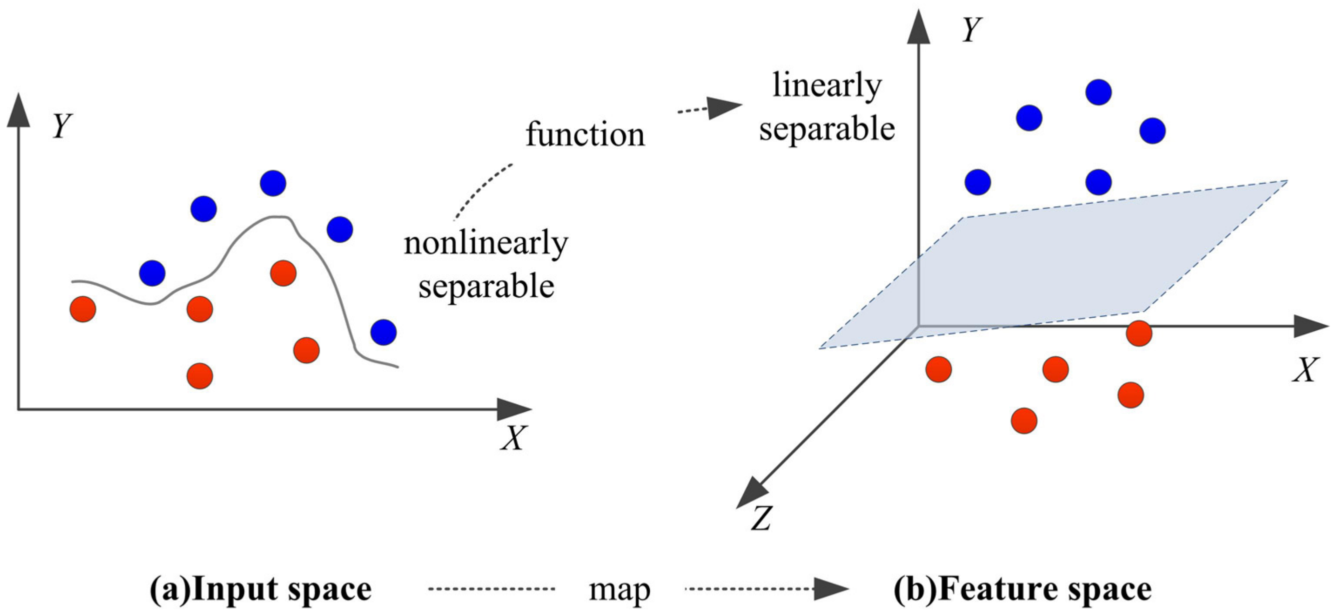


Figure 6.2: 2D input space mapping into 3D feature space to separate data linearly. [22]

Using a feature transform $\phi(x)$ we transform x_i and x_j to find their new feature vectors. Then the dual form involves the dot product between these transform vectors.

$$\hat{y}(x) = sign[w.\phi(x) + b]$$

For example, consider the polynomial features given here are x_1, x_2 , x_1^2 , x_2^2, cross products.

$$\phi(x) = (1\sqrt{2}x_1\sqrt{2}x_2 \ldots x_1^2x_2^2 \ldots \sqrt{2}x_1x_2\sqrt{2}x_1x_3 \ldots)$$

For the dual form, the inner product must be computed to expand feature vectors $\phi(x_i)$ and $\phi(x_j)$. Let us consider x_i and x_j as a and b respectively, then features for both points can be computed as.

$$\phi(a) = (1\sqrt{2}a_1\sqrt{2}a_2 \ldots a_1^2a_2^2 \ldots \sqrt{2}a_1a_2\sqrt{2}a_1a_3 \ldots)$$

$$\phi(b) = (1\sqrt{2}b_1\sqrt{2}b_2 \ldots b_1^2b_2^2 \ldots \sqrt{2}b_1b_2\sqrt{2}b_1b_3 \ldots)$$

The dot product of both points as stated below can be manipulated into much simpler computation. This non-linear functional similarity is called as kernel and it can be computed in time linear in the original number of features a and b regardless of how many features our expansion ϕ has.

$$\phi(a)^T\phi(b) = 1 + \sum_j 2a_j b_j + \sum_j a_j^2 b_j^2 + \sum_j \sum_{k>j} 2a_j a_k b_j b_k + \dots$$
$$= (1 + \sum_j a_j b_j)^2$$

7 One-Class Support Vector Machines

In one class learning, only samples from the positive class i.e. datasets of only non-failure cases is considered for training. Once the support vector machine is completely trained, then this knowledge can be modelled and applied to classify the test datasets. If the test data instance is of a failure case i.e measurements are very different, the model will classify this particular instance as out-of-class.

This same technique can be used to predict a train component failure. We can train the SVM by providing it data from the normal working conditions of the train, i.e. when there is no component failure. Therefore first and the most crucial task is the selection of the most appropriate instances for training SVM. Further when the SVM is trained, the goal is to evaluate whether any new data instance is alike or different from the training data. If the instance is classified different, it has to be further established that it is indeed from a component failure case and not just a false negative. Proper instance selection for training of SVM is the key here as it directly influences the output of classification hence determining the overall accuracy.

7.1 Working of One Class SVM

The best suited SVM parameters must be selected for performance variation and the representation of data. The choice of the kernel depends on the number of features which are used for classification. The performance difference is clearly visible by changing the kernels and other parameters. Usually SVM is two class algorithm with positive and negative examples. Schölkopf [23] proposed one-class SVM method for solely positive data.

Suppose we have a dataset $x_1 \ldots x_l$ which belongs to X which is drawn from an unknown probability distribution ρ. Now we need to estimate a set S which is completely new and unseen before, $x_l + 1$ lie in S with a specified probability. S is bounded by a prior specification

$\nu = \epsilon$ (0,1)

The solution for this problem is by estimating a function f which is negative on $\overline{S}$ and positive on S. In other words, the function f takes the value $+1$ in a small region which captures most of the data vectors.

$f(x) = +$ 1 if $x \epsilon S$

$f(x) =$ - 1 if $x \epsilon \overline{S}$

This can be done by using an appropriate kernel function. We can map the data into the feature space H, then try to separate the mapped vector from the origin which has maximum margin.

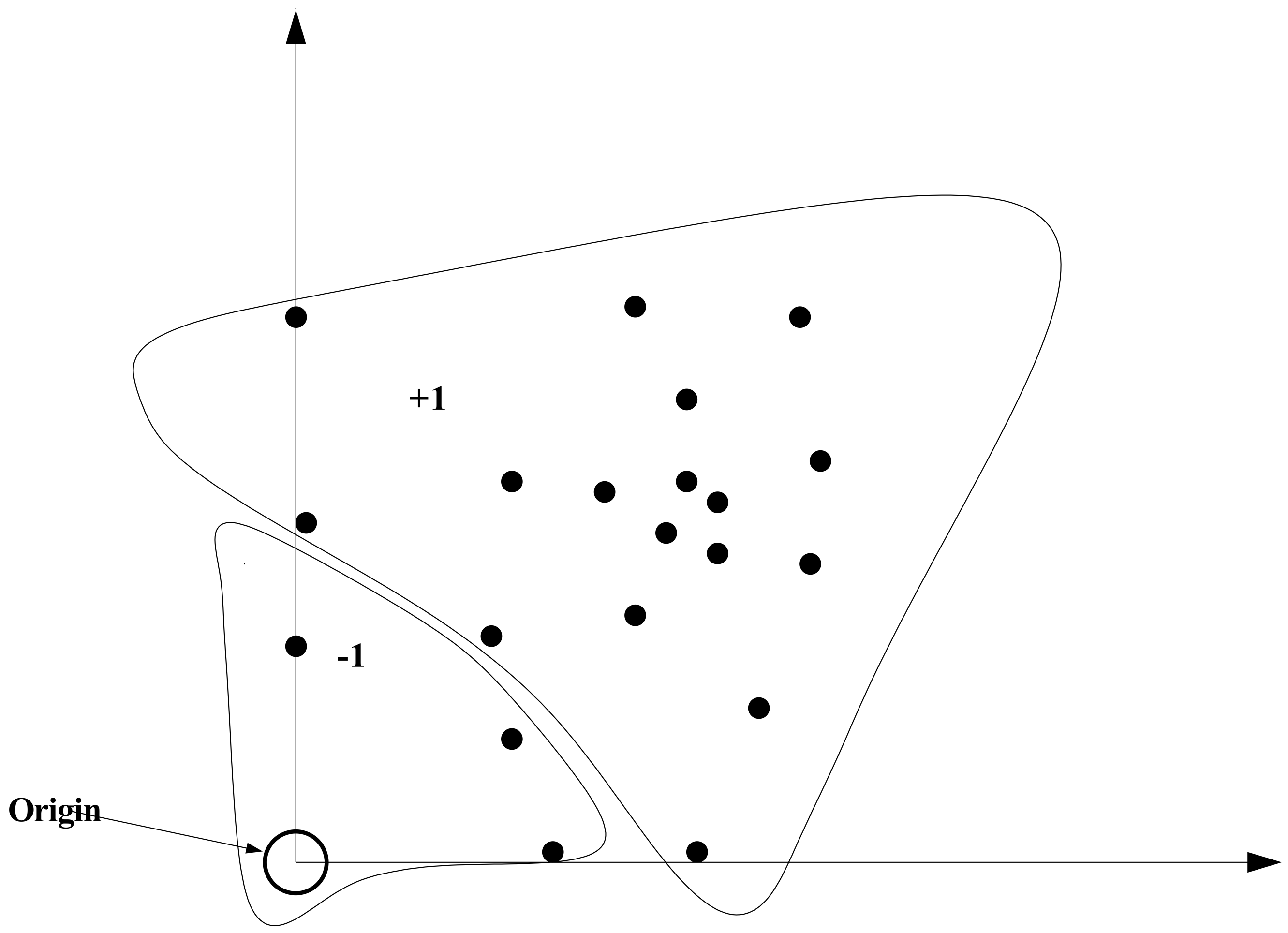


Figure 7.1: One Class SVM Classifier. The orgin is the only orginal member of the second class [24]

For example, let $x_1, x_2 \ldots x_l$ be the training examples which belongs to one class X, where $X \epsilon R^N$.

Let Φ be the kernel map which is used to transform the training examples to another feature space.

$\Phi : X \rightarrow H$

By solving the below quadratic formulas, we can separate the dataset from the origin.

$\min \frac{1}{2}||w||^2 + \frac{1}{\nu l} + \sum_{i=1}^{l} \xi_i - \rho$

Subject to,

$(w.\Phi(x_i)) >= \rho$ - ξ_i $i = 1{,}2, \ldots l$ $\xi_i >= 0$

If w and ρ solve this problem, then the decision function

$f(x) = \text{sign}((w.\Phi(x)) - \rho)$

The decision function $f(x)$ will be positive for most of the examples x_i in the training set.

7.2 One Class SVM Kernels

Using the kernel trick we can convert a non-linear classifier into a linear classifier. The kernel function in SVM is a similarity function. We can compute similarity between two inputs by defining a single kernel function. Normally we take the input data and find feature vectors from them and we feed labels and feature vectors to the learning classifier. For implementing a kernel function we have one mathematical solution that is every kernel function can be expressed as a dot product in feature space. By using kernel function, the features can be carried to high-dimension space. Kernel functions also make the calculation process easier and faster.

In One Class SVM, we use the following kernel types,

7.2.1 Linear Kernel

Linear kernel is the simplest form of kernel function. These kernels are used to select optimal hyper planes. By using a linear kernel function we get equivalent results to their non-linear functions. It is mathematical expressed as ,

$$K(x_i, x_j) = x_i^T x_j$$

7.2.2 Polynomial Kernel

These are called as non-stationary kernels. With a polynomial kernel function we can separate data linearly in higher dimensional space. Polynomial kernel is well suited when all the training data is normalized. This function allows us to model features at the same point in time or space up to the polynomial order. It is mathematical expressed as ,

$$K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$$

7.2.3 Radial Basis Function

Radial Basis Function (RBF) is used as default kernel in one-class SVM. It is one of the most used kernel, it fits a gaussian distribution over near by points of test point. The Gaussian kernel is an example of RBF function. The closeness is defined by standard distance metric. RBF function approximates the separating boundary by using piece wise gaussians. RBF allows us to pick out hyperspheres. It is mathematically expressed as ,

$$K(x_i, x_j) = exp(-\gamma ||x_i - x_j||^2), \gamma > 0$$

γ plays crucial part in performance of this function. This value should be used very carefully because if it is underestimated, RBF function lacks regulation and the boundary will be highly sensitive to data noise. On the other hand, if overestimated, high dimensional projection will loose non-linearity as exponent starts behaving linearly.

7.2.4 Sigmoid

Sigmoid kernel function is also called as Hyperbolic tangent kernel function. This function comes from Neural networks as we use bipolar sigmoid as activation function. When using this function in SVM, it is equivalent to a to a two-layer, perceptron neural network [25]. It is mathematically expressed as ,

$$K(x_i, x_j) = \tanh(\gamma x_i^t x_j + r)$$

7.3 Kernel Parameters

As seen in the mathematical descriptions above, all the kernels have some parameters which must be fine tuned to get the best results, so lets take a look into all these kernel parameters.

7.3.1 NuSVC

This parameter controls the training errors and number of support vectors. It is a re-parametrization of the c parameter. Parameter c is the penalty factor for non-separable points. With c-parameter we have no boundary. So if c is set too large, we have a high penalty for nonseparable points and we may store many support vectors and overfit. If it is too small, we may have underfitting[26]. When $C = \infty$ it gives a hard margin SVM. The value of parameter ν is in range (0,1], in this the lower bound is for the support vectors relative to the total number of examples and an upper bound on training errors. This specifies minimum number of support vectors and maximum ratio of training errors relative to total number of training samples.

The problem with parameter c is that, there is no direct interpretation and it can take any positive value. Parameter ν has a direct interpretation and we have bounds between 0 and 1. Parameter c and ν are mathematically equivalent but regulation with ν is easier when compared to parameter c. That is the reason we use ν in one-class SVM. The default value of ν is 0.5. In the range of 0 and 1 the larger the value, the decision boundary will be smoother. NuSVC(ν) is the parameter N that we define in the experiments.

7.3.2 Degree

This parameter is only applicable to polynomial kernel function. It is the degree of the polynomial. The default value is 3. Degree is the parameter D we define in the experiments.

7.3.3 Gamma

Parameter γ defines how far the influence of a single example reaches. Low values indicate that every point has a far range, while high values indicate that it is a close range example. The gamma parameters are the inverse of radius of the influence of the samples.

$$\gamma = \frac{1}{The\ number\ of\ features}$$

The value of gamma influences the behavior of the model. If gamma is too large, then the influence radius of the support vector includes only support vectors itself. If gamma is too small then the model cannot capture the complexity of the data and it is too constrained so it can include the whole training set. Then the resulting model would be a linear, with set of hyper planes which has higher density of any two classes. We need to change γ in conjunction with the selected value for ν. Gamma(γ) is the parameter G that we define in the experiments.

7.3.4 Coef0

This function is applicable only for polynomial and sigmoid kernels. It is an independent term. Coef parameters are the weights which represent the hyperplane in linear SVM, by giving the coordinates of the vector which is orthogonal to that hyperplane. With this vector we can determine the direction. The dot product of any point with this vector can determine whether the point belongs to positive class or negative class.

So we could say that the absolute size of coef is relative to the other features gives us an indication of this particular feature's importance for the separation. In the case of large number of variables that separate the data, one can prefer correlation coefficients instead of ranking criteria to distinguish between top ranking variables. If the number of variables are larger than number of examples, we need to use coef to restore the selected variables. Coef0 is the parameter R that we define in the experiments and the default value is 0.

7.3.5 Tolerance of termination criterion

This is a converge tolerance parameter. It specifies the tolerance of termination criteria. That means to compute support vector, we need to specify the maximum gradient of the quadratic function. Training will be terminated when the gradient is less than or equal to tolerance value. The default value is 0.001. It sets around 0.001 to 0.1. Tolerance (Epsilon) is the parameter E that we define in the experiments.

7.3.6 Shrinking heuristics

This parameter is used to determine whether we should use the shrinking heuristics or not. Shrinking heuristics are used to speed up the process of optimization or train the models faster. Shrinking reduces the problem size by temporarily eliminating variables. First it removes some bounded components in the iterations. It also reduces number of kernels to update gradient factor. The implementation with and without shrinking gives similar iteration reduction but without shrinking gives better time improvement than with shrinking. With shrinking, the iteration ratio is larger. The default value of shrinking heuristics is on. Shrinking heuristics is the parameter H that we define in the experiments.

7.3.7 Cache size

It specifies the size of cache in Megabytes. Caching is also used to speed up the process of decomposition. After shrinking, it allocates some memory to store recently used positive samples and it can significantly reduce the number of kernel evolutions. When cache is large enough to store all the kernel elements, the time reduction does not match that of iteration due to higher cost on selecting working set per iteration and hence ratio is smaller. The default cache size is 40MB. Cache size is the parameter M that we define in the experiments.

7.3.8 Verbose

Verbose parameter output is used for methods that can converge slowly. The default value is false.With this parameter enabled, it may not work good in multithreaded environment that is why we did not use this parameter.

7.3.9 Random state

The seed of random number generator to use for data shuffling probability distribution. The default value would be none.

8 Rest Web Service

Internet with web services as its underlying core has been contemplated as the most pivotal mechanism for information and resource dissemination. Web services were first introduced in 2000. The figure below depicts the evolution of various web based services

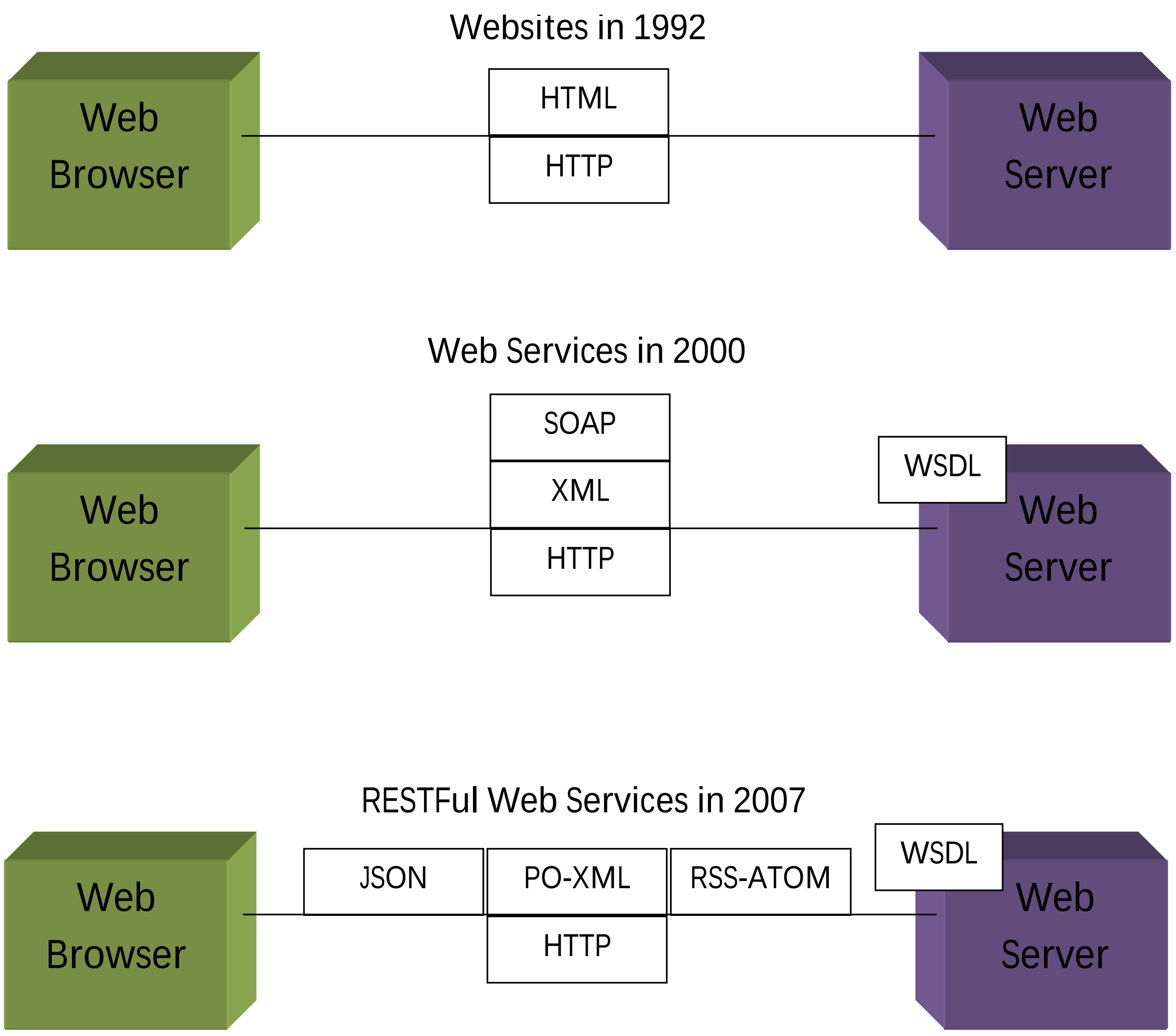


Figure 8.1: Evolution of Web based Services.

The term REST was first coined by Roy Thomas Fielding, an American computing scientist and one of the chief authors of REST web services and HTTP specifications. REST stands for Representational State Transfer. Fielding was of the opinion that REST will be majorly targeted towards inducing a simplified overview that can analyze the behavior of a web application in the maximum possible demeanor. When a user progresses through an application through selection of links (state transitions) and move in to the next page that represents the next application state, REST web services intervenes and renders information and resources for utilization by the user [27].

REST is an architectural style for distributed web applications. An application which adheres to REST architectural style is called "RESTful". In contrast to SOAP, REST is not a protocol. REST is also not a standard in the sense of the W3C. REST could be implemented using many different technologies, but HTTP is the most popular approach. REST uses a wide array of HTTP operations like GET, PUT, POST and DELETE for changing the state of remote resources. The REST web services utilizes a pull based client-server interaction style. The services are stateless which increases transparency of information fetching. In the process of REST mechanism, it is ensured that each request from client to server is understood properly. Apart from ensuring that the request is understood in an appropriate manner it also guarantees that the request is unique, so that no advantage of any stored context on the server is taken into the forefront while processing the request.

REST uses cache for improving the efficiency of the responses of the network. It thereby properly segregates the information into cacheable and non-cacheable categories. Web Services expose their data and functionality through resources identified by Uniform Resource Identifier(URI). Each of these resources can then be then accessed through a unique Uniform Resource Locator(URL). The resource representations are interconnected through URLs and help the user to progress from one state to another state in a sequential manner without any blockage. For strengthening security trajectory and subsequently increase performance, the REST web services employ layered components or intermediaries. There is a provision of inserting intermediaries like proxy servers, cache servers and gateways that takes good care of the performance and security domains.

The resource representations of REST are XML, HTML, GIF, JPEG etc. The MIME type representations are text/xml, text/html, image/gif, image/jpeg etc. REST web services engage very little infrastructure support and processing technologies and are well supported by majority of the programming languages today. The REST web services are quite simple and effective as they use HTTP which is the most widely utilized application protocol. It has been found that in many of the cases, the simplicity of HTTP supersede the complexity of implementing an additional transport layer protocol [28].

8.1 HTTP Methods

HTTP is an underlying protocol of REST web services. The information, messages and the requests are transmitted in the HTTP format. Each resource in REST is accessed by a common interface using standard HTTP methods. The most popular HTTP methods[29] which are commonly used in REST based architecture are as follows,

- GET
  This method provides a read only access to a resource which is identified through the requested URL.

- PUT
  This particular HTTP method actually helps in requesting the server for storing the enclosed entity under the requested URL. According to the HTTP specification, the server has the capacity to create the resource if there is no existence of the entity

- POST
  This method is used to submit data which is to be processed by the resource

- DELETE
  As its name suggests, this method is basically directed towards removal of resources.

- OPTIONS
  This method is used to get the the list of operations supported by the resource specified in URL.

Method	**Safe**	**Idempotent**
POST	Yes	No
GET	Yes	Yes
PUT	No	Yes
DELETE	No	Yes
OPTIONS	Yes	Yes

Table 8.1: HTTP Methods Categories.

Further HTTP operations can be classified into safe and idempotent as shown in the table above,

- Safe
  Safe methods do not modify or change the state of the requested resources. So only data retrieval is possible.

- Idempotent
  An idempotent method even if called multiple times doesn't change the outcomes. In other words, it would not matter if the method is called only once or hundred times, the result will remain the same.

8.2 REST with JAVA using JERSEY Framework

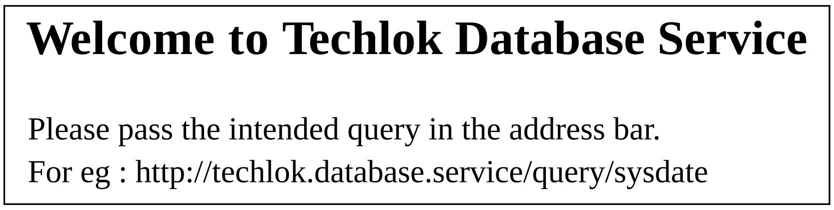
For the purpose of varied media type representation and abstracting the low-level details of the client-server communication within the REST web services, there is a necessity of a reliable and robust toolkit. Java Jersey framework [30] acts as that robust toolkit that makes various tasks of REST web service development simpler. Java Jersey framework can be regarded as a set of Application Programming Interface(API) that helps in allowing the development of applications that access or direct in accessing the features of an operating system and application. A standard as well as portable API of Java Jersey framework known as JAX-RS is being developed that is basically open source framework providing full implementation support for REST web services. The APIs of Java Jersey framework actually is dedicated towards simplifying the writing task for REST web services clients [31]. JAX-RS is most predominantly used in the domain of production and development today and it is backed by Oracle. The framework helps in streamlining the process of integration, configuration and pointing out the bugs within the server in an automatic manner. It is basically deployed within a servlet container and it can be easily embedded with simple java programs [32].

8.3 Web Service Implementation

In this thesis we designed a REST based web service in JAVA using the JERSEY framework. This web service acts a gateway through which we can access the database.

8.3.1 Interface and Queries

The figure below shows the welcome page of the web service interface. The interface can be accessed by passing the url followed by query to be executed. The web service is presently a read only service and any kind of addition, alteration or deletion of data is not supported at this moment.


Welcome to Techlok Database Service

Please pass the intended query in the address bar.
For eg : http://techlok.database.service/query/sysdate

Figure 8.2: Welcome page of the web service

By running the query "version" we can ascertain that the web service is running. We have provided a basic query "sysdate" which returns the database date and time. If this query executes, then we can ascertain that the database server can be reached. Presently we support basic queries to retrieve tours which are relevant to power converter unit. The query "getIncidents" fetches all the power converter failure tours. While query "getNonincidents" returns all the relevant tours with no imminent failures. These results(tours) of these queries are outputted as a arff file. All the other supported queries is shown in the table below.

Query	Description
get_incident_metadata	Retrieves all(40) incident tour metadata,(StartTime, EndTime, TourNo)
get_incident_metadata_for?trainno=	Retrieves incident tour metadata for specific train
get_nonincident_metadata	Retrieves all(300) non incident tour metadata,(StartTime, EndTime, TourNo)
get_nonincident_metadatafor?trainno=	Retrieves nonincident tour metadata for specific train
get_all_pcf_failures	Retrieves all diagnostic code relevant to power convertor unit
get_all_env_vars	Retrieves all environmental variables relevant to power convertor unit
getIncidents	Retrieves all the power converter failure tours
getNonIncidents	Retrieves all the relevant tours with no imminent failures

Table 8.2: Queries supported by the web service

8.3.2 Advantages

1. Access the database through fixed set of queries.
   Database queries are realized in the Web Service. User can access the database only through these fixed set of queries, avoiding directing querying or making local queries on client which can crash or are having a long run time. This interface also creates a controlled environment for data modification or addition minimizing the data corruption risks.

2. Creating new versions is easy.
   Certain queries can be extended later on by adding a new version of the query. It is not necessary to stop the web service to add these new version or even to pull down an old version.

3. Separation of concern.
   The layer architecture makes the design quite simple and easier to extend.

9 Experiment Setup

We ran a total of 43335 experiments. This couldn't be possible without a solid experiment architecture. Although the number seems a lot to handle, the architecture is quite easy to understand and implement, and also easily scalable. Creating experiments is done in the steps as shown in the figure.

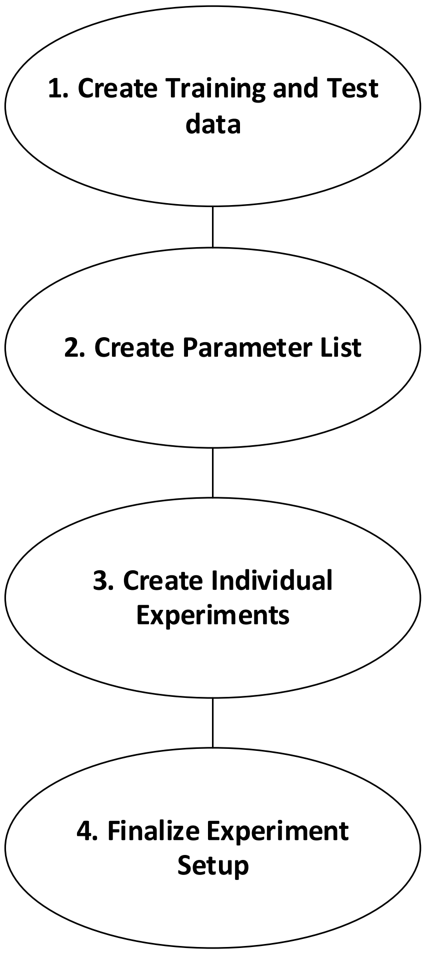

1. Create Training and Test data
2. Create Parameter List
3. Create Individual Experiments
4. Finalize Experiment Setup

Figure 9.1: Steps done while creating experiments

9.1 Create training and test data

A key consideration for creating experiments is the test and training data relevant to that experiment. For all experiments, we trained the one class SVM exclusively only on non incident tours(when there was no failures) and tested this model on a combination of incident and non incident tours to understand the performance of the model created in all cases.

We had a total of 300 non incident and 40 incident tours to work with. So we created 5 folds, each fold containing 240 non incident tours for training, and combination of 40 incident and 60 non incident tours for testing. In each fold 60 non incident tours used for testing varies, so we make sure each non incident tour is used for training four times and exactly one time for testing. While the same 40 incident tours are tested in each of the 5 folds.

9.2 Create Parameter List

We focused on the four main kernels of one class SVM individually. For each kernel, based on their relevant parameters, experiments sets were created programmatically by shuffling their parameters values. Lets consider an example, suppose kernel X has two parameters a and b.

Parameters	**Values**		
a	0	1	
b	5	10	15

Table 9.1: Example with two parameters

Then for parameter values as shown in the table above, we can make a total of six experiments by shuffling parameters and creating different combinations such as,

1. $-a0-b5$
2. $-a0-b10$
3. $-a0-b15$
4. $-a1-b5$
5. $-a1-b10$
6. $-a1-b15$

9.2.1 One class SVM Parameters

As discussed in the One class SVM chapter earlier, parameter selection is the key to derive meaningful results. For beginners who are not familiar with SVM this guide[33] proposes simple procedure which usually gives reasonable results.

Parameter value ranges,

- NuSVC
  As discussed earlier, parameter ν controls the training errors and number of support vectors. The value of ν is in the interval (0,1]. In this the upper bound is on the fraction of training errors and a lower bound of the fraction of support vectors. The default value of ν is 0.5. In our experiment we considered 9 values for ν between 0.1 and 0.9.

- Tolerance of termination criterion
  When the gradient is less than or equal to the tolerance value the training will be terminated. The default value is 0.001. In our experiment we tried values in the range of 0.001 to 0.1.

- Degree
  This is the degree of the polynomial function. The default value is 3. We tried a range from 1 to 5.

- Gamma
  Since, $\gamma = \frac{1}{The\ number\ of\ features}$
  We have 900 features, so we considered values from range of $\frac{1}{900}$ to $\frac{1}{100}$.

9.2.2 Linear Kernel Parameters

For linear kernel, the most relevant parameters are ν(N) and tolerance of termination criterion(E). We also tried the shrinking heuristics only in the case of linear kernel as other parameters were less, and we wanted to see whether it makes a significant difference or not. Table below shows all parameter values for linear kernel considered in our experiments, a total of 198 combinations can be made from these values.

Parameter	**Sign**	**Values**										
SVM Type	S	2										
Kernel Type	K	0										
NuSVC	N	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9		
Tolerance	E	0.001	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
Shrinking heuristics	H	0	1									
Cache memory size	M	100										

Table 9.2: Linear kernel parameters values

9.2.3 Polynomial Kernel Parameters

For polynomial kernel, the most relevant parameters are degree(D) which is the degree of the polynomial function, kernel coefficient gamma(G) and independent coefficient term (R) . Table below hows all parameter values for polynomial kernel considered in our experiments, a total of 1125 combinations can be made from these values.

Parameter	**Sign**	**Values**				
SVM Type	S	2				
Kernel Type	K	1				
NuSVC	N	0.1	0.3	0.5	0.7	0.9
Degree of Polynomial	D	1	2	3	4	5
Gamma	G	0.001111	0.001429	0.002	0.003333	0.01
Coeff0	R	0.01	0.05	0.09		
Tolerance	E	0.01	0.05	0.09		
Cache memory size	M	100				

Table 9.3: Polynomial kernel parameters values

9.2.4 Radial Basis Function Kernel Parameters

For radial basis function kernel, the most relevant parameters are are ν(N), tolerance of termination criterion(E) and kernel coefficient gamma(G). Table below shows all parameter values for RBF kernel considered in our experiments, a total of 891 combinations can be made from these values.

Parameter	**Sign**	**Values**										
SVM Type	S	2										
Kernel Type	K	2										
NuSVC	N	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9		
Gamma	G	0.001111	0.00125	0.001429	0.001667	0.002	0.0025	0.003333	0.005	0.01		
Tolerance	E	0.001	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
Cache memory	M	100										

Table 9.4: Radial Basis Function Kernel parameters values

9.2.5 Sigmoid Kernel Parameters

For sigmoid kernel, the most relevant parameters are kernel coefficient gamma(G) and independent coefficient term (R). Table below shows all parameter values for sigmoid kernel considered in our experiments, a total of 675 combinations can be made from these values.

Parameter	**Sign**	**Values**								
SVM Type	S	2								
Kernel Type	K	3								
NuSVC	N	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Gamma	G	0.001111	0.001429	0.002	0.003333	0.01				
Coeff0	R	0.01	0.03	0.05	0.07	0.1				
Tolerance	E	0.001	0.05	0.09						
Cache memory size	M	100								

Table 9.5: Sigmoid Kernel parameters values

9.3 Create Individual Experiments

Now with different kernel parameter, data fold and decay rates we can create a range of experiments. Decay rate and what each experiment does is explained in detail in the next chapters.

$$Total\ Experiments = Parameters * Number\ of\ Folds * Decay\ Rates$$

$$Total\ Experiments = (198 + 1125 + 891 + 675) * 5 * 3 = 43335$$

9.4 Finalize Experiment Setup

Finally we make a parent group folder named after that respective kernel. This group folder will have all experiments related to this kernel with all parameter value and data combinations. We can then execute all experiments in this entire group together and obtain group level summaries to understand how good the parameters combinations were. The figure below shows how the architecture eventually looks

10 Experiments

All individual experiments are similar to each other, they have the same execution pattern, the same set of data of that experiment group and they also generate the same output. However the parameter combination in each experiment is unique. Each experiment also has a run experiment batch command, so experiments can also be run individually.

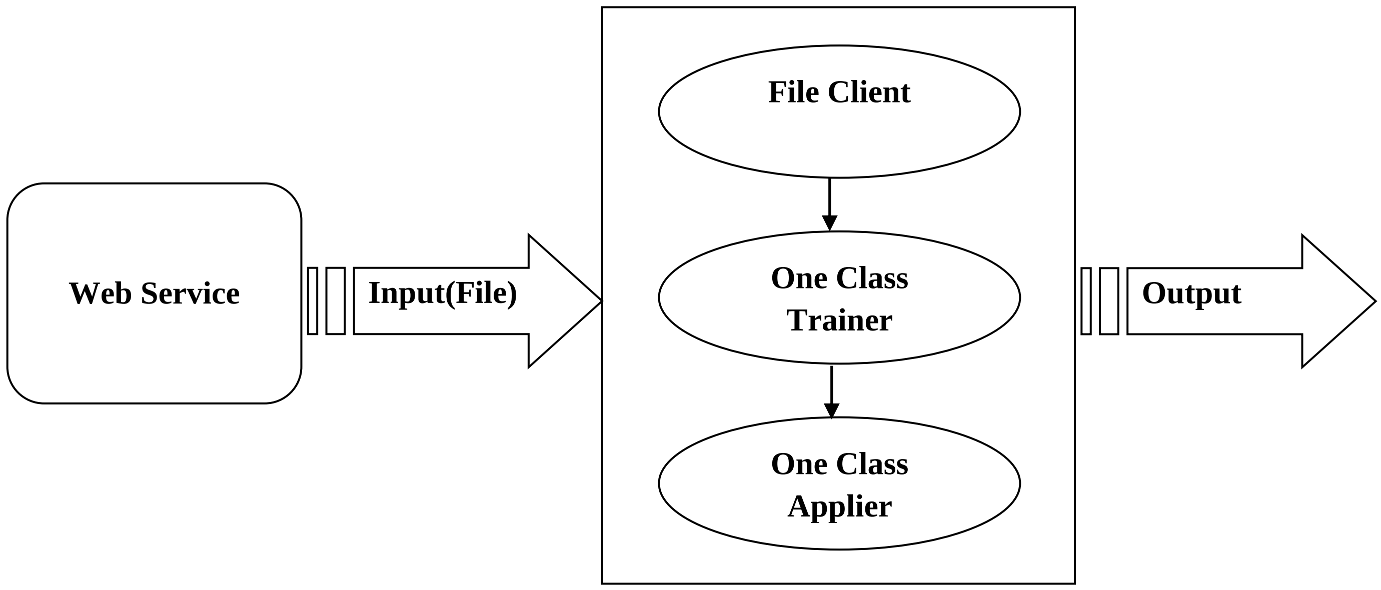


Figure 10.1: Experiment Architecture

10.1 File Client

Handles the files handed over by the web service. Training can be done on the entire sets of non incident tours combined in one huge file, however the model created after training must be applied to individual tour files for testing. This tells us how good the model is able to classify the test instances for that particular tour. For this purpose we created a routine to split sets of tours to individual tour file.

10.2 One Class Trainer

The one class trainer trains the support vector machine based on the parameters settings provided. Training is done on one complete training file consisting of all non incident tours. As an outcome a trained model is created, which can be then applied to test data to classify instances.

10.3 One Class Tester

Once the SVM is trained, the model can be applied on test instances. Each individual test tour file is handed over as input to the model, which in turn classifies each instances and creates an output. This output is then written to a CSV file.

10.4 Output

Considering the huge number of experiments, the process of collecting meaningful results is also a challenge. To do this we used an hierarchical approach on lines with experiment setup. The results are available on three levels.

1. Tour level : Each tour details
2. Individual Experiment level : Summary of this particular experiment
3. Data fold Level : Summary of all experiments in this particular fold

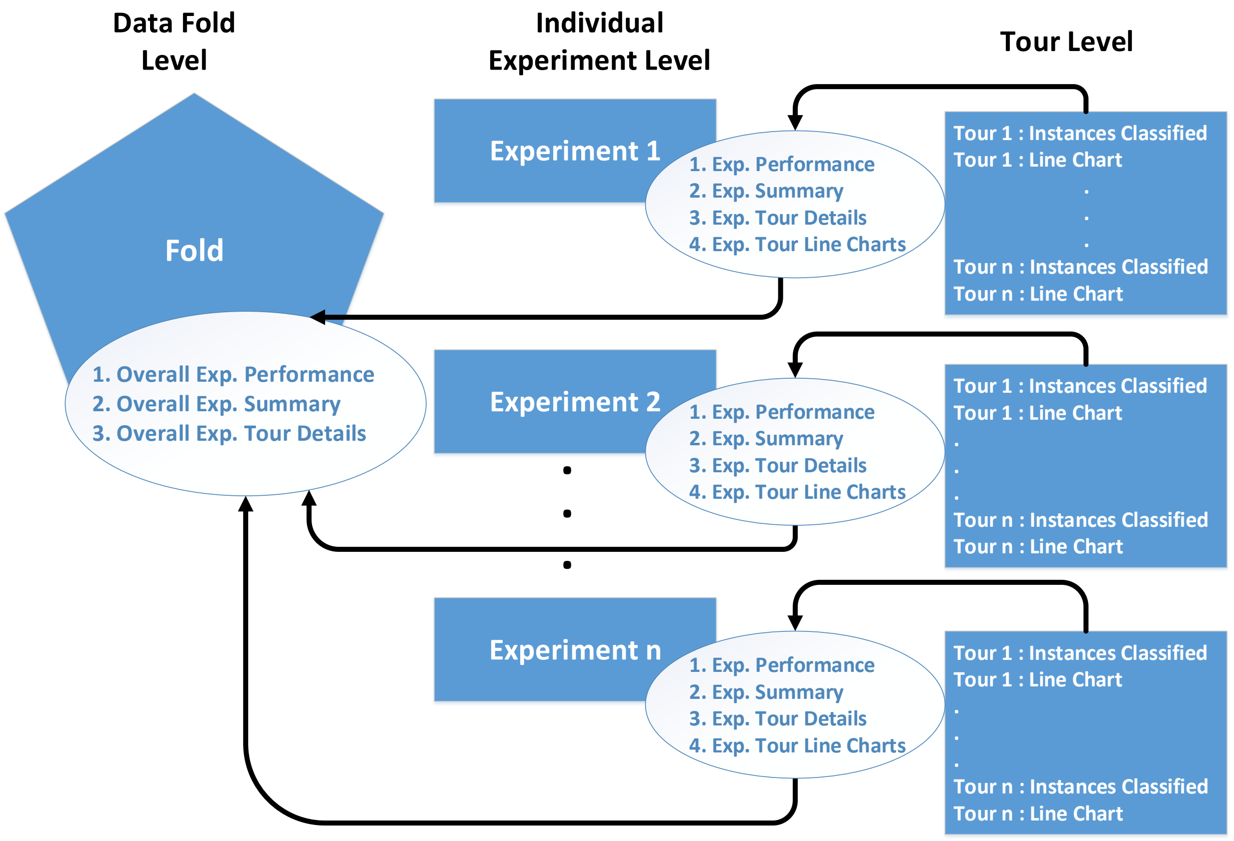


Figure 10.2: Experiments Results Hierarchy

The diagram above elaborates the results collection hierarchy. The tour details is the most basic level of results which in turn add up to create experiment level result. Then the results from all individual experiments make up the data fold level summary. Summary of the entire setup can be made by adding summaries from all data folds. Higher level results are actionable. The lower level reports are good for viewing minute details, but given the number of experiments and corresponding number of tours tested this cant be directly used.

10.4.1 Tour level results

Two tour level results are created,

1. Instances Classified

   These are the most basic results generated by each experiment on each tour tested(provided as input to classify). A csv file is generated as output for each tour, a sample of which is shown in the table below.

Instance Classified As	Linux Timestamp	Actual Timestamp
1	1417378591	30/11/2014 21:16:31
?	1417378600	30/11/2014 21:16:40
1	1417378620	30/11/2014 21:17:00
?	1417378626	30/11/2014 21:17:06
1	1417378627	30/11/2014 21:17:07
?	1417378645	30/11/2014 21:17:25
?	1417378649	30/11/2014 21:17:29
1	1417378755	30/11/2014 21:19:15
?	1417378759	30/11/2014 21:19:19
1	1417378761	30/11/2014 21:19:21
?	1417378767	30/11/2014 21:19:27
1	1417378792	30/11/2014 21:19:52
?	1417378795	30/11/2014 21:19:55

Table 10.1: Sample Classified Tour Details

In the sample table shown above, we have two distinct values, ? and 1. These questions marks indicates that those particular instances could not be classified. These instances which could not be classified is considered as an anomaly. If a large number of instances are unclassified then we can assume that it was an incident tour. The output also has Linux time stamp which are useful in indicating the progress of the tour along with realizing the time span of incident occurrence. So it is and easier to imagine failures along with time as a dimension. The actual file has all the existing features which were considered for classification only. There are a total of 900 features(i.e columns), however all of them are not relevant for our results so they are not shown here too.

2. Tour Line Chart

A line chart is generated for each tour with the inputs being the instances table as shown above.

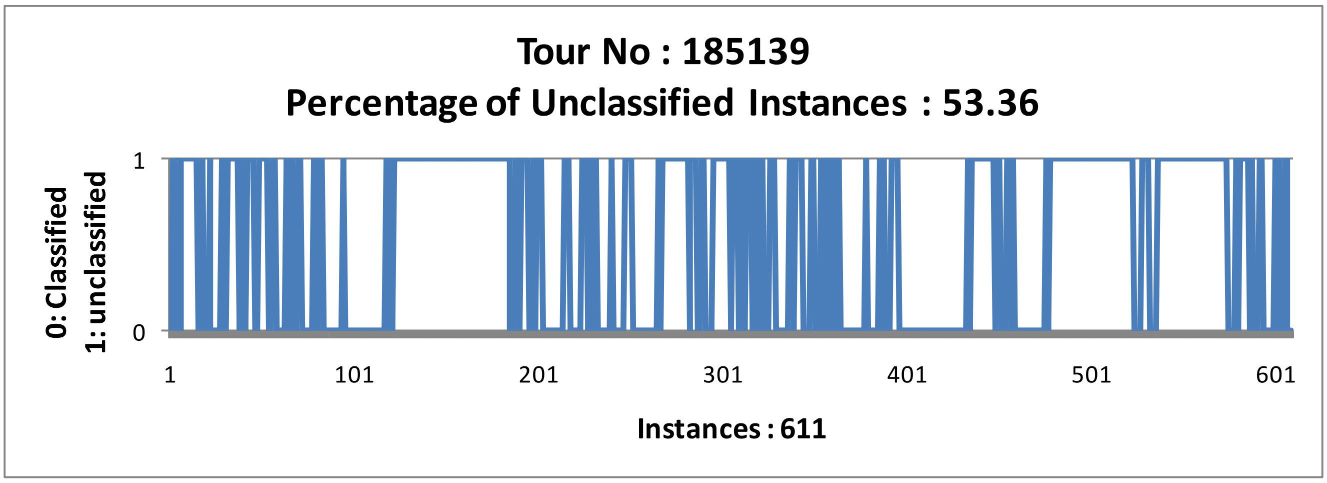


Figure 10.3: Incident line chart

The instance level details are good to study minute details of each tour, but each tour has far too many instances, plus they are also generated as a csv file which is difficult to comprehend. The line chart is a visualization of this csv file. We can also understand at what specific time things go wrong. We used the JFreeChart [34] which is an open source Java chart library for designing these line charts

Two sample line charts are shown here, the first one being of an incident case. In that line chart we can clearly observe that there are far too many unclassified instances. We can also notice that there is a concentrated set of unclassified instances in the beginning of the tour, which keep on shuffling as the tour progresses. However the non incident case as shown the line chart figure below, the trend is quite stable.

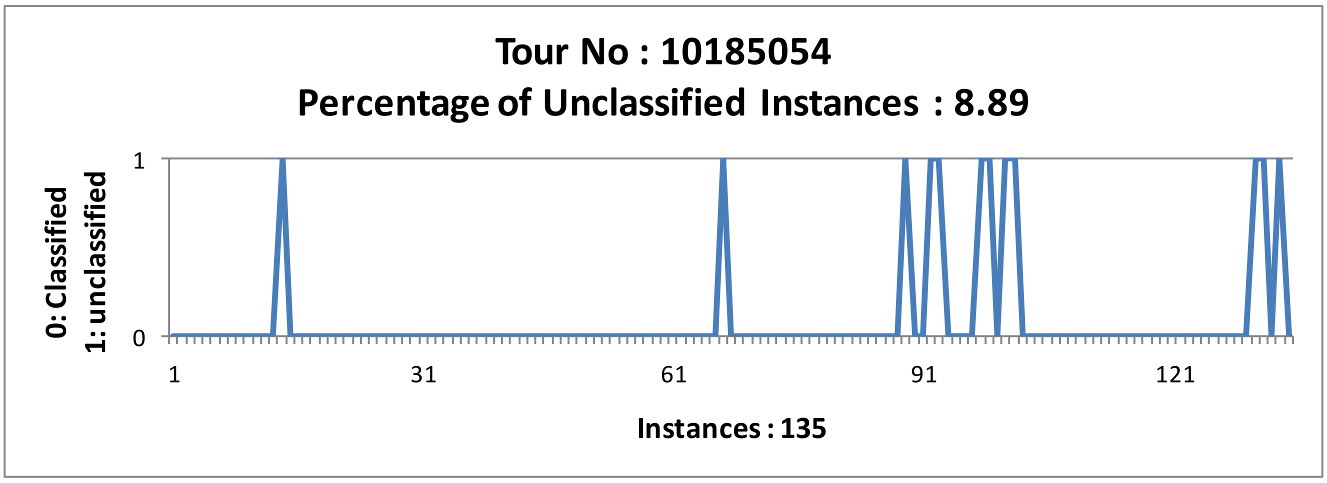


Figure 10.4: Non Incident line chart

10.4.2 Individual Experiment level results

The next level of reporting is on each individual experiment, this is required because for each experiments there are 100 tours which are tested. These generates 100 line charts which cannot be always checked considering the large number of experiments. So we make the following experiment level reports.

1. Experiment Tour Line Charts

   All the line charts generated by the experiment is collected and saved in a single pdf file. So instead of viewing 100 images separately, they all can be seen in one document. We used itextpdf [35] java classes to create these pdfs.

2. Experiment Tour Details

   An experiment tour details is generated as a csv file. This includes all the tours which were tested, percentage of unclassified instances and maximum activation value(These two values are explained in the next chapter). We already know whether the tour is incident or non incident (Tour no range above 10000000 are non incident tours) as we provide them as input for testing. This is indicated in the column ,"Actual". Based on assumptions made for setting threshold for calculating performance, the routine can decide whether that specific tour is incident or non incident. This is indicated in the column "Prediction".

Tour No	**Percentage of UnClassified Instances**	**Max Activation Value**	**Actual**	**Prediction**
185118	16.51376	9.028	Incident	Non_Incident
12185195	2.5	1.985	Non_Incident	Non_Incident
12185199	8.387097	2.028	Non_Incident	Non_Incident
13185132	43.58047	156.798	Non_Incident	Incident
185160	98.2906	108.741	Incident	Incident
185092	25.15337	25.074	Incident	Non_Incident
13185175	15.72327	9.306	Non_Incident	Non_Incident
14185099	8.974359	1.184	Non_Incident	Non_Incident
12185115	4.444444	1.549	Non_Incident	Non_Incident
13185084	18.56287	5.978	Non_Incident	Non_Incident
185095	30.76923	27.881	Incident	Non_Incident
185166	48.91775	169.238	Incident	Incident
12185120	2.173913	0.999	Non_Incident	Non_Incident

Table 10.2: Sample Experiment Tour Details

In the above table we can notice that tours 12185195, 12185199, 185160, 13185175, 14185099, 12185115, 13185084, 185166 and 12185120 are classified properly. While the other tours could not be classified accurately. Based on these, we can determine the performance of this test, which is then outputted into the experiment performance file. More the number of tours classified properly better the performance and vice versa.

3. Experiment Summary

   This table indicates the averages of the activation values and percentage of unclassified instances for both non incident and incident cases separately. This information is quite useful when deciding the optimal threshold to be set for determining the performance.

ExpNo	Parameters	Avg perc of Unclassified Instances for Incidents	Avg perc of Unclassified Instances for NonIncidents	Avg Activation for Incidents	Avg Activation for NonIncidents
14	-S 2 -K 2 -N 0.1 -G 0.0011 -E 0.001 -M 100	13.32324942	7.57221	11.1155	5.456417

Table 10.3: Sample Experiment Summary

4. Experiment Performance

   It is a single line csv file, which indicates the calculated performance metrics of this particular test. The various considerations and assumptions made to set threshold values to determine the performance is elaborated in detail in the next chapter.

ExpNo	Parameters	TP	TN	FP	FN	TPR	TNR	PPV	NPV	ACC	MCC
456	-S 2 -K 2 -N 0.5 -G 0.0011 -E 0.001 -M 100	58	3	37	2	0.966667	0.075	0.610526	0.6	0.61	0.093659

Table 10.4: Sample Experiment Performance

10.4.3 Data fold level results

These are top level reports, which describe how the entire experiment setup performed. These reports become extremely crucial because of the scale of experiments we run under a single fold. Also, the SVM kernels performance can be judged on this level. There are three types of reports available at this level.

1. Overall Experiments Summary

   All the individual experiment level summaries are collected into this one overall summary file. By analyzing this file we determine the corresponding threshold to be applied for all experiments. Further details is discussed in the next chapter.

ExpNo	Parameters	Avg perc of Unclassified Instances for Incidents	Avg perc of Unclassified Instances for NonIncidents	Average Activation for Incidents	Average Activation for NonIncidents
21	-S 2 -K 2 -N 0.1 -G 0.00125 -E 0.1 -M 100	13.27028	7.648251	20.0063	8.655467
64	-S 2 -K 2 -N 0.1 -G 0.0025 -E 0.09 -M 100	13.34165	7.993895	20.17205	9.414383
91	-S 2 -K 2 -N 0.1 -G 0.01 -E 0.03 -M 100	14.03482	8.457051	21.99705	10.81478
154	-S 2 -K 2 -N 0.2 -G 0.0025 -E 0.001 -M 100	25.36083	14.99148	45.57225	18.41163
219	-S 2 -K 2 -N 0.3 -G 0.00125 -E 0.1 -M 100	35.92032	22.48602	77.458	35.92395
306	-S 2 -K 2 -N 0.4 -G 0.001111111 -E 0.09 -M 100	45.61025	31.2322	103.7573	48.81212
362	-S 2 -K 2 -N 0.4 -G 0.0025 -E 0.1 -M 100	45.58046	30.93987	105.8358	50.94808
399	-S 2 -K 2 -N 0.5 -G 0.001111111 -E 0.03 -M 100	53.11886	38.62003	128.3816	61.19738
520	-S 2 -K 2 -N 0.6 -G 0.001428571 -E 0.03 -M 100	63.0996	46.96037	161.1412	74.51748
654	-S 2 -K 2 -N 0.7 -G 0.0025 -E 0.05 -M 100	73.35267	59.19977	186.7456	91.2449
790	-S 2 -K 2 -N 0.8 -G 0.01 -E 0.09 -M 100	85.54404	74.34831	212.2238	111.4929
868	-S 2 -K 2 -N 0.9 -G 0.003333333 -E 0.1 -M 100	96.00648	90.39755	228.3374	131.1063

Table 10.5: Sample Overall Experiments Summary

2. Overall Experiment Performance

   This is the most important report which describes the performance of the entire experiment setup. The number of experiments is equivalent to number of rows in the file. Each experiment number, along with the parameters used and performance achieved is properly mentioned in this csv file. Performance is indicated by using the metrics, true positives(TP), true negatives(TN), false positives(FP), false negatives(FN), true positive rate(TPR), true negative rate(TNR), positive predicate values(PPV), negative predicate values(NPV), accuracy(ACC) and Matthew's correlation coefficient(MCC). The table below shows a sample overall performance file.

ExpNo	**Parameters**	**TP**	**TN**	**FP**	**FN**	**TPR**	**TNR**	**PPV**	**NPV**	**ACC**	**MCC**
20	-S 2 -K 0 -N 0.1 -E 0.1 -H 0 -M 100	57	1	39	3	0.95	0.025	0.59375	0.25	0.58	-0.0625
22	-S 2 -K 0 -N 0.2 -E 0.001 -H 0 -M 100	51	2	38	9	0.85	0.05	0.573034	0.181818	0.53	-0.15657
43	-S 2 -K 0 -N 0.2 -E 0.1 -H 1 -M 100	51	2	38	9	0.85	0.05	0.573034	0.181818	0.53	-0.15657
44	-S 2 -K 0 -N 0.3 -E 0.001 -H 0 -M 100	48	3	37	12	0.8	0.075	0.564706	0.2	0.51	-0.1715
65	-S 2 -K 0 -N 0.3 -E 0.1 -H 1 -M 100	48	3	37	12	0.8	0.075	0.564706	0.2	0.51	-0.1715
66	-S 2 -K 0 -N 0.4 -E 0.001 -H 0 -M 100	45	5	35	15	0.75	0.125	0.5625	0.25	0.5	-0.15309
87	-S 2 -K 0 -N 0.4 -E 0.1 -H 1 -M 100	45	5	35	15	0.75	0.125	0.5625	0.25	0.5	-0.15309
88	-S 2 -K 0 -N 0.5 -E 0.001 -H 0 -M 100	37	5	35	23	0.616667	0.125	0.513889	0.178571	0.42	-0.28186
109	-S 2 -K 0 -N 0.5 -E 0.1 -H 1 -M 100	37	5	35	23	0.616667	0.125	0.513889	0.178571	0.42	-0.28186
110	-S 2 -K 0 -N 0.6 -E 0.001 -H 0 -M 100	30	10	30	30	0.5	0.25	0.5	0.25	0.4	-0.25
131	-S 2 -K 0 -N 0.6 -E 0.1 -H 1 -M 100	30	10	30	30	0.5	0.25	0.5	0.25	0.4	-0.25
132	-S 2 -K 0 -N 0.7 -E 0.001 -H 0 -M 100	23	14	26	37	0.383333	0.35	0.469388	0.27451	0.37	-0.26133

Table 10.6: Sample Overall Experiments Performance

   In the table above we can clearly see a trend of decreasing true positive rate and other factors with increase in value of N. What is also worth notable is that in all the other parameters do not influence the output at all. We can use the knowledge acquired from here to design more specific and better tests.

3. Overall Experiment Tour Details

This is just a basic addition of all the individual experiment level tour details into one csv file. These reports are used in conjunction with the overall accuracy reports. When accuracy of certain sets of experiments are good, we can just browse to the lower level summaries of those by filtering this file. This can be useful to understand which specific tours are creating problems and hampering the accuracy across various tests. A sample of this summary is shown below.

ExpNo	**TourNo**	**Percentage of UnClassified Instances**	**Max Activation Value**	**ExpectedResult**	**ActualResult**
19	185003	10.12365	12.039	Incident	Non_Incident
59	185003	10.66461	14.308	Incident	Non_Incident
73	185003	10.97372	14.87	Incident	Non_Incident
86	185003	11.59196	15.118	Incident	Non_Incident
104	185003	21.09737	76.9	Incident	Non_Incident
138	185003	21.25193	77.9	Incident	Non_Incident
372	185003	58.65533	545.269	Incident	Incident
441	185003	73.57032	737.269	Incident	Incident
638	185003	98.60896	1060.25	Incident	Incident
850	185003	100	1078.25	Incident	Incident

Table 10.7: Sample Overall Experiments Tour Details

In the table above we can clearly see that Tour Number 185003 which was being classified incorrectly as non incident in the beginning few sets of experiments, is later on classified correctly. So it can be inferred that parameters of those experiments are best suited for this tour.

11 Experiment Results

To obtain the results as shown in the last chapter, three sequential steps needs to be done. First creating the experiments, then running the experiments and the last one being gathering results. Out of which creating experiments and gathering results can be done relatively easily. However the time required for running all the experiments sets is huge, and cannot be done on a local machine.

11.1 Large scale parallel processing of experiments

From the fundamental code blocks for support vector training and classification to the overall architecture of our experiment, everything was designed in such a manner to exploit any parallel processing opportunities. So each experiment can run individually and generate results which can be later on picked up in the third step of gathering results.

For executing the experiments we used the Lichtenberg High Performance Computer of TU Darmstadt. The cluster provides 800 high performance nodes. We scheduled each experiment on individual nodes and ran them in parallel. In almost all cases we were able to get results from the entire experiment suite in couple of hours.

11.2 Threshold

Two different kinds of approaches to calculate the threshold were devised.

11.2.1 Percentage of unclassified instances threshold

If the percentage of unclassified instances for a single tour exceeds a certain threshold , then tour is classified as an incident tour, else it is classified as non incident tour. This is relatively simple and in the past has worked quite well.

$$Percentage\,of\;Unclassified\;Instances \quad = \quad \frac{Total\;Number\;of\;Unclassified\;Instances}{Total\;Number\;of\;Instances} \quad * \quad 100$$

11.2.2 Max activation threshold

We devised an activation routine and applied it on each tour after the support vector machine has finished classifying each instance of the tour. The activation function has following characteristics.

- Increment the activation value by 1, when an unclassified instance occurs.
- No changes to the activation value if a classified instance occurs.
- Gradually decrease the activation value with time. The decrease is based on the decay rate specified.

Activation values decay with time, this is calculated as,

$$Decay = (Previous\,Activation\,Value - (timeDiff * decayRate))$$

Where initially $Previous\,Activation\,Value = 0$

The current activation value is calculated by using the formula

$$Current\,Activation\,Value = Decay + Current\,Instance\,Value$$

Where $Current\,Instance\,Value = 1$ for unclassified instance and 0 for classified instance.

If the activation value is negative, it is reset back to 0. To explain this concept further lets considers some sample values as indicated in the table below.

Time	Value	Calculated Activation
1	1	1
7	0	0.94
10	1	1.91
60	0	1.41
77	1	2.24
170	1	2.31
400	1	1.01
412	0	0.89
566	1	0.35

Table 11.1: Sample instance values for a timeframe and corresponding activation

If the decay rate is set to 1.0% per second, then corresponding activation values obtained are indicated in the third column in the table above. The maximum activation value obtained is 2.31. The graph below illustrates how the activation function worked.

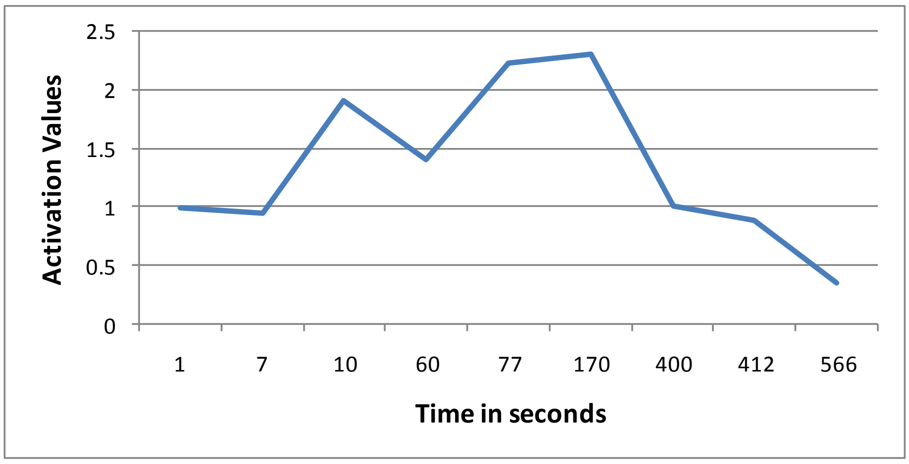


Figure 11.1: Sample activation graph

When the maximum activation value for a single tour exceeds a certain threshold, the tour is classified as an incident tour. This is based on the fact that when a failure occurs, many things go wrong simultaneously. A lot of diagnostic codes indicating various kinds of failures and warnings pop up in relatively small duration of time. We can hope this same behavior is reflected in the results too, and a lot of unclassified instances occur together indicating a failure.

11.2.3 Determining the optimal threshold

In both the approaches mentioned above, we use the statement, when the value of the tour exceeds a 'certain threshold', the tour is classified as an incident tour. Determining the optimal value to set is the challenge. We tried out different random values as thresholds but couldn't achieve good results. So we devised a strategy to calculate this threshold value by averaging the values of percentage of unclassified instances and maximum activation obtained from each tour of each experiment. As discussed in previous chapter, this information is stored in the overall experiment summary file. This value is then further averaged on each fold and final value obtained is set as the threshold. The table below shows one such example.

	Average Activation Values for Incidents	**Average Activation Values for NonIncidents**
Fold1	129.9683763	65.08846169
Fold2	127.266342	59.44375625
Fold3	133.8117052	94.92399261
Fold4	130.6260365	68.23726646
Fold5	129.8215529	57.40147005
Average	130.2988026	69.01898941
Threshold	**99.658896**	

Table 11.2: Example for determining optimal threshold value

11.3 Performance metrics

We considered the following performance metrics,

Metric	**Description**
True Positive (TP)	Non Incidents predicted as Non Incidents
True Negative (TN)	Incidents predicted as Incidents
False Positive (FP)	Incidents predicted as Non Incidents
False Negative (FN)	Non Incidents predicted as Incidents
True Positive Rate (TPR)	How often does the classifier predict Non Incident, when actual value is Non Incident
True Negative Rate (TNR)	How often does the classifier predict Incident, when actual value is Incident
Accuracy(ACC)	How often the classifier is correct
Matthews correlation coefficient(MCC)	Correlation between all the predicted values with the actual values

Table 11.3: Performance Metrics

The goal is to accurately predict both incident and non-incident scenarios. However we cant focus on accuracy alone as false alarms(false negatives), which predict an incident when there is no imminent failure, cannot be tolerated. So we set TPR (i.e non incident prediction rate)of above 90%. If the experiment passes this threshold, only then we consider the corresponding experiment's NPR (i.e incident prediction rate).

11.4 Results based on percentage of unclassified instances threshold

To recall, training was done on 260 non incident tours and testing was done on 60 non incident and 40 incident tours. The best results of five fold classification performed using all the four kernels is presented in the table below. Please note that these are average results of all five folds (Hence some of the TP, TN, FP, FN values have decimals).

Kernel	Parameters	TP	TN	FP	FN	TPR	TNR	ACC	MCC
Rbf	-S 2 -K 2 -N 0.3 -G 0.0016 -E 0.001	55.6	6.4	33.6	4.4	0.926	0.16	0.62	0.144755
Rbf	-S 2 -K 2 -N 0.3 -G 0.01 -E 0.001	54.2	6.4	33.6	5.8	0.903	0.16	0.606	0.109036
Rbf	-S 2 -K 2 -N 0.3 -G 0.00125 -E 0.001	55.8	6.2	33.8	4.2	0.93	0.155	0.62	0.142572
Rbf	-S 2 -K 2 -N 0.3 -G 0.0025 -E 0.001	55.4	6.2	33.8	4.6	0.923	0.155	0.616	0.131053
Rbf	-S 2 -K 2 -N 0.3 -G 0.0011 -E 0.001	55.8	5.8	34.2	4.2	0.93	0.145	0.616	0.129257
Sigmoid	-S 2 -K 3 -N 0.5 -G 0.00142 -R 0.05 -E 0.01	54	3.2	36.8	6	0.9	0.08	0.572	-0.01966
Sigmoid	-S 2 -K 3 -N 0.5 -G 0.002 -R 0.01 -E 0.05	57.6	1.2	38.8	2.4	0.96	0.03	0.588	-0.01275
Linear	-S 2 -K 0 -N 0.1 -E 0.001 -H 0	58.6	0.8	39.2	1.4	0.976	0.02	0.594	-0.01765
Poly	-S 2 -K 1 -N 0.1 -D 1 -G 0.0011 -R 0.01 -E 0.01	58.6	0.8	39.2	1.4	0.976	0.02	0.594	-0.01765
Poly	-S 2 -K 1 -N 0.1 -D 5 -G 0.0011 -R 0.01 -E 0.01	58.2	0.2	39.8	1.8	0.97	0.005	0.584	-0.08096

Table 11.4: Results based on percentage of unclassified instances threshold

In this set of experiments Rbf gives the best results with a maximum incident prediction rate of 16%. In comparison, the results from other kernels are not good.

11.5 Results based on max activation threshold

Based on the results from smaller trial runs, we considered three decays rates,

- Decay of 1.0% per second which is slightly fast decay rate.
- Decay of 0.5% per second which is a moderate decay rate.
- Decay of 0.1% per second which is a slightly slow decay rate.

11.5.1 Decay of 1.0% per second

The best results of five fold classification performed using all the four kernels and maximum activation calculated with a decay rate of 1.0% is presented in the table below.

Kernel	Parameters	TP	TN	FP	FN	TPR	TNR	ACC	MCC
Rbf	-S 2 -K 2 -N 0.3 -G 0.0011 -E 0.001	54	7	33	6	0.9	0.175	0.61	0.115994
Sigmoid	-S 2 -K 3 -N 0.5 -G 0.002 -R 0.07 -E 0.05	54.8	5.4	34.6	5.2	0.913	0.135	0.602	0.07883
Sigmoid	-S 2 -K 3 -N 0.5 -G 0.0011 -R 0.01 -E 0.05	54.8	4.6	35.4	5.2	0.913	0.115	0.594	0.057011
Rbf	-S 2 -K 2 -N 0.1 -G 0.01 -E 0.001	57.2	4	36	2.8	0.953	0.1	0.612	0.124118
Linear	-S 2 -K 0 -N 0.2 -E 0.001 -H 0	54.8	3	37	5.2	0.913	0.075	0.578	-0.01656
Linear	-S 2 -K 0 -N 0.1 -E 0.1 -H 1	58.6	2	38	1.4	0.976	0.05	0.606	0.098102
Poly	-S 2 -K 1 -N 0.1 -D 1 -G 0.0011 -R 0.01 -E 0.01	58.6	2	38	1.4	0.976	0.05	0.606	0.098102
Poly	-S 2 -K 1 -N 0.1 -D 3 -G 0.0011 -R 0.01 -E 0.01	58.4	2	38	1.6	0.973	0.05	0.604	0.092178

Table 11.5: Results based on max activation threshold calculated with a decay rate of 1.0%

The results of all kernels have improved in contrast to the percentage of unclassified instances threshold. However rbf still continues to give best incident prediction rate of 17.5%

11.5.2 Decay of 0.5% per second

The best results of five fold classification performed using all the four kernels and maximum activation calculated with a decay rate of 0.5% is presented in the table below.

Kernel	Parameters	TP	TN	FP	FN	TPR	TNR	ACC	MCC
Sigmoid	-S 2 -K 3 -N 0.2 -G 0.002 -R 0.05 -E 0.09	53.6	10	30	6.4	0.893	0.25	0.636	0.150988
Sigmoid	-S 2 -K 3 -N 0.2 -G 0.002 -R 0.07 -E 0.09	53.8	9.4	30.6	6.2	0.896	0.235	0.632	0.153285
Rbf	-S 2 -K 2 -N 0.3 -G 0.0011 -E 0.001	54.6	7	33	5.4	0.91	0.175	0.616	0.130292
Sigmoid	-S 2 -K 3 -N 0.2 -G 0.002 -R 0.07 -E 0.01	54.6	7	33	5.4	0.91	0.175	0.616	0.095531
Rbf	-S 2 -K 2 -N 0.2 -G 0.0011 -E 0.001	57.4	4	36	2.6	0.956	0.1	0.614	0.127007
Linear	-S 2 -K 0 -N 0.2 -E 0.001 -H 0	55.4	2.4	37.6	4.6	0.923	0.06	0.578	-0.02298
Linear	-S 2 -K 0 -N 0.1 -E 0.001 -H 0	58.8	2	38	1.2	0.98	0.05	0.608	0.106435
Poly	-S 2 -K 1 -N 0.1 -D 1 -G 0.01 -R 0.09 -E 0.09	58.8	2	38	1.2	0.98	0.05	0.608	0.106435
Poly	-S 2 -K 1 -N 0.1 -D 2 -G 0.0011 -R 0.01 -E 0.01	58.4	2	38	1.6	0.973	0.05	0.604	0.092178

Table 11.6: Results based on max activation threshold calculated with a decay rate of 0.5%

With the change in decay rate, the results for all kernels are fairly similarly with the exception of sigmoid in which results improves significantly and give best incident prediction rate of 25% (although the TPR is just below 90%).

11.5.3 Decay of 0.1% per second

The best results of five fold classification performed using all the four kernels and a decay rate of 0.1% is presented in the table below.

Kernel	Parameters	TP	TN	FP	FN	TPR	TNR	ACC	MCC
Rbf	-S 2 -K 2 -N 0.3 -G 0.00125 -E 0.001	54	10.2	29.8	6	0.9	0.255	0.642	0.210818
Sigmoid	-S 2 -K 3 -N 0.2 -G 0.00142 -R 0.03 -E 0.01	54	9.2	30.8	6	0.9	0.23	0.632	0.175534
Sigmoid	-S 2 -K 3 -N 0.2 -G 0.002 -R 0.07 -E 0.01	54.2	8.4	31.6	5.8	0.903	0.21	0.626	0.160145
Sigmoid	-S 2 -K 3 -N 0.2 -G 0.0011 -R 0.01 -E 0.01	54.8	7.6	32.4	5.2	0.913	0.19	0.624	0.168128
Rbf	-S 2 -K 2 -N 0.2 -G 0.0011 -E 0.001	57	5.8	34.2	3	0.95	0.145	0.628	0.181455
Linear	-S 2 -K 0 -N 0.2 -E 0.001 -H 0	55.8	2	38	4.2	0.93	0.05	0.578	-0.01585
Linear	-S 2 -K 0 -N 0.1 -E 0.01 -H 0	59	1	39	1	0.983	0.025	0.6	0.056569
Poly	-S 2 -K 1 -N 0.1 -D 1 -G 0.0011 -R 0.01 -E 0.09	59	1	39	1	0.983	0.025	0.6	0.056569
Poly	-S 2 -K 1 -N 0.1 -D 2 -G 0.0011 -R 0.01 -E 0.01	58.8	1	39	1.2	0.98	0.025	0.598	0.050337

Table 11.7: Results based on max activation threshold calculated with a decay rate of 0.1%

With the slowest decay rate, sigmoid gives an incident detection rate of 23%. However for rbf results improve further with the highest incident detection rate of 25.5%, which is also the best amongst all other experiments we had conducted in this thesis.

11.6 Top 20 results of all experiments

The table below shows the top 20 results of all experiments done in this thesis. It is quite clear that rbf and sigmoid kernels are best suited for our data. By using the activation function we were able to improve our results. Similarly setting a slower decay rate for activation function also yields better results.

Exp Type	Kernel	Parameters	TP	TN	FP	FN	TPR	TNR	ACC	MCC
Decay 0.1%	Rbf	-S 2 -K 2 -N 0.3 -G 0.00125 -E 0.001	54	10.2	29.8	6	0.9	0.255	0.642	0.210818
Decay 0.5%	Sigmoid	-S 2 -K 3 -N 0.2 -G 0.002 -R 0.05 -E 0.09	53.6	10	30	6.4	0.893	0.25	0.636	0.150988
Decay 0.5%	Sigmoid	-S 2 -K 3 -N 0.2 -G 0.002 -R 0.07 -E 0.09	53.8	9.4	30.6	6.2	0.896	0.235	0.632	0.153285
Decay 0.1%	Sigmoid	-S 2 -K 3 -N 0.2 -G 0.001428 -R 0.03 -E 0.01	54	9.2	30.8	6	0.9	0.23	0.632	0.175534
Decay 0.1%	Sigmoid	-S 2 -K 3 -N 0.2 -G 0.002 -R 0.07 -E 0.01	54.2	8.4	31.6	5.8	0.903	0.21	0.626	0.160145
Decay 0.1%	Sigmoid	-S 2 -K 3 -N 0.2 -G 0.0011 -R 0.01 -E 0.01	54.8	7.6	32.4	5.2	0.913	0.19	0.624	0.168128
Decay 0.5%	Rbf	-S 2 -K 2 -N 0.3 -G 0.0011 -E 0.001	54.6	7	33	5.4	0.91	0.175	0.616	0.130292
Decay 0.5%	Sigmoid	-S 2 -K 3 -N 0.2 -G 0.002 -R 0.07 -E 0.01	54.6	7	33	5.4	0.91	0.175	0.616	0.095531
Decay 1%	Rbf	-S 2 -K 2 -N 0.3 -G 0.0011 -E 0.001	54	7	33	6	0.9	0.175	0.61	0.115994
Percentage	Rbf	-S 2 -K 2 -N 0.3 -G 0.0016 -E 0.001	55.6	6.4	33.6	4.4	0.926	0.16	0.62	0.144755
Percentage	Rbf	-S 2 -K 2 -N 0.3 -G 0.01 -E 0.001	54.2	6.4	33.6	5.8	0.903	0.16	0.606	0.109036
Percentage	Rbf	-S 2 -K 2 -N 0.3 -G 0.00125 -E 0.001	55.8	6.2	33.8	4.2	0.93	0.155	0.62	0.142572
Percentage	Rbf	-S 2 -K 2 -N 0.3 -G 0.0025 -E 0.001	55.4	6.2	33.8	4.6	0.923	0.155	0.616	0.131053
Decay 0.1%	Rbf	-S 2 -K 2 -N 0.2 -G 0.0011 -E 0.001	57	5.8	34.2	3	0.95	0.145	0.628	0.181455
Percentage	Rbf	-S 2 -K 2 -N 0.3 -G 0.0011 -E 0.001	55.8	5.8	34.2	4.2	0.93	0.145	0.616	0.129257
Decay 1%	Sigmoid	-S 2 -K 3 -N 0.5 -G 0.002 -R 0.07 -E 0.05	54.8	5.4	34.6	5.2	0.913	0.135	0.602	0.07883
Decay 1%	Sigmoid	-S 2 -K 3 -N 0.5 -G 0.0011 -R 0.01 -E 0.05	54.8	4.6	35.4	5.2	0.913	0.115	0.594	0.057011
Decay 0.5%	Rbf	-S 2 -K 2 -N 0.2 -G 0.0011 -E 0.001	57.4	4	36	2.6	0.956	0.1	0.614	0.127007
Decay 1%	Rbf	-S 2 -K 2 -N 0.1 -G 0.01 -E 0.001	57.2	4	36	2.8	0.953	0.1	0.612	0.124118

Table 11.8: Top 20 results of all experiments

12 Conclusion and Future work

The main goal of this thesis was to try and predict a train component failure (power converter failure) by analyzing system generated logs. In order to achieve this goal, the following tasks were performed.

- Determining the SVM parameter values best suited for our data.
- Designing a web service to access the data source.
- Creating unique experiments with different parameterization.
- Designing an architecture to handle and parallely run multiple experiments.
- Devising an activation function to determine the threshold.
- Collection of meaningful results at all stages.

The results from the experiments are quite promising. RBF and Sigmoid kernels are particularly well suited our data. We achieved a maximum incident prediction rate of 25.5% which is quite good considering we put a cap of 90% and above on non incident prediction rate or false prediction rate.

The tasks accomplished in this master thesis, i.e the web service, experiment architecture design and the activation functions can be definitely be used in the future. The web service can be extended by adding more queries as per future requirements. Likewise similar experiment architecture and result collection strategy can be utilized for executing large number of experiments in the future. The activation function improved our results, this can be used in other problem cases too. As for one class support vector machines, the knowledge of the kernels, different parameterization and settings can be used to analyze other train component failures.

13 Appendix

13.1 Web Service Servlet

```
<servlet>
   <servlet-name>Jersey REST Service</servlet-name>
   <servlet-class>org.glassfish.jersey.servlet.ServletContainer
   </servlet-class>
   <init-param>
     <param-name>jersey.config.server.provider.packages</param-name>
     <param-value>techlok.database.service</param-value>
   </init-param>
   <load-on-startup>1</load-on-startup>
 </servlet>
 <servlet-mapping>
   <servlet-name>Jersey REST Service</servlet-name>
   <url-pattern>/query/*</url-pattern>
 </servlet-mapping>
 <resource-env-ref>
   <resource-env-ref-name>jdbc/techlok</resource-env-ref-name>
   <resource-env-ref-type>javax.sql.DataSource</resource-env-ref-type>
 </resource-env-ref>
```

13.2 Web Service Queries

```
// A non database query
@Path("version")
@GET
@Produces(MediaType.TEXT_HTML)
public String version() {
      return "Version : " + version;
}
// Return database date and time
@Path("sysdate")
@GET
@Produces(MediaType.TEXT_HTML)
public String sysdate() {
      return executeSqlQuery("select sysdate() from dual");
}
```

13.3 One Class SVM Trainer

```
//Train a SVM
public static LibSVM train(String trainingdata, String Parameters) {
        // Read Training data
        StringReader reader = new StringReader(trainingdata);
        Instances    train  = new Instances(reader);
        trainingdata        = "";
        // Set Training Class Index
        train.setClassIndex(train.numAttributes() - 1);
        //Get Parameters
        String[] options = Parameters.split(" ");
        // Set Parameters
        LibSVM       svm    = new LibSVM();
        svm.setOptions(options);
        // Train
        svm.buildClassifier(train);
        return svm;
}
```

13.4 One Class SVM Tester

```
//Apply  SVM on Test Data
public static Instances test(LibSVM svm, String testdata) {
        // Read Test data
        StringReader reader = new StringReader(testdata);
        Instances    test   = new Instances(reader);
        testdata            = "";
        // Set Testing Class Index
        test.setClassIndex(test.numAttributes() - 1);
        //Create new output instances
        Instances output = new Instances(test);
        //Test & fill output
        for(int i=0; i < test.numInstances(); i++) {
                double clslabel = svm.classifyInstance(test.instance(i));
                output.instance(i).setClassValue(clslabel);
        }
        return output;
}
```

13.5 Get Maximum Activation Value

```
// Decay rate is stated as a percentage.
// So if decay rate = 1 means, 1% decay every second.
public static Double getActivation(double decayRate, int[][] timevalueLst)
{
      // Variables
      Integer lastTime   = timevalueLst[0][0];
      Integer timeDiff   = 0;
      Double  decay      = 0.0;
      Double  currentAct = 0.0;
      Double  highestAct = 0.0;
              decayRate  = decayRate/100;
      // For each time value
      for (int i = 0; i < timevalueLst.length; i++){
          //Calculate time difference between current and last times
          timeDiff   = timevalueLst[i][0] - lastTime;
          //Calculate decay
          decay = (currentAct - (timeDiff * decayRate));
          //Calculate current activation
          currentAct = timevalueLst[i][1]  + decay;
          //Last time is incremented to current time for next loop
          lastTime   = timevalueLst[i][0];
          // Reset current activation value if it is below 0
          if (currentAct < 0) currentAct = 0.0;
          // Record the highest activation value
          if (currentAct > highestAct) highestAct = currentAct;
      }
      return highestAct;
}
```

Bibliography

[1] H. J. Shin, D.-H. Eom, and S.-S. Kim, "One-class support vector machines an application in machine fault detection classification," *Computers and Industrial Engineering*, vol. 48, no. 2, pp. 395 – 408, 2005.

[2] S. Mahadevan and S. L. Shah, "Fault detection and diagnosis in process data using one-class support vector machines," *Journal of Process Control*, vol. 19, no. 10, pp. 1627 – 1639, 2009.

[3] J. Downs and E. Vogel, "A plant-wide industrial process control problem," *Computers and Chemical Engineering*, vol. 17, no. 3, pp. 245 – 255, 1993.

[4] P. Lyman and C. Georgakis, "Plant-wide control of the tennessee eastman problem," *Computers and Chemical Engineering*, vol. 19, no. 3, pp. 321 – 331, 1995.

[5] D. Martínez-Rego, O. Fontenla-Romero, and A. Alonso-Betanzos, "Power Wind Mill Fault Detection via one-class nu-SVM Vibration Signal Analysis," in *International Joint Conference on Neural Networks (IJCNN 2011)*, pp. 511–518, 2011.

[6] S. Kauschke, *Nutzung Bahnbezogener Sensordaten Zur Vorhersage Von Wartungszyklen*. Darmstadt, Germany: Knowledge Engineering Group, TU Darmstadt, 2014.

[7] S. Kauschke, I. Schweizer, M. Fiebrig, and F. Janssen, *Learning to predict component failures in trains*. Aachen, Germany: CEUR Workshop Proceedings, 2014.

[8] S. Kauschke, I. Schweizer, and F. Janssen, *Advances in Predictive Maintenance for a Railway Scenario - Project Techlok*. Darmstadt, Hessen, Germany: Knowledge Engineering Group - TU Darmstadt, 2015.

[9] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling Up Machine Learning: Parallel and Distributed Approaches*. Cambridge , United Kingdom: Cambridge University Press, 2012.

[10] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining Practical Machine Learning Tools and Techniques - Third Edition*. Burlington, MA , USA: Morgan Kaufmann Publishers, 2011.

[11] A. O. Sykes, *An Introduction to Regression Analysis*. Chicago , USA: University of Chicago Law School, 1993.

[12] *Machine Learning - What it is and why it matters*. Jakarta , Indonesia: PT SAS Institute Indonesia, 2016.

[13] Harmelen, *Including Prestigious Applications of Intelligent Systems*. Lyon, France: 15th European Conference on Artificial Intelligence, July 21-26, 2002, 2002.

[14] X. Zhu, *Semi-Supervised Learning*. Madison, Wisconsin, USA: University of Wisconsin-Madison, 2008.

[15] J. Barnes, *Microsoft Azure Essentials Azure Machine Learning*. USA: Microsoft Press, 2015.

[16] B. Matthews, “Comparison of the predicted and observed secondary structure of t4 phage lysozyme,” *Biochimica et Biophysica Acta (BBA) - Protein Structure*, vol. 405, no. 2, pp. 442 – 451, 1975.

[17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The weka data mining software: An update,” *SIGKDD Explor. Newsl.*, vol. 11, pp. 10–18, Nov. 2009.

[18] L. Bulusu, *Open Source Data Warehousing and Business Intelligence*. USA: CRC Press, 2012.

[19] C. Cortes and V. Vapnik, *Support-vector networks*. Holmdel, NJ, USA: Kluwer Academic Publishers, 1995.

[20] B. E. Boser, I. M. Guyon, and V. N. Vapnik, *A Training Algorithm for Optimal Margin Classifiers*. New York, NY, USA: ACM, 1992.

[21] H. Yamada and Y. Matsumoto, *Statistical Dependency Analysis With Support Vector Machines*. Japan: In Proceedings of IWPT, 2003.

[22] W.-J. N. Chun-Tian Cheng, Zhong-Kai Feng and S.-L. Liao, *Heuristic Methods for Reservoir Monthly Inflow Forecasting: A Case Study of Xinfengjiang Reservoir in Pearl River, China*. Dalian, China: Dalian University of Technology, 2015.

[23] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, J. C. Platt, *et al.*, *Support Vector Method for Novelty Detection*. USA: MIT Press, 1999.

[24] L. M. Manevitz and M. Yousef, “One-class svms for document classification,” *J. Mach. Learn. Res.*, vol. 2, pp. 139–154, Mar. 2002.

[25] C. R. Souza, *Kernel functions for machine learning applications*. Creative Commons, 2010.

[26] E. Alpaydin, *Introduction to machine learning*. MIT Press, 2014.

[27] J. Cardoso, S. HansjÃűrg, Fromm. Nickel, G. Satzger, R. . Studer, and C. Weinhardt, *Fundamentals of Service Systems*. Switzerland: Springer, 2015.

[28] P. L. D. Zhao, *Geospatial Web Services: Advances in Information Interoperability*. USA: IGI Global, 2010.

[29] L. Dewailly, *Building a RESTful Web Service with Spring*. United Kingdom: Packt Publishing Ltd, 2015.

[30] P. Bucek, *Release Notes - Jersey 2.23.2*. Boston, USA: Free Software Foundation, 2016.

[31] M. Kalin, *Java Web Services: Up and Running*. USA: O’Reilly Media, Inc, 2014.

[32] B. Burke, *RESTful Java with JAX-RS*. USA: O’Reilly Media, Inc, 2010.

[33] C. wei Hsu, C. chung Chang, and C. jen Lin, *A practical guide to support vector classification*. Taiwan: Department of Computer Science, National Taiwan University, 2010.

[34] D. Gilbert, *The JFreeChart class library version 1.0.9: Developer’s guide*. Hertfordshire: Refinery Limited, 2008.

[35] B. Lowagie, *iText PDF Library Release Notes*. Cambridge, United States: iText Software Corp., 2009.