UNIVERSITY OF AGDER

Activity Forecasting for the Android Smartphones Using Gaussian Processes

Ines Abdennadher

Supervisor
Ole-Christoffer Granmo

This Master's Thesis is carried out as a part of the education at the University of Agder and is therefore approved as a part of this education. However, this does not imply that the University answers for the methods that are used or the conclusions that are drawn.

University of Agder, 2012

Faculty of Engineering and Science

Department of Information and Communication Technology

Abstract

Smartphones have become amazingly popular the last few years, most likely due to the operating system revolution that has taken place, in particular when it comes to the introduction of sophisticated applications. The Android operating system is nowadays known by the most famous and used one thanks to the new and multiple functionalities it offers.

Despite the advantages of this revolution, it has come with a high cost: The level of sophistication and the multiple functions now offered typically rely on high quality screen, intensive CPU usage, and extensive networking. As a result, power consumption increases persistently, hindering further advances and lead to a short battery life time. So, techniques for predicting the future behavior of the Android phone are needed and can serve as a basis for power management.

The present Master's thesis seeks to develop a better understanding of how machine learning can be efficient and useful to predict the future behavior of the smartphone. Two learning approaches *KNN Regression* and *Gaussian Process Regression* are used to do the forecast of the CPU usage, the screen utilization and the network activity based on historical collected measurements of these three components. The results reveal that the Gaussian Process Regression model is most efficient and accurate than the KNN regression model with a prediction error rate between 1% and 7%. By using the Gaussian Process Regression model, it becomes possible to apply power management techniques to smartphones. In fact, by predicting the future inactivity periods and the idle states of the phone, components of the smartphone can be turned off in the right time and then avoid extensive switching between on- and off-modus, which is expensive in itself and consumes too much power; or by reducing the CPU frequency, disconnecting the phone from the network and adjusting the screen quality to be less bright in idle mode, battery power consumption can be significantly reduced.

Preface

The master thesis that is presented in front of you is the result of half a year of research done in the context of obtaining the degree Master of Science program in Information and Communication Technology (ICT) at the University of Agder (UIA), Faculty of Engineering and Science in Grimstad, Norway. The work on this project was started at the end of October and was accomplished on May.

The main purpose of this thesis work is to present an accurate learning approach that can be used as a basis for power management to reduce battery power usage for smartphones, by predicting the future behavior of the system based on historical measurements.

The idea behind this project was suggested by me as well as my supervisor Professor Ole-Christoffer Granmo in collaboration with ST-Ericsson Company of Grimstad, Norway.

I would like to acknowledge my supervisor Ole-Christoffer Granmo for his continuous assistance, his excellent guidance and precious advices during the fulfillment of this project.

I also want to thank Jonny Ervik, the Technical Manager at ST-Ericsson for pleasant and helpful technical discussions due to the challenges of the project.

I would like to give thanks to Stian Haugen from ST-Ericsson who cooperated so generously with me and help me in the collection of data.

Additional thanks go to Professor Frank Li for his valuable contributions.

Grimstad, 2012

Ines Abdennadher

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The revolution of mobile phones, during the last few years, has transformed them into multi-functional mobile computers known as smartphones. Launched in the 90s, smartphones represent the latest generation of mobile phones and they have extensively evolved. Different definitions of smartphone have been proposed; including:

- "*A smartphone is a mobile phone built on a mobile computing platform, with more advanced computing ability and connectivity than a feature phone* [1][1]".
- "*A category of mobile device that provides advanced capabilities beyond a typical mobile phone. Smartphones run complete operating system software that provides a standardized interface and platform for application developers* [2]".

A smartphone is a device dedicated to mobile communications, using an open operating system, and accepting third-party applications written by developer community. It offers advanced capabilities that you can find only on personal digital assistant[2] or computer but not in previous standard phones. It is based on an operating system that presents platform and interfaces for application developers.

Figure 1.1 shows the principle hardware components of a mobile architecture presented by four blocks: the computer hardware containing different kind of processors, the storage elements, the output/input components and the radio communication module for voice and data exchange.

[1] **Feature Phone**: Mobile phone which is not considered as smartphone.
[2] **PDA:** Personal Digital Assistant, Electronic equipment of pocket used primarily for calendar, phone book and notebook, but technological advances added to it multimedia features.

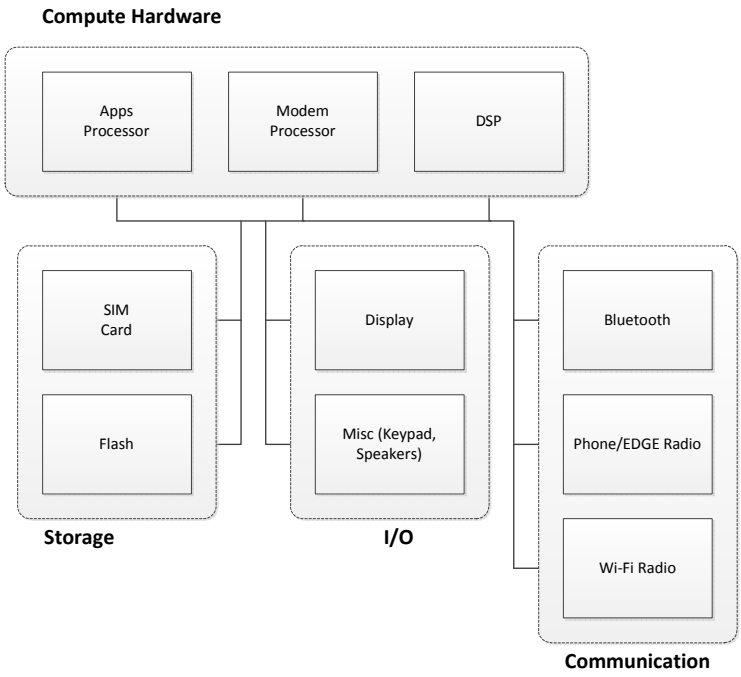


Figure 1.1: High-Level overview of mobile architecture

Smartphone is able to run different software applications and offers functions which include not only calls and messaging but also web browser, GPS navigator, media player, games, network applications, etc. Many other kinds of software and applications can be installed on the smartphone and they give it even greater value. Thanks to these different functionalities, a smartphone has become an important and indispensable tool for people's daily life.

This wide range of functionality proposed by smartphones needs advanced operating systems similar to the ones used by computers. Hence, a revolution of operating systems of mobile phone and more precisely smartphones has taken place, which is the key factor of its popularity. This revolution consists of the fact that smartphone operating system operates under computer operating system software, such as dedicated version of UNIX or Microsoft Windows.

In fact, a mobile Operating System is a software platform specified for mobile devices such as smartphones, tablets, etc and it provides basic functionality for these mobile devices. The mobile OS is responsible to run different programs and applications in mobile devices at once. It supervises the wireless and cellular network connectivity, identifies the functions and features available in the mobile device and it also manages the third-party applications that can be used. Most of mobile Operating Systems are designed around touch-screen input

and offer consistent graphical user interfaces (GUI) to deal with the available applications and better interact with the mobile devices.

So, without its operating system, smartphones are unable to run the multiple applications they offer; and the revolution of OS comes with significant improvements in features, in addition of the innovation of new services. Different kind of operating systems have been announced and used by smartphones, we can mention the iPhone OS[3], Android, Blackberry OS, Windows Phone 7, etc.

1.1. Motivation

In these last years, the Android platform has become the most dominant and the world's most popular mobile platform [3]. Developed by Google, the Android OS based on Linux, is expected to interact and integrate the different existing services offered by Google such as mail service, Google Maps, Google Calendar, Google Talk, etc. Thanks to the market dominance of Android, which is increasing during these recent years, several researchers have concentrated their works on this operating system which has become the most famous and used one. Others are focusing on inventing new applications which help to attract more users.

Despite the opportunities offered by the Android phone, the users can face a big problem with this kind of smartphones. The regular and continuous use of the different functionalities comes with a high cost which is excessive power consumption and battery drain. In fact, the previous generation of mobile phones was mainly used for calling and text messaging perhaps only few times a day due to the expensive cost of communication compared to today. As a result, they could have a standby time of up to 67 days [4], however the android smartphone does not last more than 3 days. By comparing the standby time of the two kinds of mobile phones, we can remark the big gap between them and this gap can be explained by the exaggerated use of resources by the multiple applications running simultaneously on the smartphone. These applications are heavily based on the use of different phone features like GPS, 3G, Wi-Fi, Processors, high definition camera, touch screen, etc, which lead to an elevated power consumption and then short battery life. So, extending smartphones battery life time becomes the main challenge for most of the smartphones developers and the need

[3] **OS:** Operating System

of a complete, objective and effective methodology to avoid the fast drain of the battery becomes critical especially that the users' satisfaction with smartphones is strongly affected by battery performance.

Recent researches have thus focused on studying and understanding power consumption in embedded and mobile architectures and the power estimation and optimization for smartphones becomes an increasingly important and popular field of research.

1.2. Problem and research questions

Larry Page, a co-founder of Google, declared at a conference during the Google I /O [5]:

"Your Android battery life should last a day. If it's not, then blame your apps."

Larry Page declares if the Android phone does not have enough power, this is not the fault of the phone itself but the applications have been installed on it; and then an excessive use of the different components and resources on the Android phone will lead to a high power consumption.

In fact, Smartphones consume the battery power faster than the normal mobile phones and the lack of empowerment come from the fact that the applications are running at the same time in the background of the phone and they are connected to Wi-Fi by default and most of them, after started, remain running and do not stop. However, there are many elements on the smartphone that can reduce the battery life; we mention some of them [6]: Long backlight duration, a high brightness, the excessive use of Bluetooth and Wi-Fi, Playing music, videos and viewing images for a long period, downloading of some applications that consume a lot of energy such as the social networking applications.

As a consequence, these applications will rely on high quality screen, intensive CPU[4] usage, and extensive networking; and the amplified use of these three hardware components increases power consumption persistently, hindering further advances and lead to a short battery life time and even the drain of the smartphone battery.

[4] **CPU:** Central processing unit is the main microprocessor of a computer. It is subject to various parameters such as cadence, frequency, cache. It is responsible for the execution of program instructions and it is the predominant component of a computer configuration.

So, techniques for extending battery life time and optimizing power consumption are needed. Indeed, by putting the CPU in idle mode, disconnecting the phone from the network (by disabling the Wi-Fi or 3G when it is not needed as an example), or reducing screen quality (by adjusting the backlight to be less bright) at times of inactivity, battery consumption can be significantly reduced. By predicting the future inactivity periods, components on the smartphone can be turned off in the right time and then we can avoid extensive switching between on- and off-modus, which is expensive in itself and consumes too much power.

Therefore, techniques for predicting and forecasting the future behavior of the phone based on its historical behavior are therefore needed. Many researchers have focused on machine learning approach to develop power estimation and optimization model for smartphones. They used different learning techniques such as the Markov chain, the Bayesian classification and other methods for estimation and prediction but these methods have not been efficient enough since they do not reach a very high accuracy and the error rate is significant in most of the cases. In fact, most of the methods used by previous researchers are not tolerant to measurement feedback noise [7].

Research Questions

In this paper, we will propose new learning approaches to predict the behavior of the system regarding resource usage and we will present two activity forecasting models one based on the Gaussian Process Regression and the other is based on the KNN [5]Regression. The main question about our research is:

How learning approaches such as KNN and Gaussian Processes can be useful techniques to forecast the behavior of screen activity, CPU usage, and network activity based on historical measurements?

The sub-questions to be investigated are:

RQ1 how efficient can be each approach between the two proposed approaches to perform the prediction and making the right decision about the future behavior of the android phone components?

[5] **KNN:** K-Nearest Neighbors

RQ2 Which of KNN regression and Gaussian processes are the most accurate activity forecasting model?

RQ3 How can an activity forecasting model serve as a basis for battery power management?

1.3. Literature review

Battery power usage is considered as a crucial element for power management systems, one of the reasons why it becomes an important field of research and has been studied through both hardware and operating system in several mobile devices.

Previous works [8, 9, 10 and 11] have been focused only on the power consumption of the CPU component of mobile devices and most of the developed energy models had a relatively high error rate (between 5% and 20%). Those models are thus very limited because they focus on one component of the smartphone which consumes only a part of the total power consumption [12].

Two efficient analytical models are proposed in both [13] and [14]. The model proposed in paper [13] builds a connection between current profile (user's current activities) of the smartphone and battery lifetime. In paper [14], a prediction model is designed to estimate the remaining battery capacity where the recharge cycle aging and temperature are taking into consideration within this model. Another approach proposed in paper [15] which combines analytical[6] methods and statistical[7] methods for the estimation.

The models mentioned above differ from the models what we proposed. In fact, their estimation is based on complete discharge profile during a battery's life which means that they formulate a "calculation" instead of a "prediction".

Based on the work presented in [16], the authors used statistical methods to make the prediction. They estimated the battery lifetime by creating a linear relationship between the system load and the time required to reach a particular voltage.

[6] **The analytical method:** is a procedure used to analyze and interpret collected data while leading an evaluation.

[7] **The statistical method:** is a procedure used to estimate the values of some parameters based on measured data.

In the article [17], Wattch estimated microprocessor power consumption based on the use of low-level architectural features. Other solution proposed by Cignetti in [18] focused on power measurements to derive a power breakdown for Palm devices.

While in paper [19] Tan introduced function-level power models for software implemented power estimation. In [20], SoftWatt studied the power consumption of the parameters processor, memory, and disk by using a simulation model.

1.4. Research approach

Our research approach holds three main phases in order to achieve an effective activity forecasting model for smartphones. The first phase consists of doing a quantitative research through the collection of specific measurements from real smartphones when different applications are running in the devices. In the second phase, we are based on the domain of machine learning to implement two learning approaches (the K-nearest Neighbors and the Gaussian Process). The collected data set serves as historical measures for the two modeled learning approaches to do the forecast and the prediction of the future behavior of the smartphone. The last phase presents an empirical and qualitative study of the proposed learning approaches by making a comparison between the two models in order to find the most accurate one when doing the forecast.

1.5. Key assumptions and limitations

We choose collecting the data from a specific kind of smartphones which is the android phone because the android operating system is now dominating the market and can be found in several brands of mobile phones and it is not specific to a single mark. However, the solution that is presented in the thesis can be generalized and applied to any type of smartphones.

In addition, the collection of measures is done through many users since the battery power consumption depend on real user activity patterns which differ from one user to another. It is implicit that when collecting the data, users are using their smartphones in a regular way in order to conduct a research with real scenarios.

The proposed solution can be helpful and interesting for real life usage since it can be used as a basis to conduct studies and researches concerning the optimization of smartphones battery life. The activity forecasting models can serve as first step or foundation for researchers who are focusing on the domain of smartphones' power management.

Despite the presented assumptions, some limitations can be presented in the work being done. Due to the fixed time plan of the thesis project that we are limited with, we collect data from a limited number of users (only 6 users) to achieve our study. In the other hand, the measures collected are specified for 3 types of components of the smartphone. These components are the CPU, the screen and the network. We choose to focus on these components since we conclude from the researches already done that these three components are the most power consuming components for smartphones, other elements will not be involved during our work.

1.6. Contribution to knowledge

Differently from the previous researches which are concentrated on studying the activity of one component of the smartphone (in most cases), our work consists of collecting measurements from different components which are the CPU, Screen and components of network activities since these elements are known by their large power consumption. These measurements will be taken when different types of applications are running in the smartphone.

The solution that we propose is based on a learning approach and it differs from the previous solutions by its precision at making prediction and decision. We are interested during our work in two kinds of machine learning and the goal is to make a comparison between these two models in order to find the most accurate one that will help to predict the future behavior of the android phone. The first model is based on the Gaussian process which is known by its tolerance to measurement feedback noise, the second one is based on the KNN which is a rapid and incremental learning method and it is very easy to understand. The two models that we focus on are not used in previous works although several previous studies have been concentrated on other types of machine learning.

The proposed models are expected to provide an overview of the CPU usage, the network activity (by capturing the received and transmitted bytes during a connection) and the screen utilization, and then predict their behavior in the future. By predicting the future behavior of these different components in different scenarios running on the smartphone, we can help energy optimization mechanisms to reduce the power consumption and to achieve energy efficiency.

1.7. Thesis outline

This thesis report is organized as follows. In chapter 2, we introduce the theoretical background. We present the android smartphone and we describe battery problems. We reveal the most power consuming components on which we concentrate our thesis work and lastly we define the two learning approaches that are used to forecast the behavior of these components.

We present our solution in chapter 3. We explain the method used for the collection of the data and we give a detailed explanation of the KNN regression and the Gaussian process regression, the two models used for the prediction.

In chapter 4, we present our empirical results. We show here how the KNN regression model and Gaussian Process Regression model are used in different scenarios to forecast the future behavior of the CPU, the screen and the network activity.

We discuss and evaluate our results in chapter 5. We present the accuracy of the forecasts for the two learning models and make a comparison between them. The research questions presented in the introduction chapter (section 1.2) are investigated in this part.

And finally we draw the conclusion in chapter 6. We give a concise summary of the solution, the major findings and their implications and we provide points for future work and improvements.

Chapter 2

Technological overview

In this chapter, we will present a general overview about the android smartphones. We will illustrate reasons of making the screen, the CPU and the network as the most power consuming components in smartphones, and finally, we will give a theoretical introduction to the learning approaches, in which is based our work, which are the K-nearest neighbors and the Gaussian Processes.

2.1. Android smartphones

Android is an open source operating system based on the Linux kernel for smartphones, PDAs and mobile devices. Android is purchased by Google in August 2005 and was officially announced on November 15, 2007 [21]. It is specialized in developing mobile applications. Different versions of Android have been known since the release of version 1.1 up to version 4.0. (See Appendix B)

Advantages of the android OS

Unlike the operating system iOS[8] (iPhone OS) Which only runs on the iPhone smartphones, the Android operating system is available on mobile phones from various manufacturers such as HTC, Samsung, Sony Ericsson, Motorola, etc.

The opportunity to integrate an operating system which is powerful, free and can enrich of third party applications has opened the way for several projects. In fact, the Android phone is multitasking; it can run different types of applications at the same time. A user is able, for example, to browse the net, see Facebook notifications and listen to music. Moreover, the

[8] **iOS:** iPhone Operating System

user can access the android market and install a lot of applications even for free. Until April 2011, the number of applications compatible with the android operating system has reached 250 000 applications [22], in addition to the other services offered by Google such as Gmail, Google calendar, Google talk, etc.

The drawbacks of the android OS

The advantages presented above can be seen at the same time as inconvenient for the smartphone battery. In fact, the applications which are running on the Android phone drain the battery and most of the users of the android OS complain against Android wasteful batteries. This problem can also be due to continuous internet connection since most of the applications need internet connection to run and this consumes too much power of the smartphone battery.

Battery problems

A main problem for smartphones is thus excessive power consumption. We propose a definition for the term “Power Consumption” withdrawn from the “Webster’s Online Dictionary” [23] and presented as follow:

“In electrical engineering, power consumption refers to the electrical energy over time that must be supplied to an electrical device to maintain its operation.”

Mobile devices pull the power needed for their functioning from batteries. In the case of smartphones, the capacity of the battery is extremely limited due to constraints on the size and weight of the unit, which implies that the power efficiency of these mobile phones is very critical for their usability [24]. In addition, the functionalities of smartphones are increasing rapidly and have negative effect on the battery life time and can cause its drain in a very short time. So, we can say that the development of batteries for smartphones did not achieve a high revolution comparing to the rapid revolution of smartphones; and the problem is the energy offered by the battery is not proportionate to the needs of different functions suggested by the smartphone.

Use of HTC Sensation phone

During our work on this thesis, we used the smartphone HTC Sensation for taking measurements from the different components of this Android phone and forecasting the future behavior of the system according to resource usage. The different properties of the HTC Sensation phone will be presented in the following table.

Table 2.1: Characteristics of the HTC sensation components

Component	Description
HTC Sensation	We used the version 3.0
Operating System	It is based on the Android OS version 2.3.4
Battery	Standard battery, Li-Ion [9]1520 mAh.
CPU	Processor 1.2 GHz Dual-core.
Display	3.4 inch with QHD[10] resolution, 16M colors and multi-touch option.
3G Network	mobile network type HSDPA[11]
WLAN	Wi-Fi 802.11 b/g/n
Other components	It includes also GPS[12], Sensors (Accelerometer, gyro, proximity, compass), camera, Bluetooth, etc. (See [25])

2.2. The most power-consuming hardware components

Most of the researches done according to the power consumption in mobile phones and especially smartphones have agreed that there are three principle elements that consume power more than the other components. The screen, the CPU and the network activities are

[9] **Li-Ion:** an electrochemical accumulator that uses the lithium in an ionic form.
[10] **QHD:** quarter of full HD (High Definition).
[11] **HSDPA:** High Speed Downlink Packet Access connects 3G mobile phones to the Internet with a maximum speed of 2 Mb/s.
[12] **GPS:** Global Positioning System is a Satellite geolocation system.

the largest power consuming components in smartphones. The other phone components have small influence on power consumption.

Screen or display: We can say that the screen is the first element that consumes too much power since it is the primary output element with which the user always interacts. It shows the patterns of user activity. In addition, with the invention of the touch-screen technology which has been integrated in most of the smartphones (by 2012, 40% of mobile phones will use the touch-screen technology [26]); the screen becomes more power consuming than before. In fact, any contact with the surface of the screen triggers its activity and then maximize its energy consumption.

In another hand, the display backlights have negative effects on the battery life time and the power consumption is depending on the size of the screen and if it is colorful or not. In fact, a big screen size consumes more power than a small one and a screen displaying a lot of colors (red, blue, green, etc) also consumes more than one based on deep colors (like black, grey, etc). Moreover, turning the screen On and Off many times causes a vast battery power usage. So, when brightness of backlights increases, more power is wasted by the smartphone [27].

CPU: The power consumption of the CPU depends on the CPU usage and the frequency used by the CPU. For example, having multiple applications running at the same time in the smartphone utilizes a lot the CPU and then the power consumed by CPU will increase. In addition, using a very high frequency by the CPU is also more consuming than using a lower one.

In fact, the excessive CPU power consumption is due to execution of some instructions and their fetching. If the portion of code that the system should fetch from the caches or memories is small then the consumed power will be less [27]. In addition, the frequent switching between idle mode and active mode by the CPU is also a reason of high power consumption.

Network activities: The last generation of mobile phones includes many types of wireless communication technologies such as Wi-Fi, 3G, Bluetooth, etc; and these new technologies offers new services and more multimedia applications which are growing power

consumption to extremely high values. One of the most important reasons that makes network activities one of the most power-consuming components is that when opening or starting a connection in a smartphone, the user does not close or stop it even if he/she does not use it. In addition, when we are connected to a network and working with it, the amount of received and sent data increase and then the power consumption will increase too.

2.3. Machine learning

Machine learning is known as a subfield of the artificial intelligence (AI[13]). It provides computer algorithms that automatically learn perhaps without human help or intervention. The main goals of a learning approach can be to make prediction accurately, achieve tasks or to behave in a smart way. It is based on observations or a set of data which provides experience about making the right decision in the future [28].

In a formal way, learning approach can be presented as follow: Learn from an experience $\mathbf{E}$ by respecting some tasks $\mathbf{T}$ and some performance measures $\mathbf{P}$ so that the performance $\mathbf{P}$ on task $\mathbf{T}$ is improved with the experience $\mathbf{E}$ [29].

Figure 2.1 presents the execution of machine learning. At first, the training set is used by one of the learning algorithms such as Bayes Naive, KNN, Support vector machines (SVM), Genetic programming, Linear Regression, etc. The output of these algorithms will be a hypothesis $\mathbf{h}$. This hypothesis creates new input $\mathbf{x}$ and gives out an estimated output $\mathbf{y}$ or a class. The estimated output $\mathbf{y}$ presents a prediction on some of the unseen new input $\mathbf{x}$ and we say that the hypothesis parameters are learned [29].

[13] **AI:** Artificial Intelligence: is a science whose purpose is to make tasks by machine that humans perform using their intelligence.

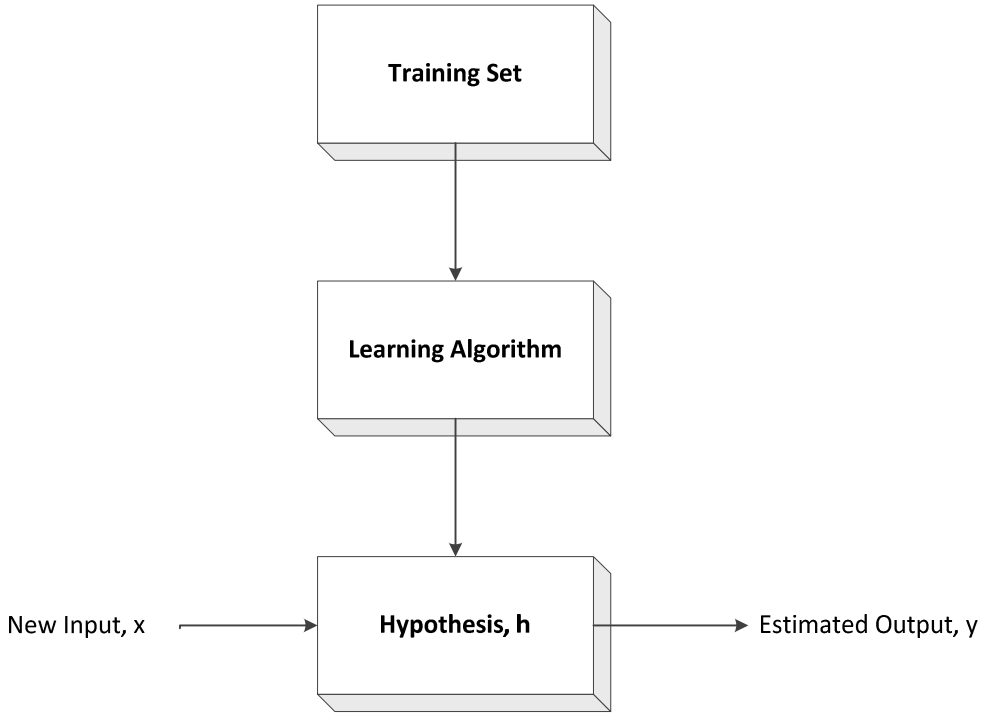


Figure 2.1: Machine Learning

We can distinguish two types of machine learning: the supervised and the unsupervised machine learning.

The supervised machine learning enables a computer program to complete a task based on a concise guidance. The learning algorithms used in this kind of machine learning present the final product or some examples of the process and can give feedback over the training process (for example label for data points in the case of classification). The supervised learning algorithm should make the right prediction of the output values from the input objects based on a classifier or regression function.

The unsupervised machine learning, unlike the supervised learning, does not give any guidance that can help to complete the task and does not give any feedback over the training process. This leads the computer program to create a procedure to cluster similar data and there is no clear semantics about this cluster even if we find the right one for the data points.

The goal of machine learning is to develop efficient algorithms for general purpose of practical value. The output of these algorithms should be accurate prediction rules that are easily interpretable by human experts. The principle advantage of using machine learning is

that the results are more accurate than the ones presented by direct programming and it also can treat a large amount of data [28].

A learning approach often uses pattern recognition which is necessary to analyze and find the meaning of a sample of data. Many steps are presented to achieve pattern recognition. The most important ones are [30]:

- Collect data by taking the necessary measurements.
- Present the data in an understood form (charts, graphs...) by plotting the measures.
- Extract features from the data such as mean, standard deviation, variance, etc. These features are obtained by a specific transformation of the collected data.
- Analyze the results and make a conclusion to the study.
- Apply appropriate procedures such as regression or discrimination based on training set of exemplar patterns.
- Make an interpretation to the results.

During our work, we are based on learning approaches to solve the problem of battery power usage. The benefit of using a learning approach is it is based on statistical components that rapidly analyze, in a dynamic way, the changing workloads and operating conditions. In fact, our proposed solution consists on modeling two learning approaches (KNN for regression and Gaussian process regression) and making a comparison between them. The more appropriate one should be characterized by its rapidity since we would like to make a lifetime forecast and then choosing the right real-time decision about the future behavior of the system. It should also be power-efficient so that it can be able to run on the battery-powered device and it should be dynamic by the fact that it considers user workloads, and environmental operating conditions.

2.4. K-nearest neighbors approach

The method of k-nearest neighbor is a supervised learning method. It is also appointed as lazy learning (or instance-based learning (IBL) or memory-based learning). The k-nearest neighbors' algorithm is one of the simplest learning algorithms. The K nearest neighbors' algorithm is represented by a set of objects called examples (or instances) for which results are known. Each example is represented by an observation consists of a set of independent

values to which correspond a set of dependent outcomes. The dependent and independent variables can be continuous or categorical. In the case of continuous dependent variables, we are dealing with a regression problem, in the other case we are dealing with a classification problem [31].

Classification An object is classified by examining its neighbors and it is attributed to the class that its k nearest neighbors belong to (where k is an integer to be chosen). Let's consider that we have a database of elements of a known class (We talk here about learning base). As soon as we receive a new element that we want to classify, we calculate its distance to all elements of the base. If this base has 100 elements for example, we calculate 100 distances. If k = 25, we look for the 25 smaller numbers of those 100 numbers. These 25 numbers correspond to the 25 elements of the database that are the closest to the item we want to classify. So, we decide to assign to the item that we want to classify the majority class among the 25 elements. The number k can be varied depending on what you want to do.

Regression: The method presented above can be used in the case of regression. The difference is that the property value of the object must be assigned to the average values of its k nearest neighbors. This method allows to the nearer neighbors to more contribute to the average than those which are further. The neighbors used for the classification or the regression will present the training set for the KNN algorithm [32].

The choice of the number K: The choice of K is important to create the model of K Nearest Neighbors. In fact, K is one of the most important parameters of the model, and can have big effect in the quality of forecasts. The number of nearest neighbors K can be seen as a smoothing parameter. For a given problem, a low value of K can cause a large variance in the forecast. Instead, if a high value is assigned to K, this can introduce a significant bias in the model. Therefore, the value of K must be large enough to minimize the probability of misclassification but also reasonably low (relative to the number of observations in the sample examples) so that the K nearest points are sufficiently close to the query point [31].

Distance Metric: For a given query point, the method of K Nearest Neighbors will achieve its forecasts from the outcome of K nearest neighbors of this point. Therefore, to achieve the prediction by the K Nearest Neighbors, we must first define some metric to measure

distance between the query point and the observations from the sample examples. Different measures can be used in the K Nearest Neighbors approach, we mention some of them which are most used: Euclidean, Euclidean squared, City-block, and Chebyshev [31].

$$D(x, p) = \begin{cases} \sqrt{(x-p)^2} & \text{Euclidean} \\ (x-p)^2 & \text{Euclidean Squared} \\ Abs\ (x\text{-}p) & \text{City-block} \\ Max\ (|x\text{-}p|) & \text{Chebyshev} \end{cases} \quad (2.1)$$

The main advantage of the KNN algorithm is its simplicity and its rapidity for learning. It is a very easy to understand method, suitable to parallelization and it has an asymptotic performance close to the optimum. On the other side, this learning approach is known by its slow prediction and its sensitivity to irrelevant and correlated attributes.

2.5. Multiple Regression

A regression is defined by a statistical measure, presented by a formulated equation, between a dependent variable and one or more independent variables having parametric coefficients that help to predict future values of the dependent variable. In fact, the regression is characterized by the prediction of continuous values (contrary to classification which treats discrete values) and it has an important role in pattern recognition. There are two types of regression: linear regression and multiple regression [38].

Linear regression is a statistical technique to estimate the linear relationship between two variables (one dependent variable and one independent variable). It uses only one independent variable for prediction of the outcome. The principle is to calculate the equation of the line, passing through the point cloud formed by the pairs of values of two variables, which minimizes the squared distances between points and the line. This kind of regression is used in many domains likely due to its simplicity and good interpretability.

Multiple regression is a generalization of the linear regression model, it uses many independent variables for prediction of the outcome. The method of multiple linear regressions allows:

- The analysis of the relationship between a dependent quantitative variable and several independent quantitative variables.
- The determination of the equations of a nonlinear polynomial fit for the analysis of the relationship between the two kinds of quantitative variables.
- The determination of the polynomial equations of a geometric correction model applicable to vectors and/or data.

The starting point is the estimation of parameters of regression involving an endogenous variable $\mathbf{Y}$ and $\mathbf{p}$ exogenous variables $\mathbf{X_j}$. When we have $\mathbf{n}$ observations, the regression equation is written:

$$y_i = a_0 + a_1 x_{i,1} + \cdots + a_p x_{i,p} + \varepsilon_i \quad (2.2)$$

Where $\mathbf{y_i}$ is the i-th observation of the variable $\mathbf{Y}$, $\mathbf{x_{i,j}}$ is the i-th observation of the j-th variable and $\boldsymbol{\varepsilon_i}$ is the model error called noise, it summarizes the missing information that allow to linearly explain the values of $\mathbf{Y}$ with the $\mathbf{p}$ variables $\mathbf{X_j}$. We need to estimate $\mathbf{(p+1)}$ parameters, by taking matrix writing:

$$Y = Xa + \varepsilon \quad (2.3)$$

The dimensions of matrices are respectively:

$$Y \rightarrow (n, 1)$$

$$X \rightarrow (n, p+1) \quad (2.4)$$

$$a \rightarrow (p+1, 1)$$

$$\varepsilon \rightarrow (n, 1)$$

The matrix $\mathbf{X}$ of size $\mathbf{(n, p+1)}$ contains the set of observations of exogenous, with the first column formed by the value 1 to indicate that we integrate the constant $\mathbf{a_0}$ in the equation.

$$X = \begin{pmatrix} 1 & x_{1,1} \cdots & x_{1,p} \\ \vdots & \ddots & \vdots \\ 1 & x_{n,1} \cdots & x_{n,p} \end{pmatrix} \quad (2.5)$$

2.6. Gaussian Processes

Over the past decade, it has been an increasing researches done in machine-learning that have been concentrated in the Gaussian processes (GP) which become more popular and sophisticated learning approach. The Gaussian process can be viewed as a generalization of the Gaussian distribution. In fact, the Gaussian distribution was characterized by a mean and a variance however the Gaussian process is specified by a mean function $\mathbf{m(x)}$ and covariance function $\mathbf{k(x, x')}$. The mean of a function is defined by the average value of the function over its domain [33] and the covariance measures the joint change between two random variables around their respective mean [34]. We say that the Gaussian process is over functions and we denote it:

$$f \sim GP(m, k) \tag{2.6}$$

Figure 2.2 presents a comparison between the Gaussian distribution and the Gaussian process. By applying input values $\mathbf{x}$ and time step $\mathbf{t}$ to the Gaussian process, we will have a density function $\mathbf{p}$. The covariance function will be helpful for solving regression since it presents a-priori knowledge which is the training data [35].

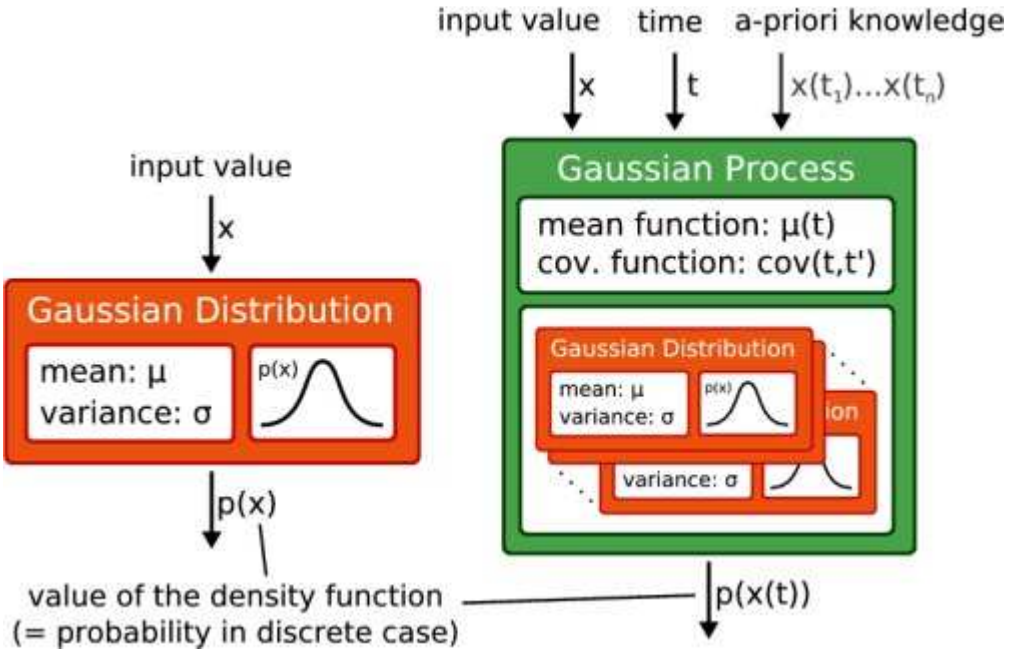


Figure 2.2: Gaussian distribution vs. Gaussian process [35]

A formal definition of Gaussian process is presented in paper [36] as follow:

"A Gaussian process generates data located throughout some domain such that any finite subset of the range follows a multivariate Gaussian distribution".

Gaussian process is a stochastic process that consists of random values associated with every point in a range of times where each random variable is presented by a normal distribution. Each random variable is indexed by its position $\mathbf{i}$ in the vector of the Gaussian distribution. In the Gaussian process, each input $\mathbf{x}$ is associated to a stochastic random function $\mathbf{f(x)}$ and it is the input $\mathbf{x}$ which has the role of the index set [37].

The importance of Gaussian processes appears in the fact that its properties are inherited from the normal distribution. For instance, a random process is considered as a Gaussian process if the distributions of derived quantities can be achieved explicitly. The principle advantage of Gaussian process is it shows efficient and sophisticated view with computational tractability. In addition, its mean and covariance functions are known by their simplicity of working with them to have a best estimation. GP offers a probabilistic learning approach within kernel machines. It is considered as a powerful technique for constructing comprehensive probabilistic models of real world problems within Machine-learning. It can be practical and used to supervised, unsupervised, reinforcement learning, and principal component analysis optimization.

Gaussian Process Regression (GPR) is known as nonparametric statistical method for functional regression analysis. It is concerned with the prediction of continuous quantities and presents methods which are based in Gaussian process [39]. GPR can be seen as the best tool to perform a wide range of applications very well. For this reason, it can be considered as a popular probabilistic regression method. GPR can be able to fit arbitrary-shaped curves thanks to its nonparametric nature. Furthermore, GPR is free from pathological behavior for regions where there are few data points, unlike relevance vector machines, which present another model for probabilistic nonparametric regression.

Chapter 3

Solution

In this chapter, we will talk about the tools used for the collection of the data from the Android smartphones and how this data will help as historical measurements to forecast the future behavior of the CPU, the screen and the network. We will give a detailed description of our solution by presenting the KNN Regression and the Gaussian Process Regression approaches used for the forecast.

3.1. Data collection

To collect measures from the Android phone, we used an application that has been released on the Android market called *Nu JamLogger*[14]. *Nu JamLogger* is an application which was developed by a group of researchers of the Northwestern University[15]. By installing the application in an android device, it will be possible to collect real data when user is working with the smartphone.

The main goal of this application is to log user activity and system performance. The data is collected on a periodic basis (each 1 second) and by clicking the button "*Send log to SD card*" presented on the GUI of the application (see Appendix C), a log file containing detailed information and measures about the phone usage (Appendix C) will be saved on the SD card[16] of the phone. The *Nu JamLogger* application allows us to collect data concerning the

[14] https://play.google.com/store/apps/details?id=edu.northwestern.jamlogger_r2&hl=no
[15] http://www.ece.northwestern.edu/microarchitecture/jamlogger/
[16] The original NU JamLogger application does not save the log file to the SD card; it only uploads the log file to the server of the Northwestern University. By making some modifications to the application, it becomes possible to send the log file to the SD card of the smartphone.

CPU utilization, the system load, network traffic (received and sent bytes and packets), display brightness and state (on or off), battery level, etc[17].

The collection of data through the *Nu JamLogger* application is done each 1 second during 3 hours. During these 3 hours, the battery of the Android phone should not be powered by electric current which means that the battery is not charging in order to better study the usage patterns when the smartphone is battery-constrained and then it will have no influence on the power consumption. In the other hand, since we are studying real user activity patterns, the user should use his/her smartphone in a normal way during the collection of the data. For example, the user can make calls, browse the web, play games or music, access social network such as Facebook or Twitter, etc.

By taking a look to the log file obtained from the *Nu JamLogger* application, we can remark that it contains a lot of detailed information that is not very easy to understand. So, to solve this problem, we tried to parse[18] this log file and extract the measures needed for our work concerning the CPU usage, the network activity and the screen utilization. The extracted information was presented in a simple and easy to understand way (Appendix C).

CPU_utilization			
time	total	system	user
14:25:05	12.39%	7.43%	4.96%
14:25:06	16.00%	11.00%	5.00%
14:25:07	19.00%	11.00%	8.00%
14:25:08	33.00%	25.00%	8.00%
14:25:09	18.00%	6.00%	12.00%
14:25:10	31.00%	14.00%	17.00%
14:25:11	6.00%	2.00%	4.00%
14:25:12	17.00%	9.00%	8.00%
14:25:13	9.00%	5.00%	4.00%
14:25:14	9.00%	3.00%	6.00%
14:25:15	12.00%	6.00%	6.00%
14:25:16	12.00%	6.00%	6.00%
14:25:17	15.00%	10.00%	5.00%
14:25:18	7.00%	6.00%	1.00%
14:25:19	11.00%	2.00%	9.00%
14:25:20	33.00%	23.00%	10.00%
14:25:21	49.00%	29.00%	20.00%
14:25:22	13.00%	11.00%	2.00%
14:25:23	5.00%	2.00%	3.00%
14:25:24	30.00%	16.00%	14.00%
14:25:25	13.00%	4.00%	9.00%

Figure 3.1: Example of measures of the CPU usage extracted from a log file

[17] No personal information will be presented by the Nu Jamlogger application.
[18] We create a java code to parse the log file. The code extracts information related to each component that we focus on and save them in a separate excel file. So, the output will be 3 excel files, one for the CPU, another for the screen and another one for the network.

Since the power consumption is very influenced by the usage behavior of the user and the power breakdown among hardware components of the Android phone can vary dramatically from one user to another, we suggest collecting traces from more than one user. So we gather data from 6 users[19] and we study their activity patterns in order to understand the power consumption of Android phone architectures in real environments.

The following figures show how the usage of the different components of the smartphone depends on the users' activities. In fact, figures 3.2 and 3.3 present the average CPU usage breakdown and the average screen utilization breakdown (during 100 seconds) for each user. Figures 3.4 and 3.5 present the number of received and transmitted bytes of the network activity for each user (during 100 seconds). We can observe that the resource usage is not the same for the 6 users, it varies from one user to another and it depends on the applications utilized by each user during the collection of measures.

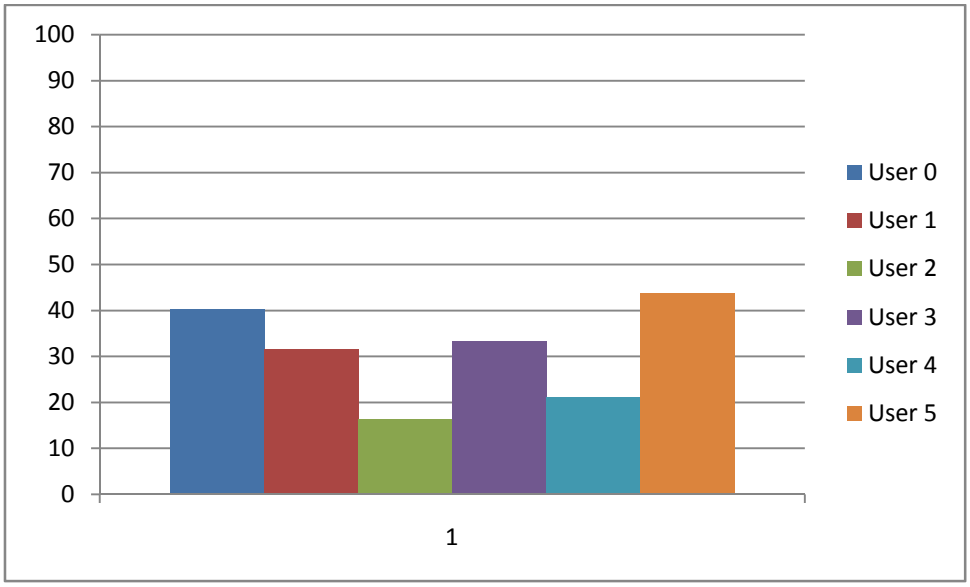


Figure 3.2: Average CPU Usage (%) breakdown from real user traces

[19] The 5 users are employees from the ST-Ericsson company that voluntarily install the Nu JamLogger application on their Android phones and send the log files to us.

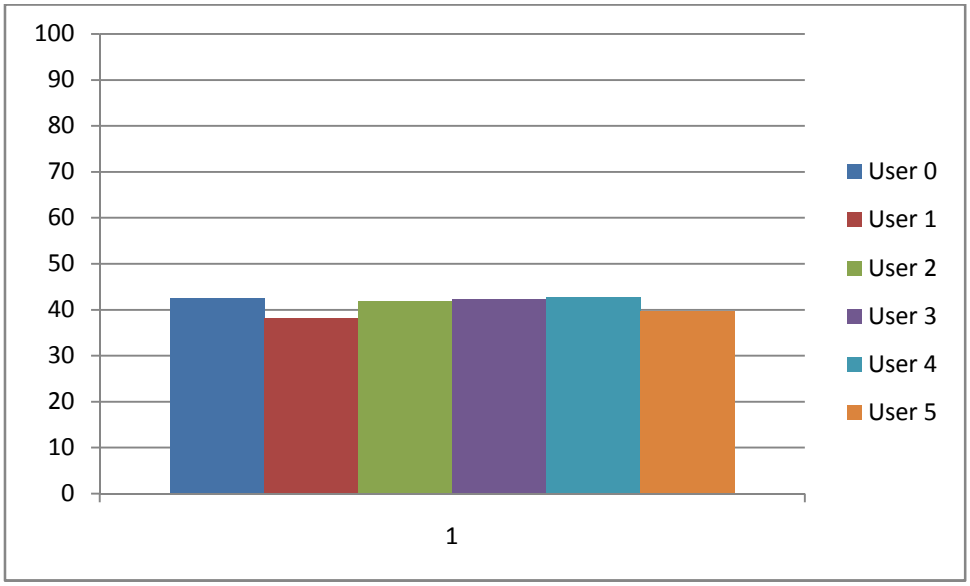


Figure 3.3: Average screen utilization (%) breakdown from real user traces

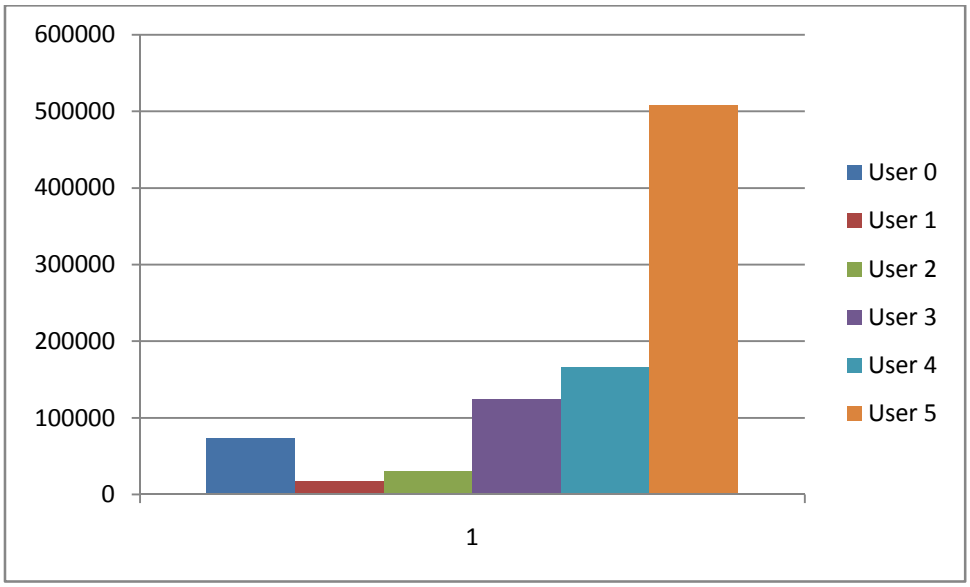


Figure 3.4: Sum of received bytes of the network for each user during 100 seconds

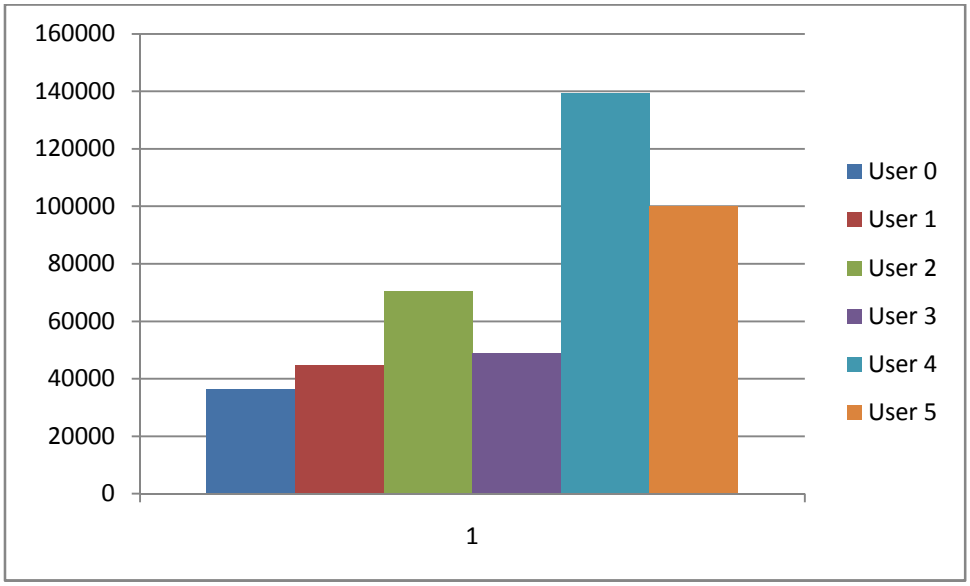


Figure 3.5: Sum of transmitted bytes of the network for each user during 100 seconds

The collected data will serve as historical measurements (or training set) to forecast the future behavior of screen activity, CPU usage, and network activity by using the learning approaches the KNN regression and the Gaussian process regression.

3.2. The K nearest neighbors regression model

We applied the learning approach 1-NN[20] regression to our data already collected. As we mentioned in the previous chapter(section 2.5), the nearest neighbor method is a nonparametric method in which a new observation is classified in the class membership of the observation of the training sample which is the closest one , in terms of covariates used. The determination of their similarity is based on distance measurements. We consider $\mathbf{L}$ as our data set or training simple.

$$L = \{(y_i, x_i), i = 1, \dots, n_L\}$$

Where $y_i \in \{1, \dots, c\}$ denotes the class of the individual $\mathbf{i}$ and the vector $x_i = (x_{i1}, \dots, x_{ip})$ represents the predictor variables of individual $\mathbf{i}$.

[20] We choose K=1 since the measures collected previously vary drastically and it may be hard to do the regression with more than 1 nearest neighbor.

In order to determine the nearest neighbor, we used the function of the Euclidean distance. For a given vector $\mathbf{x}$ and each compound in the training set with individual vector $\mathbf{x_i}$, the Euclidean distance is calculated as follow:

$$D = \sqrt{\|x - x_i\|^2} \tag{3.1}$$

The forecasts of K Nearest Neighbors are calculated as the average of the K nearest neighbors outcome:

$$y = \frac{1}{K}\sum_{i=1}^{K} y_i \tag{3.2}$$

Where $\mathbf{y_i}$ is the ith observation of the examples sample and $\mathbf{y}$ is the prediction (outcome) of the query point.

The piece of code presented below shows the implementation done for the forecast[21]. The output of this code is show in the appendix C.

```
def k_nearest_neighbours(k, history, target_x):
    distance_history = []
    for (x, y) in history:
        distance = math.sqrt((x - target_x)**2) #application of the Euclidean distance
        distance_history.append((distance, x, y))
    distance_history.sort()
    y_sum = 0.0
    for (distance, x, y) in distance_history[0:k]:
        y_sum += y;
    return y_sum/k      #average of the KNN outcome
```

We calculate also the standard error of the prediction by using the following formula:

$$error = \sqrt{\frac{1}{k-1}\sum_{i=1}^{k}(y - y_i)^2} \tag{3.3}$$

[21] We used the Python language to implement our 1-NN regression model.

Finally, we plotted[22] the collected values and the predicted ones at the same graph to see how efficient is the work done by the KNN regression approach and to analyze the obtained results (Chapter 4).

3.3. Gaussian process regression model

To do the implementation of our GPR model, we are referred to the work done by Carl Edward Rasmussen and Chris Williams [39]. Our model was implemented in python language and in this section; we present an explanation of the model.

Let's consider that our historical collected values are associated to attributes $x_i \in \mathbb{R}^D$ and an observation$y_i \in \mathbb{R}$. The observation $\mathbf{y_i}$ is presented by the following equation:

$$y_i = f + \epsilon_i \tag{3.4}$$

Where $\epsilon_i \sim N(0, \sigma^2)$ is the Gaussian noise presented by a normal distribution with 0 mean and variance σ^2 and $f \sim N(0, K)$ is an underlying function presented by a normal distribution with 0 mean and $\mathbf{K}$ covariance (where $\mathbf{K}$ is the covariance matrix). We assumed that the mean function of the Gaussian process is zero so that the GP is completely based on its covariance function. The covariance function that we used in our implementation is a squared exponential covariance function with isotropic distance measure presented by the following equation:

$$K(x_i, x_j) = k^2 \, exp\left(-\frac{\rho^2}{2} \sum_d^D (x_{i,d} - x_{j,d})^2\right) \tag{3.5}$$

Where $\mathbf{k}$ and $\boldsymbol{\rho}$ are the hyperparameters of the covariance function and $\mathbf{x_{i,d}}$ is the d-th dimension of the vector $\mathbf{x_{i.}}$

In fact, the covariance is presented by two hyperparameters: the length scale characterizing the smoothness of the GP and the standard deviation of the noise. Since the GPR is a linear smoother [40], these hyperparameters will help to control the degree of smoothness. This

[22] We used the free library "matplotlib", this library is very used in Python to draw graphics in two and three dimensions.

degree of smoothness called also degree of freedom of the smoother is defined as follow [41]: ($\mathbf{I}$ is the identity matrix)

$$tr(K(K+\sigma^2 I)^{-1}) \tag{3.6}$$

The aim of using the smoothness is to reduce the noise of the signal where there is a large change in the amplitude from one point to another. So, if one point is very higher or lower than its immediately adjacent points, it will be reduced or increased respectively and the signal will be smoother.

By having Gaussian likelihood and Gaussian prior, the marginal likelihood will be Gaussian too and presented by:

$$p(y|\theta) = N(0, K + I\sigma^2) \tag{3.7}$$

Where **θ** presents the hyperparameters of the covariance function.

By applying Rasmussen and Williams' conjugate gradient maximization [39] of the log marginal likelihood, optimization was done to the hyperparameters. In addition, some log-transformation was done to the observation set. The dataset was divided into 1/2 training set and 1/2 test set. By using the test set, we calculated measures of evaluation to get a good and efficient picture of the predictive performance of trained models. The following measures were also taken into consideration during the implementation of our GPR.

- *Root Mean Squared Error:*

$$RMSE = \sqrt{\frac{1}{m}\sum_i (y_i - f_i)^2} \tag{3.8}$$

- *Mean Absolute Error:*

$$MAE = \frac{1}{m}\sum_i |y_i - f_i| \tag{3.9}$$

- *Negative Log Predictive Density:*

$$NLPD = -\log P(y) = \frac{1}{m}\sum_i \left(\frac{1}{2}\log(2\pi{\sigma_i}^2) + \frac{(y_i - f_i)^2}{2{\sigma_i}^2}\right) \tag{3.10}$$

As we have done with the KNN regression model, we plotted the collected values and the estimated ones at the same graph to see how efficient our model is to predict the future behavior of the system (the results will be shown in the next chapter). We plot also:

$$\bar{y} \pm 1.96\sqrt{var(y)} \tag{3.11}$$

It presents 95% confidence interval obtained by applying the Gaussian Process Regression where the standard Gaussian noise is presented when the hyperparameters are chosen to maximize the marginal likelihood.

Chapter 4

Empirical Results

In this chapter, we will present the results achieved by applying the two learning models the 1-NN regression and the Gaussian process regression. The graphs that will be presented in the following sections will show the behavior patterns of the 3 studied components of the smartphone in a time interval of 100 seconds[23]. In each following section, we will give a detailed explanation of the results for one user. The results for the other 5 users will be presented in the appendices D and E.

4.1. Results of the 1-NN regression model

To forecast the behavior of the 3 more power consuming components of the smartphone which are the CPU usage, the screen utilization and the network activity (presented by its received and transmitted bytes), we are based on one historical measurement. In fact, we tried to do three experiments, the first one by doing the prediction based on the exact previous value of the one that we want to predict (that's mean after 1 second), the second and the third tests are done by using the historical collected measures to do the prediction of the future behavior of the next 5 seconds and the next 10 seconds respectively.

To make our work easier, we arrange our data points in three matrices related to the three experiments. The matrices have the following presentation (N presents the number of the training points):

[23] For good legibility and understanding of the behavior of the system and to better show the forecast, we chose a time interval 100 seconds to present our results in the graphs.

$$\begin{pmatrix} Y_1 & Y_2 \\ Y_2 & Y_3 \\ Y_3 & Y_4 \\ \vdots & \\ Y_{N-1} & Y_N \end{pmatrix} \quad \begin{pmatrix} Y_1 & Y_5 \\ Y_2 & Y_6 \\ Y_3 & Y_7 \\ \vdots & \\ Y_{N-4} & Y_N \end{pmatrix} \quad \begin{pmatrix} Y_1 & Y_{10} \\ Y_2 & Y_{11} \\ Y_3 & Y_{12} \\ \vdots & \\ Y_{N-9} & Y_N \end{pmatrix}$$

The first column presents the historical values that have been used to make the prediction of the values presented in the second column of the matrix.

4.1.1. Scenario 1: Results based on 1 second historical measures

In this section, we present the forecast done for the behavior of the three components, in which we concentrated our work, based on the immediate previous historical values.

Figure 4.1 shows the forecast of the behavior of the CPU utilization (in %) for one user. The forecast presented in this figure is done during 100 seconds. The blue line presents the plot of our collected data points and the red one shows the predicted values based on one historical measurement (the same legend will be used for the screen utilization and the network activity). The error rate of the CPU utilization is equal to 3.26 which is relatively high.

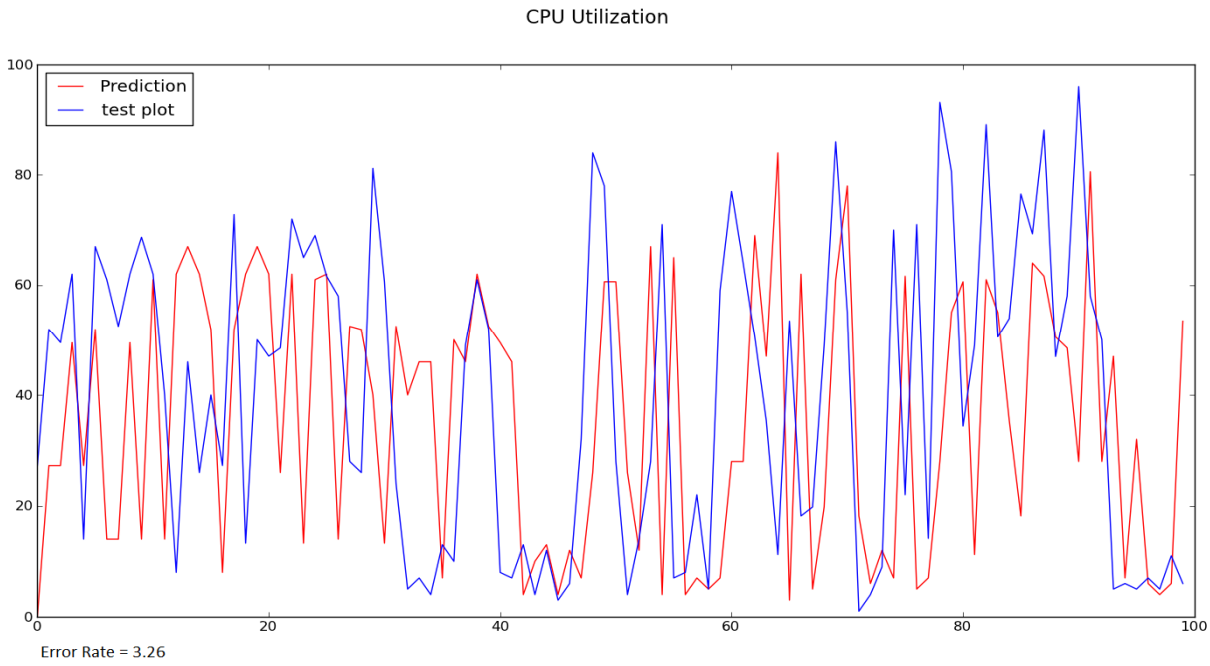


Figure 4.1: Prediction of CPU usage behavior during 100 seconds

Figure 4.2 presents the forecast of the behavior of the screen utilization (in %) for the same user in 100 seconds time interval.

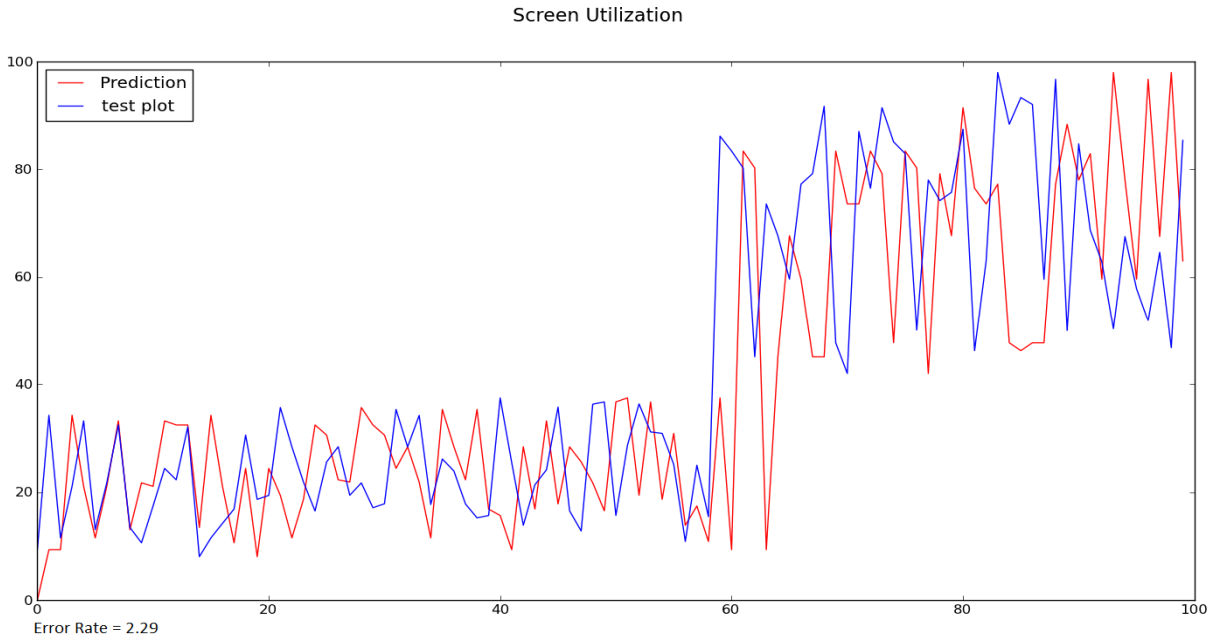


Figure 4.2: Prediction of the screen utilization behavior during 100 seconds

We can remark that the percentage of the use of the screen is increasing from the second 60 which means that the user was using his Android phone with multiple applications and the

display was continuously on during this time and it is using a very high backlight. The error rate of the screen utilization for this user is equal to 2.29.

The two following figures present the received and transmitted bytes of the network during 100 seconds (for the same user). The errors of the received and transmitted bytes were respectively 6.1 and 8.15.

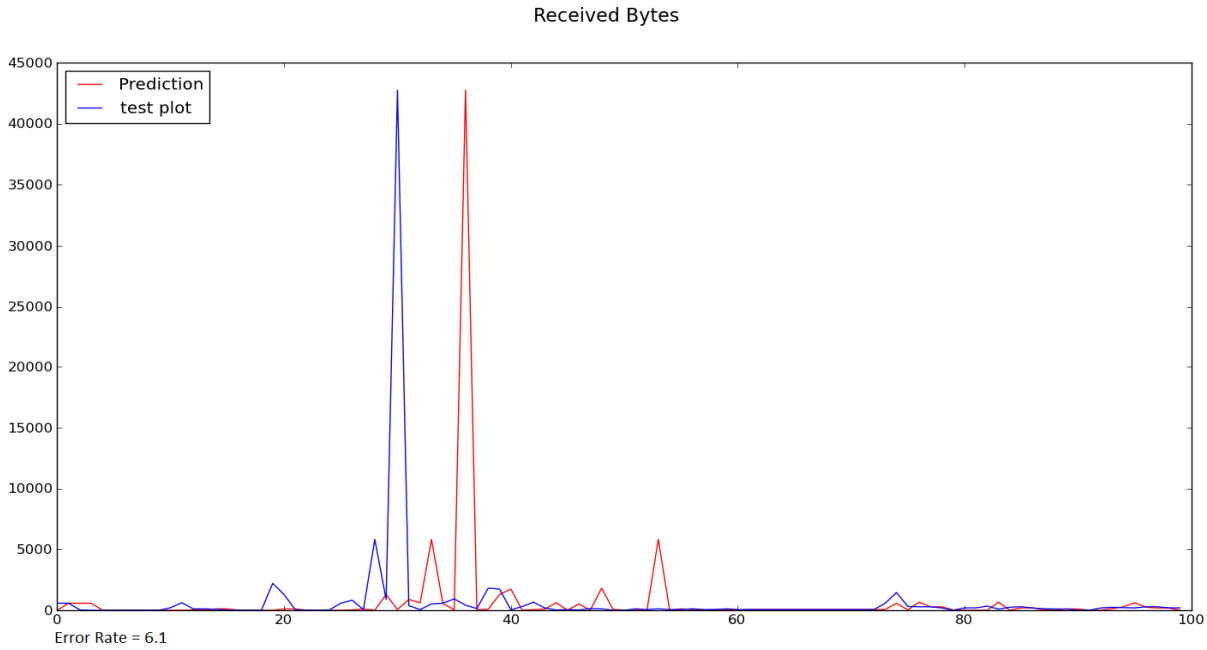


Figure 4.3: Prediction of the behavior of received bytes during 100 seconds

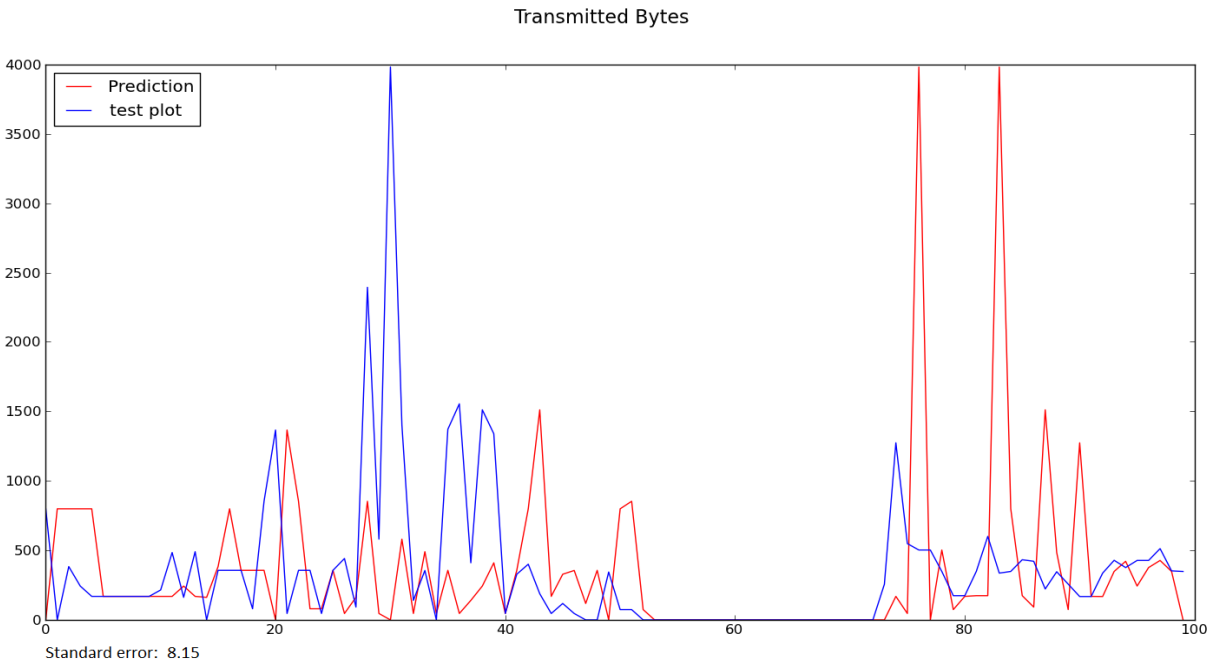


Figure 4.4: Prediction of the behavior of transmitted bytes during 100 seconds

The graphs present some picks showing that the network is used in this time interval. We can remark also that the error rate of the network activity is higher comparing with the ones found by the CPU usage and the screen utilization.

4.1.2. Scenario 2: Results based on 5 second historical measures

In this part, we use the data points collected to predict the behavior of the system after 5 seconds. Figure 4.5 presents the forecast of the CPU usage. We can see that the error rate here is equal to 13.78%. So, the error becomes higher when we want to predict the behavior of the CPU for the next 5 seconds.

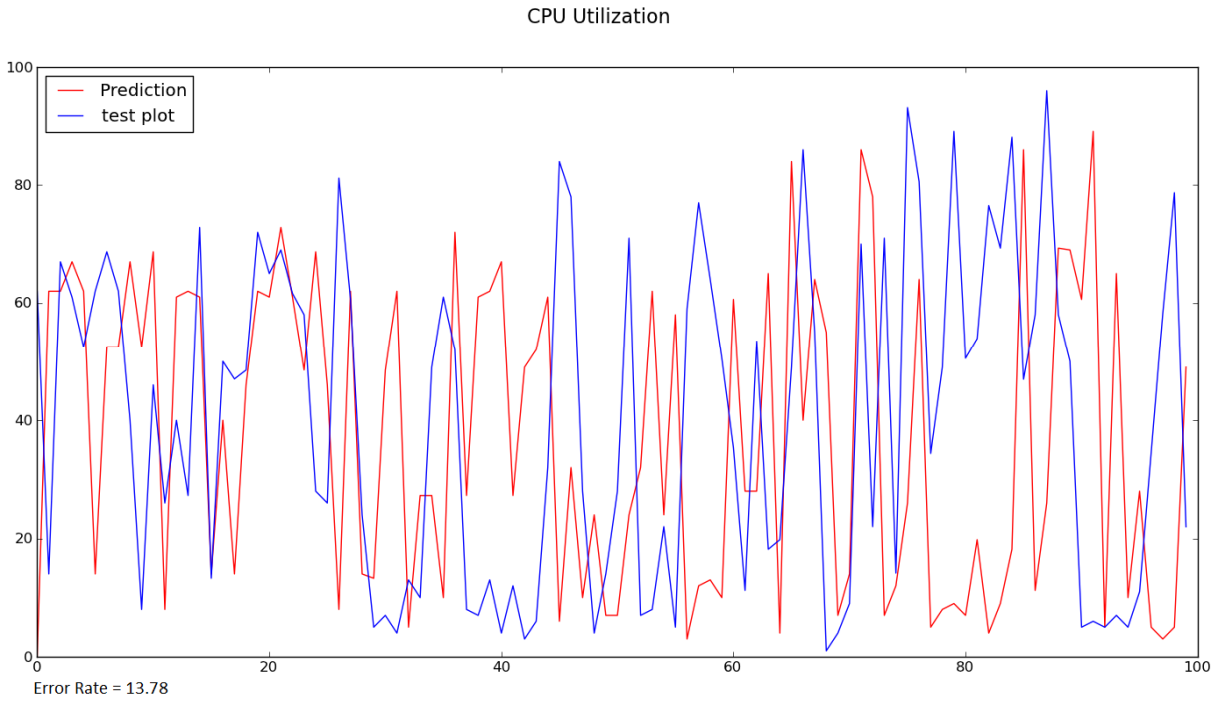


Figure 4.5: Forecast of the CPU usage

Figure 4.6 shows the screen utilization, its error rate is equal to 9.41.

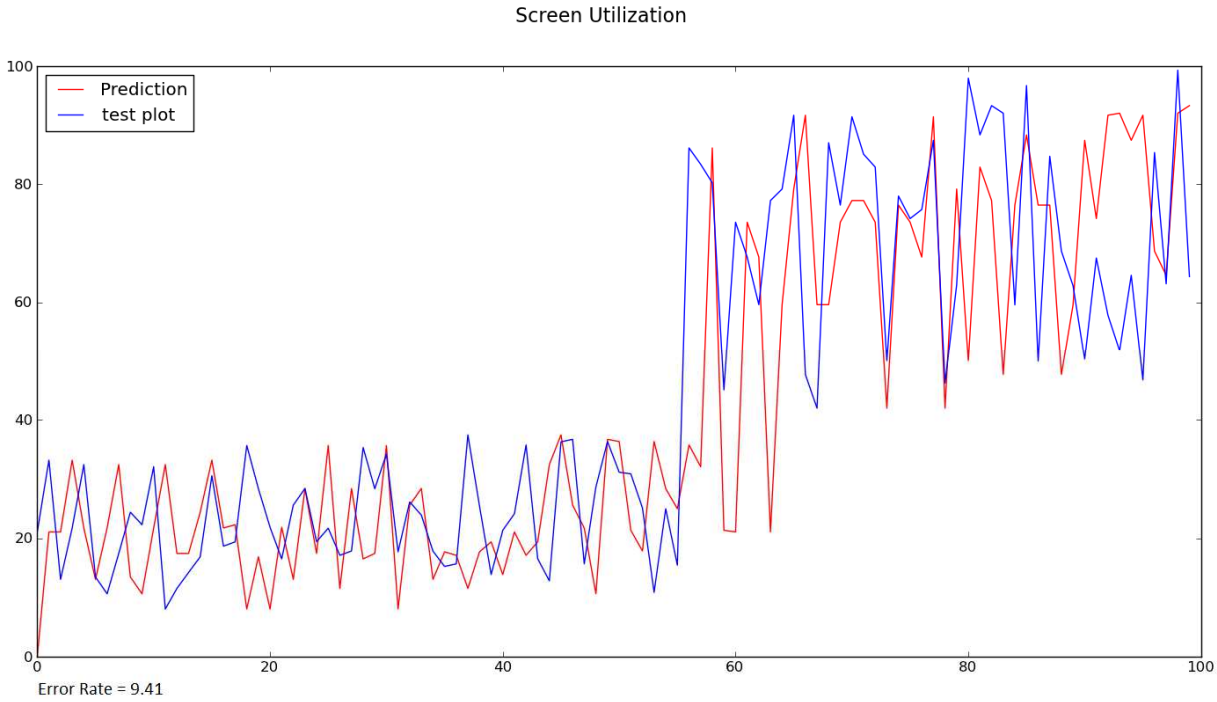


Figure 4.6: Forecast of the screen utilization

Figures 4.7 and 4.8 present the forecast of the received and transmitted bytes of the network by using the same method for the prediction. Their respective error rates are 14.6 and 15.13.

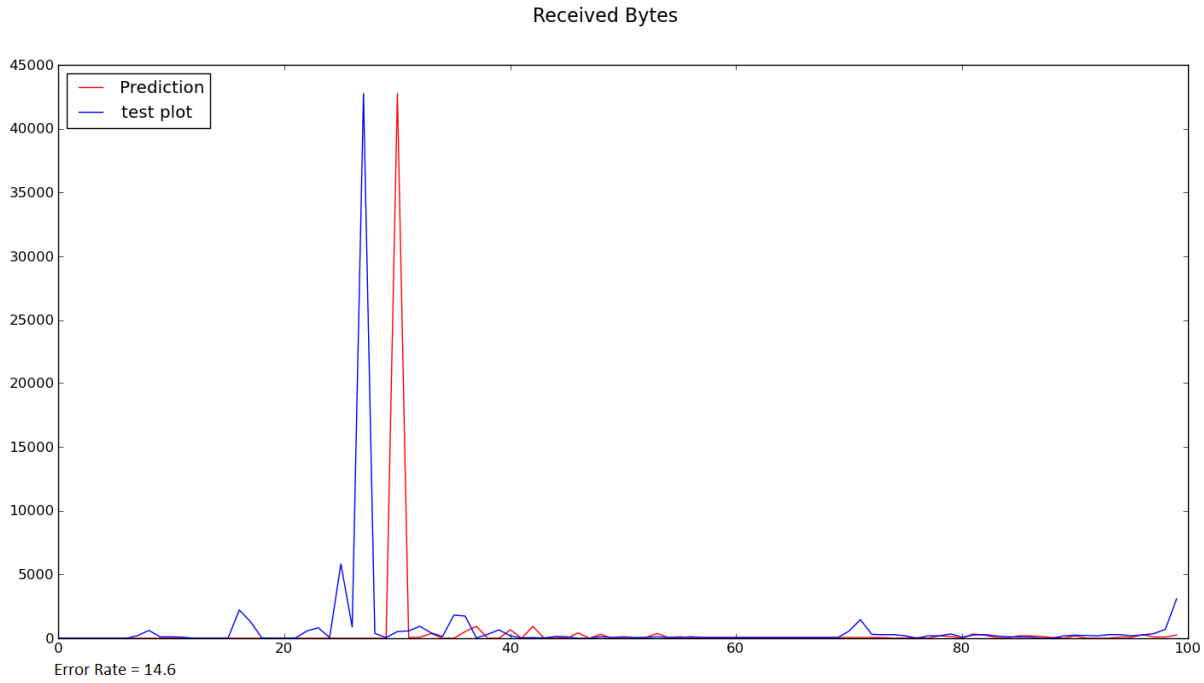


Figure 4.7: Forecast of the received bytes of the network

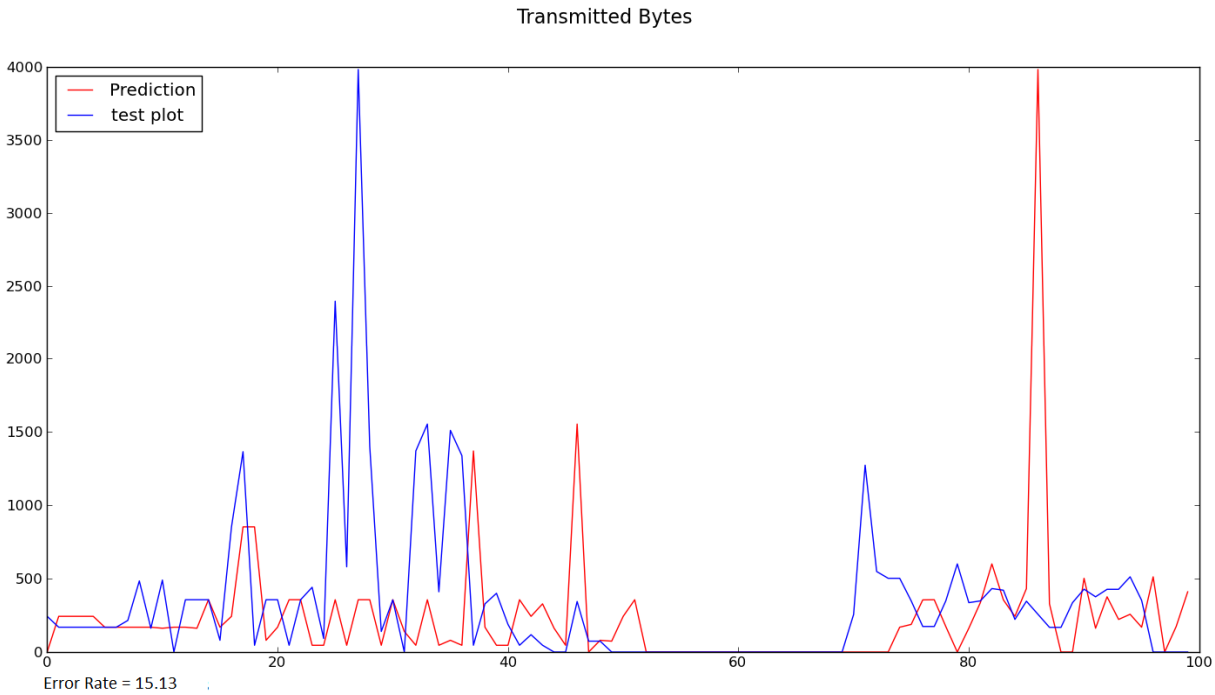


Figure 4.8: Forecast of the transmitted bytes of the network

4.1.3. Scenario 3: Results based on 10 second historical measures

The same work is done in this section to forecast the behavior of the CPU usage, the screen utilization and the network activity but this time we used the collected data to forecast the values of the next 10 seconds. The error rate of each component can be seen in the graphs below.

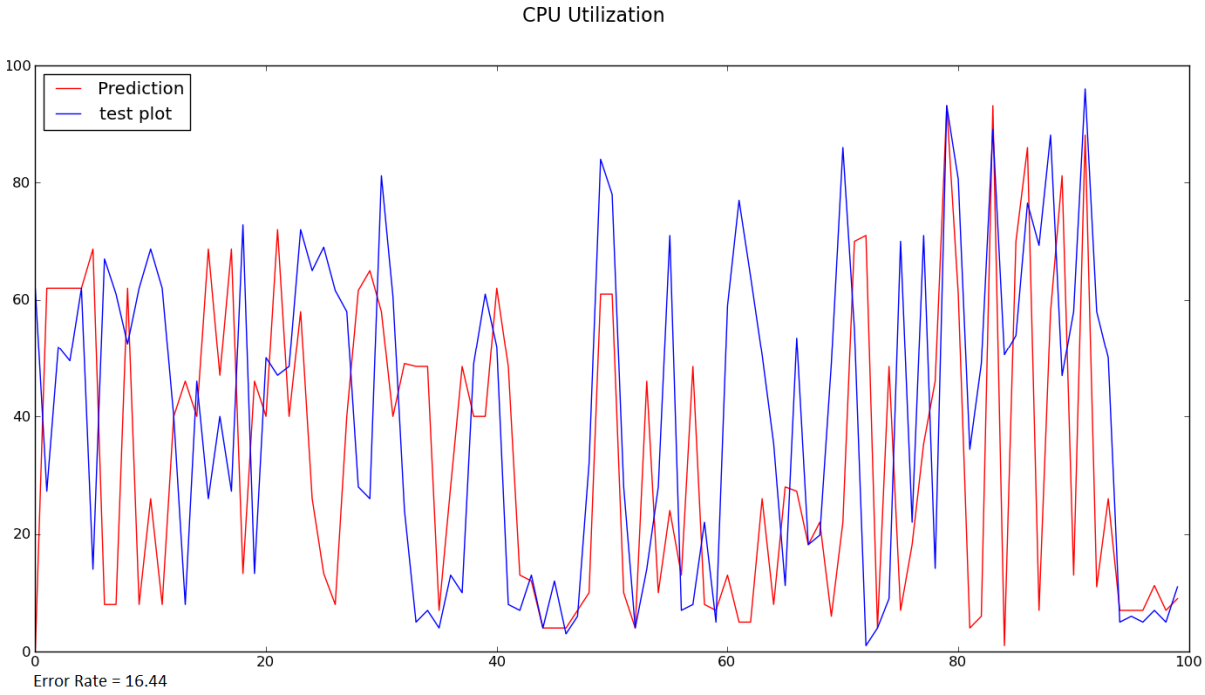


Figure 4.9: Prediction of the CPU usage based on 10 seconds historical values

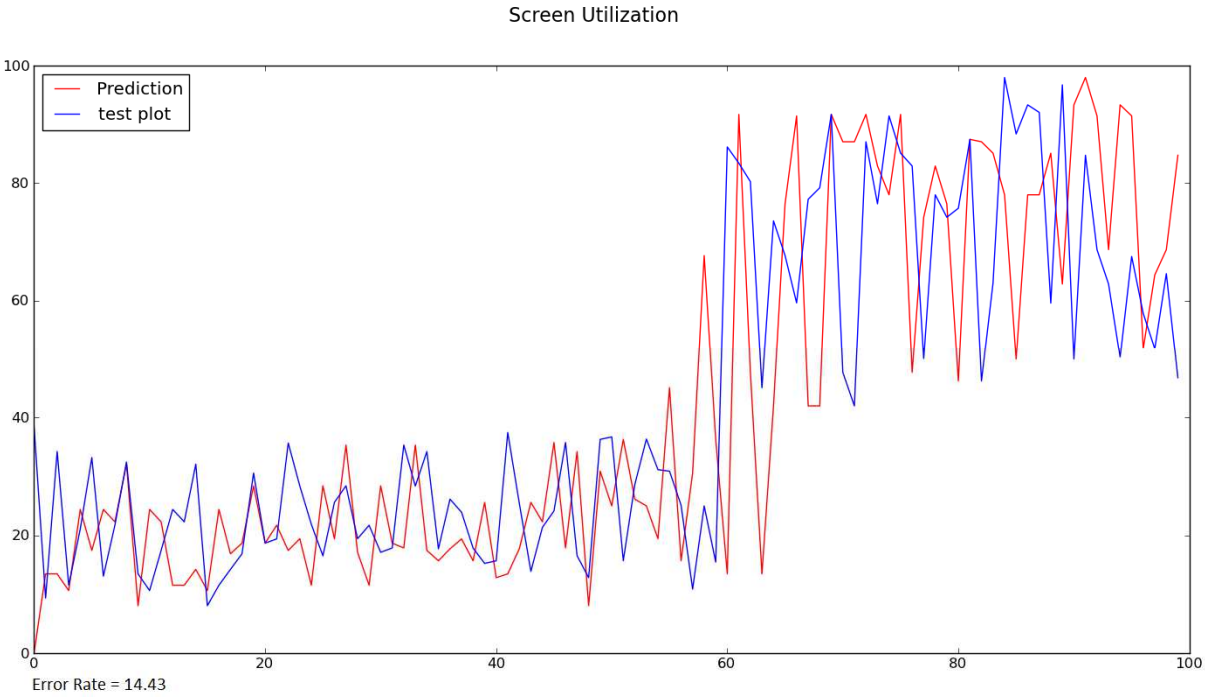


Figure 4.10: Prediction of the screen utilization based on 10 seconds historical values

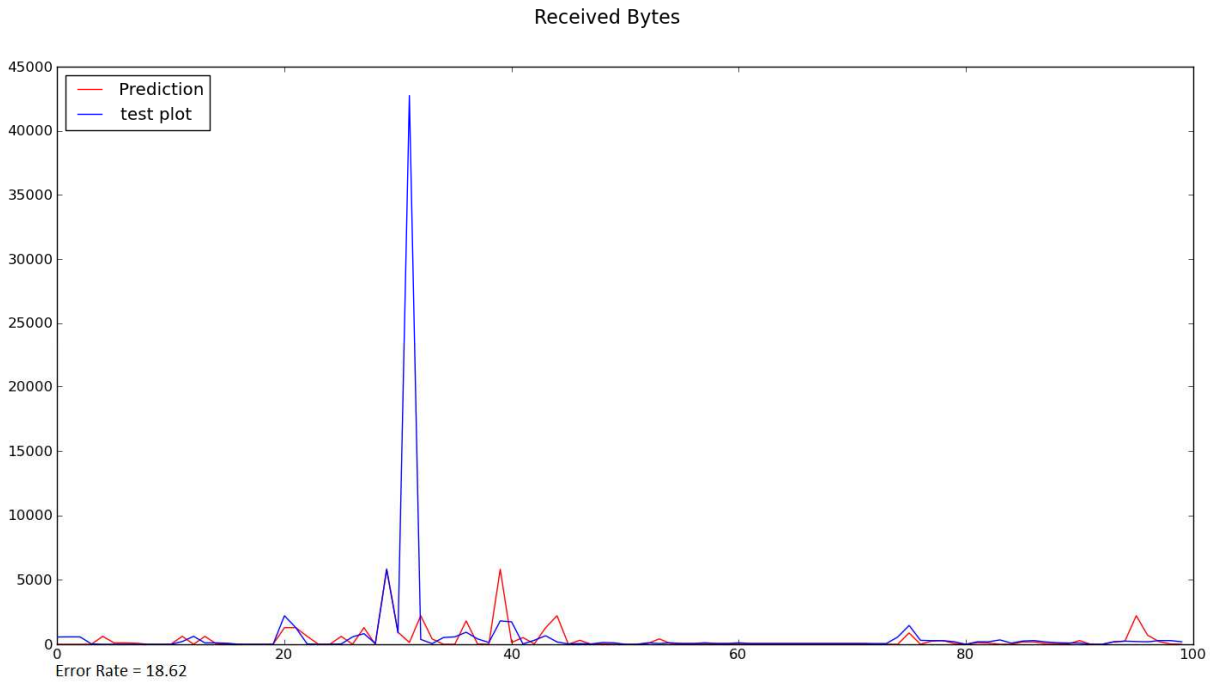


Figure 4.11: Prediction of the received bytes based on 10 seconds historical values

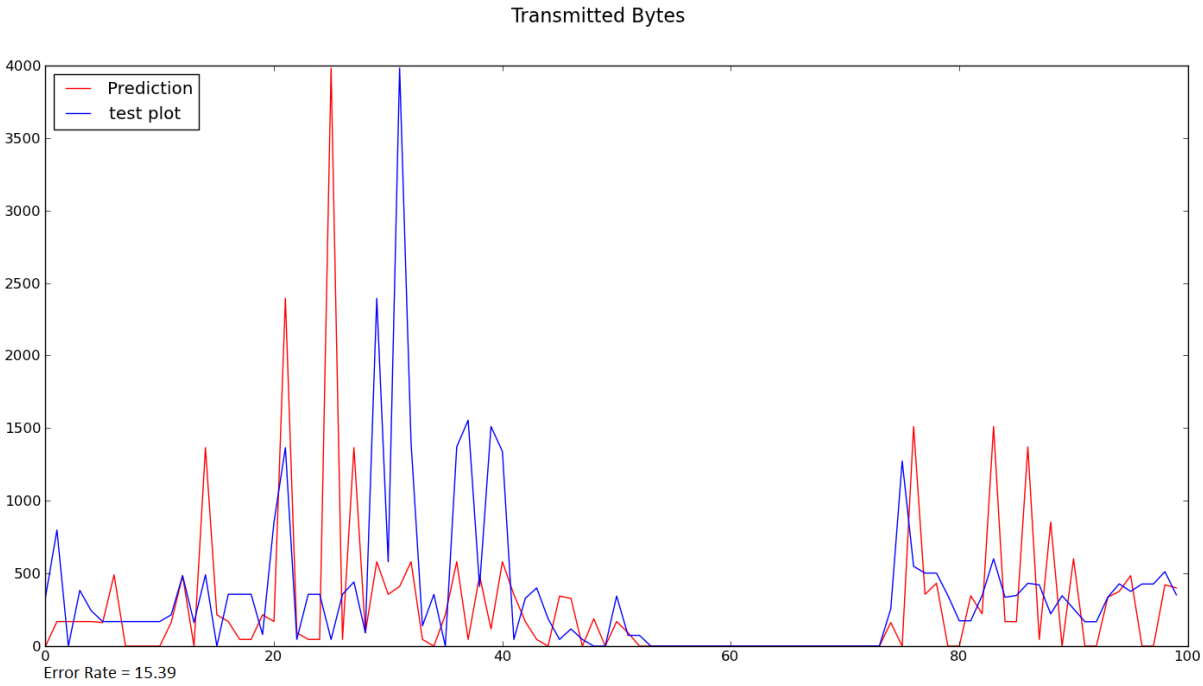


Figure 4.12: Prediction of the transmitted bytes based on 10 seconds historical values

A general conclusion can be done is that by doing the forecast based on the KNN regression model, the error rate will be more important if we will use the historical values to forecast the next 5 seconds and 10 seconds values than to forecast the exact next values (1 second) and then the accuracy will deteriorate.

4.2. Results of the Gaussian process regression model

To predict the future behavior of the system based on the historical measures taken from the smartphone, we follow the same methods described in the section 4.1 above.

We will present in this part the results for one user concerning the CPU usage, the screen utilization and the network activity. The results of the other users can be found in the appendix E.

4.2.1. Scenario 1: Results based on 1 second historical measures

As we have done with the KNN regression model, we are based on the exact previous measures to predict the next points.

Figure 4.13 presents the CPU usage (in %), the blue points characterize the plot of our collected data during 100 seconds and the red line characterizes the plot of the prediction values when applying the Gaussian process regression model. The error rate of the forecast of the CPU usage for this user is equal to 1.09 %.

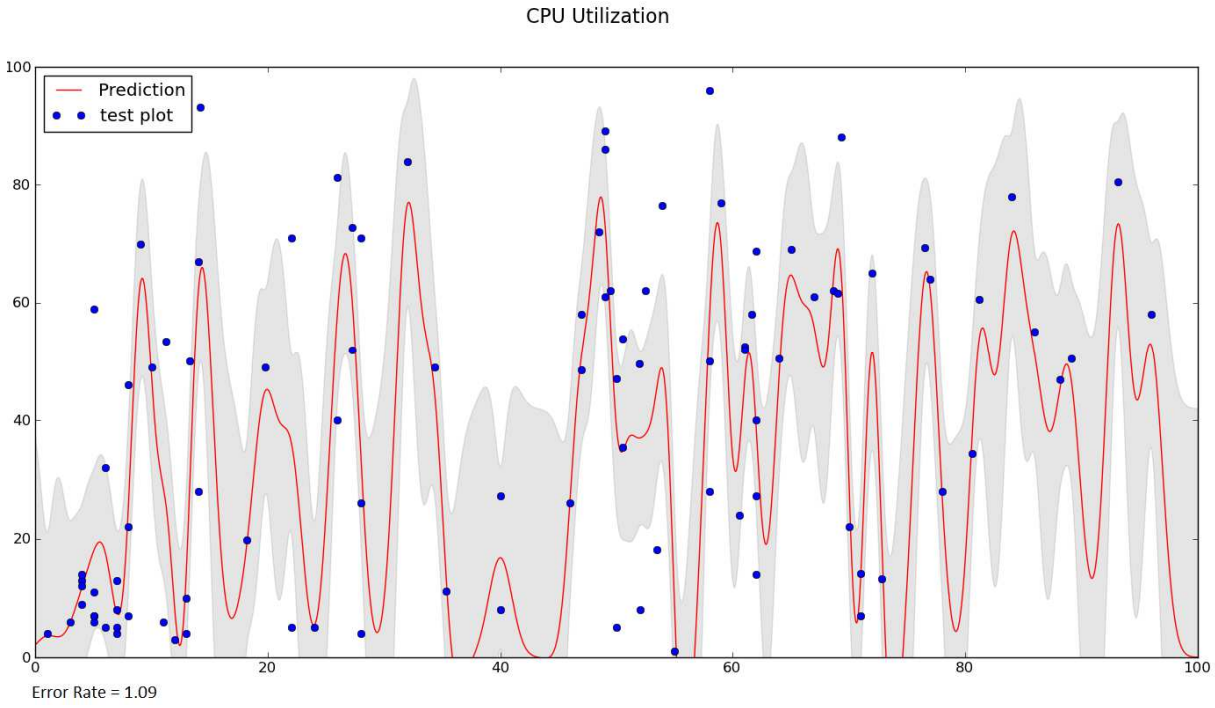


Figure 4.13: Forecast of the CPU usage based on exact previous historical values

The grey bands represent 95% confidence interval. The graph above corresponds to optimizing the marginal likelihood of the Gaussian process but we can remark that the signal is noisy, so we tried to reduce the noise by making the hyperparameter “length scale” of the covariance function longer and then the prediction function will be smoother. Figure 4.14 shows how the forecast of the CPU usage becomes smoother.

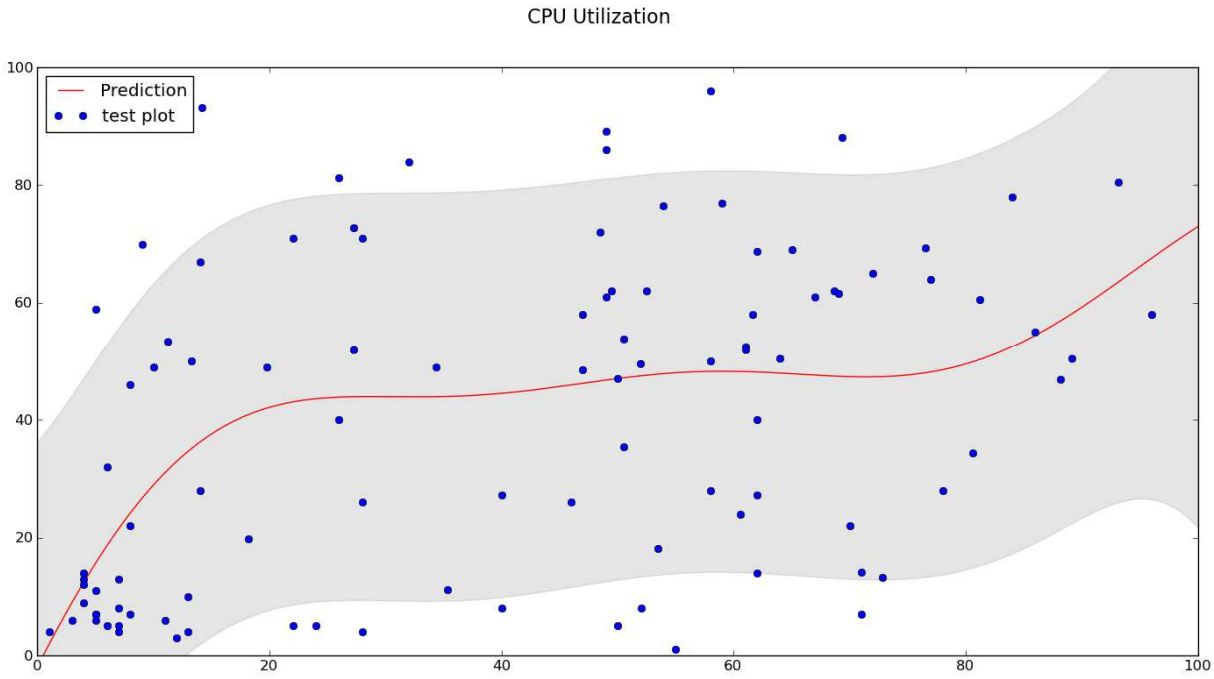


Figure 4.14: Smoother forecast of the future behavior of the CPU

Figure 4.15 shows the screen utilization behavior for the same user. Its error rate is equal to 1.21%; figure 4.16 presents the prediction of the screen utilization when reducing the noise.

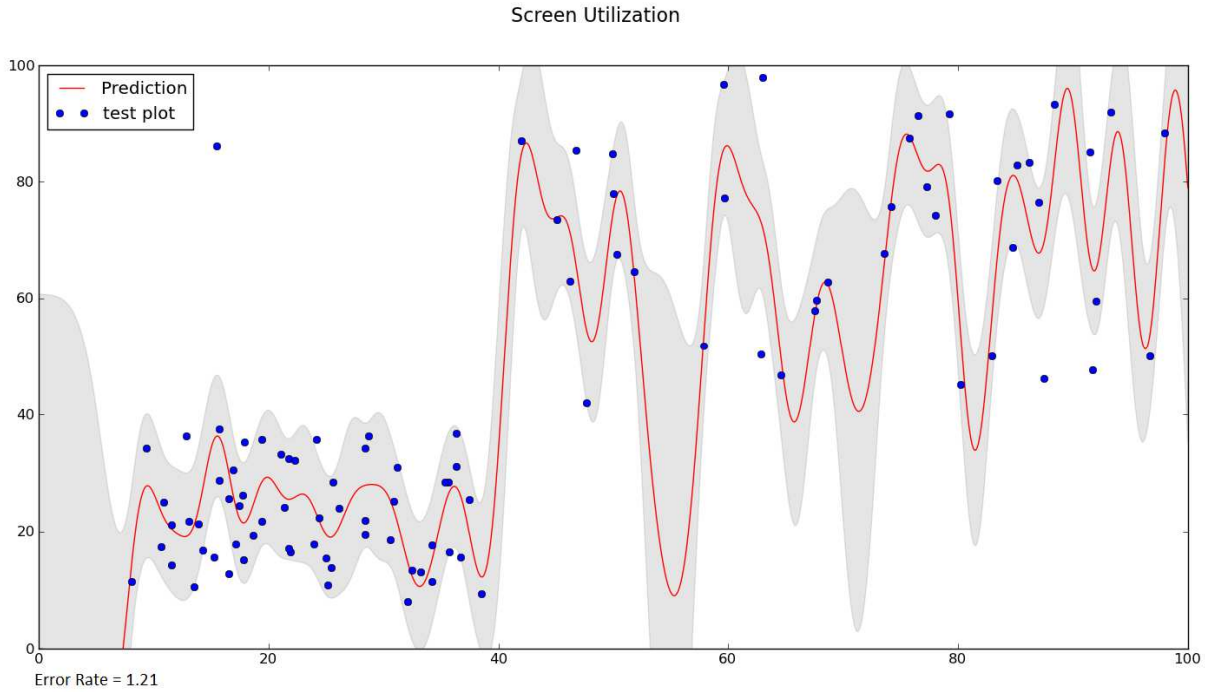


Figure 4.15: Forecast of the screen utilization based on the exact previous historical values

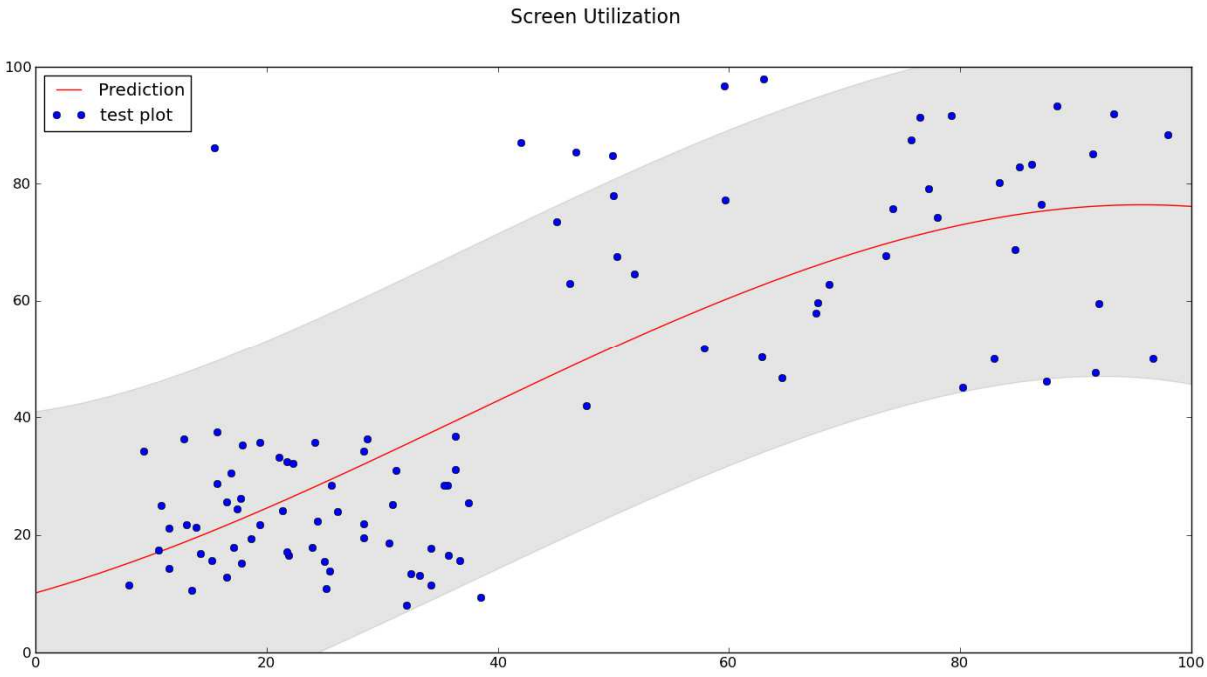


Figure 4.16: Smoother forecast of the future behavior of the screen

The two figures bellows present the received and transmitted bytes of the network in 100 seconds time interval. Their error rates are respectively 1.59 % and 2.11 %.

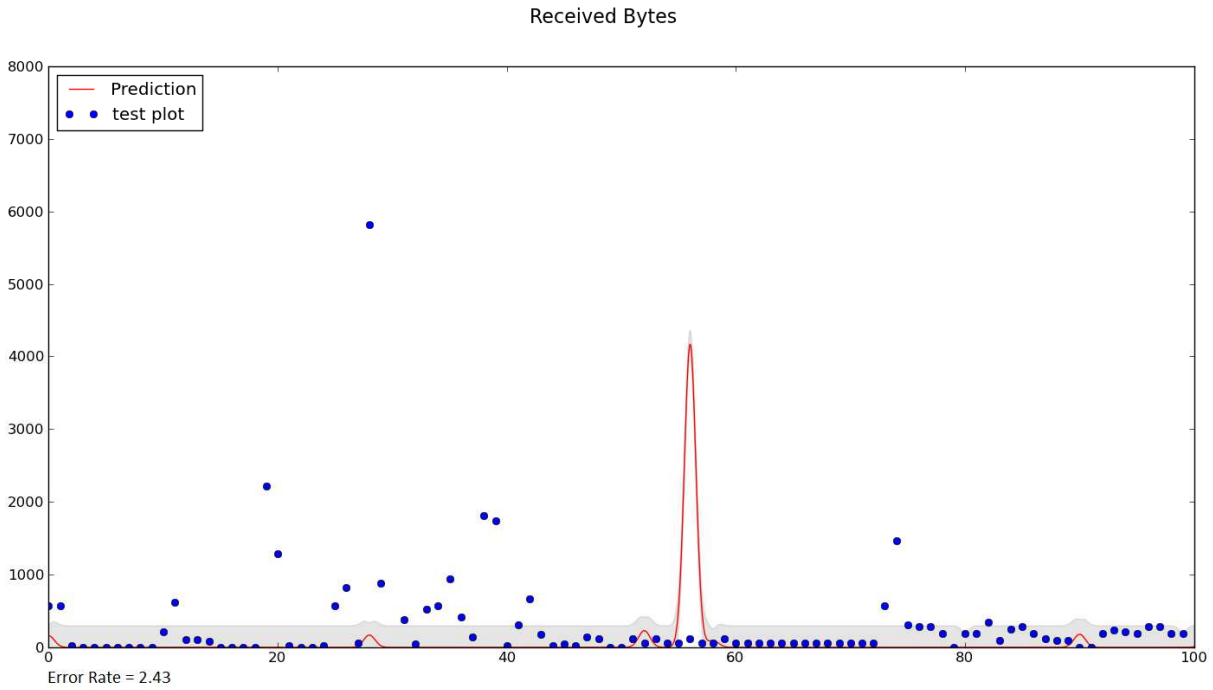


Figure 4.17: Forecast of the received bytes of the network based on the exact previous historical values

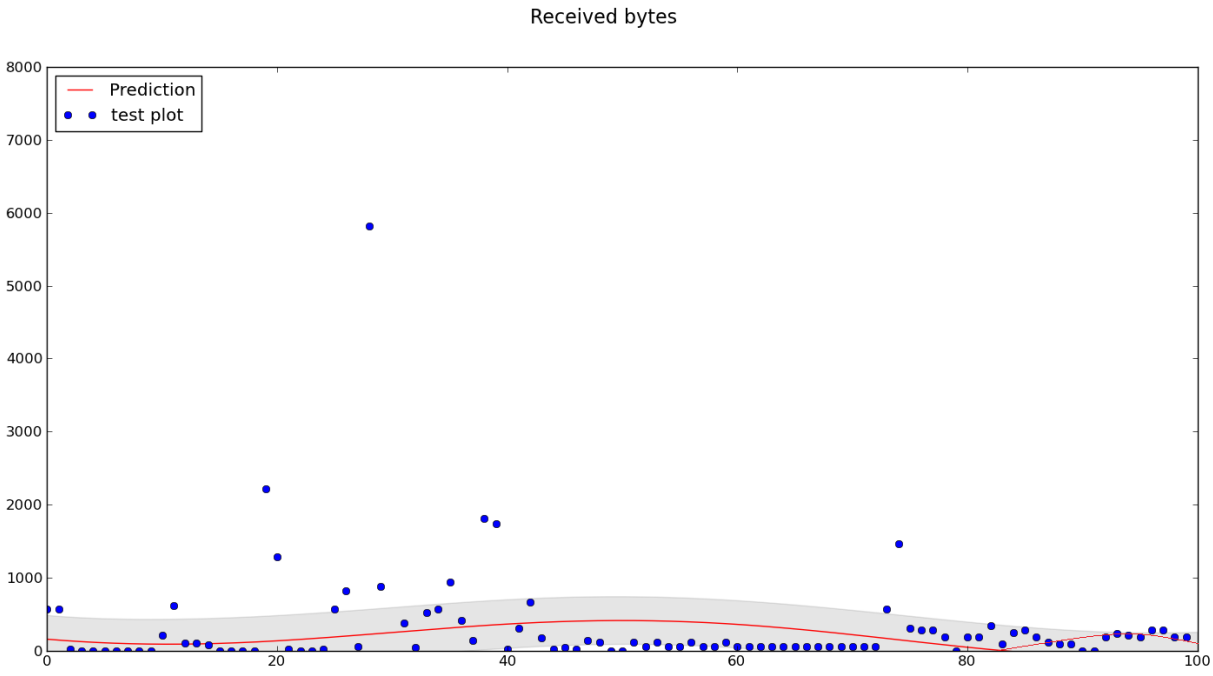


Figure 4.18: Smoother forecast of the future behavior of the network presented by its received bytes

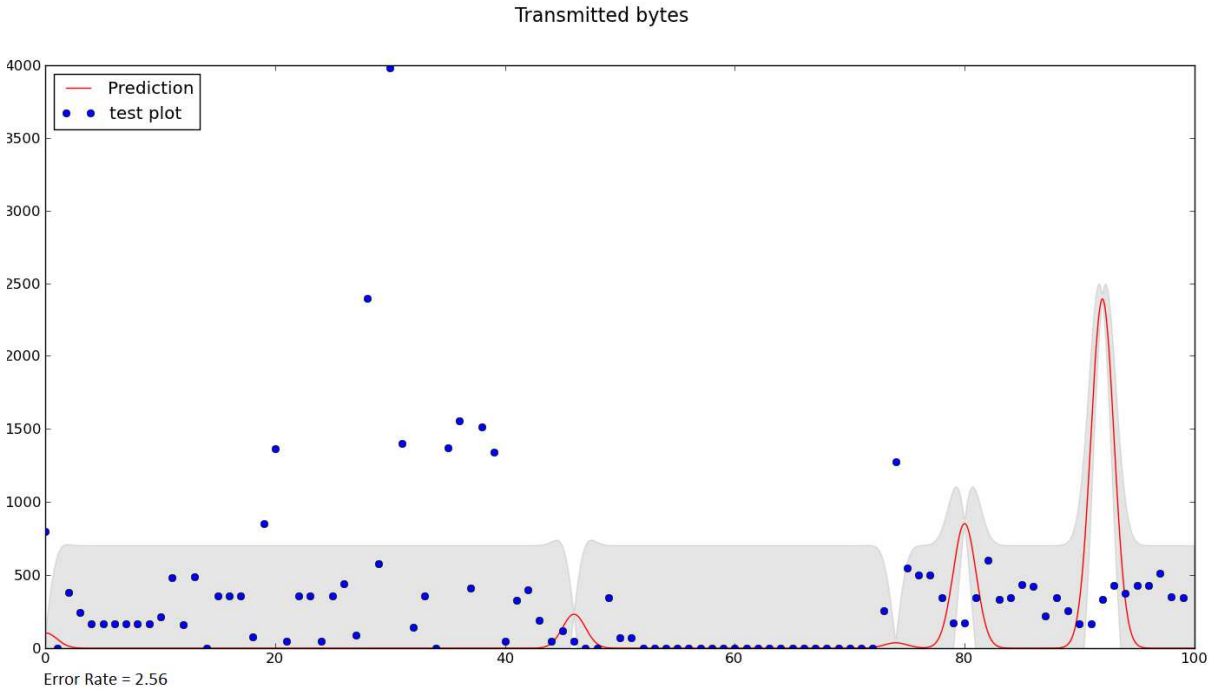


Figure 4.19: Forecast of the transmitted bytes of the network based on the exact previous historical values

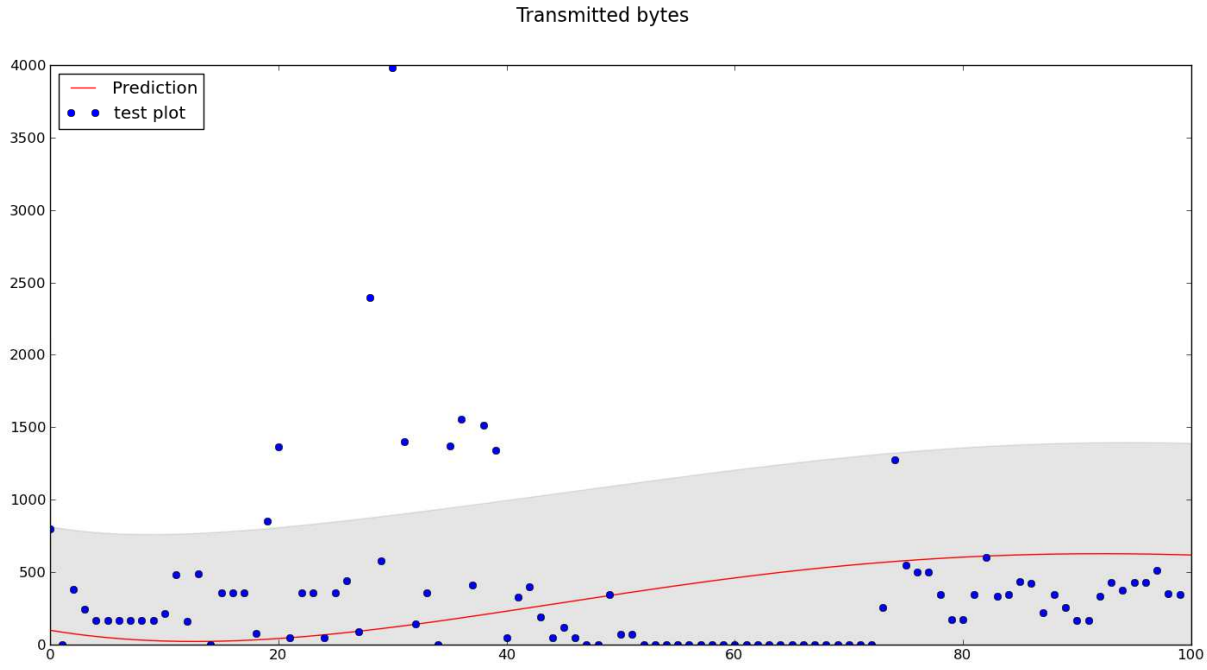


Figure 4.20: Smoother forecast of the future behavior of the network presented by its transmitted bytes

By looking to the error rate of the forecasts of the three components, we can say that the accuracy of the prediction by the Gaussian process regression model looks interesting and high which means that the modeled GPR is doing a good job.

4.2.2. Scenario 2: Results based on 5 second historical measures

In this section, we show the forecast of the future behavior of the studied components when using the collected values to predict what will happen in the next 5 seconds.

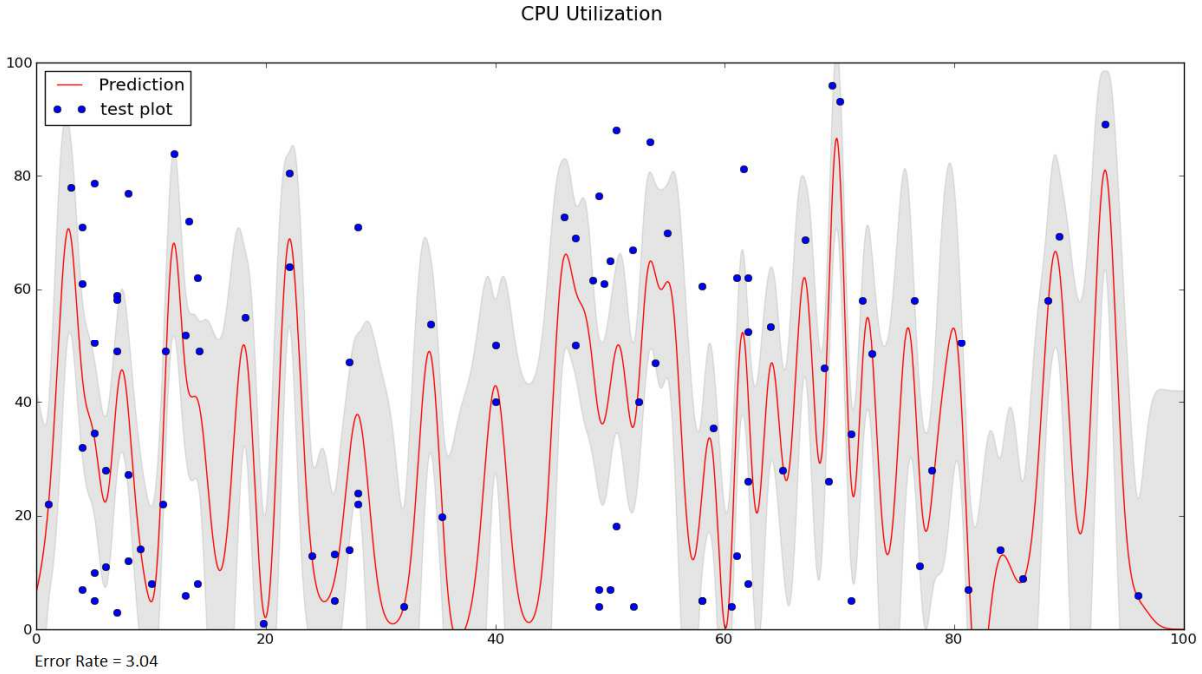


Figure 4.21: Forecast of the CPU usage based on 5 second historical values

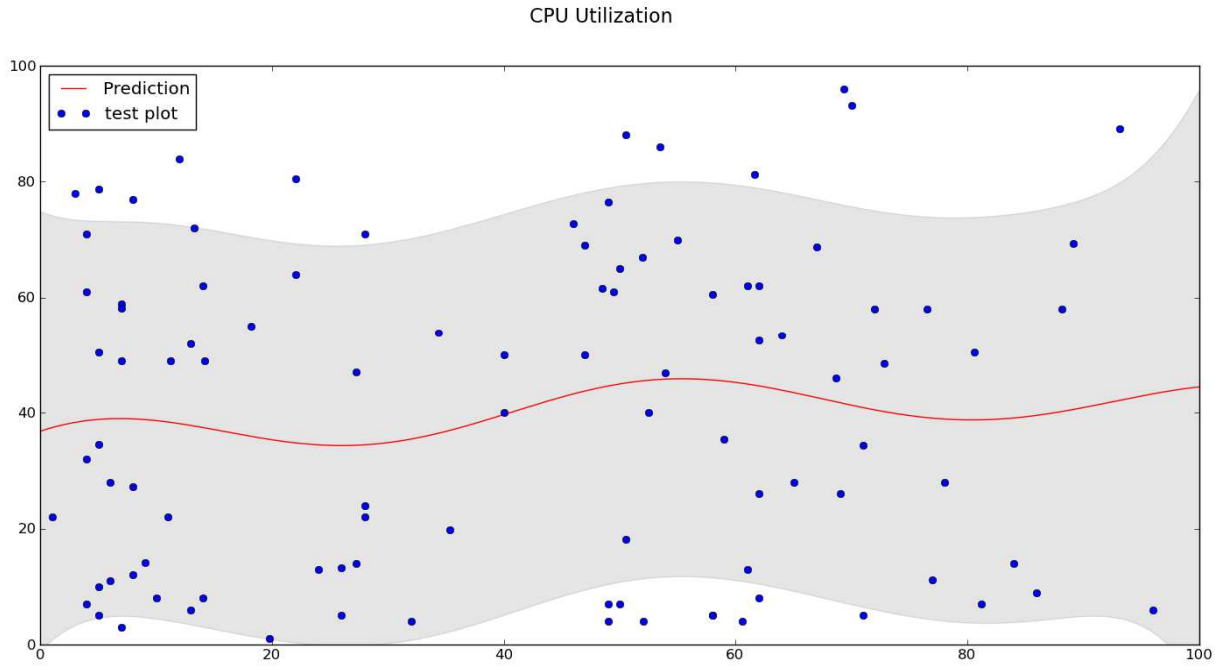


Figure 4.22: Smoother forecast of the future behavior of the CPU

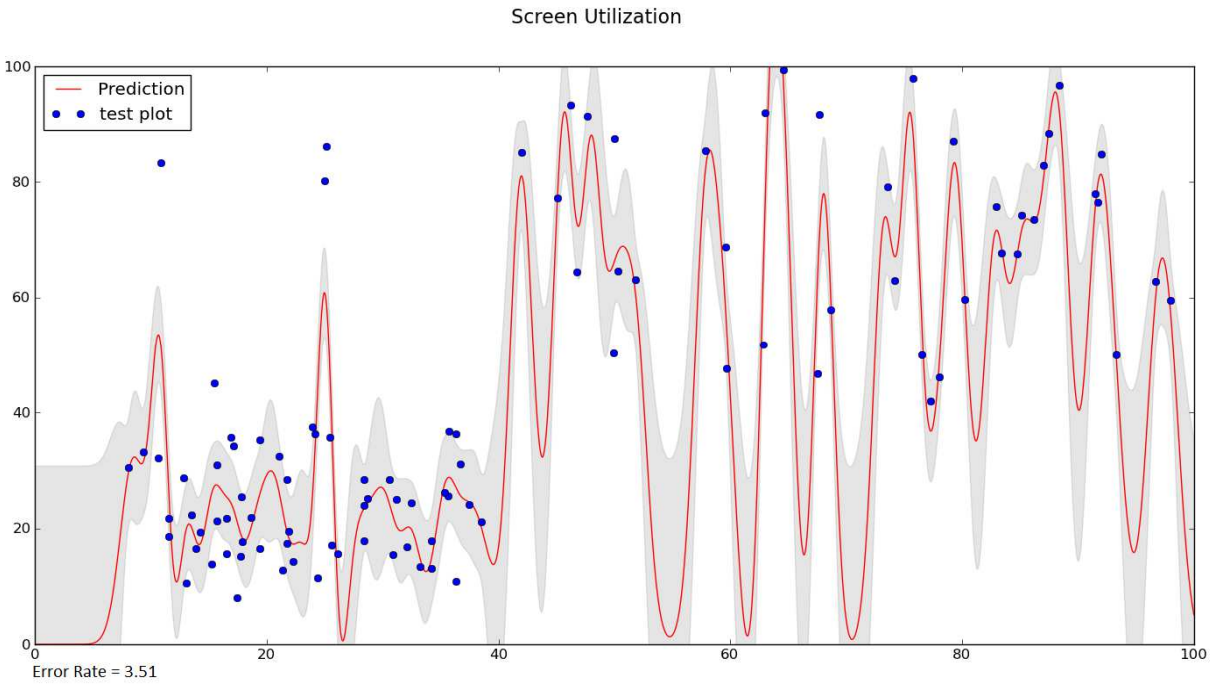


Figure 4.23: Forecast of the screen utilization based on 5 second historical values

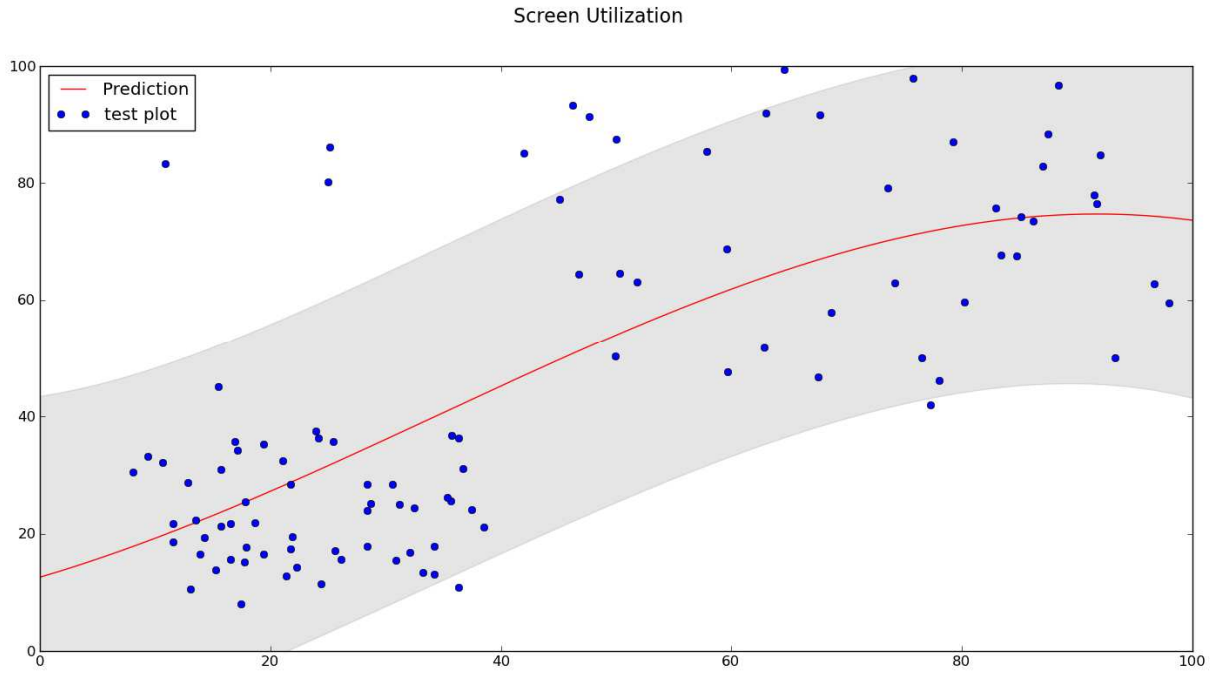


Figure 4.24: Smoother forecast of the future behavior of the screen

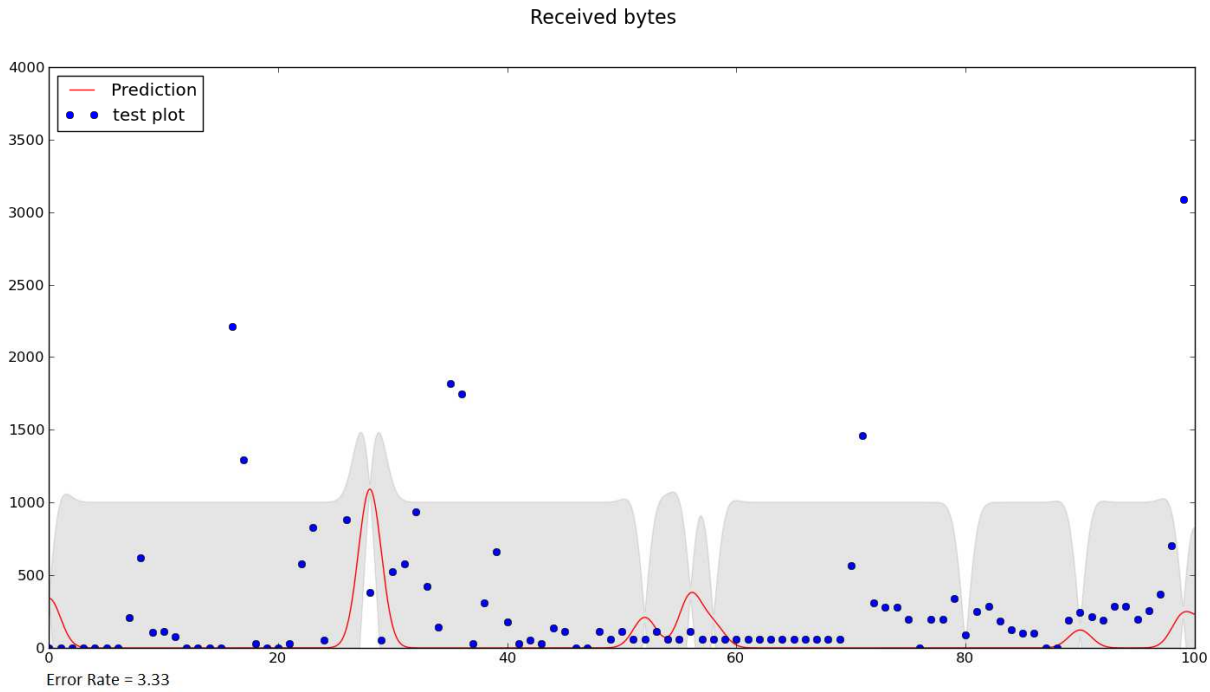


Figure 4.25: Forecast of the received bytes of the network based on 5 second historical values

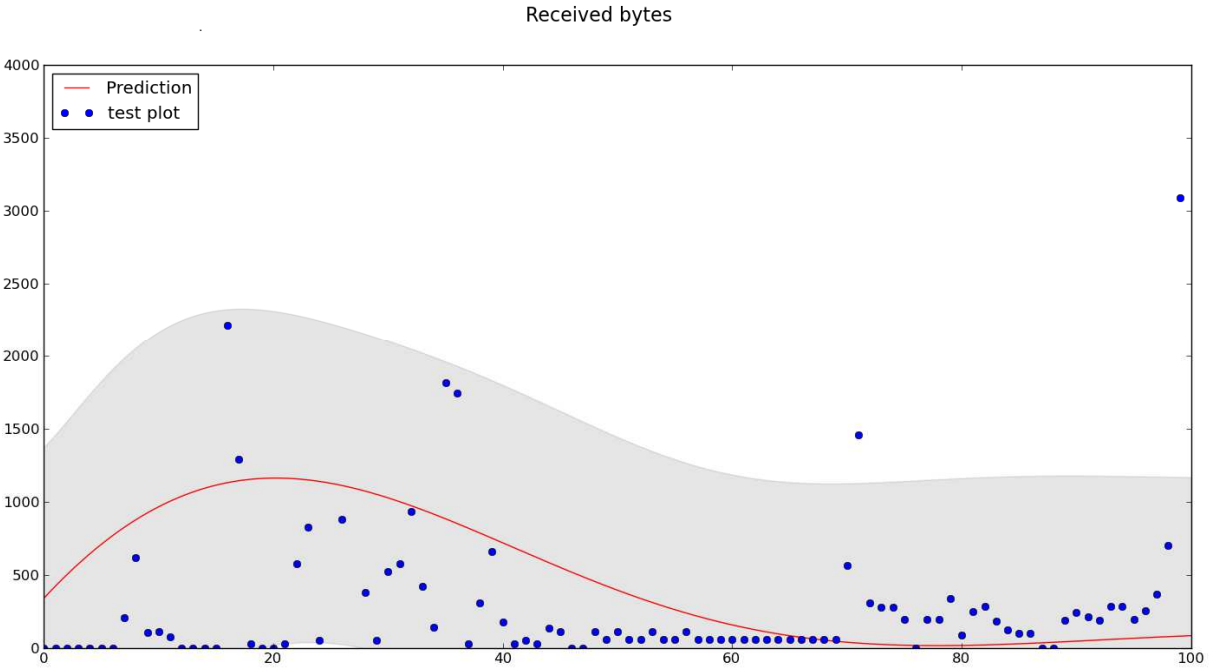


Figure 4.26: Smoother forecast of the future behavior of the network presented by its received bytes

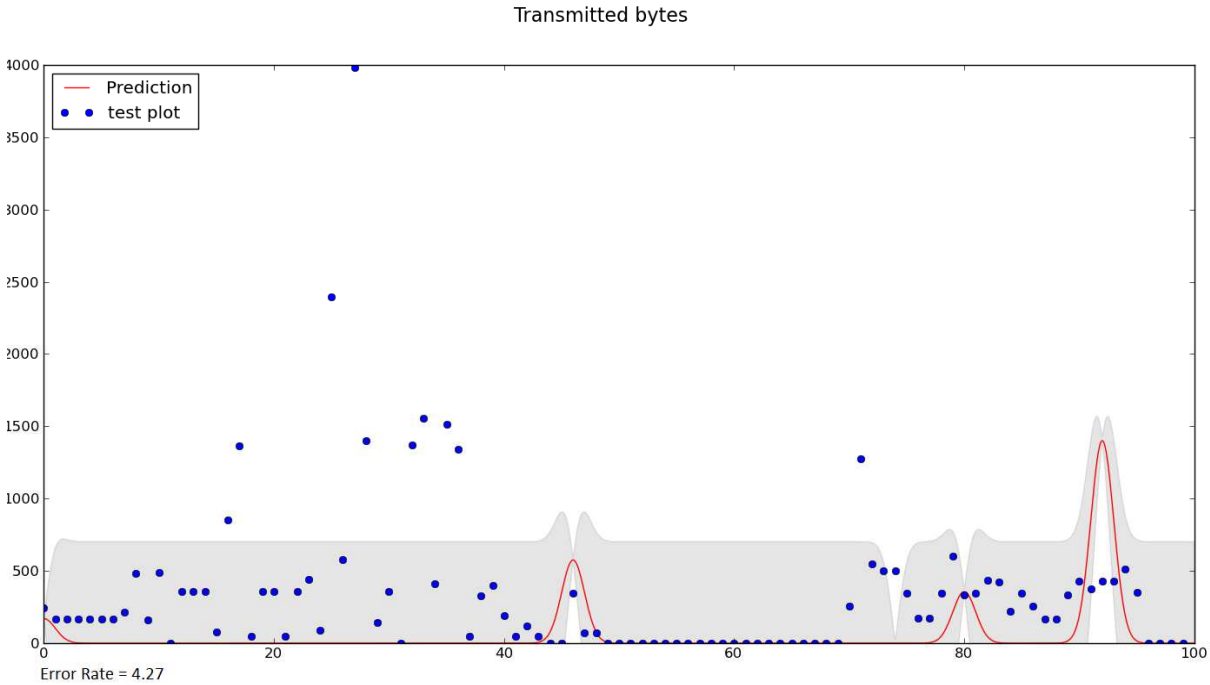


Figure 4.27: Forecast of the transmitted bytes of the network based on 5 second historical values

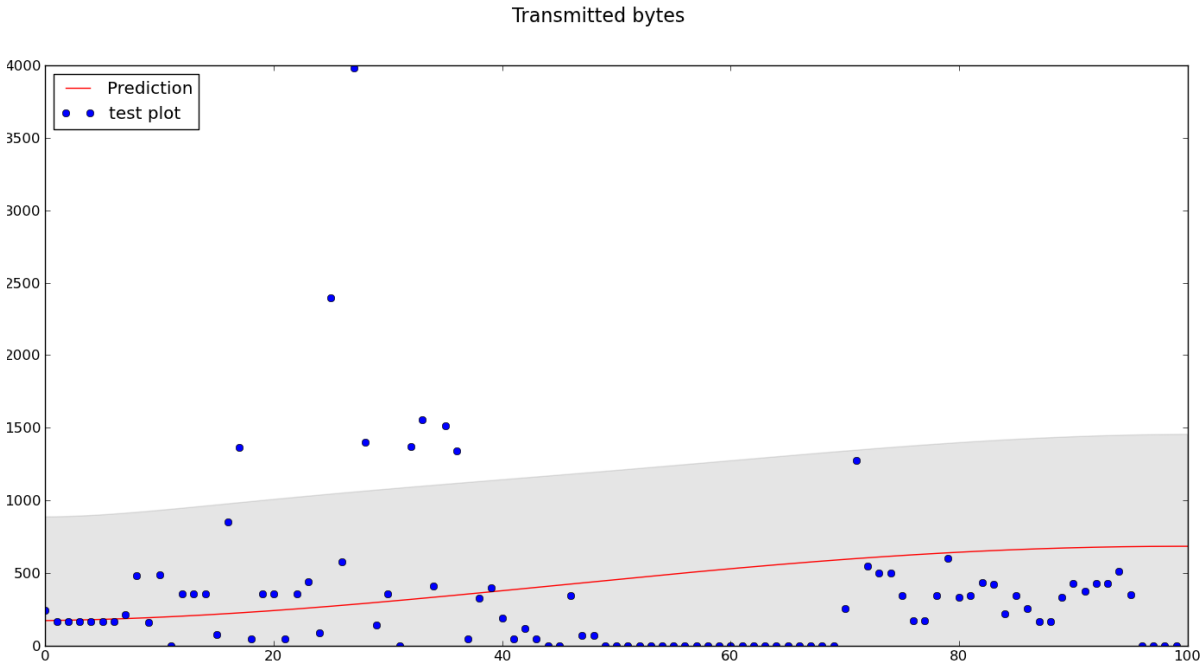


Figure 4.28: Smoother forecast of the future behavior of the network presented by its transmitted bytes

We can remark that when we used the historical measurement to predict how the system will behave in the next 5 seconds, the error rate is increased comparing to the error rate found when we did the prediction to examine the behavior of the system in the immediate next second.

Let's take as example the screen utilization; by looking to the error rates found in the two scenarios, we see that the error in the second scenario becomes almost twice the error rate found in the first scenario.

4.2.3. Scenario 3: Results based on 10 second historical measures

By examining the graphs presented in this section, we can conclude that the error becomes increasingly important by increasing the period of prediction based on the historical collected values . the error rates when doing the prediction of the next 10 seconds is higher than the error rates when doing the prediction of the next 5 seconds and the error rates found by doing the prediction of the next 5 seconds is more important than the ones found by doing the prediction of the next 1 second.

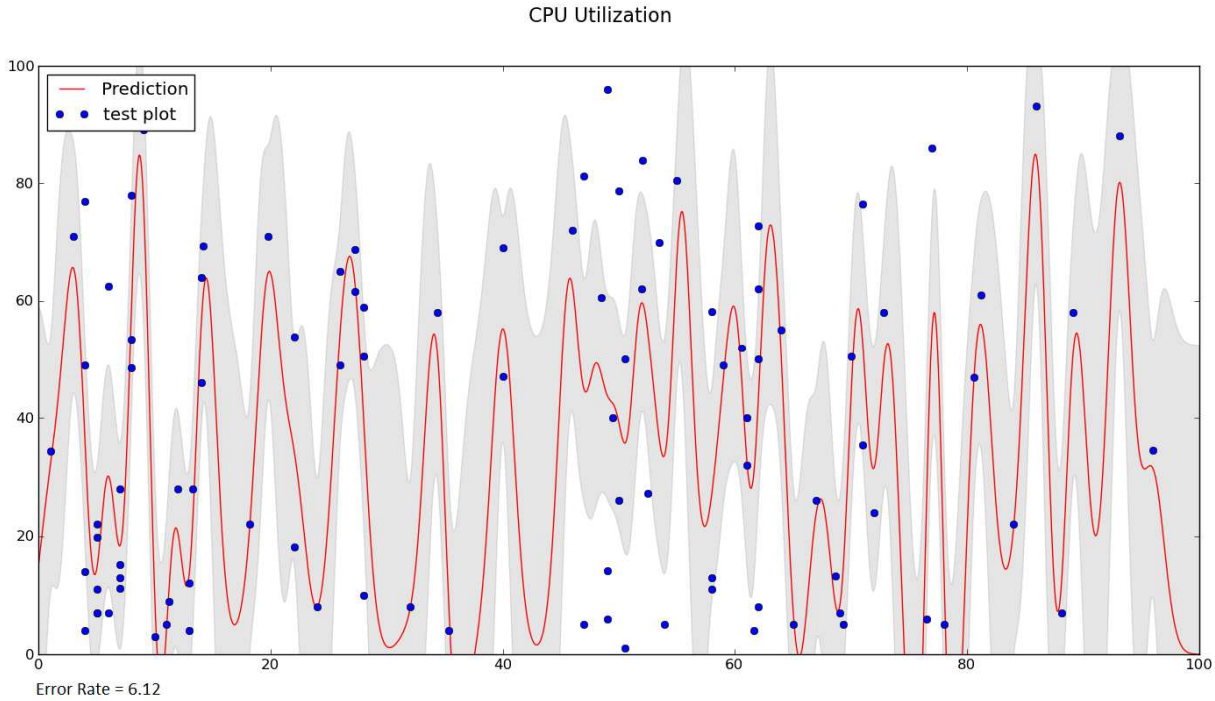


Figure 4.29: Forecast of the CPU usage based on 10 second historical values

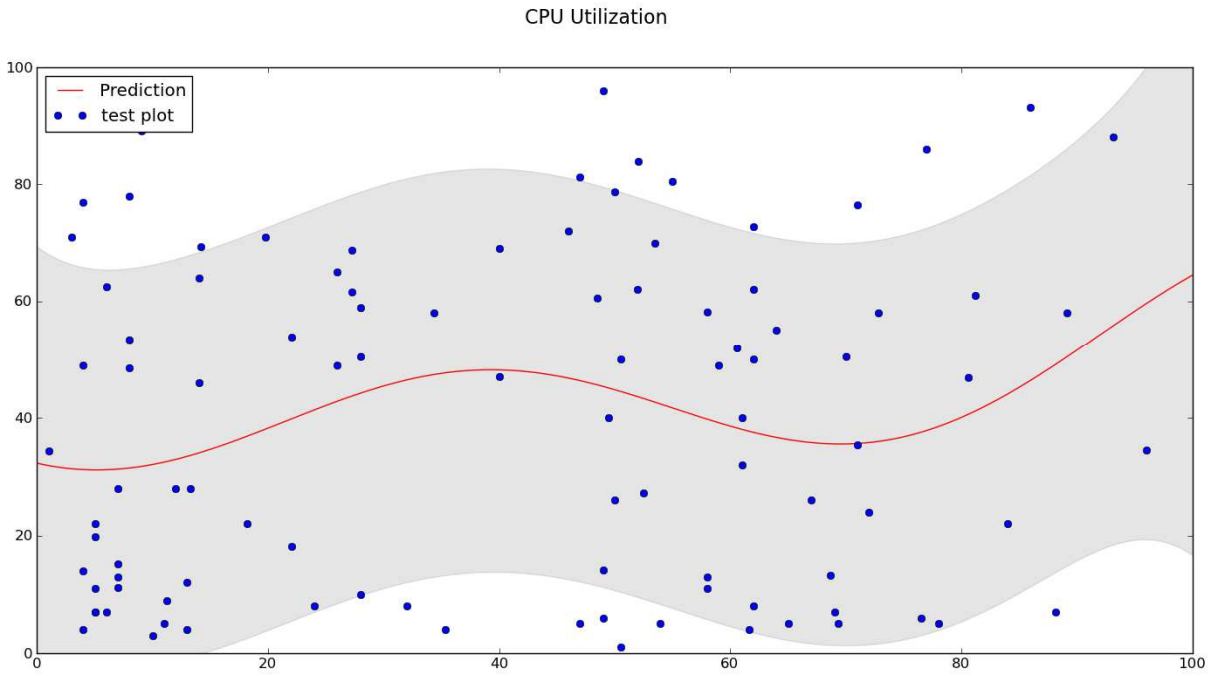


Figure 4.30: Smoother forecast of the future behavior of the CPU

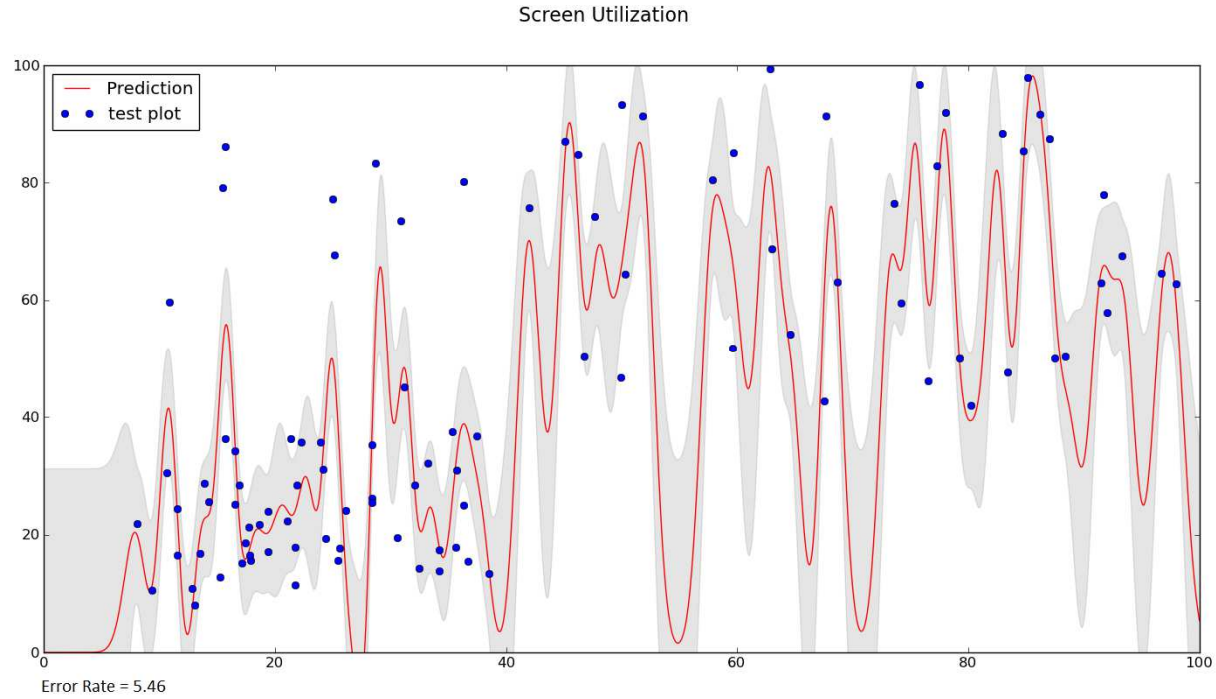


Figure 4.31: Forecast of the screen utilization based on 10 second historical values

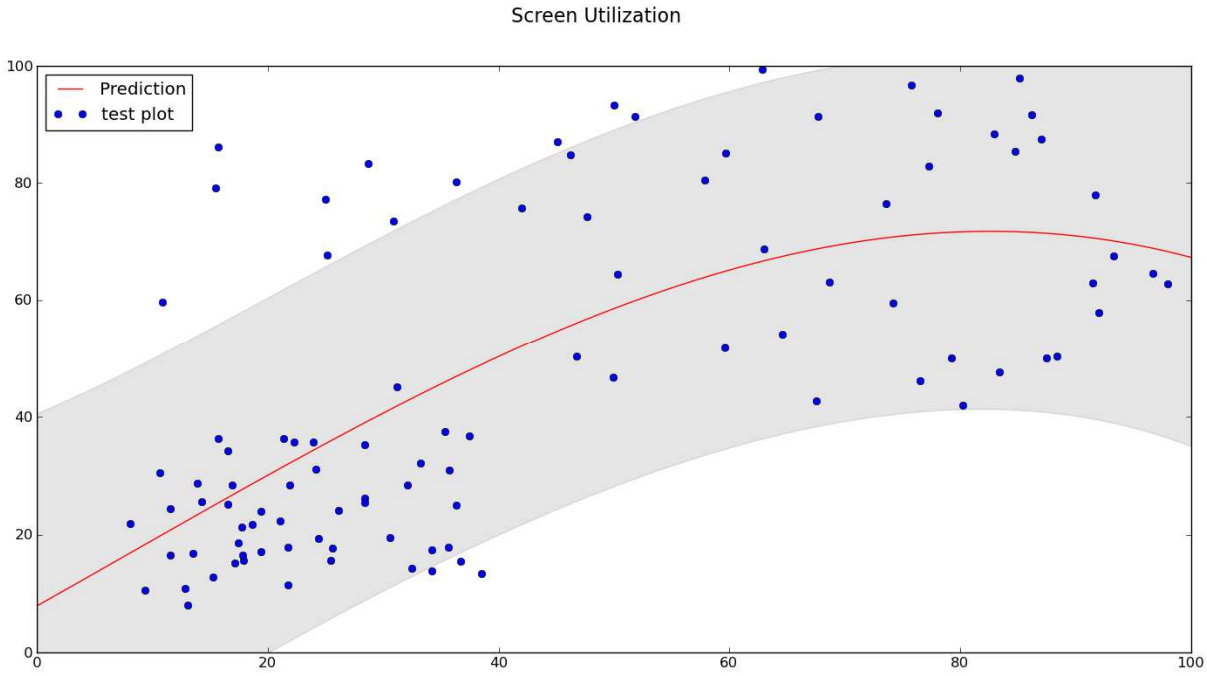


Figure 4.32: Smoother forecast of the future behavior of the screen

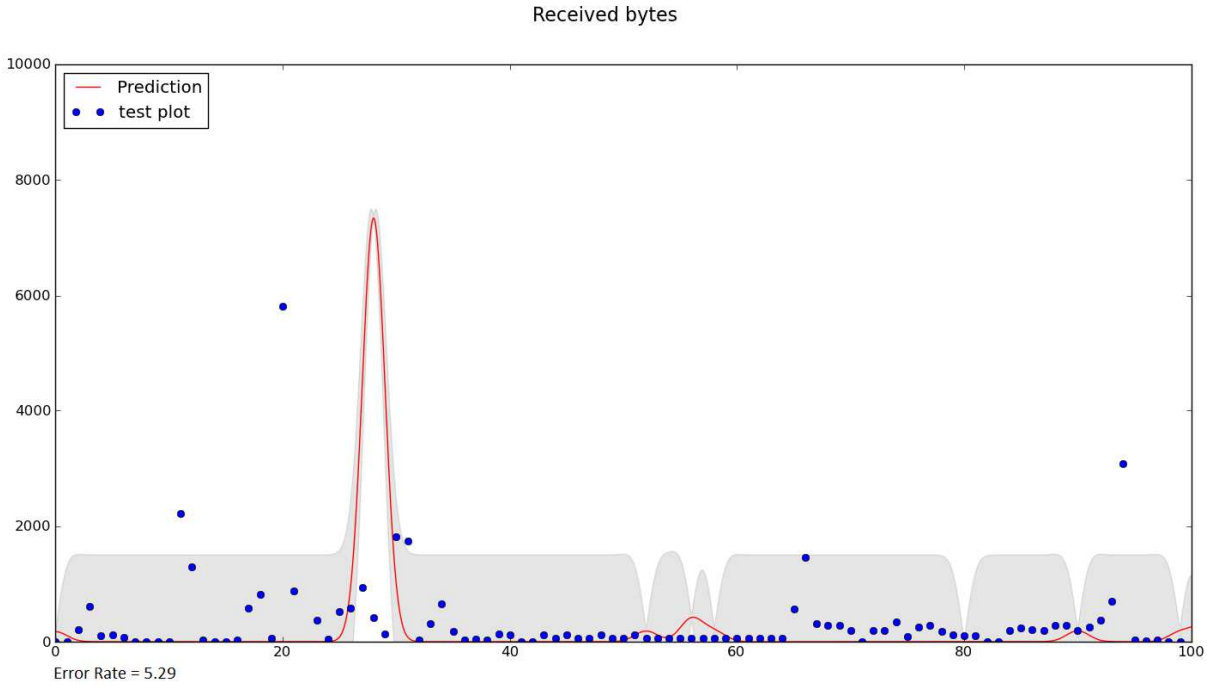


Figure 4.33: Forecast of the received bytes of the network based on 10 second historical values

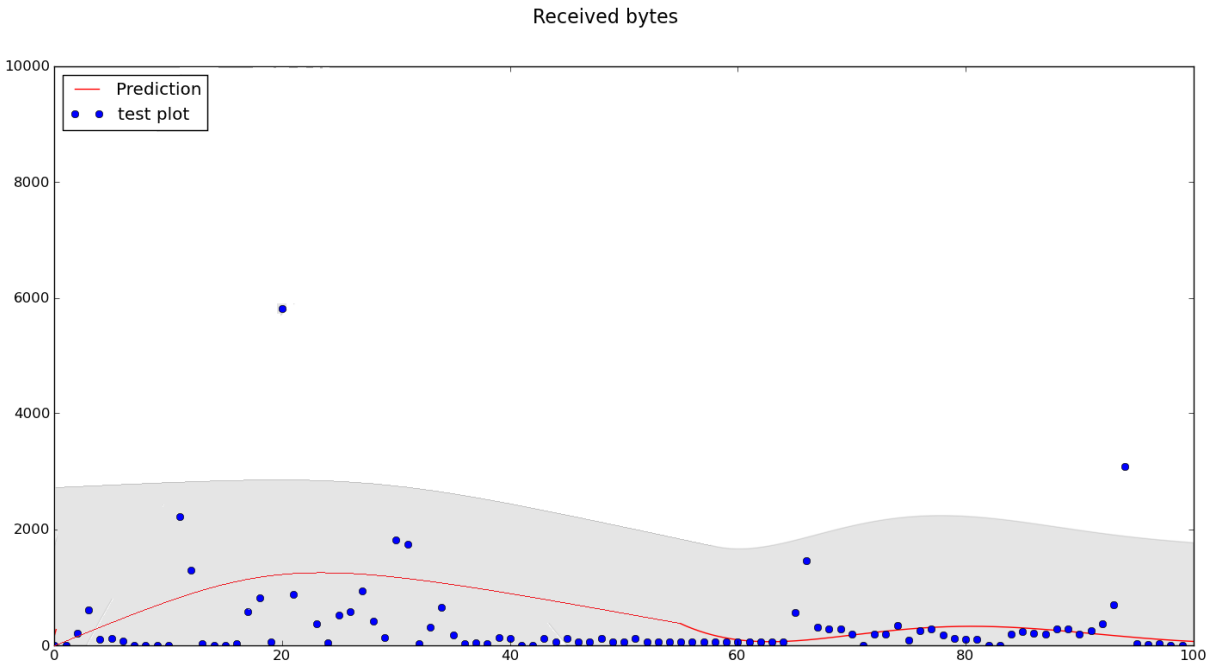


Figure 4.34: Smoother forecast of the future behavior of the network presented by its received bytes

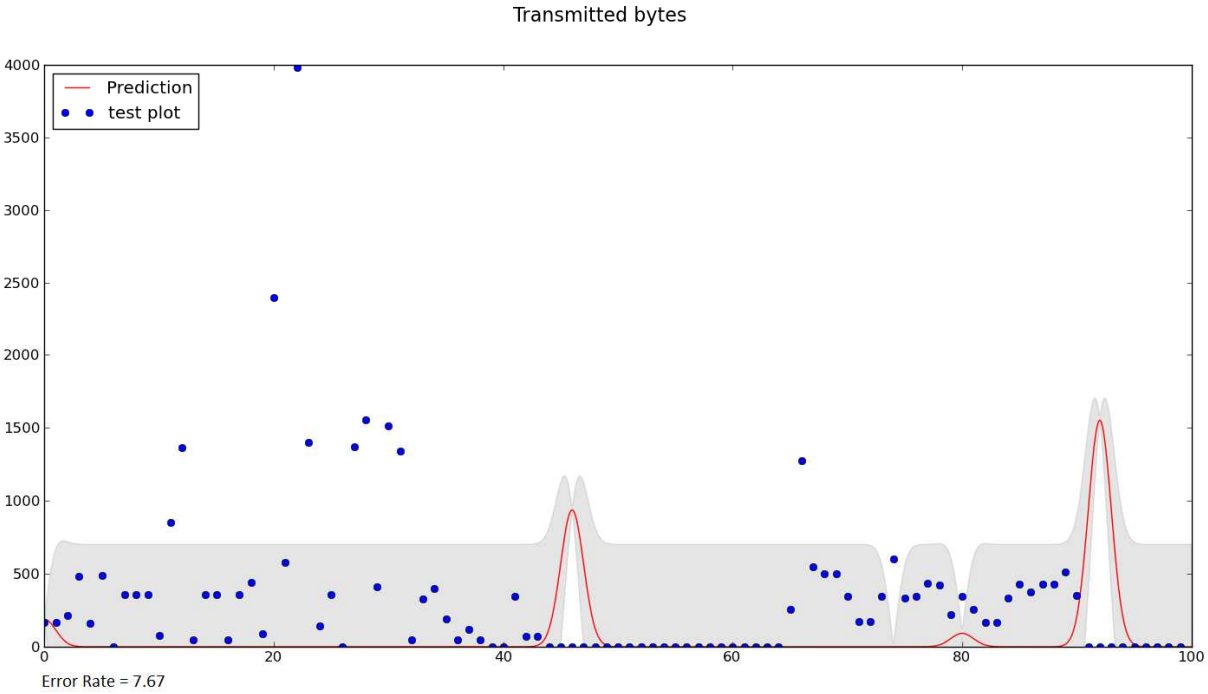


Figure 4.35: Forecast of the transmitted bytes of the network based on 10 second historical values

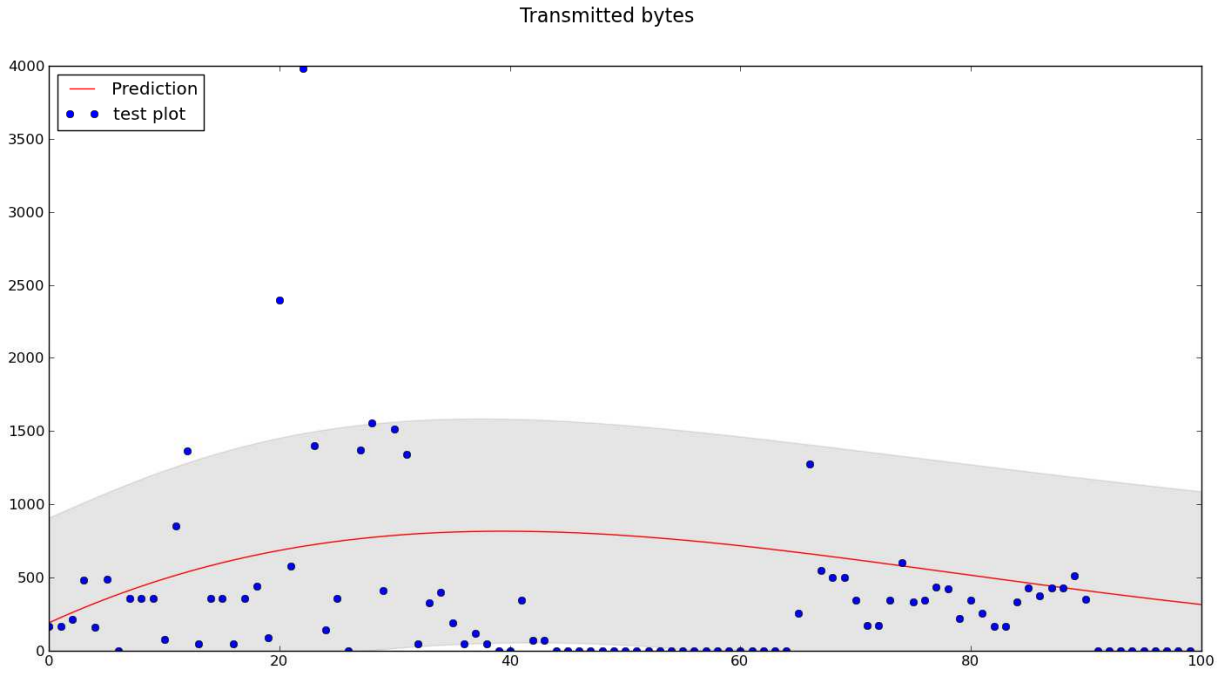


Figure 4.36: Smoother forecast of the future behavior of the network presented by its transmitted bytes

In the next chapter, we will present the accuracy of these two learning models in order to find the more accurate and efficient one that will give us the best way to do the forecast of the three studied components of the Android phone based on their historical measurements.

Chapter 5

Discussion

After presenting the prediction of the future behavior of the smartphone components based on their historical measurements in term of graphs (Chapter 4), we will validate in this chapter our results in term of accuracy. Since we concentrated our work on the CPU usage, the screen utilization and the network activity, we will study the accuracy of each component through the three scenarios presented in the previous chapter when applying the KNN regression and the Gaussian Process regression models. A comparison between the two models will be made and a conclusion about the most accurate one will be done.

5.1. Accuracy of the 1-NN regression model

Table 5.1 shows the accuracy of the CPU usage, the screen activity and the received and transmitted bytes of the network for each user and the average accuracy of each component when we used the collected data points to predict the exact next values (the next 1 second values) by applying the 1-NN regression model.

Table 5.1: Prediction accuracies when applying scenario 1 of the 1-NN regression model

	CPU Usage	Received bytes	Transmitted bytes	Screen Utilization
User 0	96.74	93.9	91.85	97.71
User 1	97.6	95.56	96.41	97.07
User 2	96.16	96.82	95.12	97.42
User 3	96.13	94.17	95.32	97.82
User 4	96.4	94.29	94.8	97.56
User 5	96.02	94.43	96.68	97.47
Average accuracy	96.5	94.86	95.03	97.5

We can remark that the accuracies of the 3 components vary between 91% and 97%. For example, the average accuracy of the prediction of the CPU usage is 96.5% which means that we achieve an error rate equal to 3.5%. In fact, by using the 1-NN regression model in scenario 1, the average accuracy of the forecast of these different components is ≈ 96% and then the error rate will be ≈ 4%.

Table 5.2 presents the prediction accuracy of the future behavior of these components when doing the forecast based on the second scenario which means when using the historical measures to forecast the behavior of the system in the next 5 seconds.

Table 5.2: Prediction accuracies when applying scenario 2 of the 1-NN regression model

	CPU Usage	Received bytes	Transmitted bytes	Screen Utilization
User 0	86.22	85.4	84.87	90.59
User 1	8689	86.5	89.9	88
User 2	84.55	87.35	88.77	93.12
User 3	91.02	83.28	87.03	94.09
User 4	90.77	88.66	91.48	93.19
User 5	93.65	83.34	87.7	93.71
Average accuracy	88.85	85.75	87.29	92.11

By taking a look on this table and compare it with the previous one, we can see that by applying scenario 2, the accuracies of the prediction become lower and the error rates increase and reach ≈ 12%. So, it is more accurate to do the prediction of the system based on the immediate previous historical measures than doing the prediction of the next 5 seconds even if this error rate is not very high and the prediction is still possible.

The table below presents the accuracy of the three components of the smartphone based on the third scenario with the 1-NN regression model.

Table 5.3: Prediction accuracies when applying scenario 3 of the 1-NN regression model

	CPU Usage	Received bytes	Transmitted bytes	Screen Utilization
User 0	83.56	81.38	84.61	85.57
User 1	84.29	82.99	85.71	89.4
User 2	83.71	81.37	85.65	87.85
User 3	86.23	82.92	83.82	88.97
User 4	86.62	83.06	84.79	88.41
User 5	85.97	83.68	87.58	88.1
Average accuracy	85.06	82.56	85.36	88.05

We can remark that the accuracies become weaker when we use the historical measurements to see how the system will behave in the next 10 seconds. The average accuracy of the 3 components by applying scenario 3 does not reach 86% and the error rate is almost equal ≈ 15%. This error rate is relatively high and in this case it becomes harder to make the right decision about the future behavior of the system and the future inactivity periods of the different components of the smartphone.

As a conclusion, we can say that the 1-NN regression model is doing a good forecast in general despite its simplicity but the accuracies found by this machine learning does not differ from the accuracies found by previous works done by other researchers (error rate between 5% and 20%); so, this approach does not help us to achieve our goal which is achieve better accuracy of prediction and present an activity forcasting model that is more accurate and efficient.

5.2. Accuracy of the Gaussian Process Regression model

By applying the Gaussian process regression model to our data, we obtain accuracies which are more interesting than the ones found by the 1-NN regression model. Table 5.4 shows the accuracies obtained by the first scenario when applying the Gaussian process regression approach to our collected data. These accuracies are specified to the CPU usage, the screen utilization and the network activity and presented for each user.

Table 5.4: Prediction accuracies when applying scenario 1 of the Gaussian process regression

	CPU Usage	Received bytes	Transmitted bytes	Screen Utilization
User 0	98.91	97.57	97.44	98.79
User 1	98.73	91.87	91.59	99.31
User 2	99.05	96.33	96.48	99.42
User 3	98.69	97.19	97.06	99.4
User 4	99.38	97.4	96.91	99.59
User 5	98.25	97.83	97.13	99.5
Average accuracy	98.83	96.36	96.1	99.33

We can see that the accuracy of the forecast concerning the 3 components is quite interesting and high. We can take as an example the screen utilization, the average accuracy of prediction of the screen utilization exceeds 99% which has as a consequence a very low error rate that is smaller than 1%. In general, by applying the Gaussian process regression approach, the average accuracy of the forecast of the 3 components is nearly equal to 98% and then the average error rate is approximately equal to 2%. So, the forecast of the future behavior of the smartphone and the prediction of the inactivity periods of the CPU, the screen and the network will be more precise and then the reduction of the power consumption will be easier.

Table 5.5 presents the accuracies of the prediction by using the data collected for each user per component when applying scenario 2.

Table 5.5: Prediction accuracies when applying scenario 2 of the Gaussian process regression

	CPU Usage	Received bytes	Transmitted bytes	Screen Utilization
User 0	96.96	96.67	95.73	96.49
User 1	97.23	92.71	91.26	97.51
User 2	97.76	96.99	94.61	97.82
User 3	96.87	95.92	95.89	97.36
User 4	96.09	96.86	95.7	96.93
User 5	95.85	96.5	91.33	97.54
Average accuracy	96.79	95.94	94.08	97.27

By using the historical collected measures to do the forecast of the behavior of the android phone in the next 5 seconds, we can see that the accuracies decline a little bit. By applying the second scenario with the Gaussian process regression approach in the training data, the average accuracy of the 6 users and the 3 components is almost equal to ≈ 96% which is still relatively high comparing to the accuracy found when applying the 1-NN regression model (accuracy is equal to ≈ 88%), and the error rate is not so important and it does not reach 4% (error rate ≈ 12% for 1-NN regression model with scenario 2). So, even by applying scenario 2, the prediction is still possible and we can say accurate with the Gaussian process regression model.

The table below shows the accuracies of the forecast when using historical data points to predict the future behavior of the Android components for the next 10 seconds.

Table 5.6: Prediction accuracies when applying scenario 3 of the Gaussian process regression

	CPU Usage	Received bytes	Transmitted bytes	Screen Utilization
User 0	93.88	94.71	92.33	94.54
User 1	93.95	89.9	88.27	94.83
User 2	93.57	95.23	91.68	95.11
User 3	94.71	87.8	89.17	95.26
User 4	94.7	94.59	92.78	96.01
User 5	94.42	93.96	89.65	96.6
Average accuracy	94.2	92.7	90.64	95.39

As we mentioned in the case of the 1-NN regression model, the accuracies of the forecast become lower when we want to predict the future behavior of a far period (10 seconds) than the future behavior of a close period (1 second). In fact, the average accuracy achieved in scenario 3 is decreased by 5% compared by the average accuracy obtained by scenario 1. On the other hand, the accuracy achieved by the GPR model in the third scenario (average error rate equal 6.77%) is still better than the accuracy achieved by the 1-NN regression model (average error rate equal 14.75%).

By looking to the accuracies obtained by the two models, we can say that the learning approach is a good technique to forecast the behavior of screen activity, CPU usage, and network activity in particular and the behavior of the smartphone in general based on historical measurements.

By comparing the accuracies achieved by the two learning approaches that we used to do the forecast, we can conclude that the Gaussian process regression approach does better job than the KNN regression approach. So, as a consequence it is more efficient to be based on the GPR model to predict the future behavior of the smartphone regarding the battery power usage and to make the right decision about the future inactivity periods which will help as a basis for power management. By having an accurate activity forecasting model, it will be easy to reduce battery power consumption of smartphones and optimize the power usage. By looking to the graphs presented in the previous chapter (chapter 4) concerning the

forecast of the CPU usage, the screen utilization and the network activity, we can remark that these components are not always used (inactive state) and even if they are used there exist two types of states which are the idle state[24] (or standby mode) and the active state[25]. For example, by looking to figure 4.13 which shows the behavior of the CPU usage, we can see that the peaks directed upwards present the active state of the CPU and the peaks directed downwards present the idle sate. In addition, figure 4.17 and 4.19 which characterize the behavior of the network regarding the received and transmitted bytes; show that the network is only used in the period presented by the big and small peaks and in the other periods the network is not used.

So, during the periods of inactivity or the idle state, we can apply different power management techniques that will help to save the power of the battery of the smartphone. In the paper [42], the author presents two techniques to optimize power in mobile architectures. The first one is specified for the screen called "*change blindness"* and aims to reduce the brightness of the screen and the second one is applied for the CPU and aims to control the CPU frequency.

The main advantage of our proposed solution is that by predicting the future inactivity periods based on the historical behavior of the smartphone, we can apply power management techniques, as the ones proposed in paper [42], and we can significantly avoid the excessive use of the power. By discovering the future inactive or idle period, we can reduce the screen quality by adjusting the Backlight to be Less Bright, reduce the frequency of the CPU slowly and even turn off the network when it is not used while maintaining the satisfaction of the end user.

As a future area of research, we can study the prediction of the components of the smartphone per application. If we are using the web browser, playing game or making calls (as an example), we can focus on the behavior of the phone components for each application to see what degree on power is used by each application and then try to apply power management techniques to the apps that use high quality screen, intensive CPU usage and extensive networking and then reduce battery power consumption.

[24] **The idle state:** the processor of the applications is not working but the processor of the modem is still operational and we say that the phone is in a low-power sleep mode.
[25] **The active state:** the processor of the applications is working and it occurs when a system wake lock is held.

Chapter 6

Conclusion

By the invention of the smartphones, mobile devices become indispensable tools for most of people thanks to the different features they offer for their users such as web browsing, e-mails, games, multimedia, social networking, etc; in addition to the basic functionalities which are calling and text messaging. With the integration of these different features, the users of the smartphones need a good level of performance and sophistication in mobile architecture especially regarding the battery. In fact, the main problem that faces the users of smartphones is the excessive battery power usage which leads to a short battery lifetime; and this problem is essentially due to the use of the multiples functions offered by the phone which rely to a very high quality screen, intensive CPU usage, and extensive networking. The excessive use of these different components of the smartphone leads to a very high power consumption and causes the drain of the battery.

Since the Android operating system is known as the most used and popular OS for mobile devices in these recent years, we concentrated our work on the Android smartphones. In the other hand, since the battery power usage depends on the real user activity patterns which differ from one user to another, we conducted usage studies with real users. As a first step, we collected data of the CPU usage, screen utilization and network activity (the most power consuming components) from 6 users having Android smartphones. As a second step, we modeled 2 learning approaches: the KNN regression model and the Gaussian process regression model. These models are used to forecast the future behavior of these components based on their historical behavior through the collected data. Three scenarios are presented in our solution using the historical measurements to see how the system will behave in the next 1 second, in the next 5 seconds and in the next 10 seconds.

by looking to the accuracies found by applying the KNN regression model and the Gaussian process regression model, we concluded that machine learning is a good choice to do the prediction of the future behavior of the system since the error rate of the 2 models is relatively low compared by previous solutions . When we compare the accuracy of the 2 proposed learning approaches, we can say that the Gaussian process is doing better job with accuracy almost equal to ≈ 98% than the KNN regression model which has lower accuracy nearly equal to ≈ 96% (results of the first scenario).

As a consequence, the Gaussian process regression model is most accurate and efficient to do the forecast of the future behavior of components of smartphones based on their historical behavior.

The main advantage of our proposed solution compared by the solution presented in previous works is that our solution is more accurate and efficient since it achieves an error rate lower than 2% however the error rates of the previous solutions done by other researches vary between 5% and 20%.

The Gaussian process model that we presented in our work can be used as a basis for power management. In fact, by predicting the future inactivity periods and the idle states of the phone components, we can apply techniques of power management, such as reducing the brightness of the screen or adjusting the CPU frequency, and components can be turned off in the right time, such as the Wi-Fi and the 3G. These methods will avoid the extensive use of the phone resources and then battery power consumption can be significantly reduced.

Our thesis work presented in this paper can be extended by not only focusing on the 3 components of the smartphone which are the CPU, the screen and the network but also monitoring other components of the smartphones that can consume too much power and can cause the drain of the battery. In fact, we can study the sensors embedded on smartphones such as the accelerometer[26], the gyroscope[27], the compass[28], microphone, camera, GPS, etc. These sensors present new challenge to the smartphones since they give

[26] **Accelerometer:** or acceleration sensor is a complex electronic system that is attached to a mobile object used to measure its acceleration.
[27] **Gyroscope:** is a sensor of angular position.
[28] **Compass:** is able to calculate the heading direction by determining the rotation of the device depending on the Earth's magnetic north pole. It provides the phone with its exact orientation and position.

additional functionalities to the phone but they come with high cost which is excessive power consumption.

In the other hand, we are focusing during our work on doing the forecast of the behavior of the whole system. As a future work, we can study the behavior of the components of each application running in the smartphone separately and then we will have detailed information that will help to manage the power consumption of each application.

Moreover, since we choose to study the behavior of the system in its real environment which is the end user, we can enlarge our database and collect data from more than 6 users since the system behavior depend on the user activity patterns.

The work shown in this paper was presented for a specific mobile device which is the Android smartphone but it can be applied to any kind of smartphones and Android devices such as the Android tablets that are starting to get famous and popular as the Android smartphones.

References

[1] Smartphone. Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Smartphone

[2] Smartphone. Retrieved from Phone Scoop:

http://www.phonescoop.com/glossary/term.php?gid=131

[3] Android, the world's most popular mobile platform. Retrieved from Android: http://www.android.com/about/

[4] Samsung Xcover E2370 product page. (2011). Retrieved from Samsung: http://www.samsung.com/se/consumer/mobile/mobilephones/mobilephones/GTE2370FSAXEE/index.idx?pagetype=prd_detail

[5] Douglas, P. (2010, May 19). “Larry Page: Your Android battery life should last a day”. Retrieved from Techradar: http://www.techradar.com/news/phone-and-communications/mobile-phones/larry-page-your-android-battery-should-last-a-day-690439

[6] Cutlack, G. (2010, June 21). “Android battery life: how to improve it”. Retrieved from Techradar: http://www.techradar.com/news/phone-and-communications/mobile-phones/android-battery-life-how-to-improve-it-697772

[7] Selim. G, Chandra. K. (2006, October). “A RunTime, Feedback-Based Energy Estimation Model For Embedded Devices”.

[8] W. L. Bircher, M. Valluri, J. Law and L. K. John (2005). “Runtime identification of microprocessor energy saving opportunities”. In Proceedings of the International symposium on Low power electronics and design (ISLPED).

[9] R. Joseph and M. Martonosi (2001). “Run-time power estimation in high performance microprocessors”. In Proceedings of the 2001 international symposium on Low power electronics and design (ISLPED).

[10] C. Isci and M. Martonosi (2003). “Runtime power monitoring in high-end processors: Methodology and empirical data”. In Proceedings of the 36th ACM/IEEE International Symposium on Micro architecture (MICRO).

[11] G. Contreras and M. Martonosi (2005). « Power prediction for Intel xscale processors using performance monitoring unit events". In Proceedings of the International symposium on Low power electronics and design (ISLPED).

[12] M. Anand, E. B. Nightingale, and J. Flinn (2004). "Ghosts in the machine: interfaces for better power management". In Proceedings of the 2nd international conference on Mobile systems, applications, and services (MOBISYS).

[13] D. Rakhmatov, S. Vrudhula, and D. A. Wallach (2002, August). "Battery lifetime prediction for energy-aware computing". In Proceedings of the International Symposium on Low Power Electronics and Design.

[14] P. Rong and M. Pedram (2003, March). "Remaining battery capacity prediction for lithium-ion batteries". Conference of Design Automation and Test in Europe.

[15] D. Rakhmatov and S. Vrudhula (2001, August). "Time-to-failure estimation for batteries in portable electronic systems". In Proceedings of the International Symposium on Low Power Electronics and Design.

[16] K. C. Syracuse and W. Clark (1997, January). "A statistical approach to domain performance modeling for oxyhalide primary lithium batteries". In Proceedings of Annual Battery Conference on Applications and Advances.

[17] D. Brooks, V. Tiwari, and M. Martonosi. Wattch (2000). "A framework for architectural-level power analysis and optimizations". In Proceedings of the Intl. Symposium on Computer Architecture, pages 83–94.

[18] T. L. Cignetti, K. Komarov, and C. S. Ellis (2000, August). "Energy estimation tools for the PalmTM". In Proceedings of the Intl. Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems.

[19] T. K. Tan, A. Raghunathan, G. Lakshiminarayana, and N. K. Jha (2001, June). "High-level software energy macro-modeling". In Proceedings of Design Automation Conference, pages 605–610.

[20] S. Gurumurthi, A. Sivasubramaniam, M. J. Irwin, N. Vijaykrishnan, M. Kandemir, T. Li, and L. K. John (2002, February). "Using Complete Machine Simulation for Software Power Estimation: The SoftWatt Approach". In Proceedings of the Intl. Symposium on High Performance Computer Architecture, pages 141–150.

[21] Android (operating system). Retrieved from Wikipedia:

http://en.wikipedia.org/wiki/Android_(operating_system)

[22] android advantages and disadvantages. Retrieved from syahrulsyaputra: http://syahrulsyaputra.com/android-advantages-and-disadvantages.html

[23] POWER CONSUMPTION. Retrieved from Webster's Online Dictionary: http://www.websters-online-dictionary.org/definitions/POWER+CONSUMPTION?cx=partner-pub-0939450753529744%3Av0qd01-tdlq&cof=FORID%3A9&ie=UTF-8&q=POWER+CONSUMPTION&sa=Search#906

[24] Carroll, A. (2011). “An Analysis of Power Consumption in a Smartphone”.

[25] HTC Sensation. Retrieved from gsmarena: http://www.gsmarena.com/htc_sensation-3875.php#

[26] Koprowski, G. J. (2006, July 14). “Interest in Touch-Screen Technology for Mobile Phones Growing”. Retrieved from TechNewsWorld:

http://www.technewsworld.com/story/51710.html

[27] N. Sklavos, K. Touliou. (2007, May). “A System-Level Analysis of Power Consumption & Optimizations in 3G Mobile Devices”.

[28] Schapire, R. (2003, February 4). “Foundations of Machine Learning”.

[29] Trivedi, S. (2009, March 22). “Why are Support Vectors Machines called so?”. Retrieved from Onionesque Reality:

http://onionesquereality.wordpress.com/2009/03/22/why-are-support-vectors-machines-called-so/

[30] Fukunaga, K. (1990). “Introduction to statistical pattern recognition”.

[31] k-Nearest Neighbors. Retrieved from StatSoft: http://www.statsoft.com/textbook/k-nearest-neighbors/

[32] k-nearest neighbor algorithm. Retrieved from Wikipedia:

http://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm

[33] Mean. Retrieved from Wikipedia:

http://en.wikipedia.org/wiki/Mean#Mean_of_a_function

[34] Covariance. Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Covariance

[35] Geiger, A. (2007). “Gaussian Processes for Machine Learning”. Retrieved from rainsoft: http://www.rainsoft.de/projects/gausspro.html

[36] Ebden, M. (2008, August). “Gaussian Processes for Regression: A Quick Introduction”.

[37] Rasmussen, C. E. “Gaussian Processes in Machine Learning”.

[38] Regression. Retrieved from investopedia:

http://www.investopedia.com/terms/r/regression.asp#axzz1kx83gnqI

[39] C.E.Rasmussen & C.K.I.Williams. (2006). “Gaussian Processes for Machine Learning.”

[40] Andreas Buja; Trevor Hastie; Robert Tibshirani. (2008, April). “Linear Smoothers and Additive Models”.

[41] Hastie, T. J., and Tibshirani, R. J. (1990). “Generalized Additive Models”, sec. 3.5.

[42] Alex Shye, Benjamin Scholbrock, Gokhan Memik. (2009, December). “Into the Wild: Studying Real User Activity Patterns to Guide Power Optimizations for Mobile Architectures”.

Appendix A - List of abbreviations & Glossary

AI	**A**rtificial **I**ntelligence
CPU	**C**entral **P**rocessing **U**nit
GP	**G**aussian **P**rocess
GPR	**G**aussian **P**rocess **R**egression
GPS	**G**lobal **P**ositioning **S**ystem
HSDPA	**H**igh **S**peed **D**ownlink **P**acket **A**ccess
HTC	**H**igh **T**ech **C**omputer
iOS	**i**Phone **O**perating **S**ystem
KNN	**K n**earest **n**eighbors
OS	**O**perating **S**ystem
PDA	**P**ersonal **D**igital **A**ssistant
QHD	**Q**uarter of full **H**igh **D**efinition
SD	**S**ecure **D**igital
Wi-Fi	**W**ireless **F**idelity
WLAN	**W**ireless **L**ocal **A**rea **N**etwork

Mean and Covariance functions:

The mean and the covariance function of the Gaussian process of a real process ***f(x)*** are presented respectively by:

$$m(x) = E[f(x)]$$

$$k(x, x') = E\big[\big(f(x) - m(x)\big)\big(f(x') - m(x')\big)\big]$$

Hyperparameters:

The hyperparameters present a reference to the parameters of the covariance function to insist on the point that these parameters are used in a non-parametric model.

The likelihood:

The likelihood is the probability density of the observations given the parameters.

The marginal likelihood:

The marginal likelihood refers to the marginalization over the function values ***f***.

Appendix B - Versions of the Android OS

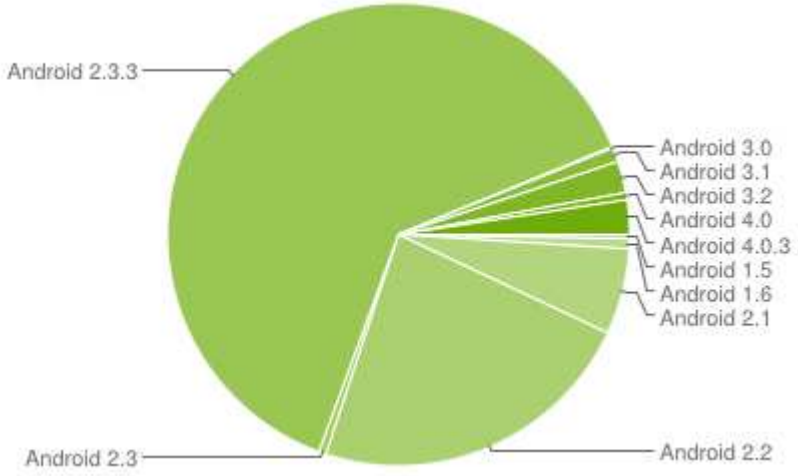


Fig: The different versions of the Android OS

Platform	Codename	API Level	Distribution
Android 1.5	Cupcake	3	0.3%
Android 1.6	Donut	4	0.7%
Android 2.1	Eclair	7	6.0%
Android 2.2	Froyo	8	23.1%
Android 2.3 - Android 2.3.2	Gingerbread	9	0.5%
Android 2.2.3 - Android 2.3.7		10	63.2%
Android 3.0	Honeycomb	11	0.1%
Android 3.1		12	1.0%
Android 3.2		13	2.2%
Android 4.0 – Android 4.0.2	Ice Cream Sandwich	14	0.5%
Android 4.0.3		15	2.4%

Tab: Data collected during a 14-day period ending on April 2, 2012

Link: http://developer.android.com/resources/dashboard/platform-versions.html

Appendix C - Examples of outputs of the Nu Jamlogger application and the KNN regression model

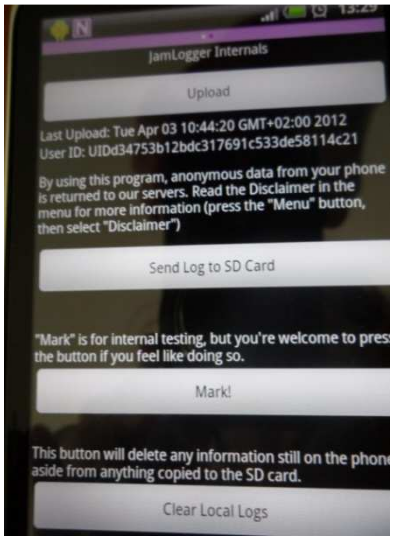


Fig: The GUI of the NU JamLogger application

```
1330183825327 : Logger_Version v1.7-17 3600000
1330183825328 : Logger_Version v1.7-17 3600000
1330183825328 : Platform_Info Manf:"HTC" Brand:"htc_europe" Product:"htc_pyramid" Model:"HTC Sensation Z710e" Device:"pyramid" CPU_abi:
1330183827273 : Network_Traffic_t eth0 5109 1025 8 0 1744 12 0
1330183828286 : Load_Avg 4.16 6.96 9.63 2 1013 4844
1330183829314 : CPU_Freq 13
1330183829319 : CPU_Utilization 23.47 19.7 4.40
1330183831351 : Network_Traffic_t eth0 4079 812 16 0 17488 21 0
1330183832384 : Load_Avg 4.15 6.91 9.60 2 1011 4849
1330183841264 : CPU_Freq 13
1330183841268 : CPU_Utilization 25.49 13.53 11.96
1330183843284 : Network_Traffic_t eth0 11933 3054 21 0 26737 27 0
1330183849149 : Load_Avg 4.38 6.91 9.59 2 1014 4859
1330183960525 : CPU_Freq 13
1330183960530 : CPU_Utilization 26.59 15.67 10.91
1330183960532 : GTalk_Service_Connected
1330183962548 : Network_Traffic_t eth0 119264 1217 10 0 901 6 0
1330183963563 : Load_Avg 4.35 6.87 9.56 2 1014 4863
1330183965593 : CPU_Freq 13
1330183965612 : CPU_Utilization 4.14 3.16 0.99
1330183967632 : Network_Traffic_t eth0 5084 0 0 0 356 1 0
1330183968648 : Load_Avg 4.32 6.82 9.53 2 1013 4867
1330183970684 : CPU_Freq 13
1330183970704 : CPU_Utilization 10.22 8.84 1.38
1330183972735 : Network_Traffic_t eth0 5103 0 0 0 356 1 0
1330183973750 : Load_Avg 4.29 6.77 9.50 2 1013 4871
1330183975780 : CPU_Freq 13
1330183975796 : CPU_Utilization 11.76 9.61 2.16
1330183978830 : Load_Avg 4.27 6.73 9.47 2 1013 4875
```

Fig: Example of a log file from the Nu JamLogger application

Network_Traffic_t								
timestamp	time	net_name	rx_packets	tx_packets	tx_errs	rx_bytes	tx_bytes	rx_errs
1330183827273	25-02-2012 15:30:27	eth0	8	12	0	1025	1744	0
1330183831351	25-02-2012 15:30:31	eth0	16	21	0	812	17488	0
1330183843284	25-02-2012 15:30:43	eth0	21	27	0	3054	26737	0
1330183962548	25-02-2012 15:32:43	eth0	10	6	0	1217	901	0
1330183967632	25-02-2012 15:32:48	eth0	0	1	0	0	356	0
1330183972735	25-02-2012 15:32:53	eth0	0	1	0	0	356	0
1330183993058	25-02-2012 15:33:13	eth0	0	1	0	0	356	0
1330183998155	25-02-2012 15:33:18	eth0	1	1	0	28	46	0
1330184023639	25-02-2012 15:33:44	eth0	0	1	0	0	356	0
1330184028741	25-02-2012 15:33:49	eth0	1	1	0	28	46	0
1330184049111	25-02-2012 15:34:09	eth0	1	1	0	40	70	0
1330184585072	25-02-2012 15:43:05	eth0	22	24	0	14145	2692	0
1330184590155	25-02-2012 15:43:10	eth0	23	18	0	17488	1988	0
1330184595246	25-02-2012 15:43:15	eth0	1	1	0	52	70	0
1330184950996	25-02-2012 15:49:11	eth0	3	3	0	338	235	0
1330185705015	25-02-2012 16:01:45	eth0	18	21	0	1496	2440	0
1330185844364	25-02-2012 16:04:04	eth0	34	35	0	2589	3918	0
1330186746579	25-02-2012 16:19:07	eth0	7	6	0	590	651	0
1330188182572	25-02-2012 16:43:03	eth0	4	5	0	266	352	0
1330190350445	25-02-2012 17:19:10	eth0	4	6	0	259	446	0
1330191256430	25-02-2012 17:34:16	eth0	3	4	0	238	306	0
1330192212443	25-02-2012 17:50:12	eth0	6	6	0	550	717	0
1330193953530	25-02-2012 18:19:14	eth0	3	8	0	182	514	0

Fig: Example of information about the network activity extracted from the log file

time	screen_utilization
14:25:05	95
14:25:06	43
14:25:07	79
14:25:08	63
14:25:09	72
14:25:10	75
14:25:11	56
14:25:12	66
14:25:13	53
14:25:14	48
14:25:15	70
14:25:16	64
14:25:17	87
14:25:18	92
14:25:19	43
14:25:20	43
14:25:21	77
14:25:22	89
14:25:23	81
14:25:24	65
14:25:25	69
14:25:26	82
14:25:27	42
14:25:28	64

Fig: Example of measures (in %) of the screen utilization extracted from the log file

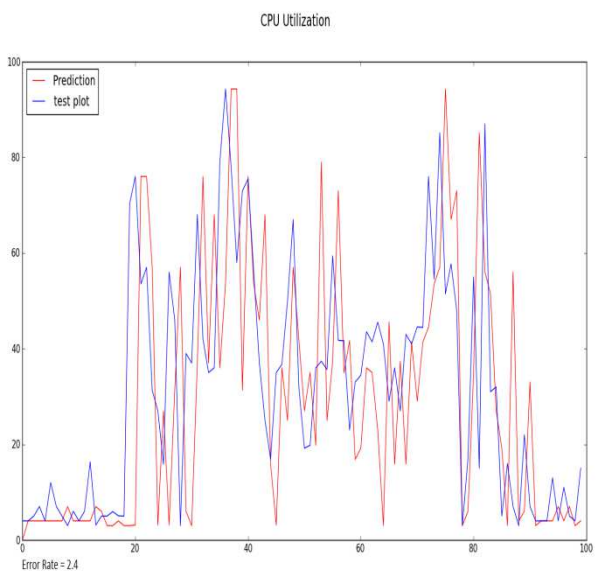
```
Estimated y:  3.03      Correct y:  8.0         Absolute Error:  4.97
Estimated y:  5.0       Correct y:  15.0        Absolute Error:  10.0
Estimated y:  4.0       Correct y:  6.0         Absolute Error:  2.0
Estimated y:  3.0       Correct y:  25.0        Absolute Error:  22.0
Estimated y:  27.0      Correct y:  6.0         Absolute Error:  21.0
Estimated y:  3.0       Correct y:  5.0         Absolute Error:  2.0
Estimated y:  3.0       Correct y:  9.0         Absolute Error:  6.0
Estimated y:  4.0       Correct y:  6.0         Absolute Error:  2.0
Estimated y:  3.0       Correct y:  6.0         Absolute Error:  3.0
Estimated y:  3.0       Correct y:  4.0         Absolute Error:  1.0
Estimated y:  4.0       Correct y:  13.0        Absolute Error:  9.0
Estimated y:  11.0      Correct y:  6.0         Absolute Error:  5.0
Estimated y:  3.0       Correct y:  76.0        Absolute Error:  73.0
Estimated y:  41.0      Correct y:  24.0        Absolute Error:  17.0
Estimated y:  27.0      Correct y:  6.0         Absolute Error:  21.0
Estimated y:  3.0       Correct y:  8.0         Absolute Error:  5.0
Estimated y:  5.0       Correct y:  6.0         Absolute Error:  1.0
Estimated y:  3.0       Correct y:  21.0        Absolute Error:  18.0
Estimated y:  5.0       Correct y:  8.0         Absolute Error:  3.0
Estimated y:  5.0       Correct y:  4.0         Absolute Error:  1.0
Estimated y:  4.0       Correct y:  5.0         Absolute Error:  1.0
Estimated y:  3.0       Correct y:  5.0         Absolute Error:  2.0
Estimated y:  3.0       Correct y:  46.0        Absolute Error:  43.0
Estimated y:  69.7      Correct y:  72.73       Absolute Error:  3.03
Estimated y:  47.0      Correct y:  40.59       Absolute Error:  6.41
Estimated y:  5.0       Correct y:  5.0         Absolute Error:  0.0
Estimated y:  3.0       Correct y:  5.0         Absolute Error:  2.0
Estimated y:  3.0       Correct y:  5.0         Absolute Error:  2.0
Estimated y:  3.0       Correct y:  4.0         Absolute Error:  1.0
Estimated y:  4.0       Correct y:  8.0         Absolute Error:  4.0
Standard error:  23.8412054435
```

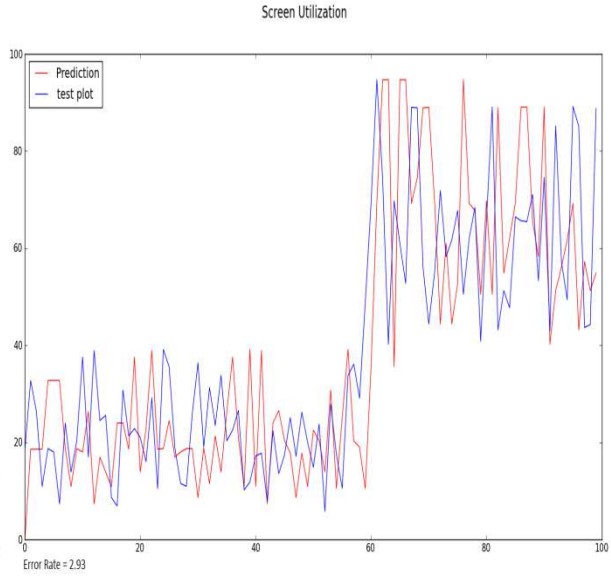
Fig: Example of output of the 1-NN regression script

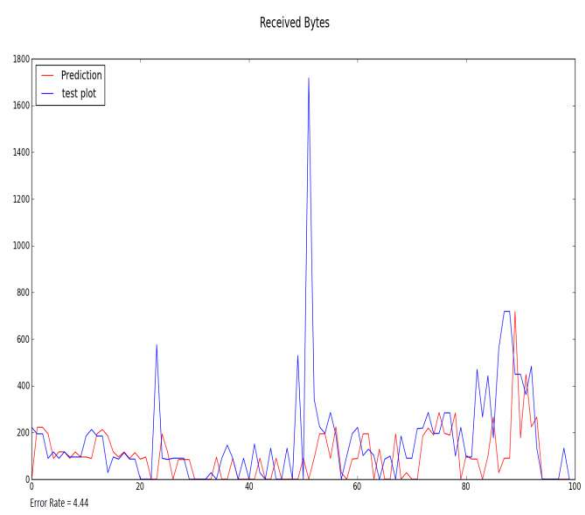
Appendix D - Results of the 1-NN regression model for the 5 users

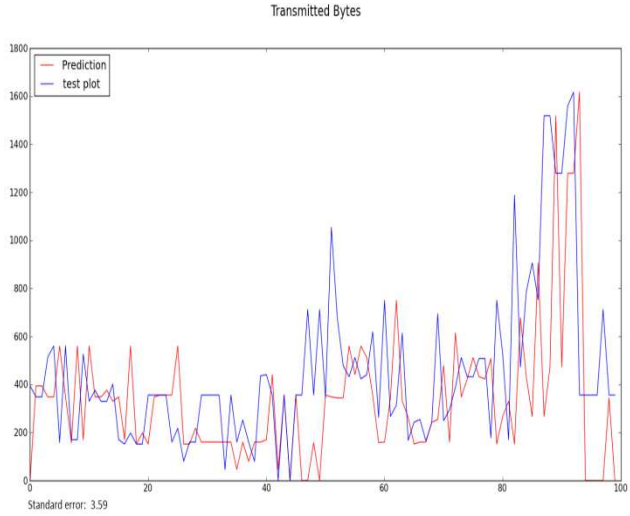
Results based on 1 second historical measures

User 1:

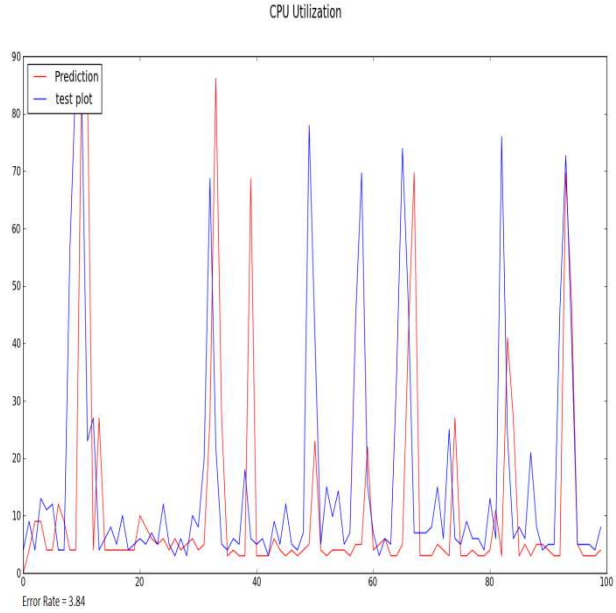


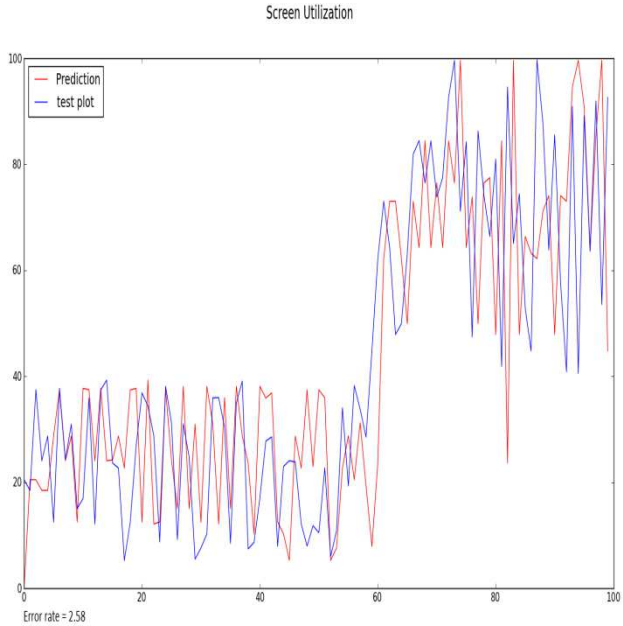


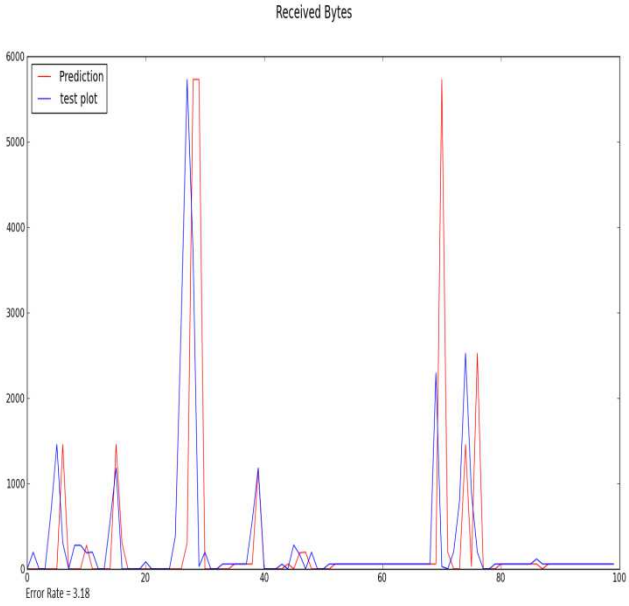


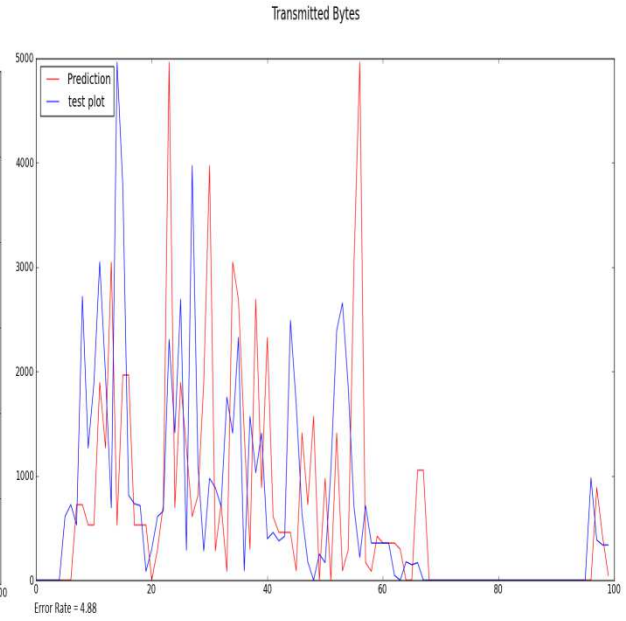


User 2:

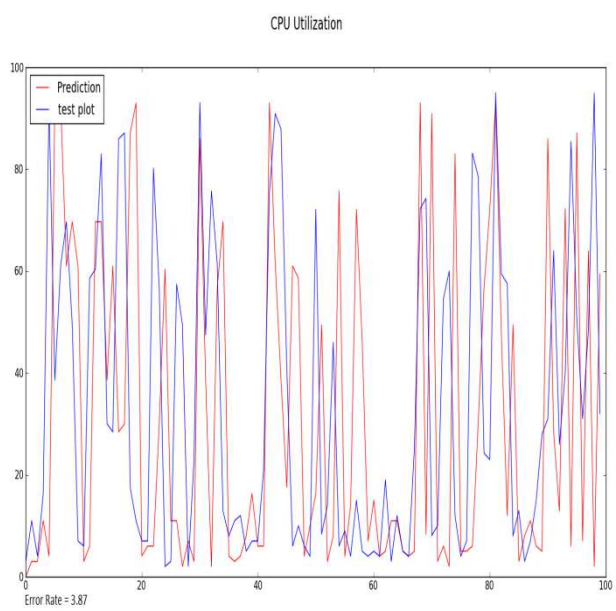

CPU Utilization
Prediction
test plot
Error Rate = 3.84

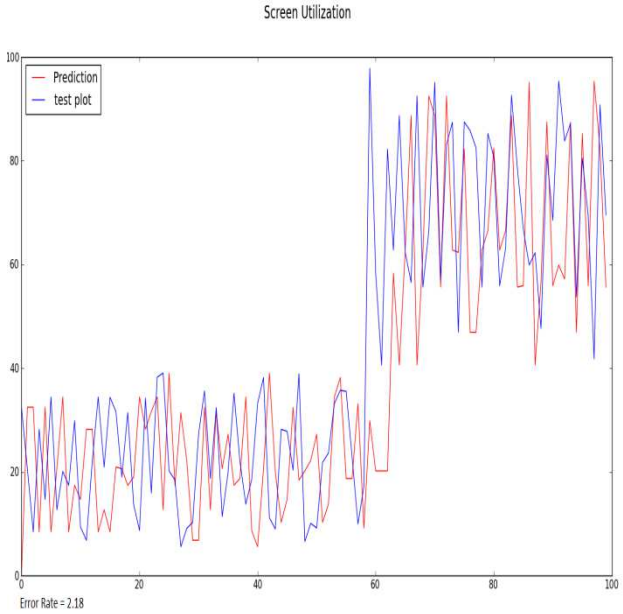

Screen Utilization
Prediction
test plot
Error rate = 2.58

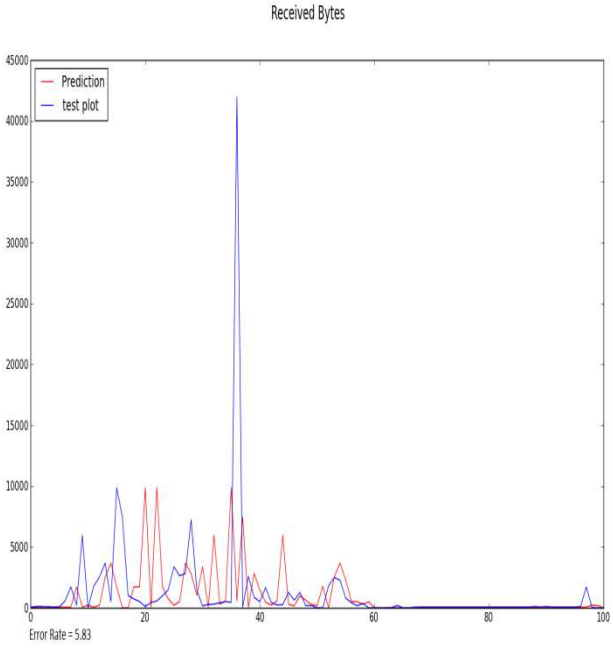

Received Bytes
Prediction
test plot
Error Rate = 3.18

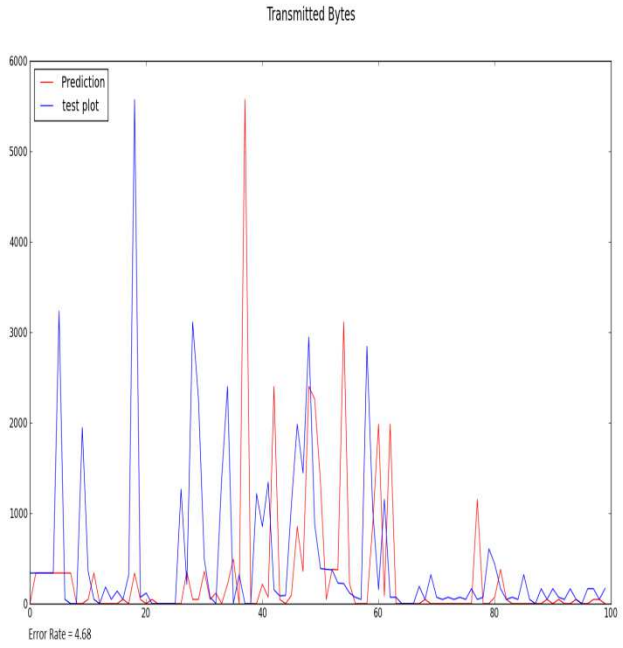

Transmitted Bytes
Prediction
test plot
Error Rate = 4.88

User 3:

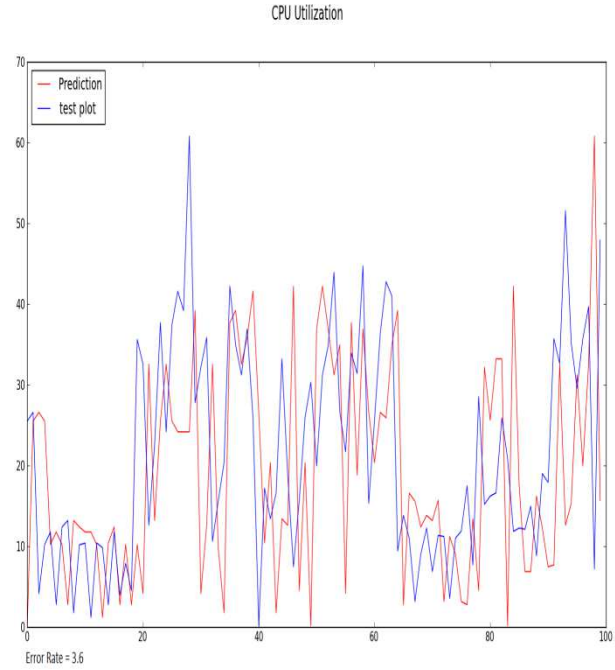

CPU Utilization
Prediction
test plot
Error Rate = 3.87

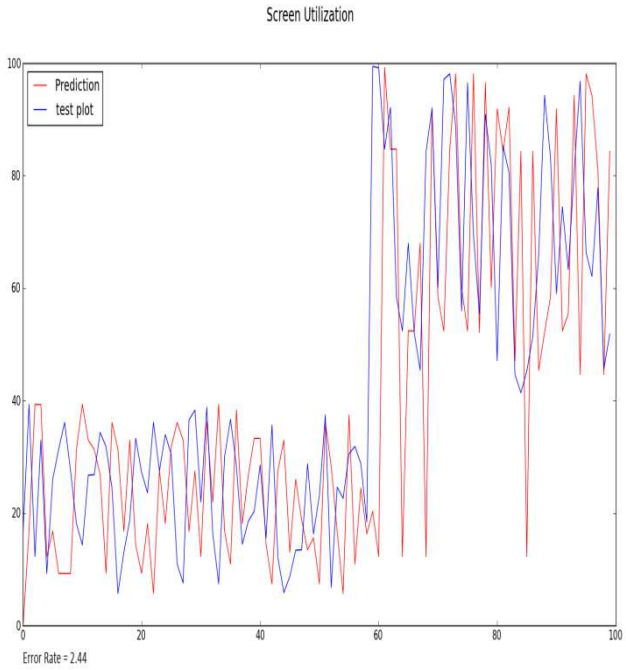

Screen Utilization
Prediction
test plot
Error Rate = 2.18

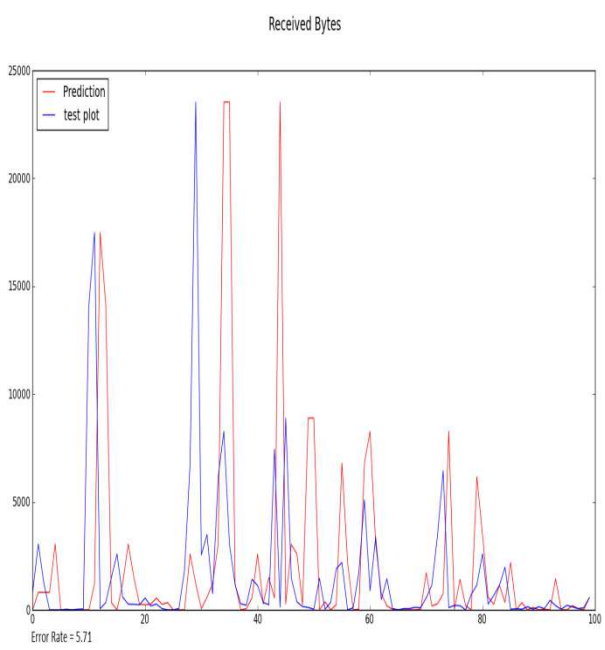

Received Bytes
Prediction
test plot
Error Rate = 5.83

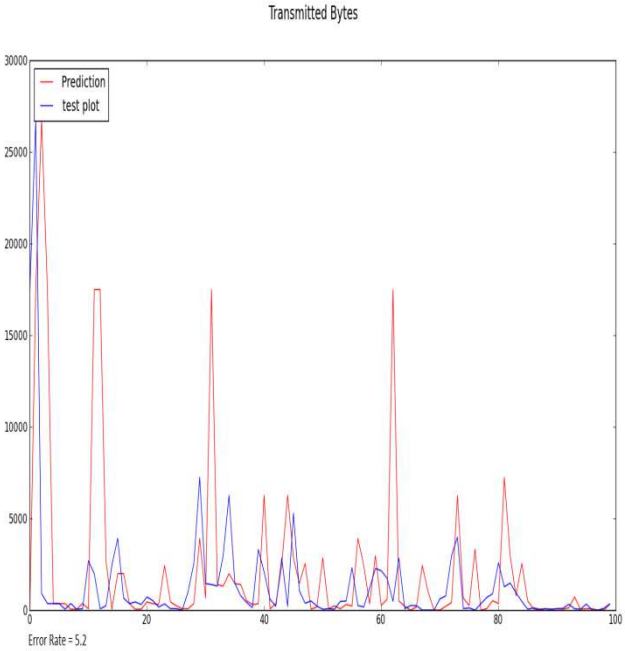

Transmitted Bytes
Prediction
test plot
Error Rate = 4.68

User 4:

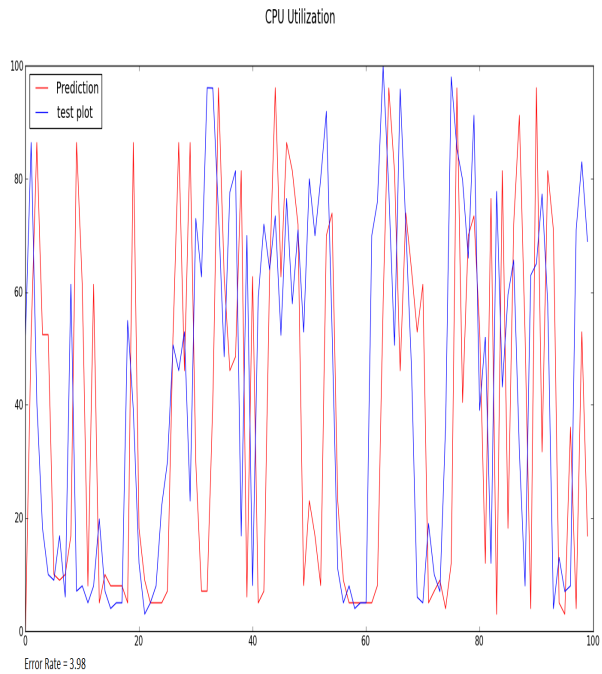

CPU Utilization
Prediction
test plot
Error Rate = 3.6

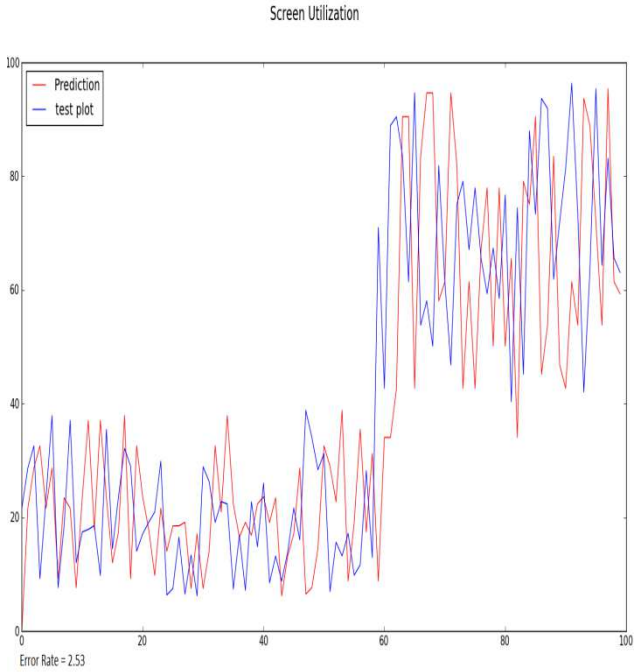

Screen Utilization
Prediction
test plot
Error Rate = 2.44

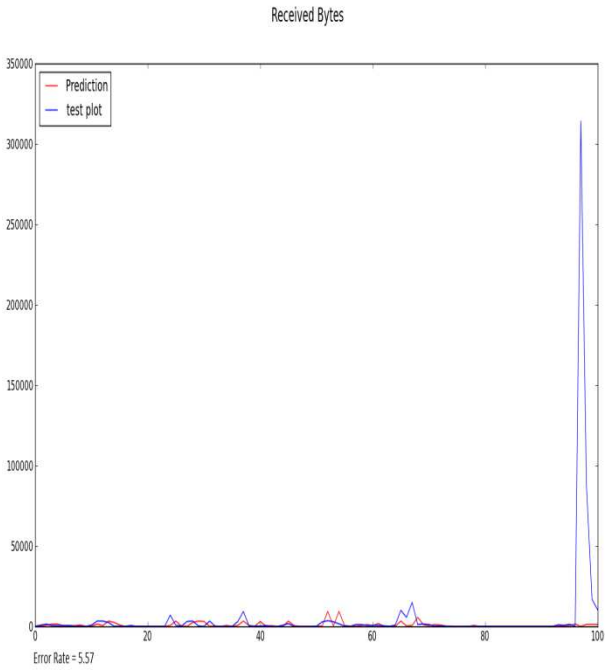

Received Bytes
Prediction
test plot
Error Rate = 5.71

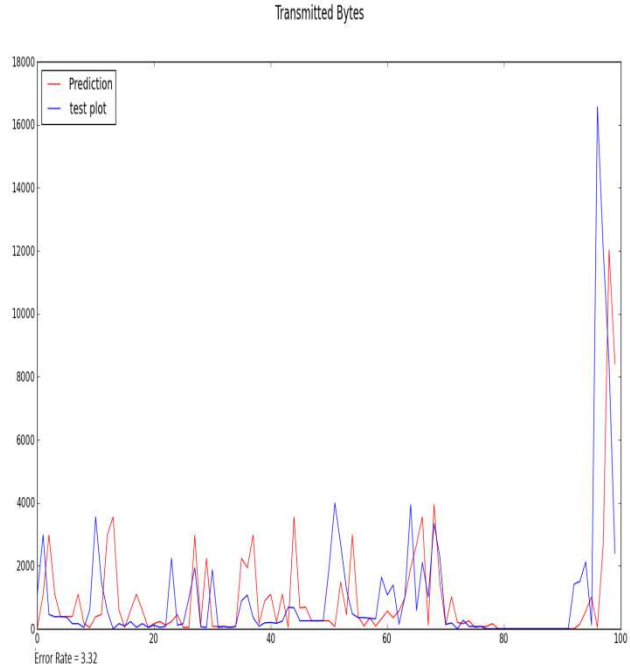

Transmitted Bytes
Prediction
test plot
Error Rate = 5.2

User 5:


CPU Utilization
Prediction
test plot
Error Rate = 3.98

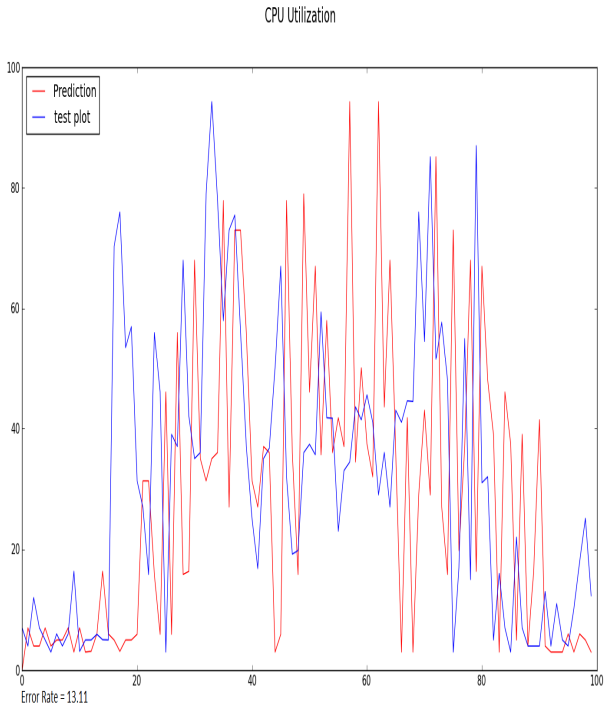

Screen Utilization
Prediction
test plot
Error Rate = 2.53

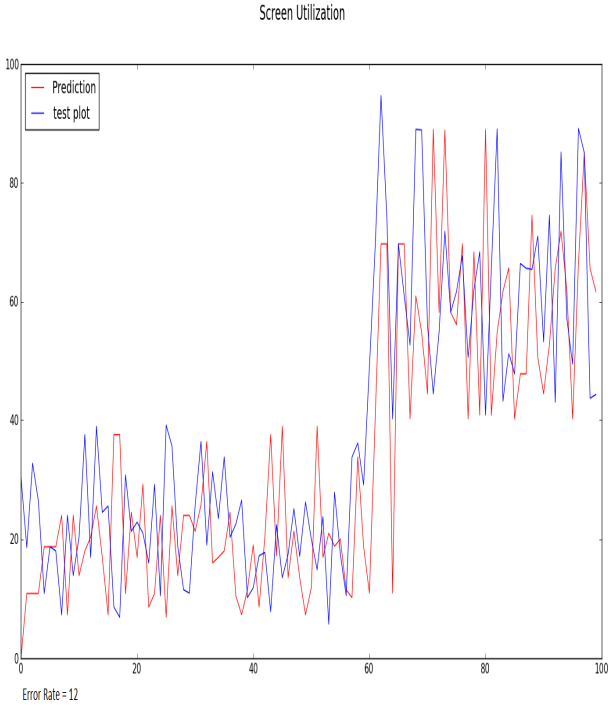

Received Bytes
Prediction
test plot
Error Rate = 5.57

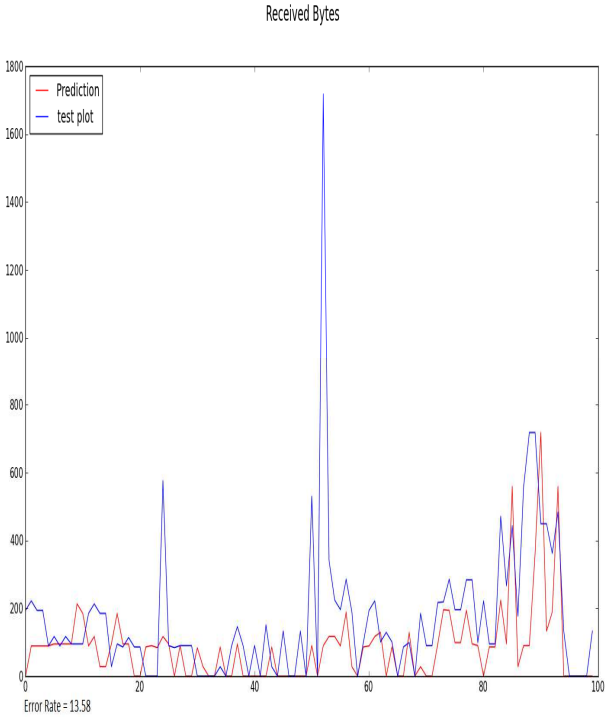

Transmitted Bytes
Prediction
test plot
Error Rate = 3.32

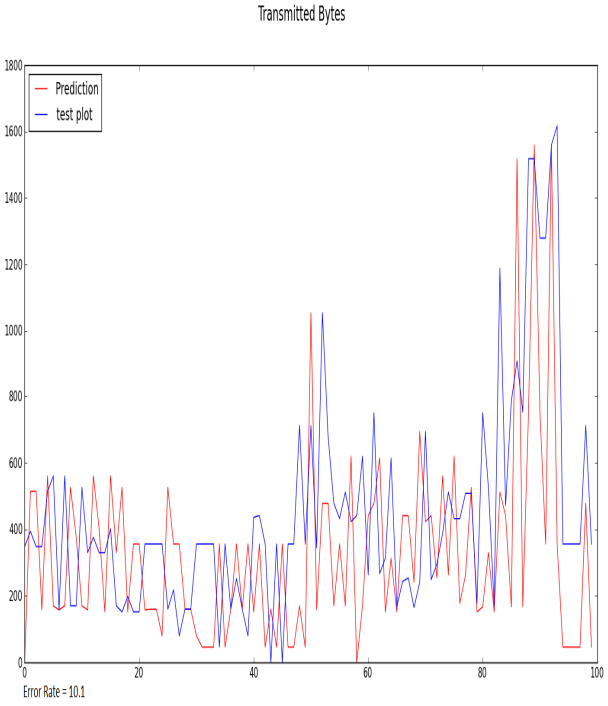
Results based on 5 second historical measures

User 1:

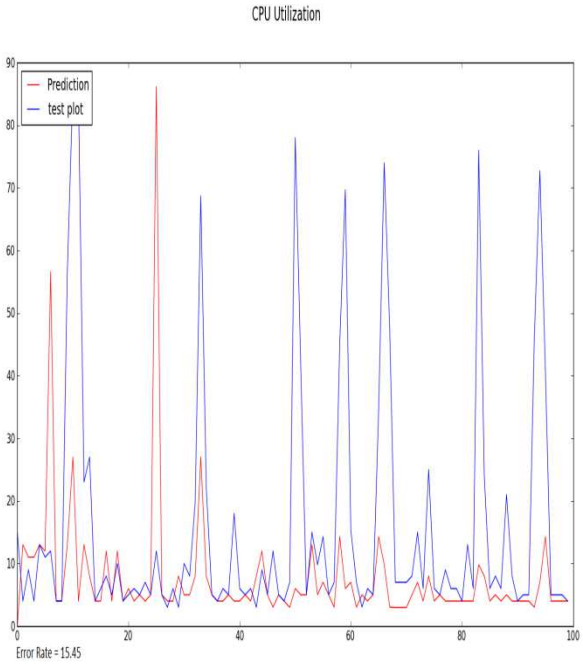


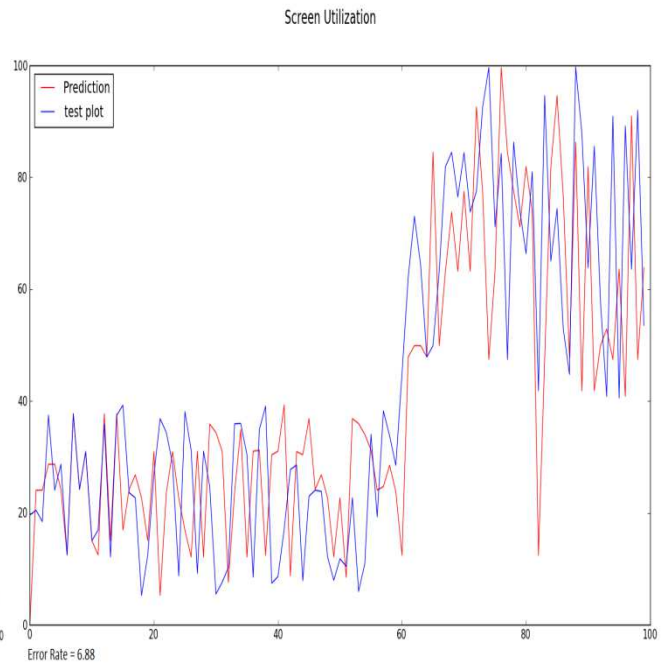


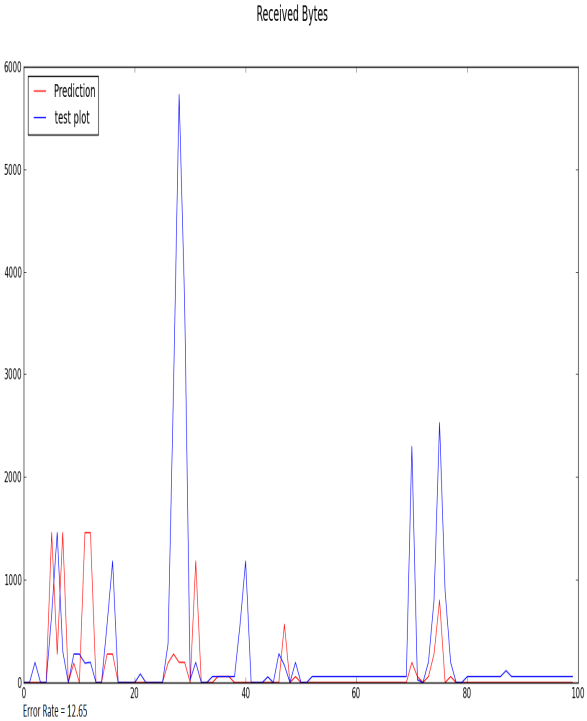


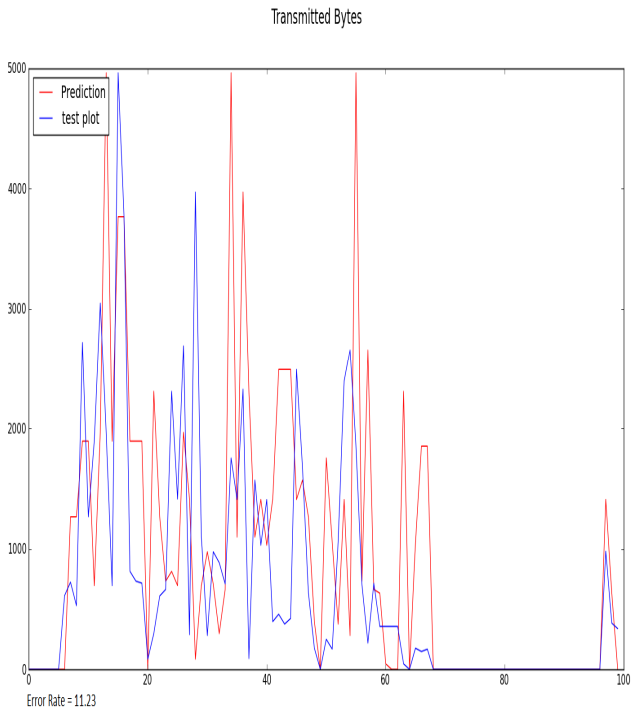


User 2

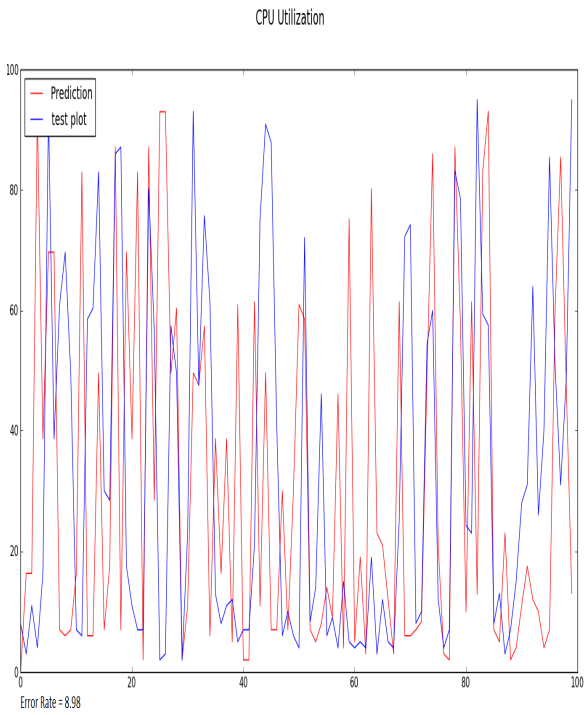

CPU Utilization
Prediction
test plot
Error Rate = 15.45

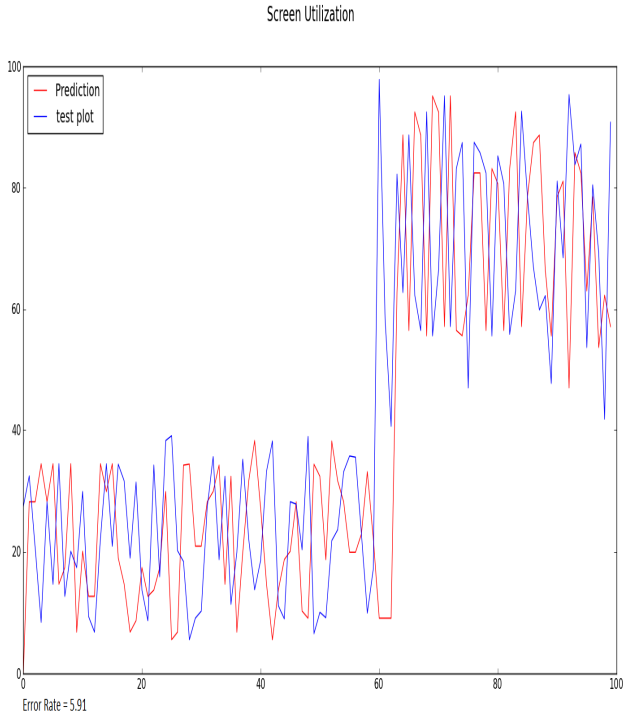

Screen Utilization
Prediction
test plot
Error Rate = 6.88

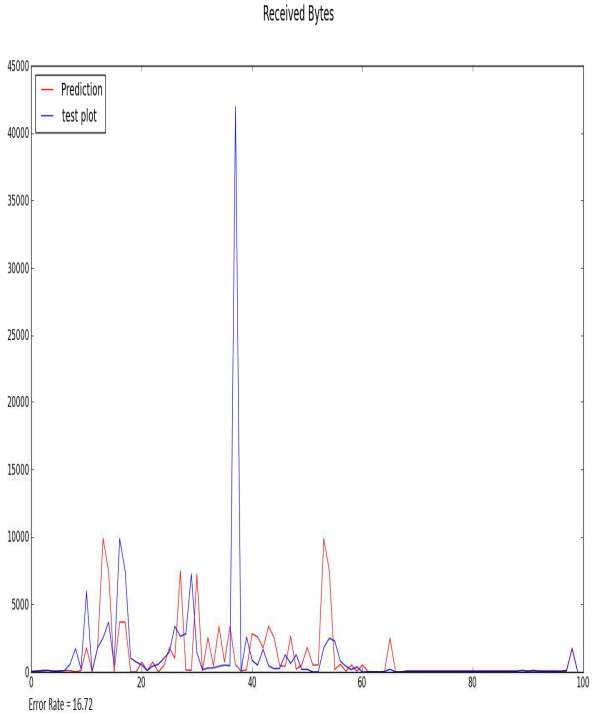

Received Bytes
Prediction
test plot
Error Rate = 12.65

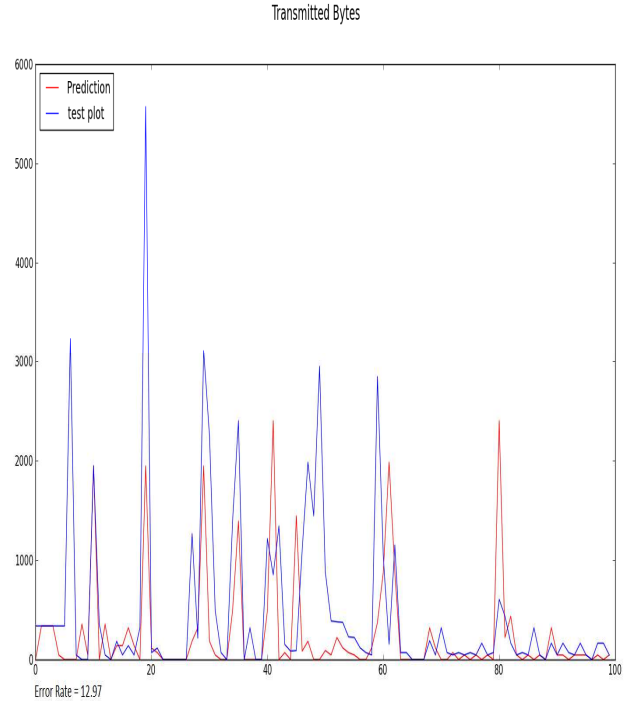

Transmitted Bytes
Prediction
test plot
Error Rate = 11.23

User 3

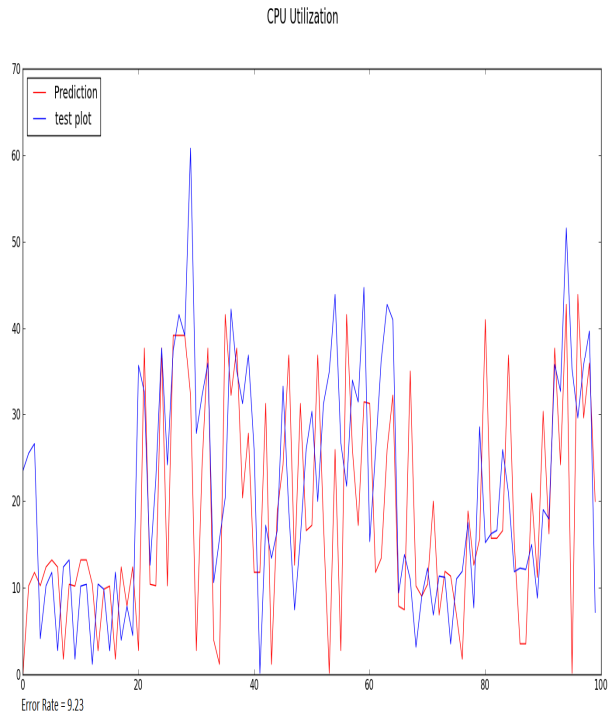

CPU Utilization
Prediction
test plot
Error Rate = 8.98

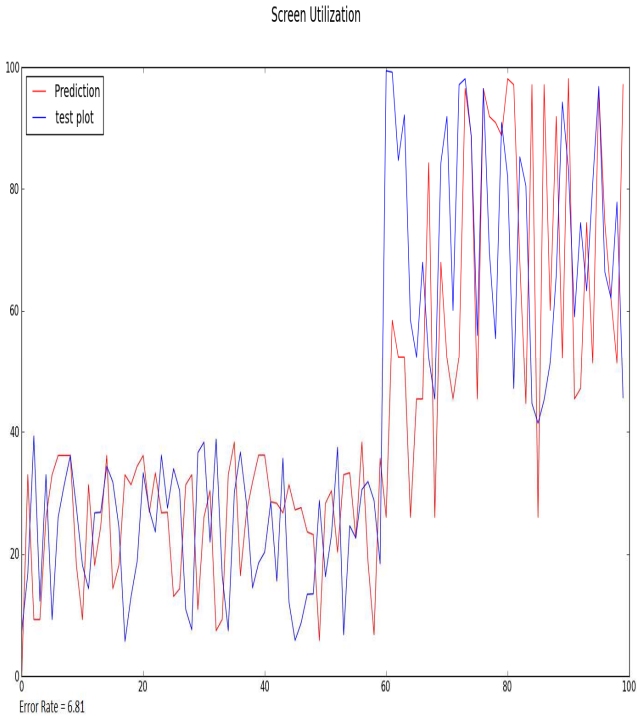

Screen Utilization
Prediction
test plot
Error Rate = 5.91

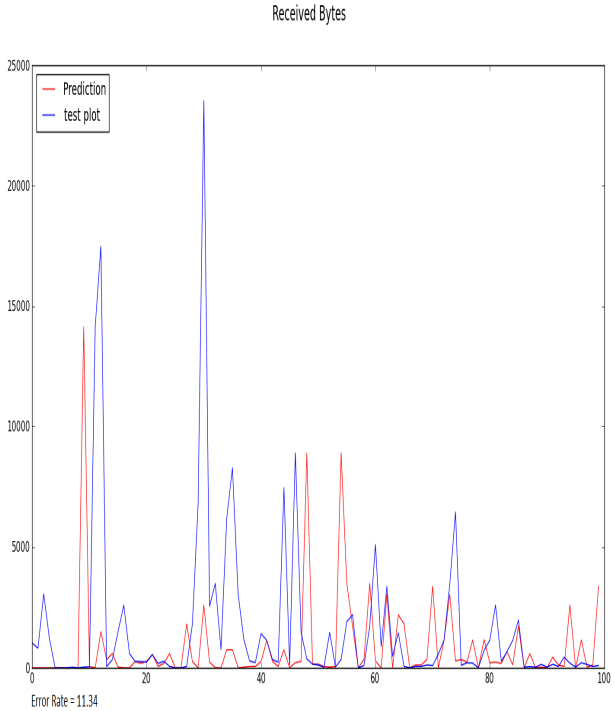

Received Bytes
Prediction
test plot
Error Rate = 16.72

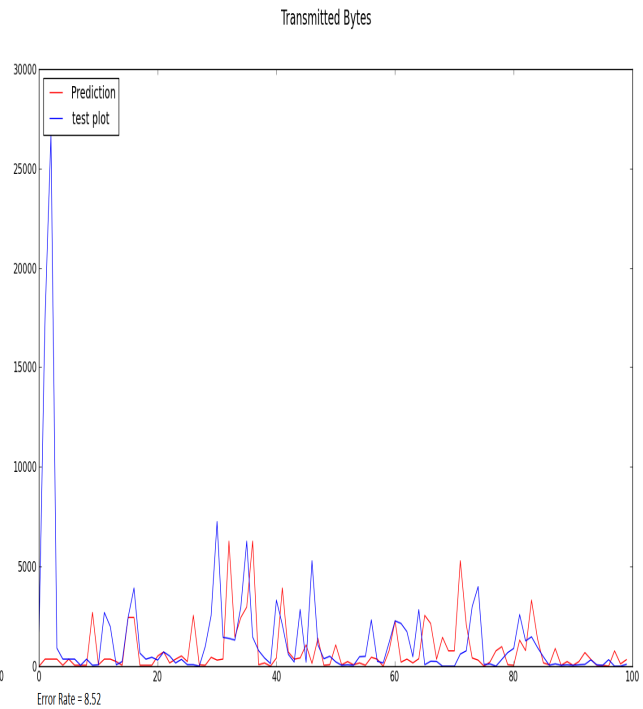

Transmitted Bytes
Prediction
test plot
Error Rate = 12.97

User4

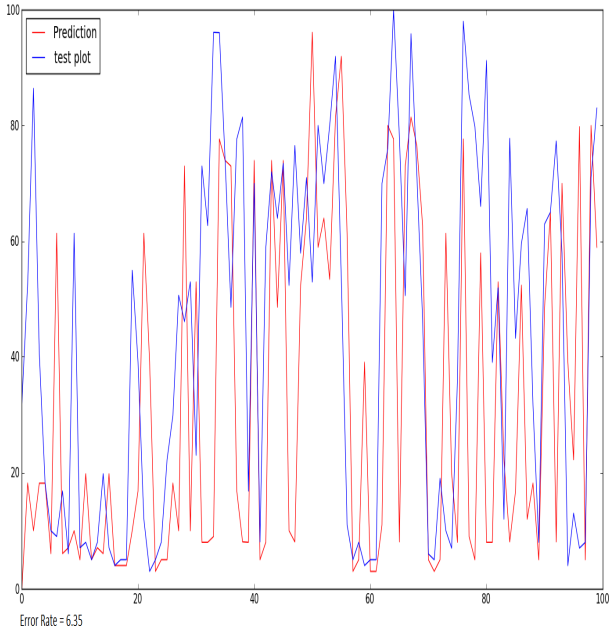

CPU Utilization
Prediction
test plot
Error Rate = 9.23

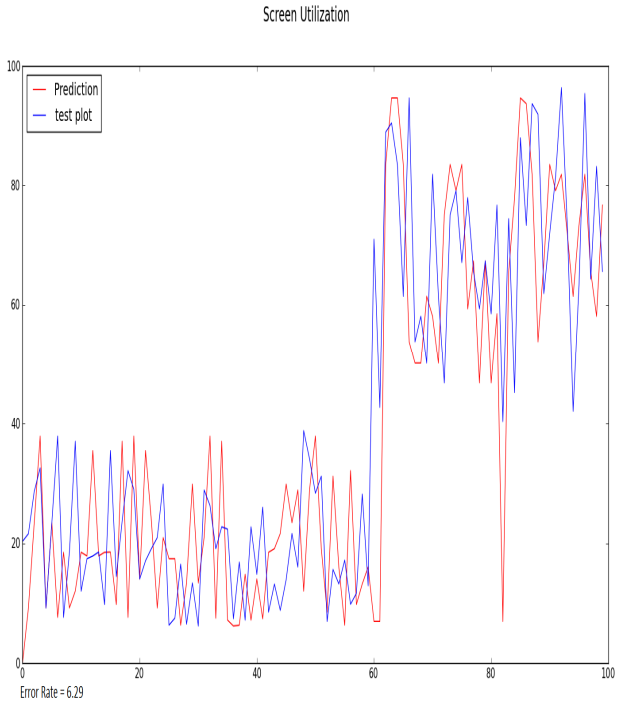

Screen Utilization
Prediction
test plot
Error Rate = 6.81

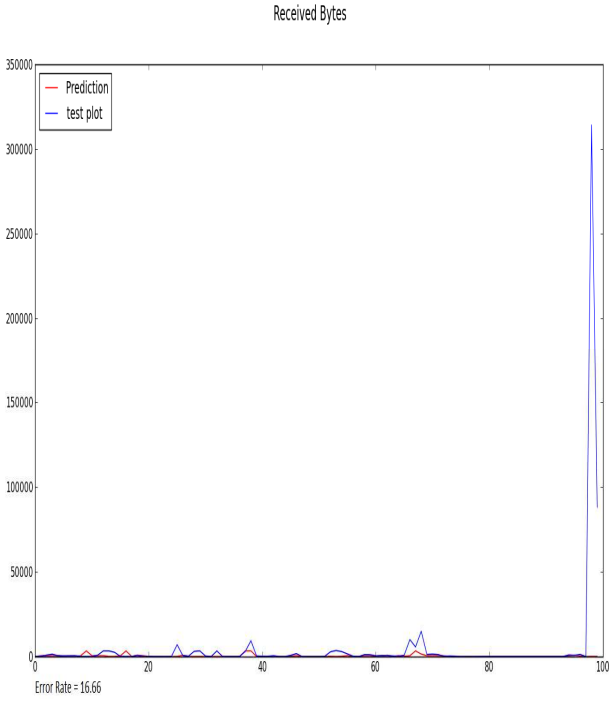

Received Bytes
Prediction
test plot
Error Rate = 11.34

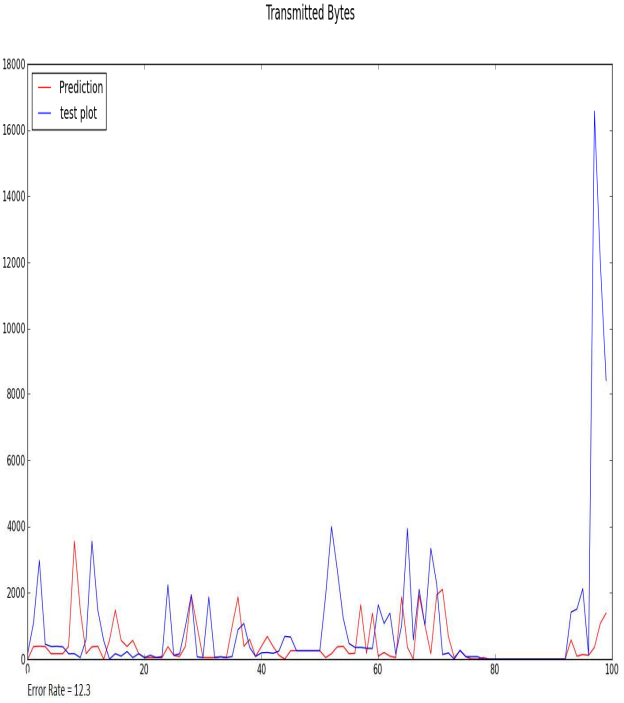

Transmitted Bytes
Prediction
test plot
Error Rate = 8.52

User 5


CPU Utilization
Prediction
test plot
Error Rate = 6.35

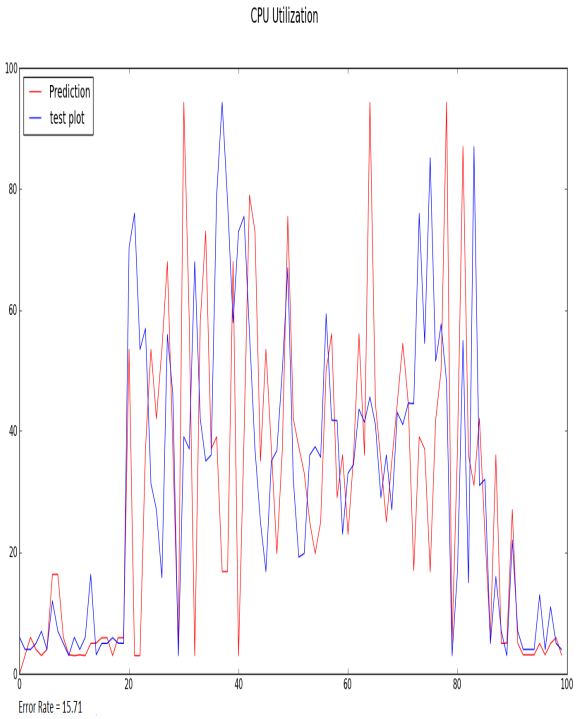

Screen Utilization
Prediction
test plot
Error Rate = 6.29

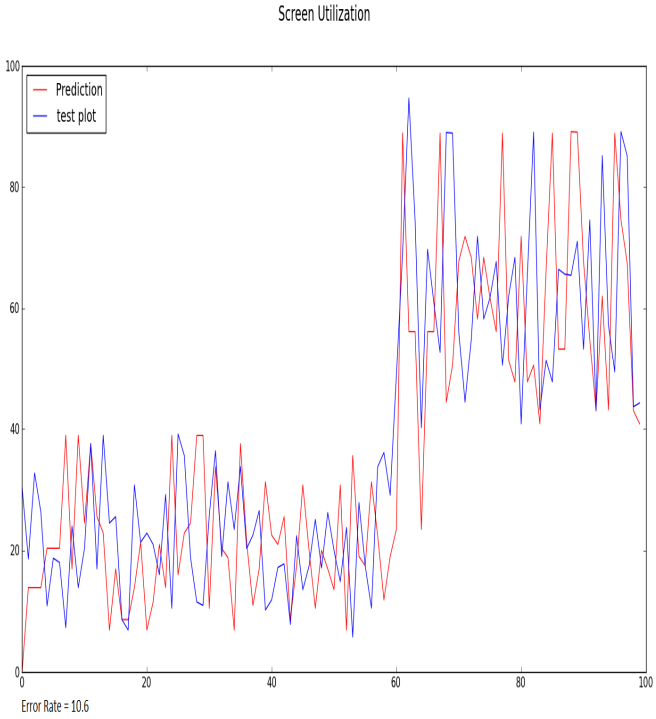

Received Bytes
Prediction
test plot
Error Rate = 16.66

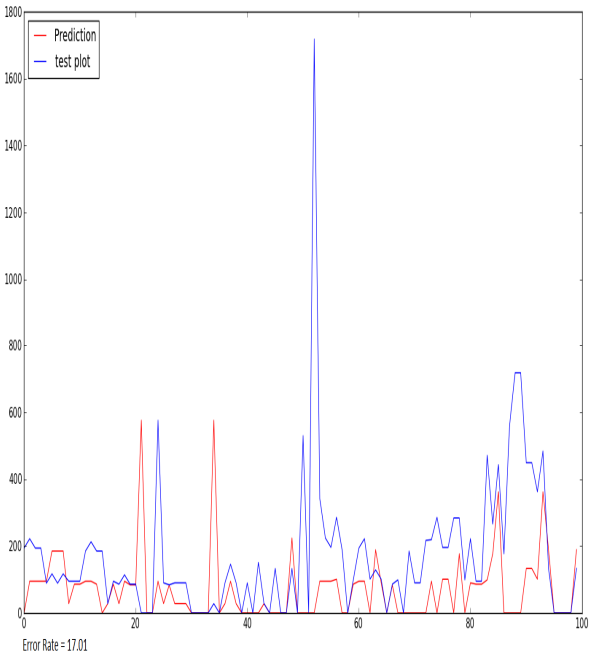

Transmitted Bytes
Prediction
test plot
Error Rate = 12.3

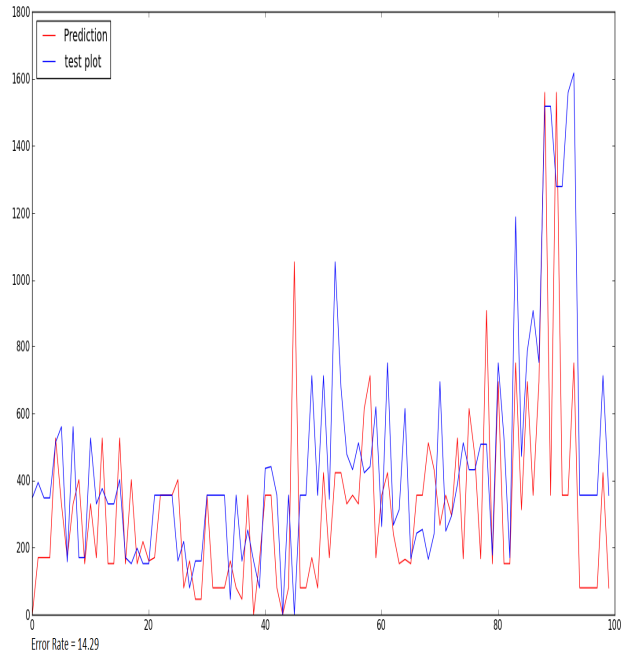
Results based on 10 second historical measures

User 1:

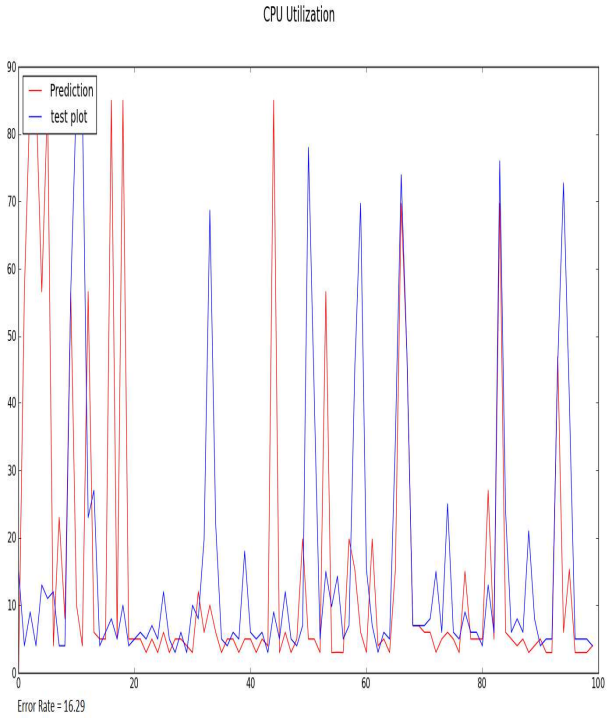


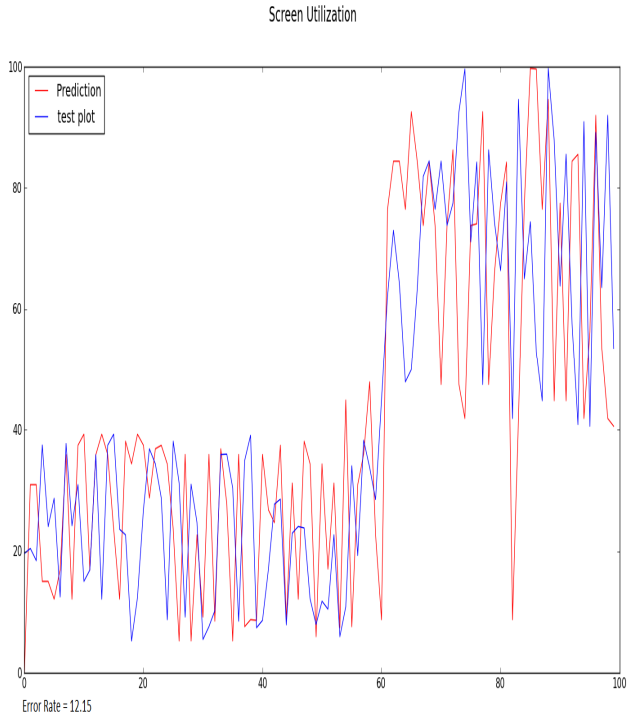


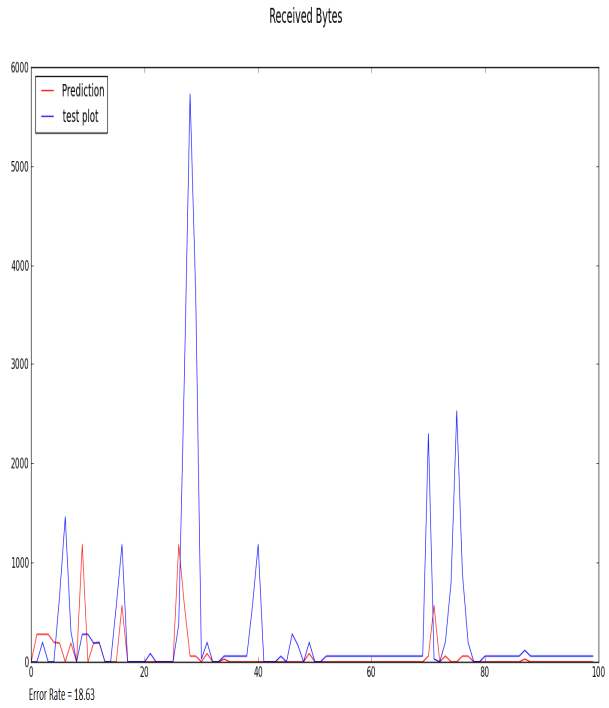


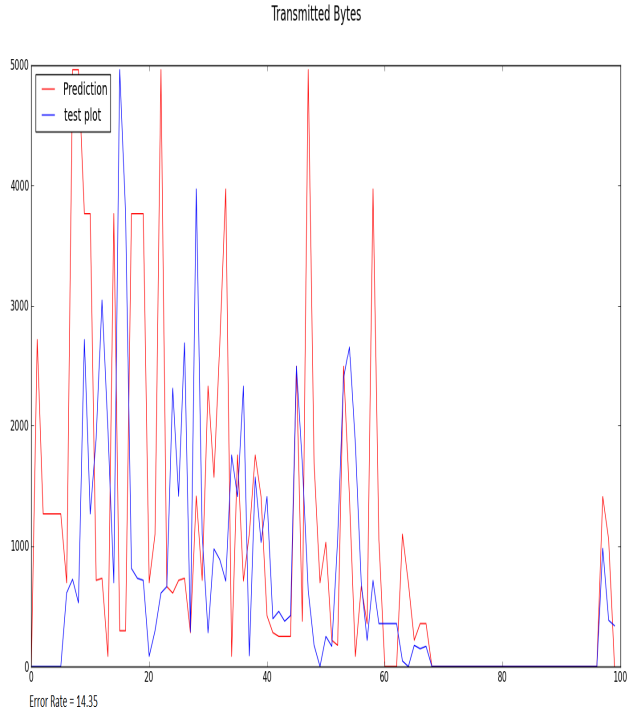


User 2

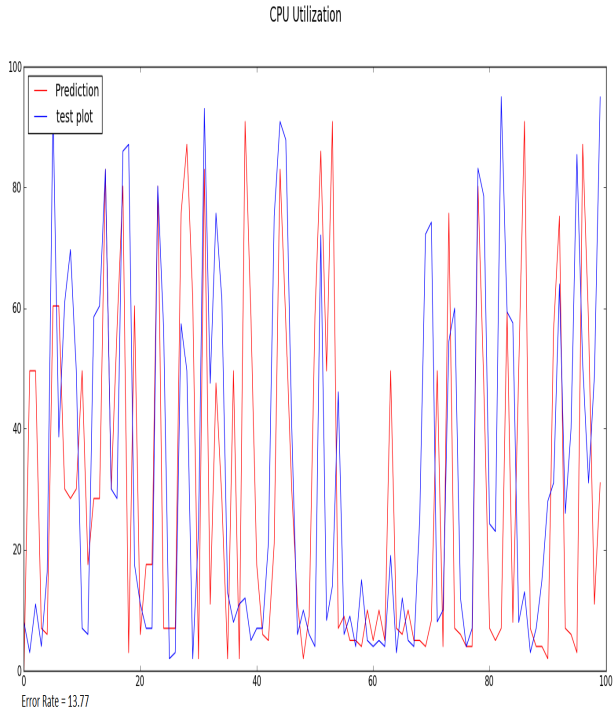

CPU Utilization
Prediction
test plot
Error Rate = 16.29

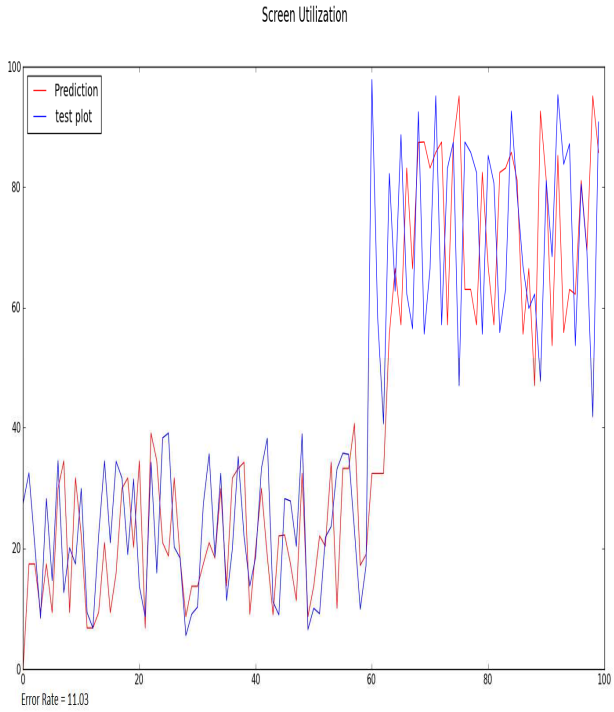

Screen Utilization
Prediction
test plot
Error Rate = 12.15

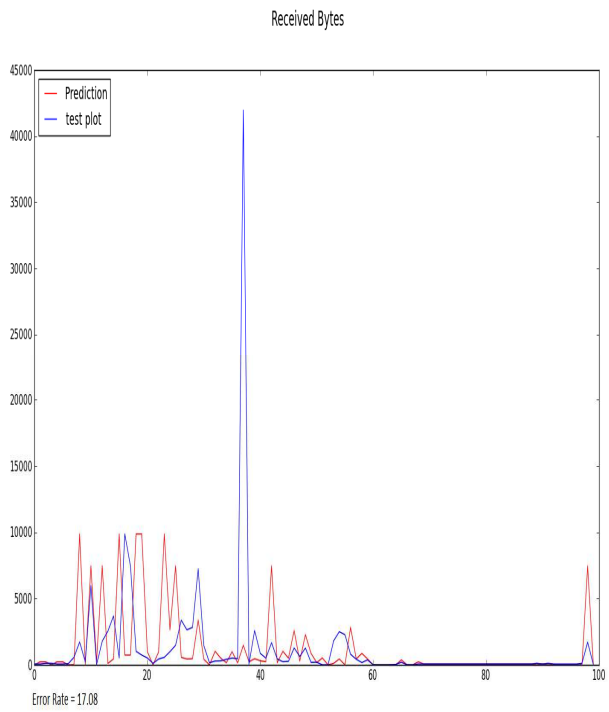

Received Bytes
Prediction
test plot
Error Rate = 18.63

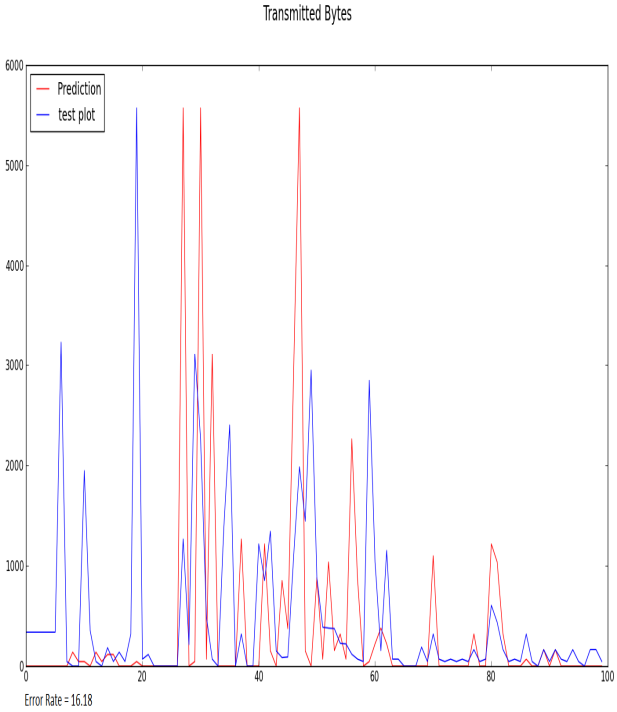

Transmitted Bytes
Prediction
test plot
Error Rate = 14.35

User 3

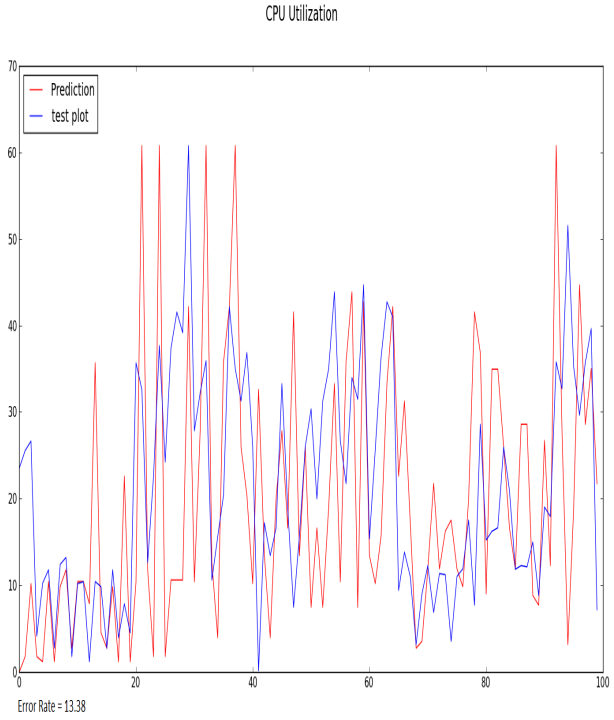

CPU Utilization
Prediction
test plot
Error Rate = 13.77

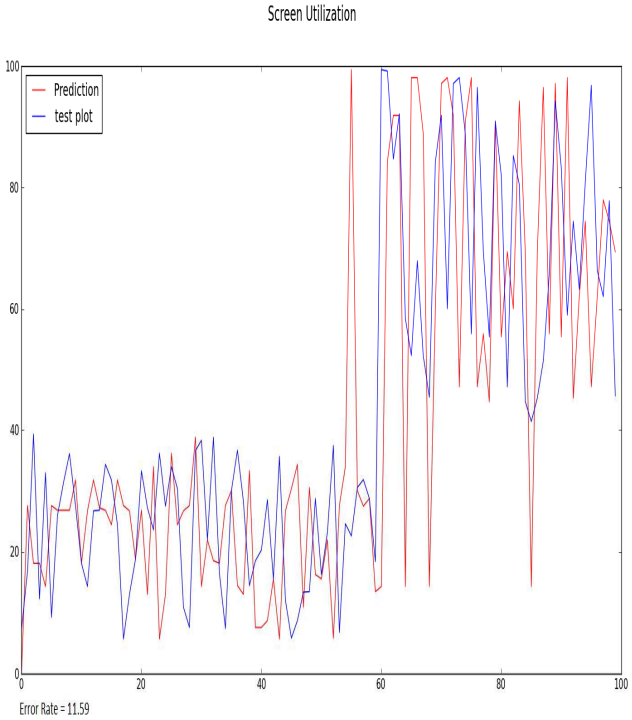

Screen Utilization
Prediction
test plot
Error Rate = 11.03

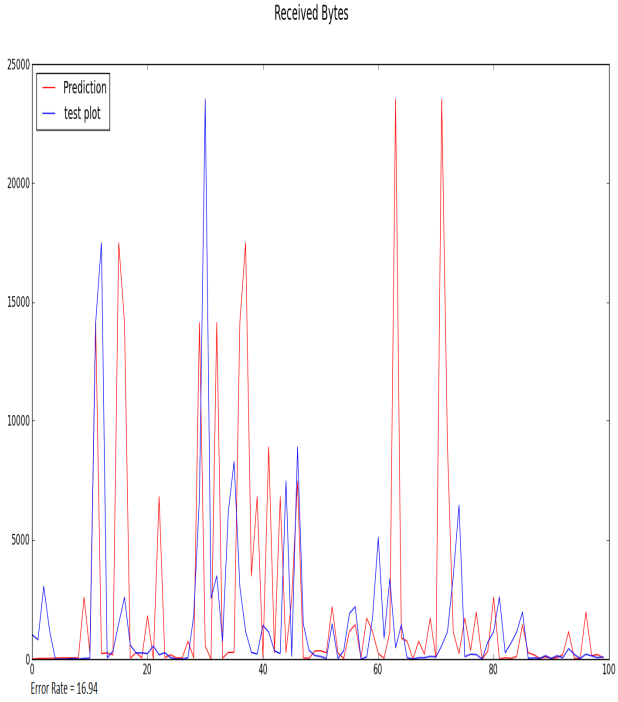

Received Bytes
Prediction
test plot
Error Rate = 17.08

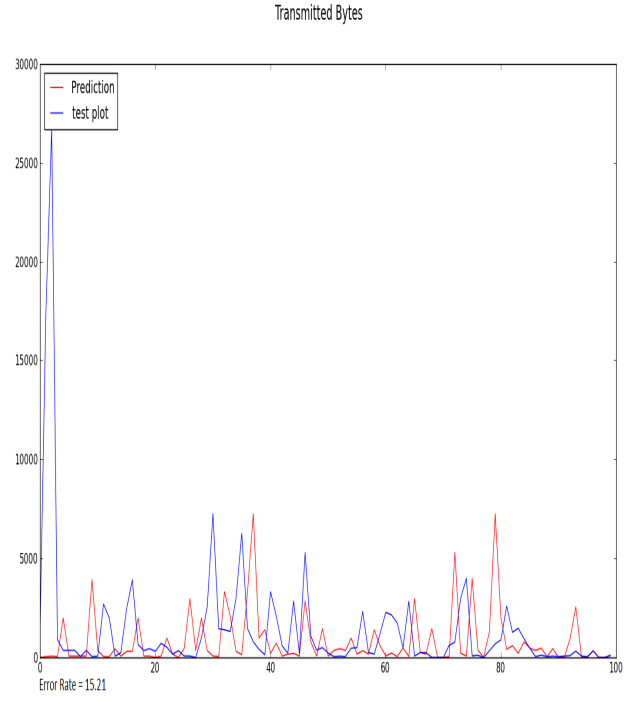

Transmitted Bytes
Prediction
test plot
Error Rate = 16.18

User 4

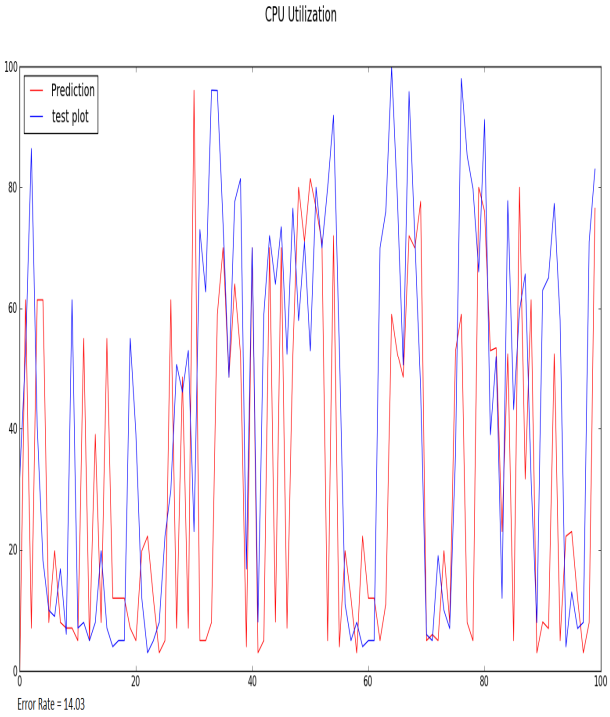

CPU Utilization
Prediction
test plot
Error Rate = 13.38

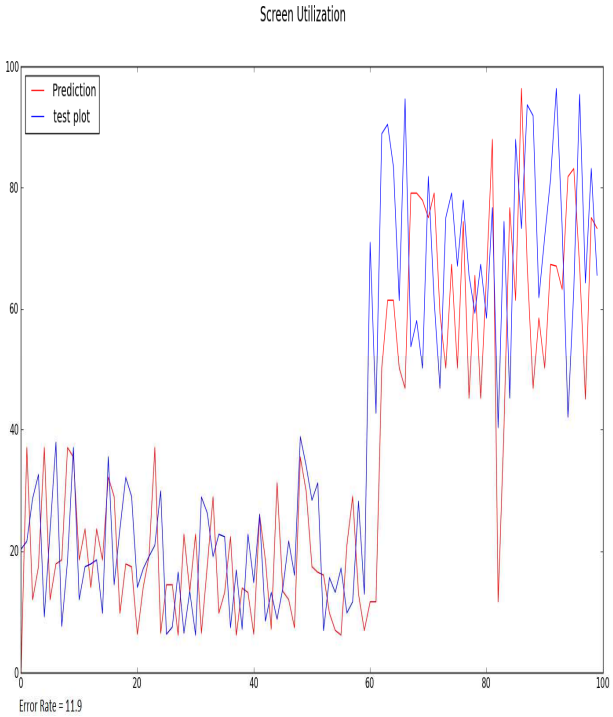

Screen Utilization
Prediction
test plot
Error Rate = 11.59

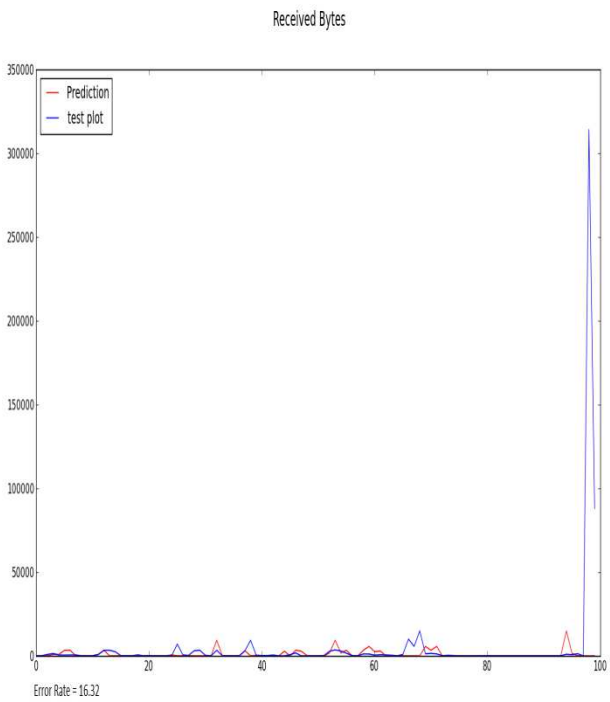

Received Bytes
Prediction
test plot
Error Rate = 16.94

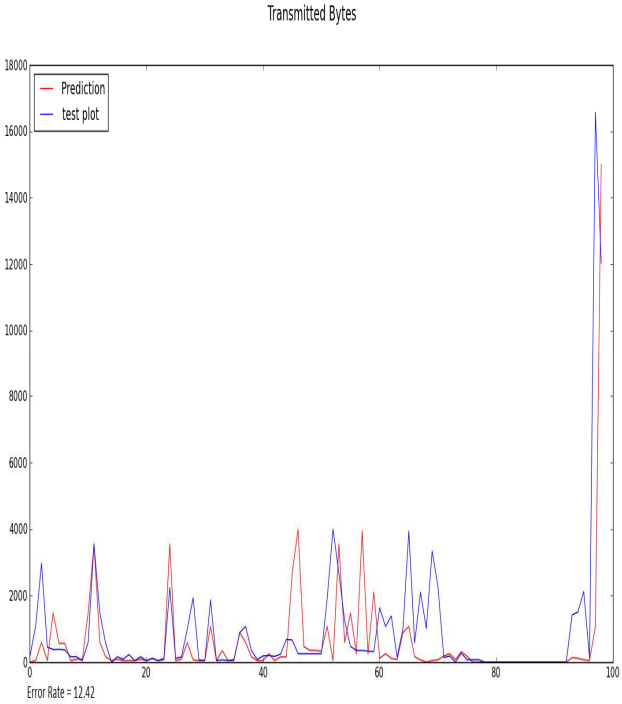

Transmitted Bytes
Prediction
test plot
Error Rate = 15.21

User 5


CPU Utilization
Prediction
test plot
Error Rate = 14.03


Screen Utilization
Prediction
test plot
Error Rate = 11.9

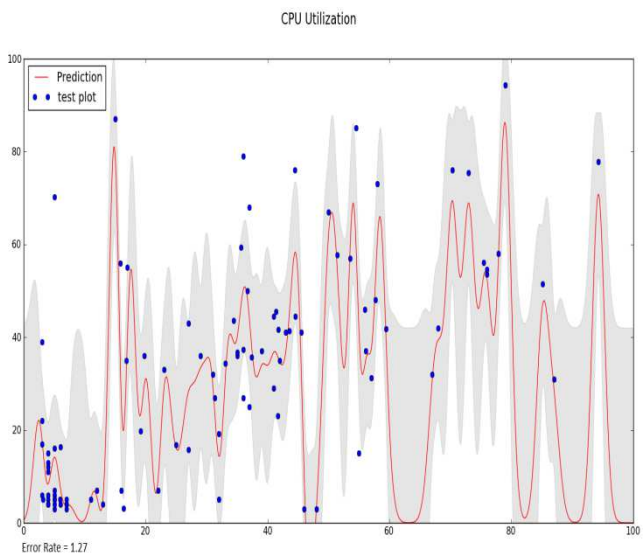

Received Bytes
Prediction
test plot
Error Rate = 16.32

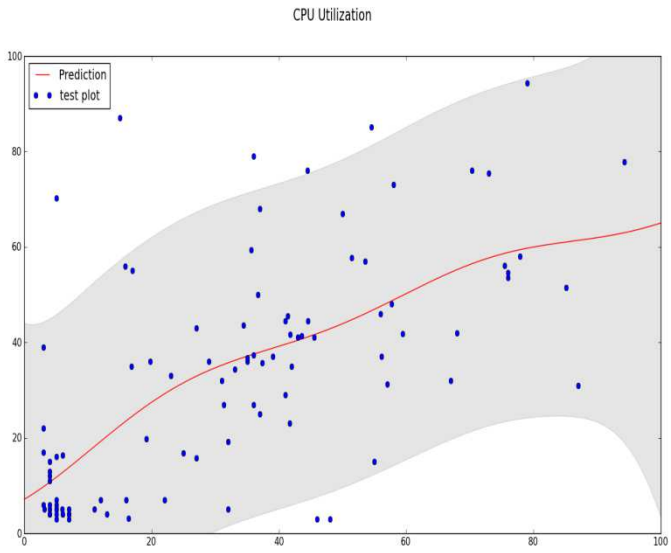

Transmitted Bytes
Prediction
test plot
Error Rate = 12.42

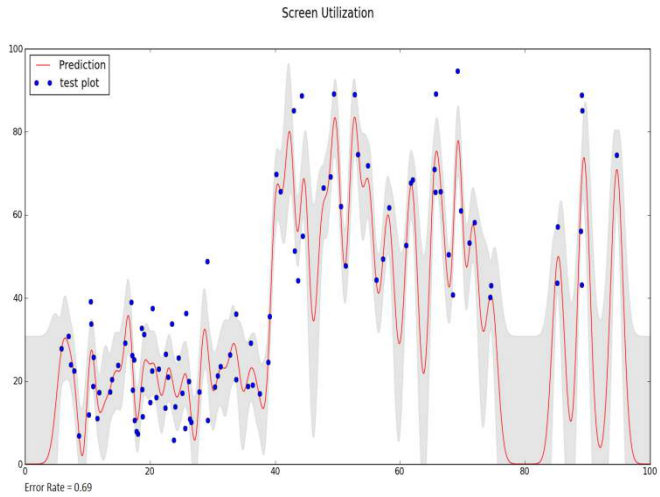
Appendix E - Results of the Gaussian process regression model for the 5 users

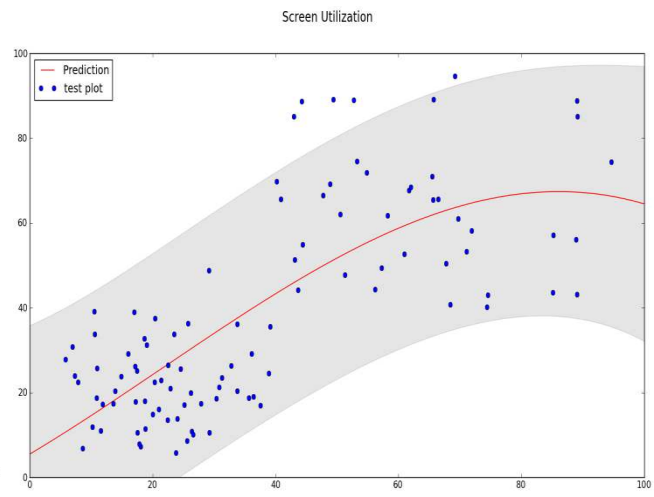
Results based on 1 second historical measures

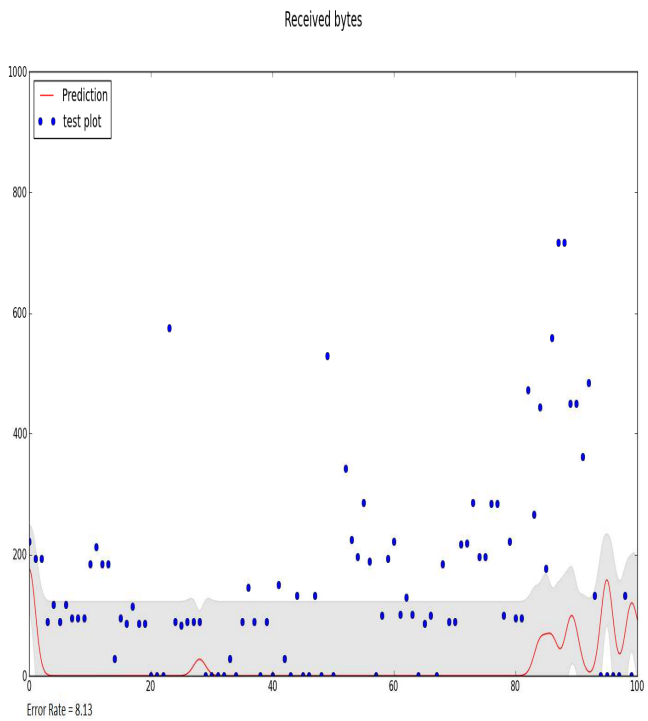
User 1:

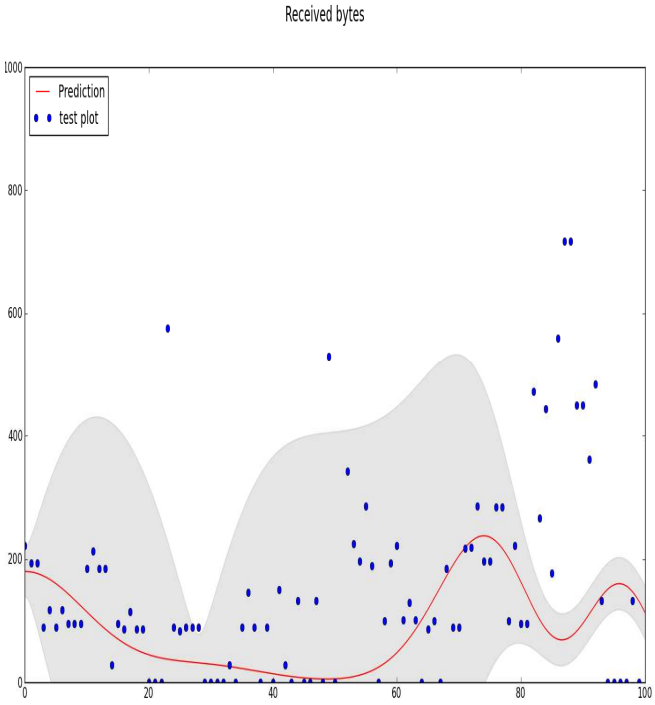


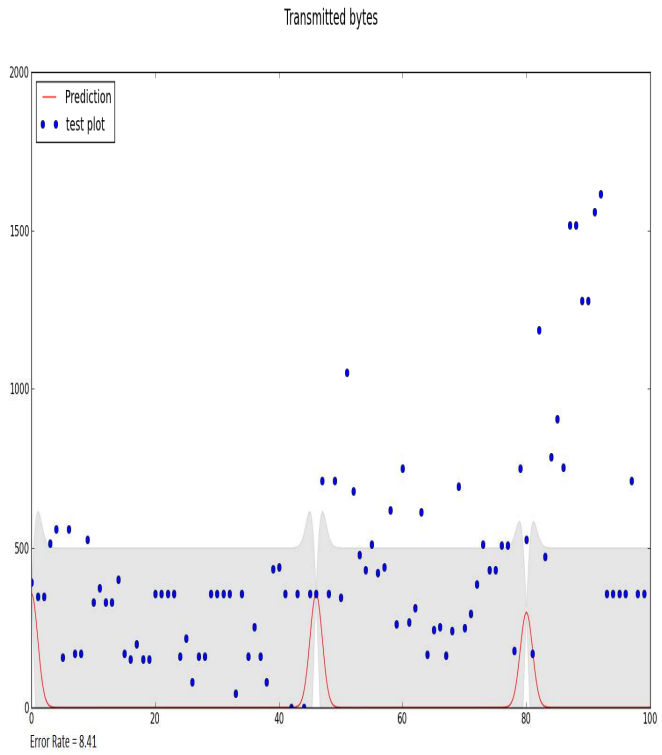


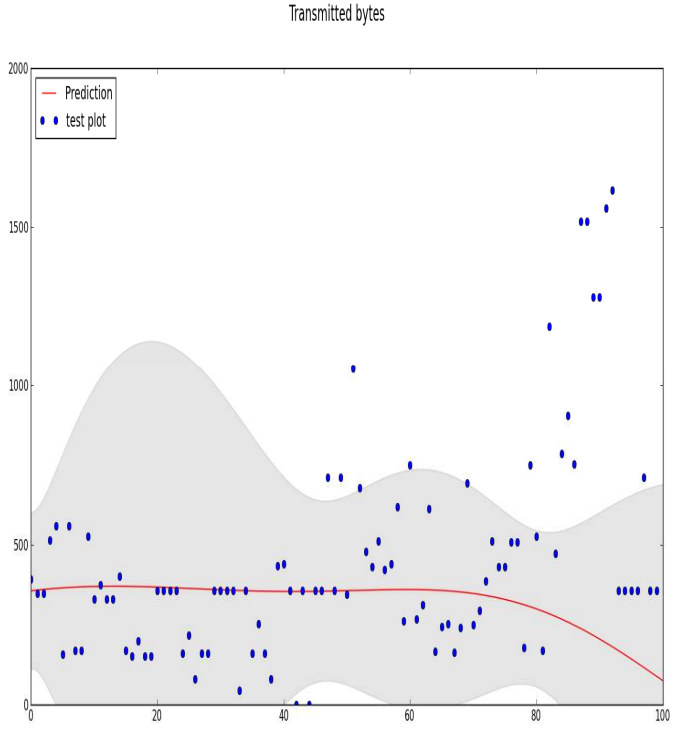




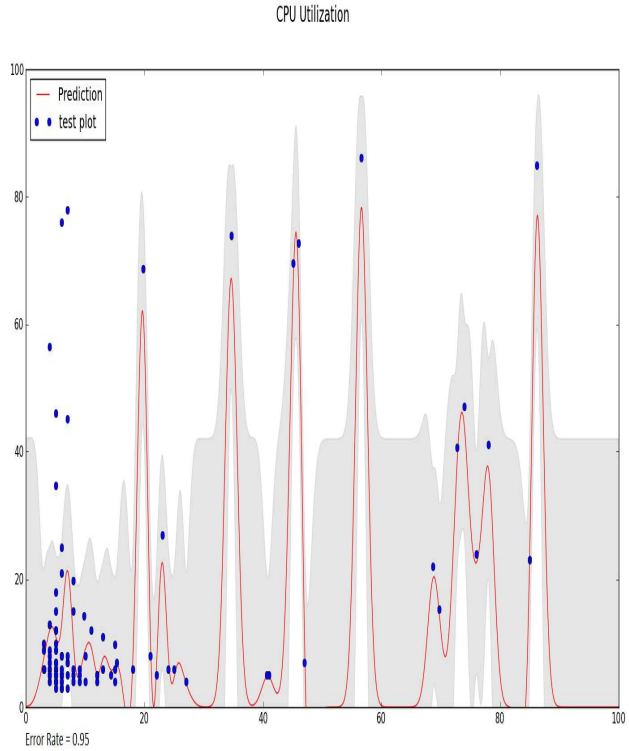

Received bytes
Prediction
test plot
Error Rate = 8.13

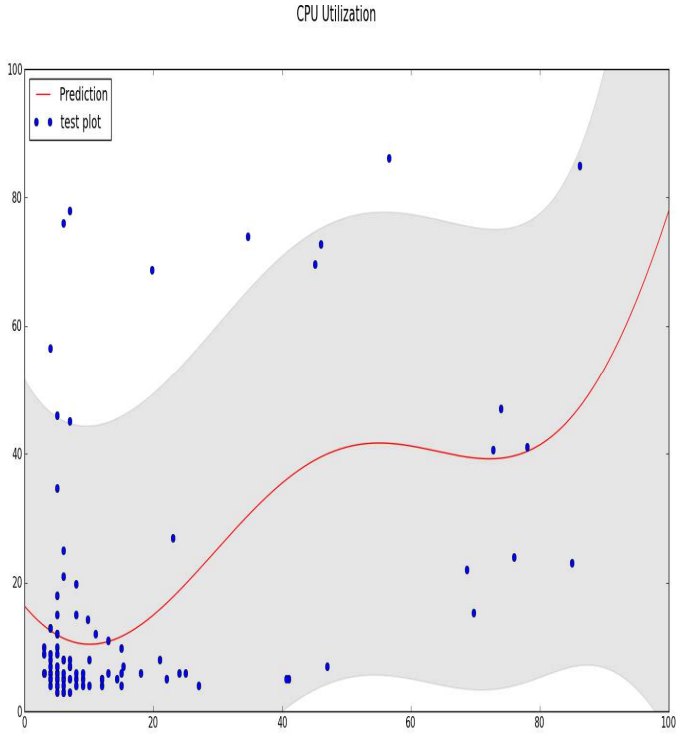

Received bytes
Prediction
test plot

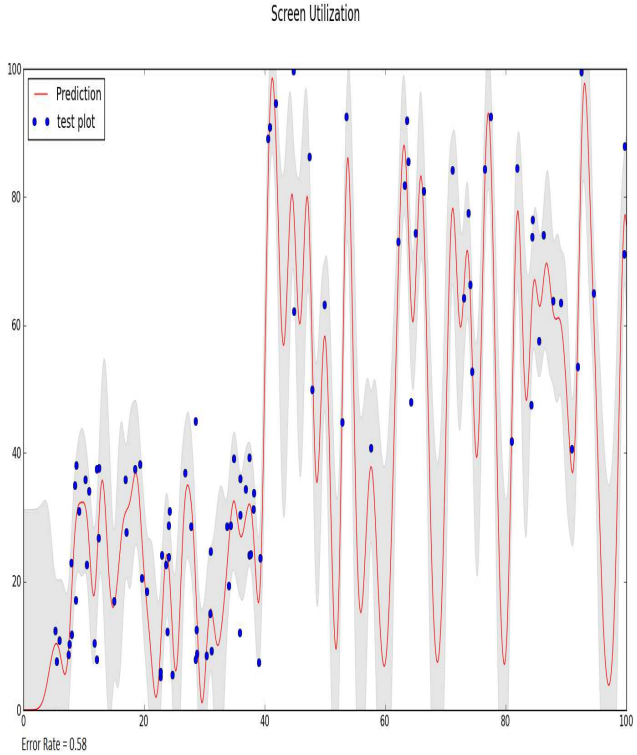

Transmitted bytes
Prediction
test plot
Error Rate = 8.41

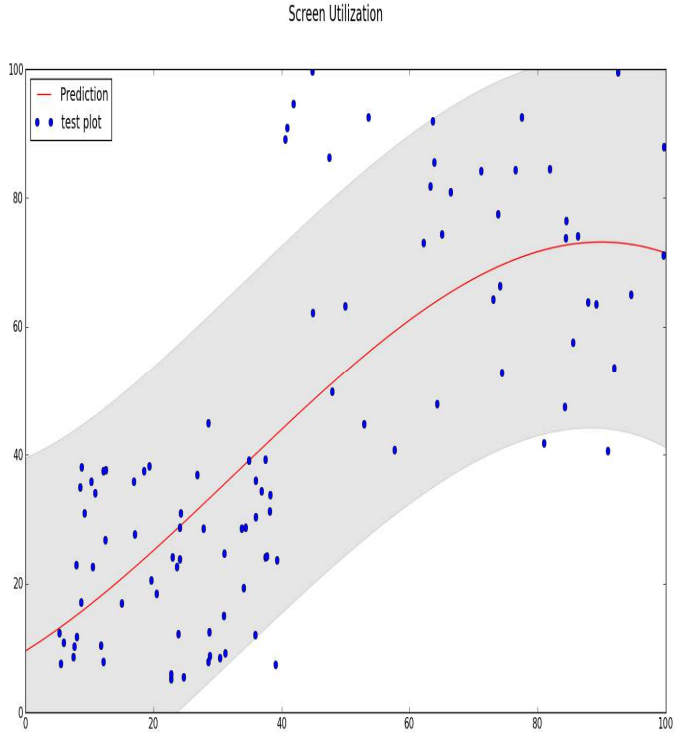

Transmitted bytes
Prediction
test plot

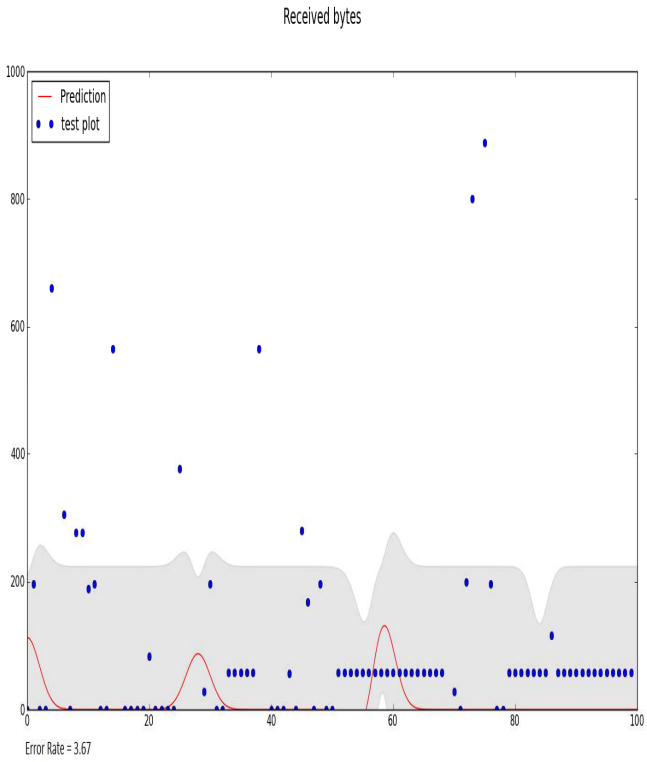
User 2:

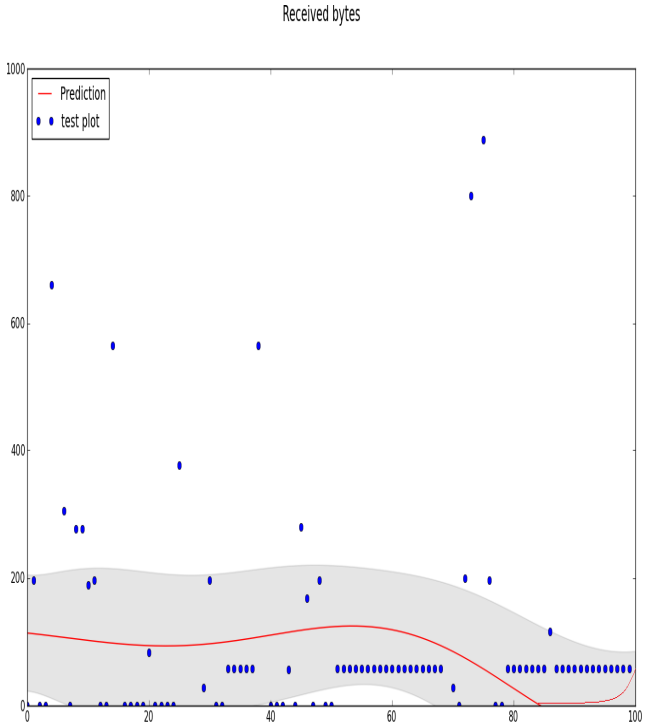

CPU Utilization
Prediction
test plot
Error Rate = 0.95

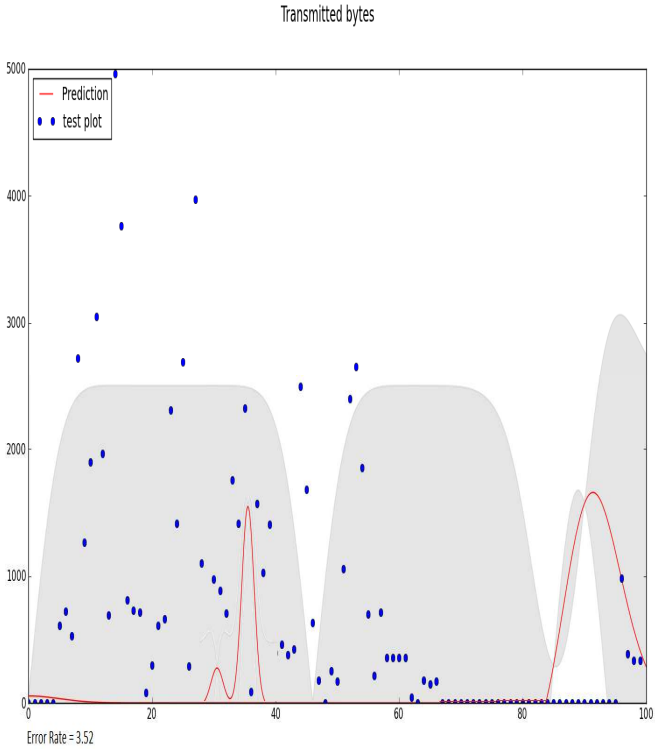

CPU Utilization
Prediction
test plot

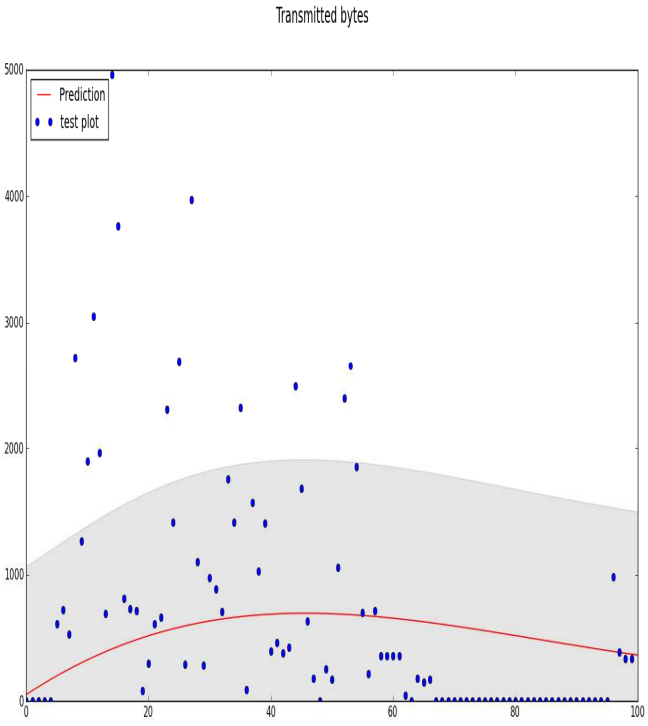

Screen Utilization
Prediction
test plot
Error Rate = 0.58


Screen Utilization
Prediction
test plot

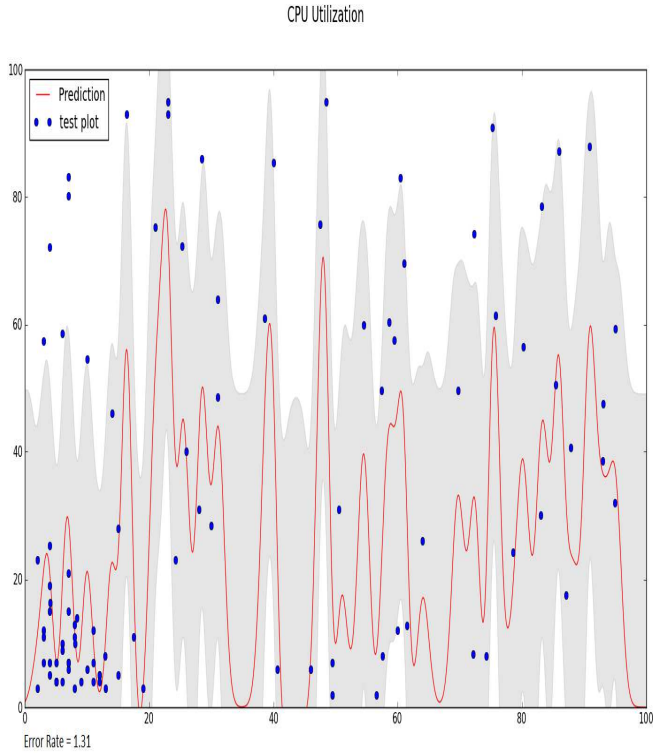

Received bytes
Prediction
test plot
Error Rate = 3.67

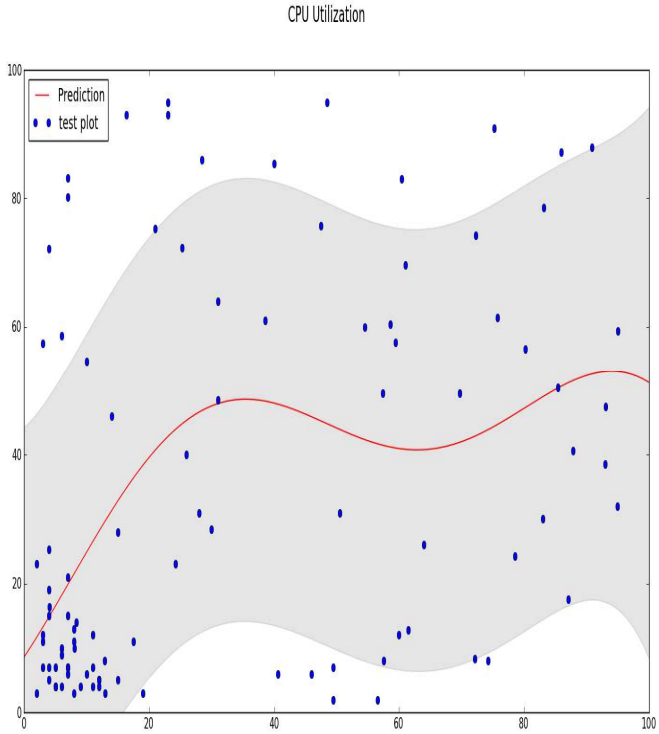

Received bytes
Prediction
test plot

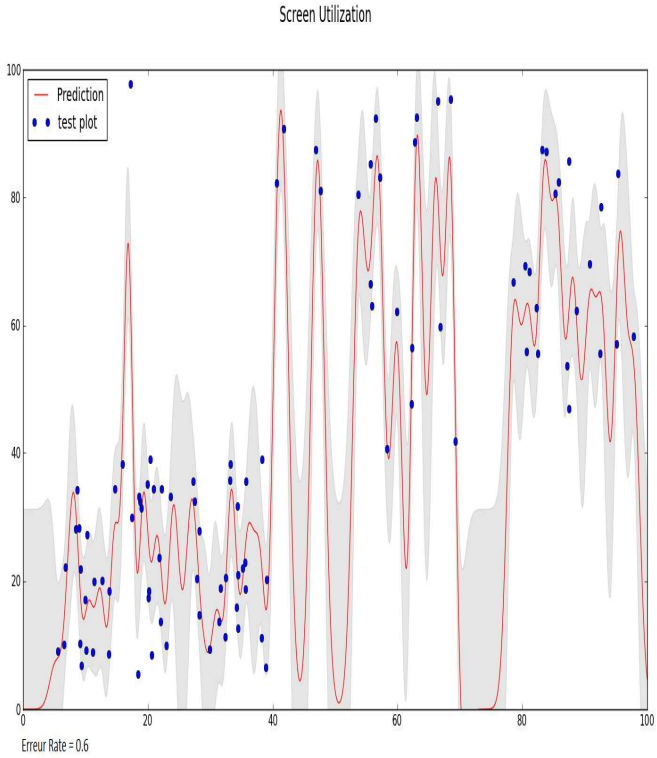

Transmitted bytes
Prediction
test plot
Error Rate = 3.52

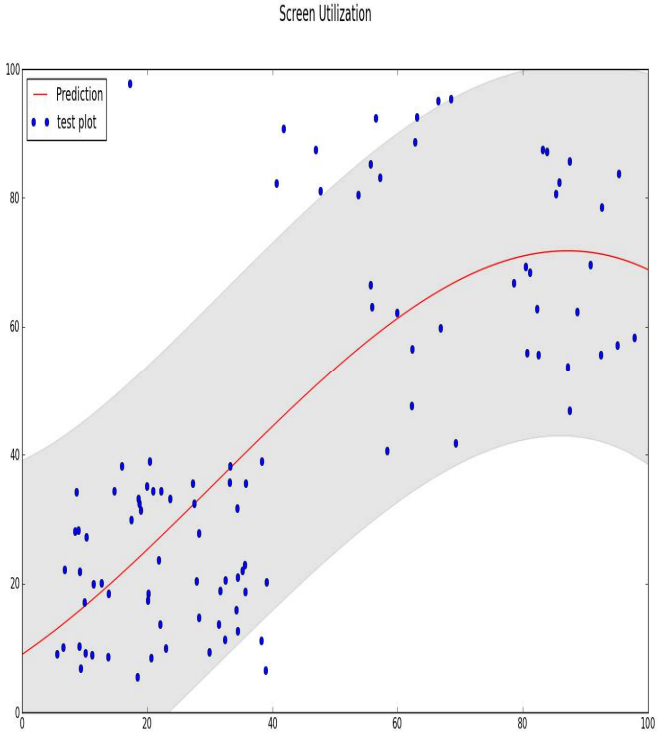

Transmitted bytes
Prediction
test plot

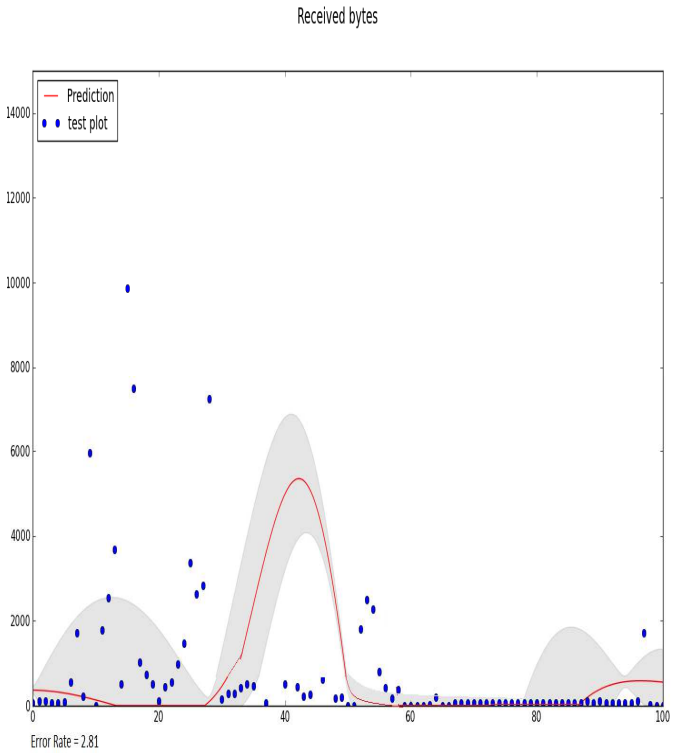
User 3:

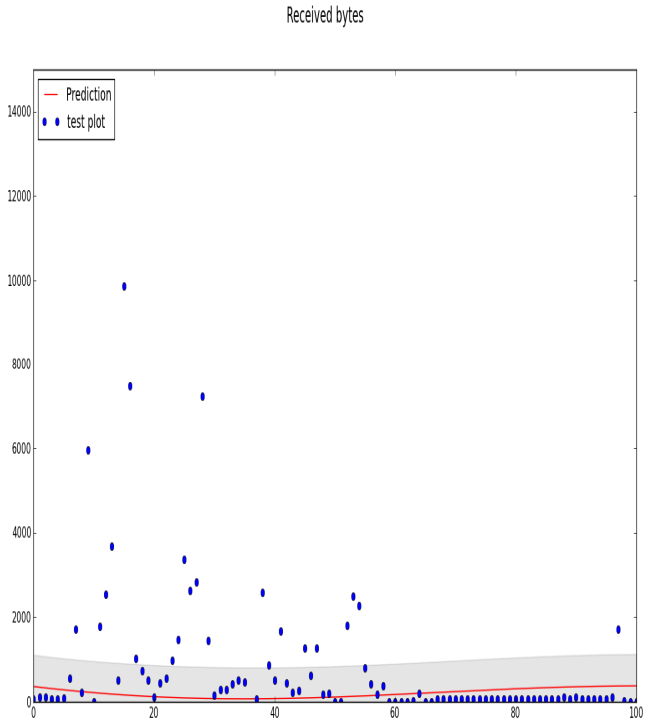

CPU Utilization
Prediction
test plot
Error Rate = 1.31

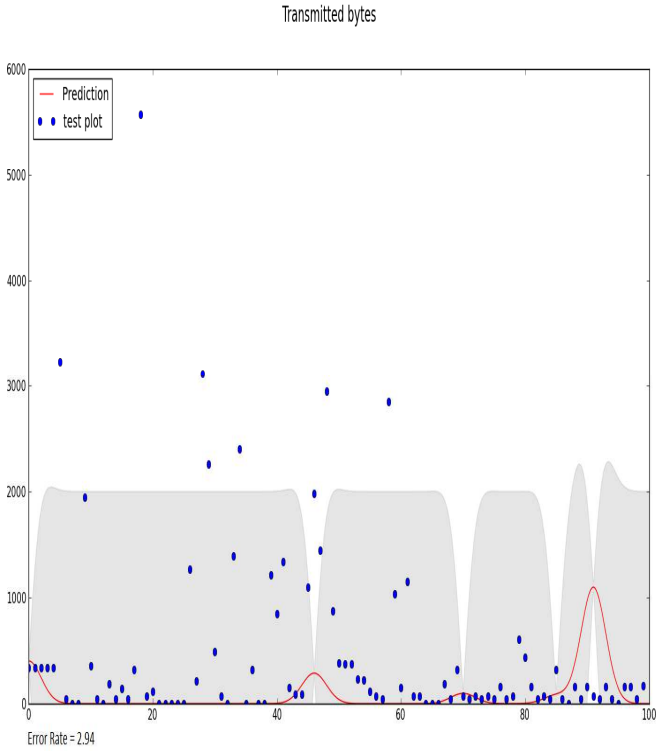

CPU Utilization
Prediction
test plot

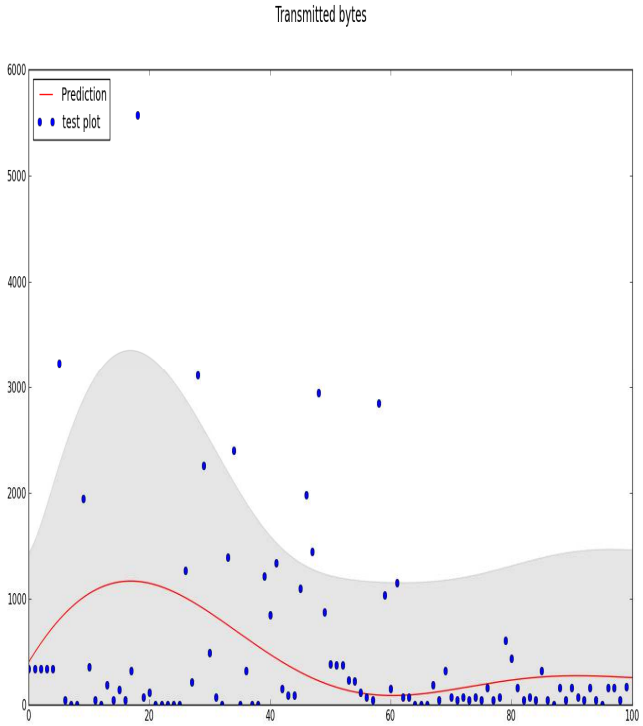

Screen Utilization
Prediction
test plot
Erreur Rate = 0.6


Screen Utilization
Prediction
test plot

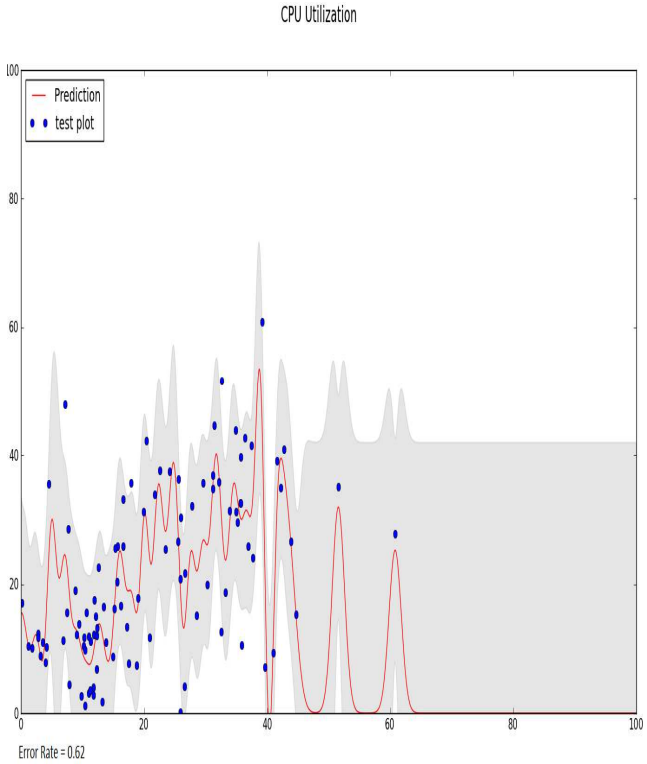

Received bytes
Prediction
test plot
Error Rate = 2.81

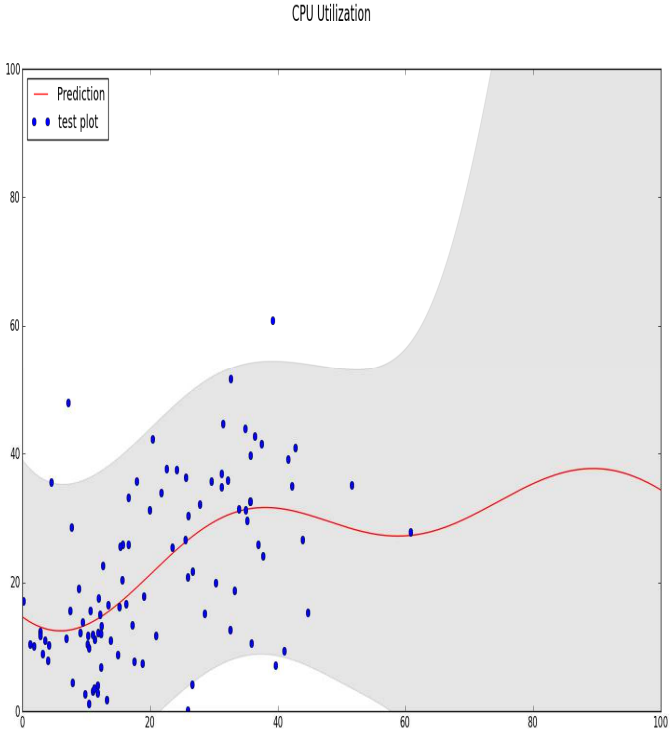

Received bytes
Prediction
test plot

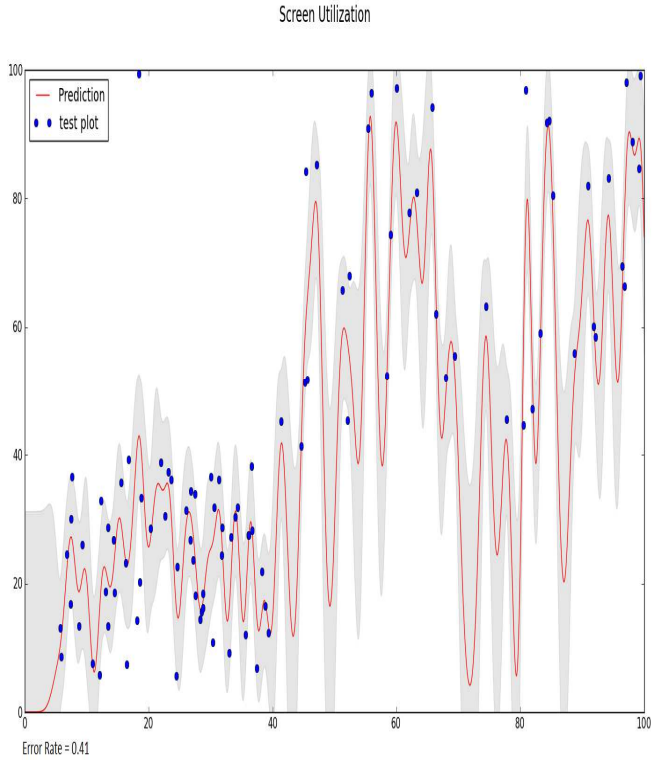

Transmitted bytes
Prediction
test plot
Error Rate = 2.94

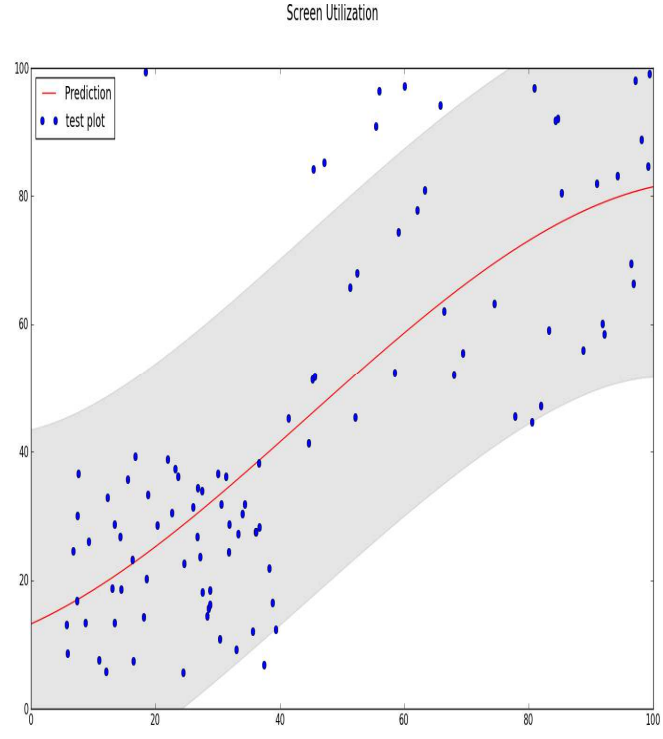

Transmitted bytes
Prediction
test plot

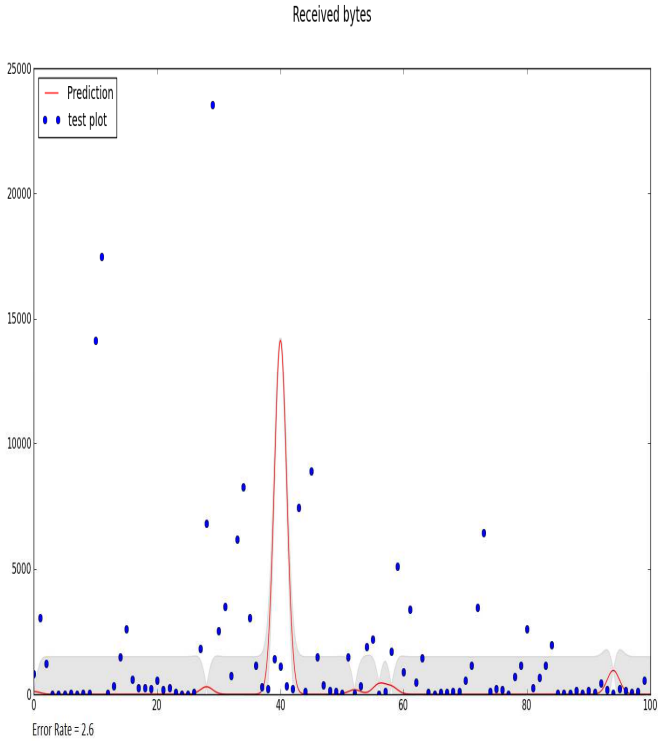
User 4:

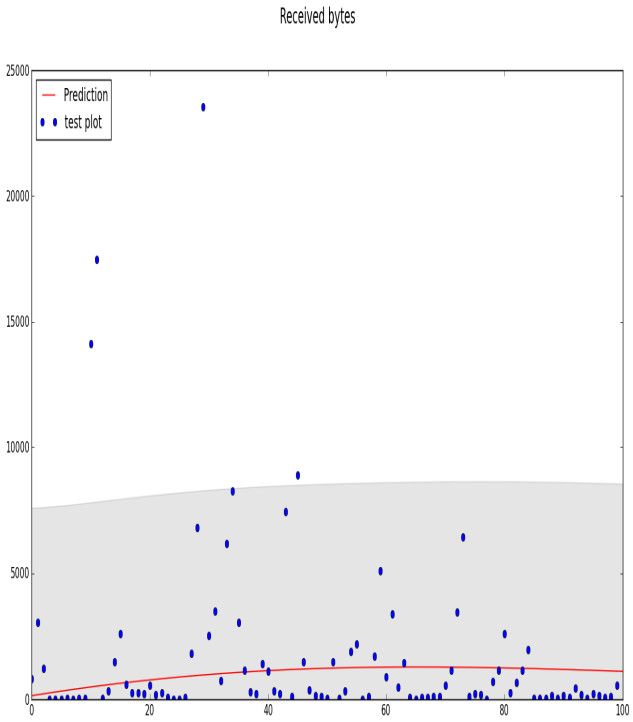

CPU Utilization
Prediction
test plot
Error Rate = 0.62

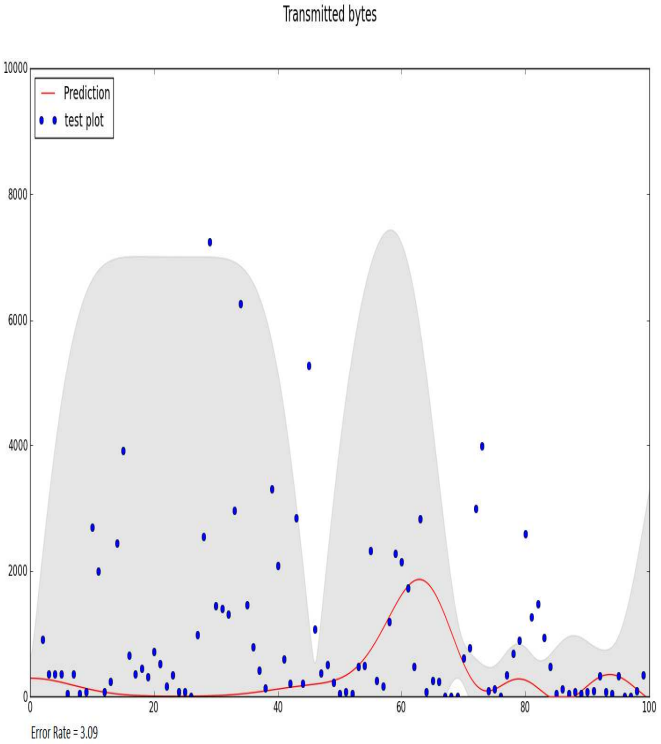

CPU Utilization
Prediction
test plot

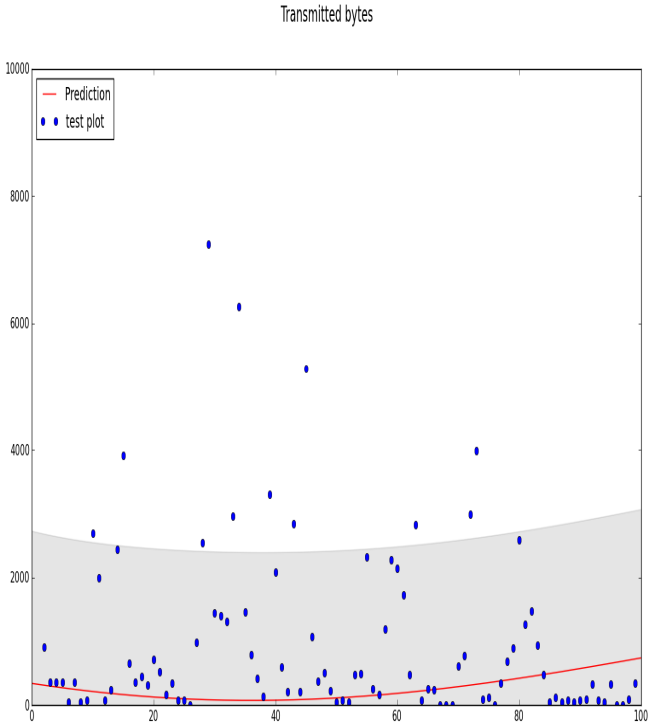

Screen Utilization
Prediction
test plot
Error Rate = 0.41


Screen Utilization
Prediction
test plot

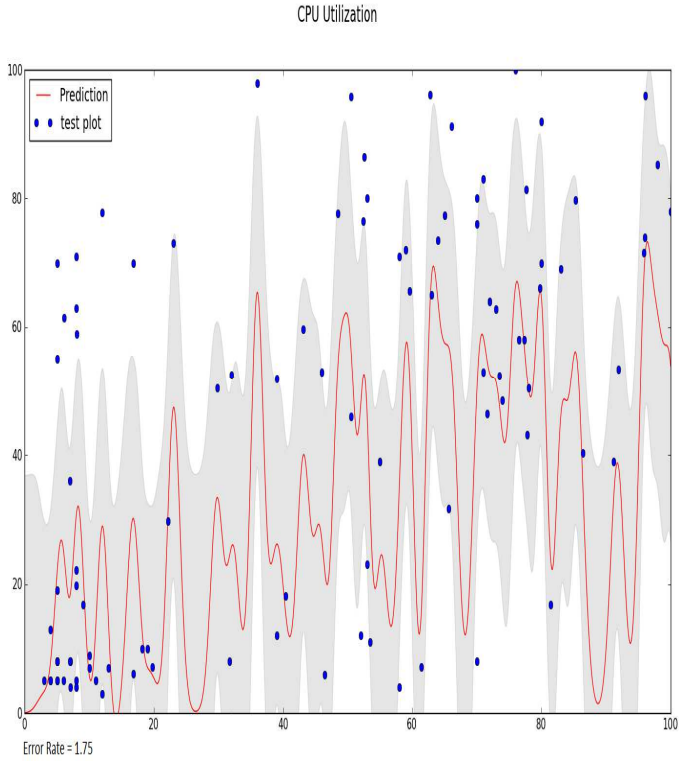

Received bytes
Prediction
test plot
Error Rate = 2.6

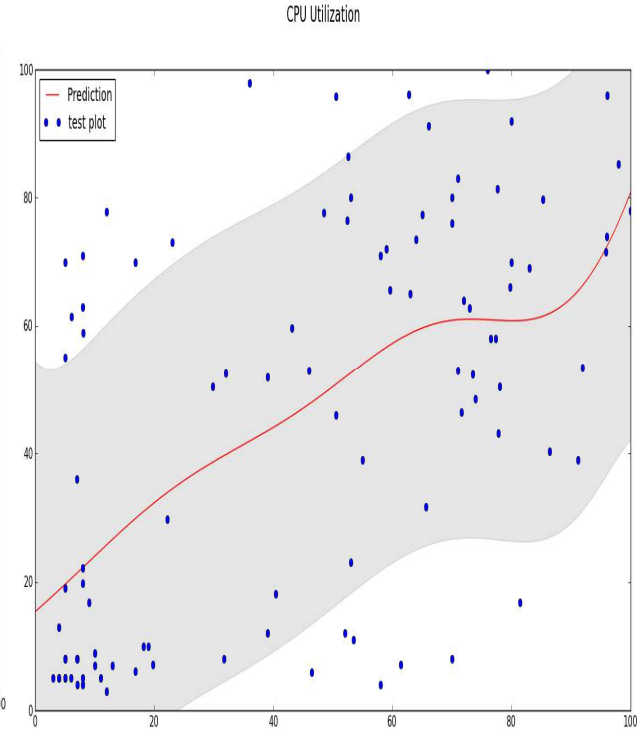

Received bytes
Prediction
test plot

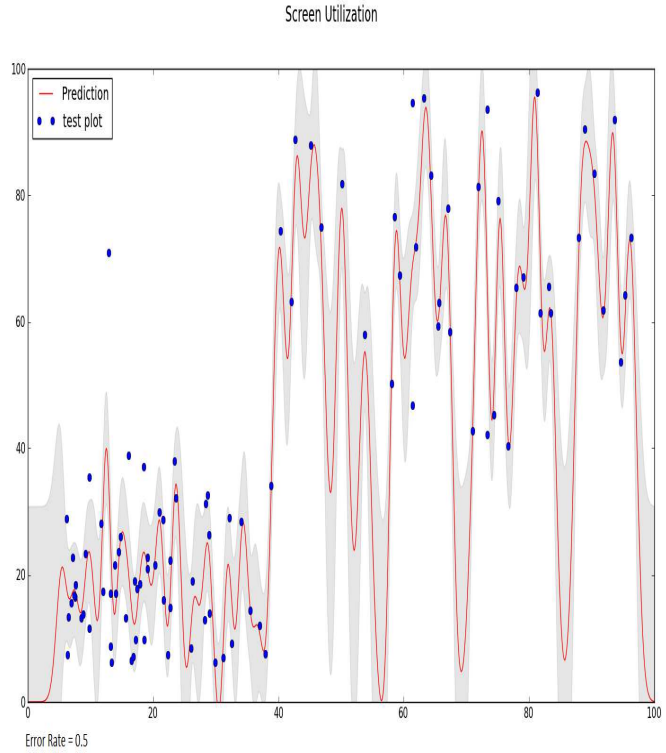

Transmitted bytes
Prediction
test plot
Error Rate = 3.09

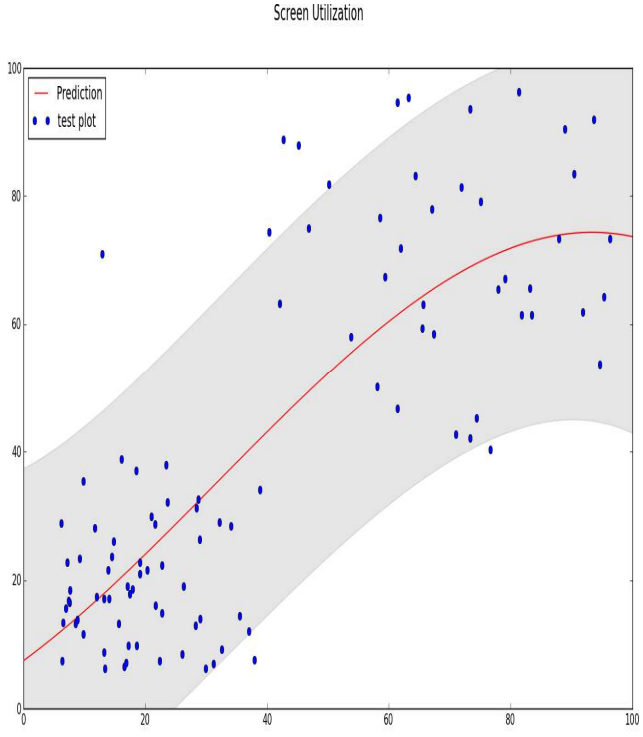

Transmitted bytes
Prediction
test plot

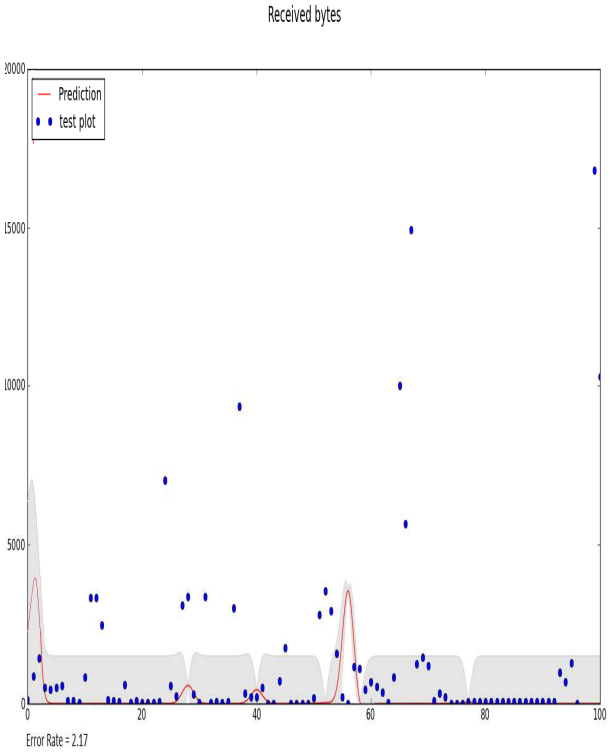
User 5:

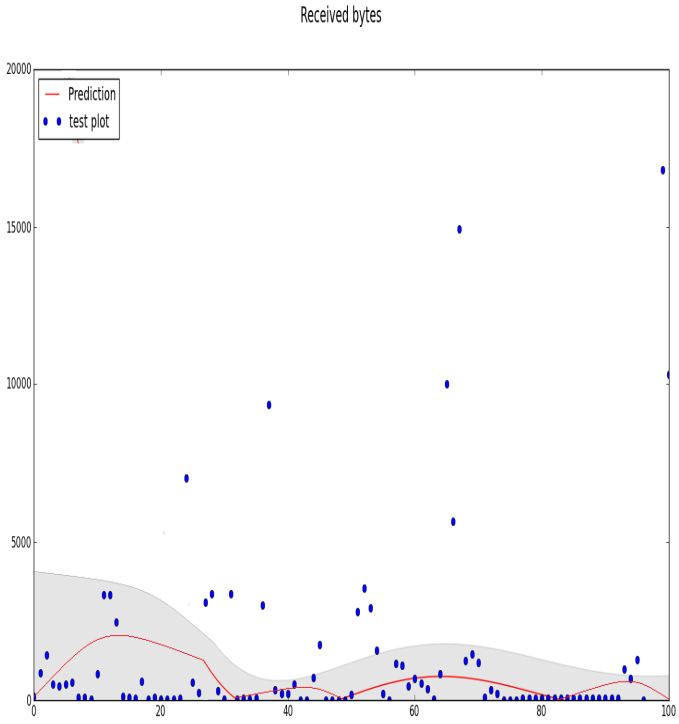

CPU Utilization
Prediction
test plot
Error Rate = 1.75

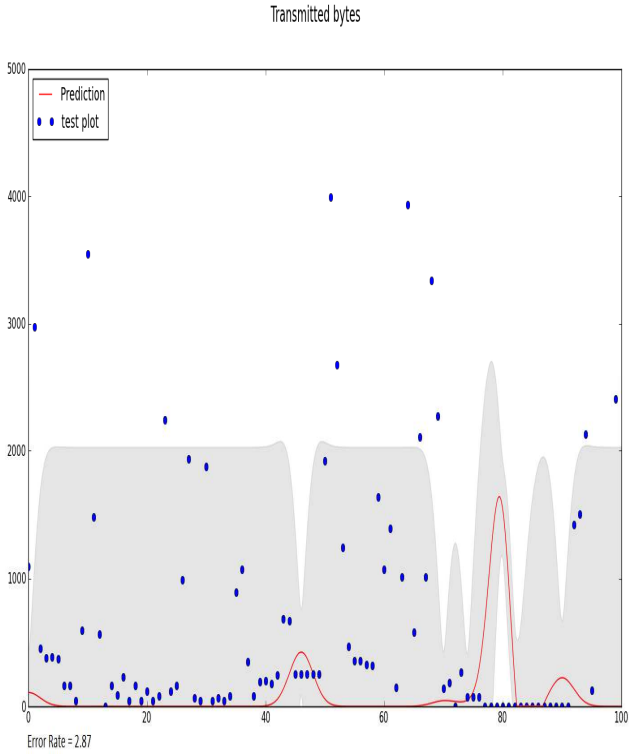

CPU Utilization
Prediction
test plot

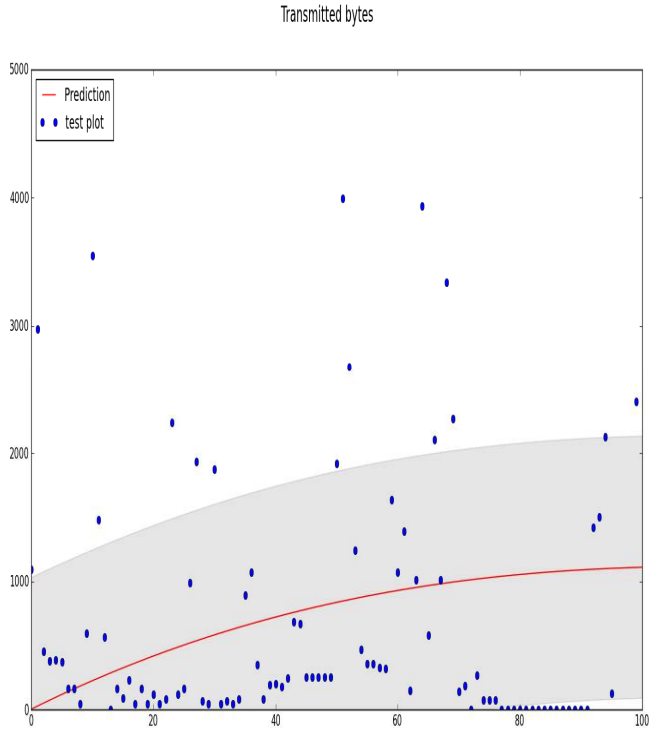

Screen Utilization
Prediction
test plot
Error Rate = 0.5


Screen Utilization
Prediction
test plot


Received bytes
Prediction
test plot
Error Rate = 2.17

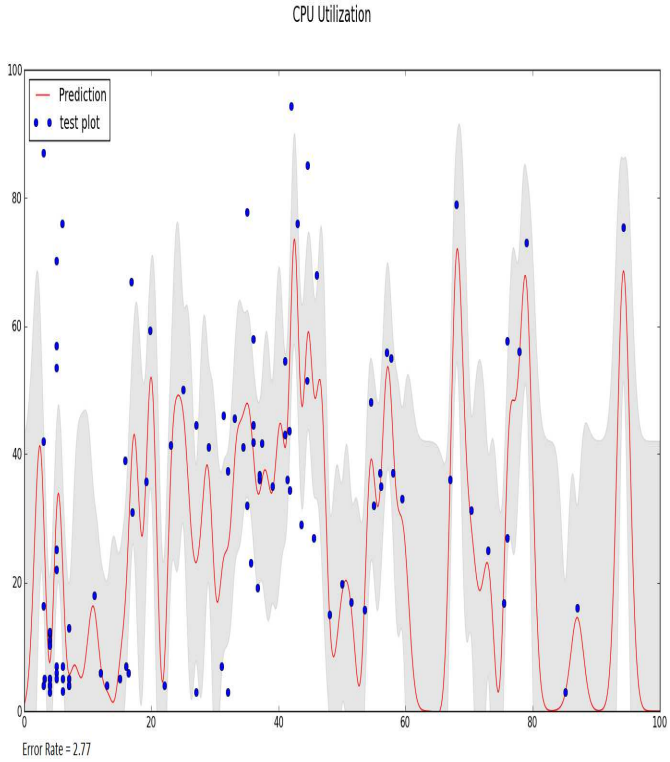

Received bytes
Prediction
test plot

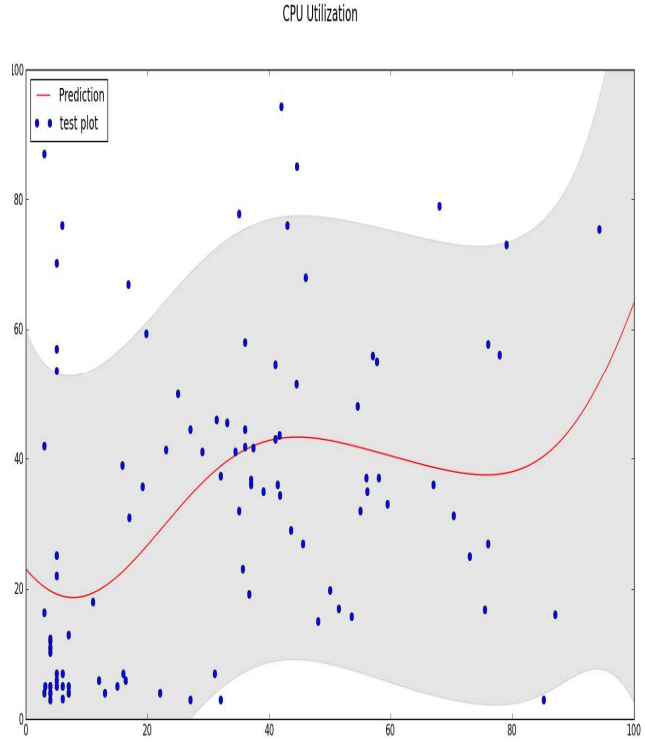

Transmitted bytes
Prediction
test plot
Error Rate = 2.87

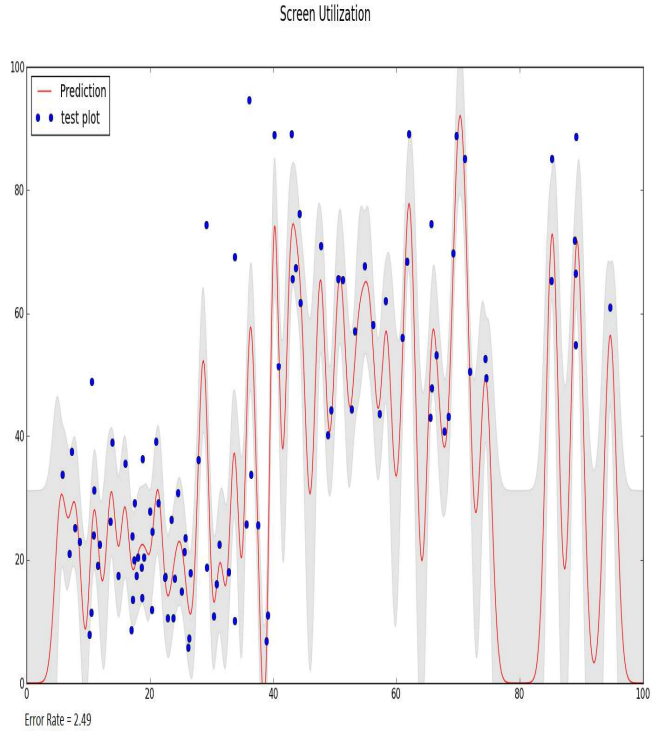

Transmitted bytes
Prediction
test plot

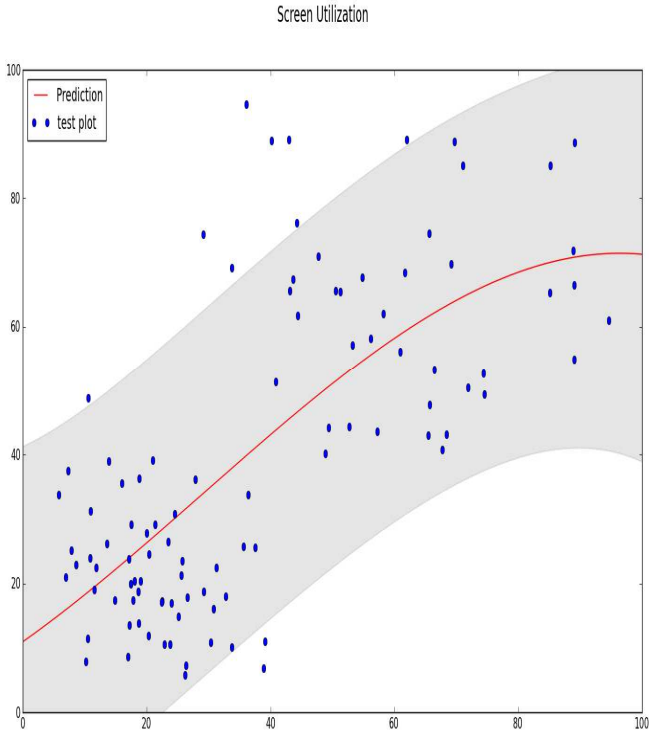
Results based on 5 second historical measures

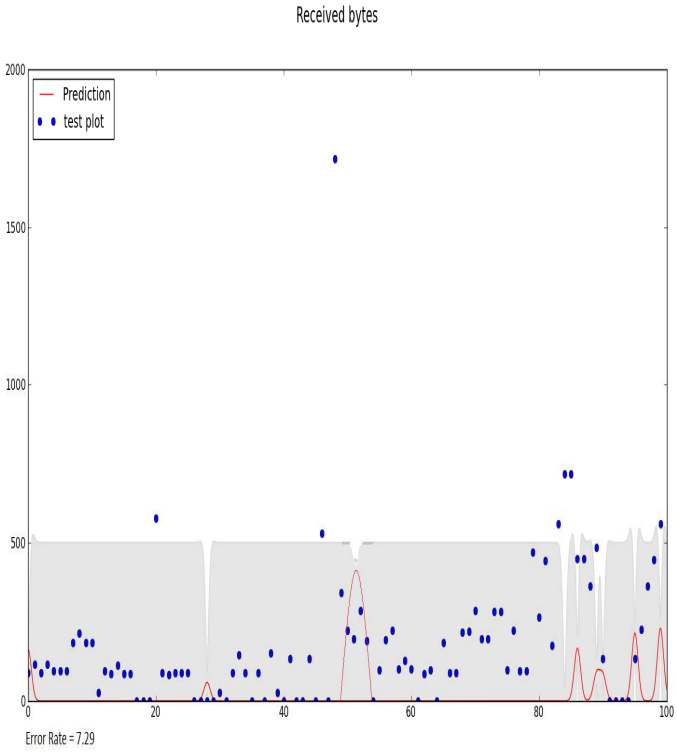
User 1

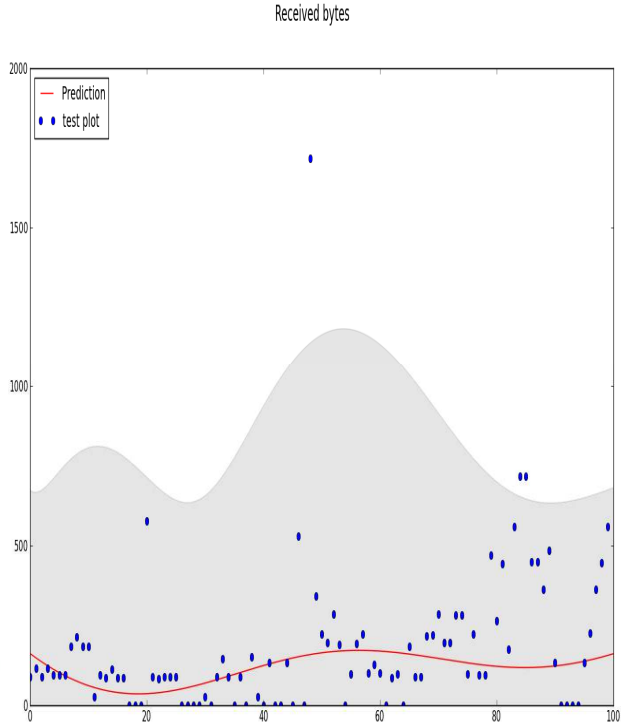


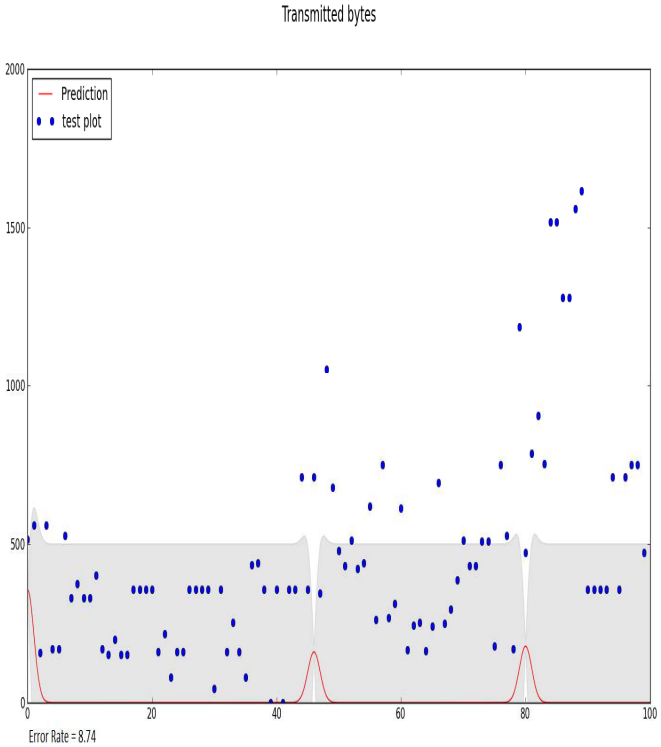


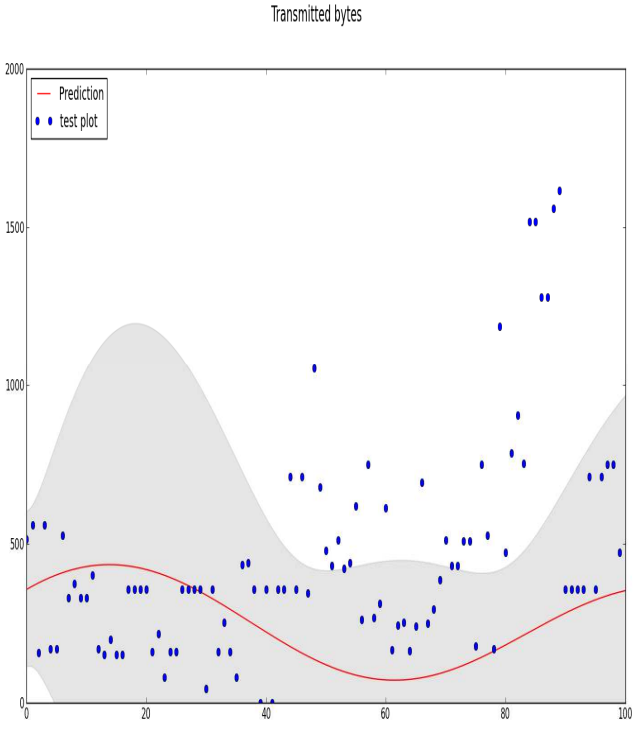




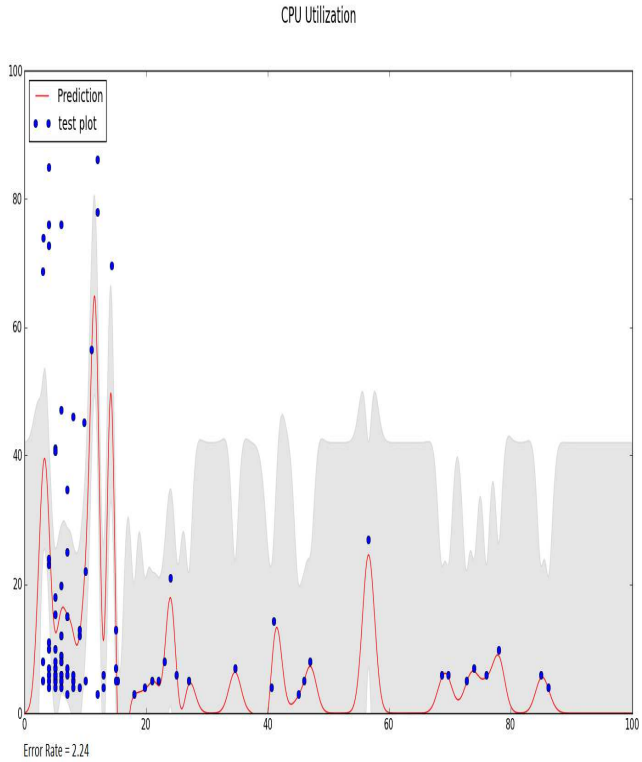

Received bytes
Prediction
test plot
Error Rate = 7.29

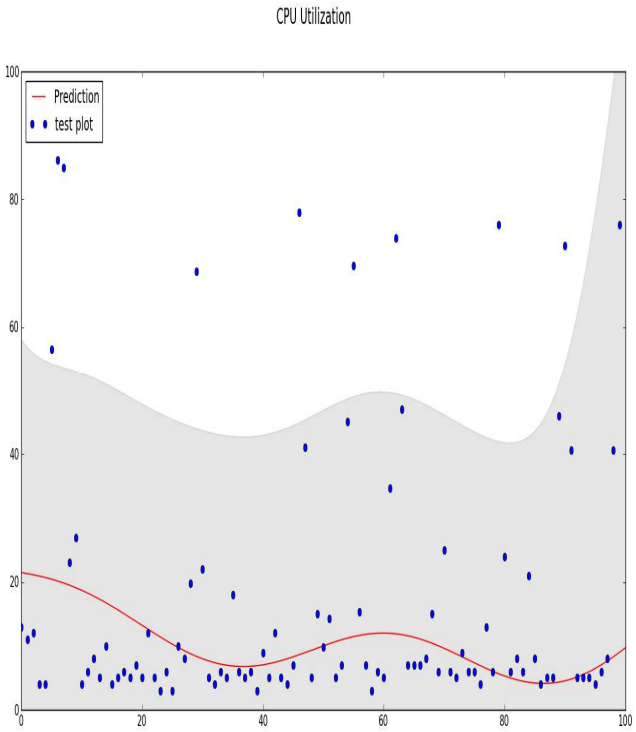

Received bytes
Prediction
test plot

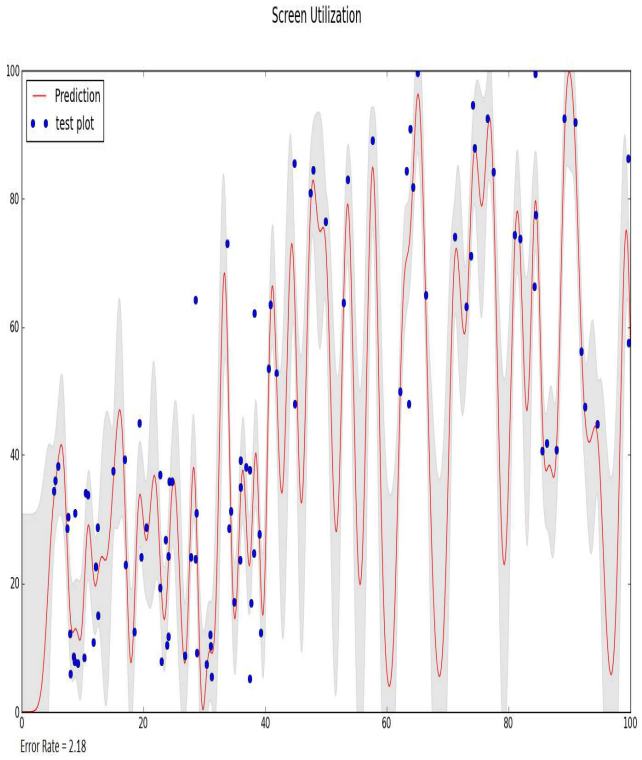

Transmitted bytes
Prediction
test plot
Error Rate = 8.74

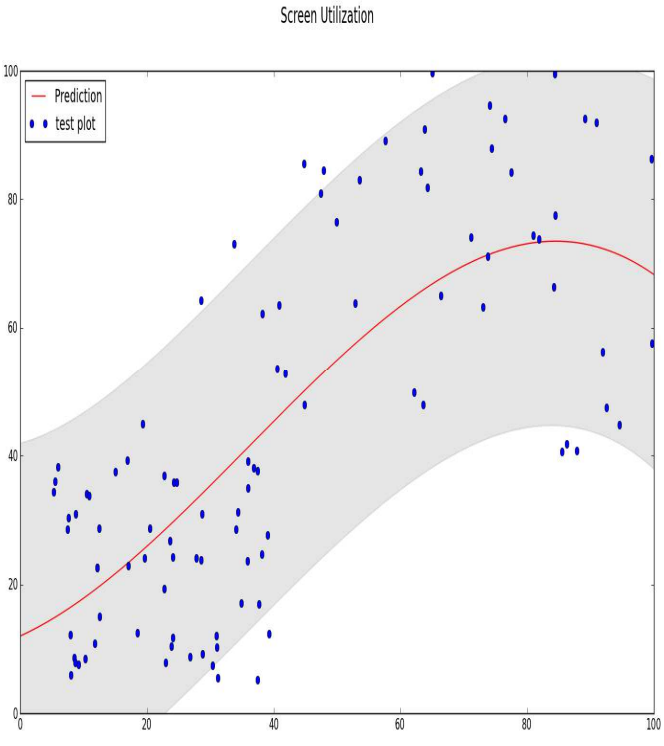

Transmitted bytes
Prediction
test plot

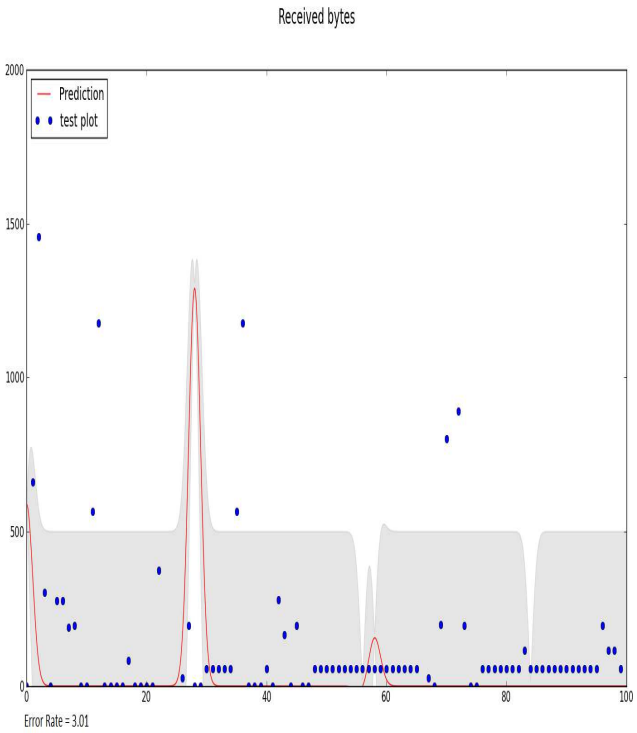
User 2

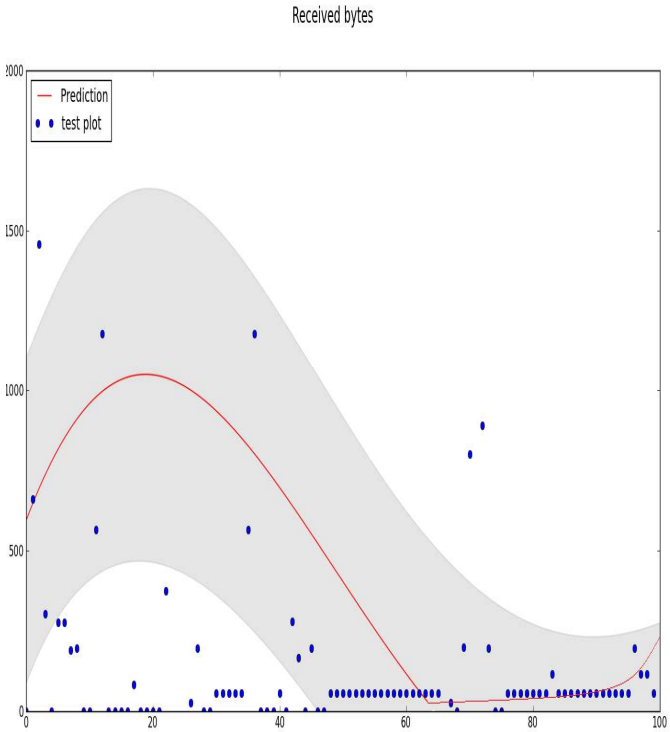

CPU Utilization
Prediction
test plot
Error Rate = 2.24

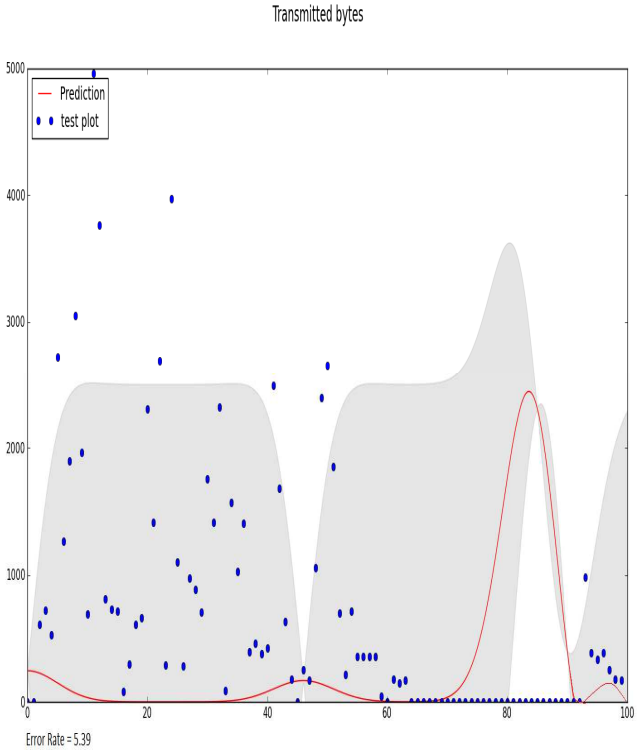

CPU Utilization
Prediction
test plot

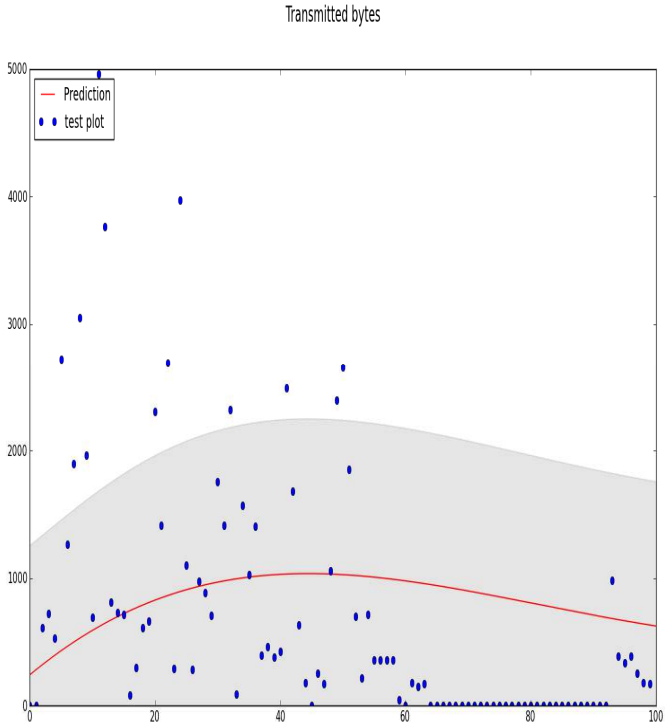

Screen Utilization
Prediction
test plot
Error Rate = 2.18


Screen Utilization
Prediction
test plot

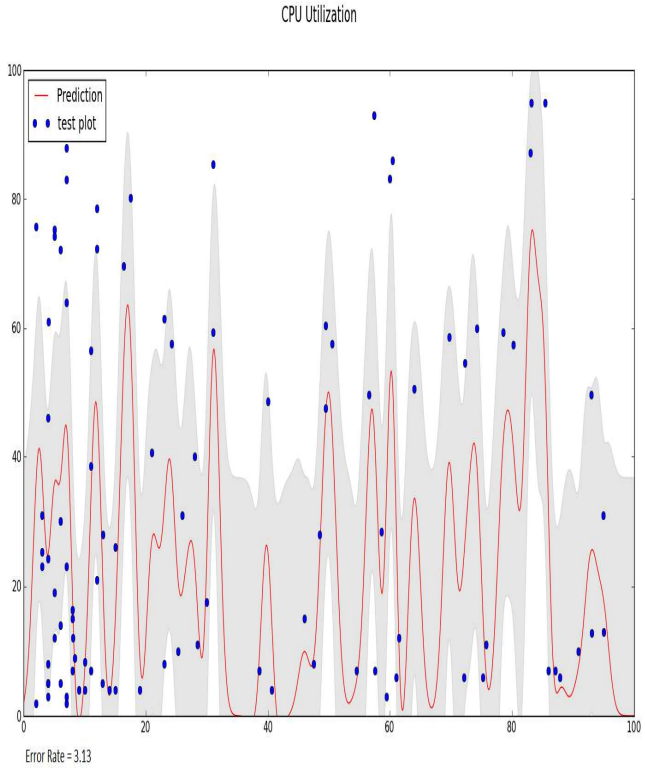

Received bytes
Prediction
test plot
Error Rate = 3.01

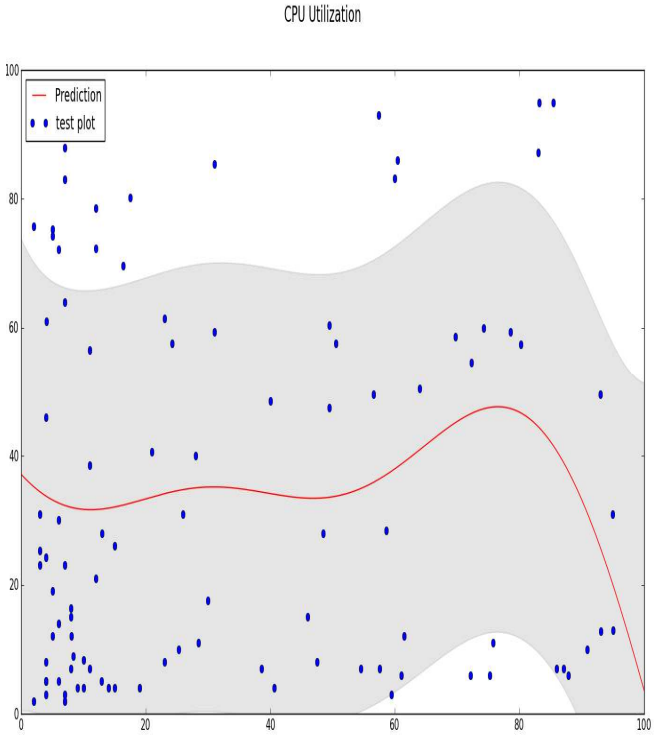

Received bytes
Prediction
test plot

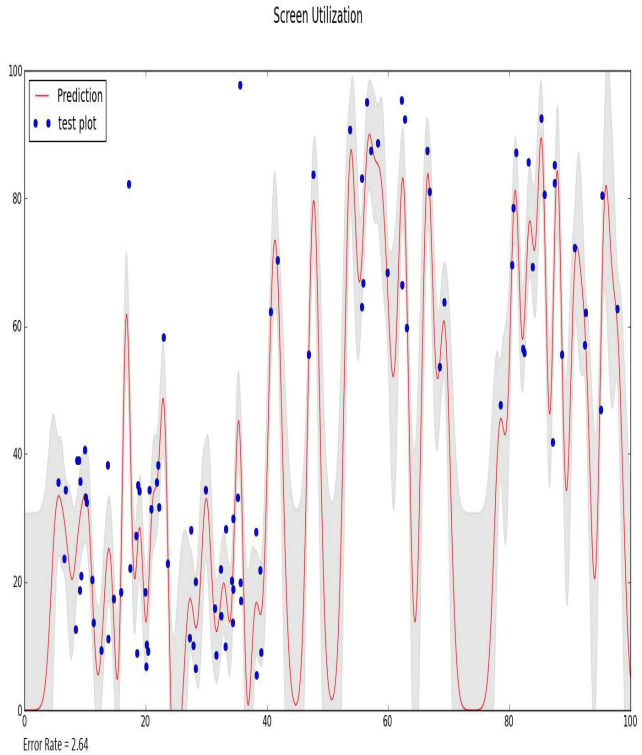

Transmitted bytes
Prediction
test plot
Error Rate = 5.39

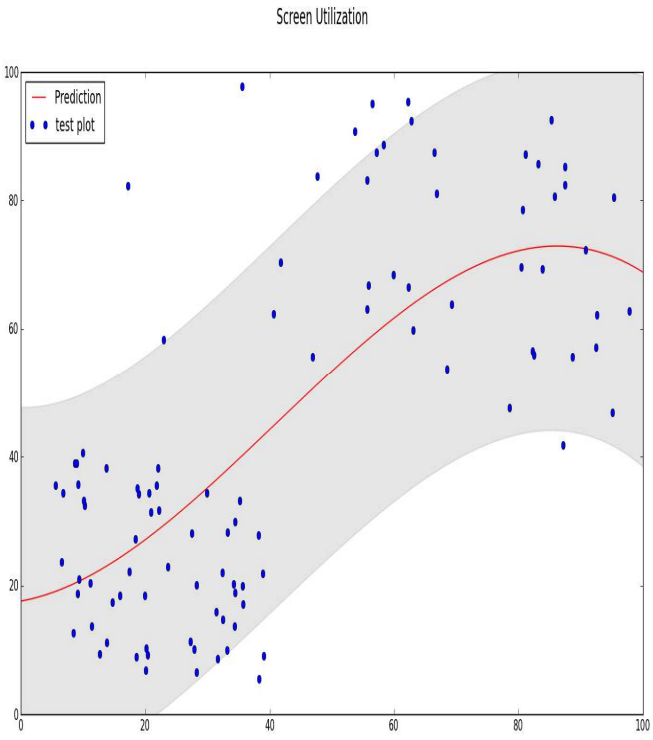

Transmitted bytes
Prediction
test plot

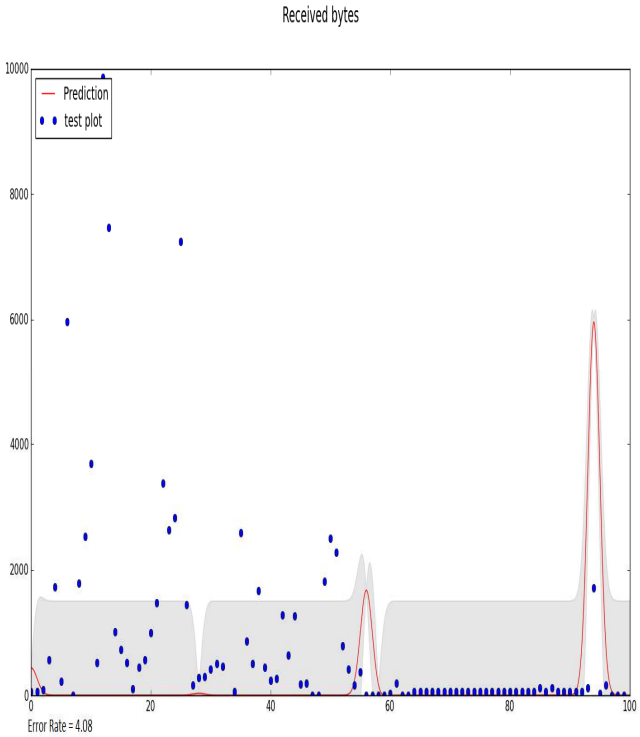
User 3

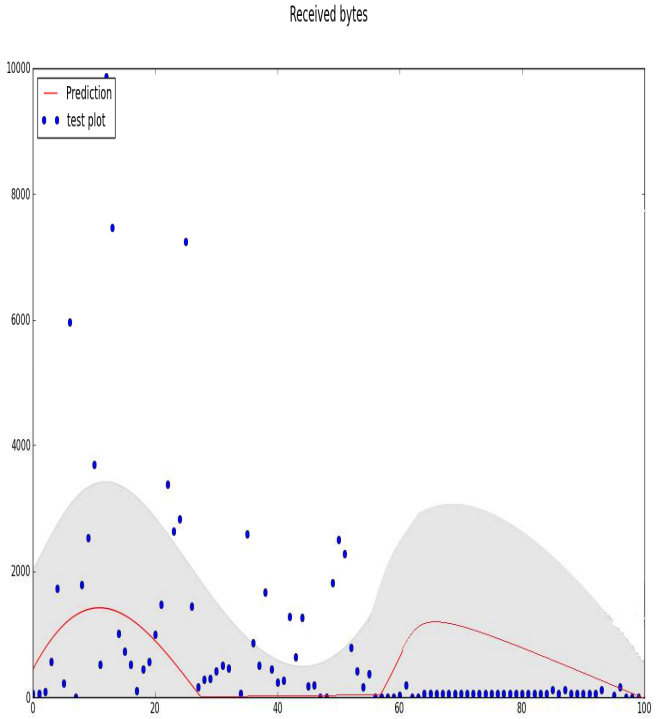

CPU Utilization
Prediction
test plot
Error Rate = 3.13

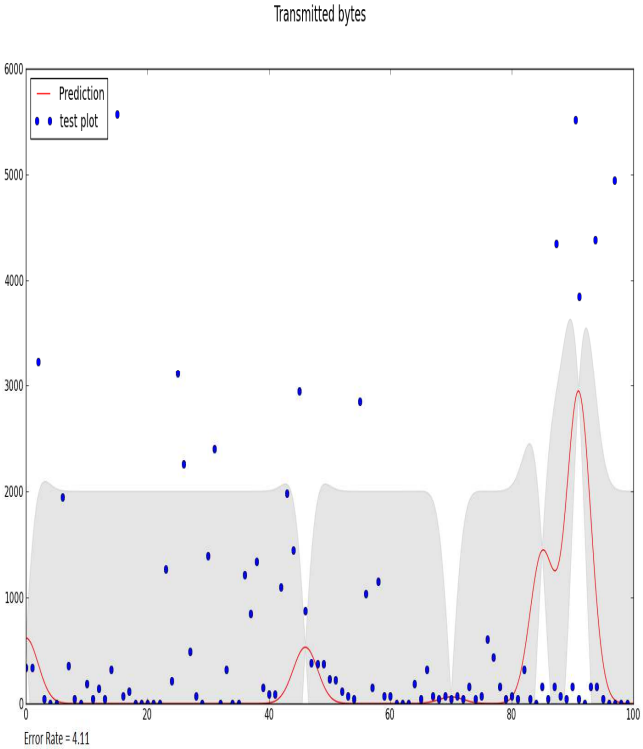

CPU Utilization
Prediction
test plot

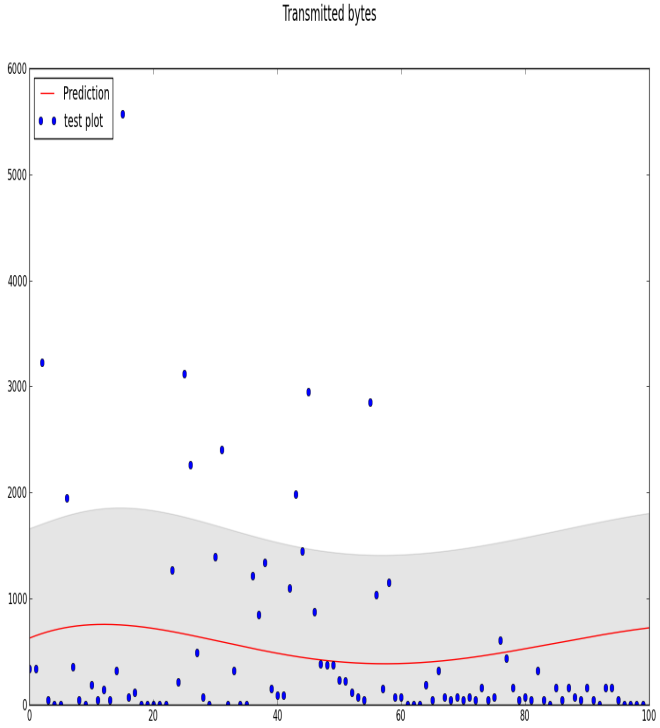

Screen Utilization
Prediction
test plot
Error Rate = 2.64


Screen Utilization
Prediction
test plot

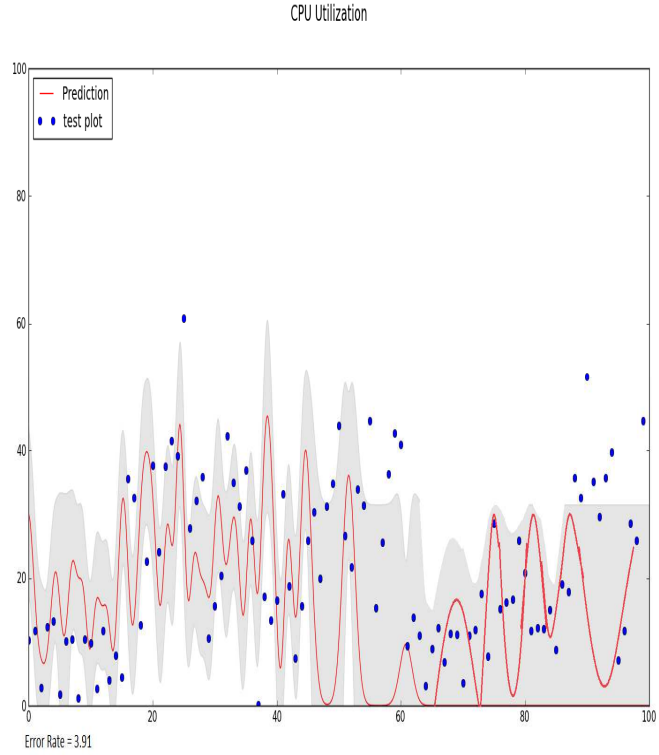

Received bytes
Prediction
test plot
Error Rate = 4.08

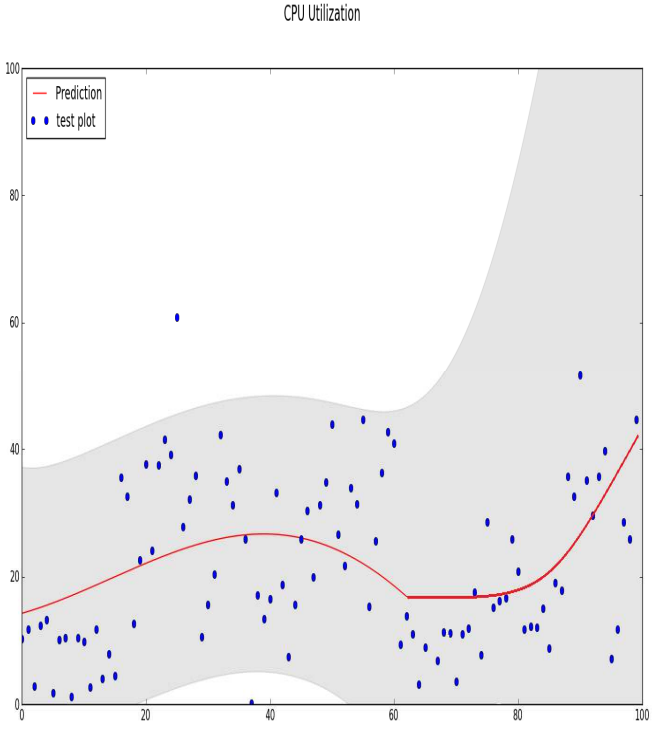

Received bytes
Prediction
test plot

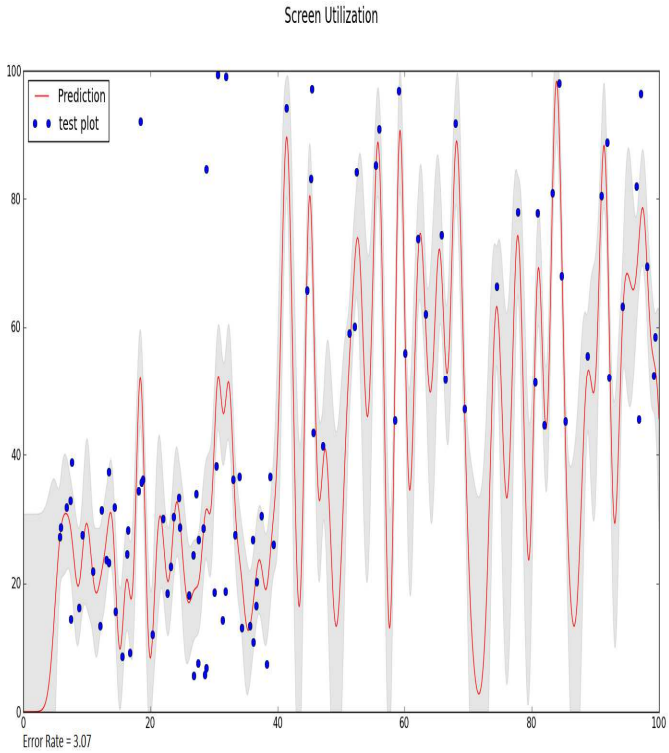

Transmitted bytes
Prediction
test plot
Error Rate = 4.11

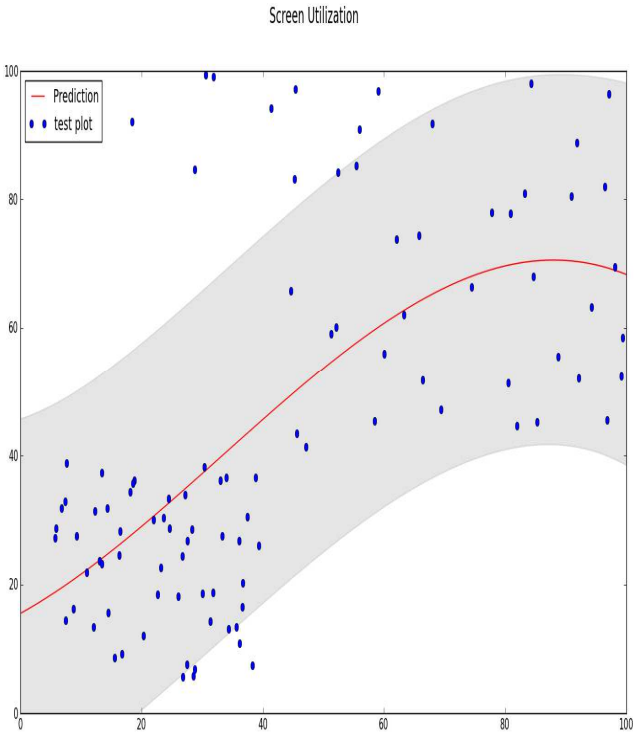

Transmitted bytes
Prediction
test plot

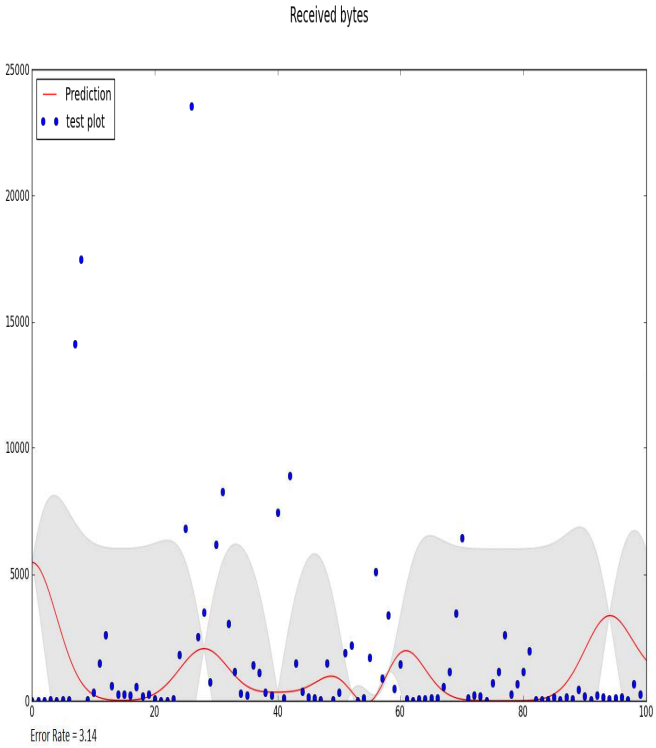
User 4

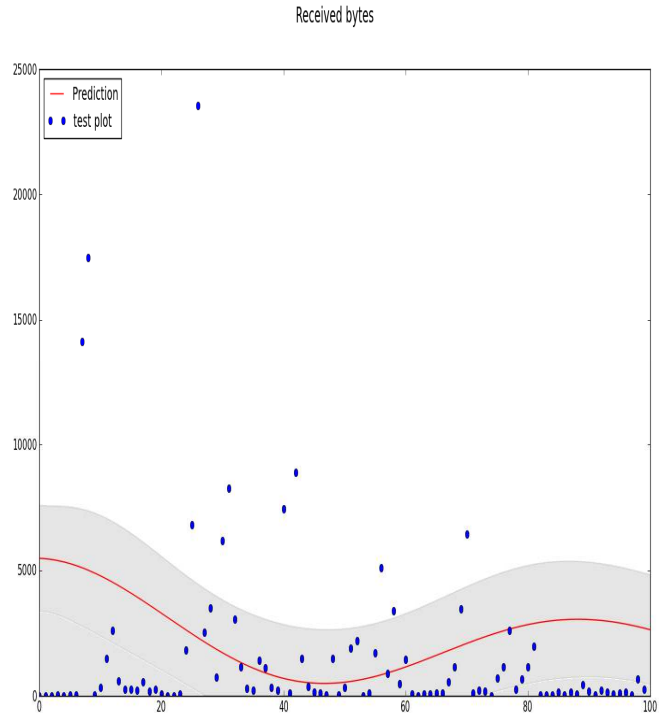

CPU Utilization
Prediction
test plot
Error Rate = 3.91

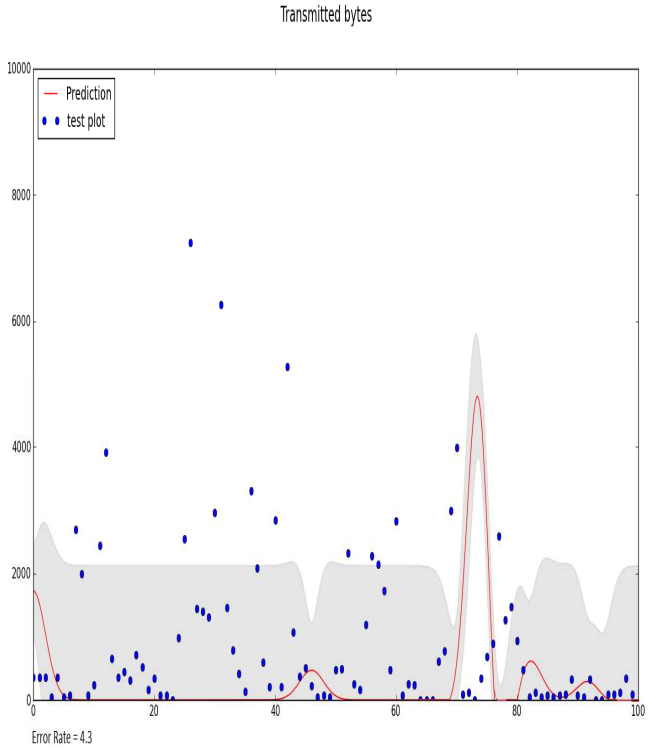

CPU Utilization
Prediction
test plot

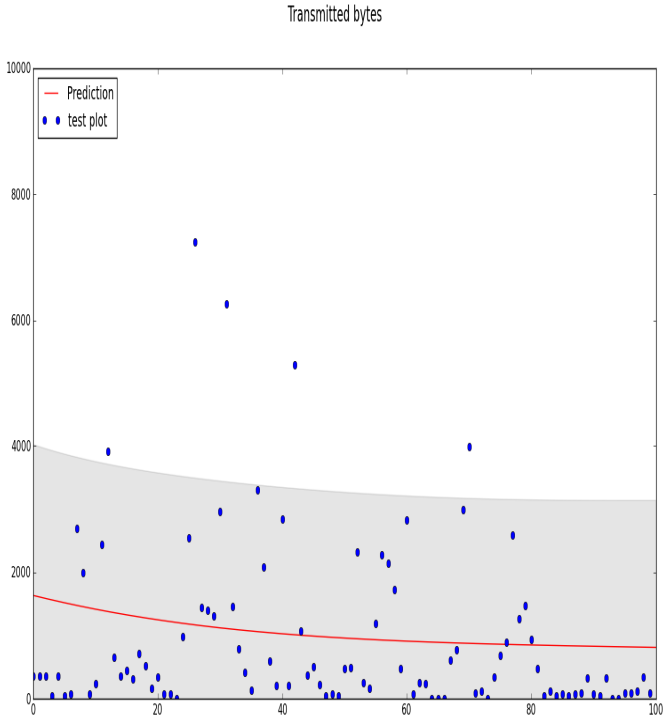

Screen Utilization
Prediction
test plot
Error Rate = 3.07


Screen Utilization
Prediction
test plot

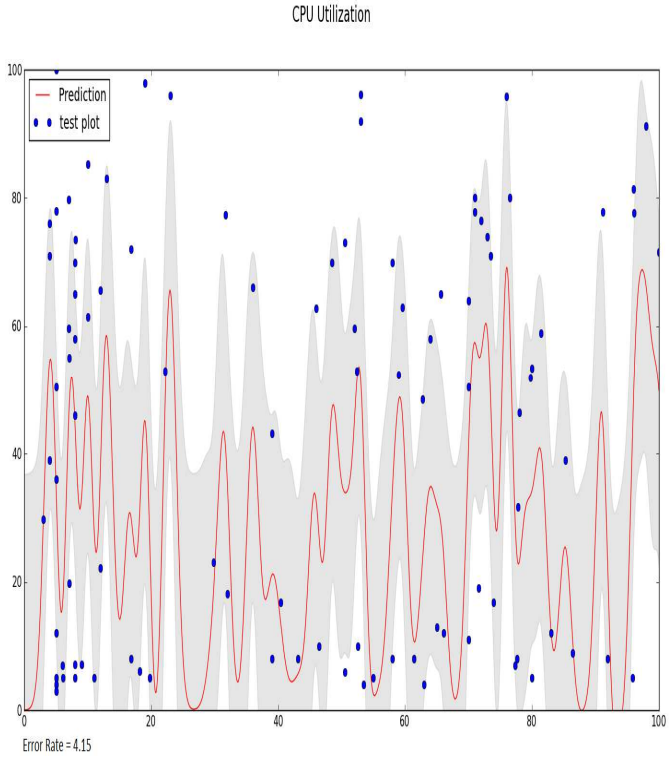

Received bytes
Prediction
test plot
Error Rate = 3.14

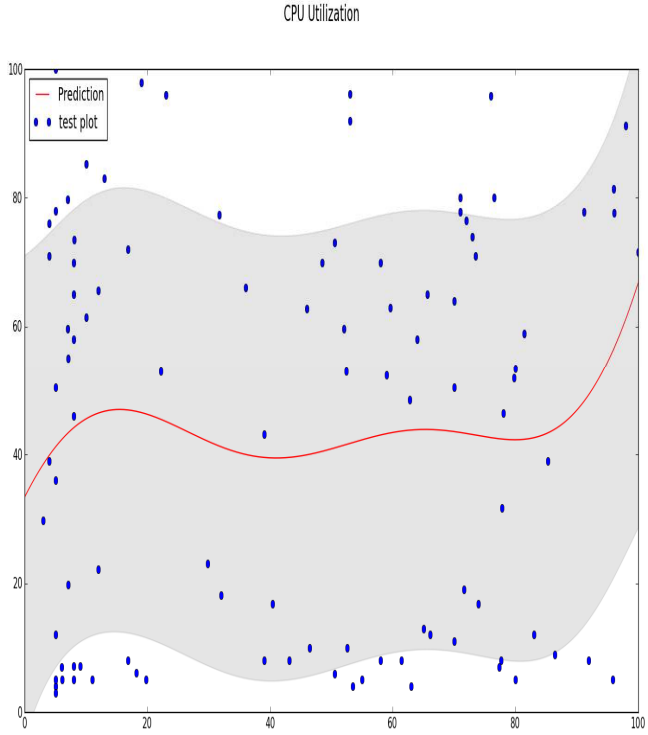

Received bytes
Prediction
test plot

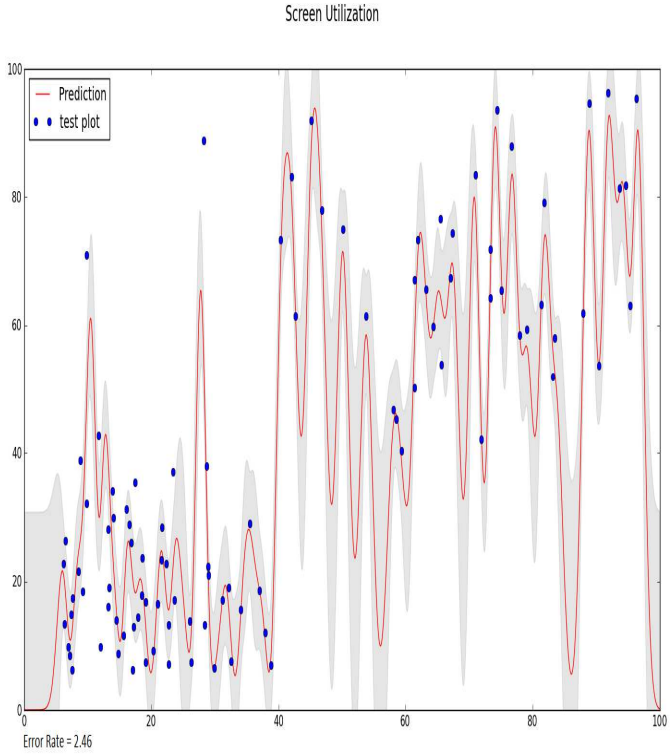

Transmitted bytes
Prediction
test plot
Error Rate = 4.3

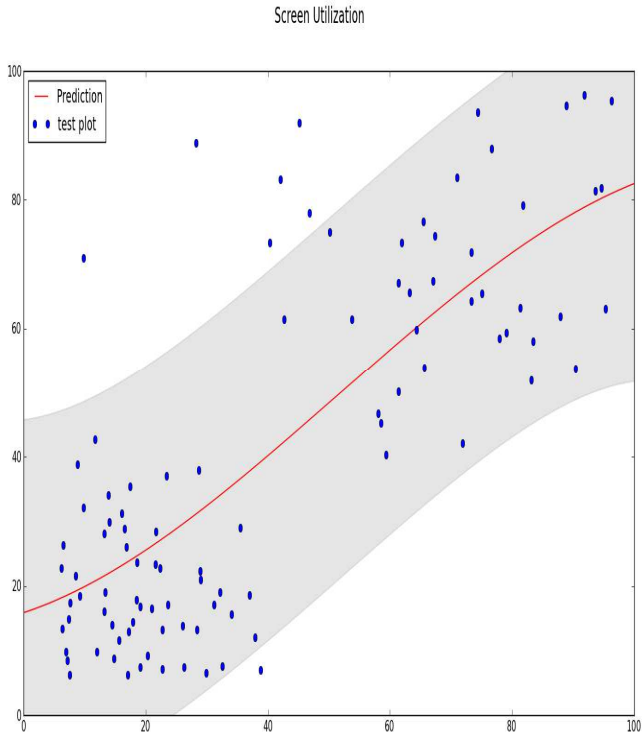

Transmitted bytes
Prediction
test plot

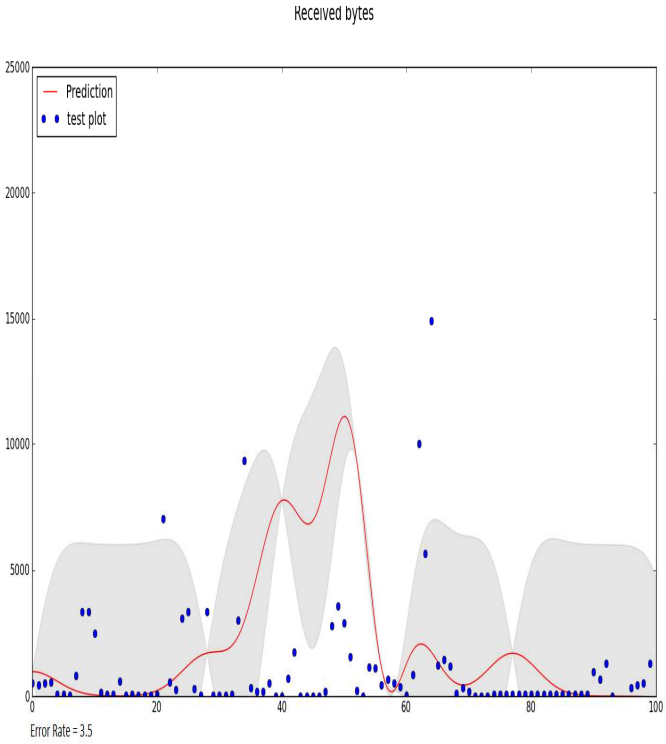
User 5

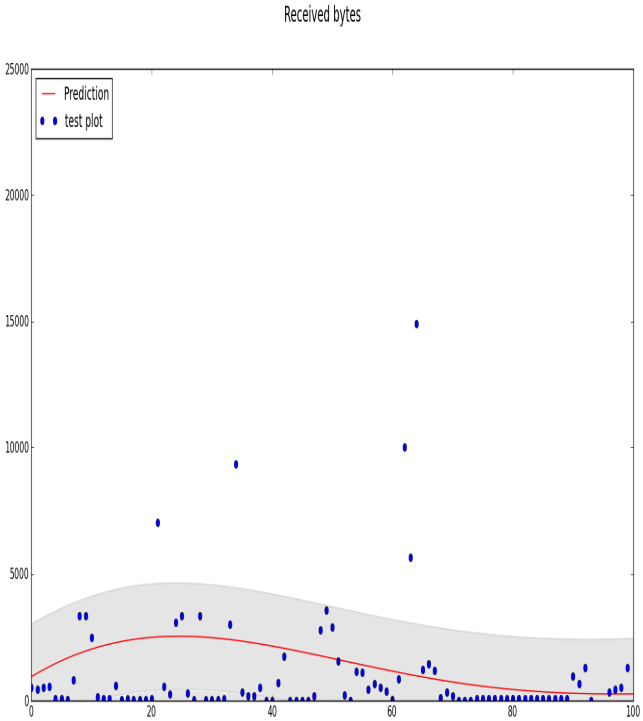

CPU Utilization
Prediction
test plot
Error Rate = 4.15

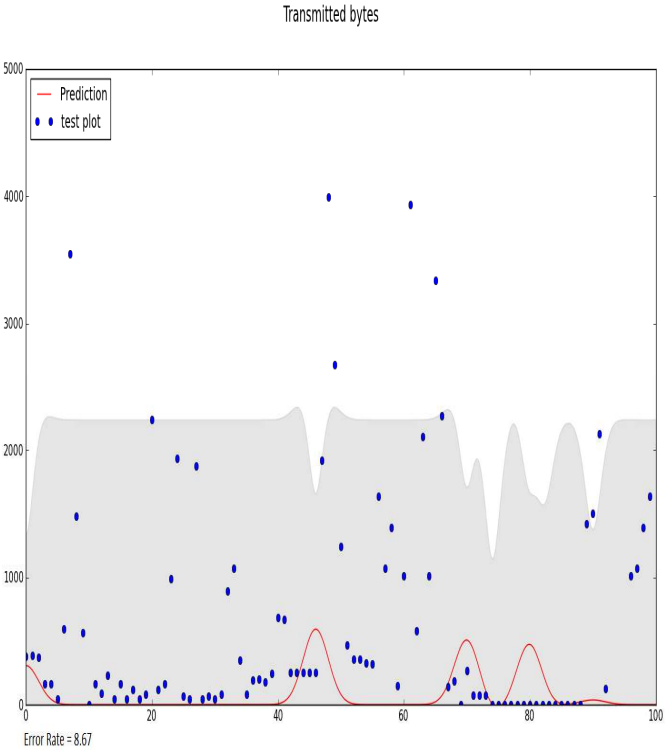

CPU Utilization
Prediction
test plot

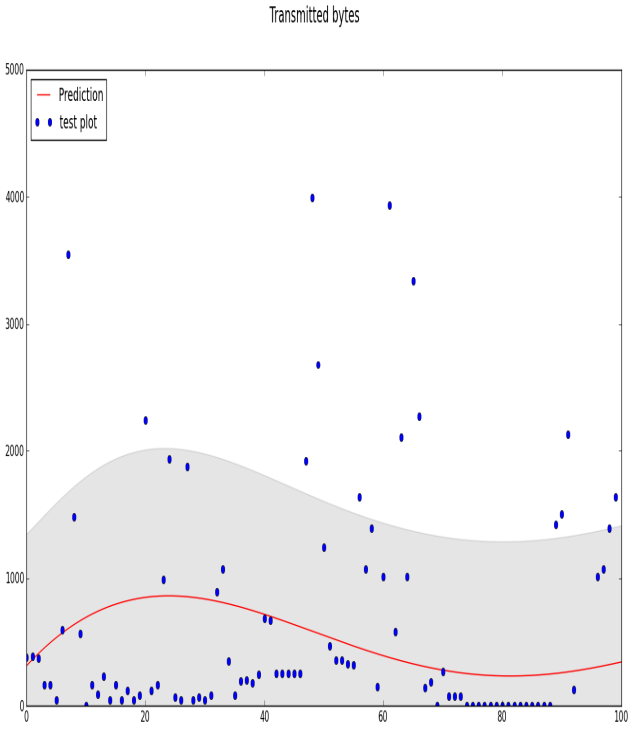

Screen Utilization
Prediction
test plot
Error Rate = 2.46


Screen Utilization
Prediction
test plot


Received bytes
Prediction
test plot
25000
20000
15000
10000
5000
0
0
20
40
60
80
100
Error Rate = 3.5

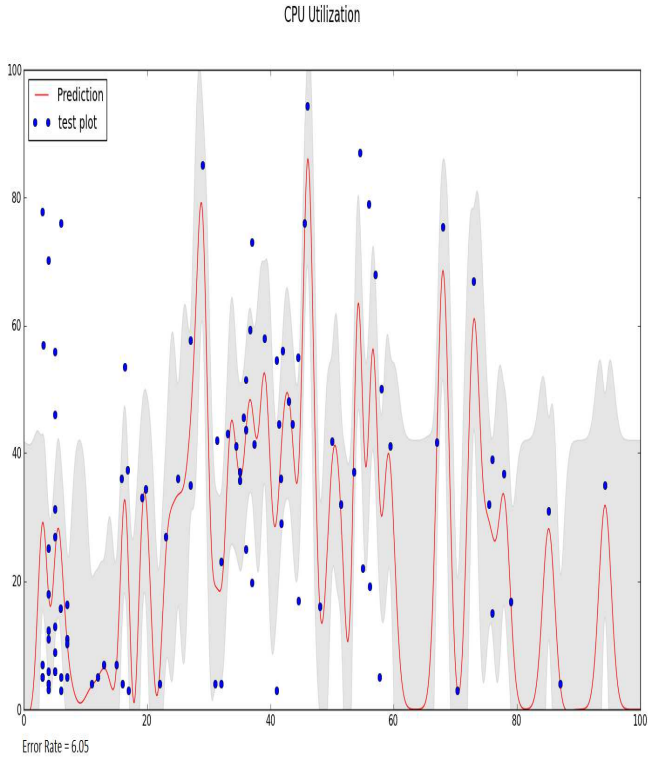

Received bytes
Prediction
test plot
25000
20000
15000
10000
5000
0
0
20
40
60
80
100

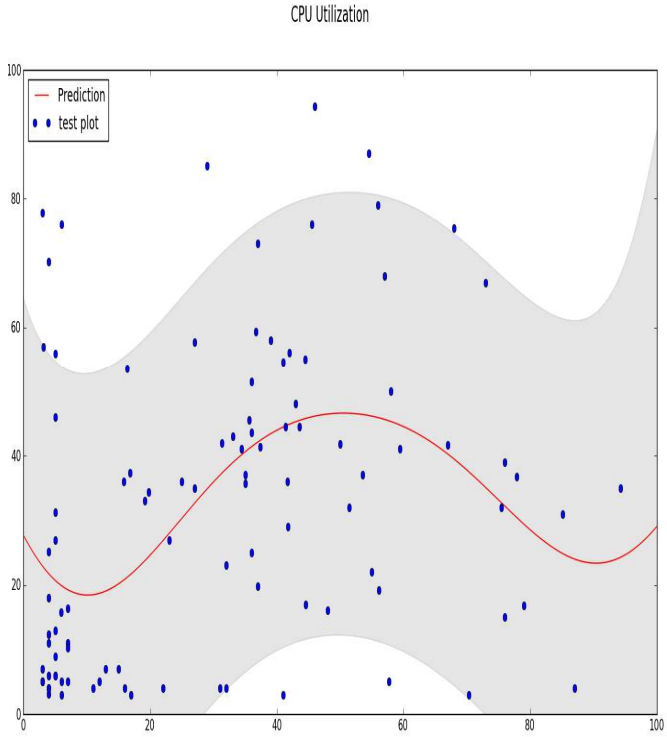

Transmitted bytes
Prediction
test plot
5000
4000
3000
2000
1000
0
0
20
40
60
80
100
Error Rate = 8.67

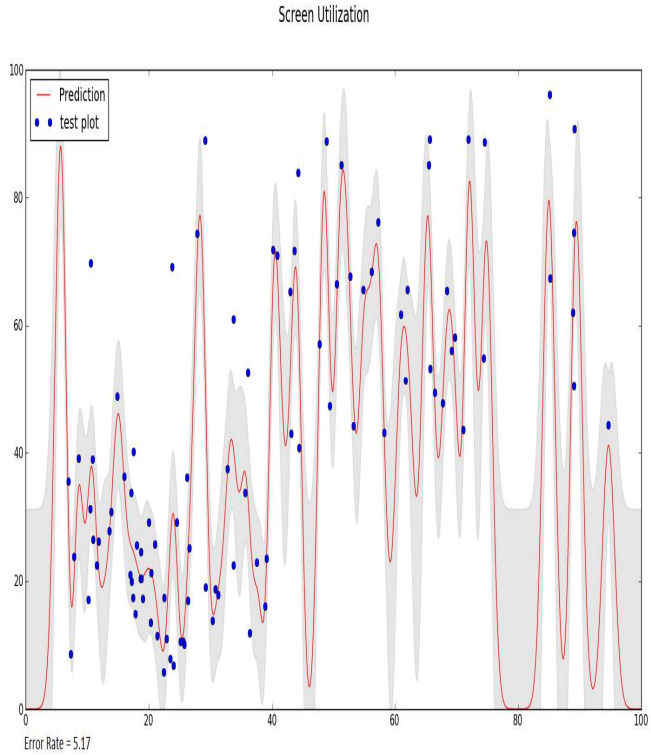

Transmitted bytes
Prediction
test plot
5000
4000
3000
2000
1000
0
0
20
40
60
80
100

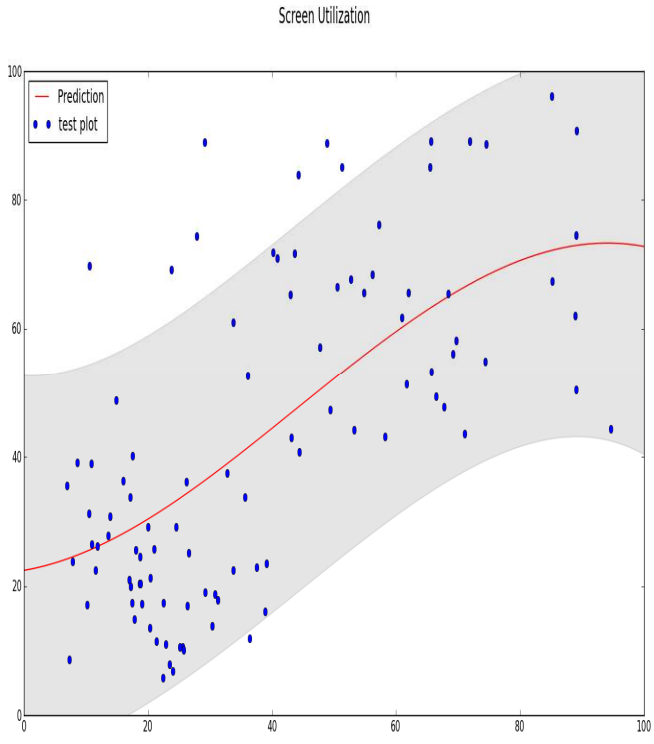
Results based on 10 second historical measures

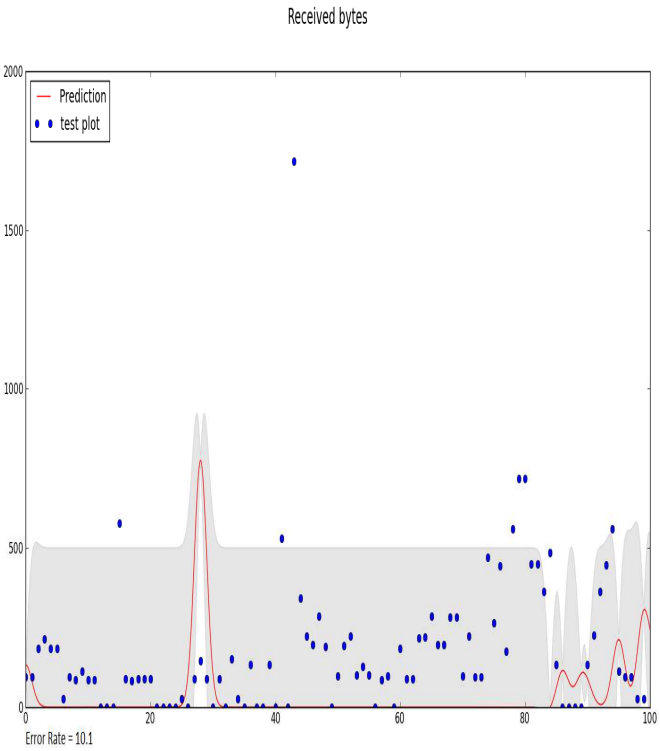
User 1

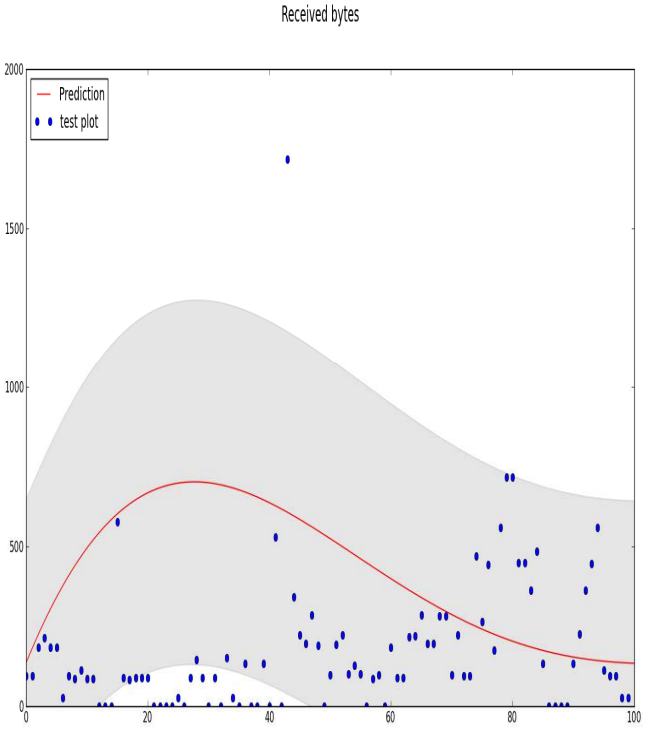


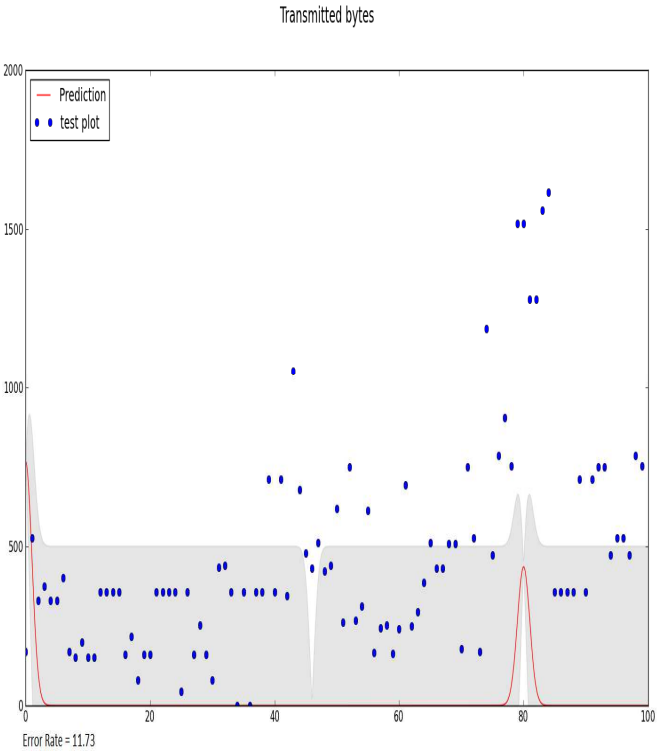


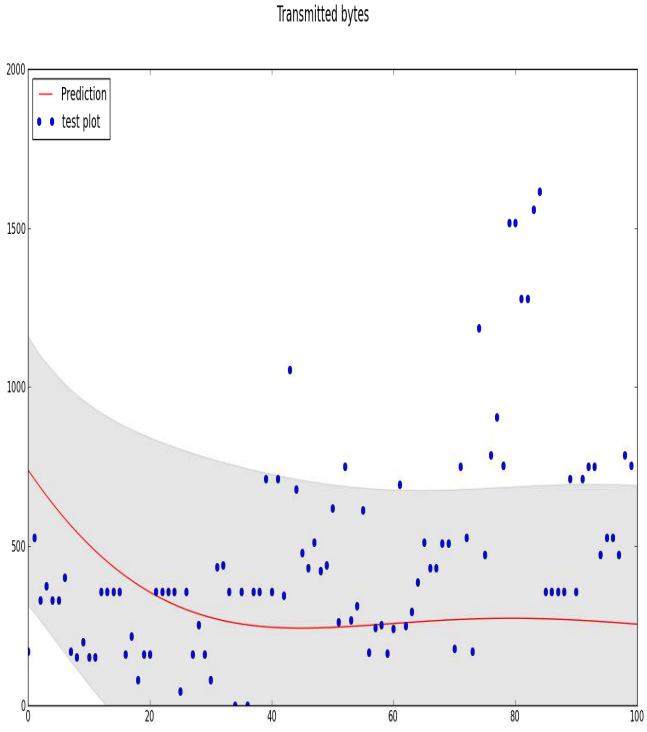




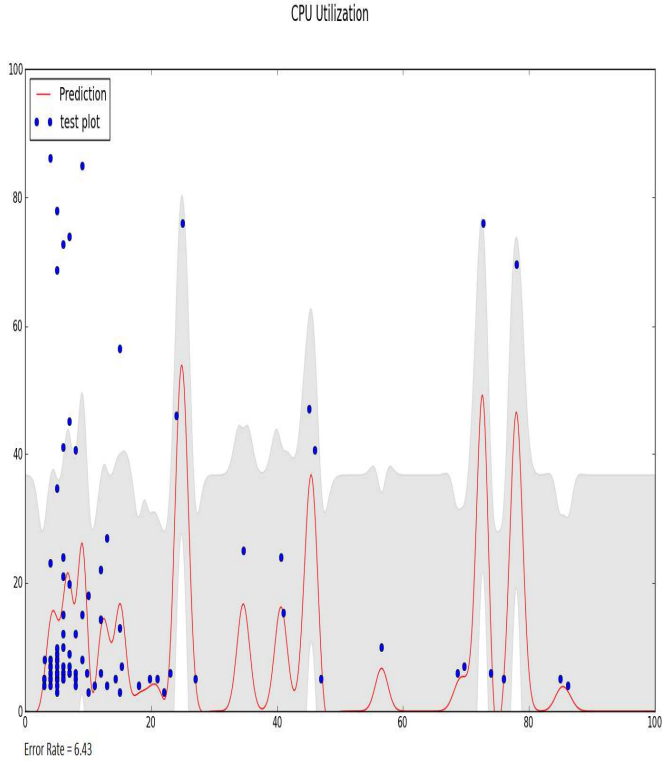

Received bytes
Prediction
test plot
Error Rate = 10.1

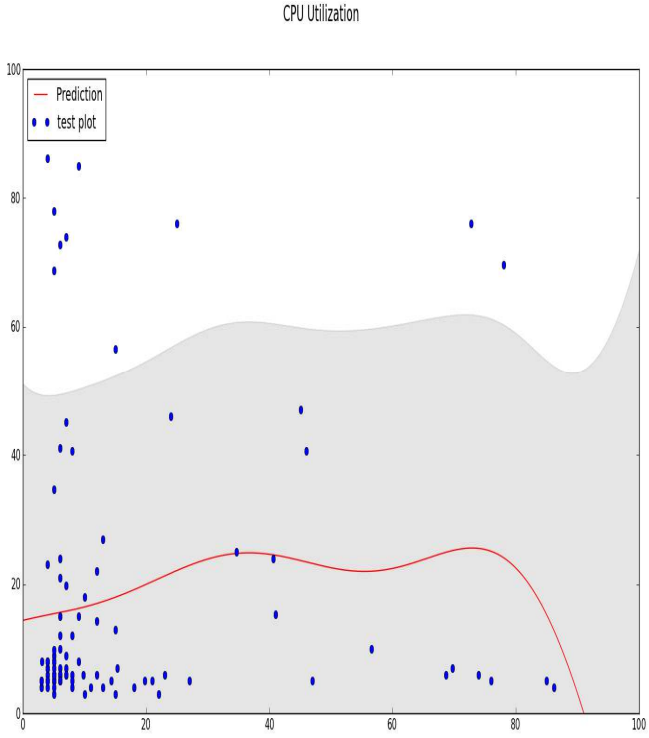

Received bytes
Prediction
test plot

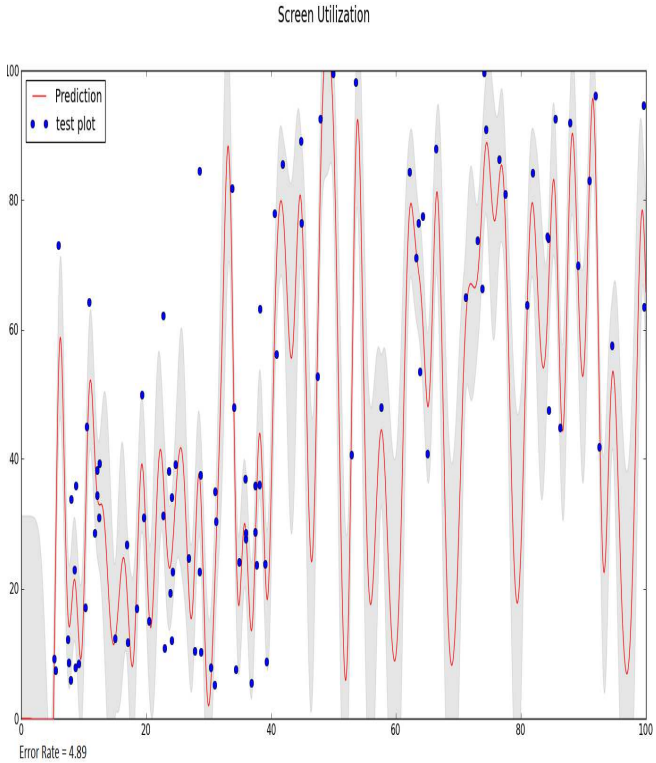

Transmitted bytes
Prediction
test plot
Error Rate = 11.73

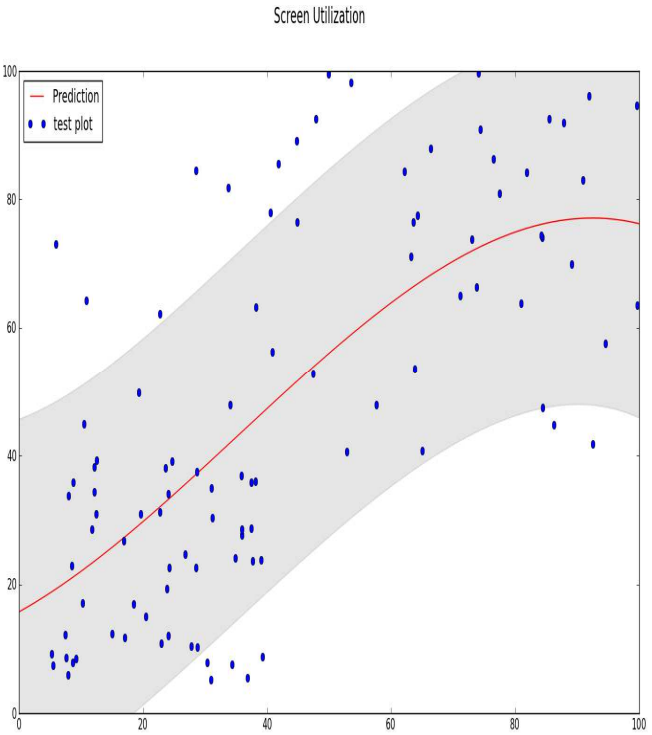

Transmitted bytes
Prediction
test plot

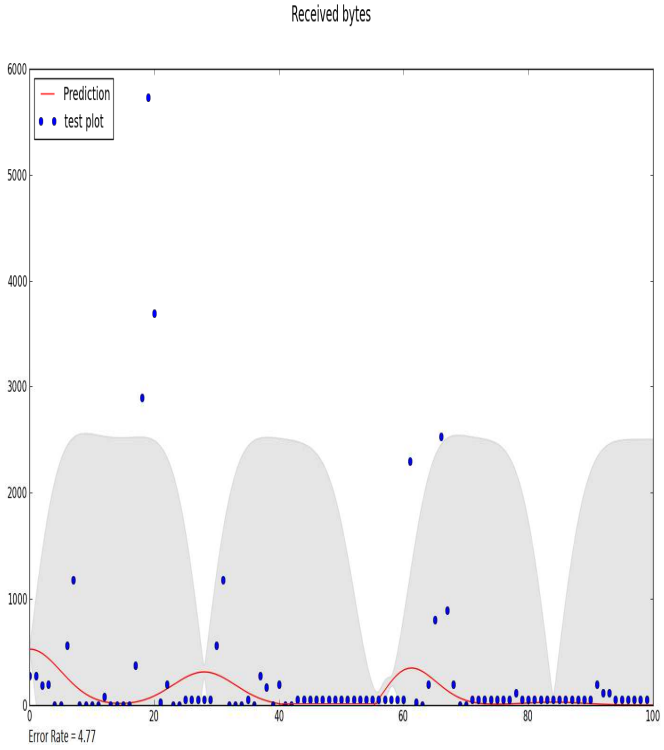
User 2

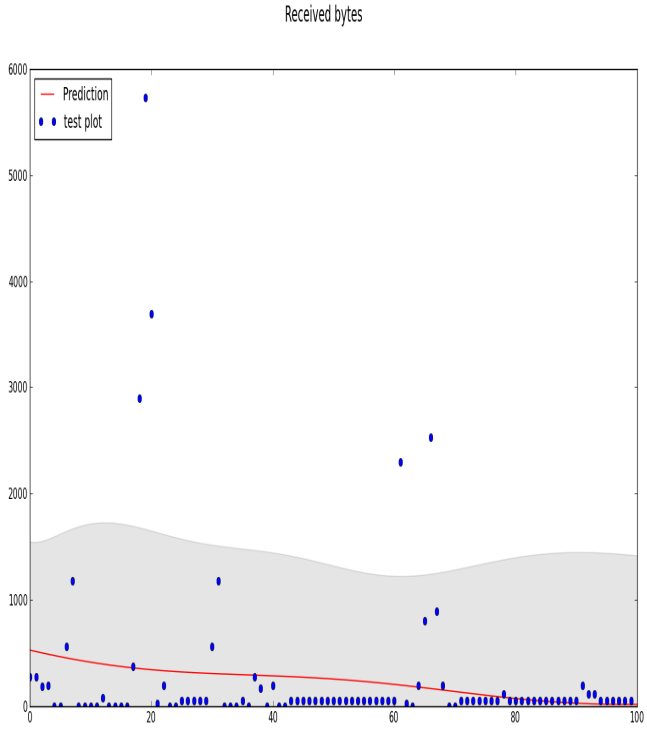

CPU Utilization
Prediction
test plot
Error Rate = 6.43

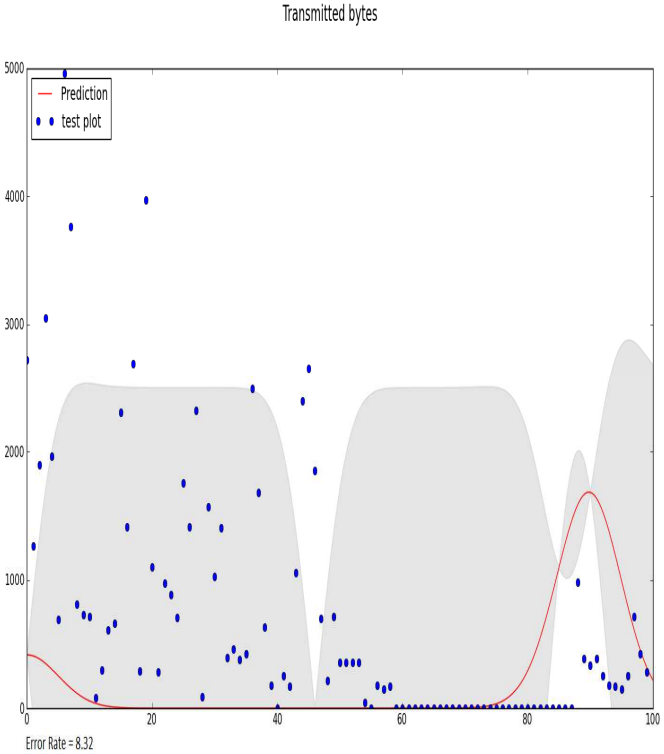

CPU Utilization
Prediction
test plot

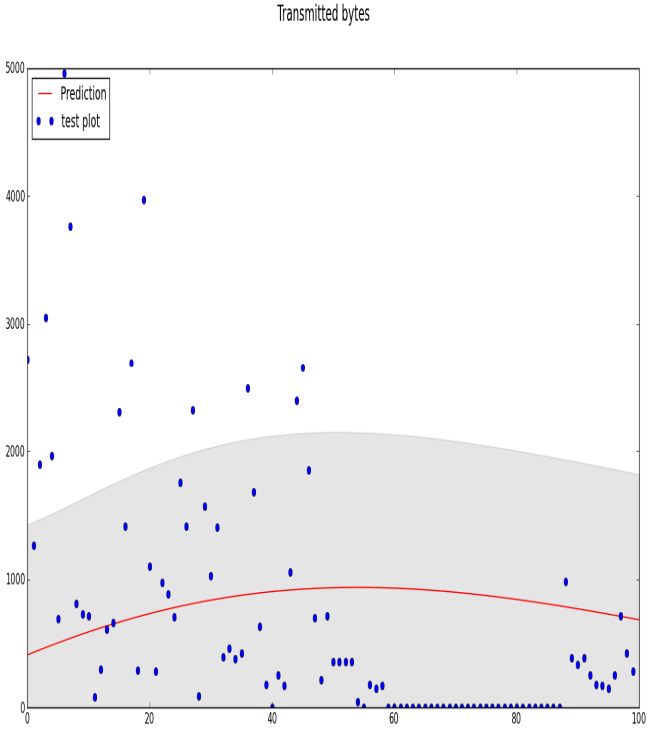

Screen Utilization
Prediction
test plot
Error Rate = 4.89


Screen Utilization
Prediction
test plot

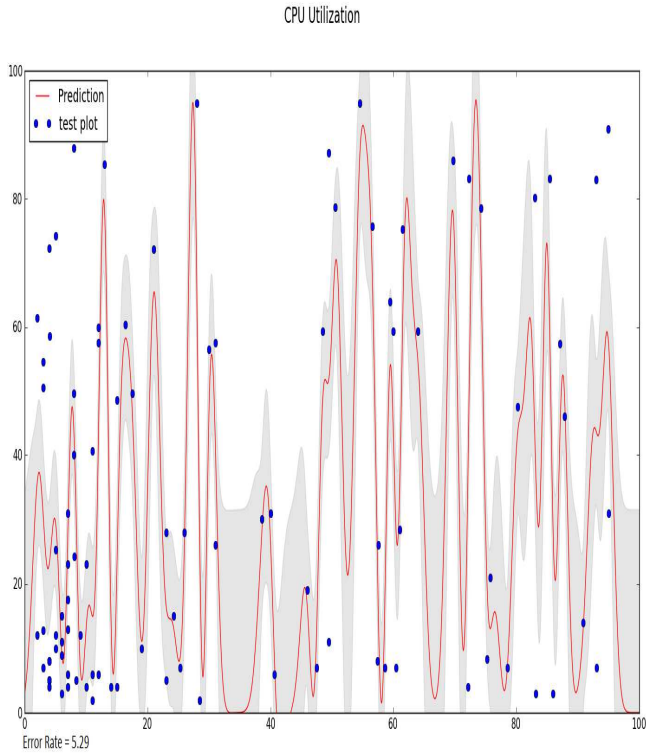

Received bytes
Prediction
test plot
Error Rate = 4.77

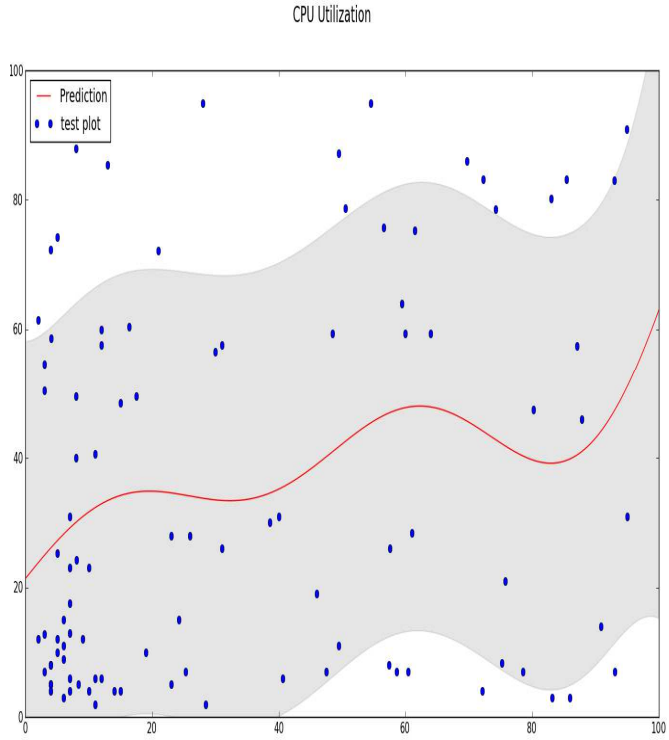

Received bytes
Prediction
test plot

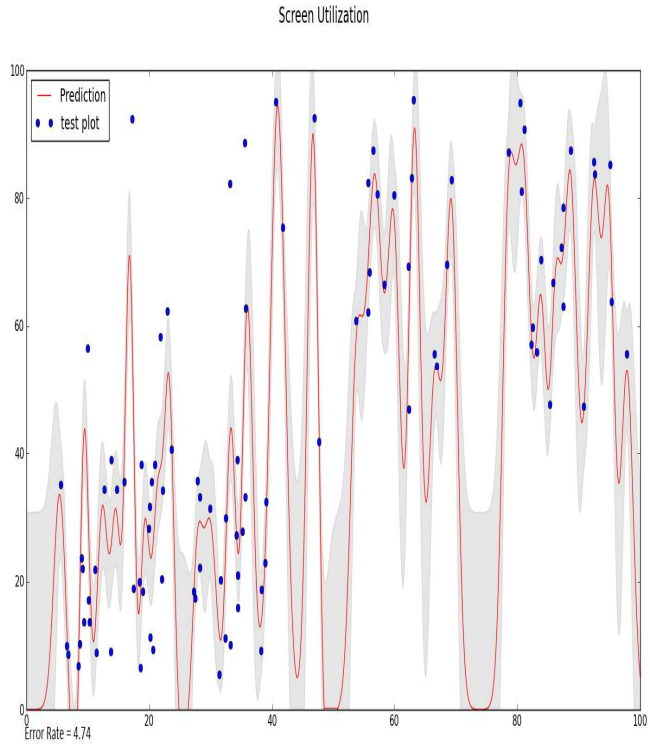

Transmitted bytes
Prediction
test plot
Error Rate = 8.32

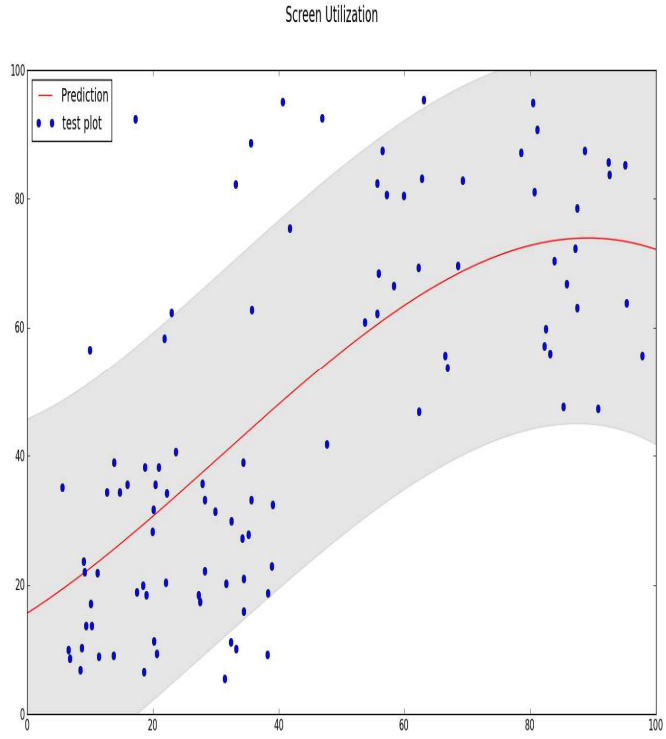

Transmitted bytes
Prediction
test plot

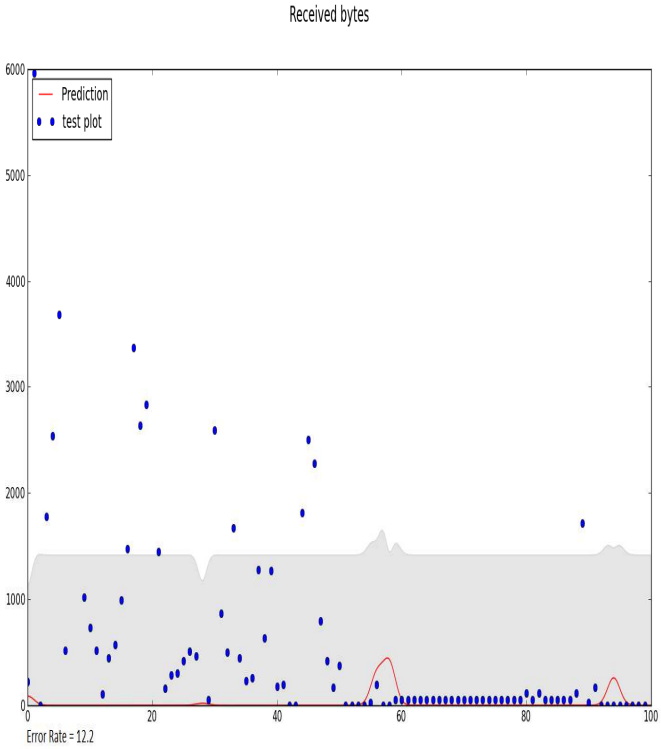
User 3

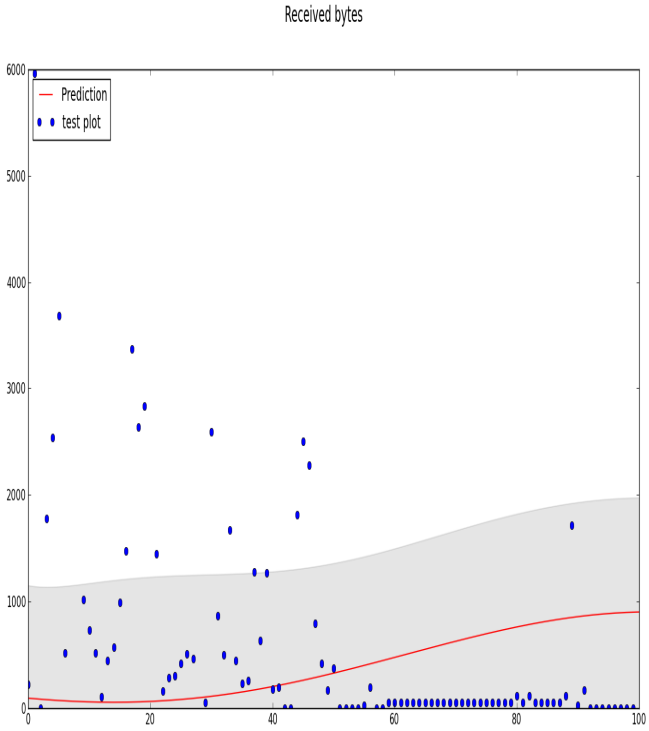

CPU Utilization
Prediction
test plot
Error Rate = 5.29

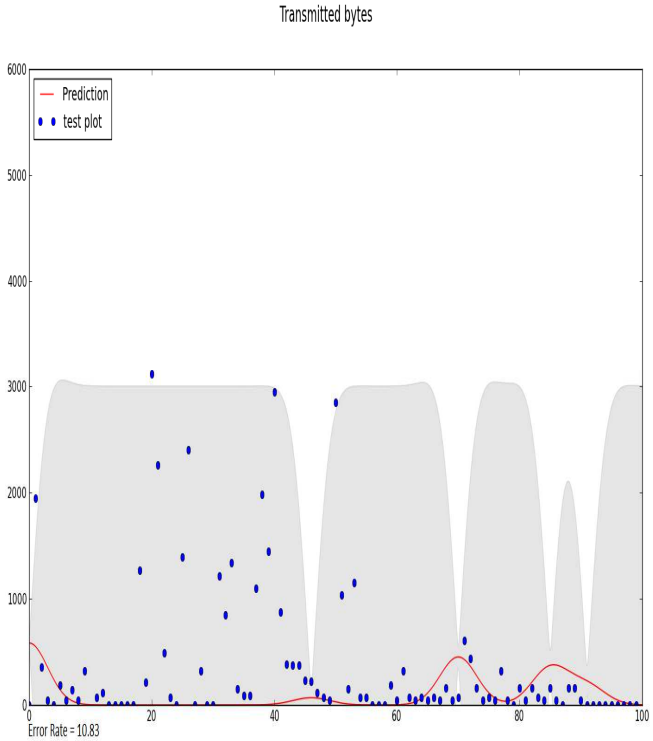

CPU Utilization
Prediction
test plot

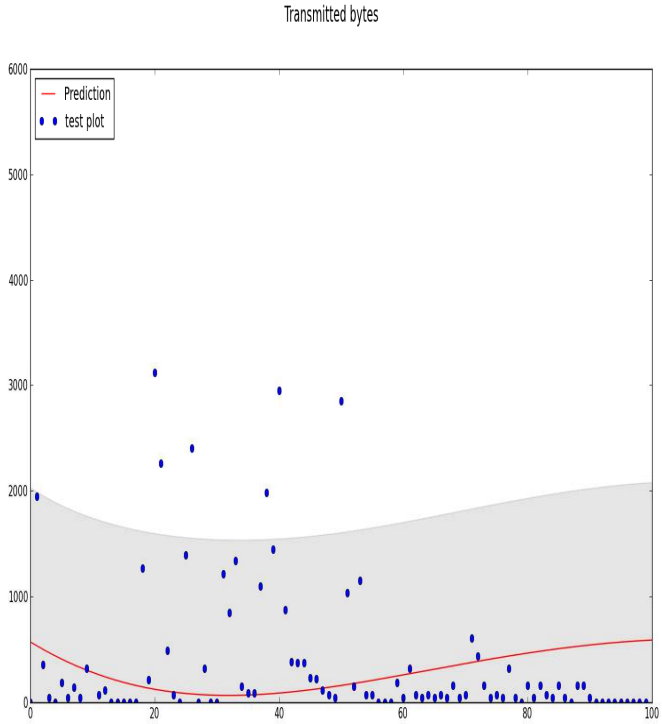

Screen Utilization
Prediction
test plot
Error Rate = 4.74


Screen Utilization
Prediction
test plot

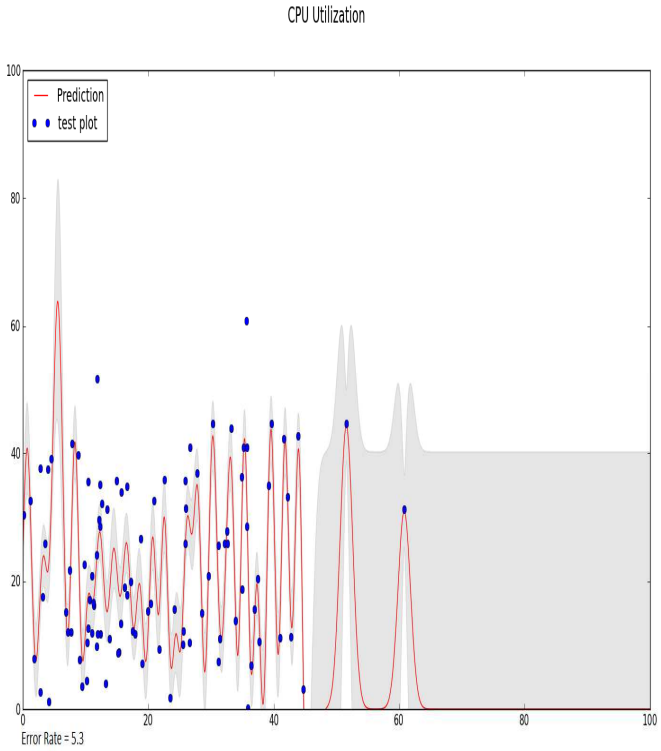

Received bytes
Prediction
test plot
Error Rate = 12.2

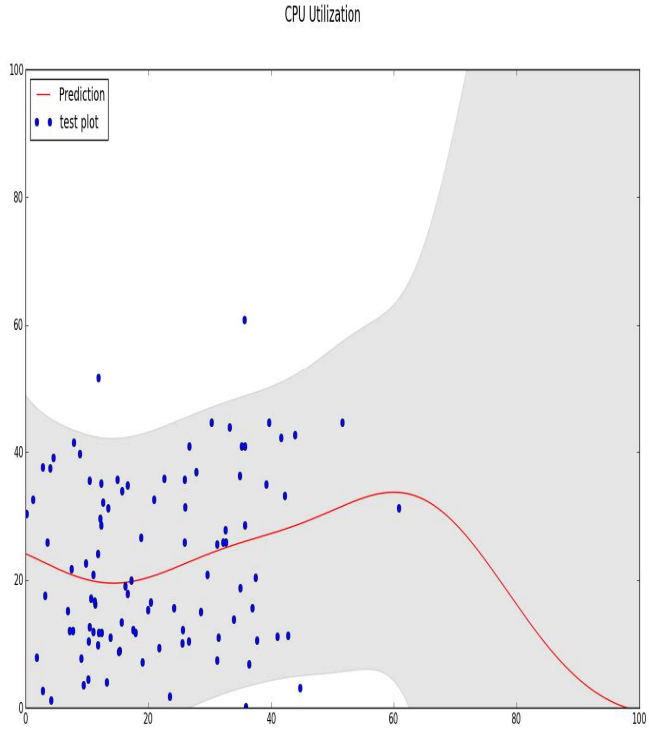

Received bytes
Prediction
test plot

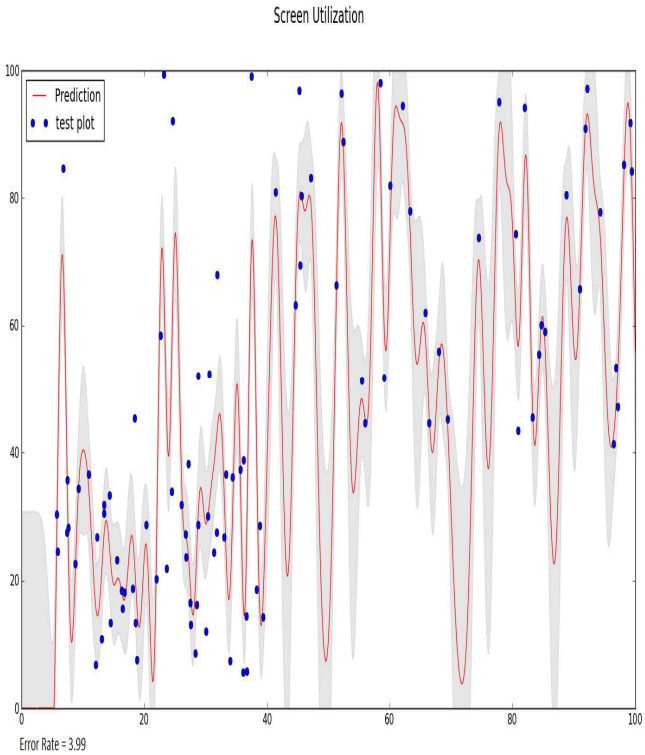

Transmitted bytes
Prediction
test plot
Error Rate = 10.83

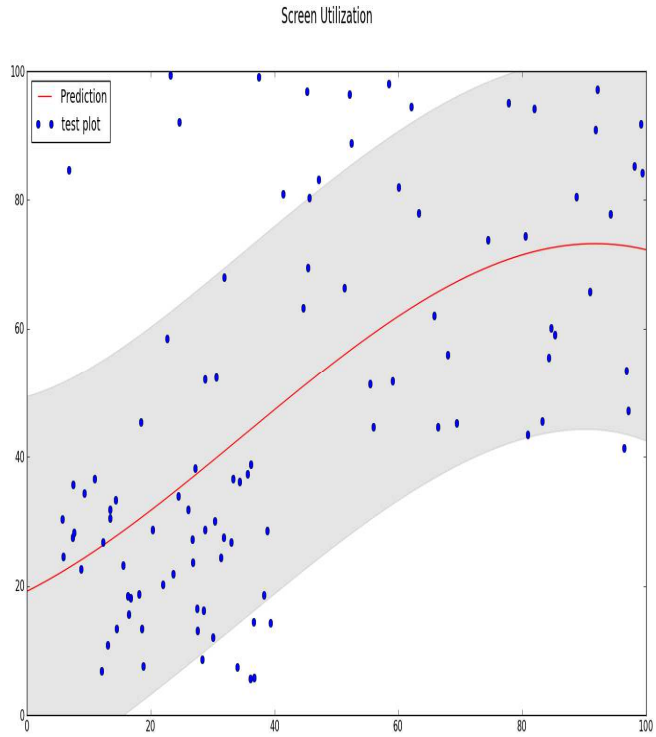

Transmitted bytes
Prediction
test plot

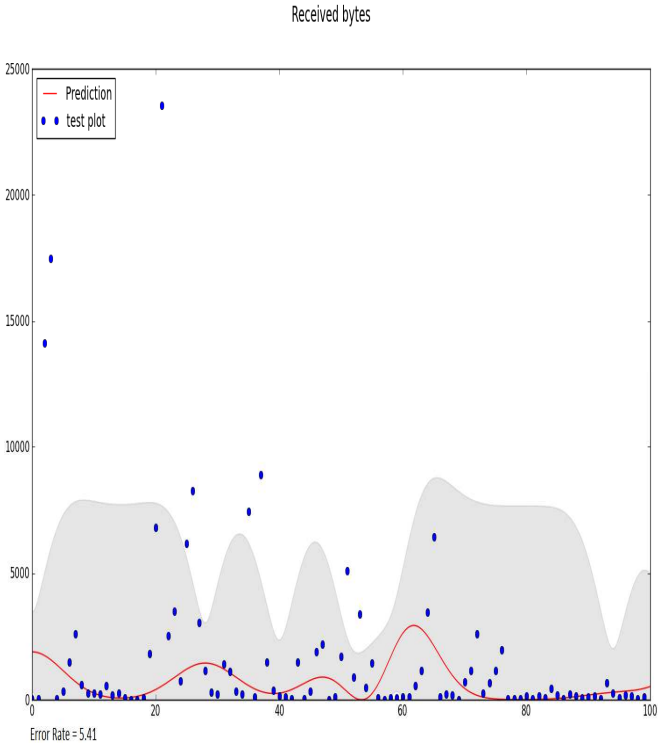
User 4

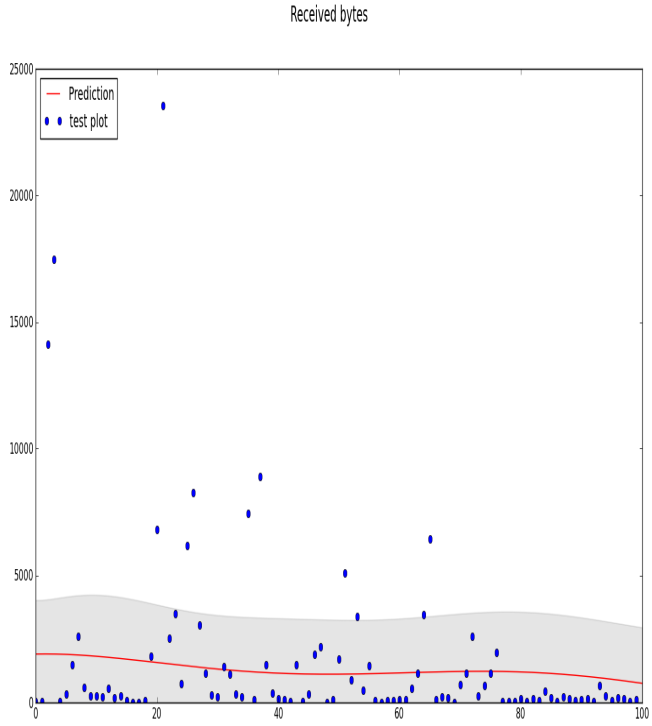

CPU Utilization
Prediction
test plot
Error Rate = 5.3

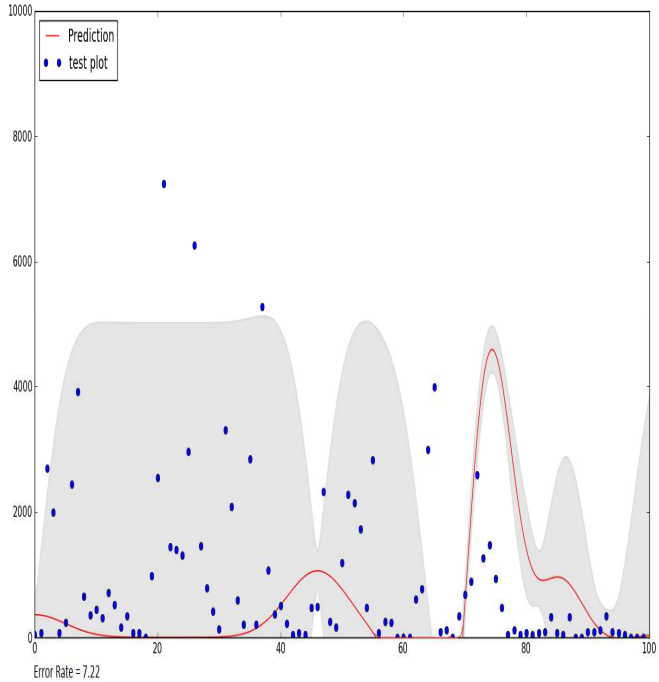

CPU Utilization
Prediction
test plot

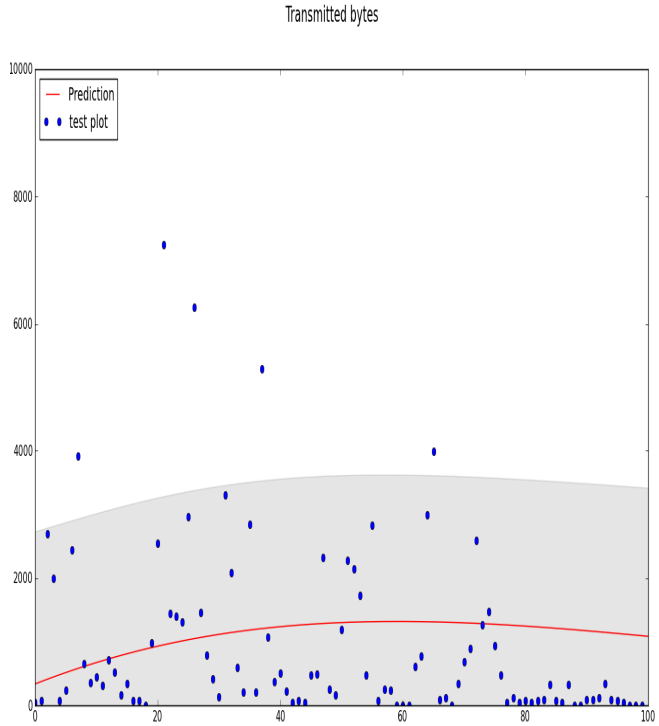

Screen Utilization
Prediction
test plot
Error Rate = 3.99


Screen Utilization
Prediction
test plot

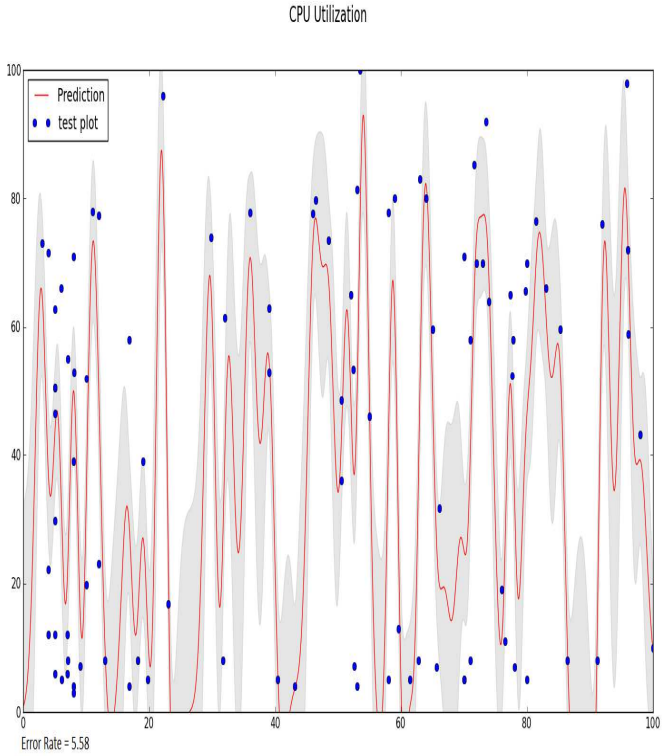

Received bytes
Prediction
test plot
25000
20000
15000
10000
5000
0
0
20
40
60
80
100
Error Rate = 5.41

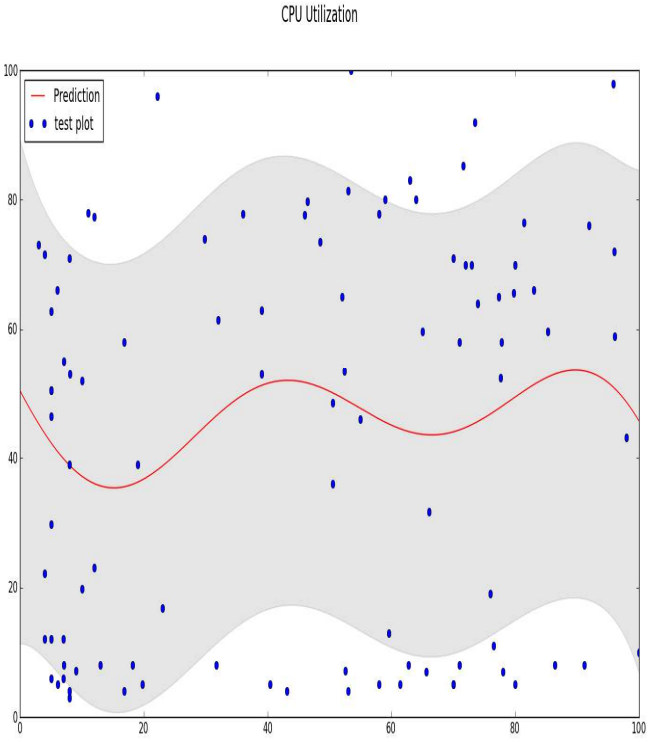

Received bytes
Prediction
test plot
25000
20000
15000
10000
5000
0
0
20
40
60
80
100

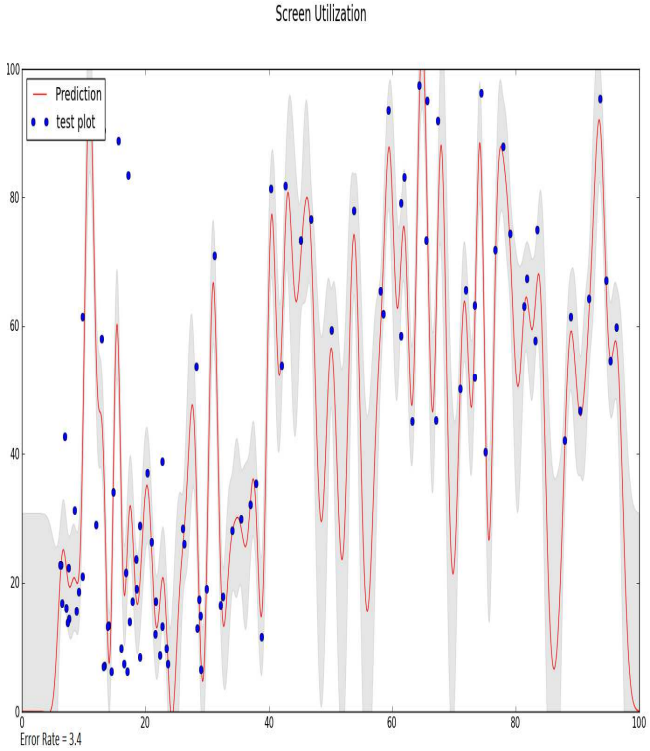

Transmitted bytes
Prediction
test plot
10000
8000
6000
4000
2000
0
0
20
40
60
80
100
Error Rate = 7.22

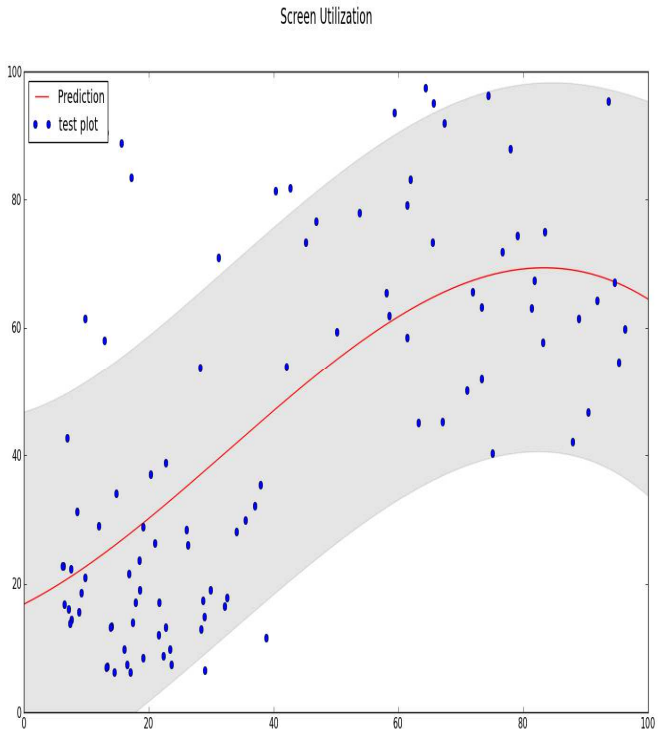

Transmitted bytes
Prediction
test plot
10000
8000
6000
4000
2000
0
0
20
40
60
80
100

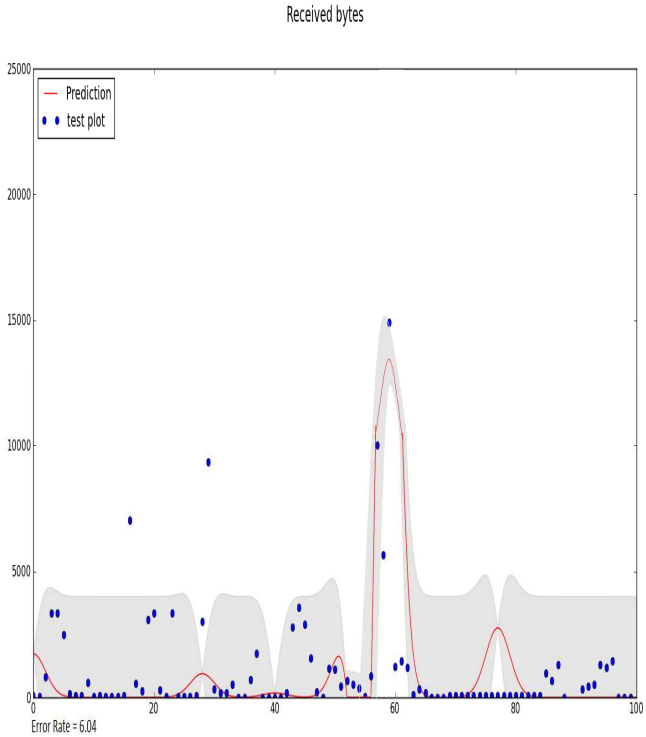
User 5

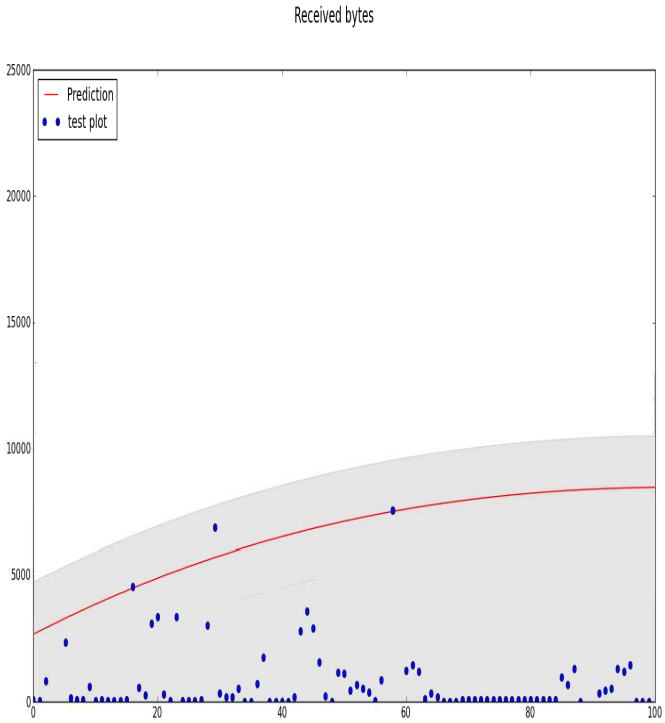

CPU Utilization
Prediction
test plot
Error Rate = 5.58

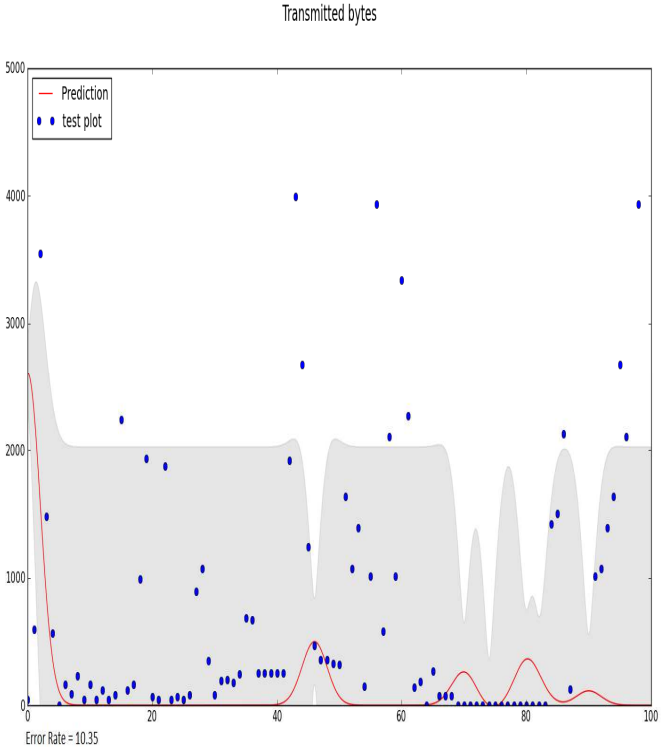

CPU Utilization
Prediction
test plot

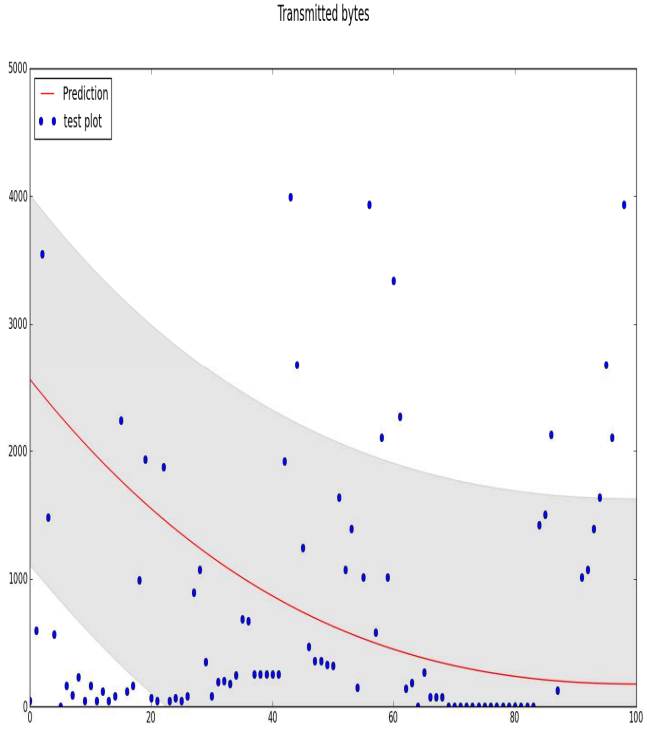

Screen Utilization
Prediction
test plot
Error Rate = 3.4


Screen Utilization
Prediction
test plot


Received bytes
Prediction
test plot
Error Rate = 6.04


Received bytes
Prediction
test plot


Transmitted bytes
Prediction
test plot
Error Rate = 10.35


Transmitted bytes
Prediction
test plot