Performance Analysis: A Case study on Network Management System using Machine Learning

Poonem Kristhombuge Ananda Manoj Kumara

University of Tampere
School of Information Sciences
Computational Big Data Analytics
M.Sc. Thesis
Supervisor: Prof. Martti Juhola
June 2018

University of Tampere
School of Information Sciences
Computational Big Data Analytics
Poonem Kristhombuge Ananda Manoj Kumara: Performance Analysis: A Case study on Network Management System using Machine Learning
M.Sc. thesis, 66 pages, 35 index and appendix pages
June 2018

Businesses have legacy distributed software systems which are out of traditional data analysis methods due to their complexities. In addition, the software systems evolve and become complex to understand even with the knowledge of system architecture. Machine learning and big data analytic techniques are widely used in many technical domains to get insight from this large business data due to performance and accuracy. This study was conducted to investigate the applicability of machine learning techniques on performance utilization modelling on Nokia's network management system. The objective was to study and develop resource utilization models based on system performance data and to study future business needs on capacity analysis of the software performance to minimize manual tasks.

The performance data was extracted from network management system software which contains resource usages on system level and component level measurements based on input load. In general, the simulated load on a network management system is uniform with less variance. To overcome this during the research, different load profiles were simulated on the system to assess its performance. Later the data was processed and evaluated using set of machine learning techniques (linear regression, MARS, K-NN, random forest, SVR and feed forward neural networks) to construct resource utilization models. Further, the goodness of developed models was evaluated on simulated test and customer data.

Overall, no single algorithm performed best on all resource entities, but neural networks performed well on most response variables as a multivariable output model. However, when comparing performance across customer and test datasets, there were some differences which were also studied. Overall, the results show the feasibility on modeling system resource that can be used in capacity analysis. In future iterations, further analysis on remaining system nodes and suggestions have been made in the report.

Keywords and terms: Statistical modeling, machine learning, performance analysis, network management system.

Acknowledgement

I would like to specially thank Prof. Martti Juhola for his continuous support and providing positive feedback to improve my thesis during this period. Furthermore, I would like to thank Dr. Henry Joutsijoki and Prof. Jaakko Peltonen for reviewing their comments to improve the study.

I am also grateful to the members of Performance Estimation Team at Nokia, Tampere for their guidance and support in numerous occasions to improve this study. Specially to Markus Nenonen for providing me this opportunity and constant interest towards the study. Further, I would like to thank Petri Puustinen, Jyri Heinonen, Mikko Lahdensivu, Osku Hamalainen, Kari Lahdensuo, Lauri Makela, Jouni Kokkila, Kati Raunio, Carita Jarvinen and Panu Kortelainen for assisting me in numerous occasions.

I would like to thank all the lectures and fellow students at University of Tampere for their cooperation and of course friendship. Last but not the least, I would like to thank my wife and parents for supporting me throughout my studies and my life in general.

Contents

Table of Figures

1. Introduction

The development of mobile and radio networks has heightened the need for evolution on network management industry. During past two decades, mobile network technology has changed dramatically. The change is ongoing and expected to increase exponentially. In recent years, network traffic volumes have increased in the order of several magnitudes in a short period of time due to technologies and concepts such as 5G, IoT and smart devices. The compound annual growth rate for the period 2012–2016 was 78 percent. Based on the technology forecast, the industries are now preparing for an astounding data traffic increase by 2020 and beyond [1]. Therefore, network management companies need to facilitate the growth in underlying mobile and radio networks.

In general, designing an enterprise software system with overestimated capacity can cause extra unused resources with early purchase costs [2]. Furthermore, an overestimated capacity will bring extra associated costs such as energy, network, labor, and maintenance all of which are proportional to the scale of the infrastructure [3]. Conversely, underestimated capacity can cause high failure rates, performance issues and Service Level Agreement (SLA) penalties for the operators [2].

In every organization software applications cannot be fully independent from underlying legacy systems which are developed over their lifetimes using traditional or sometimes obsolete technologies [3]. Depending on the complexity and number of subsystems interacting with each other, system migration needs to be carefully addressed and it takes time. Further, on complex software system with its lifetime there can be problems on understanding the source code, increases on system deployment times, scalability issues with intensive data loads long-term commitment to selected technologies would initiate eventually as the number of subsystems and system size starts to grow [4].

Above facts presents the importance of performance modelling to efficient resource allocation, performance analysis and scalability. Further designing performance models should consider system hardware, software, system architecture, network connectivity and workloads in such a way that these models could be used to

analyze system performance as well as to predict performance on system which could variate based on workload and architectural changes. Another important aspect when modeling large systems is its scalability when including additional users, hardware or software to existing system [5].

System performance can be defined as a system's capability to handle effectively the tasks that it has been assigned to do in a timely manner [6]. Further, performance metrics can be categorized into three main categories: time taken to perform a service, the rate by which the service is performed and resource consumption of the service. In a short form, this can be defined as responsiveness, productivity (throughput) and utilization metrics [6]. Primarily the aim of this study is to investigate system performance with respect to resource utilization of network management system's computer nodes and responsiveness in certain computer nodes depending on the availability.

This study is conducted according to research requirements defined by Nokia Solutions and Networks, which provides network management solutions to mobile and radio networks. Current dimensioning tool used for software dimensioning and testing is mainly based on system expert's knowledge and initial set of performance models. As discussed earlier, incorrect estimates can cause situations where network management system is tested with over or underestimated dimensioning values which could eventually lead to problems in customer environments. By developing performance models based on system performance data, more accurate results can be obtained to understand the performance and scalability of the software system. The goodness of performance models can be validated with real business customer operated data.

This study investigates application of machine learning techniques in performance engineering to analyze, model and predict system capacity for future business requirements. The scope of the study does not include business application of the result models using performance models in real-life business scenario. The performance utilization data was extracted from the system, pre-processed and applied different machine learning methods based on selected set of base predictors (Multiple Linear Regression, Multivariate Adaptive Regression Splines (MARS), K-nearest neighbor (K-NN), Random Forest, Support Vector Machine) to implement univariable-output

performance models. Later this data was evaluated using Neural Networks to create multivariable performance models and finally the performance was compared against each method. This study expects to present performance models representing system utilization i.e. CPU utilization, memory consumption, disk I/O operation averages and Network I/O operation averages based on software related measurements to correspond to a selected subset of computer nodes in the network management system.

The objectives of this study are to: (1) evaluate the goodness of modelling performance utilization and responsiveness of the software, (2) understand performance bottlenecks of system and (3) understand any limitations of current performance metrics used on system modelling. The findings of the study are presented as univariable and multivariable-output models across distributed nodes in network management system, given by different machine learning techniques. The performance models in this study will facilitate the organization to determine the software and system performance in their current business process to,

- Determine the optimum sizing of the software system based on customer requirements
- Compare different software versions and environments
- Performance anomaly detection compared to baseline models
- Understand available capacity of system for scaling

This thesis report proceeds as follows. Next, chapter 2 provides background information and overview of network management domain and network management software system in question. Chapter 3 summarizes the literature review of data mining and performance analysis in distributed computer systems. Chapter 4 discusses data mining tasks, machine learning algorithms used in data modelling, and data mining tools used. Chapter 5 presents application of data mining and modelling on performance data. Chapter 6 and 7 discusses the methodology, which includes data preparation, modelling, evaluation and evaluation of the results. Finally, chapter 7 presents the summarized results with conclusions, recommended approaches and future works.

2. Machine Learning Applications in Performance Analysis

Many studies on performance analysis on cloud software systems have been presented in literature on the areas of cloud monitoring, failure recovery, auto-scaling, cloud capacity planning, response time and throughput analysis and load prediction using time series analysis.

Bai et al. [7] have evaluating performance of heterogeneous data centers using an analytical model. Based on the proposed model, several performance measures including mean response time, mean waiting time and the probability of immediate execution were analyzed. Moreover, to confirm the validity of the proposed model the experiment was followed by a simulation and authors claim that the proposed model can effectively estimate performance of heterogeneous data centers. Kafhali and Salah [8] report in their study about an analytical queuing model that can determine minimum resources required for hosting cloud application based on given workload conditions. In addition, these models are based on a defined set of key performance indicators (KPI) such as response time, waiting time, probability of immediate execution, CPU utilization, and throughput and finally cross validated by simulation on Java Modeling Tools. The study also used these analytical models to estimate overall system cost. Qiu et al. [9] presented hierarchical three phase recovery mechanism with rapid repair, diagnostic repair and complete repair actions based on the phenomenon of failure for distributed cloud systems.

In cloud computing, failure recovery is considered as one requirement which determines the performance of its systems. Qiu et al. [10] presented a theoretical model based on Markov chain to recovery process of the failed server as an efficient failure recovery mechanism. In another case, Bai et al. [11] presented a cloud service evaluation method for failures in virtual machines and servers based on complex networks according to their functional complexity. To support the required demand while maintaining service availability at minimum deployment cost, Azeez [12] presented a web service solution on Amazon EC2 to automatically scale web service applications to ensure required scalability requirements under optimum cost. In addition, Azeez addresses the limitations on cluster deployments of servers with the

concept of a few membership schemes to handle failures and dynamic load balancing on Amazon EC2 clusters.

Resource allocation and optimal workload allocation studies based on performance metrics such as response time, cluster consumption and workload packet loss rates are studies on [13-15] considering the importance of service level agreement (SLA) fulfilments. Furthermore, Xiong and Perros [16] discusses different approaches on minimizing the total cost of resources used by its applications in a cluster of computers while satisfying the quality of service. In their study, Kundu et al. [17] presented performance models that can predict system performance with a sufficient accuracy level. During modeling, they have selected a set of key system parameters which facilitate detailed reasoning for data center administrators that influence performance in virtualized environments. Further, they evaluated several techniques for modeling application performance and selected artificial neural network (ANN) as final approach for performance comparison which includes performance parameters such as CPU, memory, disk and network I/O [17].

In literature, there are many studies on performance modeling based on correlation between application performance and peak or average CPU utilization of the system. Dinda et al. [18] presented their models for application placement and predicting run time performance. Stewart et al. in [19] and [20] presented their models for capacity planning based on CPU utilization prediction under different workload conditions. Wood et al. [21] presented system profiling and modeling virtualized resource usage in cloud applications. A pattern matching prediction to identify similar past occurrences based on short-term workload history was presented by Caron et al. [22]. However, the method can be inefficient and time consuming for larger data sets as this requires searching similar patterns on the dataset. Further, Kim et al. [23] presented prediction technique using segment of most recent requests which define a boundary of data points to be analyzed. This can again not perform well in a generalized system due to scoping only to recent user request patterns.

An application of time series analysis techniques on performance and load prediction has broadly been used by many researchers in literature. As traditional techniques such as curve-fitting, moving averages and auto-regression methods

sometimes might not be effective compared to modern techniques due to drastic fluctuations in host load patterns in cloud environments, researchers try to create more effective methods on performance prediction. In their study, Di et al. [24] presented a Bayes method for cloud load prediction to achieve a better accuracy with a lower mean squared error. The suggested method predicts CPU and memory load on a host machine up to period of 16 hours. Cao et al. [25] suggests an ensemble model with the ability to update its base predictors dynamically so it can adapt the time series pattern changes. The base predictors include Auto-regression model, Exponential smoothing model, Weighted nearest neighbors (WNN) model and Most similar pattern model. Kourentzes et al. [26] propose a model ensemble operator based on kernel density estimation for one-step ahead forecast. Jheng et al. [27] presents a model which is capable on predicting future trends from the workload and shifts low priority tasks outside peak operating intervals to efficiently utilize the available resources. Wolski et al. [28] presents the effect of autocorrelation between successive CPU measurements in their study and developed one step ahead CPU prediction model to forecast CPU on a dynamic system.

Finally, this study was to explore performance modeling of network management system which deployed and operated on top of cloud native infrastructure. Focusing this aspect, the literature review was conducted to investigate the research on performance engineering and capacity analysis of distributed systems. Furthermore, the expectation was to shed light on the study by incorporating relevant concepts, practices and existing research findings. Overall, this review helped to brainstorm and shortlist study areas which suitable for the study.

Earlier System Modeling Study on Network Management System

In the previous study done by Tuisku [29] suggest the feasibility of modelling CPU performance of a single virtual machine in network management system. However, currently limited studies have being done on the subject network management system performance and amount of knowledge is known mainly based on system experts understanding the system. Considering the recent studies found in literature the expectation was to further evaluate machine learning approaches on software performance of network management system.

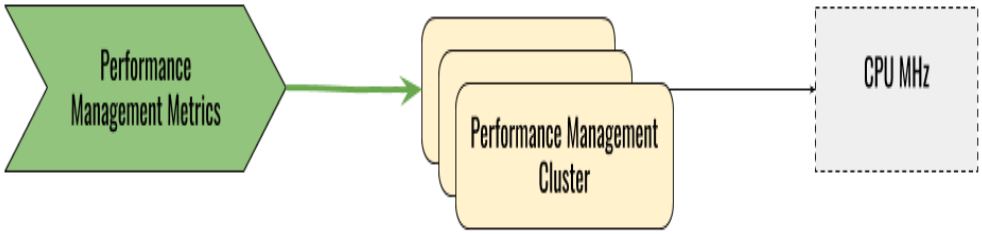


Figure 1 : Summary Representation of Earlier Study

Figure 1 shows the summary of the earlier study. This study was conducted on performance management service nodes to predict CPU consumption based defined performance management metrics. Further, the created models were evaluated using a real-world network management system, and as a result fairly accurate prediction on CPU utilization could be made.

Based on the background information there was some limitations in this early study. Firstly, the earlier study evaluated performance of CPU consumption only using multivariate regression technique. Secondly, during that stage there was limitations in collecting performance data from the software system which limited the scope of that study where only few load profiles could be tested. Thirdly, no customer data was available to compare performance levels between different environments of a created model.

3. Background and Overview

Nokia released its initial network management tool in the early 1990s and with the evolution in network management industry it has undergone tremendous changes to date in its lifecycle. In addition, telecommunication and radio network environments became more complex in the past decades and customers are interested in accurate resource usage predictions and indicators. In some scenarios as customers only have few network element types they are interested in a minimal software setup which can fulfil their requirements without overhead nodes. Based on these facts it is essential for the business to understand the performance of each network type and management software itself.

As discussed earlier, in practice, software systems evolve with time and become complex to analyze and troubleshoot. Even though it is recommended to understand and test the performance during product development stage, in practice it is difficult as developers prioritize functionality first. In many cases product performance is only evaluated at final stages of its release lifecycle. When considering the layout of enterprise systems, they consist of complex configurations, heterogeneous communication protocols, heterogeneous and geographically distributed servers with several network interconnections, proprietary middleware, large distributed database systems, load balancers and so on which make it difficult to understand its operation on runtime. During modeling understanding its system interactions is the most difficult and important stage during the process [5].

As defined by National Institute of Standards and Technology (NIST), cloud computing enables convenient, on-demand network access to a shared pool of configurable computing resources which includes servers, network, storage, applications and services [30]. Further, these resources should be efficiently provisioned and released with minimum effort based on vendor requirements which makes every organization to step into cloud infrastructure with the growth of their businesses.

In the meantime, by leveraging cloud services, organizations can deploy their software systems to address some of their scalability and performance issues with a minimum set of changes to their systems. To achieve this application scalability, one should use scalable architecture in the first place. Microservices technology is one of the

most famous cloud native architecture which enables availability and scalability in its design by facilitating the migration of on-premise architectures to cloud environments. In addition to this, microservices can simplify business processes by including them in collection of small services which could be deployed and scaled independently, as well as different technology stacks and are easily understood [31].

3.1. Mobile Network Evolution

During the past two decades, mobile network technology has evolved drastically. This technology hype is still ongoing and expected to increase exponentially with the upcoming technologies. In recent years, big traffic volume increased in the order of several magnitudes in a short period of time due to technologies such as 5G, IoT and smart devices. During the period of 2012–2016, the yearly growth rate of the market was 78 and now, based on their marker research telecom industry expects astonishing network traffic increases after 2020 [1].

When considering these future trends, network management aspects also need to evolve with the advancements in technology. During this, accurate load models will be a handy tool in the process of designing and dimensioning software systems.

3.2. Network Management

In general, mechanisms for monitoring, control and coordination of its resources is defined by system management standards. In a telecommunications management network, its resources are viewed as independent managed objects with well-defined properties to clearly define its managed operations. This is defined by Open Systems Interconnection (OSI) - Systems management overview published by International Telecommunication Union [32]. This defines the primary requirements for understanding the key functions of network management system as a model.

For convenience, requirements and specifications related to system management is categorized into five groups by OSI Management Framework and network management model which defines these major functions of network management systems. These groups are fault management, configuration management, accounting management, performance management and security management. This is sometimes defined by the

acronym FCAPS model. The accounting management category is sometimes replaced with administration on non-billing organizations [34].

3.3. Configuration Management (CM)

This module is responsible on managing, monitoring and tracking changes on system configurations of network hardware and software elements on the system. Some possible examples are updating OS version of a network device, adding a new device to the network and modifying running configuration of a device. It is important to keep track about updated configuration changes, software versions and system changes during troubleshooting network issues, and configuration management software facilitate this. In general, configuration management facilitates [32]:

- initialize and close managed objects
- collecting, storing and change the configuration of open system
- simplifying managing configurations of the devices, associate names with managed objects
- set the parameters that control the routine operation of the open system
- assisting future expansion and network scale planning

3.4. Fault Management (FM)

To distinguish different fault scenarios, elements in the managed network consist of monitoring and diagnostic tools. Each fault in an element is represented as an event and sent to the software system. The main requirement of this module is to recognize, isolate, correct and log faults that occur in the network. In addition, FM module facilitates trend analysis on error prediction, to detect abnormalities in network operations and to configure notifications to keep the network administrator informed about problems. These notifications can be set to trigger activities that can gather more information on recognizing the nature of the fault. Fault management function facilitates [32],

- maintain and examine error logs
- accept and act upon error detection notifications
- trance and identify faults
- carry out sequences of diagnostic tests
- correct faults

3.5. Performance Management (PM)

To ensure acceptable level of network performance, this module should facilitate continuous monitoring of network and guarantee optimum service to mobile subscribers. The module performance addresses the throughput, network response times, packet loss rates, link utilization, percentage utilization, error rates and so on. Based on these information network managers can evaluate the current network efficiency and prepare for future network demands.

Actively monitoring current network performance is an important step to identify existing and future issues to ensure reliability during operation. In business it is important to recognize system reliability and capacity issues before they affect any services in the system. This can be done based by network health monitoring and trend analysis using system performance data. This information in management system, can be monitored in real-time, or passively by configuring to alert based on predefined thresholds when performance deviates from the expected range. Furthermore, performance thresholds can be defined to trigger alarms depending on the severity level of the events which can be handled by the FM module. Performance management function facilitates [32],

- gather statistical information
- maintain and examine logs of system state histories
- determine system performance under natural and artificial conditions
- alter system modes of operation for the purpose of conducting performance management activities

3.6. Security Management

This module guarantees the basic security requirements of confidentiality, integrity and availability of its elements considering its users, data, software and network. This includes managing network authentication, authorization, and auditing to set correct permissions to access permitted network resources based on pre-defined security policies. Security management module is responsible to ensure network environment security and gathering security-related information to be analyzed. Security management function facilitates [32],

- creation, deletion and control of security services and mechanism
- distribution of security relevant information

- reporting of security relevant events

3.7. Accounting Management

This module enables charging capability to be established for the use of resources in open system interconnection environment, and for costs to be identified for the use of those resources. Accounting management function facilitates [32],

- inform users of costs incurred or resources consumed
- enable accounting limits to be set and tariff schedules to be associated with the use of resources
- enable costs to be combined where multiple resources are invoked to achieve a given communication objective

3.8. System Architecture

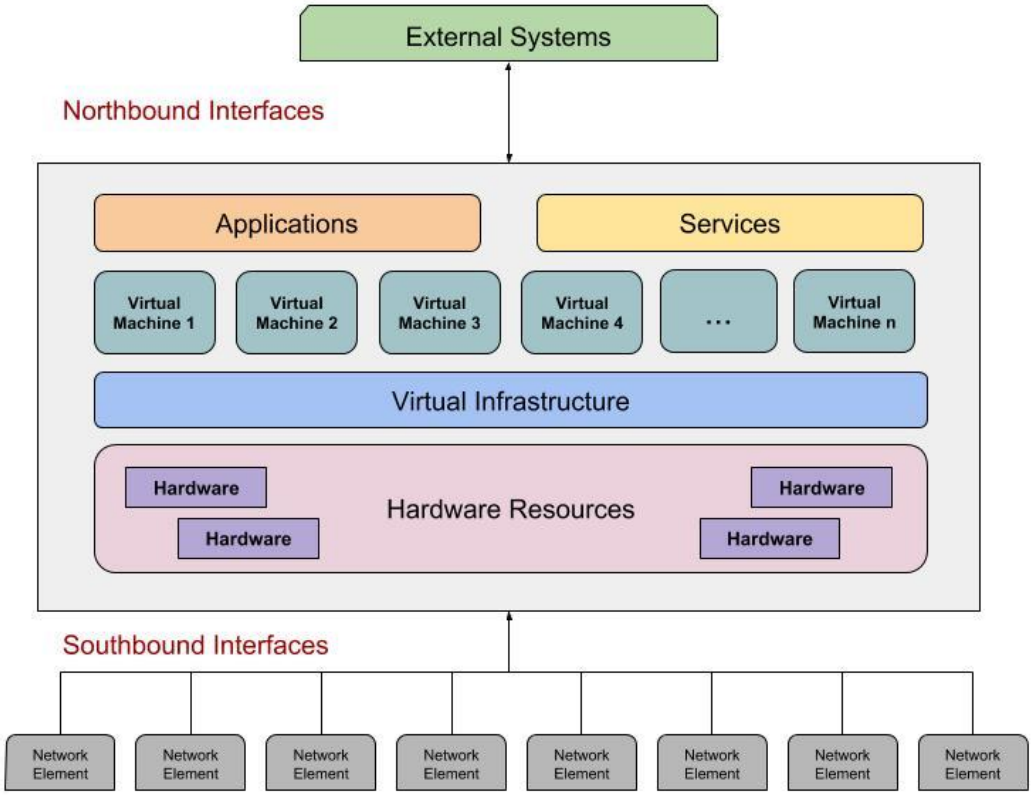


Figure 2 : Network Management System Architecture Diagram [33]

By its architecture, network management system has a modular based software system, which enables customers to customize required software features based on their

business requirements. These software modules are categorized into four main components based on the system management specifications as configuration, fault, performance and security management. Even though initial versions of a network management system were running on a dedicated hardware on physical servers, currently the system is running on the top of a virtualized environment aligning its way towards a fully automated cloud environment. This enables efficient resource allocation, scalability, reduced downtime, disaster recovery and ability for automation.

Through its southbound interfaces, communication happens between network elements and lower level systems of the managed network. This is mainly to obtain and provision data from the network. Further, the interfaces used to communication between software system and network elements are typically proprietary. Like in any other Infrastructure as a Service (IaaS) system, the hardware resources perform by means of pooled resources for the virtualized environment. The bridge between hardware and virtualized machines achieved by the virtualized layer. The northbound interfaces facilitate integrating software system with high level systems used for service management.

3.9. System Hardware

The software system can function independently from the underlying hardware resources due to its virtualized architecture. the division of hardware resources to virtual machines (VM) with a designated amount of hardware resources is handled by the platform virtualization software. Each virtual machine is allocated as per the configured amount of hardware resources to perform the intended task and these configurations can be defined as required based on the role of the individual VM.

3.10. System Performance

Performance is an indicator of how well the software meets its requirements for timeliness. System performance can be defined as a system’s capability to handle effectively the tasks that it has been assigned to do in a timely manner. Response time and throughput is used to measure the timely manner of the performance, and utilization metrics are used to measure the resource consumption. Moreover, the response time is defined as the time required to respond to an incoming request whereas throughput is the measure on how many requests can be processed in each time interval. Performance

parameters can be selected based on characteristics which influences system performance. In short, performance metrics are categorized based on above criteria's [6].

In software systems, there can be many parameters which affect system performance. Because there can be dozens of parameters, it is important to precisely select important parameters and their effect on performance. Furthermore, based on domain knowledge and earlier studies, certain parameters can be omitted or combined to create new features. During feature extraction process, statistical methods such as Principal Component Analysis (PCA) or Canonical Correlation Analysis (CCA) can be used. Based on an initial study, selected set of important parameters are shortlisted to examine during the analysis process as this will provide more simplified and generalized performance models.

3.11. Software Metric / Parameter

Standard measure of some characteristic or properties in a software system/ process can be defined as a software metric [35]. By defining metrics, different reproducible and measurable entities are expected to be obtained which could be used to have several applications in business analysis including software performance optimization.

In any business analysis process, it is really critical to understand and select important metrics to the business process. The 'HBR Guide to Data Analytics Basics for Managers' written by Harvard Business Review states "*You can't pick your data, but you must pick your metrics*." which implies the importance of defining proper metrics in any analytical study. In his presentation, Haff has presented some important rules when defining metrics [36].

- what's important to business/success criteria
- tied to business outcomes
- traceable to root cause(s)
- not too many metrics

3.12. Performance Metric

Performance metrics can be split into three main categories as time consumed on a given task, service performance rate and resource consumption of the service. In a short form, this can be defined as responsiveness, productivity (throughput) and utilization metrics [6].

When considering performance of a distributed computer system, the important operations to developer and system administrators are mainly corresponding to cluster health, resource utilization, performance and outages. Further when analyzing the system, following principles are important to consider:

- Define what you need to measure
- Selecting relevant metrics
- Quantity may not lead to quality of the process
- Understanding about what different measurements serve on different purposes
- Understanding how measurements drive behaviors

Performance parameters are measured mainly as utilization metrics corresponding to hardware performance and productivity metrics corresponding to workload. These performance parameters are later taken into account when deciding system capabilities and capacity analysis which will eventually decide on software dimensioning process. In Unix based systems, SAR (System Activity Report) system monitor command is widely used to collect and report system activity information. To record utilization metrics, SAR command is actively being used in computing as it not only has a wide range of measurements consisting of system load, CPU activity, memory, network I/O, disk I/O etc., but also it is easily integrated using *sysstat* package. To measure productivity metrics, software tools are used and these metrics are corresponding to performance management (PM) data (measurements and counters) and fault management (FM) data (events) discussed earlier.

3.13. Data Types in Network Management System

Even though the network traffic on mobile networks is caused as a result of its subscribers, in network management system it is different from this. The traffic on software system is based on its managed network elements such as network

performance, network failure or configuration change operations. This traffic enters the software system through its southbound interface. In this experiment PM data and FM data are mainly focused and these metrics are considered as the predictor variables during the modeling process.

Performance Management Data (PM Data)

Performance management data represents metric measurements composed by different network elements as counters. These metrics comprise of events, success rate, reset events, resource usage, signaling, etc. This measurement information can be pre-processed or post-processed in the network element based on the configuration and the type of network element. Measurement can be directly uploaded to the network management system's database as well. In addition, monitoring subscriber operations using PM data can be done by observing usage values of available services. When making management decisions based on service usage and when identifying current or future problems and opportunities, this information can be taken into consideration.

Fault Management Data (FM Data)

Fault Management data or shortly FM data mainly consist of events which can be categorized into several types for example, cancel, acknowledge, un-acknowledge and as a result FM events and alarms will be created in network management system. These alarms, when triggered represent a problem or error in a network element. In analytical perspective every FM event types are equally valued. In real life networks there could be correlations between FM and PM data as performance of the managed network can be affected by the number of fault events and the fault situation.

System Performance Data

In addition to network performance data types (PM and FM data), one can measure the data metrics related to system level performance of individual subsystems of the network management system. In practice this can be done based on individual virtual machine level considering its performance. The response variable data used in the study i.e. CPU utilization, memory consumption, disk I/O operation averages, Network I/O operation averages and response times are considered in this category.

3.14. Workload Data in Network Management System

Performance data in network management system can be divided into two categories, namely simulated workload data and actual workload data. The simulated workload can be synthetic and generated in a controlled and repeated fashion. The actual workload data consist of performance data corresponding to customer environments under real life operating environments. On capacity analysis process and modeling, data sets consisting of high variance will be useful since it will help to understand system boundaries. However, since in customer environments most cases run on given boundary levels, there can be less variance in the data. Due to this reason simulated workload data can be collected and used during analysis under controlled environment conditions. As it is flexible to variate incoming data rates to the system with well-defined simulated loads, the system can be analyzed iteratively to understand its overall behavior in a detailed manner.

When simulating workloads, an instruction mix is a specification of various instructions and their relative frequency defined based on the requirement. This can be constructed for the comparison of different processors on a given hardware environment. This approach can be utilized in a distributed computer system. [6]

Exploratory Statistical Modeling and Predictive Analytics

Based on the expected functionalities and operating conditions of the data models, model building procedure leads to numerous variances in explanatory modeling and predictive analytics methods. Depending on the context, models will contrast based on explanatory power of the models vs. predictive power of the models. There are two key differences between explanatory versus predictive analysis. The first difference is the properties lie in the data used in analysis where exploratory power is being assessed by means of in-sample goodness of fit procedures. In predictive analysis prediction accuracy measures are evaluated based on out-of-sample prediction procedures.

The second difference is in the metrics used in two techniques. Even though statistical significance is an important property when assessing exploratory power, it is not so important when assessing predictive performance. In addition, Wu et al. [37] theoretically justify that sometimes removing statistically significant predictors with small coefficients could result in improved prediction accuracy. Figure 3 1 shows some differences between two techniques with respect to different states of the analysis.

Table 1. Differences Between Explanatory Statistical Modeling and Predictive Analytics

Step	Explanatory	Predictive
Analysis Goal	Explanatory statistical models are used for testing causal hypotheses.	Predictive models are used for predicting new observations and assessing predictability levels.
Variables of Interest	Operationalized variables are used only as instruments to study the underlying conceptual constructs and the relationships between them.	The observed, measurable variables are the focus.
Model Building Optimized Function	In explanatory modeling the focus is on minimizing model bias. Main risks are type I and II errors.	In predictive modeling the focus is on minimizing the combined bias and variance. The main risk is over-fitting.
Model Building Constraints	Empirical model must be interpretable, must support statistical testing of the hypotheses of interest, must adhere to theoretical model (e.g., in terms of form, variables, specification).	Must use variables that are available at time of model deployment.
Model Evaluation	Explanatory power is measured by strength-of-fit measures and tests (e.g., R^2 and statistical significance of coefficients).	Predictive power is measured by accuracy of out-of-sample predictions.

Figure 3 : Steps in Explanatory Statistical Modeling vs Predictive Analytics [38]

4. Overview on Machine Learning

In the next few sections, machine learning algorithms used in data modeling task in the research work and software tools used for data mining are discussed shortly.

Machine learning applications in business

In any organization, a large amount of data is produced and accumulated over time in their system. This gives businesses an opportunity and competitive advantage to extract business knowledge from underlying data. Even though it is demanding to process this data, analyzing them in a timely manner is beyond the capabilities of traditional analysis methods used by many organizations. Advancements in modern machine learning and big data techniques have enabled processing databases with large volumes in efficient fashion, leading businesses to invest on knowledge discovery applications in business.

Data mining analytic techniques are evolving with time to meet new requirements and better accuracies for different use case. This enables ability to automate decision support systems in business processes with the help of integrated analytics and optimization algorithms. It is essential to process exponentially growing data volumes in real time as IBM forecasts the growth of next decade to increase from 800,000 petabytes to 35 zettabytes [39]. This motivates businesses to invest on processing data to acquire business intelligence which could help their business with modern advancements in technology. In addition, domain knowledge contributes the business analysis process to a success as it plays an important role during the process. In his study, Weiss [40] has discussed the importance of including domain experts in data mining study to improve effectiveness of the process.

4.1. Feature Selection and Extraction

In empirical modeling and machine learning, feature (variables, predictors) subset selection is done to select a subset of most relevant features during the model creation process. This will help to improve the interpretability of the constructed model. The idea of the process is to remove any redundant or irrelevant features without suffering much information loss from the original data frame, as it could consist of many features.

Redundant features are distinct from irrelevant features as, the redundancy could be due to presence of another strongly correlated feature [41]. Feature selection techniques can help when there are many features compared to the available sample data points in the data set. Feature selection stage is important as it will:

- simplify the constructed model
- reduce training time
- address curse of dimensionality
- To make models more generalized by reducing overfitting

During feature extraction, new informative and non-redundant features are derived based on of original features. One popular example for feature extraction is 'principal component analysis' which is inspired by statistics. There exist a few algorithms and variations that can be used for feature selection and extraction. Further, the result feature subset can be different based on the algorithm and properties of the data. In the next section some well-known feature selection techniques are discussed.

Exhaustive Feature Selection

The brute force technique is used on subset selection to generate every possible combination of feature subsets. This process guarantees to find the best fitting subset but as a drawback the cost of the process is high. The computation cost approximately doubles by adding one additional variable as for k number of features there will be 2^k- 1 possible subsets [42].

To reduce the computation cost without any information loss, there are few options available in exhaustive feature selection. Firstly, as it's less probable that a single response variable has many statistically significant predictor variables which equally improve the models, domain experts can help to assess on limiting maximum subset size. Secondly, effectiveness of the branch-and-bound search algorithm can be improved. These two steps can be helpful to improve the efficiency of the feature selection algorithm up to large datasets [42]. The branch-and-bound algorithm will evaluate best fitting subsets up to number of feature count. The computation cost is significantly reduced in cases where only few features are dominant compared to others.

Forward Selection and Backward Elimination

In wrapper methods a feature subset is used to train a model using input features. Iteratively by inferencing the created model, the feature selection algorithm will add or remove input features form its initial feature set. Even tough, the problem can be simplified to a search problem it can be computationally expensive. These wrapper methods can be categorized as forward feature selection and backward feature elimination which will be discussed next.

- **Forward Feature Selection**
  The algorithm starts with empty features set and iteratively adds most significant feature to the model to improve its performance. This step will be repeated until no improvements made to the model by adding of a new feature and final feature set will be selected at this stage.
- **Backward Feature Elimination**
  The algorithm starts with all the variables in its feature set and iteratively remove least significant feature from the model to improve its performance. This step will be repeated until no improvements made to the model by removing of an existing feature and final feature set will be selected at this stage.

Efficiency of these two methods is sometimes argued in comparison to each other. Some claim that forward selection is more efficient as opposed to defenders of backward elimination who claim that weaker subsets can be found by forward feature selection as the significance of variables are not assessed compared to variables not included yet [41].

4.2. Data Mining Tasks

The main categories in data analysis tasks are descriptive, predictive and prescriptive analysis [43]. In descriptive analysis task the purpose is to provide insight to business and its stakeholders based on the past data to understand any patterns which describe the phenomena related to the data [44]. In the case of predictive data analysis, the objective is to discover any patterns that can predict the unobserved future patterns. The predictive models can be utilized by organizations to make knowledge-driven proactive decision making to the questions that were complex or time consuming in general [45]. Further, by prescriptive data analysis, steps can be utilized to optimize

current procedures and to decide next steps when decision making is executed. Several data mining techniques can be presented as follows:

- ***Classification*** – The classification algorithm will map (classifies) the input data items into a pre-defined set of categories or classes. Some sample application would be identification of hand written digits from a large set of digit images. Once the model is developed it can be used in future inputs to recognize the digit in input image.

- ***Clustering*** – Clustering is a descriptive task where algorithm tries to identify a finite number of categories or clusters which could describe the data. This process is an unsupervised learning algorithm and output category is not known initially. In practice it is widely used in marketing to identify similarities between customers based on their purchase history and in many medical research studies.

- ***Prediction*** – Predictive models can be used to predict future trends or unknown conditions based on its past data or as a correlation of depending factors. For example, by using an effective predictive model to predict performance, business turnover or sales can help business to prepare for unseen future challenges.

- ***Anomaly detection*** – This technique can be used to detect significant differences compared to previously recorded data or reference levels on a given phenomenon. This technique is widely used in financial industry for fraud detection

- ***Summarization*** – This technique can be used to generalize or abstract the data into a simplified overview and comprises on providing compact description for a of dataset. This can be as simple as determining the mean and standard deviation for a feature in a table, to more sophisticated methods involving multivariate visualization methods.

- ***Dependency modeling*** – This technique can be used to find models that has significant dependencies among their variables. Dependency models can be

defined based on structural level or quantitative level of the models it specifies. The features that are locally dependent on each other and variable strengths of the dependencies will be evaluated in these two cases [44].

4.3. Machine Learning Algorithms

In this section, brief introduction on data mining and machine learning techniques which were evaluated during the modelling process are discussed. As the initial base predictors for modeling task, six machine learning algorithms were used. These algorithms are multiple linear regression, MARS, k-nearest neighbor, random forest, support vector regression and feedforward neural networks. To train the data models, supervised learning techniques were used. In supervised learning, every entry in the data set consists of precise output values which are used to train the models accordingly [46]. Introduction about selected machine learning techniques is presented below.

Multiple Linear Regression

In practice, descriptive modeling as well as predictive modeling is done using Multiple linear regression [48]. The model is constructed against response variable from a sample of data points, corresponding to its input variables. Most simple technique used in linear regression is the ordinary least squares method, which aims to minimize sum of squared error on model creating. Descriptive modeling uses available set of data to model existing features from the data. During predictive modeling, response variable values for new cases are predicted based on model constructed by existing predictor variables. The sample equation of multiple linear regression represents its response variable as a linear combination of its predictor variables. Below is a sample equation with *p* predictor variables:

$$\boldsymbol{Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots\ldots + \beta_p X_p + \varepsilon} \quad (1)$$

Equation 1 : Equation of Multiple Linear Regression

In the above equation, response variable is denoted by *Y* and the predictor variables are denoted by *X*'s. Further, β_0 denotes intercept and remaining β values represent the coefficients of each predictor variable. The error of the model is represented by ε. The process of predicting more than one response variables at once is

known as multivariate linear regression analysis. Multiple linear regression and multivariate linear regression modeling are two distinct techniques and should be used appropriately [49].

In multiple linear regression, to make reliable predictions with two or more predictor variables, the input data set should satisfy additional qualities compared to modeling using simple linear regression. The efficiency of the method depends on the ability of presenting the response variable as a linear combination of the predictor variables. The statistical significance of the model test can be disturbed by lack of linearity which causes model fit, errors and residuals. Further, to prevent multicollinearity, the predictor variables should not be correlated among each other and this can be detected through the variance inflation factor [50]. In addition, there can be situations where outlier data points get recorded due to measurement errors or unmeasured metrics [51].

Multivariate Adaptive Regression Splines

Multivariate adaptive regression splines (MARS) technique introduced by Jerome H. Friedman is a non-parametric regression technique. It's capable of modeling non-linearities and relations among features represented by the input data. MARS method builds models of the form as below,

$$\hat{f}(x) = \sum_{i=1}^{n} ci\ Bi(x) \qquad (2)$$

Equation 2 : Sample Equation of MARS Method

The model is represented as a weighted sum of basis functions $B_i(x)$ where c_i is a constant coefficient multiplied by its basis function. Each basis function can take one of the following three forms:

1. a constant
2. a hinge function. A hinge function has the form max (0, x-const) or max (0, const-x)
3. a product of two or more hinge functions

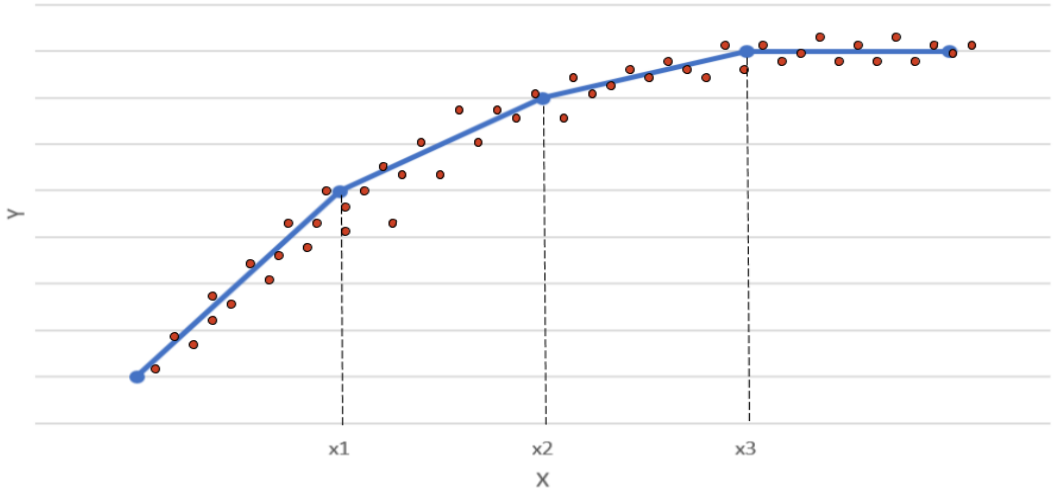


Figure 4 : Sample Model Representation using MARS Model

The term "MARS" is trademarked and in order to avoid trademark infringements, many open source implementations of MARS are called "Earth".

K-Nearest Neighbors (K-NN)

K-Nearest Neighbor algorithm is an instance based learning (IBL) technique, which is considered as one of the simplest methods. From the available data set all the known cases are stored by the algorithm to solve new cases. To determine the result for an unknown case, the algorithm will compare it with the similar instances in the training data. Further, this algorithm will assume that data points with similar attributes exist in close proximity compared to others and these nearby data points are called neighbors [46]. When predicting the class label for a new instance, the algorithm searches for K nearest training samples that are close to the new instance, and most frequent class value is assigned. Further, Euclidian distance or Cosine similarity can be used as similarity measure [52]. In practice different variations of distance functions being used are based on domain knowledge and properties of the data. Evaluating continuous variables should be done using Euclidean, Manhattan, Minkowski distance measurements and evaluation of categorical variables should be done using Hamming distance, which measures the number of instances of corresponding symbol or category. The K value used in the algorithm is a small positive, usually an odd number. The simplest way to select a suitable K value is to iteratively run the algorithm on different K values and select the one with the highest performance [48].

Determining an optimum K value is based on few criteria. Firstly, a suitable value for K can be selected by inspecting the data itself. In many practical scenarios cross-validation of performance for each K value will be evaluated iteratively based on independent input data set and suitable K value will be selected. In general, a large K value will be more precise as it can reduce the overall noise depending of the distribution of data. However, distinction between boundaries within the feature space also needs to be considered [53]. A rule of thumb to select a maximum for K is to use $\sqrt{n}$ if nothing about a suitable value is known in advance where 'n' is equal to data items.

Distance Techniques for Continuous Variable's

Continuous Variable's	Categorical Variables
Euclidean Distance = $\sqrt{\sum_{i=1}^{k}(x_i-y_i)^2}$	Hamming Distance, $D_H = \sum_{i=0}^{K} \lvert x_i = y_i \rvert = \sum_{i=0}^{K} D_i$ $x_i = y_i \rightarrow D_i = 0$ $x_i \neq y_i \rightarrow D_i = 1$
Manhattan Distance = $\sum_{i=1}^{k}\lvert x_i-y_i \rvert$	
Minkowski Distance = $\left\{\sum_{i=1}^{k}(\lvert x_i-y_i \rvert)^q\right\}^{\frac{1}{q}}$	

x_i, y_i = Coordinates or values of data points

k = Number of cases

Table 1: Distance Techniques

Random Forest

A random forest model is an ensemble learning technique that can be used on both classification and regression tasks. During learning stage, the algorithm constructs several decision trees and produces the output class which is the most occurring class for classification and mean prediction for regression tasks. A Random Forest with few trees is quite prone to overfit to noise and once more trees are added, the tendency to overfit generally decreases [54]. Random forest models make use of random selection of features in splitting the decision trees, hence the classifier built from this model is made up of a set of tree-structured classifiers.

When constructing a model using the algorithm, random k data points from the training set are taken and a decision tree is built associated with these k data points. Next, by selecting the number of trees (ntrees) desired to be built and the earlier steps are repeated. When classifying a new data point, a prediction is made on category to which the data point belongs using earlier ‘ntrees’ and will be assigned to the winning class. This process will start by one tree and then proceed to build more trees based on the subsets of data. The random forest has a major advantage that it can be used to judge variable importance by ranking the performance of each variable. The model achieves this by estimating the predictive value of variables and then scrambling the variables to examine how much the performance of the model drops.

Support Vector Machine

Support vector machine is a supervised learning method. There are two flavors of this technique which can be used to analyze both classification and regression problems. Firstly, support vector machine (SVM) can be used during classification problems. Secondly, support vector regression (SVR) can be used during regression problems with minor differences in the concept containing of all main features which are based on maximum margin algorithm. The algorithm will construct a nonlinear function based on linear mapping into a high dimensional kernel inspired from the input feature space. The parameters will control the capacity of the system. In addition, these parameters do not depend on the dimensionality of the input feature space.

During the training process of SVR classifier, algorithm will iteratively improve the support vector function. The optimization can be controlled using a tolerance parameter (ε) to set an approximation to the SVR. If the gradient of the optimized function is less or equal to the tolerance parameter value, the training is terminated. If the tolerance value is large the training algorithm can terminate before support vector function is sufficiently optimized, and for lower tolerance value, algorithm could try to attain high optimization levels which will be computationally expensive and time consuming.

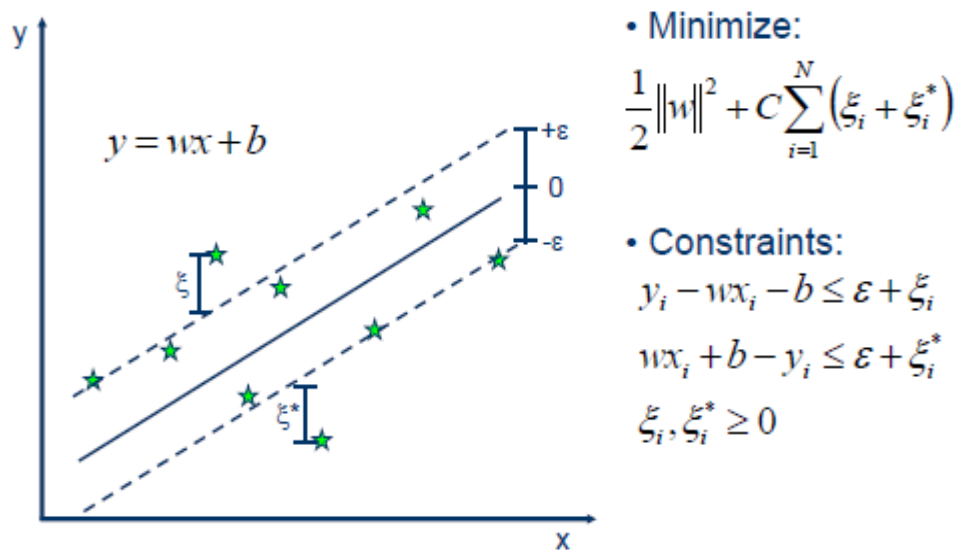


Figure 5 : One Dimensional Linear Regression with Epsilon Intensive Band [55]

In models created by support vector, models only depend on a subset of the training data. During classification task, the cost function on building the model ignores training data points lying beyond the margin. Analogously, during regression analysis, the cost function on building the model ignores training data points adjacent to the model prediction.

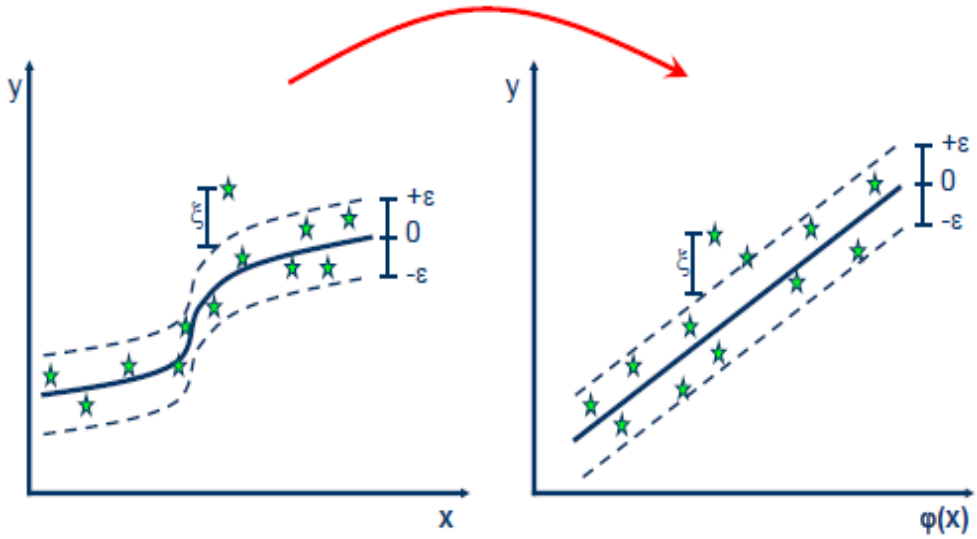


Figure 6 : Non-linear SVR Representation [55]

The estimation accuracy and performance of support vector depends on its input setting of parameters such as *C*, Ɛ and the kernel parameters. As support vector model is

a complex algorithm, the selection of optimal parameters is further complicated. In software implementations of support vector regression, these meta-parameters are given as user defined input parameters. In addition, selection of the kernel type and kernel function parameters are typically derived to reflect the distribution of the input training data and based on application domain experts [55]. Two non-linear kernel functions used during the study are presented below.

$$\text{Polynomial Kernel Function} = \boldsymbol{k(x_i, x_j)} = \boldsymbol{(x_i . x_j)^d} \quad (3)$$

$$\text{Gaussian Radial Basis Function} = k(x_i, x_j) = exp\left(\frac{||x_i - x_j||^d}{2\sigma}\right) \quad (4)$$

x_i, x_j = Coordinates or values of data points

Equation 3 : Polynomial Kernel and Gaussian Radial Basis Function

Artificial Neural Networks

Artificial Neural Network (ANN) technique and its configurations are inspired by functioning concepts of human brain, as human brain can be observed as a connecting mesh of neurons and synapses. Neurons are considered as computational units where synapses operate as the signal transferring unit. In general, every neuron is connected to several other neurons by these synapses. Even though, neurons and synaptic connections inside human brain are connected in an unorganized fashion, in ANN neurons and synapses are structured in organized way to design computationally manageable system. Sample configuration diagram of an ANN is shown in Figure 7.

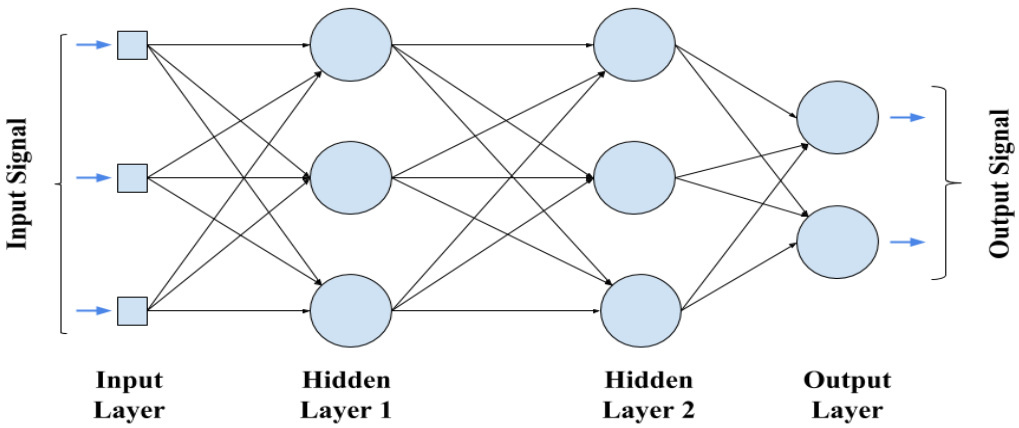


Figure 7 : Feedforward Neural Network

As the presented diagram neurons are organized in layers. The structure of neural network consists of one input layer followed by one or more hidden layers and finally an output layer. The simplest network would consist of two layers and once the network become more complex number of hidden layers will be increased to two or three (more are not necessary). The network in Figure 7 has four layers which consist of two hidden layers.

When connecting input nodes or neurons of a neural network, they typical way is to connect all nodes of the previous layer to the next layer where each connection is assigned a weight. These types of networks are known as fully connected network and Figure 7 demonstrate such network. A neuron or node computes the sum of the outputs from neurons in the previous layer multiplied by the weights assigned by the connections, and then passes it to an activation function. Activation functions enable ANNs to learn non-linear functions. There are different activation functions, e.g. sigmoid function. The effect of a sigmoid function is demonstrated in Figure 8 where activation function outputs value between 0 and 1.

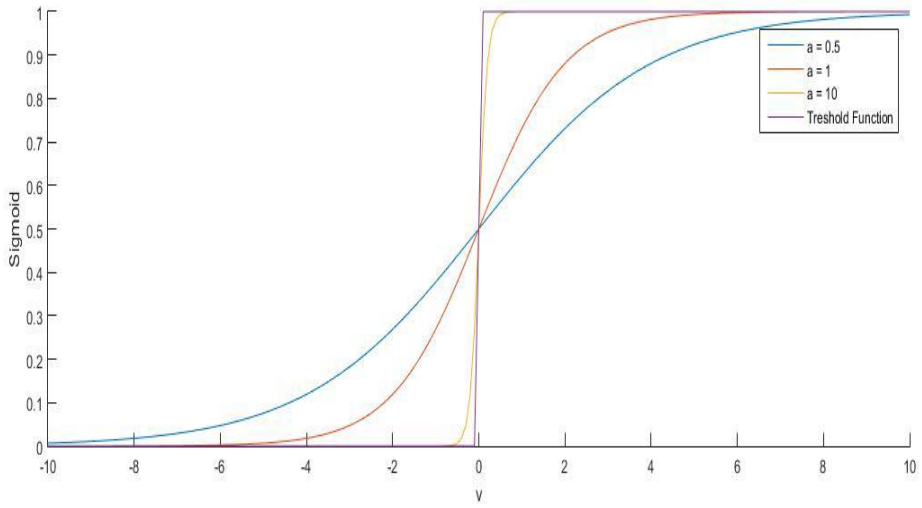


Figure 8 : The Effect of Slope Parameter in Sigmoid Function

A neural network supports both supervised learning (for networks such as one in Figure 7) and unsupervised learning techniques. Self-Organizing Maps (SOM) is the most well-known application in unsupervised neural networks. The available neural network types can be mainly categorized into feedforward and feedback networks.

Feedforward neural network is a non-recurrent network which consists of input, output and hidden layers where input signals only travel in one direction. First, inputs are assigned to input nodes and then they are passed into first processing layer of nodes. When designing a network, the number of input and output neurons is equal to the number of input and output variables in the network. The computations inside a neuron is done based on the weighted sum of its input data and this output value become the input values which fed into the preceding layer. This procedure will be followed iteratively through all layers and finally determines the output values. In practice, threshold transfer functions are used to quantify the values of output layer. In data mining problems, feed-forward networks are generally used. In addition, feed-forward networks (FFN) also include Perceptron and Radial Basis Function networks.

Feedback networks consist of loop like paths which can transmit the signals in both directions between layers allowing all possible connections among neurons. Due to these characteristics the network becomes a non-linear dynamic system with continuous changes until the network reaches a state of equilibrium. These feedback networks are generally used in optimization problems and associative memories [56].

Recurrent Neural Network (RNN)

Recurrent neural networks can process sequences of inputs as the networks use their internal state (memory) by feeding back the output signal of the neurons to the neurons in the same layer. This enables the network to exhibit dynamic temporal behaviors on a given time sequence. RNN's are generally applicable in unsegmented, connected handwriting recognition or speech recognition problems. There exist many possibilities of connecting feedback between neurons and some common ways are:

- Self-feedback: Along with the next input data sample, the output signal is fed back into the same neuron.
- No self-feedback: Along with the next input data sample, the output signal is fed back to all other neurons of the same layer except the neuron itself. This case is illustrated in Figure 9.
- Full feedback: Along with the next input data sample, the output signal is fed back to all other neurons of the same layer.

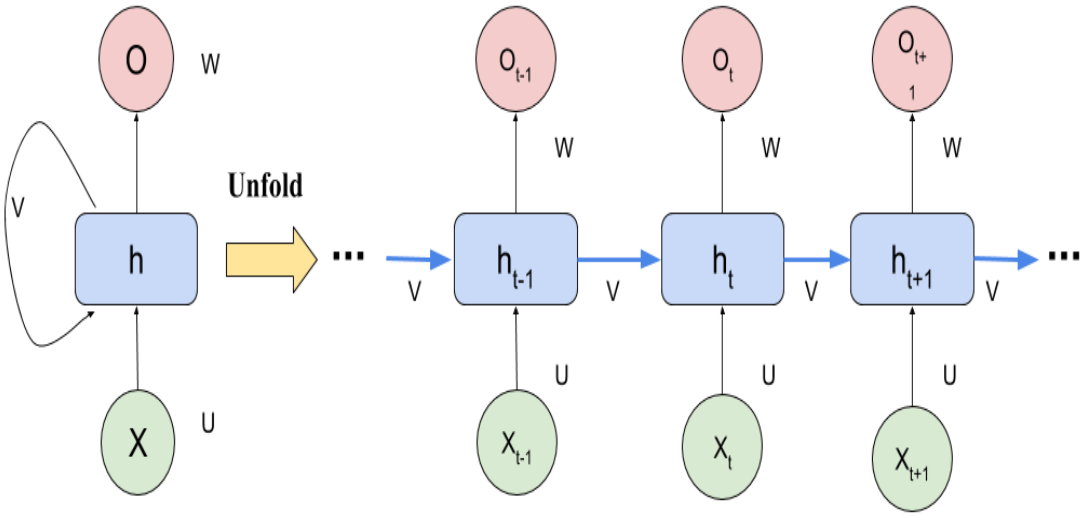


Figure 9 : One Unit Recurrent Neural Network (RNN)

5. Methodology

In this chapter, the methodology used during the study is presented. The main objective of the study was to create descriptive performance models of a software system to understand its behavior based on business usage and requirements. In theory, not only 'Analytical Performance Modelling' concepts can be used to model system operation, but also to performance testing as a faster and economical option. Once we have the required understanding about the hardware utilization based on our models this can further be used on evaluating design options and system sizing [57].

As discussed earlier, currently system dimensioning and performance is being predicted mainly based on an expert's knowledge and it would require manual work and methods can be biased and many practicalities were reported which encouraged Nokia to research more data intensive approaches. Even though the current dimensioning tool supports complex network design, it not only requires continuous maintenance but also testing to adopt changes which makes the process tedious. Further, over or under estimations in system capacity can create business impact not only on revenue but also on customer loyalty. As the ultimate result of the study, in addition to system level behavioral knowledge, stakeholders can estimate system scalability based on workloads.

Based on the market research and domain knowledge by the experts, Nokia expects that in the future customer environments can be substantially different from one another. With the advancements in cloud computing systems can evolve to fine granted tailor-made customer environments (e.g. microservices) which are more economical for their business needs. Developing accurate performance models can contribute on precise dimensioning needs where customers can efficiently use available system resources in their business which customers will definitely appreciate. It is expected to iteratively improve system understanding by continuous studies that can ultimately result new dimensioning technique which can overcome limitations in current dimensioning solutions and perform well with future business needs. This study is only one iteration for that process and the study is mainly focused on evaluating the goodness of different data mining and machine learning techniques on performance modelling of Nokia's network management system. Further, during modelling the system performance

unitization in computer nodes many univariable and multivariable-output techniques are evaluated.

As original metric data files contain substantial number of attributes depending on the scope of our study, a subset of input output features needs to be chosen. The initial set of required predictor variables were selected with the assistance of software domain experts. This further reduced using attributes subset selection algorithms to select the best set of predictor attributes for the modelling purpose. Response variables attributes representing system level utilization were selected including CPU utilization, memory consumption, disk I/O operation averages and Network I/O operation averages as per the defined scope.

The research was intended to evaluate the goodness of modelling system performance using machine learning. To ensure the set of predictor algorithms was chosen and evaluated the goodness of each method. In addition, the results are presented as univariable and multivariable-output models representing system level resource utilizations. By univariable-output model's single resource utilization metric will be represented as its output against the software measured predictor variables. In multivariate models, each resource utilization metric will be represented as a multi-output model against its predictor variables.

In addition, for analytical performance modelling goodness of result models is evaluated first using k-fold cross validation method and then against available customer datasets. Testing with customer datasets were mainly done to test the generality of the created models and applicability on cross environments.

As the requirements of the study have been defined based on the business requirement of the target company, the data definition and relevance of an initial feature list are clearly defined and during the study this initial list is further processed based on feature selection algorithms. Furthermore, if there is any inconsistency in the final models, it is also expected to investigate possible new feature areas to be included in future studies. The collected raw data files consist of different system utilization measurements and software related measurements on software systems IaaS, PaaS and SaaS layers. As there are many more attributes available compared to ones interested in the study depending on the scope of this thesis, subsets of predictor and response

variables were defined. Data was mainly collected from a set of software counters available in the target software system.

In the study, the response variables consist of a selected set of resource utilization metrics and predictor variables include a selected set of software related performance measurements that represent incoming workload (even rates, file rates, counter rates) in the system. Further, resource utilization metrics are measured in more granular basis (5 min) intervals compared to management software performance measurements (1 hour) by design as this requires accessing a database system and to avoid any performance decrease of normal operation of the software.

In addition, in the scope of this study we are only analyzing performance management computer nodes as the initial subset during system modeling. Depending on the metric type they could have different collection intervals; as an example, system level metrics are collected in 1-minute intervals and software related metrics are collected in 1-hour intervals. Due to this reason data frame needs to be generalized before analyzing them. Data sets are sampled depending on the environment (test labs, customer environments) and then depending on the computer node assigned with specific service (job) during processing. Further as defined in Section 3.13 the datasets could have different variations based on the operating procedure of the environment mainly with simulated data and customer data.

5.1. Workload Characterization and Load Modelling

During workload testing, it’s essential that workloads are repeatable and easily reproducible to simulate multiple alternative scenarios with identical settings. Even though it is necessary and important to study customer environments with real data they are not repeatable. This process is known as workload characterization and it’s necessary to observe the key characteristics when developing repeatable workload models. Once the workload models are defined, their effect based on its characteristic features can be defined and the system can be studied in a controlled manner by considering parameters of the model.

Performance Data for Analysis

Considering the network management system, performance data can be categorized to:

- **Actual workload data -** Performance data collected on customer environments containing real operating data
- **Simulated workload data -** Performance data which are synthetic and generated under controlled conditions in lab factory environments.

As in most customer environments will be run on pre-defined load boundaries there can be less variance in the data except sudden peaks in a managed network due to some failure condition or high demand situation. Further access to customer environment data is also very limited due to accessibility. During performance testing it is essential to map the collected data related to a workload in terms of business process, which then can be defined as the service demand of the system. In production systems, the possibility of controlling the environment is minimal or restricted [5].

If in capacity analysis process and modelling data consist of high variance, this will be useful since it will help to understand system boundaries. Due to above factors, it is expected to collect data by simulating input data under controlled environment conditions. As it is flexible to variate incoming data rates to the system with well-defined simulated loads, the system can be analyzed iteratively to understand overall behavior in detailed manner based on input features. In addition, to collect input data for performance models, load tests can help to evaluate both performance and scalability aspects of the system as well [5]. These models are expected to be used in the capacity analysis of the software system.

Before simulating the data, system performance architects will define the bounds to be tested based on maximum expected throughput based on PM counter rate and FM event rates from the software system and the input load will be variated according to this boundary condition.

Workload Description - Training Data

During this study, fault management and performance management functionalities were mainly considered when workloads for three categories were defined based on the requirement:

- Fault Management workloads
- Performance Management workloads
- Fault and Performance Management workloads

Each workload plan consists of combinations of different load scenarios related to different network element types. The intention by variating loads related to different network element types is to understand generality of the process and the test executed for several hours in each case where minimum duration is 2 hours. The variation in resource utilization is small corresponding to a single load profile due to a constant predetermined load and measurements during this time frame appeared to be as a cluster of data points. In the test environment, settings of predictor variables are determined by the system tester or architect who defines the predetermined characteristics of a load profile. In reality, predictor variables naturally correlate with the response variables. Finally, once the workloads are defined, a test was executed for one lab environment using a set of test simulators to generate the input data. When test rounds corresponding to different load profiles are run, we could observe small data clusters aggregated along the test data frame. This study is designed to simulate as many test rounds as possible within the time frame to collect a comprehensive training data set during a 30-day period between December and January.

Workload Type	Number of different load profiles
Fault Management	14
Performance Management	16
Fault and Performance Management	55

Table 2 : Simulated Load Profile Summary

As the number of network elements and associated system resources are fixed on a customer environment, the response and predictor variable measurements has less

variation with time. Therefore, single test round on workload test acts similar to a data from a single customer due to this reason. Further, Nokia as the sponsoring company has given the access to data and its environment details for the study purpose, but any business or confidential information will not be presented in the report.

5.2. Data Preparation

The first step was identifying the necessary data and accessing it. The main source of the performance data is the performance monitors installed in each computer node of the network management system. These collected data will be in raw format and all metric measurements corresponding to one resource (virtual machine) will be stored in a single file which will roll over daily. Each record will mainly consist of a timestamp, metric name, the measurement and a hash value per record.

For each virtual machine (VM), the records that contain the necessary data were joined to a single data frame, which enabled to create a dataset for each virtual machine. Even though the data set of each virtual machine is different, the number and type of attributes are the same for all the data sets. The number of attributes extracted was around 30. The names of predictor variables are not listed in this report based on confidentiality requirements by the company sponsoring the study. The dataset for each VM was exported to a comma separated value (CSV) file. Finally, the CSV file was imported to RStudio to analyze them.

Preprocessing

As data collected from system monitoring framework is not only in raw data format but also contains lots of unrelated metrics to the scope of this study, a pre-processing step needs to be followed before analyzing the data. As per the design of the monitoring framework, raw data files in each virtual machine collect metric information related to services run on that node. Due to this reason before analyzing correct raw datafiles from required service nodes need to be processed. As predictor variables are measured in hourly intervals, resource utilization metrics (response variables) are also averaged to hourly intervals before analysis. This was one reason to simulate constant hourly input loads when collecting performance data to be able to map the actions together and to find correlations between predictor and response features. Then the data

frames corresponding to predictor and response variables can be merged to construct the final data frame. Finally, the feature attributes were normalized as the final set of preprocessing. Even though the idea behind the pre-processing procedure is quite simple, in practice it requires lot of time consuming effort. Therefore, once the pre-processing steps and the requirements are defined, the procedure can be automated.

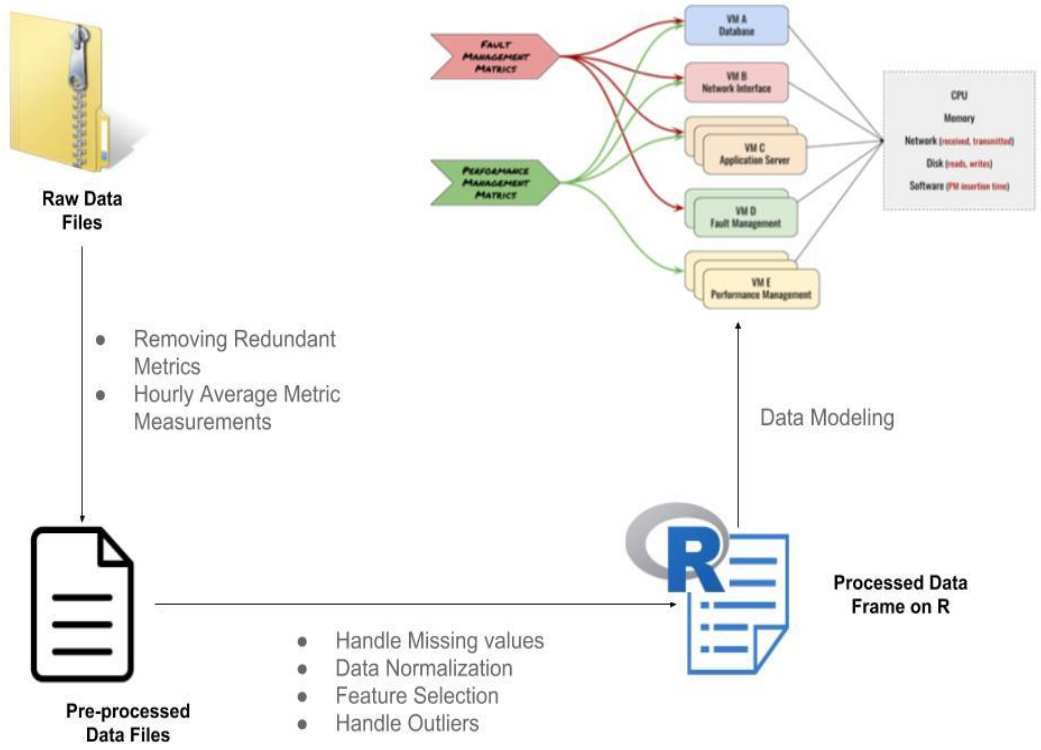


Figure 10 : Data Preparation Steps

Index	CPU MHz Average	Memory Consumed Average	Disk Read Average	Disk Write Average	Network Received Average	Network Transmitted Average	PM Insertion Time	Software Metric 1	Software Metric 2	Software Metric 3	Software Metric 4	Software Metric 5	Software Metric 6	Software Metric 7
1	2424.36	12257843	0.00	115.54	2116.63	1716.09	166	2152712	90844	36	3.81	85938579	944	41.39
2	2543.08	12257784	0.00	114.50	2257.83	1729.75	173	2664021	97675	39	3.58	81949077	851	40.55
3	2661.50	12257271	0.00	115.91	2310.33	1747.00	184	2708758	96715	34	3.89	87624156	905	41.35
4	2468.08	12258015	0.00	113.83	2194.58	1708.83	138	2925857	97402	34	3.93	70421815	712	40.88
5	2803.58	12258683	0.25	174.75	2311.83	1798.33	275	3333732	119945	42	3.90	78443863	659	17.40
6	2857.66	12257130	0.00	118.16	2534.83	1801.66	220	3190747	98198	35	4.19	100751303	1016	45.82
7	2561.08	12259054	0.08	114.08	2266.75	1748.00	146	2973978	97492	31	4.03	72923812	748	29.27
8	2358.00	12260766	0.00	111.91	2162.25	1688.66	132	3382847	99634	34	3.82	56891267	571	26.65
…	…		…	…	…	…	…	…	…	…	…	…	…	…
…	3175.25	12028907	0.41	123.08	2676.33	1846.41	285	2726081	93318	33	4.55	126446279	1355	41.02

Table 3 : Sample section of data consisting system level metrics (response variables) and software metrics (predictor variables)

Handling Missing Values

The raw data set sometimes could have missing measurements due to issues in data collection framework or service unavailability. These data rows were eliminated due to unavailability of predictor variables. As these missing measurements were recorded only in few occurrences during the entire test period, the effect to the dataset by these eliminations are assumed to be insignificant. The other option was to replace missing values using average values or most frequently occurring values. Replacing missing values using averaged values will result in a realistic way as data were simulated with predefined constant input loads which run a few hours based on the test plan.

In addition to this there can be cases where only some features are related to network element types within missing measurements. This scenario occurs when the given network related load is missing the simulated load and in this case, we would consider attribute values as zero for the given measurement interval.

Attribute Reduction

Limiting the number of predictor variables is necessary. As original data extracted contains many attributes related to different network element types and input load attributes, it is necessary to reduce the number of features to those attributes that are relevant for modelling purpose. Initially a pre-study was done considering all the predictor variables and the results was discussed with the domain experts. Based on this discussion it was suggested to define few composite features by aggregating metrics for similar network element types to construct models with more generalized features. In addition to this, some unrelated features were also ruled out based on domain knowledge. Finally, to select the best feature subset forward selection and backward elimination was evaluated on the selected feature set.

Performance Data for Modelling

During the initial modelling of the performance variables a few base predictor algorithms were selected as discussed earlier. As response variables, system level measurements available by VMware cloud framework were considered due to

simplicity and availability of measurements. During the modeling process each individual response variable (system level metrics) will be modeled against corresponding predictor variables (software level metrics). As the result for a single virtual machine, multiple models will be available representing each system level resource in descriptive way.

System Area	Metric name (VMware)	Description	Unit
CPU	cpu_usagemhz_average	CPU usage in megahertz during the interval	MHz
Memory	mem_consumed_average	Memory Consumed Average	KB
Disk	read_average	Average number of kilobytes read from the disk during the interval	KB/s
	write_average	Average number of kilobytes written to disk during the interval	KB/s
Network I/O	net_received_average	Average rate at which data was received during the interval	KB/s
	net_transmitted_average	Average rate at which data was transmitted during the interval	KB/s
Software Performance Management	PM insertion_time	Insertion Time Per Hour total	seconds

Table 4 : Resource Utilization Metrics List for Modeling (Response Variables) [47]

When constructing performance models, it is important to consider about different aspects such as physical hardware, software architecture, software system, interconnections and workload model. These models can be used to analyze the current and future system performance along with changeable workload and architecture changes [5]. When determining the response variables, system areas which can represent all aspects of the computer system were considered as listed in the above table. These models can be used during the capacity analysis process to determine how the system will operate under different load conditions.

This research mainly focuses on performance analysis on Network Management Software system used in the study using machine learning methods. As defined in

Section 4 several univariable and multivariable-output techniques were evaluated to model system performance and reliability of each method was evaluated as supervised learning problem. These algorithms consist of a set of base predictors popular in data science and then to evaluate with some more advanced algorithms and comparison of performance on each method. All these methods claimed to be reliable options in supervised learning problems with the support of many practical applications in the literature. To model single performance utilization metric, below univariable-output algorithms were evaluated:

- Multiple Linear Regression
- Multivariate Adaptive Regression Splines (MARS)
- K-nearest Neighbor (K-NN)
- Random Forest
- Support Vector Regression

In addition to represent all performance utilization metrics using a single performance model Feedforward Neural Networks (FNN) were evaluated. Further validity of the models can be evaluated using separate datasets of customer environments.

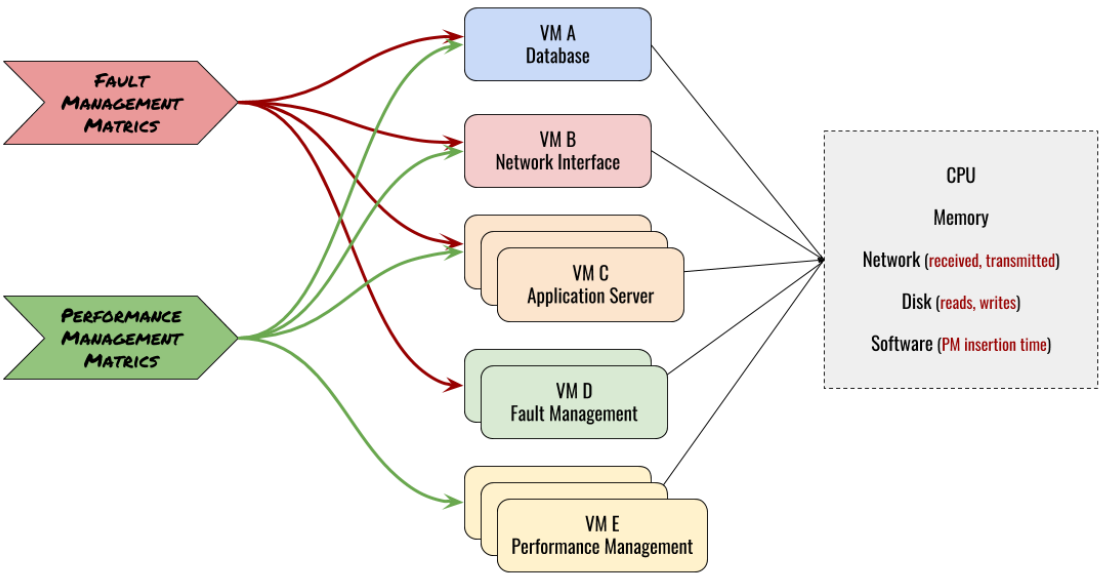


Figure 11: Modeling Approach of the Study

As per the defined scope of the study, it is intended to study only a subset of the computation nodes in the Network Management Software system depending on the business importance (performance management nodes, fault management nodes, database node). These nodes are selected based on business criticality as modelling all the nodes will be infeasible during the study period. As a limitation in the current performance data sets is the software system related measurements (predictor variables) that are only available in hourly intervals even though resource utilization measurements (response variables) are available in more granular way (1 min intervals). As to system analytical performance modelling point of view this might overfit the models as when averaging the metrics for hourly intervals certain properties of the data will be lost. By having more fine-grained data intervals more sensitive modelling could be possible.

5.3. Model Cross Validation

Once the performance models are created the next important stage is to validate the goodness of the models. Overfitting of models related to its training data is discussed by [48], [58] when using the same data for both training and subset selection. In his paper Miller states overfitting is a common problem in every model building process [42]. Overfitting also could happen when model building process uses the same dataset for selecting predictor variables and estimating regression coefficients as an example in a regression model. As a result of overfitted models could explain the current data more accurately but less performance on other datasets as for a given customer based on their integrated network elements, work load properties can be different from test data.

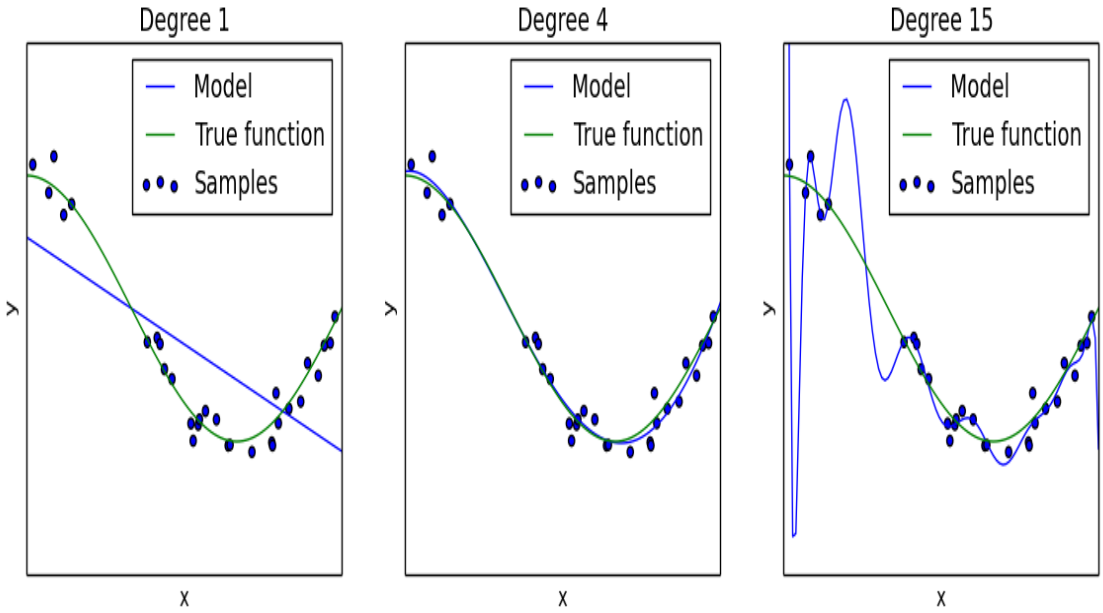


Figure 12 : Diagram on Underfit vs Overfit [59]

Sample cases of overfitting and underfitting problem is illustrated in Figure 12 and usage of linear regression with polynomial features to approximate the fit of nonlinear functions. The plot in green color represent the original function which needs to be approximated by the model equation and sample data points are displayed in dots. The models have polynomial features of different degrees. The first plot presents 'underfitting' scenario where approximation using linear function (polynomial with degree 1) which is not sufficient to fit the training samples. The polynomial function with degree 4 in the second plot approximates the true function almost perfectly. However, once the degree of the model increases it will overfit the training data and learns the noise of the training data as well.

To overcome this issue cross validation approaches can be applied by distributing the training set into multiple randomly selected subsets which follow the distribution of the data and select separate sample sets for training and validation. In holdout validation method a dataset is distributed into two separate training and validation samples, usually 70% for training and 30% for validation set. Even though this method is computationally simple, results could be more biased to data points in the training data set. More advance version k-fold cross validation method can be used where the dataset is distributed into k subsets and training and evaluation is repeated k times. Finally, statistics are calculated using average score function [60]. This method will address the

partitioning issue but require more computation processing. To further improve the results few repetitions of k-fold cross validation method can be used which is supported by many machine learning library implementations.

In addition to cross validation, during the study the created models were validated using external environment datasets to check generality of the models against customer environment data. By validation using multiple datasets the intention was to compare validation results in different software environments.

5.4. Data Mining Tools Selection

When considering software tools to be used several factors need to be considered including availability of different algorithms (machine learning libraries), ease of use and cost. At present many developers and companies are interested in available opensource software solutions over vendor specific software. One advantage over choosing opensource tool is flexibility to use and high availability of software libraries to be used for data analysis. Two famous such technologies are Python programing framework and R software which are rich in various data analysis tools. Few other available commercial data mining tools are RapidMiner, MATLAB, SPSS, and SAS. Based on the above selection criteria, RStudio was chosen as it has a rich user interface and simplicity to use. Different licenses and prices of RStudio editions are available and the open source edition of the software was chosen as it is freely available with required tooling support.

The RStudio integrated development environment (IDE) provides comprehensive facilities to develop required data mining scripts and to execute them. In addition, R software is rich with many opensource libraries which already implemented most machine learning algorithms and techniques. Furthermore, there are plenty of documentations and samples about using the tools and available libraries which makes it popular among data scientists,

5.5. Model Selection Criteria

R-Squared (R^2)

R^2 indicates the percentage of the response variable variation that is described by a linear model and also known as the coefficient of determination. In multiple regression, this is known as the coefficient of multiple determination. This statistical measure evaluates the closeness of the data to the fitted regression line.

R^2 = (Explained variation / Total variation) x 100 %

Equation 4: Definition of R^2

The value of R^2 is presented as a percentage value between 0 and 100%. R^2 value close to 0% indicates that the created model does not explain the variability of the response data around its mean value. On the other hand, value close to 100% indicates that the created model well explains the variability of the response data around its mean. In addition, when comparing different data models, higher R^2 would be preferred as they better fit the data.

Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(\widetilde{y_t} - y_t)^2}{n}}$$

$\widetilde{y_t}$ - Predicted value

y_t - Actual value of response variable

n - Number of cases

Equation 5: Definition of RMSE

This measurement criterion is frequently used in any modeling processes due to simplicity. The measurement represented by root mean squared error is the sample standard deviation of the differences between predicted values by a model and the actual data observed. If the calculation is based over data sample which is used for estimation, the difference between predicted and observed data are called residuals. In addition, if the calculations are computed using out-of-sample, they are called as prediction errors.

By aggregating the magnitudes of the RMSE's, a single measurement which represents the prediction power can be derived. Further, RMSE is a scale-dependent measurement and can only be used to compare forecasting errors of different data models corresponding to a given data set, but not between multiple datasets.

Mean Accuracy Percentage Error (MAPE)

MAPE measures the prediction accuracy of a model and presents it as a percentage value. This approach is commonly used in trend estimation in statistics. Based on the definition of this measurement, the measured absolute values are aggregated and divided by the number of data points. Finally, to make it a percentage error, the result is multiplied by 100. Although the concept is simple and convincing, this technique has some weaknesses in practice [62]:

- MAPE cannot be calculated if response variable has zero values
- For high forecast values the percentage error can exceed 100% which sometimes confuses the results.
- When comparing the accuracy of prediction models, the method tends to select a method whose forecasts are too low

$$MAPE = \frac{100}{n} \sum_{t=0}^{n} \left| \frac{A_t - F_t}{A_t} \right|$$

A_t- Actual value

F_t- Predicted value

n - Number of cases

Equation 6: Definition of MAPE

To analyze the output results Table 5 can be used to assist the evaluation based on estimated values.

Measurement	Description	Good	Average	Poor
R^2	The percentage of explained variance in data by the model.	≥ 0.8	≥ 0.6	< 0.6
Residual Mean Standard Error	An estimator for the standard deviation of the model, thus representing the variance left unexplained. Only comparable between the models explaining the same response variable.	The smallest		≥
Mean Accuracy Percentage Error	Prediction accuracy of the forecasting method expressed as a percentage	The smallest		≥

Table 5 : Model Evaluation Criteria Using Initial Estimates

6. Case Study Findings and Discussion

This study was conducted empirically on network management system software based on machine learning applications. During this task our focus is to improve the study compared to earlier feasibility studies and perform more concrete performance analysis on the software system. This section presents the findings of the research study based on the evaluated performance models. As discussed in the earlier chapter after pre-processing the data multiple univariable and multivariable-output algorithms were used to evaluate the performance by goodness of the resulted models on describing the data. The result is presented as a set of resource models based on the response variables against each individual machine learning algorithm. Table 6 shows the summary of the model results based on evaluated criteria. Even though this study involved modeling selected set of computer nodes in the network management system, for simplicity results are only presented on 'Performance Management' computer cluster nodes. When validating created models, a few approaches were evaluated based on the suggestions by domain experts at Nokia and stated below is a list of these different approaches:

- cross validation by splitting the training set
- separate validation dataset
- 3 customer environment datasets

The result graphs corresponding to validation dataset are presented in Appendix section of the report based on each response variable.

	CPU MHz Average		Memory Consumed Average		Disk I/O Read Average		Disk I/O Write Average		Network I/O Received Average		Network I/O Transmitted Average		PM Insertion Time	
	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2
Multiple Linear Regression	165.15	0.9014	651907.8	0.1840	5.5270	0.0036	86.266	0.6755	249.44	0.7473	256.38	0.7990	153.06	0.8292
Multivariate Adaptive Regression Spline	131.56	0.9362	541079.3	0.4301	6.0647	0.0070	77.935	0.7408	138.59	0.9217	180.47	0.8985	131.84	0.8759
K-Nearest Neighbors	128.37	0.9401	541122.2	0.4381	6.1446	0.0022	75.803	0.7573	112.58	0.9483	178.91	0.9022	126.61	0.8846
Random Forest	114.90	0.9520	518124.5	0.4874	6.0525	0.0011	74.279	0.7654	100.84	0.9581	171.75	0.9089	120.92	0.8936
SVR Polynomial Kernel	129.30	0.9396	674781.0	0.2649	5.6386	0.0006	77.877	0.7457	137.91	0.9234	179.80	0.8990	138.93	0.8610
SVR Radial Basis Function Kernel	122.96	0.9458	650996.7	0.2950	5.1167	0.0014	73.580	0.7667	125.72	0.9368	192.78	0.8883	117.49	0.9036
FFN (6)	223.76	N/A	982117.9	N/A	6.4371	N/A	135.07	N/A	192.17	N/A	357.27	N/A	940.83	N/A
FFN (10)	202.45	N/A	876971.7	N/A	6.3225	N/A	129.65	N/A	160.34	N/A	321.33	N/A	948.71	N/A
FFN (10, 6)	189.53	N/A	814381.1	N/A	6.3202	N/A	122.45	N/A	125.89	N/A	304.81	N/A	949.58	N/A
FFN (10, 6, 6)	181.77	N/A	777504.3	N/A	6.2851	N/A	123.05	N/A	121.32	N/A	306.98	N/A	950.98	N/A

Table 6 : Model Result Summary

As opposed to the previous study conducted on network management system, this experiment was done to overcome some limitations of the earlier research. As discussed in earlier chapter there was limitations on number of evaluated methods, ability to collect software related metrics and availability of customer datasets to compare performance. To overcome the earlier limitations, total of 85 different load profiles were run against the software system to collect a good enough training dataset and several machine learning algorithms were used to construct performance models which represent different system resource utilization metrics. Further validation of models was conducted in addition using a separate test dataset to evaluate the models using three customer datasets from separate system environments.

Results of the disk I/O read average (R^2 ~ 0.7%) is not be presented in the report as the model is weak and will not explain the data properly due to less variation in the measurement values. In memory consumption there also can be less variation on measurements due to its cached and buffered components. Technically this due to Linux operating system borrowing unused memory for disk caching to improve its performance and makes the system faster and more responsive [62]. Due to this reason measured memory consumption will represent higher measurement value irrespective of its actual operational use by the software application. The model representing memory consumed average shows moderate R^2 (~ 48.7%) value compared to other models.

Conclusions made from each result set will be presented based on univariable-output models and multivariable-output models for convenience. Table 7 represents the evaluation result summary of the regression models based on their performance against test data set. The models are related to regression analysis of performance utilization metrics when evaluating the created models RMSE were used. In addition to this MAPE is also listed in the table due to simplicity to understand, even though it is a biased measurement based on measurement values.

Algorithm Name	CPU Average (MHz) RMSE (MAPE)	Memory Consumed Average (MB) RMSE (MAPE)	Disk I/O Write Average (KB/s) RMSE (MAPE)	Network I/O Received Average (KB/s) RMSE (MAPE)	Network I/O Transmitted Average (KB/s) RMSE (MAPE)	PM Insertion Time (s) RMSE (MAPE)
Linear Regression	205.51 (7.06%)	804.748 (6.11%)	106.07 (34.63%)	300.38 (16.82%)	436.85 (18.29%)	**143.77** (184.8%)
MARS	194.27 (5.92%)	696.701 (5.10%)	87.21 (27.91%)	327.75 (14.44%)	**285.60 (**14.23%**)**	168.36 (219.9%)
K-NN	251.40 (7.31%)	715.199 (4.56%)	88.85 (24.6%)	237.20 (11.02%)	442.01 (13.47%)	216.20 (27.45%)
Random Forest	281.29 (8.13%)	699.140 (4.66%)	100.04 (28.56%)	231.05 (10.90%)	431.59 (12.51%)	208.74 (26.33%)
SVR (Polynomial Kernel)	183.21 **(**5.20%**)**	894.511 (5.76%)	**66.89 (**21.22%**)**	245.58 (11.21%)	421.14 (14.54%)	166.63 **(**237.8%**)**
SVR (Radial Basis Function Kernel)	214.88 (6.32%)	956.556 (6.28%)	74.12 (23.36%)	255.09 (13.21%)	414.92 (14.06%)	197.71 (339.4%)
FFN (6)	**173.68** (4.68%)	**695.119** (4.64%)	85.25 (25.88%)	100.20 (4.26%)	496.30 (16.06%)	148.27 (159.4%)
FFN (10)	179.73 (4.54%)	777.832 (4.90%)	89.75 (25.76%)	**97.49** (4.19%)	502.12 (15.52%)	166.98 (85.65%)
FFN (10, 6)	226.76 (5.95%)	702.149 (4.27%)	73.01 (20.14%)	116.84 (4.66%)	454.50 (12.11%)	156.01 (182.1%)
FFN (10, 6, 6)	174.99 (4.56%)	741.571 (4.59%)	69.28 (19.56%)	101.02 (3.15%)	422.16 (12.71%)	179.11 (241.1%)

Table 7 : Summary of Machine Learning Methods Against Test Dataset (Method with the lowest RMSE is highlighted in bold)

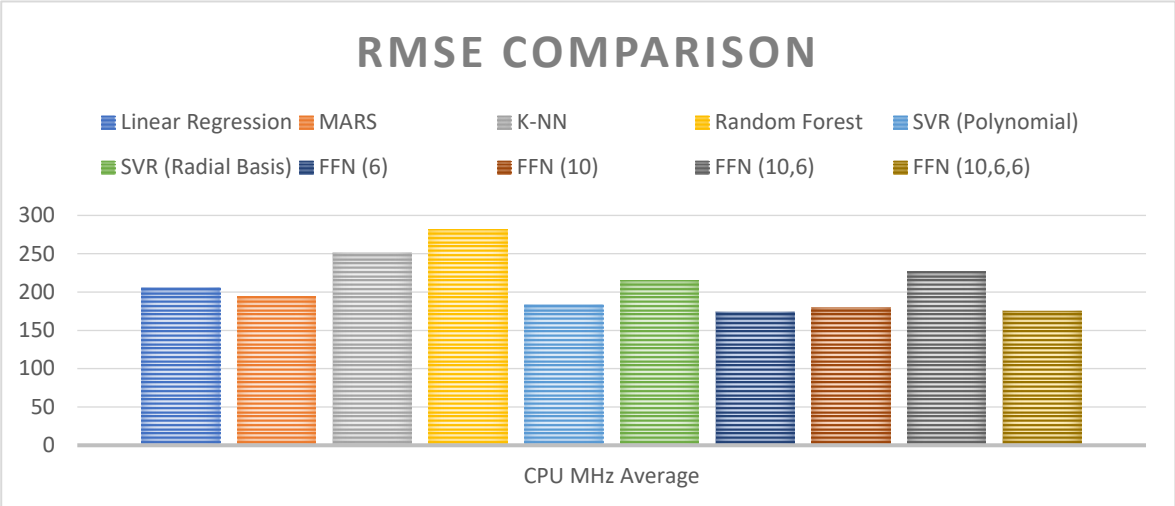


Figure 13: RMSE Comparison of CPU Average

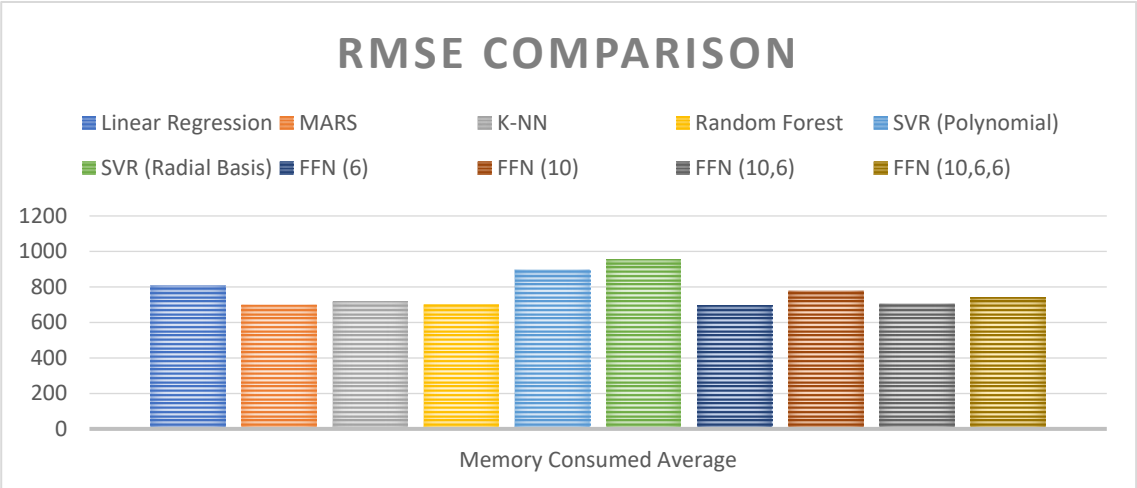


Figure 14 : RMSE Comparison of Memory Consumed Average

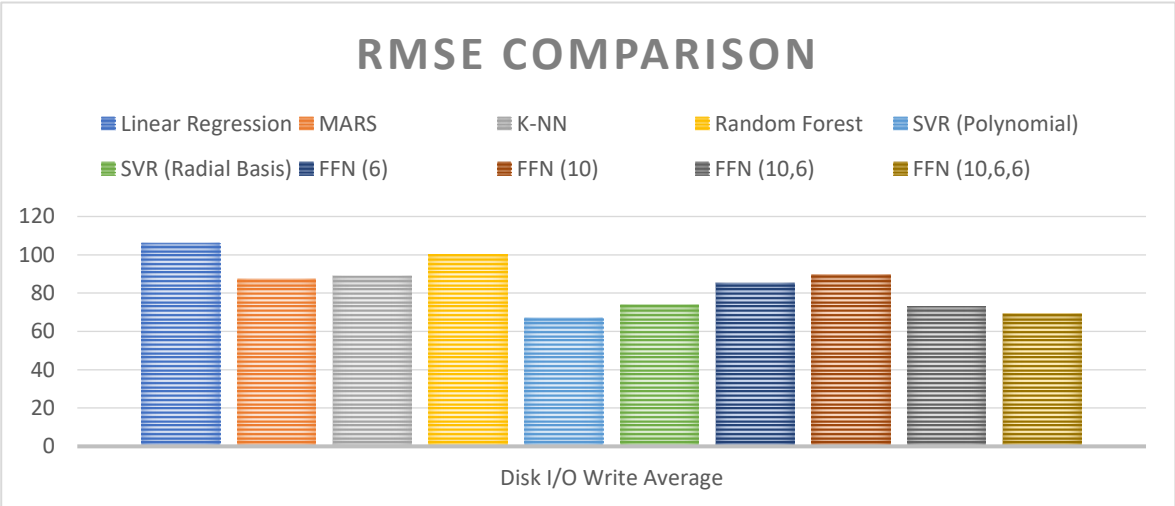


Figure 15: RMSE Comparison of Disk Write Average

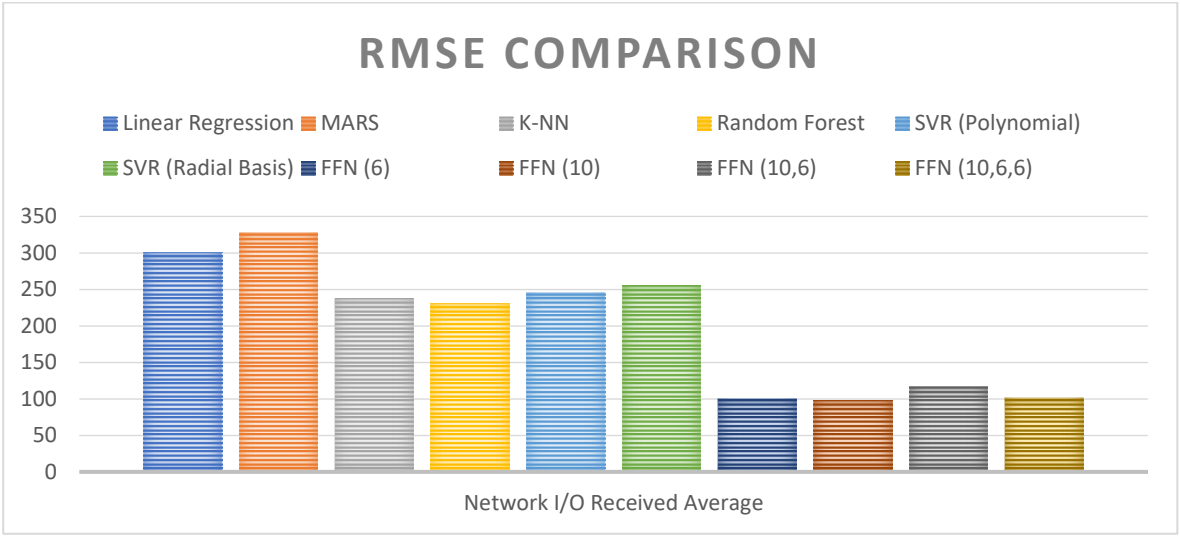


Figure 16: RMSE Comparison of Network Received Average

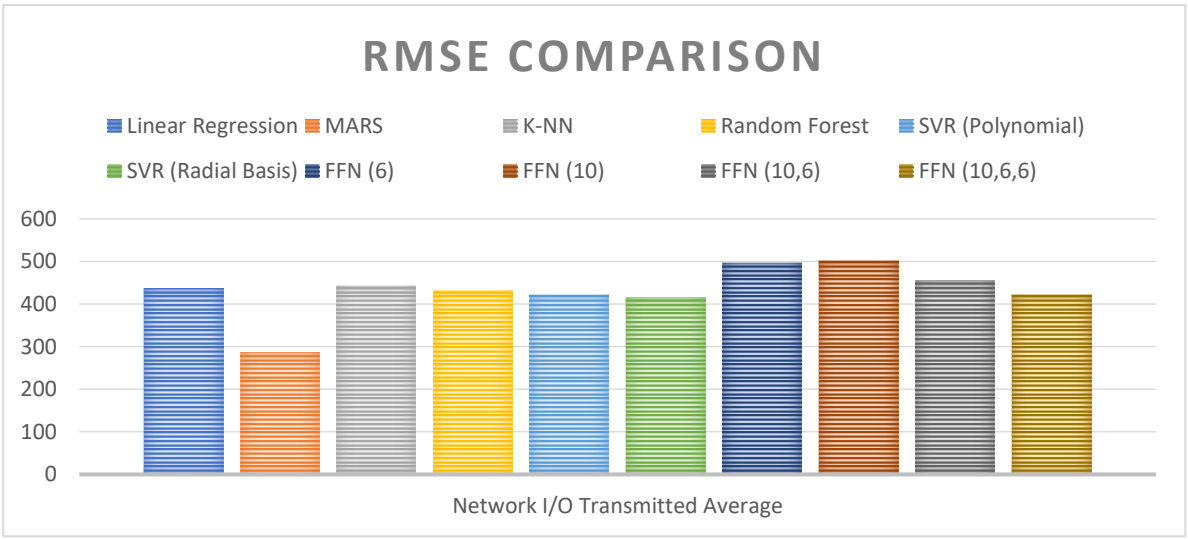


Figure 17: RMSE Comparison of Network Transmitted Average

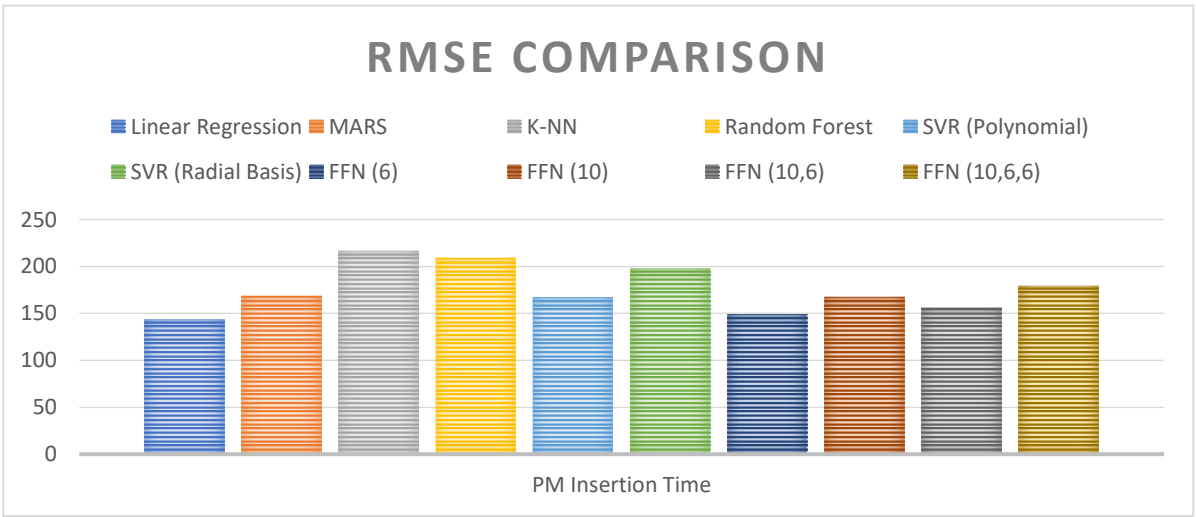


Figure 18: RMSE Comparison of PM Insertion Time

Figure 13 to 18 shows RMSE comparison among different machine learning algorithms. When evaluating performance of system along with the goodness of performance simplicity and applicability of the models is also important in real operations. Considering the results of univariable-output models, it is implicated that no single algorithm performs best for all resource models. In addition, compared to all other response variables ‘PM Insertion Time’ does not perform well with respect to percentage accuracy values and this is due to the biasness of the MAPE measurement where it causes high percentage error values with small insertion time measurements. Apart from this all models performed well based on the performance criteria’s.

On the other hand, not only feed forward neural networks perform well with low RMSE and MAPE values, since they are multivariable output methods there is advantage of having single model to represent all output metrics. Based on the results the feedforward network with 1 hidden layer with 6 neurons performs well overall as RMSE and MAPE are comparatively low on many resource utilization models. Also, it can be seen that performance of the networks slightly decreases when number of nodes increase but possibly due to limited training data points to train the network completely. Since current training dataset includes limited number of data points, there are limitations when training larger networks which could not properly learn all its weights.

7. Conclusion

The aim of the study was to assess machine learning applicability on capacity estimation of network management system software by modelling its performance utilizations. To accomplish this, following research objectives were set which laid the foundation for the study:

Objective 1: Modelling resource utilization and responsiveness of the system

Performance prediction using the created models for lab environments performed well with percentage error on CPU MHz average ~ 4.68%, Memory Consumption Average ~ 4.64%, Disk Write Average ~ 4.19% Network Received Average ~ 4.19% and Network Transmitted Average ~ 14%. Modeling on 'Disk I/O Read Average' was not considered as it cannot be modeled due to low variation in measured values and low R^2 values on models corresponding to these predictor variables. Also, the models based on artificial neural networks are well fitting with the lab data samples creating multivariable-output models with better performance values compared to univariable-output utilization models.

The overall evaluation results show that application of machine learning techniques have the potential of modeling system resource utilizations. To evaluate performance of these exploratory models, RMSE and MAPE were considered. Performance of system level resource models were evaluated across test environment and customer data. Finally, the results were evaluated based on R^2, RMSE and percentage accuracy levels against the learning method.

The comparison of results on test environment shows, that no single learning technique performed best on modeling all given metrics in univariable-output utilization models. Support Vector Machines with Polynomial Kernel seems to perform comparatively well on univariable-output models. As there wasn't any single best method to have highest performance, an ensemble method could be evaluated for further improvements. Furthermore, feedforward neural networks with multivariable output models seems to perform well compared to all other techniques with low RMSE and MAPE values.

Objective 2: Understand performance bottlenecks of the system

Firstly, we noticed some inconsistency among load balancing on PM nodes where first node in the cluster always processes more counters when increasing the system load. This can be identifiable when observing the resource utilization graphs over time. Secondly, even though the system load on customer environments are less variating again initial cluster nodes indicate more resource utilizations compared to remaining nodes under uniform input load. When comparing lab vs customer environments, response time on PM data flow (insertion time) is comparatively high on lab environments compared to customer datasets. These points highlight some performance inconsistencies in the current software system which needs insight analysis on system architecture and layout.

Objective 3: Understand limitations of current metrics used for system modelling

In the process of validating the models against performance data on customer environments, more deviations were observed on predicted results compared to actual resource utilization measurements. One reason for this being properties of the current training data set generated using test simulators are sometimes different from customer data as the test simulators are designed for capacity test of the software system. When comparing the simulated data against customer data we could see that file sizes, byte sizes in measurements and counter rates are different among the environments.

On the other hand, predictive results were deviated in customer datasets compared to the lab environment in which training samples were collected. The models perform variably on different customer datasets as properties of the input data vary depending on the environment. Furthermore, as we used the training data set to feature selection and train the models it is inevitable that the created utilization models will be overfit to the training data set. This implies the requirement of further study on finding the explanations for variation in performance of different data sets. In addition, constructing a training dataset with more feature variations could lead to better performance results to the problem.

Limitations in the current study

Network management system is a large distributed software which consists of many dataflows and over 50 virtual machine instances. Due to this complexity,

modelling each entity would require a huge effort in practice. As the initial study the PM data flow was selected depending on its importance to the business. Further studies about other system components are expected to be conducted in next iterations, which is out of the scope of this study. When defining the scope of the study it was agreed to only consider system level (IaaS) resource metrics by selecting CPU utilization, memory consumption, disk I/O operation averages and Network I/O operation averages at this stage. In addition, PM insertion time was considered as a measure of responsiveness of the dataflow.

As the training data mainly consist of simulated workloads, temporal relationships among data points cannot be studies. Due to this reason, recurrent neural networks (RNN) were not evaluated in this experiment. Further during the timeframe of the study only few sets of customer data was available to analyze exploratory analyze the data and to validate the models against them. In future it is expected to collect more datasets by collaborating with the customers and to improve performance analysis of the software in general.

Future work and recommendations

Data mining is a continuous learning process where results need to be improved iteratively depending on the research objectives. Further, in an iterative study we can recognize important facts that have been left out during the initial planning, once additional data and domain knowledge is collected. To improve the results further, more advanced data preparation steps, modeling techniques and more involvement from domain experts are required. In addition, continuous improvements can lead to results with high accuracy. In this section possible future study areas are discussed which were identified during the study:

- The results show that apart from the data collected from the lab environment, predicted performance varies across customer datasets which represent different company’s data. Since different mobile technologies are used in operator companies depending on their business, the properties of input data processed in network management system can be different in nature. This nature of system data produces dissimilar resource utilization measurements which needs further study.

- During the testing of created models against customer datasets more deviations were noticed on the predictive results compared to lab environments. Some reasons were that models are overfitting to training data where in customer datasets certain properties of data (file sizes, file rates, size per counter etc.) are different between compared to lab environment. To overcome this further generalized training dataset should be created, which aligns with customer data cases to model generalized results to improve accuracy.

- In future, with the availability of continuous customer datasets which spreads through longer time periods, additional studies to understand trends, cycles, busy hours for a given customer can be done using time series analysis and load prediction techniques.

- Based on its design network management system software is heavily dependent on underlying database operations and further analysis of its effect on system performance can be studied. Along with this study certain system responsiveness metrics can also be studied using dataflow response times and insertion times.

- This study was mainly based on system level resource measurements (IaaS layer) as defined in the scope. Next, the scope can be further extended to incorporate platform level (PaaS) and software level (SaaS) resource utilizations metrics and attributes.

- Based on the current data models, baseline performance levels can be defined for a given software version. This information can be used in future performance comparisons against different software versions to evaluate any capacity deviations against different versions.

- The results can be used by domain experts to define dimensioning properties or requirements based on system level resource usages to ensure required amount of system performance.

Overall, the current methods and the results of the study help to define the foundation for future studies. In conclusion, from the results and the findings of the

study, one could find insight and information about the network management system and about its resource usages to understand system’s capacity analysis in data intensive way. This was the main objective of the study and the solution provides the necessary information to the company.

References

[1] X. Ge *et al*, "Vehicular Communications for 5G Cooperative Small-Cell Networks," *IEEE Transactions on Vehicular Technology*, vol. 65, *No. 10*, pp. 7882-7894, 2016.

[2] J. Hamilton, "Cooperative expendable micro-slice servers (CEMS): Low cost, low power servers for internet-scale services," in *Conference on Innovative Data Systems Research*, 2009. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.479.8900&rep=rep1&type=pdf

[3] A. Greenberg *et al*, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, *No. 1*, pp. 68-73, 2008. Available: http://dl.acm.org.helios.uta.fi/citation.cfm?id=1496091.1496103.

[4] C. Richardson. “What are microservices?” [Online]. Available: http://microservices.io/index.html. [Accessed: Dec. 12, 2018].

[5] J. A. Aries *et al*, "Capacity and performance analysis of distributed enterprise systems," *Communications of the ACM*, vol. 45, *No. 6*, pp. 100-105, 2002. Available: http://dl.acm.org.helios.uta.fi/citation.cfm?id=508448.508455.

[6] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* New York, NY: John Wiley & Sons, 1990.

[7] Wei-Hua Bai *et al*, "Performance Analysis of Heterogeneous Data Centers in Cloud Computing Using a Complex Queuing Model," *Mathematical Problems in Engineering*, vol. 2015, pp. 1-15, 2015. Available: http://dx.doi.org/10.1155/2015/980945.

[8] [2] S. El Kafhali and K. Salah, "Performance analysis of multi-core VMs hosting cloud SaaS applications," *Computer Standards & Interfaces*, vol. 55, pp. 126-135, 2018. Available: https://www.sciencedirect.com/science/article/pii/S092054891730048X.

[9] Xiwei Qiu, Liang Luo and Yanping Xiang, "Performance Evaluation of a Cloud Service Considering Hierarchical Failure Recovery," *International Journal of Performability Engineering*, vol. 12, *(2)*, pp. 197, 2016. Available: https://search.proquest.com/docview/1786565910.

[10] X. Qiu *et al*, "Performability analysis of a cloud system," in *Computing and Communications Conference (IPCCC), 2015 IEEE 34th International Performance*, 2015.

[11] Y. Bai, H. Zhang and Y. Fu, "Reliability modeling and analysis of cloud service based on complex network," in 2016, Available: https://ieeexplore.ieee.org/document/7819907.

[12] Afkham Azeez, " Autoscaling Web Services on Amazon EC2", M. S. thesis, University of Moratuwa, Moratuwa, Sri Lanka. 2008.

[13] C. Chassot *et al*, "Performance analysis for an IP differentiated services network," in *Communications, 2002. ICC 2002. IEEE International Conference On*, 2002. Available: https://ieeexplore.ieee.org/document/997000.

[14] D. A. Menasce and E. Casalicchio, "A framework for resource allocation in grid computing." in *Mascots*, 2004.
Available: https://ieeexplore.ieee.org/document/1348280.

[15] S. Penmatsa and A. T. Chronopoulos, "Price-based user-optimal job allocation scheme for grid systems," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, 2006.
Available: https://ieeexplore.ieee.org/document/1639653.

[16] K. Xiong and H. Perros, "SLA-based resource allocation in cluster computing systems," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium On*, 2008.
Available: https://ieeexplore.ieee.org/document/4536272.

[17] S. Kundu *et al*, "Application performance modeling in a virtualized environment," in January 2010. Available: https://ieeexplore.ieee.org/document/5463058.

[18] P. A. Dinda and D. R. O'Hallaron, "Host load prediction using linear models," *Cluster Computing*, vol. 3, *No. 4*, pp. 265-280, 2000. Available: https://link-springer-com.helios.uta.fi/article/10.1023/A:1019048724544.

[19] C. Stewart and K. Shen, "Performance modeling and system management for multi-component online services," in 2005,
Available: http://dl.acm.org/citation.cfm?id=1251203.1251209.

[20] Christopher Stewart Terence Kelly Alex Zhang Kai Shen and U R HP L, "A Dollar from 15 Cents: Cross-Platform Management for Internet Services".

[21] T. Wood *et al*, "Profiling and modeling resource usage of virtualized applications," in *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, pp. 366-387, 2008.

[22] E. Caron, F. Desprez and A. Muresan, "Pattern Matching Based Forecast of Non-Periodic Repetitive Behavior for Cloud Clients," *J Grid Computing*, vol. 9, *No. 1*, pp. 49-64, 2011. Available: https://hal.inria.fr/hal-01426826.

[23] H. Kim, W. Kim and Y. Kim, "A pattern-based prediction model for dynamic resource provisioning in cloud environment," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 5, *No. 10*, pp. 1712-1732, 2011.

[24] S. Di, D. Kondo and W. Cirne, "Host load prediction in a google compute cloud with a bayesian model," in *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference For*, 2012.
Available: http://dl.acm.org/citation.cfm?id=2389025.

[25] J. Cao *et al*, "CPU load prediction for cloud environment based on a dynamic ensemble model," *Software: Practice and Experience*, vol. 44, *No. 7*, pp. 793-804, 2014.

[26] N. Kourentzes, D. K. Barrow and S. F. Crone, "Neural network ensemble operators for time series forecasting," *Expert Systems with Applications,* vol. 41, *No. 9,* pp. 4235-4244, 2014.

[27] J. Jheng *et al*, "A novel VM workload prediction using grey forecasting model in cloud data center," in *Information Networking (ICOIN), 2014 International Conference On*, 2014. Available: https://ieeexplore.ieee.org/document/6799662.

[28] R. Wolski, N. Spring and J. Hayes, "Predicting the cpu availability of time-shared unix systems on the computational grid," in *High Performance Distributed Computing, 1999. Proceedings. the Eighth International Symposium On*, 1999. Available: https://ieeexplore.ieee.org/document/805288.

[29] K. Tuisku, "Network Management System Dimensioning with Performance Data", M. S. thesis, University of Tampere, Tampere, Finland, 2016.

[30] Computer Security Division, Information Technology Laboratory. *Cloud Computing | CSRC*. Available: https://csrc.nist.gov/projects/cloud-computing. [Accessed: Nov. 12, 2017].

[31] A. Balalaie, A. Heydarnoori and P. Jamshidi, "Migrating to cloud-native architectures using microservices: An experience report," in *European Conference on Service-Oriented and Cloud Computing*, pp. 201-215, 2015.

[32] International Organization for Standardization. *Information Processing Systems - Open Systems Interconnection - Basic Reference Model - Part 4: Management Framework* ISO/IEC 7498-4:1989.

[33] Nokia Networks, "NetAct Architecture Description", [*Unpublished intranet content*]. [Accessed: Sep. 12, 2017].

[34] Architectural and Framework Standards: The TMN/FCAPS Model (ITU-T). [Online].
Available: http://etutorials.org/Networking/network+management/Part+I+Data+Collection+and+Methodology+Standards/Chapter+3.+Accounting+and+Performance+Standards+and+Definitions/Architectural+and+Framework+Standards+The+TMN+FCAPS+Model+ITU-T/. [Accessed Oct. 13, 2017].

[35] Software Metrics. [Online].
Available: http://www.sqa.net/softwarequalitymetrics.html. [Accessed Nov. 14, 2017].

[36] Gordon Haff, "That's not a metric! Data for cloud-native success," 2017. [Online]. Available: https://www.slideshare.net/ghaff/thats-not-a-metric-data-for-cloudnative-success. [Accessed Nov. 14, 2017].

[37] S. Wu, T. J. Harris and K. B. McAuley, "The use of simplified or misspecified models: linear case," *The Canadian Journal of Chemical Engineering,* vol. 85, *No. 4,* pp. 386-398, 2007.

[38] G. Shmueli and O. R. Koppius, "Predictive analytics in information systems research," Mis Quarterly, pp. 553-572, 2011.

[39] Chid Apte, "The Role of Data Mining in Business Optimization," IBM Research 2008. [Online]. Available: https://www.siam.org/meetings/sdm11/apte.pdf [Accessed Nov. 20, 2017].

[40] G. M. Weiss, "Data mining in the real world: Experiences, challenges, and recommendations." in *DMIN*, 2009, pp. 124-130.

[41] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, *(Mar)*, pp. 1157-1182, 2003.

[42] A. Miller, *Subset Selection in Regression.* CRC Press, 2002.

[43] M. D. Assunção *et al*, "Big Data computing and clouds: Trends and future directions," *Journal of Parallel and Distributed Computing*, vol. 79, pp. 3-15, 2015.

[44] U. Fayyad, G. Piatetsky-Shapiro and P. Smyth, "From data mining to knowledge discovery in databases," *AI Magazine*, vol. 17, *No. 3*, pp. 37, 1996.

[45] Y. Ramamohan *et al*, "A study of data mining tools in knowledge discovery process," *International Journal of Soft Computing and Engineering (IJSCE) ISSN*, pp. 2231-2307, 2012.

[46] S. B. Kotsiantis, I. D. Zaharakis and P. E. Pintelas, "Machine learning: a review of classification and combining techniques," *Artificial Intelligence Review*, vol. 26, *No. 3*, pp. 159-190, 2006.

[47] Managed Object - PerformanceManager [Online]. Available: https://www.vmware.com/support/developer/converter-sdk/conv61_apireference/vim.PerformanceManager.html. [Accessed Nov. 10, 2017].

[48] D. J. Hand, "Principles of data mining," *Drug Safety*, vol. 30, *No. 7*, pp. 621-622, 2007.

[49] Y. Fujikoshi, V. V. Ulyanov and R. Shimizu, *Multivariate Statistics: High-Dimensional and Large-Sample Approximations.* John Wiley & Sons; 2011 Aug 15.

[50] A. S. Goldberger, *A Course in Econometrics.* Harvard University Press; 1991.

[51] G. S. Maddala and K. Lahiri, *Introduction to Econometrics.* New York: Macmillan; 1992.

[52] G. Guo *et al*, "KNN model-based approach in classification," in *OTM Confederated International Conferences" on the Move to Meaningful Internet Systems"*, pp. 986-996, 2003.

[53] K Nearest Neighbors - Regression. [Online]. Available: http://www.saedsayad.com/k_nearest_neighbors_reg.htm. [Accessed Nov. 20, 2017].

[54] *Random Forest*. [Online]. Available: https://en.wikipedia.org/wiki/Random_forest. [Accessed Nov. 20, 2017].

[55] Support Vector Regression. [Online]. Available: http://www.saedsayad.com/support_vector_machine_reg.htm. [Accessed Nov. 20, 2017].

[56] Artificial Neural Network. [Online].
Available: http://www.saedsayad.com/artificial_neural_network.htm. [Accessed Nov. 20, 2017].

[57] B. Wescott, *Every Computer Performance Book: How to Avoid and Solve Performance Problems on the Computers You Work With.* (1st ed.) CreateSpace Independent Publishing Platform; 2013

[58] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," *Statistics Surveys*, vol. 4, pp. 40-79, 2010.

[59] *Underfitting vs. Overfitting — scikit-learn documentation*. [Online].
Available: http://scikit-learn.org/0.15/auto_examples/plot_underfitting_overfitting.html. [Accessed Dec. 05, 2017].

[60] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," International Joint Conference on Artificial Intelligence (Vol. 14, No. 2, pp. 1137-1145, 1995.

[61] C. Tofallis, "A better measure of relative prediction accuracy for model selection and model estimation," Journal of the Operational Research Society, vol. 66, *(8)*, pp. 1352-1362, 2015.

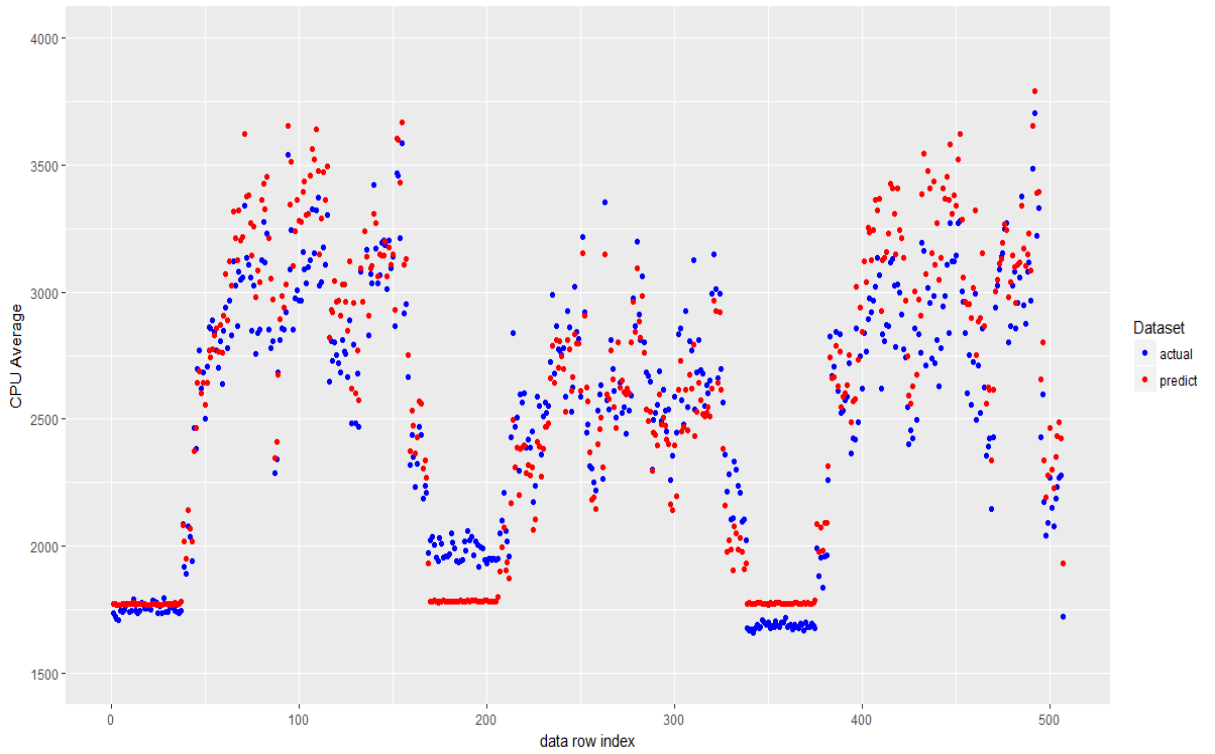
[62] “Linux Ate My Ram" [Online]. Available: https://www.linuxatemyram.com. [Accessed Jan. 08, 2018].

Appendix

Modeling Results

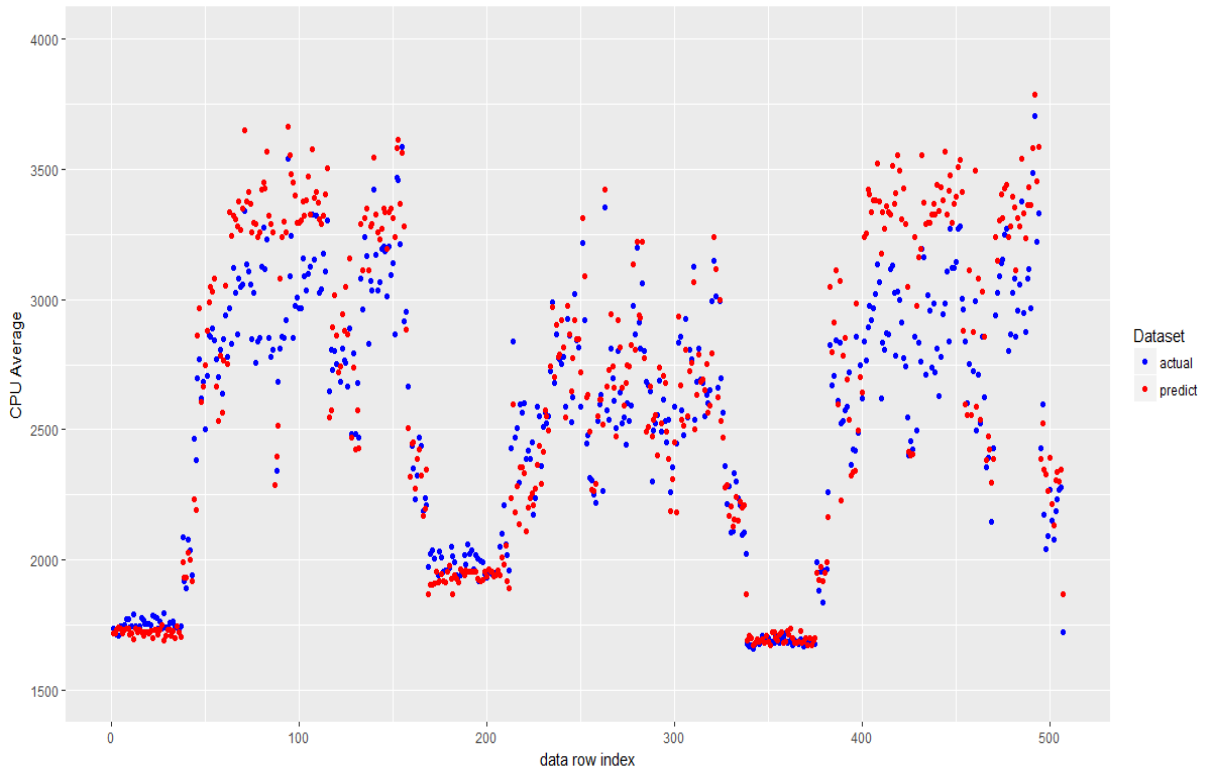
The results from Table 8 to Table 13 presents the actual and predicted results corresponding to different modeling algorithm on validation dataset. Additionally, RMSE, MAE and MAPE values are also presented for each case. In the validation result graphs, color *blue* represents actual resource usage values and color *red* represents predicted usages by the models. These results helped during the experiment to decide the suitable modeling technique against each response variable.

CPU MHz Average					
Linear Regression with Stepwise Selection					
RMSE	205.51	MAE	178.87	MAPE	7.06%



Multivariate Adaptive Regression Spline					
RMSE	194.27	MAE	194.27	MAPE	5.92%

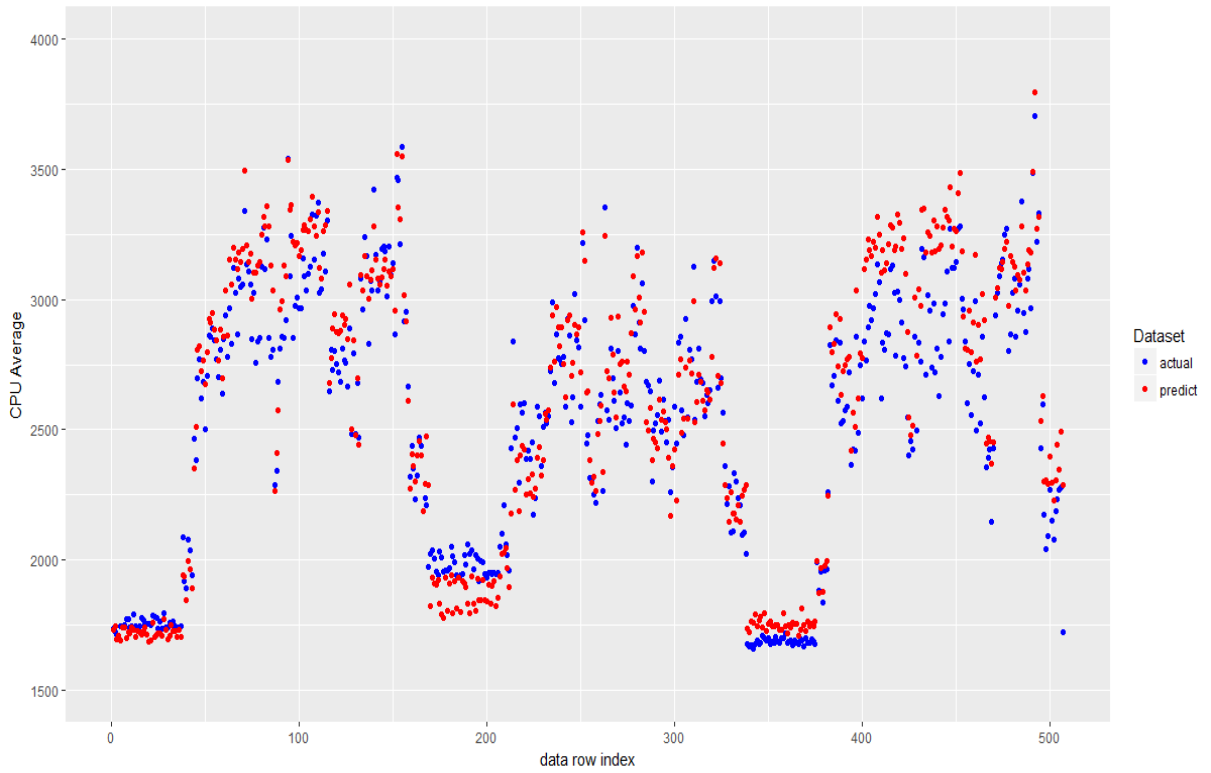
K-Nearest Neighbors					
RMSE	251.40	MAE	199.26	MAPE	7.31%



Random Forest					
RMSE	281.29	MAE	222.49	MAPE	8.13%

Support Vector Machines with Polynomial Kernel					
RMSE	183.21	MAE	131.83	MAPE	5.20%

Support Vector Machines with Radial Basis Function Kernel					
RMSE	214.88	MAE	162.48	MAPE	6.32%



Feed Forward Network - 1 Hidden Layer with (6) Nodes					
RMSE	173.68	MAE	127.62	MAPE	4.68%

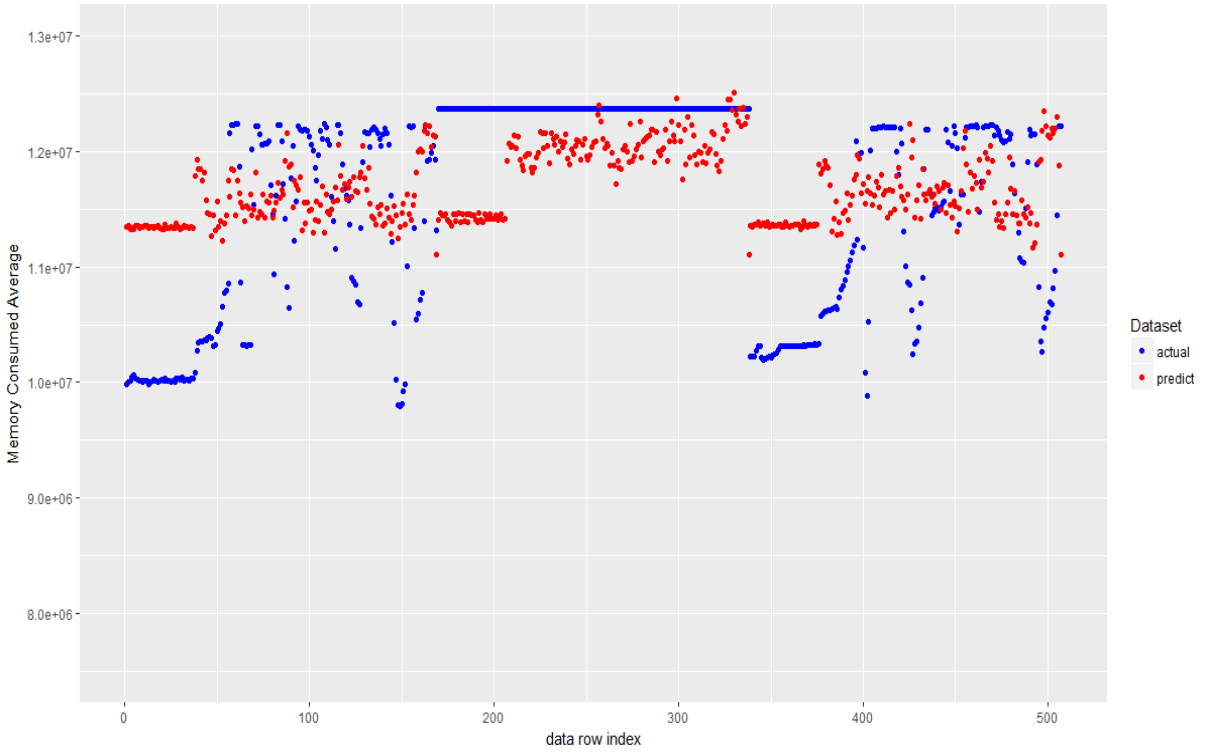
Feed Forward Network - 1 Hidden Layer with (8) Nodes					
RMSE	179.72	MAE	124.05	MAPE	4.53%

Feed Forward Network - 2 Hidden Layers with (10,6) Nodes					
RMSE	226.76	MAE	163.02	MAPE	5.95%

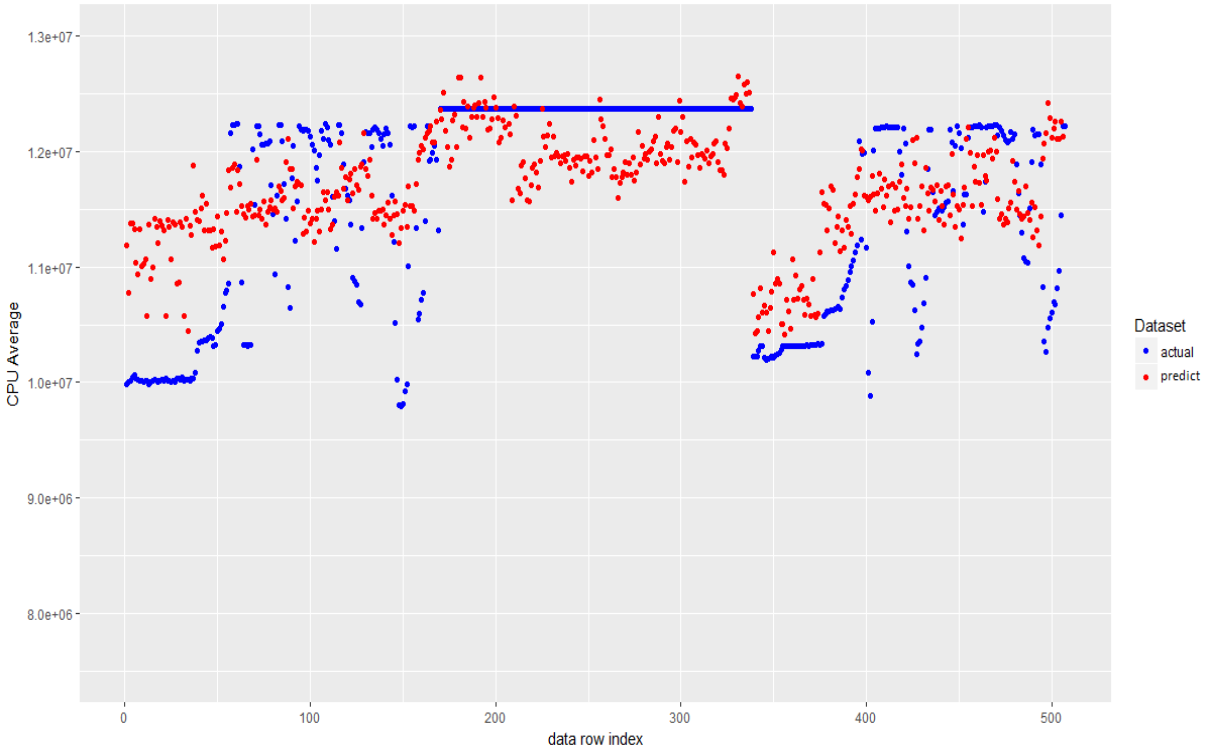
Feed Forward Network – 3 Hidden Layers with (10,6,6) Nodes					
RMSE	174.99	MAE	122.89	MAPE	4.56%

Table 8 - Validation Results of CPU Utilization Average

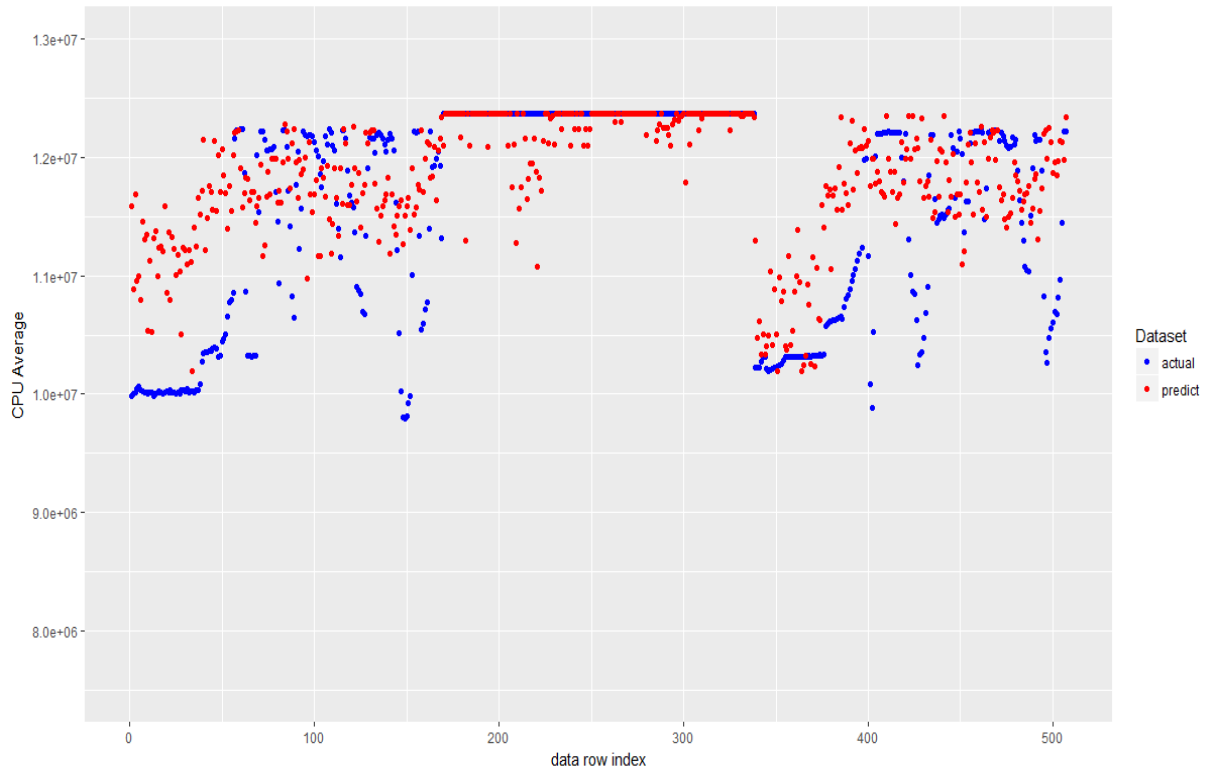
Memory Consumed Average					
Linear Regression with Stepwise Selection					
RMSE	804748.3	MAE	677964.2	MAPE	6.11%



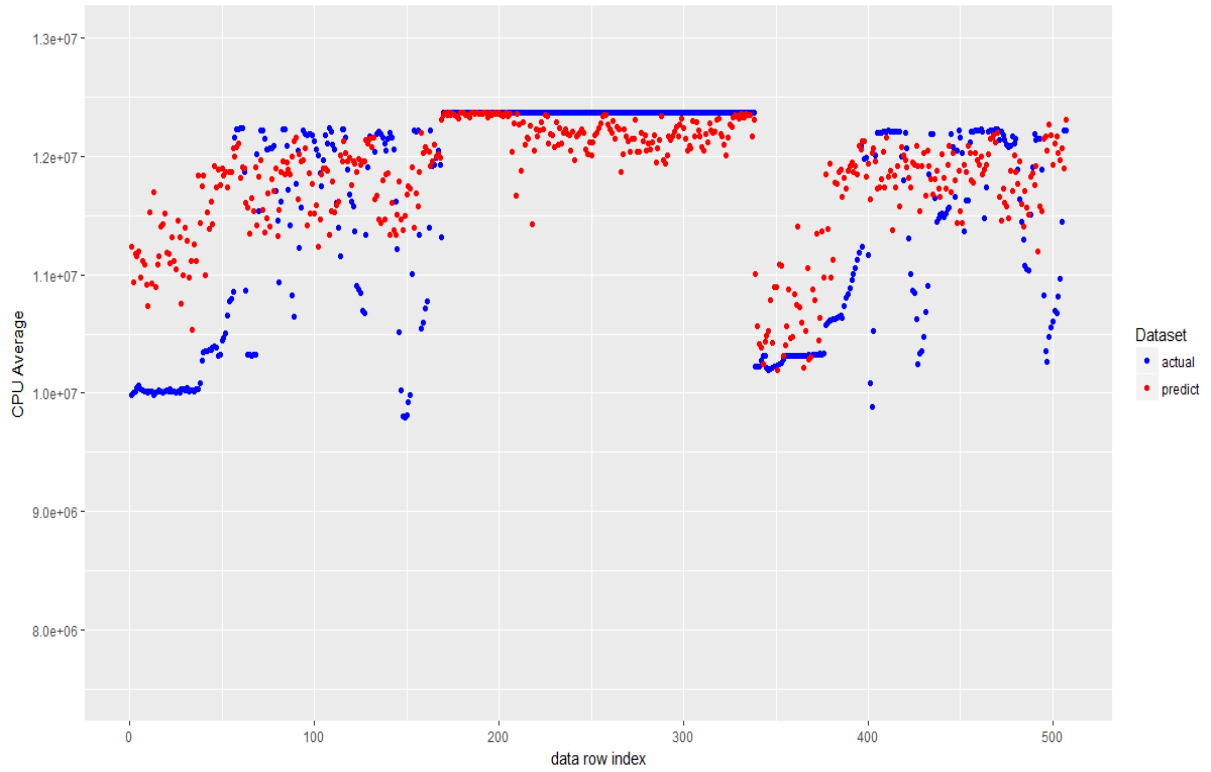
Multivariate Adaptive Regression Spline					
RMSE	696701.8	MAE	567727.9	MAPE	5.10%



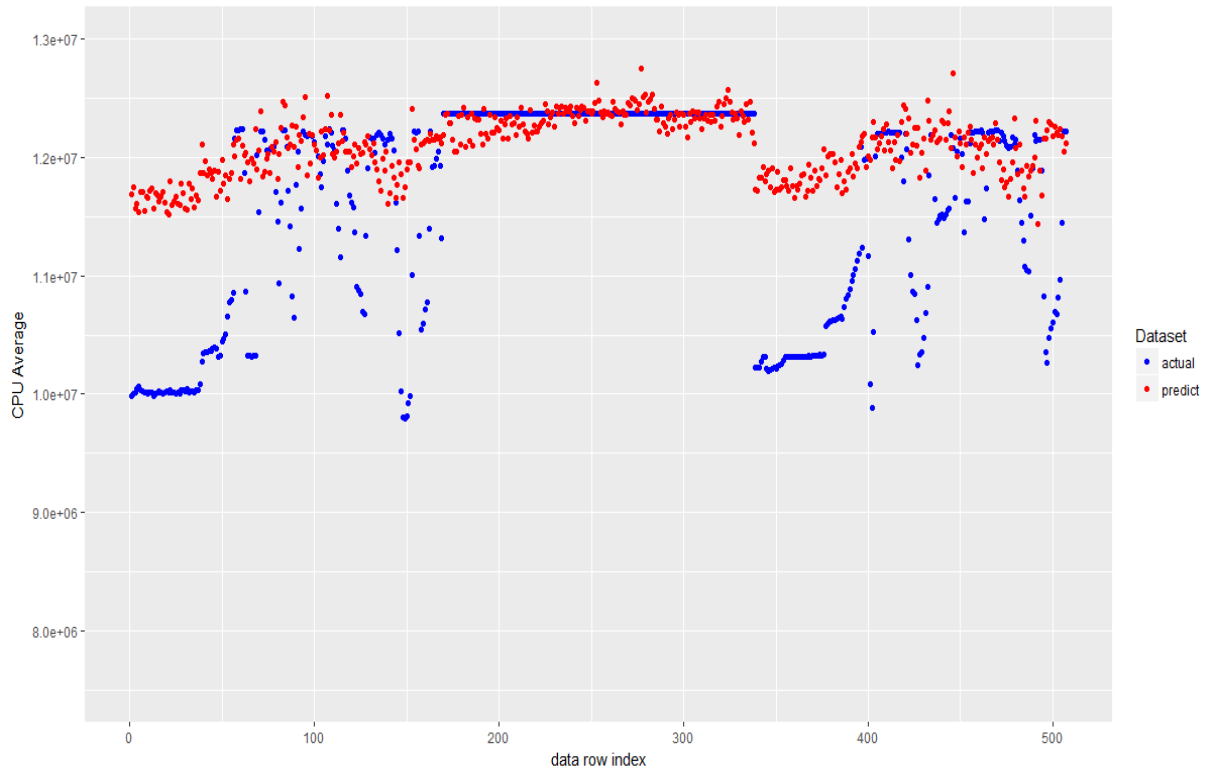
K-Nearest Neighbors					
RMSE	715199.3	MAE	495504.1	MAPE	4.56%



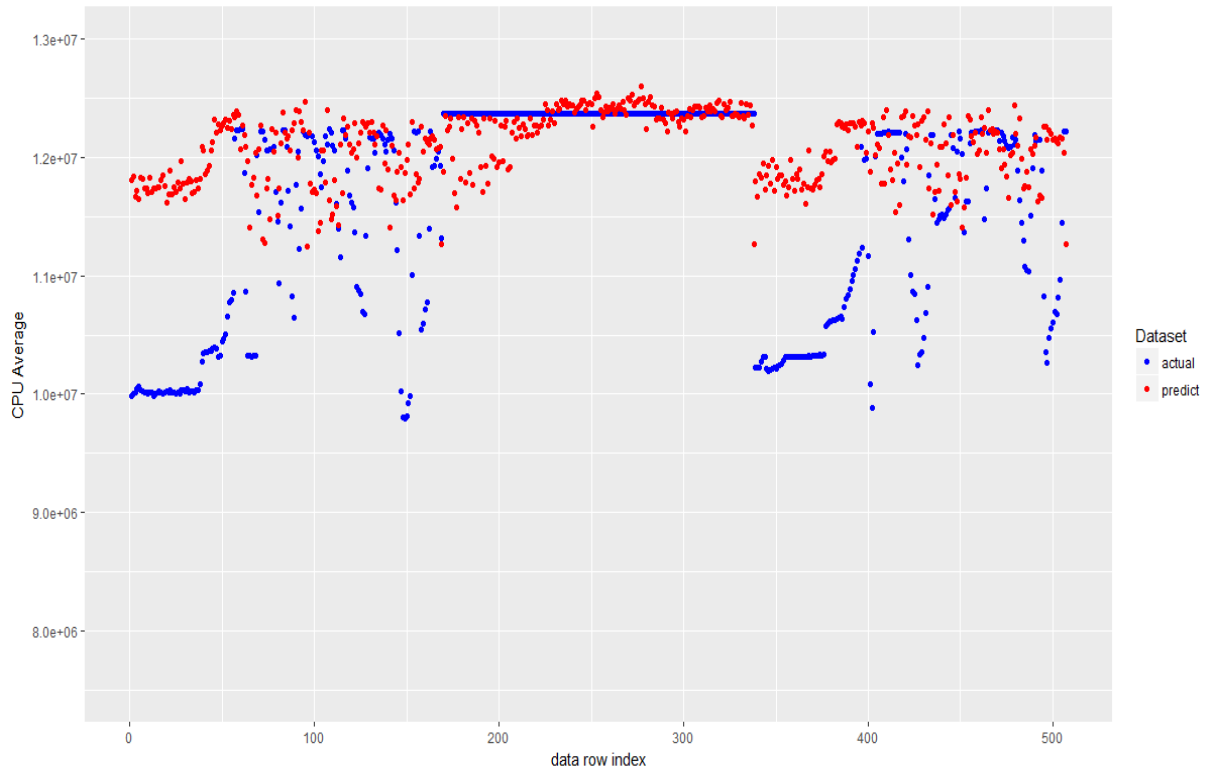
Random Forest					
RMSE	699140.2	MAE	507653.1	MAPE	4.66%



Support Vector Machines with Polynomial Kernel					
RMSE	894511.3	MAE	611632.3	MAPE	5.76%

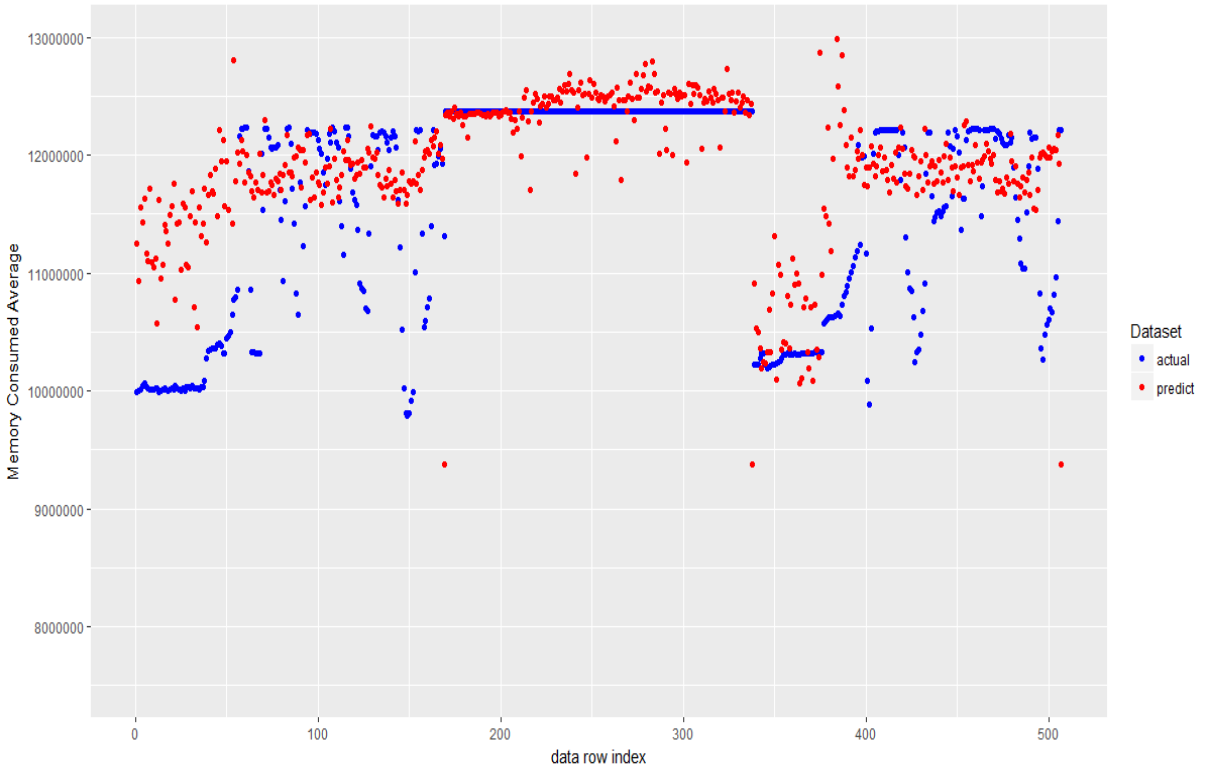


Support Vector Machines with Radial Basis Function Kernel					
RMSE	956556.6	MAE	671179.8	MAPE	6.28%

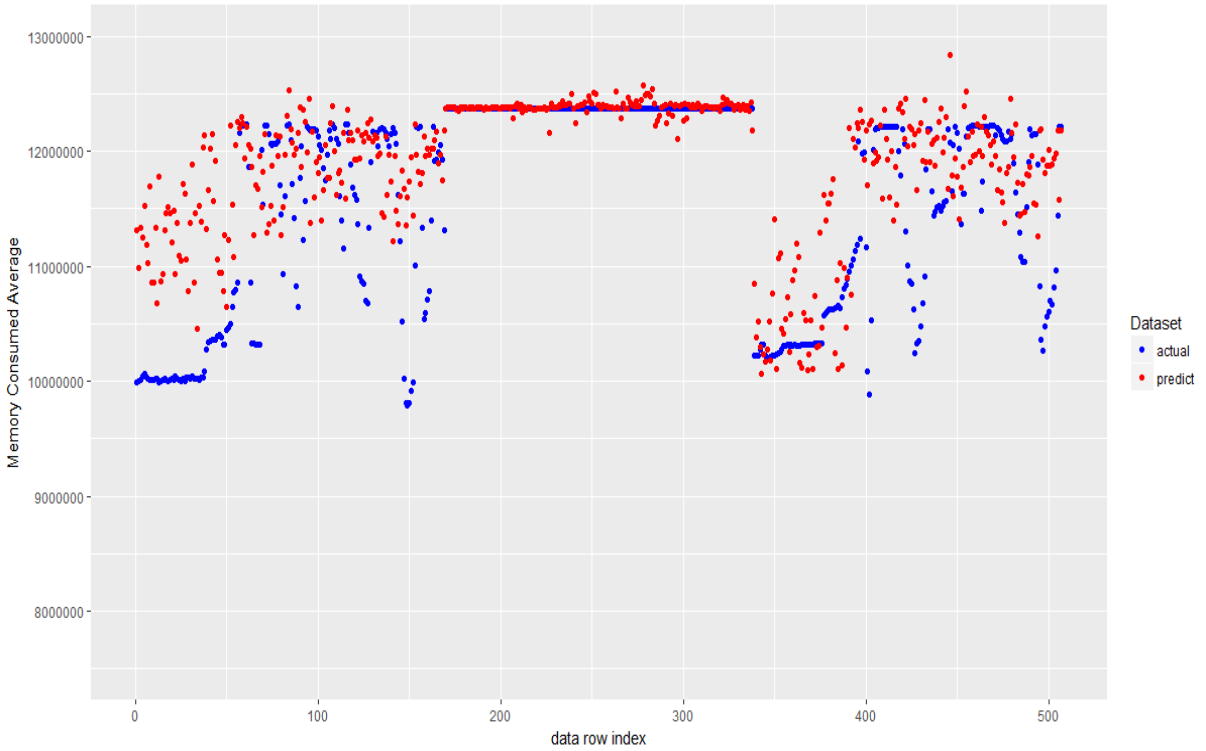


Feed Forward Network - 1 Hidden Layer with (6) Nodes					
RMSE	695119.37	MAE	504097.62	MAPE	4.64%

Feed Forward Network - 1 Hidden Layer with (8) Nodes					
RMSE	777832.43	MAE	530865.22	MAPE	4.89%



Feed Forward Network - 2 Hidden Layers with (10,6) Nodes					
RMSE	702149.94	MAE	459227.58	MAPE	4.26%



Feed Forward Network – 3 Hidden Layers with (10,6,6) Nodes					
RMSE	741571.30	MAE	498002.61	MAPE	4.58%

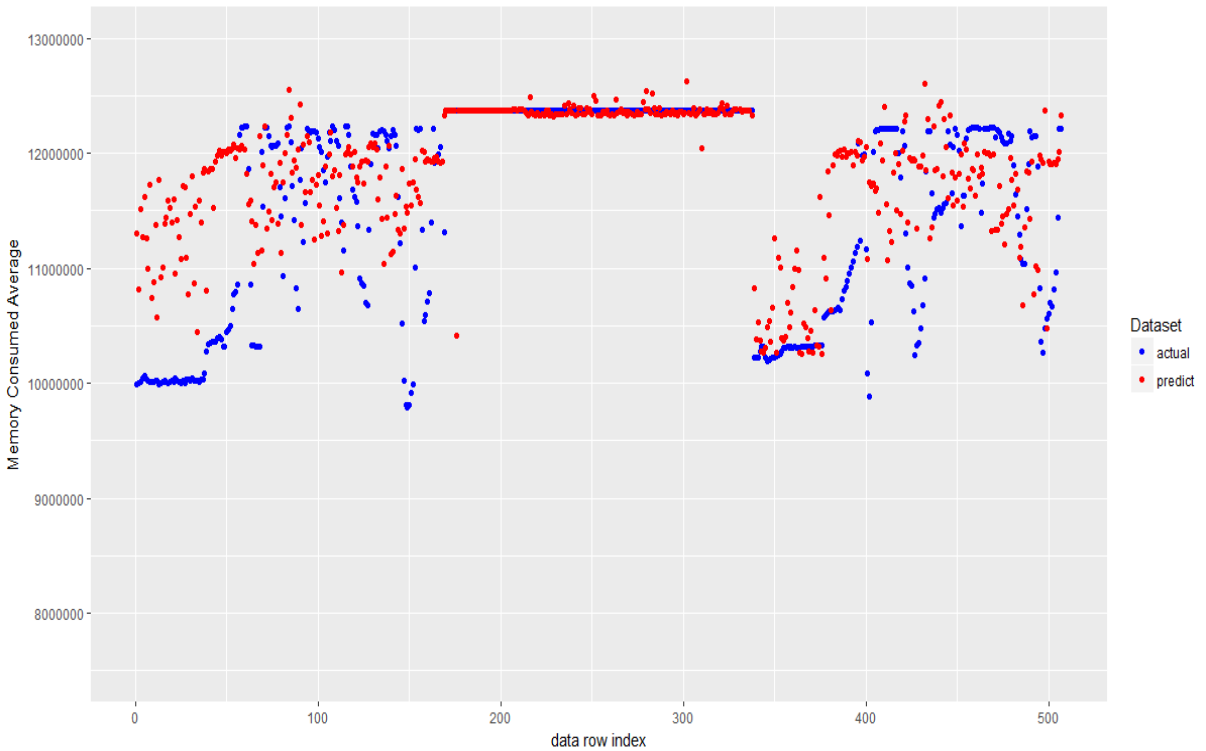
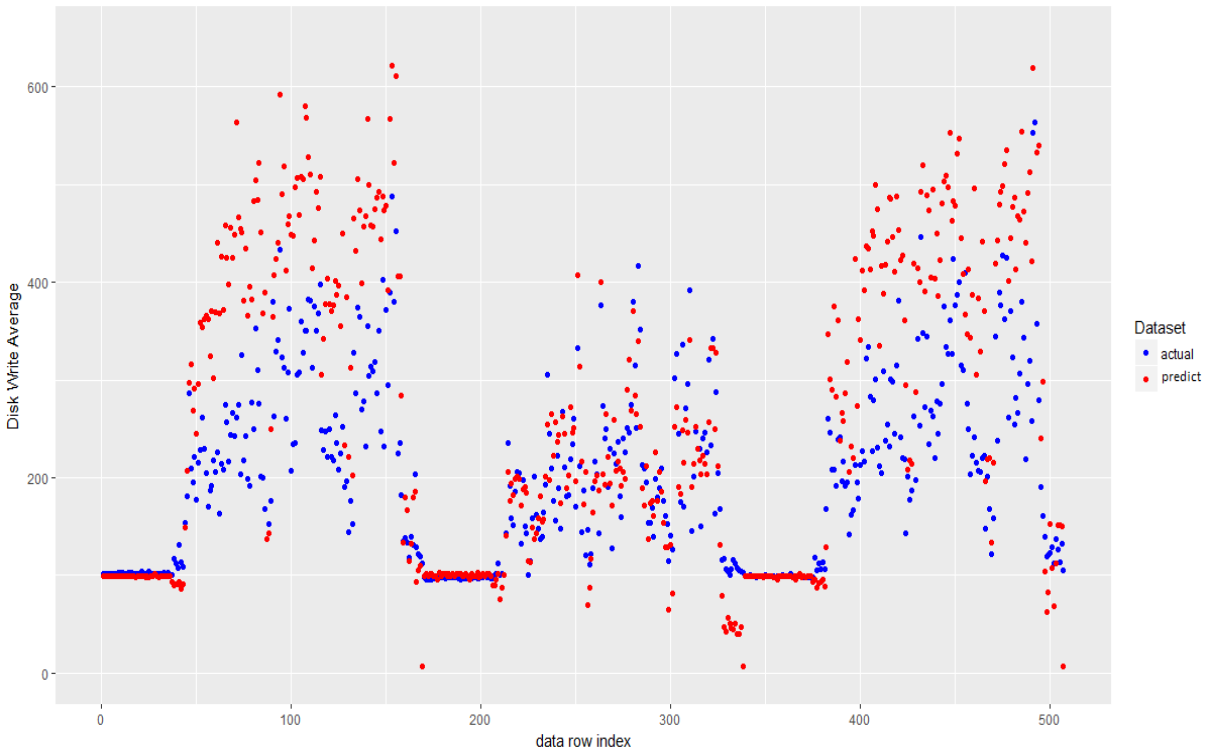


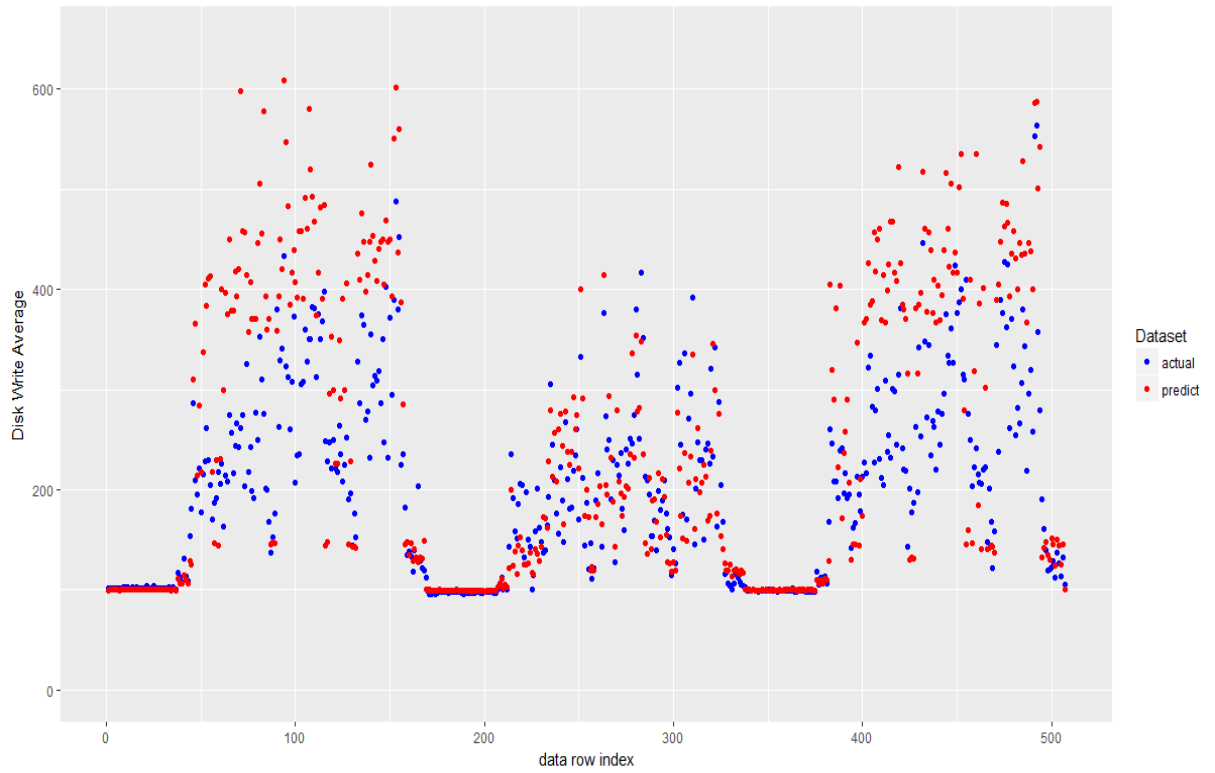
Table 9 - Validation Results of Memory Consumed Average

Disk I/O Write Average					
Linear Regression with Stepwise Selection					
RMSE	106.07	MAE	76.32	MAPE	34.63%

Multivariate Adaptive Regression Spline					
RMSE	87.21	MAE	63.35	MAPE	27.91%



K-Nearest Neighbors					
RMSE	88.85	MAE	59.91	MAPE	24.6%



Random Forest					
RMSE	100.04	MAE	69.29	MAPE	28.56%

Support Vector Machines with Polynomial Kernel					
RMSE	66.89	MAE	47.70	MAPE	21.22%

Support Vector Machines with Radial Basis Function Kernel					
RMSE	74.12	MAE	52.44	MAPE	23.36%

Feed Forward Network - 1 Hidden Layer with (6) Nodes					
RMSE	85.25	MAE	59.21	MAPE	25.88%

Feed Forward Network - 1 Hidden Layer with (10) Nodes					
RMSE	89.74	MAE	61.62	MAPE	25.75%

Feed Forward Network - 2 Hidden Layers with (10,6) Nodes					
RMSE	73.00	MAE	47.00	MAPE	4.26%

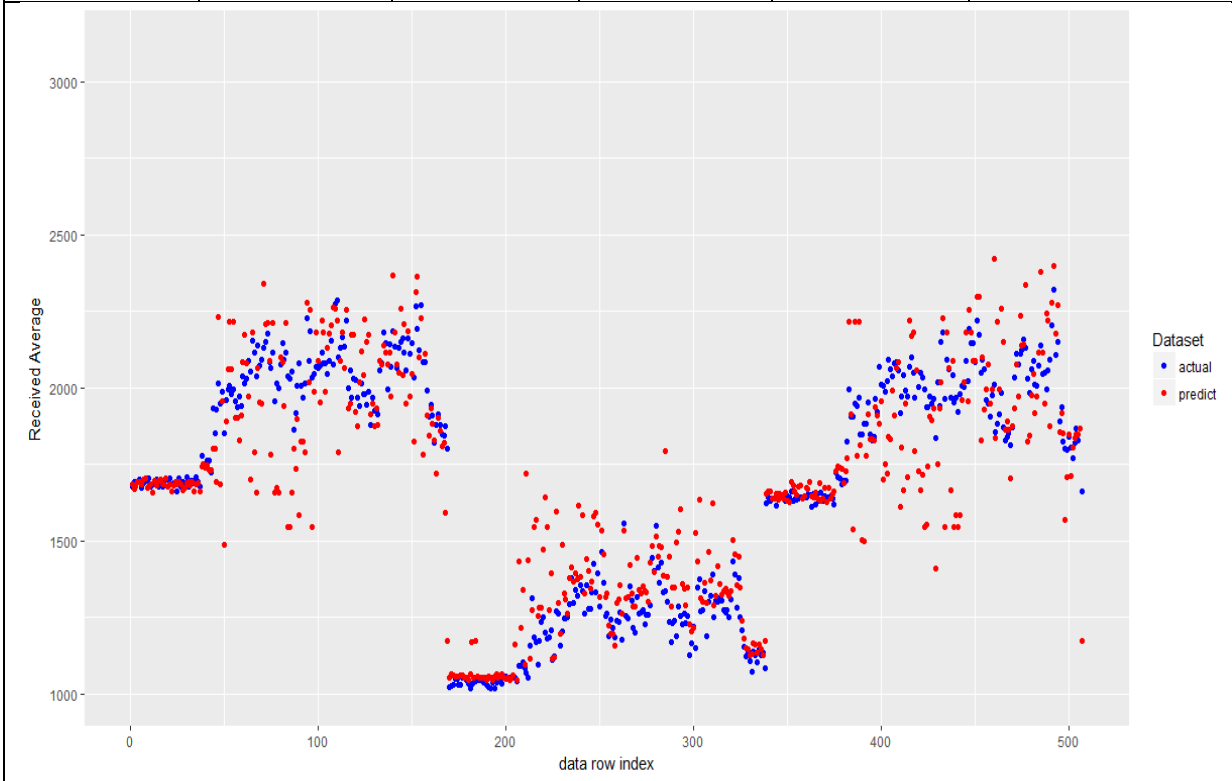
Feed Forward Network – 3 Hidden Layers with (10,6,6) Nodes					
RMSE	69.27	MAE	45.16	MAPE	19.58%

Table 10 - Validation Results of Disk I/O Write Average

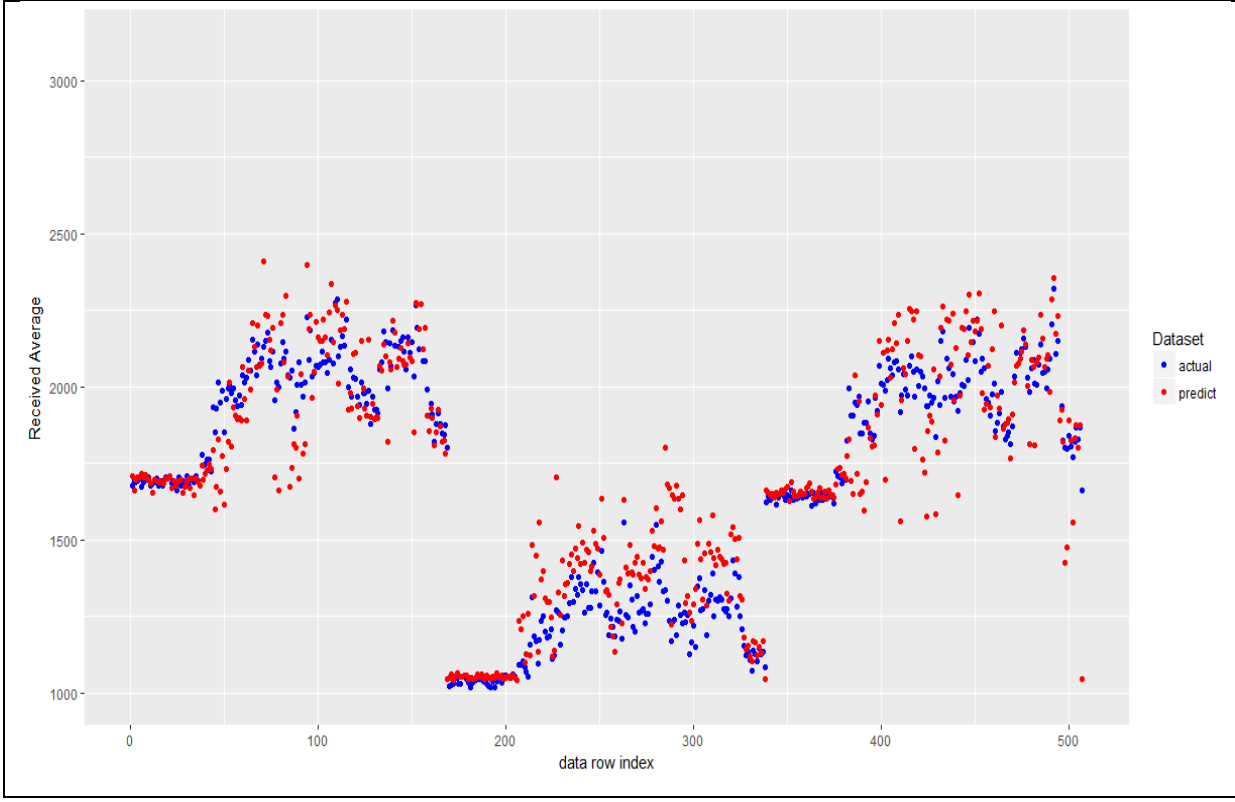
Network I/O Received Average					
Linear Regression with Stepwise Selection					
RMSE	300.38	MAE	199.56	MAPE	16.82%

Multivariate Adaptive Regression Spline					
RMSE	327.75	MAE	237.57	MAPE	14.44%

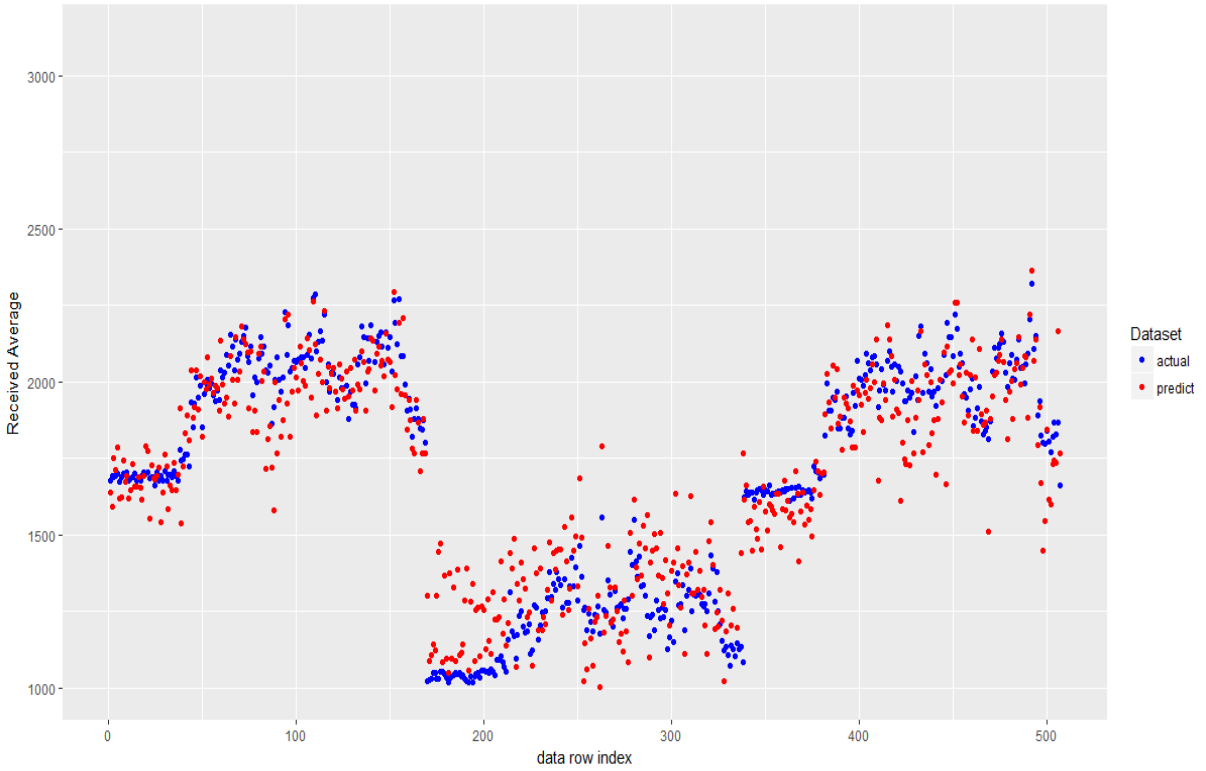
K-Nearest Neighbors					
RMSE	237.20	MAE	166.89	MAPE	11.02%



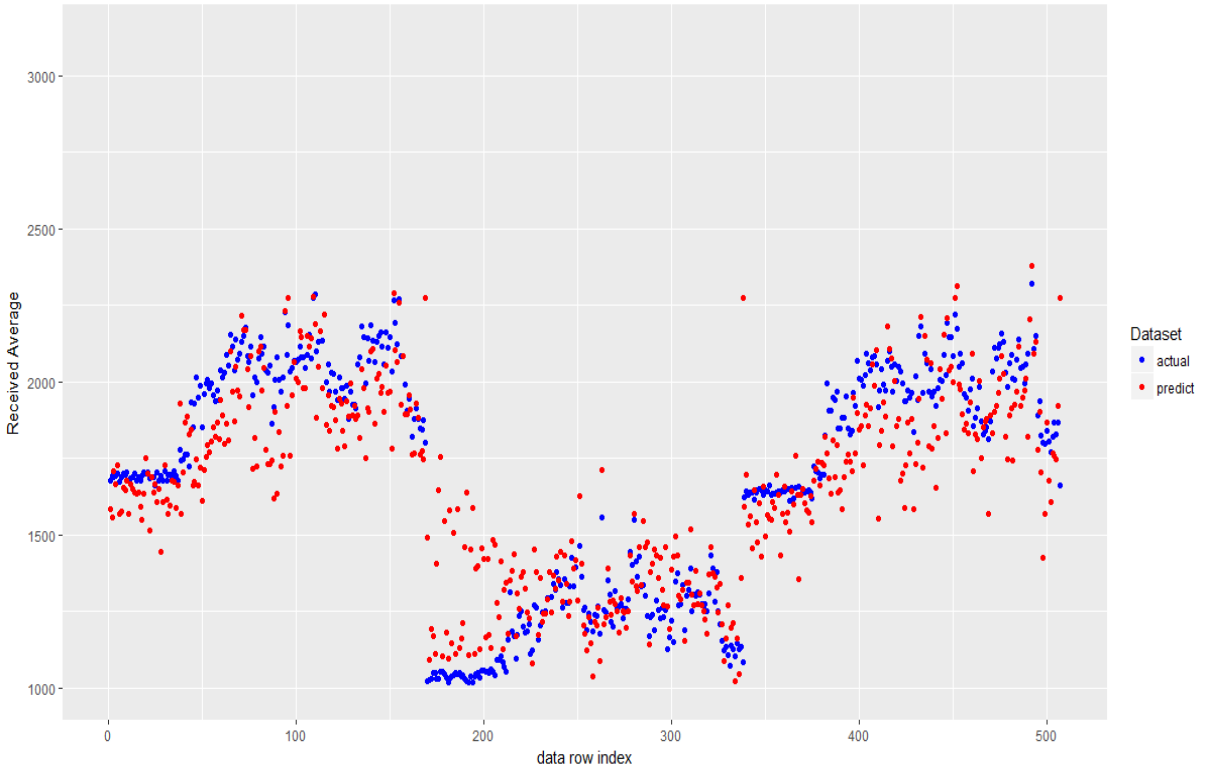
Random Forest					
RMSE	231.05	MAE	159.40	MAPE	12.21%



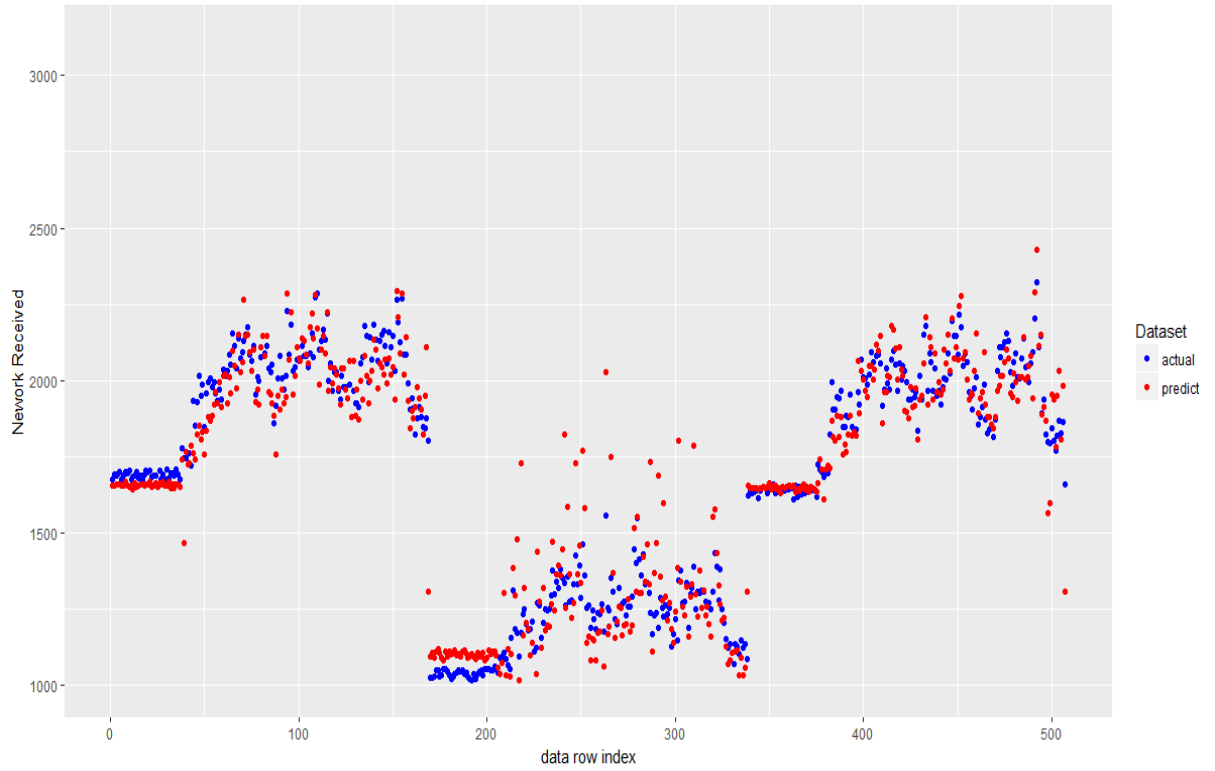
Support Vector Machines with Polynomial Kernel					
RMSE	245.58	MAE	157.68	MAPE	11.21%



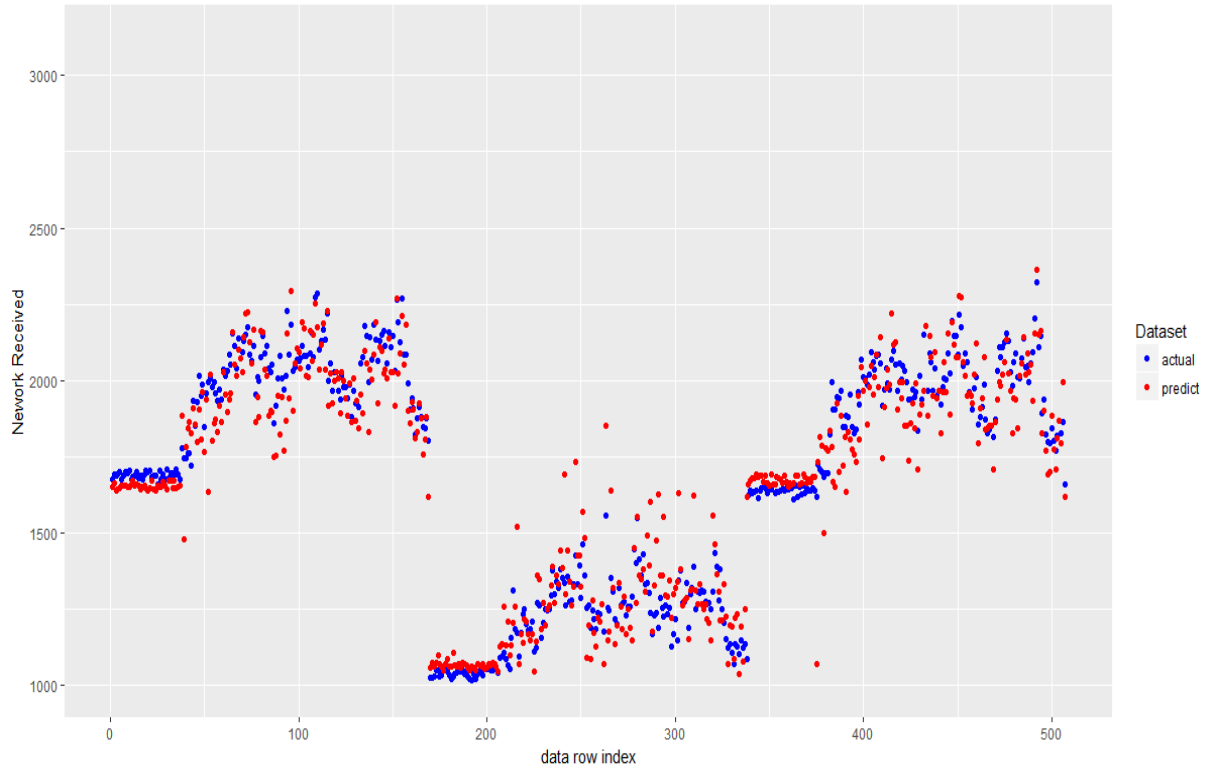
Support Vector Machines with Radial Basis Function Kernel					
RMSE	255.09	MAE	199.56	MAPE	13.21%



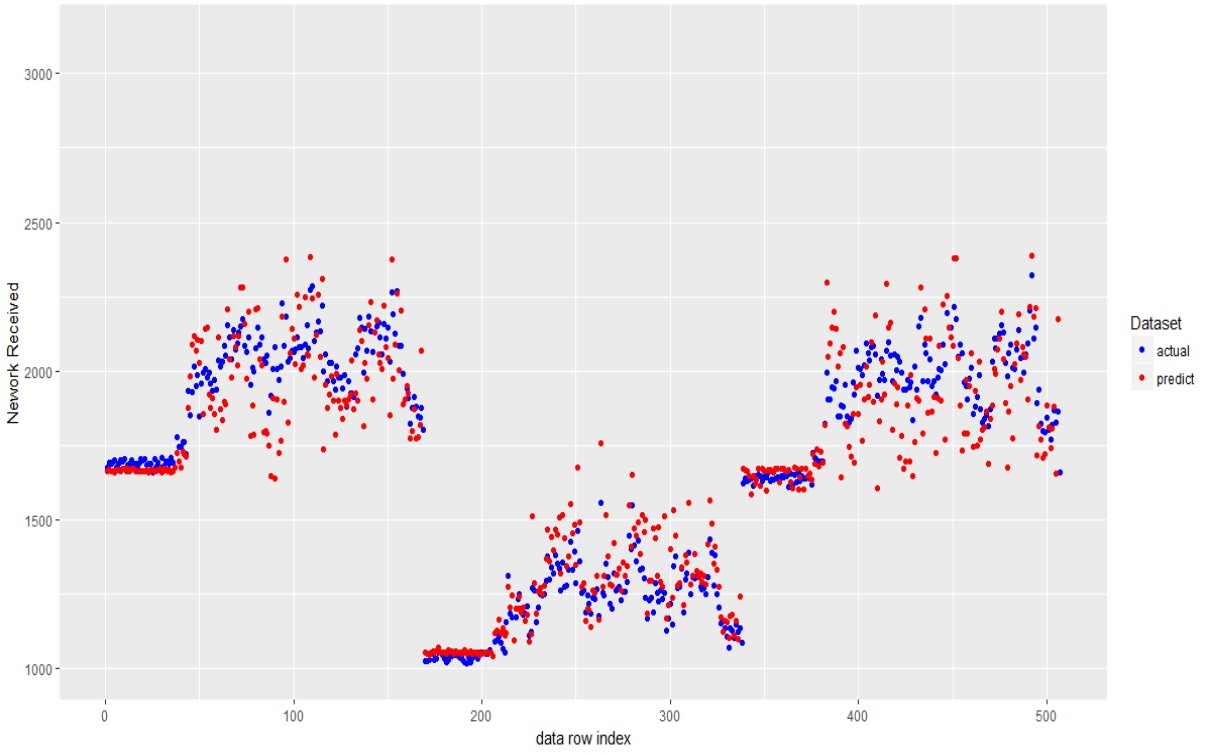
Feed Forward Network - 1 Hidden Layer with (6) Nodes					
RMSE	100.20	MAE	65.22	MAPE	4.26%



Feed Forward Network - 1 Hidden Layer with (10) Nodes					
RMSE	97.48	MAE	67.35	MAPE	4.18%



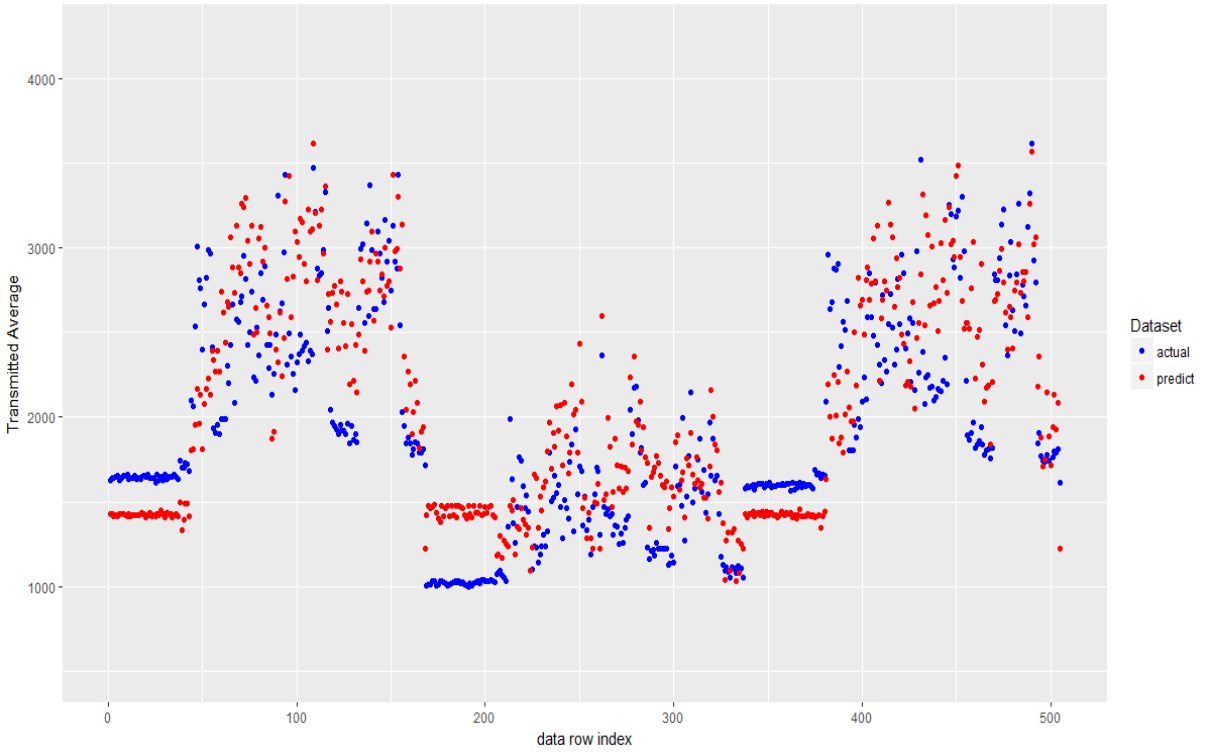
Feed Forward Network - 2 Hidden Layers with (10,6) Nodes					
RMSE	116.84	MAE	80.98	MAPE	4.66%



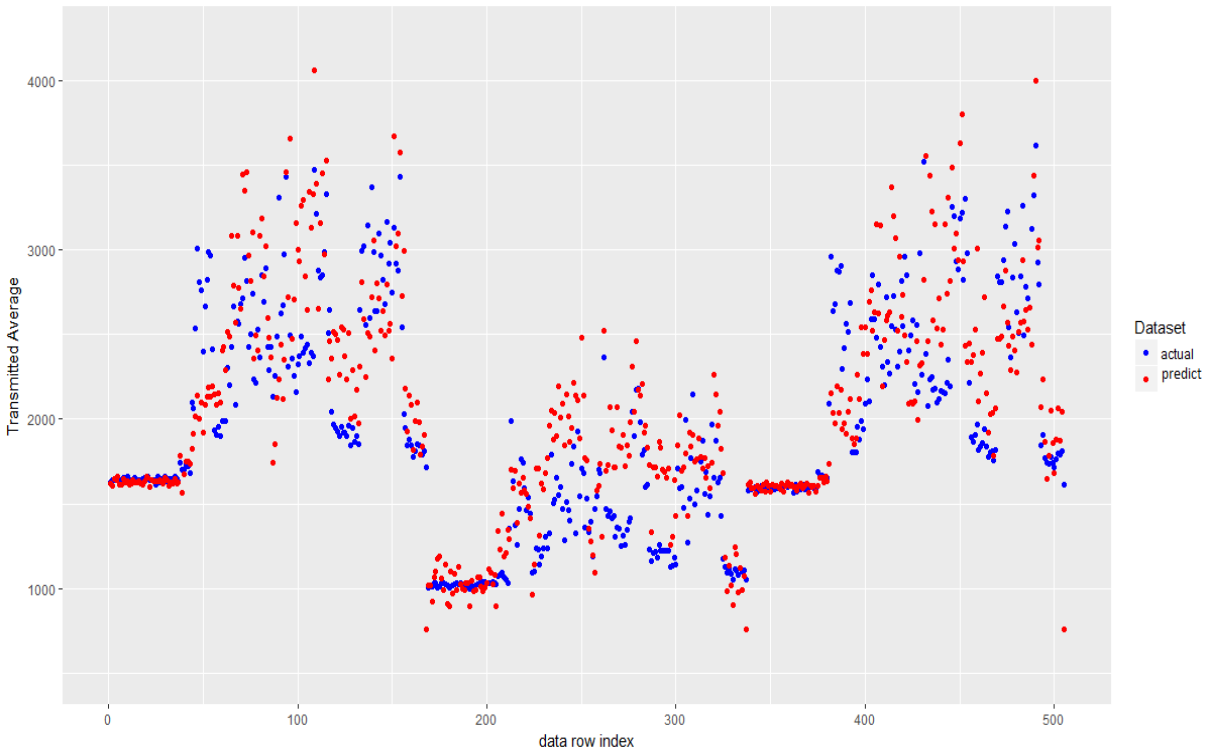
Feed Forward Network – 3 Hidden Layers with (10,6,6) Nodes					
RMSE	101.02	MAE	54.78	MAPE	3.14%

Table 11 : Validation Results of Network I/O Received Average

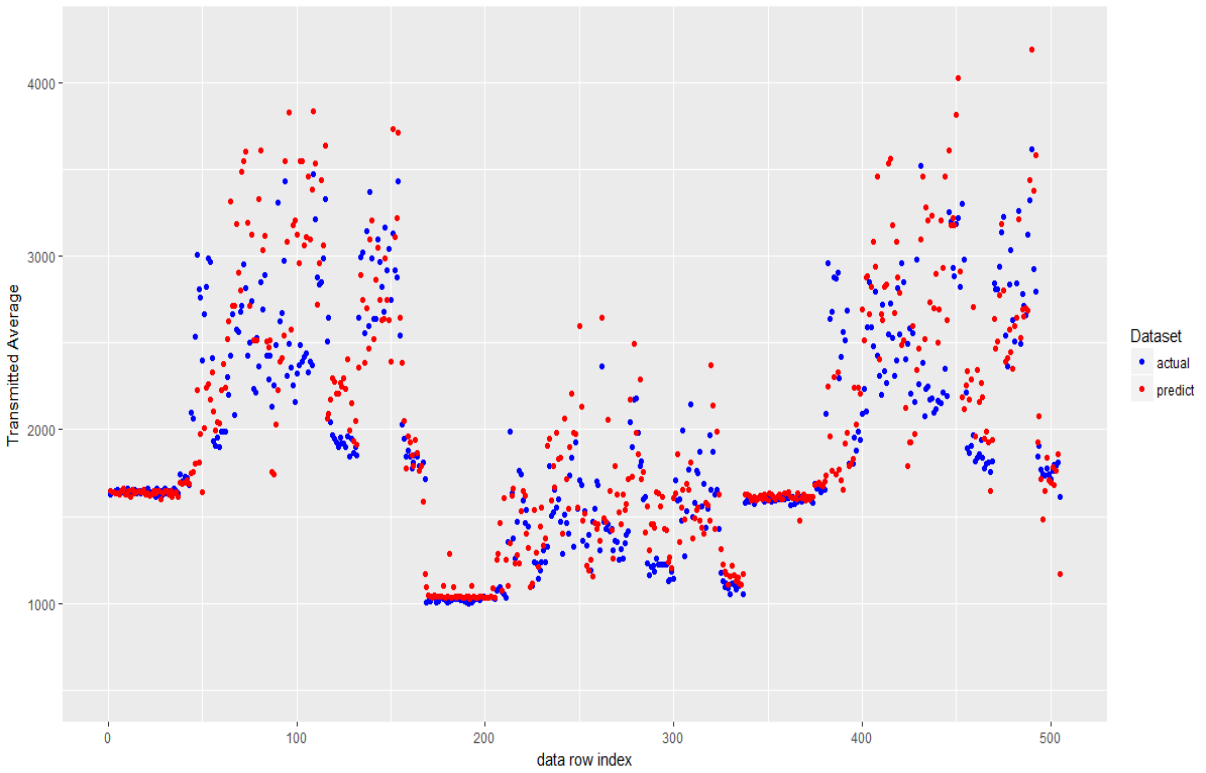
Network I/O Transmitted Average					
Linear Regression with Stepwise Selection					
RMSE	436.85	MAE	333.58	MAPE	18.29%



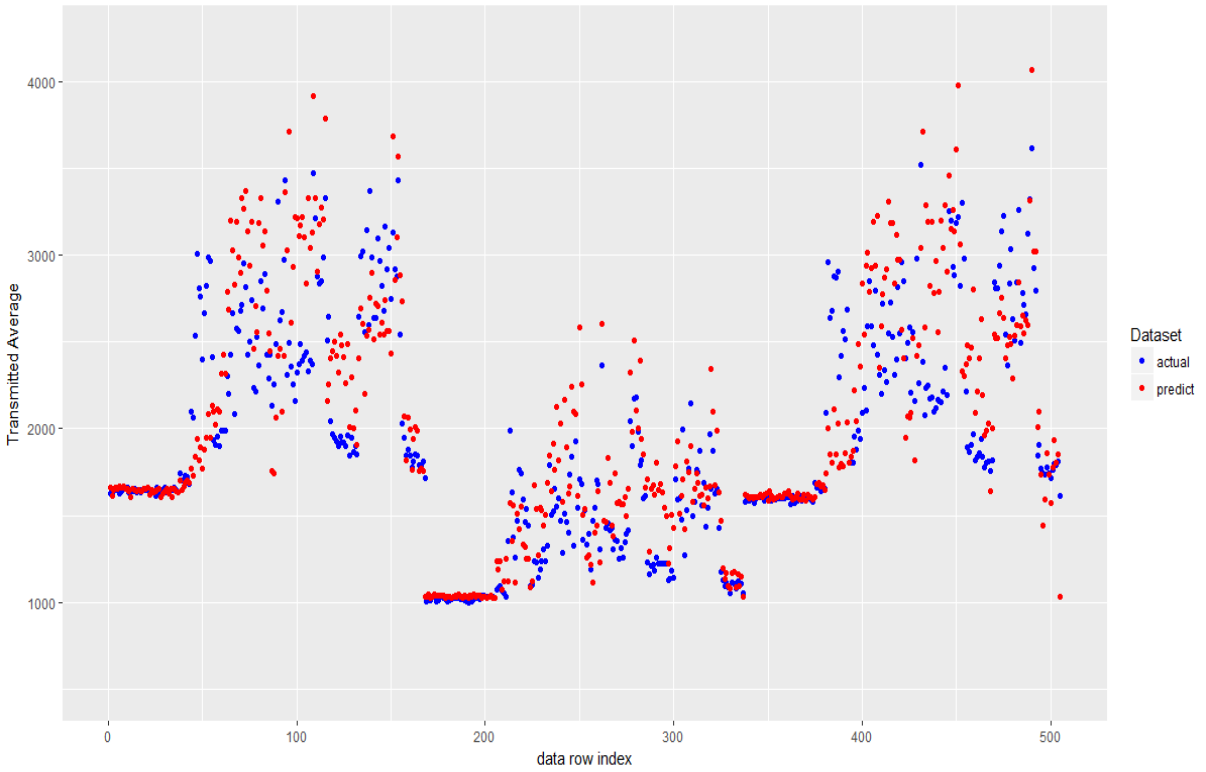
Multivariate Adaptive Regression Spline					
RMSE	285.60	MAE	436.60	MAPE	14.23%



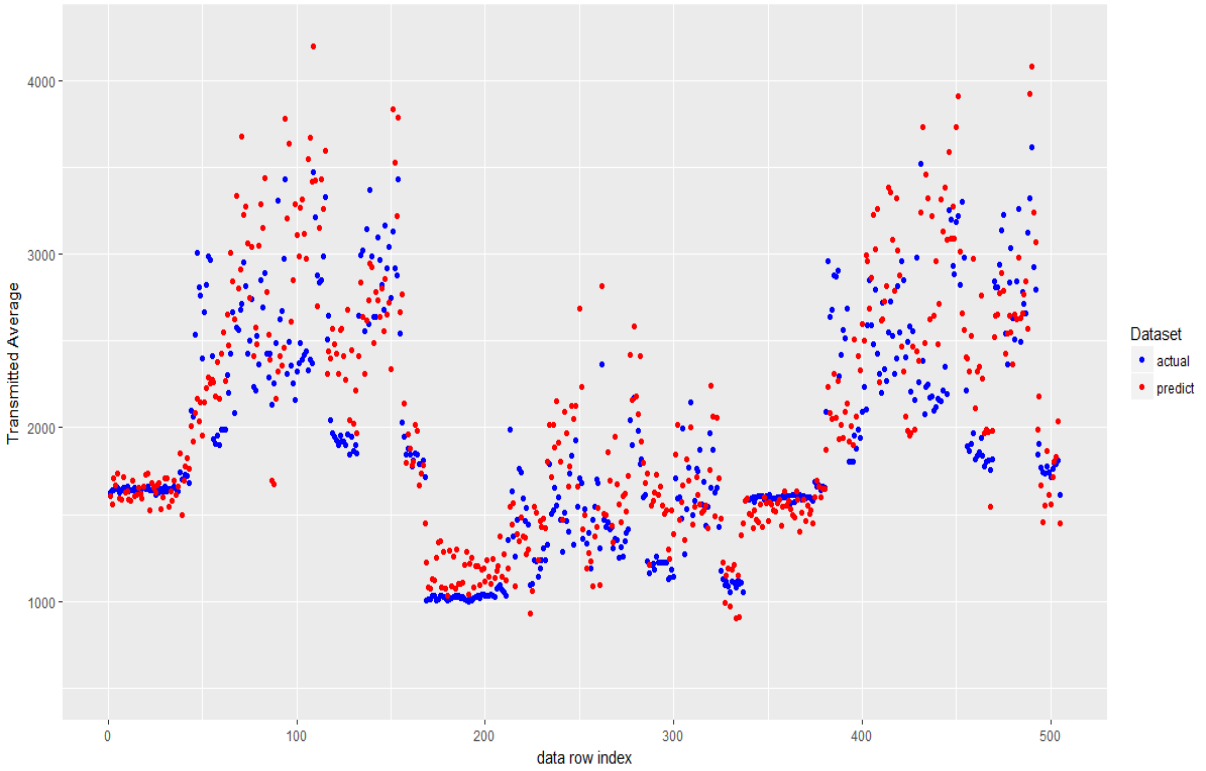
K-Nearest Neighbors					
RMSE	442.01	MAE	281.17	MAPE	13.47%



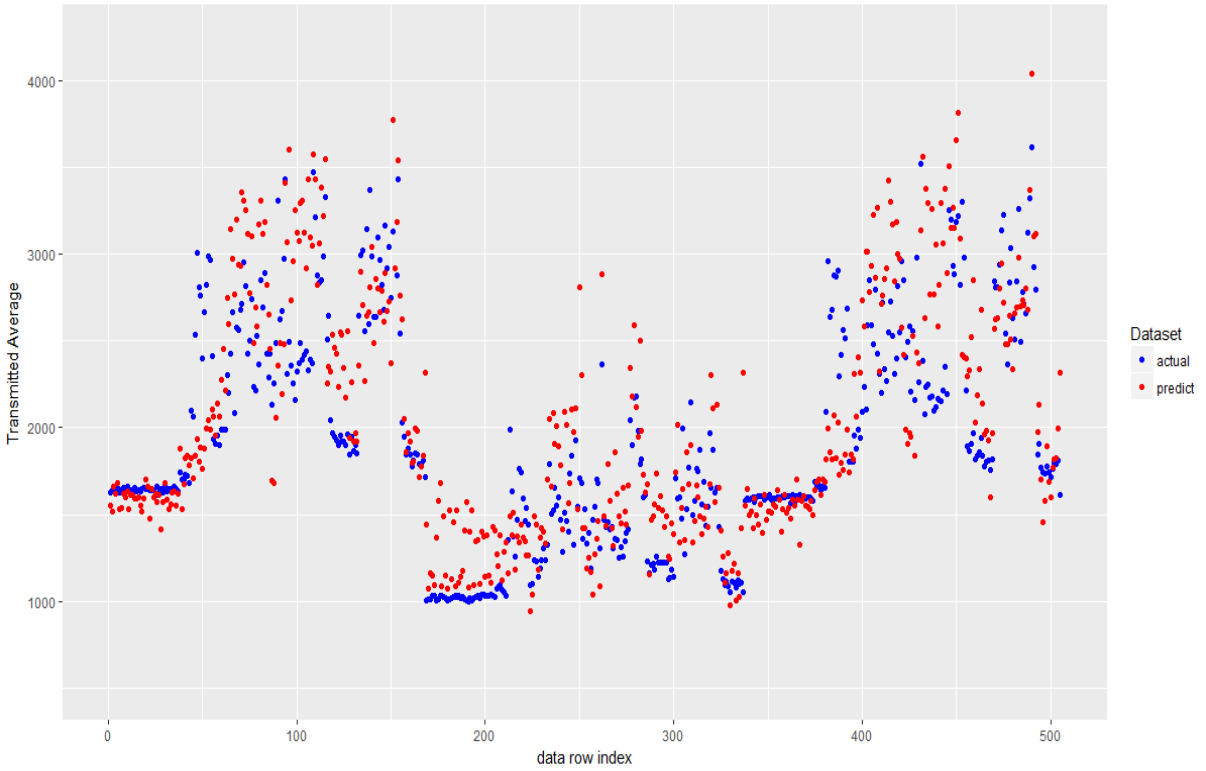
Random Forest					
RMSE	431.59	MAE	281.66	MAPE	12.51%



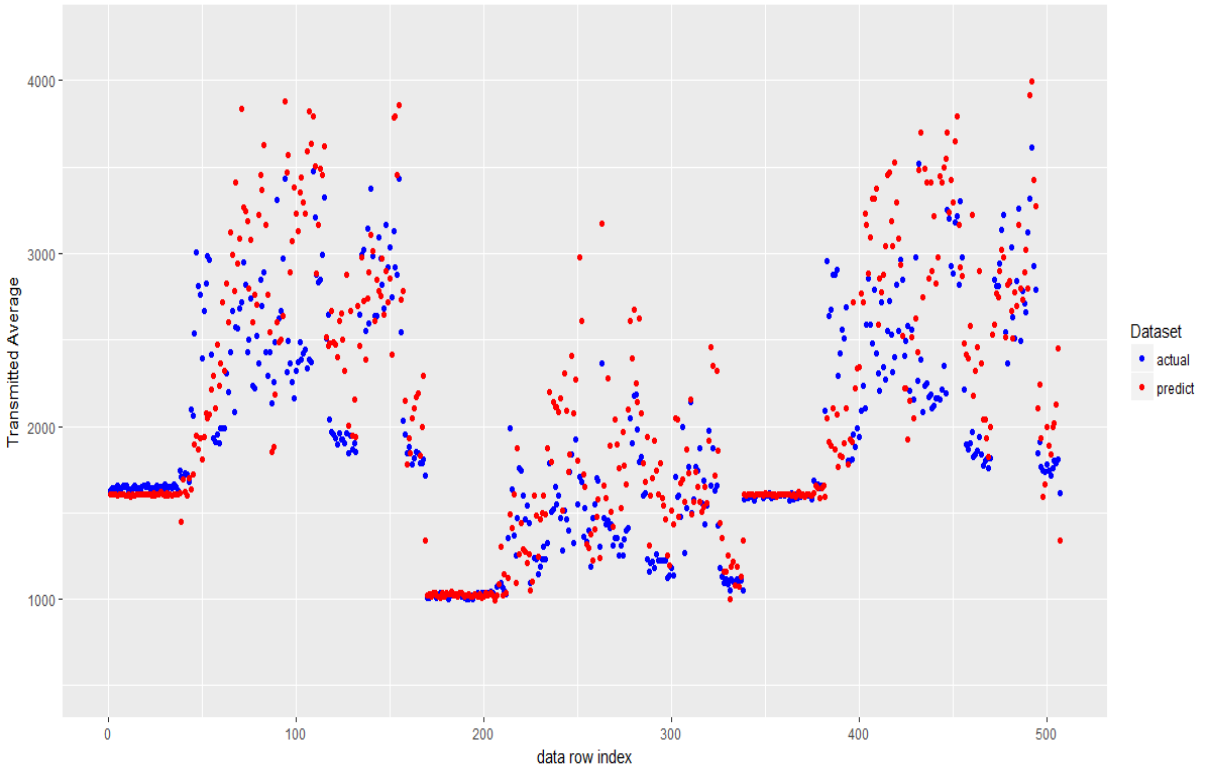
Support Vector Machines with Polynomial Kernel					
RMSE	421.14	MAE	284.41	MAPE	14.54%



Support Vector Machines with Radial Basis Function Kernel					
RMSE	414.92	MAE	279.37	MAPE	14.06%



Feed Forward Network - 1 Hidden Layer with (6) Nodes					
RMSE	496.30	MAE	333.34	MAPE	16.06%



Feed Forward Network - 1 Hidden Layer with (10) Nodes					
RMSE	502.12	MAE	330.78	MAPE	15.51%

Feed Forward Network - 2 Hidden Layers with (10,6) Nodes					
RMSE	454.49	MAE	274.28	MAPE	12.11%

Feed Forward Network – 3 Hidden Layers with (10,6,6) Nodes					
RMSE	422.16	MAE	269.92	MAPE	12.70%

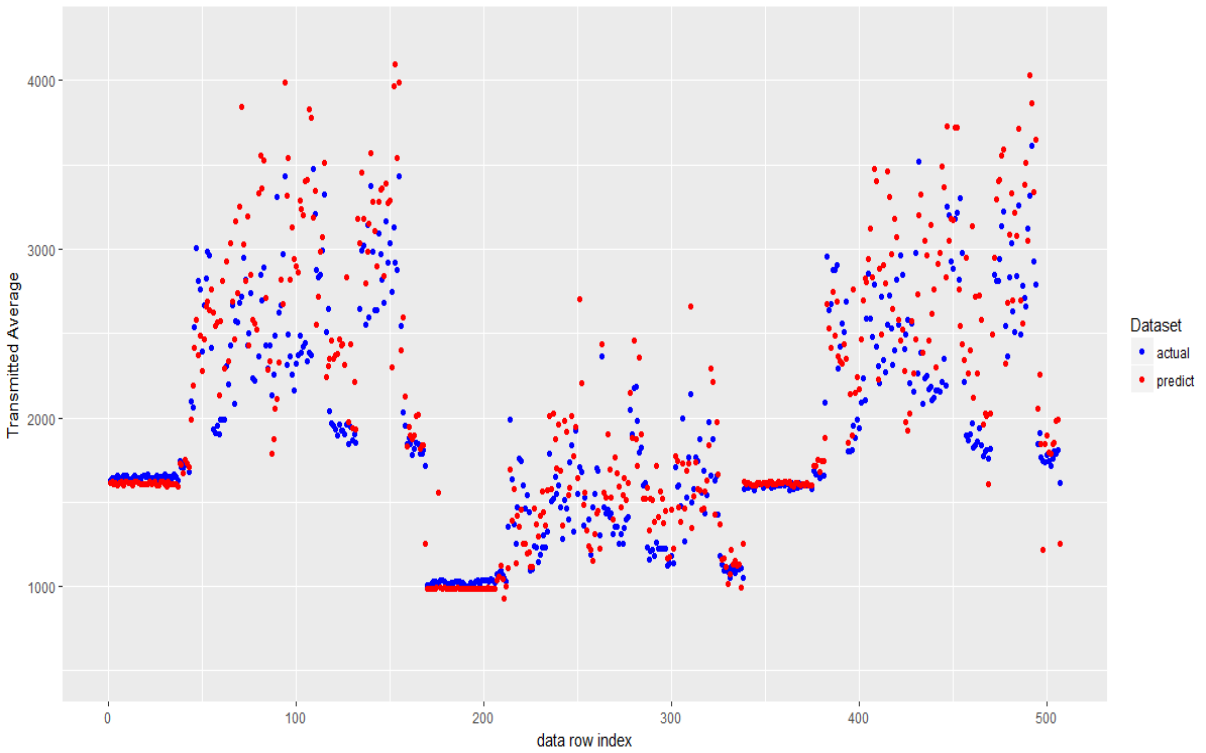
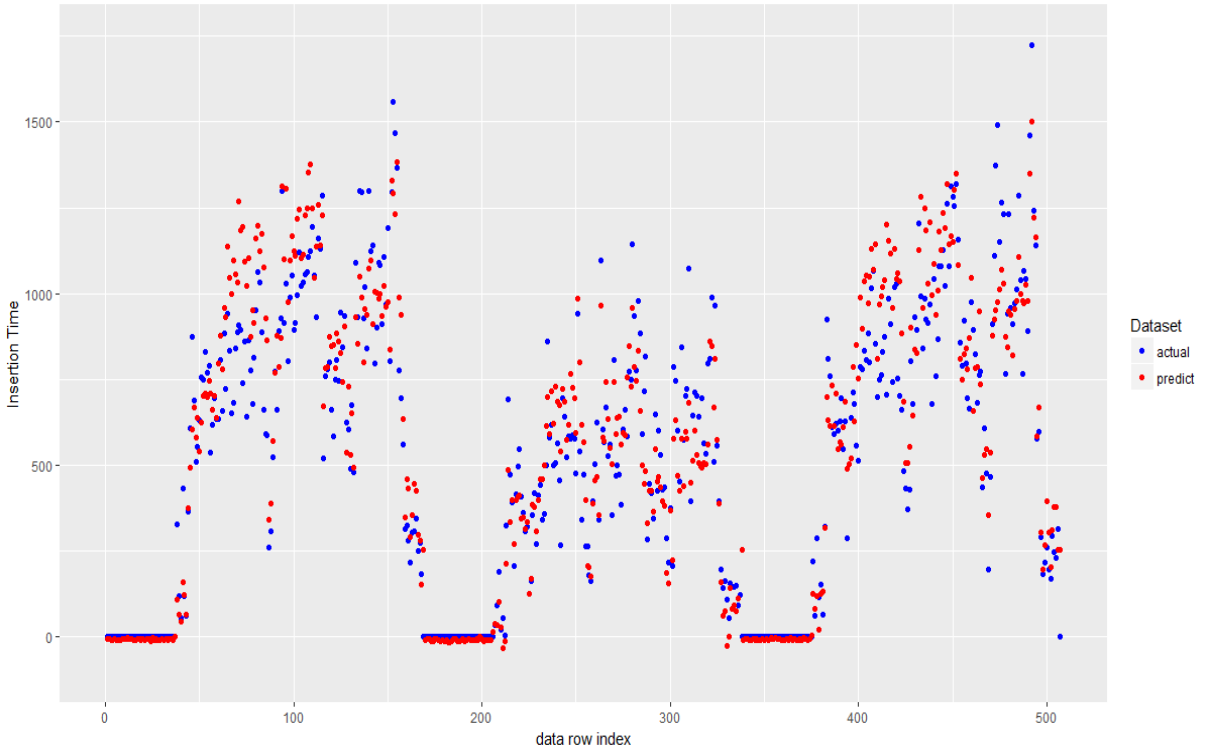
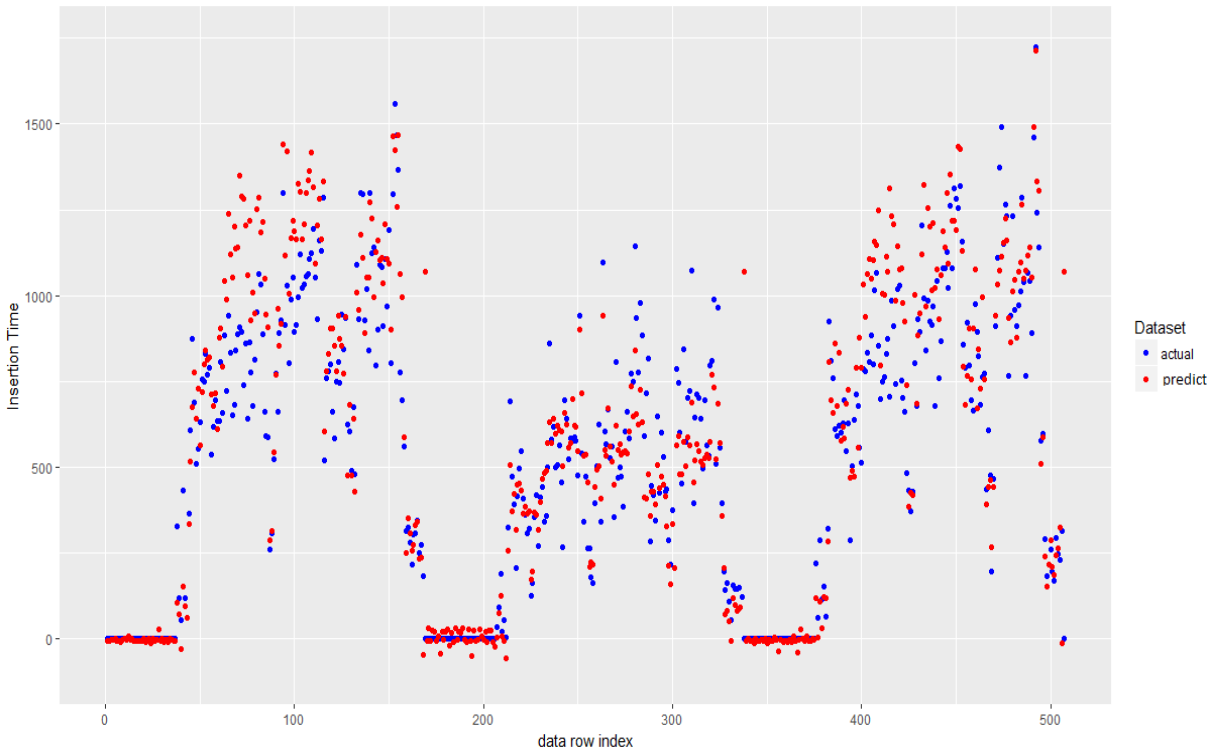


Table 12 : Validation Results of Network I/O Transmitted Average

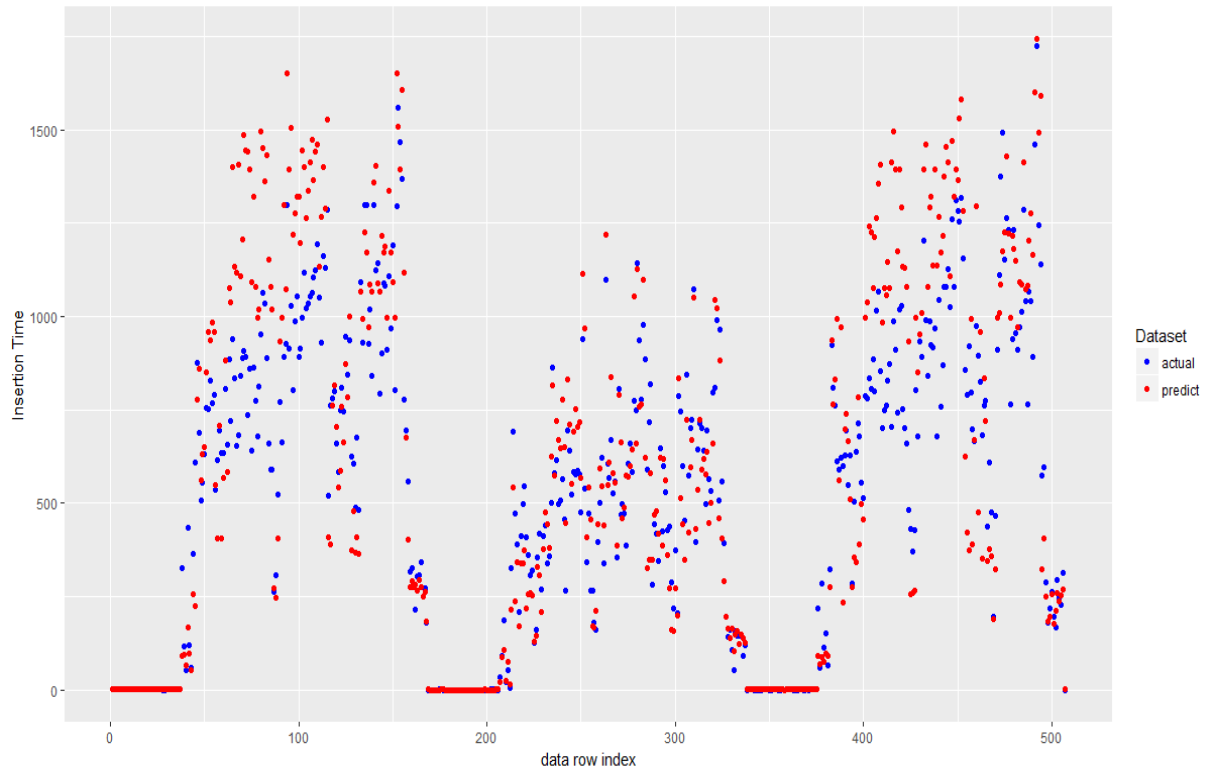
Insertion Time Average					
Linear Regression with Stepwise Selection					
RMSE	143.77	MAE	104.45	MAPE	184.8%



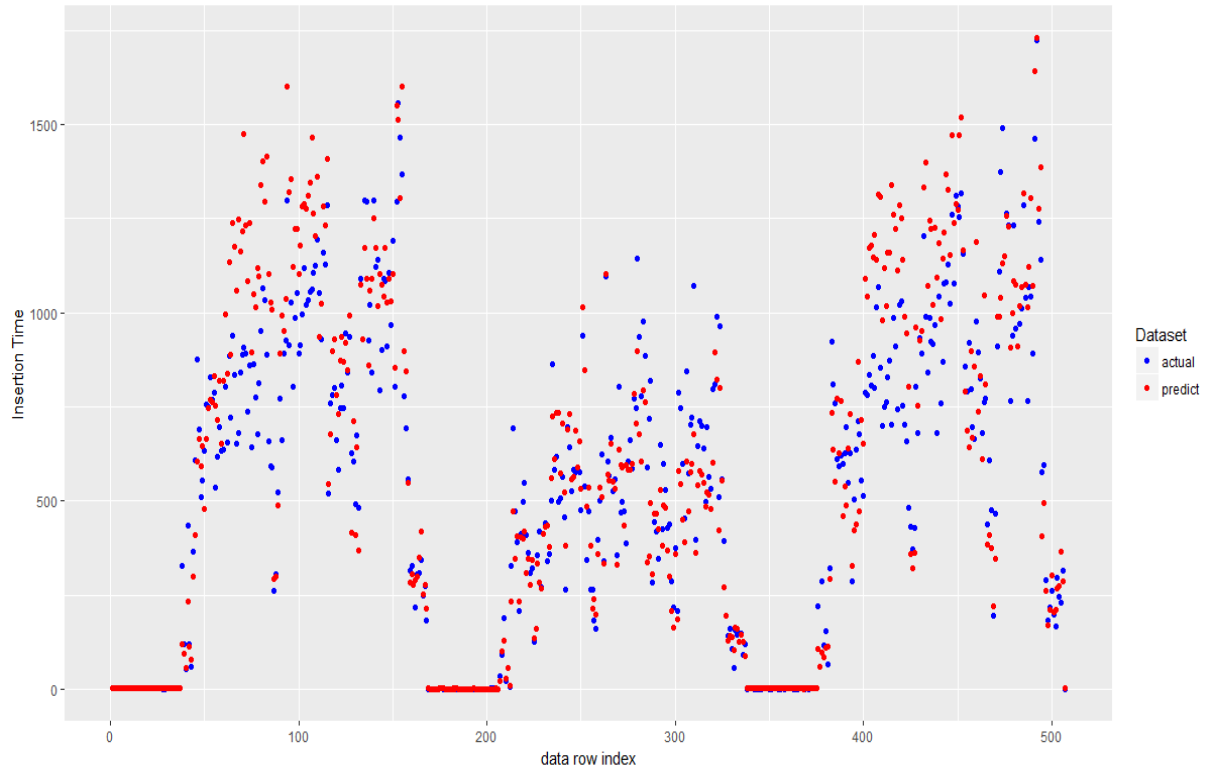
Multivariate Adaptive Regression Spline					
RMSE	168.36	MAE	115.13	MAPE	219.9%



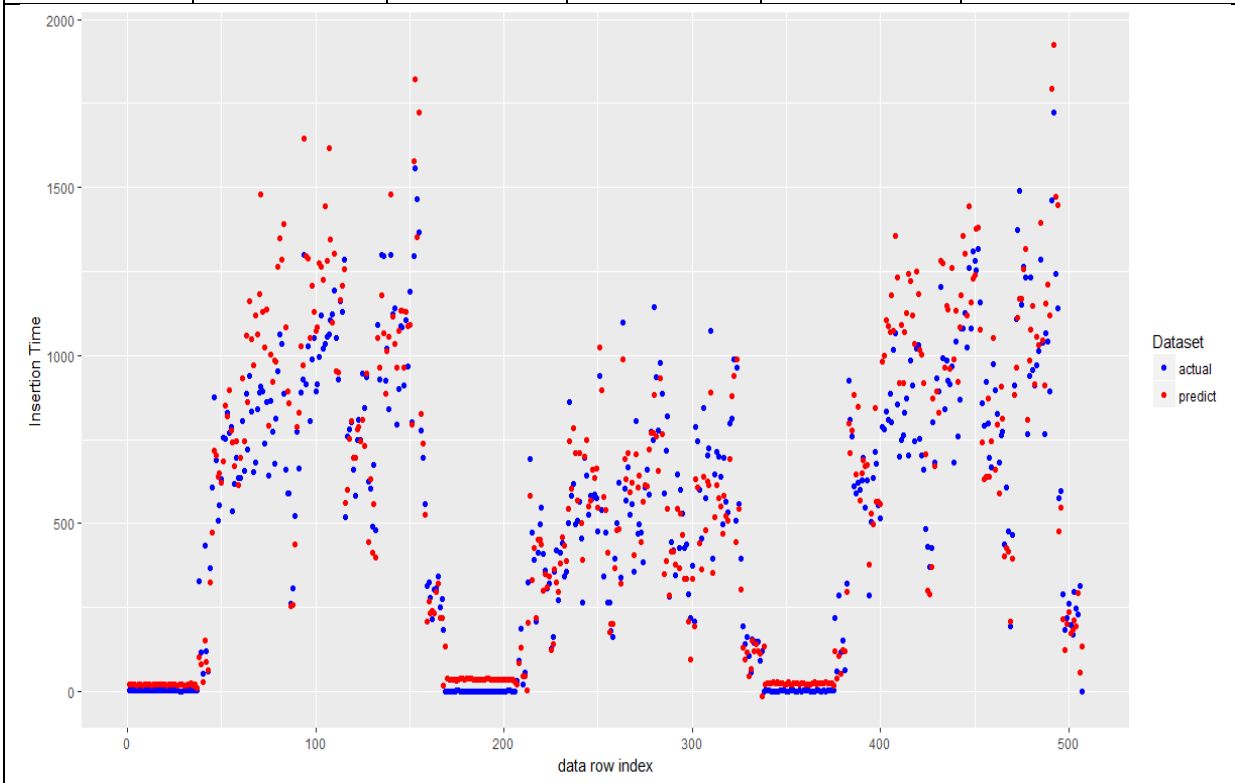
K-Nearest Neighbors					
RMSE	216.20	MAE	152.01	MAPE	27.45%



Random Forest					
RMSE	208.74	MAE	148.45	MAPE	26.33%



Support Vector Machines with Polynomial Kernel					
RMSE	166.63	MAE	118.27	MAPE	237.8%



Support Vector Machines with Radial Basis Function Kernel					
RMSE	197.71	MAE	142.82	MAPE	339.4%

Feed Forward Network - 1 Hidden Layer with (6) Nodes					
RMSE	148.27	MAE	103.97	MAPE	159.4%

Feed Forward Network - 1 Hidden Layer with (10) Nodes					
RMSE	166.98	MAE	112.44	MAPE	85.65%

Feed Forward Network - 2 Hidden Layers with (10,6) Nodes					
RMSE	156.01	MAE	104.72	MAPE	182.1%

Feed Forward Network – 3 Hidden Layers with (10,6,6) Nodes					
RMSE	179.11	MAE	125.67	MAPE	241.1%

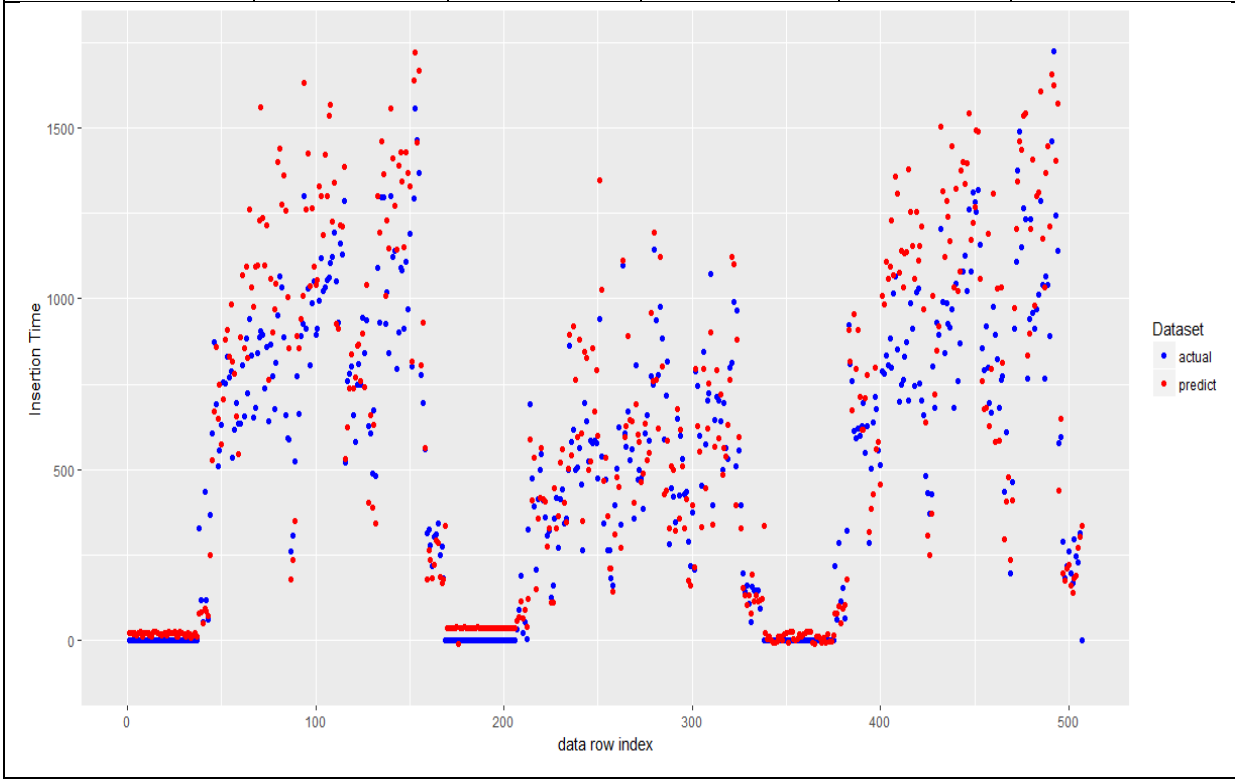


Table 13 : Validation Results of PM Insertion Time Average