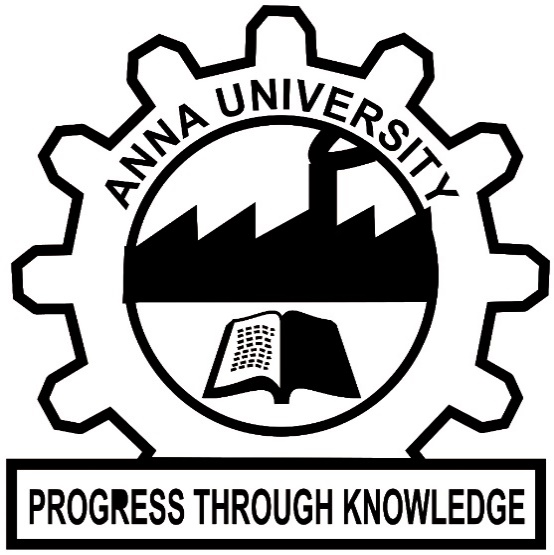
CS7711 – CREATIVE AND INNOVATIVE PROJECT

LOG ANOMALY DETECTION USING DEEP LEARNING TECHNIQUES



TEAM MEMBERS

S.MAKESH NARSIMHAN – 2015103036.

G.KRISHNA ANANDAN – 2015103596.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,

COLLEGE OF ENGINEERING, GUINDY, ANNA UNIVERSITY,

CHENNAI – 600025.

**ABSTRACT**

This document proposes an Anomaly detection and categorisation mechanism based on the voluminous amount of logs that is given as Input to the System. The application makes use of a LSTM model to detect anomalies from logs that are provided as input and uses a K-means clustering algorithm to categorise the detected anomalies into one of three possible clusters which happen to be Low, Medium and High.

The LSTM Model is one of the widely used Deep Learning Techniques. It makes use of an RNN that allows for better feedback to the successive layers. This helps the model to detect the anomalies from the logs with increased precision. The use of K-Means algorithm to cluster the detected anomalies allows the system to provide an in-depth analysis about the intensity of every detected anomaly.

**TABLE OF CONTENTS**

ABSTRACT – ENGLISH………………………………II

LIST OF FIGURES……………………………………..V

LIST OF TABLES………………………………………VI

LIST OF ABBREVIATIONS…………………………...VII

1. INTRODUCTION ………………………………………..1  
   1.1 PROBLEM DOMAIN ………………………………..1  
   1.2 PROBLEM DESCRIPTION ………………………....2  
   1.3 SCOPE ………………………………………………..2  
   1.4 CONTRIBUTION ……………………………………2  
   1.5 SWOT ANALYSIS …………………………………..3
   * 1. STRENGTHS……………………………….3
     2. WEAKNESS………………………………..3
     3. OPPORTUNITIES …………………………4
     4. THREATS ………………………………….4
   1. PESTEL ANALYSIS ………………………………4
      1. POLITICAL ………………………………..4
      2. ECONOMIC ……………………………….4
      3. SOCIAL ……………………………………5
      4. TECHNOLOGOICAL ……………………..5
      5. ENVIRONMENTAL ………………………5
      6. LEGAL …………………………………….5
2. RELATED WORK ……………………………………….6

2.1 LOGISTIC REGRESSION ………………………….6

2.2 DECISION TREE …………………………………...7

2.3 SUPPORT VECTOR MACHINES …………………8

2.4 LOG CLUSTERING ………………………………...8

2.5 PRINCIPAL COMPONENT ANALYSIS …………..9

2.6 INVARIANTS MINING …………………………….10

2.7 LONG SHORT TERM MEMORY…………………..10

2.8 OBSERVATIONS FROM THE SURVEY ……….....12

1. REQUIREMENT ANALYSIS ……………………………13

3.1 FUNCTIONAL REQUIREMENTS…………………..13

3.2 NON-FUNCTIONAL REQUIREMENTS……………13

3.2.1 HARDWARE ………………………………..13

3.2.2 USER INTERFACE …………………………13

3.2.3 SOFTWARE ………………………………....14

3.2.4 PERFORMANCE ……………………………14

3.3 CONSTRAINTS ……………………………………...14

3.4 ASSUMPTIONS ……………………………………...14

3.5 SYSTEM MODELS …………………………………..15  
 3.5.1 USE CASE MODEL …………………………15

3.5.2 SYSTEM SEQUENCE DIAGRAM …………16

4 SYSTEM DESIGN …………………………………………17

4.1 SYSTEM ARCHITECTURE …………………………17

4.2 UI DESIGN …………………………………………...18

4.3 CLASS DIAGRAM …………………………………..18

4.4 MODULE DESIGN …………………………………..19

4.4.1 LOGHUB DATASET ………………………..19

4.4.2 LOG PARSING AND FILTERING …………19

4.4.3 FEATURE EXTRACTION ………………….19

4.4.4 CLASSIFICATION (LSTM) ………………...20

4.4.5 K-MEANS CLUSTERING FOR   
 CATEGORIZING ANOMALIES ……………21

4.4.6 ACCURACY ESTIMATION ………………..21

4.5 COMPLEXITY ANALYSIS …………………………22

4.5.1 TIME COMPLEXITY ……………………….22

4.5.2 COMPLEXITY OF THE PROJECT ………...22

5 SYSTEM DEVELOPMENT ……………………………….23

5.1 PROTOTYPE ACROSS THE MODULES …………..24

5.2 LONG SHORT TERM MEMORY ALGORITHM …..25  
 5.3 K-MEANS CLUSTERING ALGORITHM …………..25

5.4 DEPLOYMENT DETAILS …………………………..26

6 RESULTS AND DISCUSSION ……………………………27

6.1 DATASET FOR TESTING …………………………..27

6.2 OUTPUT AT VARIOUS STAGES …………………..27

6.2.1 INPUT STATEMENTS ……………………...27

6.2.2 LOG PARSING AND FILTERING …………28

6.2.3 FEATURE EXTRACTION ………………….28

6.2.4 LSTM DETECTION …………………………28

6.2.5 CLUSTERING ANOMALIES……………… 29

6.3 SAMPLE SCREENSHOTS FROM TESTING ……...29

**LIST OF FIGURES**

2.1 The Standard Logistic Function ………………………..7

2.2 PCA Representation ……………………………………9

2.3 LSTM with Forget Gates ………………………………11

3.1 Use Case Diagram for the Overall System …………….15

3.2 System Sequence Diagram for the Overall System ……16

4.1 System Architecture Diagram ………………………….17

4.2 Class Diagram for the Overall System …………………18

4.3 Sliding window used in Feature Extraction. …………....20

5.1 LogHub Dataset ………………………………………...23

5.2 Logparser Directory Structure ………………………….23

5.3 LSTM Directory Structure ……………………………...24

6.1 Input Logs ………………………………………………27

6.2 Templated Logs ………………………………………...28

6.3 Logs with Anomaly Information ……………………….28

6.4 Anomalies in all 3 Clusters …………………………….29

6.5 Input Log Statements…………………………………...30

6.6 Weight Graph Indicating Detection of anomalies………30

6.7 Graph indicating the prediction of anomalies…………..31

6.8 Graph indicating the categories of anomalies ………….31

**LIST OF TABLES**

4.1 Time Complexity of Various Modules …………………..22

**LIST OF ABBREVIATIONS**

CART Classification and Regression Trees

CEC Constant Error Carousel

ECG Electro Cardio Gram

LSTM Long Short Term Memory

PCA Principal Component Analysis

RNN Recurrent Neural Network

SVM Support Vector Machine

**CHAPTER 1**

**INTRODUCTION**

* 1. **PROBLEM DOMAIN**

Anomaly detection refers to the problem of finding instances or patterns in data that deviate from normal behaviour. Depending on context and domain these deviations can be referred to as anomalies, outliers, or novelties. Anomaly detection is utilized in a wide array of fields such as fraud detection for financial transactions, fault detection in industrial systems, intrusion detection, and many more applications. Anomaly detection is important because anomalies often indicate useful, critical, and actionable information that can benefit businesses and organizations.

Anomalies can be classified into four categories:

1. *Point Anomalies*: A data point is considered a point anomaly if it is considerably different from rest of the data points. Extreme values in a dataset lie in this category.

2. *Collective Anomalies*: If there is a set of related points which are normal individually but anomalous if taken together, then the set is a collective anomaly. Time series sequences which deviate from the usual pattern come under collective anomalies.

3. *Contextual Anomalies*: If a data point is abnormal when viewed in a particular context but normal otherwise it is regarded as a contextual anomaly. Context is often present in the form of an additional variable e.g. temporal or spatial attribute. A point or collective anomaly can be a contextual anomaly if some contextual attribute is present.

In this project, we have developed a Log and Anomaly Detection System which goes a step further and categorises the anomaly based on its intensity and the potential damage it can cause to the Software solution and ultimately the Native system itself.

The application uses Long Short-Term Memory to accomplish the process of Anomaly Detection. For the purpose of Clustering the Logs and Anomalies for their categorization, the application makes use of the K-Means Clustering Algorithm.

* 1. **PROBLEM DESCRIPTION**

Given a Large set of Logs from either a Software Solution or a Large Scale System, the system must parse the logs and group them based on textual similarity (templates) whilst preserving the intricate details of the logs and discarding the unnecessary ones. The parsed and categorized logs are then fed into the Long Short-term Memory Model, which detects the possible anomalies that may arise from the Log Templates provided as input. The detected anomalies are then categorized via a K-Means implementation to differentiate them based on the level of potential damage they can cause in the smooth functioning of their native system.

* 1. **SCOPE**

The connectivity of Modern Day Networks requires a quick analysis methodology in case of a breakdown. Distributed network clusters encompass thousands of nodes, making manual methods of pinpointing errors and anomalies onerous. Systems cannot afford to go down even for a second these days owing to their popularity and unprecedented traffic they experience from various parts of the world. In such a situation, Developers need to be made aware of what and how the problem was caused the moment there’s a stagnation. If the alert also specifies how serious the stagnation is, it helps make an agile decision and as to whether it has to be dealt with immediately. With such constraints in place, the use of an anomaly detection mechanism coupled with categorization, would be very useful for organizations to plan strategies better and improve performance.

* 1. **CONTRIBUTION**

The area of Log and Anomaly research is very nascent. With increasing complexity in the physical and logical structures of Systems and Networks worldwide, maintenance of such geo-spatial setups and their fault tolerance mechanisms are major areas of concern. Our Anomaly detection and categorization applications serves both purposes efficiently. The application is capable of processing millions of Logs together and group them effectively using a generic parsing mechanism. The Long Short Term Memory Model is a promising Deep Learning Technique that promises to provide in depth analysis of every log and detect potential anomalies with precision. This helps organisations monitor their systems effectively and nullify potential threats whenever detected. The use of such a modern technique enhances the already existing fault tolerant mechanisms by letting the system know of impending threats and snub them before the threat becomes a reality. From the results that the system has produced using the Long Short Term Memory Model, it is evident that this application is one of the most advanced systems in the field of anomaly detection.

* 1. **SWOT ANALYSIS**

**1.5.1 STRENGTHS**

The application is concerned with a blooming field in Computer Science and in the digital era, the field of Anomaly Research is sure to expand exponentially in the coming days. Deep Learning is another prospective study which has shown a lot of promise with respect to handling complex and voluminous data and providing accurate predictions compared to Machine Learning. A generic prediction and categorization model that is able to process any set of logs regardless of which software solution it belongs to. A structured approach from module to module which forms a linear flow of data throughout the application with minimal complexity.

**1.5.2 WEAKNESS**

Scalability with respect to volume of data. Isolating the Statistically Significant data from Noise. Fixing the value of window size for filtering the data has not been addressed as of now. Fixing the transition time for sliding window algorithm of Log Parsing. The current duration, widely used, is 10800 seconds (~ 6 hours). Clustering algorithms are highly susceptible to outliers as they are based on mean while invariant mining algorithm focuses on identifying the outliers. Thus, the different approaches use outliers in a contrasting manner.

Adversarial impacts are far-reaching. The models trained are highly specific to a particular product/language

**1.5.3 OPPORTUNITIES**

The application in the future can be envisioned to serve its purpose for real time anomaly detection and categorization. This can be achieved by following a modular approach to the developed Long Term Short Memory Model. The increase in Fault tolerance levels of Distributed Systems and Networks would help organisation to deliver better performance to their customers and increase their commercial quotient.

**1.5.4 THREATS**

The impending threat from the Cyber World allows attackers to bypass the detection mechanisms and disrupt the system by their ominous means. With a protection layer in place that hides the detection mechanism, we can increase the security of the native application as well. Though the model accommodates for Scalability, the tightest upper bound of the level of scalability to which the model can work is still unknown.

**1.6 PESTEL ANALYSIS**

**1.6.1 POLITICAL**

With increasing concerns of native data storage and Cloud intensive systems all over the world, Governments have an increasing concern with respect to data security. This forces organisations to decentralize the networks into the continents of their presence. Such an anomaly detection mechanism would help organisations to geo-spatially pinpoint the exact location and resolve it as soon as possible. Software made for private use of the Government such as defence and intranets can be well protected from Cyber Attacks.

**1.6.2** **ECONOMIC**

The use of such an anomaly categorization mechanism allows organisations to take remedial measures in a structured manner. The more intensive anomalies are tended to first and allocation of monetary aid to resolving it is more prioritized. A decentralized system would mean a great reduction in Logistical and Cost of components to establish networks across a Geo-Spatial region.

1.6.3 **SOCIAL**

Data is the greatest fuel of the modern era. Protecting the data of billions of users worldwide is a big challenge for Multi-National Organisations. To protect such data from Cyber Attackers, they will need to remove all vulnerabilities from their network so that it’s closely knit. Even if the network is penetrated, the anomaly detection mechanism sends out an alert about a foreign element in the network and helps the organisation undertake proactive mechanisms to restore normalcy.

**1.6.4** **TECHNOLOGICAL**

The Anomaly detection and categorisation mechanism employs long short term Memory (LSTM), a state of the art Deep Learning technique for performing Anomaly Detection. For the purpose of Categorisation, the application makes use of K-Means Clustering. Both techniques are the most widely used and dependable algorithmic solutions available to suit the need for such intense and voluminous data processing and feature selection. LogHub is the dataset that is used for training and testing the model. Loghub is a vast dataset that consists of Log Statements from every perceivable Software or Operating System and hence promotes the most generic platform for training and testing the model. The application can be migrated to a better and more efficient model in the future which showcases the level of flexibility the problem possesses. The use of Deep Learning allows the model to scale as per the amount of data given for processing.

**1.6.5** **ENVIRONMENTAL**

The applied ideologies and strategies bear no harm to the deployed environment. Power consumption would be a drawback considering the complexity of the model and the level of hardware it would require for smooth functioning.

**1.6.6** **LEGAL**

Attacks can come from far and wide and any sort of legal action can be taken under International Laws against the attacker. Detection Mechanisms could help organisations in pinpointing the cause and effect of a particular attack and rectify it forever. It helps them handle their clients well.

**CHAPTER 2**

**RELATED WORKS**

This chapter gives an insight to the different methodologies that can be adopted for performing Log and Anomaly Detection. 6 of these methods were based on the Research paper that we referenced in a quest to find the proper technique that we could use for the problem at hand. This survey helped us evaluate each method on several parameters.

**2.1 LOGISTIC REGERESSION**

In statistics, the logistic model is a statistical model that is usually taken to apply to a binary dependent variable. In regression analysis, logistic regression is estimating the parameters of a logistic model. More formally, a logistic model is one where the log-odds of the probability of an event is a linear combination of independent or predictor variables. The two possible dependent variable values are often labelled as "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. The binary logistic regression model can be generalized to more than two levels of the dependent variable: categorical outputs with more than two values are modelled by multinomial logistic regression, and if the multiple categories are ordered, by ordinal logistic regression, for example the proportional odds ordinal logistic model.

The technique can also be used in engineering, especially for predicting the probability of failure of a given process, system or product. It is also used in marketing applications such as prediction of a customer's propensity to purchase a product or halt a subscription, etc. In economics it can be used to predict the likelihood of a person's choosing to be in the labour force, and a business application would be to predict the likelihood of a homeowner defaulting on a mortgage.



Fig 2.1 The Standard Logistic Function.

**2.2 DECISION TREE**

In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal, it’s also widely used in machine learning. Although, a real dataset will have a lot more features and this will just be a branch in a much bigger tree, but you can’t ignore the simplicity of this algorithm. The **feature importance is clear** and relations can be viewed easily. This methodology is more commonly known as **learning decision tree from data** and above tree is called **Classification tree** as the target is to classify passenger as survived or died. **Regression trees** are represented in the same manner, just they predict continuous values like price of a house. In general, Decision Tree algorithms are referred to as CART or Classification and Regression Trees.

The cost function used for Regression:

*Regression: sum(y — prediction)²*

The cost function used for Classification:

*Classification: G = sum (pk \* (1 — pk))*

Where

pk = proportion of same class inputs present in a particular group.

**2.3 SUPPORT VECTOR MACHINES**

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyper plane. In other words, given labelled training data, the algorithm outputs an optimal hyper plane which categorizes new examples. In two dimensional space this hyper plane is a line dividing a plane in two parts where in each class lay in either side. The learning of the hyper plane in linear SVM is done by transforming the problem using some linear algebra. Kernels are used for this purpose

For **linear kernel** the equation for prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows:

*F(x) = B(0) + sum(ai \* (x,xi))*

The Regularization parameter tells the SVM optimization how much you want to avoid misclassifying each training example.

The gamma parameter defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close’. In other words, with low gamma, points far away from plausible separation line are considered in calculation for the separation line. Whereas high gamma means the points close to plausible line are considered in calculation.

**2.4 LOG CLUSTERING**

The nature of the data to be clustered plays a key role when choosing the right algorithm for clustering. Most of the clustering algorithms have been designed for generic data have no specific assumptions about the nature of data are made. However, when we inspect the content of typical log files at the word level, there are two important properties that distinguish log file data from a generic data set. Although it is impossible to verify that the properties discovered characterize every logfile ever created, they remain common to a wide range of logfile data sets. There are many strong correlations between words that occurred frequently. This effect is caused by the fact that a message is generally formatted according to a certain format string before it is logged.

The algorithm consists of three steps – it first makes a pass over the data and builds a data summary, and then makes another pass to build cluster candidates, using the summary information collected before. As a final step, clusters are selected from the set of candidates.

**2.5 PRINCIPAL COMPONENT ANALYSIS (PCA)**

The main idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of many variables correlated with each other, either heavily or lightly, while retaining the variation present in the dataset, up to the maximum extent. The same is done by transforming the variables to a new set of variables, which are known as the principal components and are orthogonal, ordered such that the retention of variation present in the original variables decreases as we move down in the order. So, in this way, the 1st principal component retains maximum variation that was present in the original components. The principal components are the eigenvectors of a covariance matrix, and hence they are orthogonal.

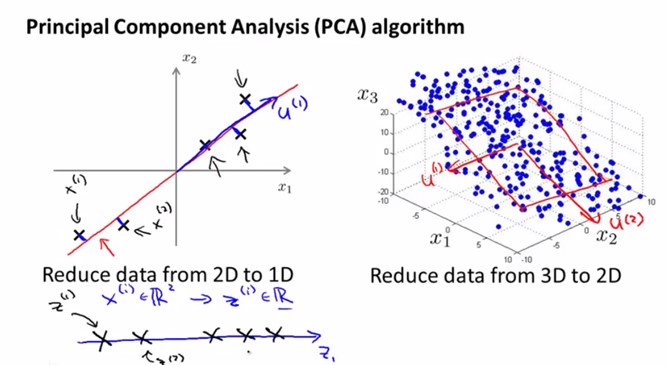


Fig 2.2 PCA Representation

The PCs possess some useful properties which are listed below:

* The PCs are essentially the linear combinations of the original variables, the weights vector in this combination is actually the eigenvector found which in turn satisfies the principle of least squares.
* The PCs are orthogonal by nature.
* The variation present in the PCs decrease as we move from the 1st PC to the last one, hence the importance.

**2.6 INVARIANTS MINING**

In general, a program invariant is a predicate that always holds the same value under different workloads or inputs. Program invariants can be defined from various aspects of a system, including system measurements (e.g. CPU and network utilization) and program variables. Besides the program variables and system measurements, program execution flows can also introduce invariants. With the assumption that log sequences provide enough information for the system execution paths, we can obtain invariants of program execution flows through analysing log sequences. Linear invariants encode meaningful characteristics of system execution paths. They universally exist in many standalone or distributed systems. An anomaly often manifests a different execution path from the normal ones. Therefore, a violation of such relations (invariants) means a program execution anomaly. Because log sequences record the underlying execution flow of the system components, we believe there are many such linear equations, i.e. invariants, among the log sequences. If we can automatically discover all such invariants from the collected historical log data, we can facilitate many system management tasks.

Each invariant contains a constraint or an attribute of a system component’s execution flow. Based on the related execution flow properties of the broken invariants, system operators can find the potential causes of failure. The invariants can help system operators to better understand the structure and behaviour of a system

**2.7 LONG SHORT TERM MEMORY MODEL**

LSTM can learn dependencies ranging over arbitrary long time intervals. LSTM overcome the vanishing gradients problem by replacing an ordinary neuron by a complex architecture called the LSTM unit or block. An LSTM unit is made up of simpler nodes connected in a specific way.

The main components of the LSTM architecture introduced in are:

1. *Constant error carousel (CEC)*: A central unit having a recurrent connection with a unit weight. The recurrent connection represents a feedback loop with a time step equal to 1. The CEC’s activation is the internal state which acts as the memory for past information.

2. *Input Gate*: A multiplicative unit which protects the information stored in CEC from disturbance by irrelevant inputs.

3. *Output Gate*: A multiplicative unit which protects other units from interference by information stored in CEC.

The input and output gate control access to the CEC. During training, the input gate learns when to let new information inside the CEC. As long as the input gate has a value of zero, no information is allowed inside. Similarly, the output gate learns when to let information flow from the CEC. When both gates are closed (activation around zero) information or activation is trapped inside the memory cell. This allows the error signals to flow across many time steps (aided by the recurrent edge with unit weight) without encountering the problem of vanishing gradients.

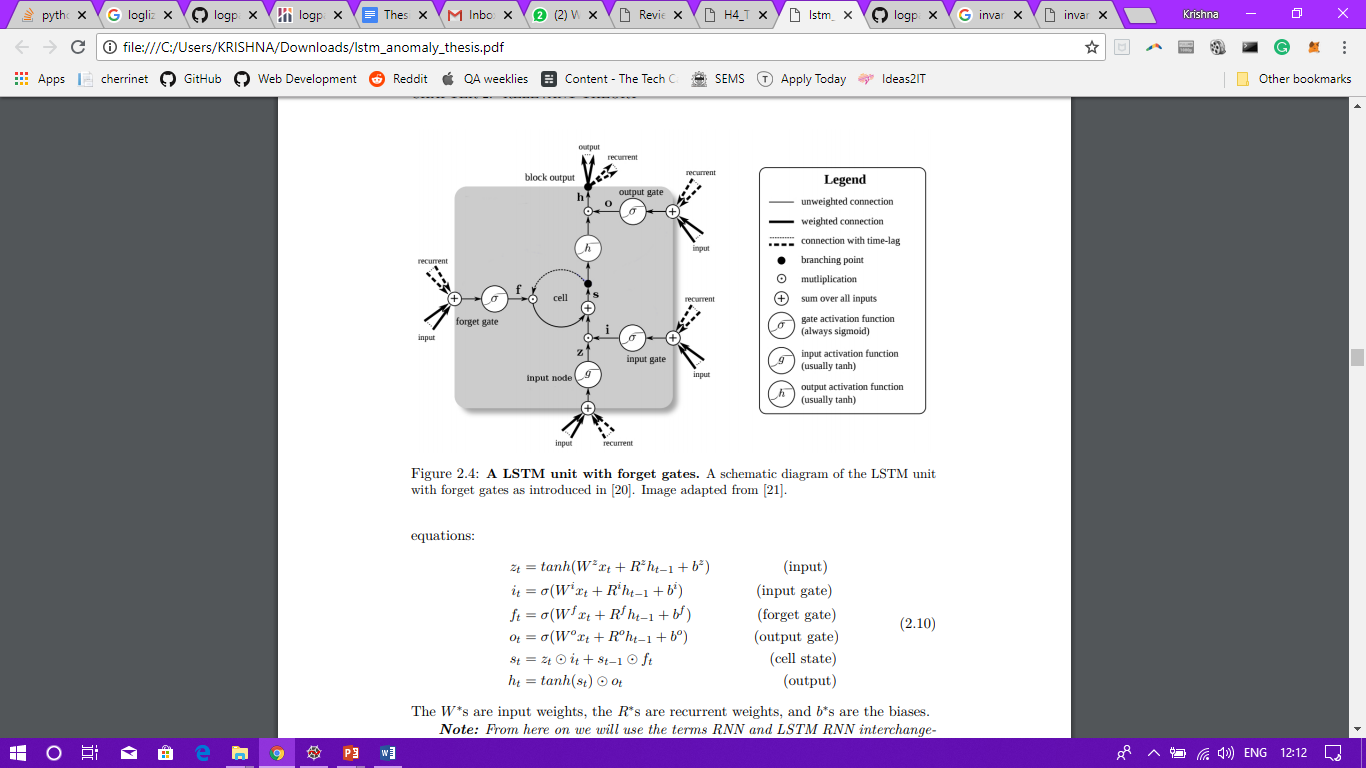
The forget gate is the mechanism through which an LSTM learns to reset the memory contents when they become old and are no longer relevant. This may happen for example when the network starts processing a new sequence. The forget gate reads xt and ht−1 and applies a sigmoid activation to weighted inputs. The result, ft is multiplied by the cell state at previous time step i.e. st−1 which allows for forgetting the memory contents which are no longer needed.

Fig 2.3 LSTM with Forget Gates

**2.8 OBSERVATIONS FROM THE SURVEY**

From the various methods that we surveyed it was clear that most of the methods employed naïve machine learning approaches. Only Long Short Term Model proved to be a standout method given the fact that it was far more advanced in terms of its architecture compared to the other six machine learning approaches.

**CHAPTER 3**

**REQUIREMENTS ANALYSIS**

**3.1 FUNCTIONAL REQUIREMENTS:**

The system categorises all anomalies that it detected from the generated log templates and the output must adhere to the following requirements:

* Every anomaly in every category must be produced as a result of detection by the LSTM algorithm.
* The K-Means clustering algorithm must use only the anomalies detected by the LSTM as Input.
* The Algorithms used to generate the Output must adhere to Time and Space Complexities and also address Scalability Issues.
* The System must be able to accept Logs from any Software Solution.
* The System must be capable of processing voluminous data efficiently.
* The Anomaly detection model must exhibit an acceptable level of accuracy to prove its functional correctness.
* The model must evolve with time and type of logs it processes.

**3.2 NON FUNCTIONAL REQUIREMENTS**

**3.2.1 USER INTERFACE:**

The user requires a readable interface where the input can be fed into the application and the generated output can be viewed, interpreted visually and saved for later.

**3.2.2 HARDWARE:**

The model requires Graphical Processing Units for both its computation and for visually representing the analysis to the user.

**3.2.3 SOFTWARE:**

* Operating System: Windows.
* Programming Language : Python
* Libraries: Pandas, numpy, scikit learn, Tensorflow.
* Dataset: LogHub.
* Tools: Anaconda, Spyder, Jupyter.

**3.2.4 PERFORMANCE:**

The system’s performance is stable, optimized for dynamic scaling and consistent but requires Graphical Processing Units.

**3.3 CONSTRAINTS**

* The system categorizes only the templated and grouped logs. It cannot detect and categorize anomalies with Raw Logs.
* The detected anomaly for now is based on a log statement that has already happened. Real time implementation is to be explored.
* An anomaly might have been flagged dangerous and would have not caused a fatal damage to the system. So the system only predicts a possible anomaly with a severity of damage tagged to that anomaly.

**3.4 ASSUMPTIONS**

* The logs of a particular group pertain to one system/software only.
* The number of templates generated is of a minimal number and templating is done based on a parent format string. All logs pertaining to that format fall under that template.
* The categorization mechanism categorises the detected anomaly based on possible intensity of damage.

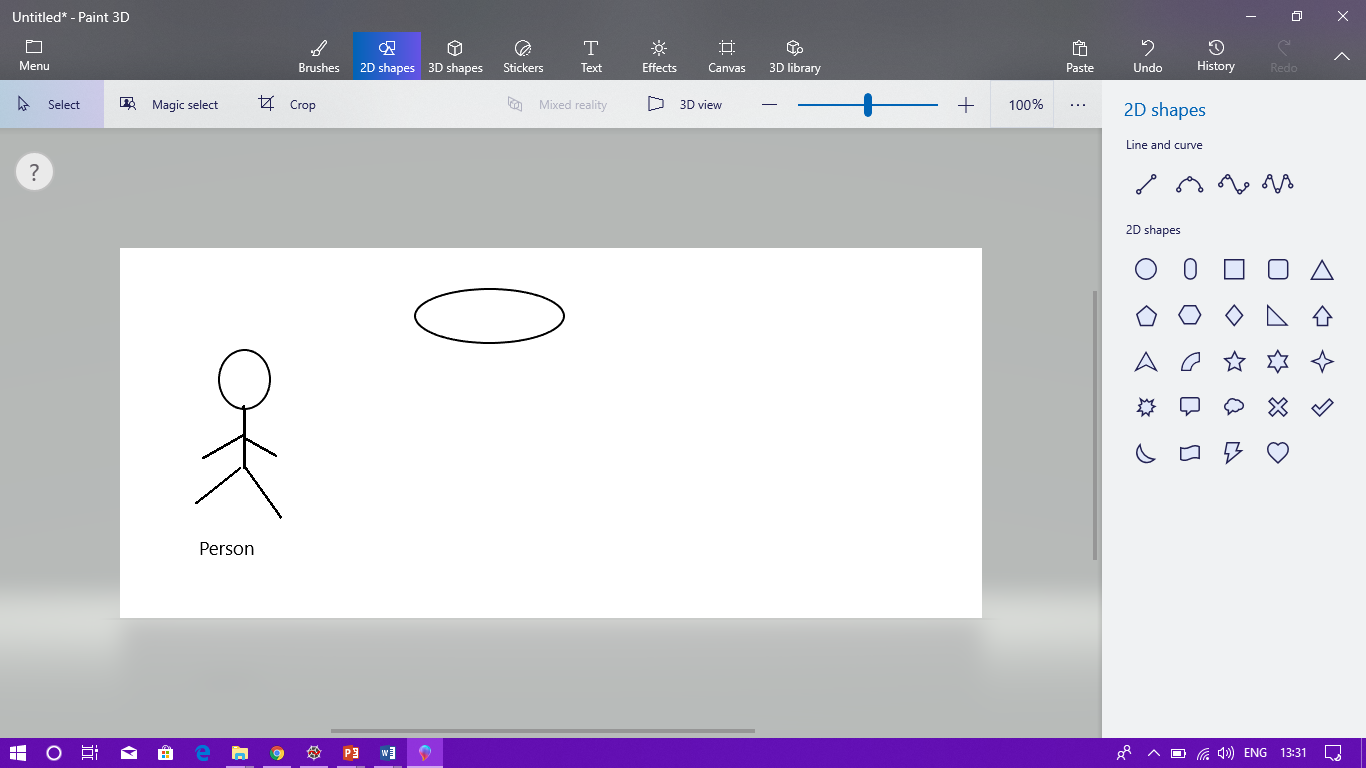
**3.5 SYSTEM MODELS**

**3.5.1 USE CASE MODEL:**

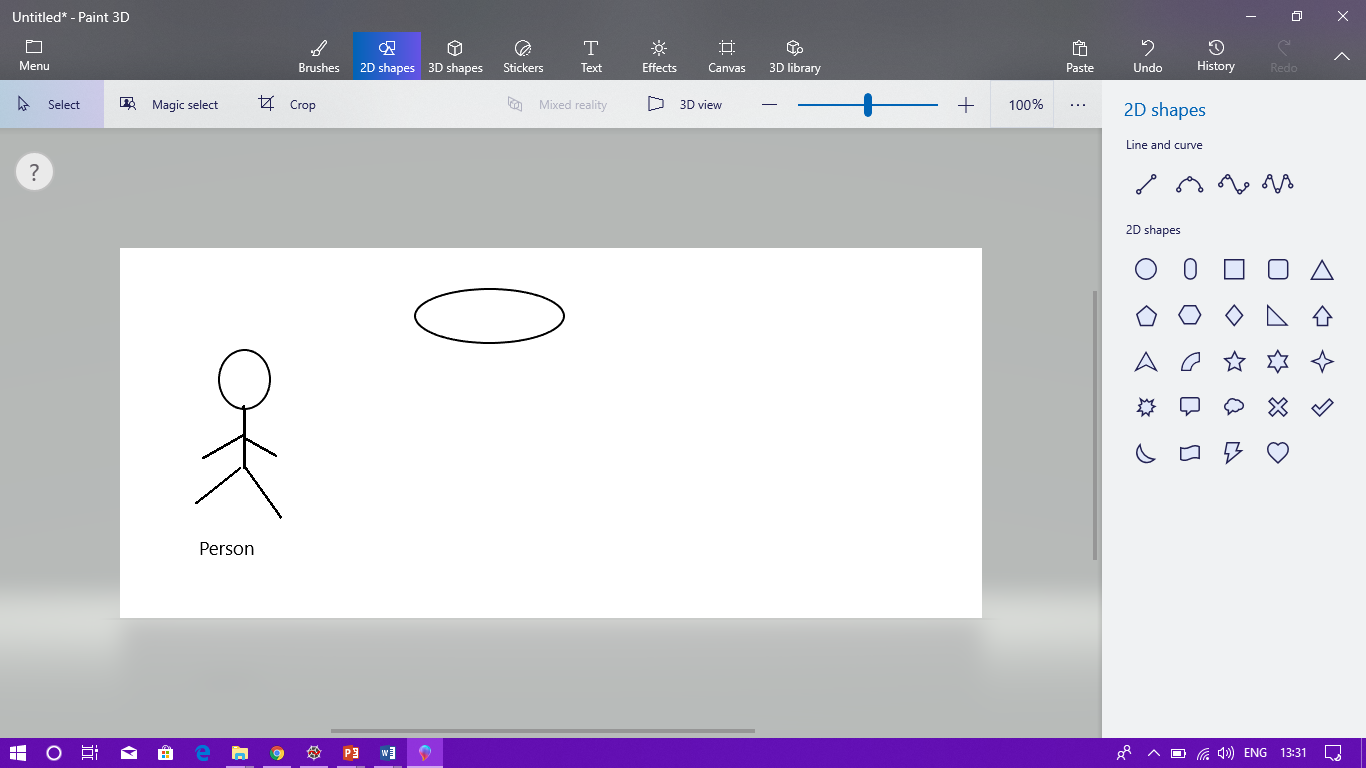
The overall Use case model for the whole system is depicted in the figure below.

*PRECONDITION*: The Large set of Logs from LogHub.

*POSTCONDITION*: The detected anomalies in their respective categories.



**Person**



**DL Model**

Fig 3.1 Use Case Model for the Overall System.

**3.5.2 SYSTEM SEQUENCE DIAGRAM**

The system sequence diagram for the application is as below. User, Loghub Dataset, the pre-processing phase, LSTM Model form the key components of the system as shown in Figure 3.2.

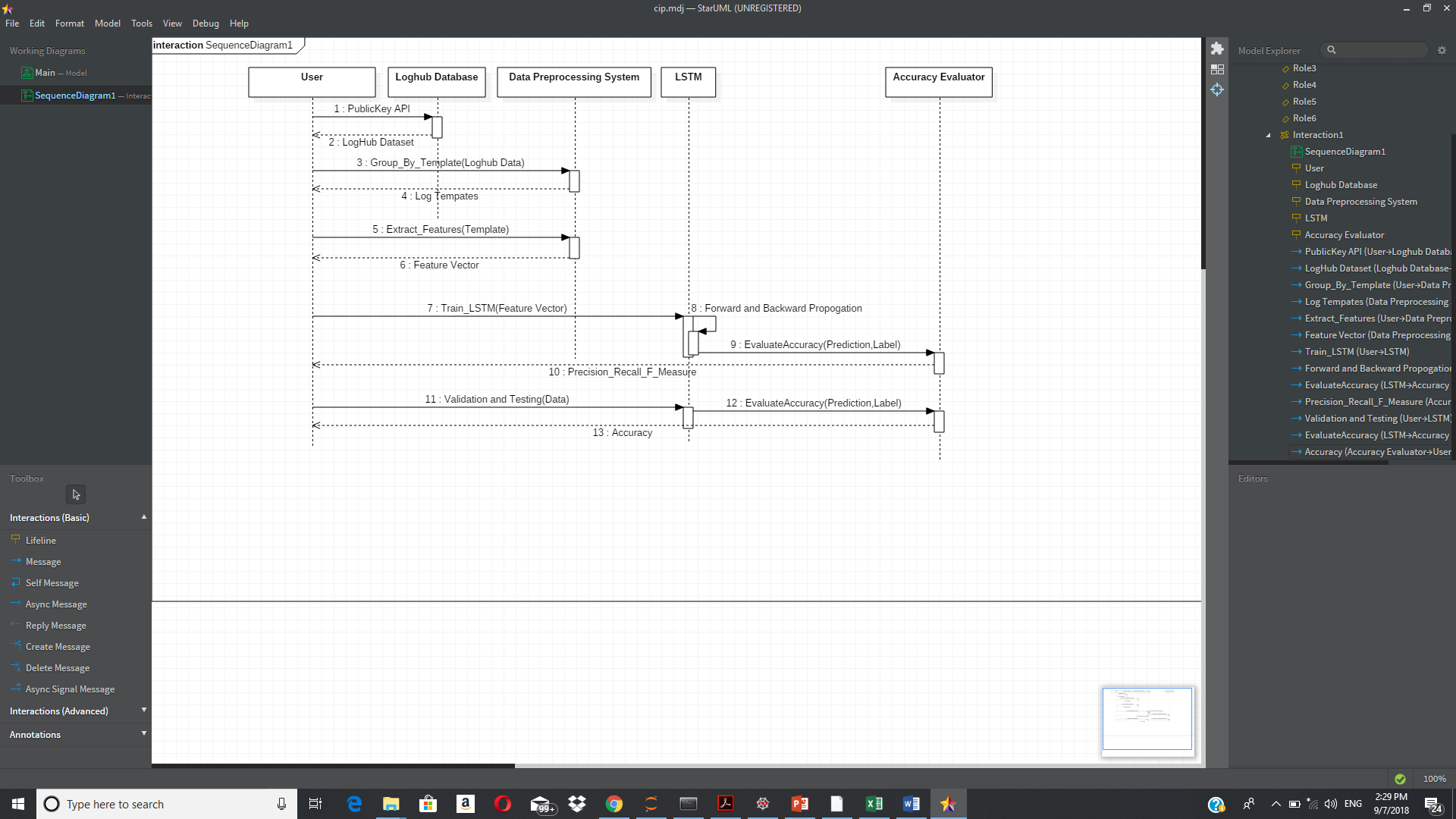


Fig 3.2 System Sequence Diagram for the OverallSystem.

**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 SYSTEM DESIGN**

The block diagram for the entire system is given in Figure 4.1.

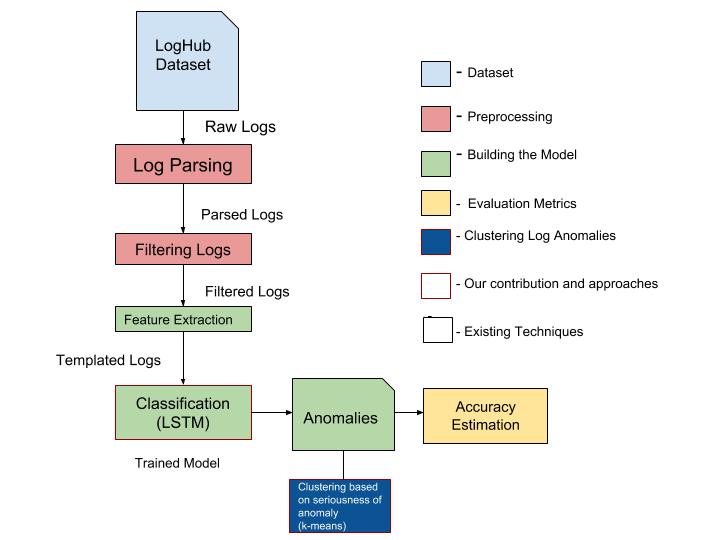


Fig 4.1 – System Architecture Diagram.

As per Figure 4.1, the input for the system comes from the LOGHUB dataset. Logs from the dataset are given as Input to the Log Parser which aims to remove all the unnecessary and system dependant information to produce skeletal logs. These skeletal logs are given as input to the Filter, which aims to categorize similar logs and create templates. The templates are subject to feature extraction which generates the feature vector and are in turn sent to the Classifier. The Classifier is based on Long Short Term Memory Model. The Long Short Term Memory classifies various logs and predicts possible anomalies. The predicted anomalies are subject to a K-Means Clustering Algorithm, which classifies the predicted anomalies into one of 3 divisions (Low, Moderate, and High) depending on the level of impact they can have on the system. Finally, the performance of the system is evaluated using Precision Measure and F-Recall.

**4.2 UI DESIGN**

The Anaconda Integrated Development Environment provides the necessary User Interface to develop, test and integrate the various modules of the Application. The working area of the IDE is divided into two parts: Workspace and Console. The workspace consists of the text editor which maintains the working code with permissions to read, write and update. The console is used for visualising data processed by the system at various checkpoints before producing the output. Jupyter Notebook provides a means by which the Output of the system and several other computations can be visualized graphically in real time.

**4.3 CLASS DIAGRAM**

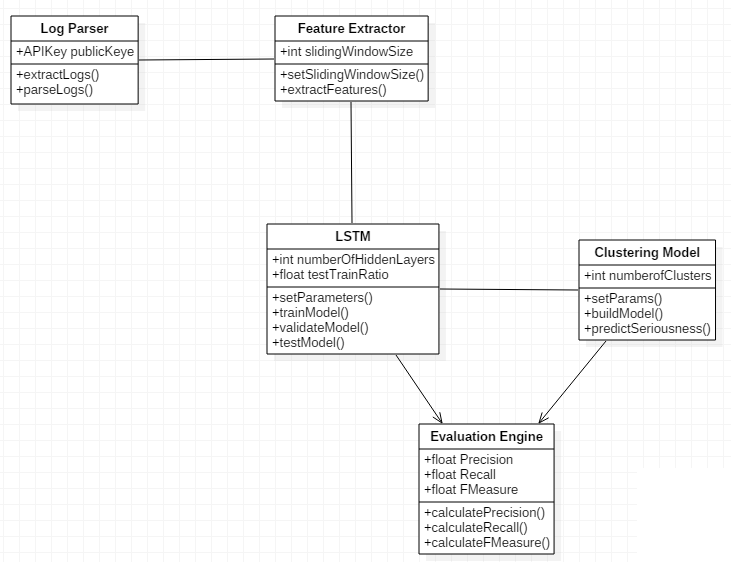


Fig 4.2 Class Diagram for the Overall System

The class diagram for the entire system is described in the diagram in Fig 4.2. The class diagram depicts the various modules present in the system and the interaction patterns they exhibit among themselves.

**4.4 MODULE DESIGN**

**4.4.1 LOGHUB DATASET**

The Loghub dataset can be retrieved using the Public API and a Unique Key. It is of approximately 88GB in size and consists of logs from various Operating Systems and Software Solutions. The datasets that are being used for training and testing are the Machine Temperature Dataset and Power Demand Dataset which are a part of the official Loghub Dataset.

**4.4.2 LOG PARSING AND FILTERING**

The primary function of the Log Parser is to accept the input of raw logs and extract templates that are invariant of variable such as memory address, Error codes and other system or software dependant information. For every such unique template, a new file is created where every log adhering to the parent format is stored.

When a new and unique log statement is encountered, a generalized regular expression is created which would serve as the point of classification for other log statements of the same type. Every regular expression generated is saved and tested on every log until it perfectly matches. If matched, the log is saved as part of the template. As a pilot study, we limit the number of such templates to fourteen. The generated templates are sent to the feature extraction phase.

**4.4.3 FEATURE EXTRACTION**

The template logs are sent to feature extraction phase. All Log templates are divided into log sequences or windows. Sliding windows are formed for every log sequence which are then aggregated to form an event count matrix. The generated Event Count Matrices for each template are consolidated to produce a unique value which serves as an input to the deep learning LSTM Model.

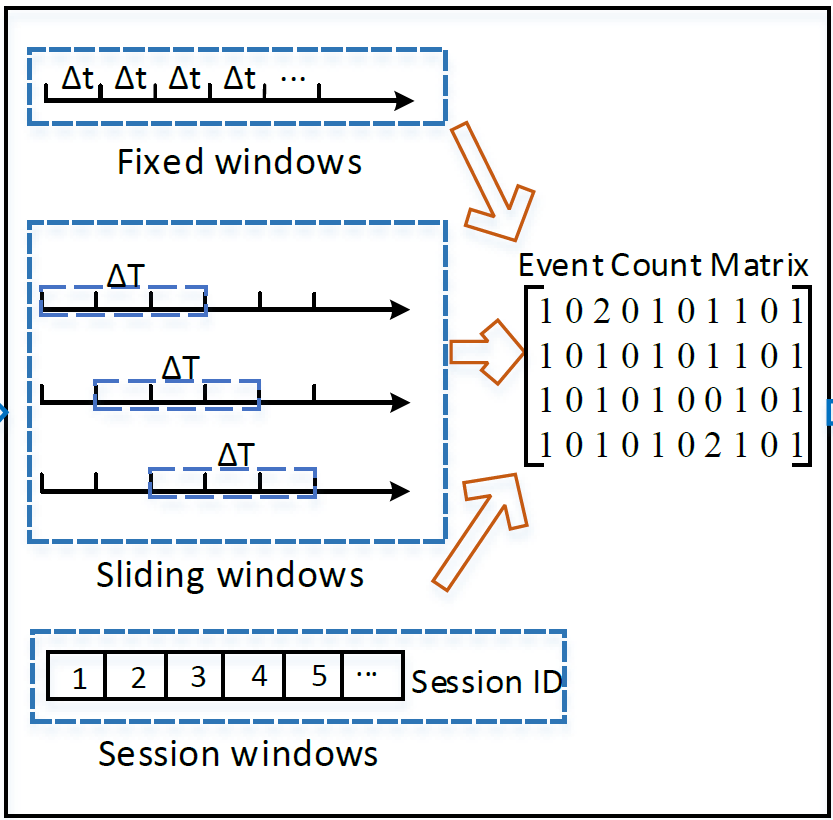


Fig 4.3 Sliding window used in Feature Extraction.

**4.4.4 CLASSIFICATION (LSTM)**

The LSTM RNN is trained only on normal data to learn normal time series patterns and optimized for prediction accuracy. For this purpose, each dataset is divided into four subsets: a training set, N, with only normal values; validation set, VN, with only normal values; a second validation set, VA, with normal values and anomalies; a dates set, T, having both normal values and anomalies. The working is as follows:

*1. Set N is used for training the prediction model. We used Bayesian optimization to ﬁnd the best values for hyper-parameters: look back, dropout, learning rate, and the network architecture (number of hidden layers and units in each layer). We use a look ahead of more than 1 only if the prediction accuracy is still reasonable. If predicting multiple time steps is not required and one needs the best prediction accuracy, look ahead can be set to 1.*

*2. VN is used for early stopping to prevent the model from overﬁtting the training data.*

*3. Prediction errors on N are modelled using Gaussian distribution. The mean and variance of the distribution are estimated using MLE.*

*4. The trained prediction model is applied on VA. The distribution parameters calculated in the previous step are used to compute the log PDs of the errors from VA. A threshold is set on the log PD values which can separate the anomalies, with as few false alarms as possible.*

*5. The set threshold is evaluated using the prediction errors from the test set T.*

We used Python programming language and Keras to code and implement the experiments for the project. Keras is an open source project which provides a highlevel API to implement DNNs and runs on top of other deep learning libraries like TensorFlow and Theano.

**4.4.5 K-MEANS CLUSTERING FOR CATEGORISING ANOMALIES**

The LSTM Model detects the possible anomalies that may arise from the voluminous amount of Logs that is provided as input to the system. Once the anomalies have been detected the system aims to categorise them into one of 3 possible categories depending on the level of impact they can have on the smooth functioning of the system. The three categories that are defined as part of the categorisation algorithm are Low, Moderate and High.

The K-Means Algorithm is one of the most established and widely used categorisation algorithms available in the field of Machine Learning. Once the detected anomalies are fed into the K means algorithm as input, the algorithm focuses on the median value of the weights associated with every detected anomaly and forms 3 clusters based on the range of weights provided.

**4.4.6 ACCURACY ESTIMATION**

The system uses Precision, Recall and F-Measure to denote the accuracy of the LSTM model in detecting anomalies and using the K-Means Implementation to categorize the detected anomalies based on level of impact.

The formulae used for estimating accuracy based on the above methods are as follows:

* Precision = # of Anomalies detected/# of Anomalies reported.
* Recall = # of Anomalies detected / # of All anomalies.
* F − measure (Harmonic Measure) =2 × Precision × Recall Precision + Recall.

**4.5 COMPLEXITY ANALYSIS**

**4.5.1 TIME COMPLEXITY**

The time complexity concerned with various phases of the system are mentioned in table 4.1

|  |  |  |
| --- | --- | --- |
| S.No. | Name of the Module | Complexity (Per Log) |
| 1. | Log Parsing and Filtering | O(n) |
| 2. | Feature Extraction | O(n) |
| 3. | Classification using LSTM | O(n) |
| 4. | K means Categorization | O(k\*n) |

Table 4.1 Time Complexity of Various Modules.

Where

k = Number of Clusters (3 as per the system).

n = Number of Words in the Input Log Statement.

**4.5.2 COMPLEXITY OF THE PROJECT**

The complexity of the System lies solely on the Number of templates created for the LogHub Dataset and on the number of hidden layers that are present in the LSTM model. The preferred number of templates to be created is 14 and the most adaptable count of hidden layers that should be present for the LSTM is 120. Increase in the number hidden layers leads to more accurate results but increases the complexity of the system exponentially.

The Number of clusters to be formed using the K-Means algorithm is kept minimal at 3 which reduces the stress on the categorization procedure. The weights are compared and plotted in such a manner that all anomalies fall into one of 3 possible categories and the accuracy of the classification is as high as possible.

**CHAPTER 5**

**SYSTEM DEVELOPMENT**

The system consists of several modules with the major modules being the Log Parser, the LSTM Model and the K means algorithm bundle. The Overall code review can be seen in the following images:

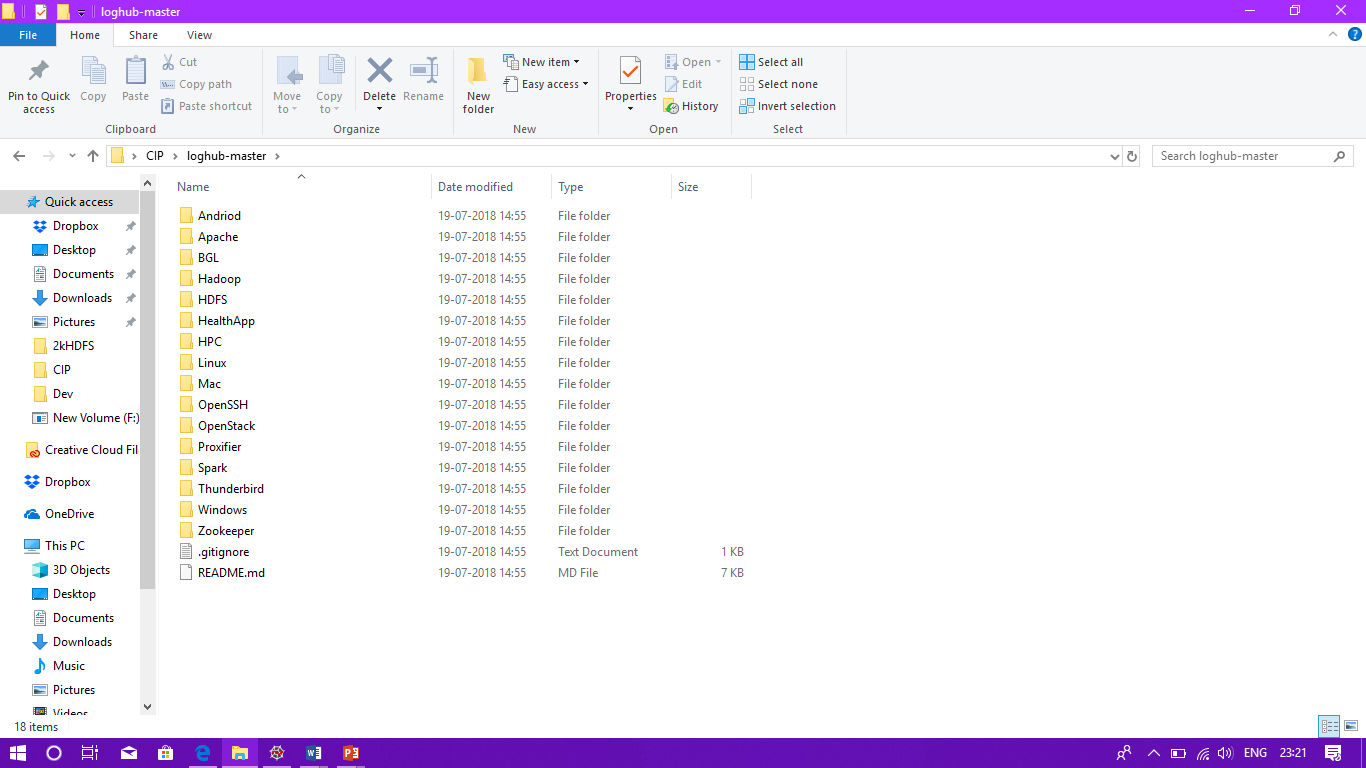


Fig 5.1 LogHub Dataset.

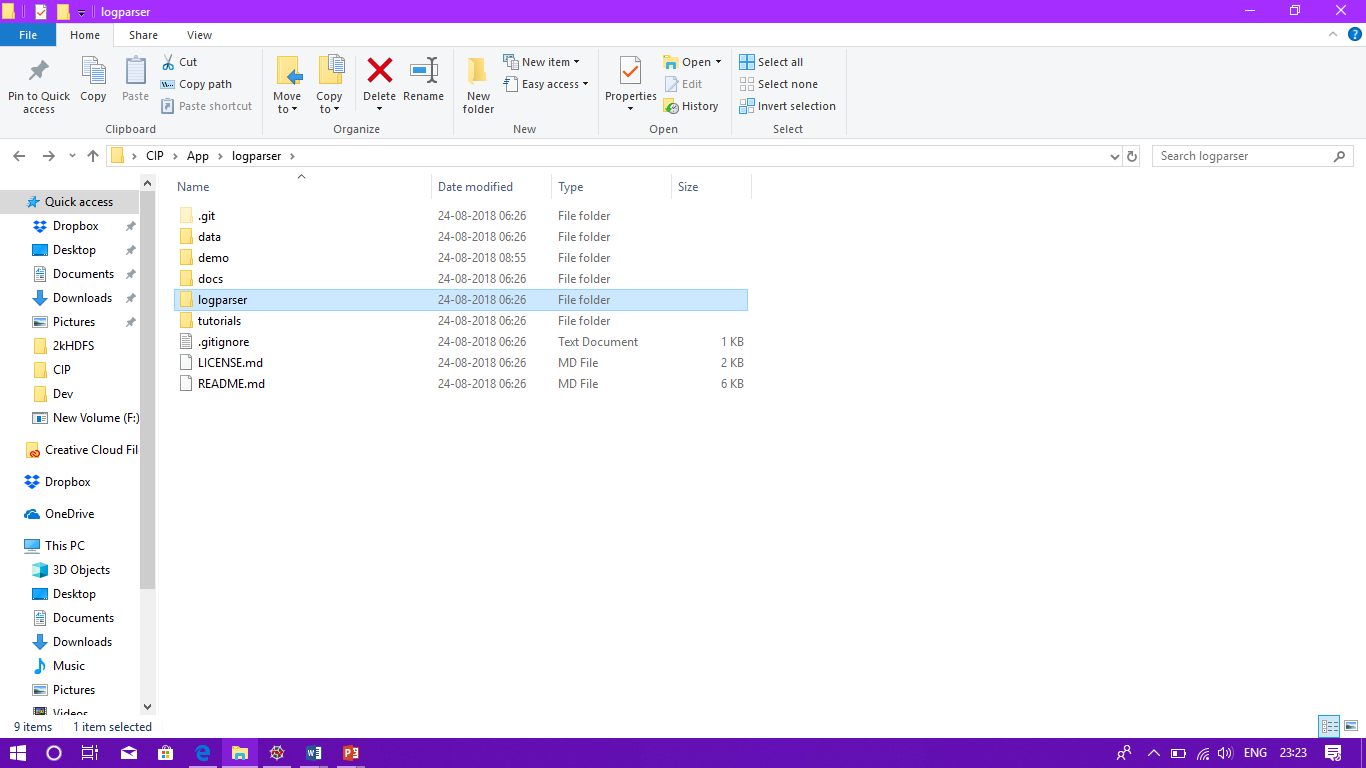


Fig 5.2 LogParser Directory Structure.

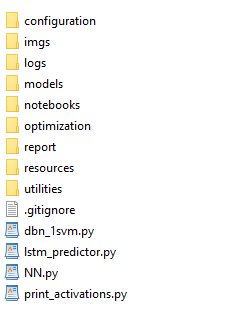


Fig 5.3 LSTM Directory Structure

An overview of the entire system is explained in the following sections. The system accepts a set of logs as input, parses them to generate feature vectors, uses the vectors in an LSTM Model which detects possible anomalies that may occur and categorises them using the K-Means algorithm depending on their individual intensity.

**5.1 PROTOTYPE ACROSS MODELS**

The Input and Output that is accepted and Generated by every module is explained in this section:

* *Log Parsing and Filtering*: The parser accepts the logs from the dataset as Input and uses Regular Expressions to categorise the logs based on templates that are generated in due course of the execution. Once all the logs have been classified as per their parent format, they are sent to the feature extraction phase.
* *Feature Extraction*: This phase accepts Templates of Logs as input and generates a feature vector for every log based on the sliding window mechanism. The generated feature vectors are sent to the LSTM Model for Anomaly Detection.
* *Classification using LSTM*: The model accepts the feature vectors as input and using RNN and Keras from Tensorflow, it aims to detect possible anomalies that might occur among the set of logs. Once detected, the anomalies are sent to cluster phase for grouping.
* *Clustering the Anomalies*: The detected anomalies are sent to the K-Means Clustering algorithm, which classifies the anomaly into one of 3 possible types depending on the level of impact it can produce on the system. The three clusters used are Low, Medium and High.

**5.2 LONG SHORT TERM MEMORY ALGORITHM**

The algorithm proceeds as follows:

1. Set N is used for training the prediction model. We used Bayesian optimization to ﬁnd the best values for hyper-parameters: look back, dropout, learning rate, and the network architecture (number of hidden layers and units in each layer).
2. We use a look ahead of more than 1 only if the prediction accuracy is still reasonable.
3. If predicting multiple time steps is not required and one needs the best prediction accuracy, look ahead can be set to 1.
4. VN is used for early stopping to prevent the model from overﬁtting the training data.
5. Prediction errors on N are modelled using Gaussian distribution. The mean and variance of the distribution are estimated using MLE.
6. The trained prediction model is applied on VA. The distribution parameters calculated in the previous step are used to compute the log PDs of the errors from VA. A threshold is set on the log PD values which can separate the anomalies, with as few false alarms as possible.
7. The set threshold is evaluated using the prediction errors from the test set T.

**5.3 K-MEANS CLUSTERING ALGORITHM**

The K-Means Clustering Algorithm is given below. The output of the clustering phase is 3 clusters (Low, Medium, High) with anomalies present in them.

1. Clusters the data into *k* groups where *k*  is predefined.(Here k=3)
2. Select *k* points at random as cluster centers.
3. Assign objects to their closest cluster center according to the *Euclidean distance* function.
4. Calculate the centroid or mean of all objects in each cluster.
5. Repeat steps 2, 3 and 4 until the same points are assigned to each cluster in consecutive rounds.

**5.4 DEPLOYMENT DETAILS**

The entire system is deployed on Anaconda’s Development Interface and the visual interpretation of the data generated by the system is hosted on Jupyter Notebooks. Keras from Tensorflow and other python libraries are natively available in the host system.

**CHAPTER 6**

**RESULTS AND DISCUSSION**

**6.1 DATASET FOR TESTING**

We use the datasets related to Machine Temperature, Power Demand and ECG. Each dataset was tested individually and these datasets were chosen because they are prone to producing better anomalies for the system to categorize. The overall process of detecting and categorizing the anomalies from the testing datasets is summarized below:

**6.2 OUTPUT AT VARIOUS STAGES**

The Output obtained at various stages of the system is listed below:

**6.2.1 INPUT STATEMENTS**

The input logs given to the system is shown in Figure 6.1

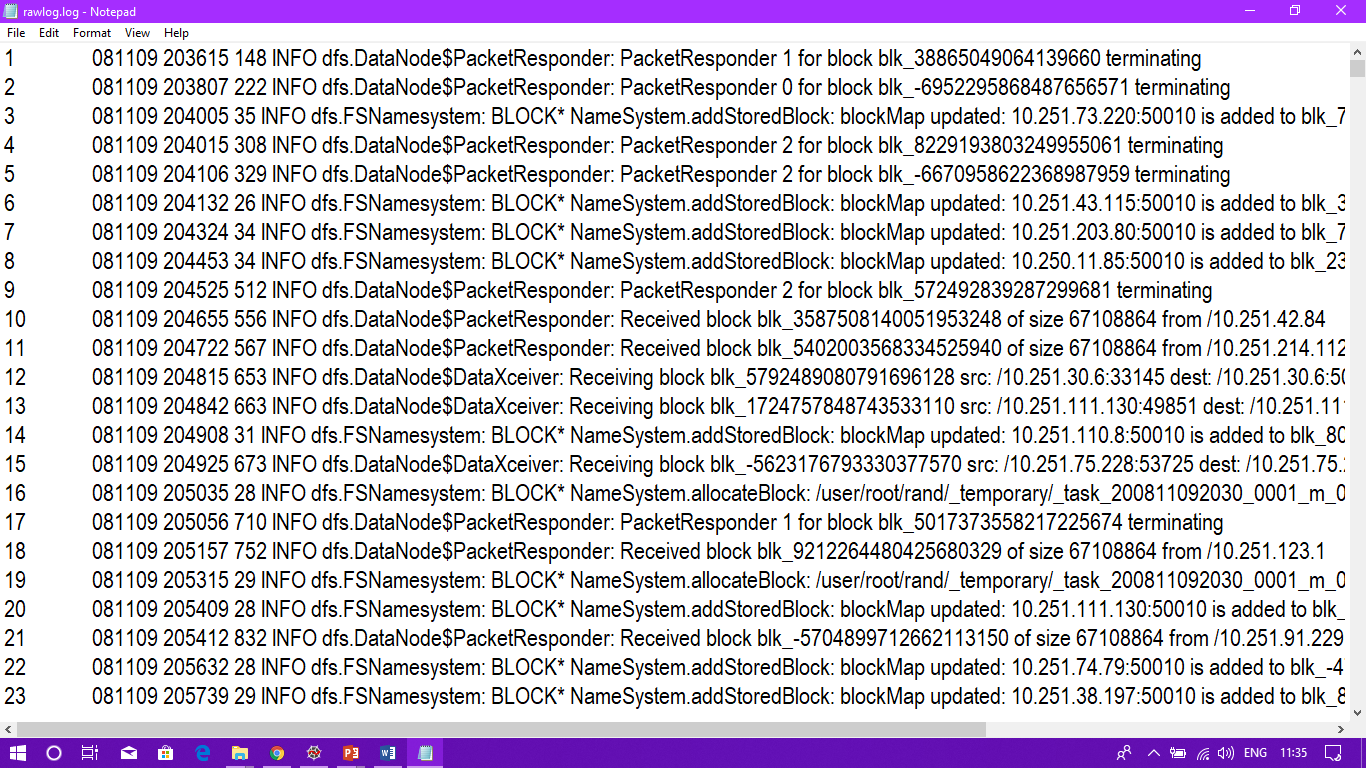


Fig 6.1 Input Logs

**6.2.2 LOG PARSER AND FILTERING**

The Output from the Log Parser and Filtering stages are given in Figure 6.2.

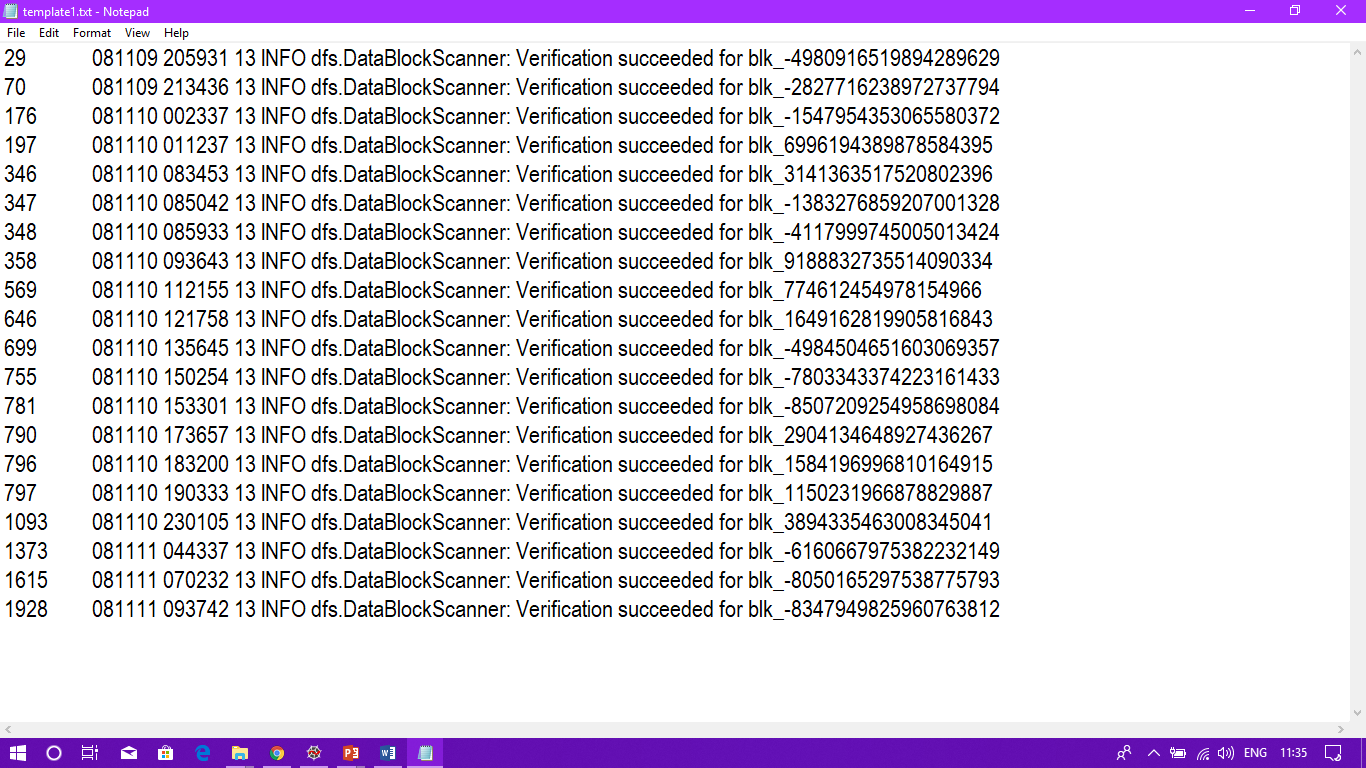


Fig 6.2 Templated Logs.

**6.2.3 FEATURE EXTRACTION**

The feature extraction phase produces a weight-based feature vector for a given log statement. This feature vector is sent to the LSTM for detection.

**6.2.4 LSTM DETECTION**

The feature vectors are sent to the LSTM for Anomaly detection. Figure 6.3 illustrates how every log statement is assigned a weight and classified as to whether it is an anomaly (A) or not an Anomaly (NA).

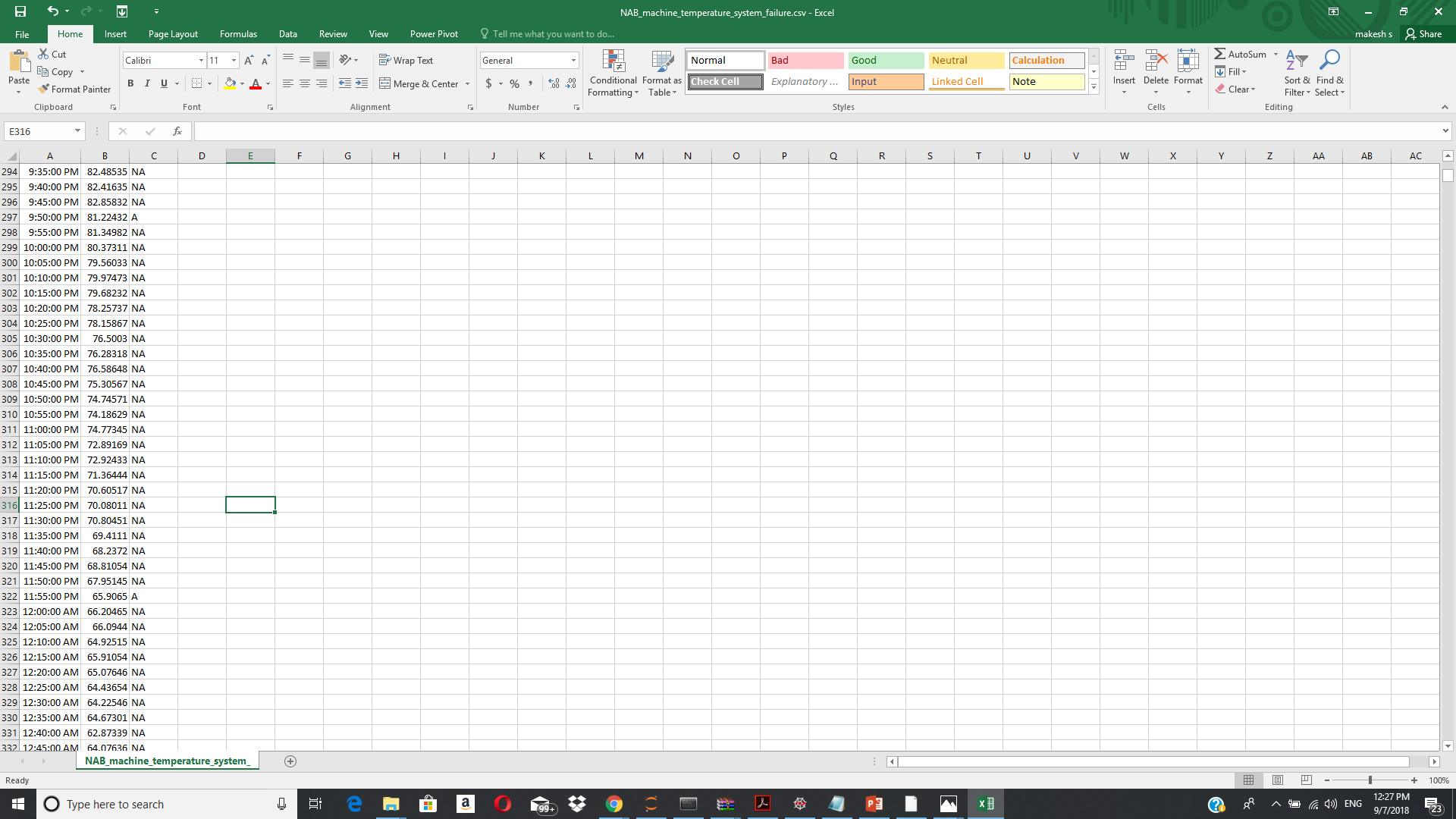
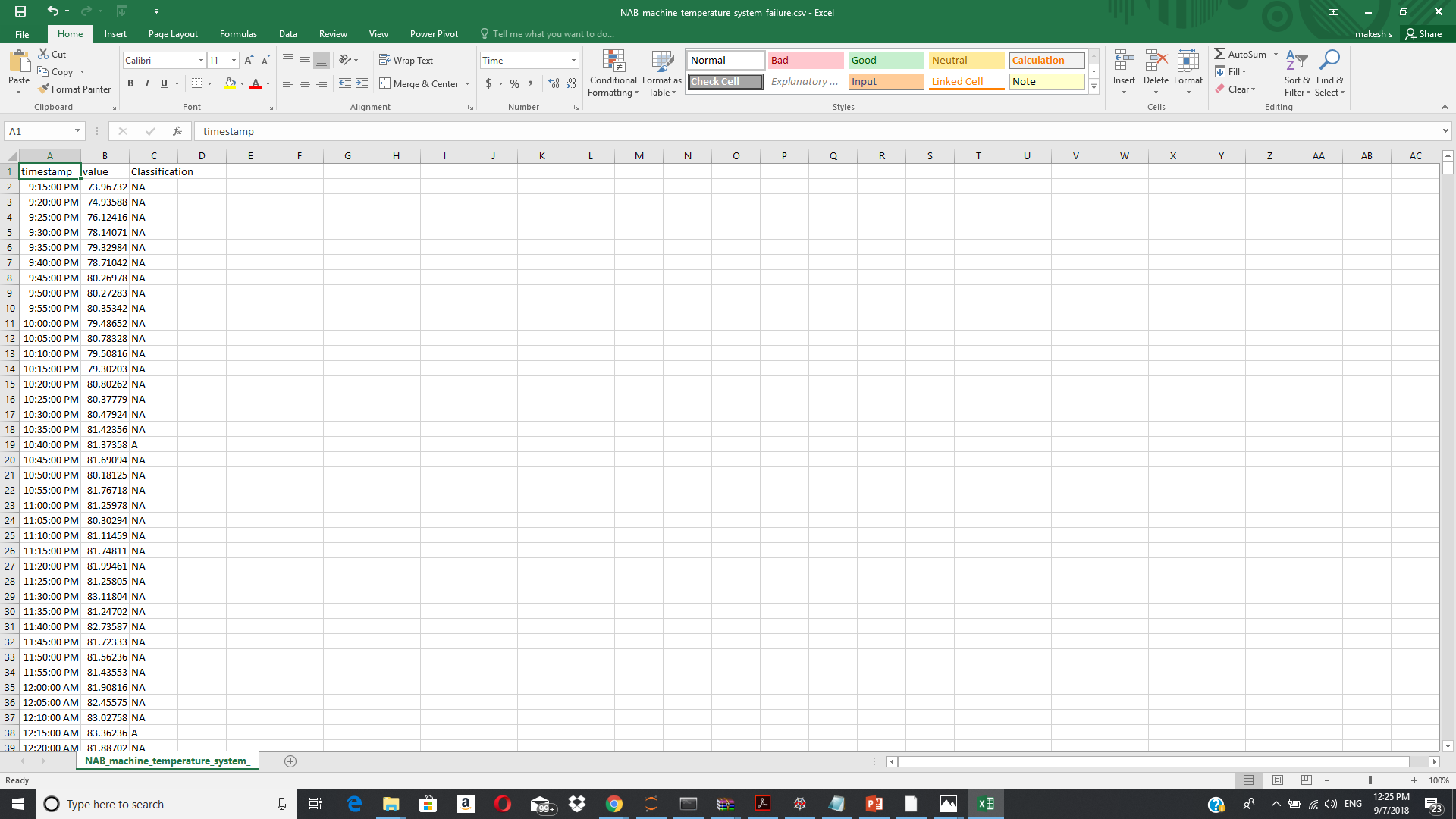


Fig 6.3 Logs with Anomaly information.

**6.2.5 CLUSTERING ANOMALIES**

The detected anomalies are clustered into one of three groups based on their intensity. As shown in Figure 6.4, 1 indicates Low priority, 2 indicates medium priority and 3 indicates high priority.

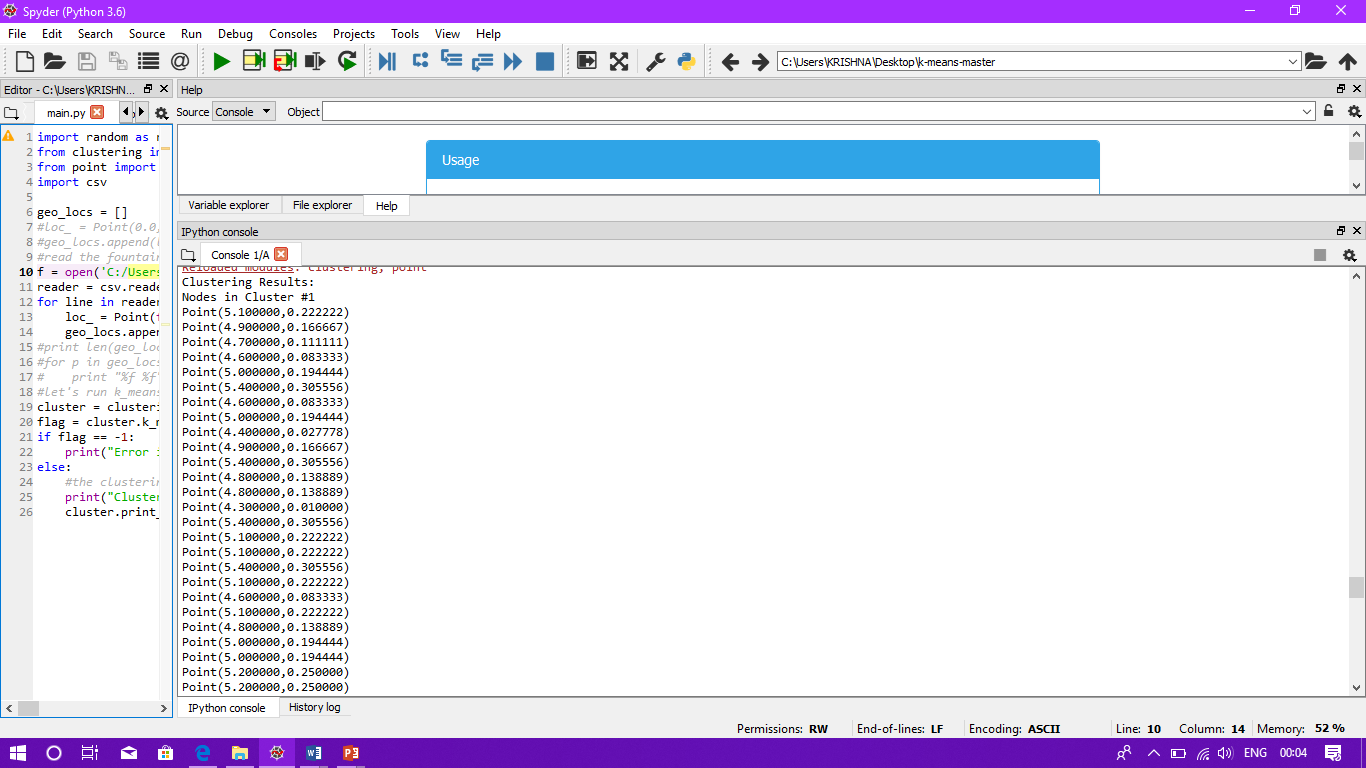
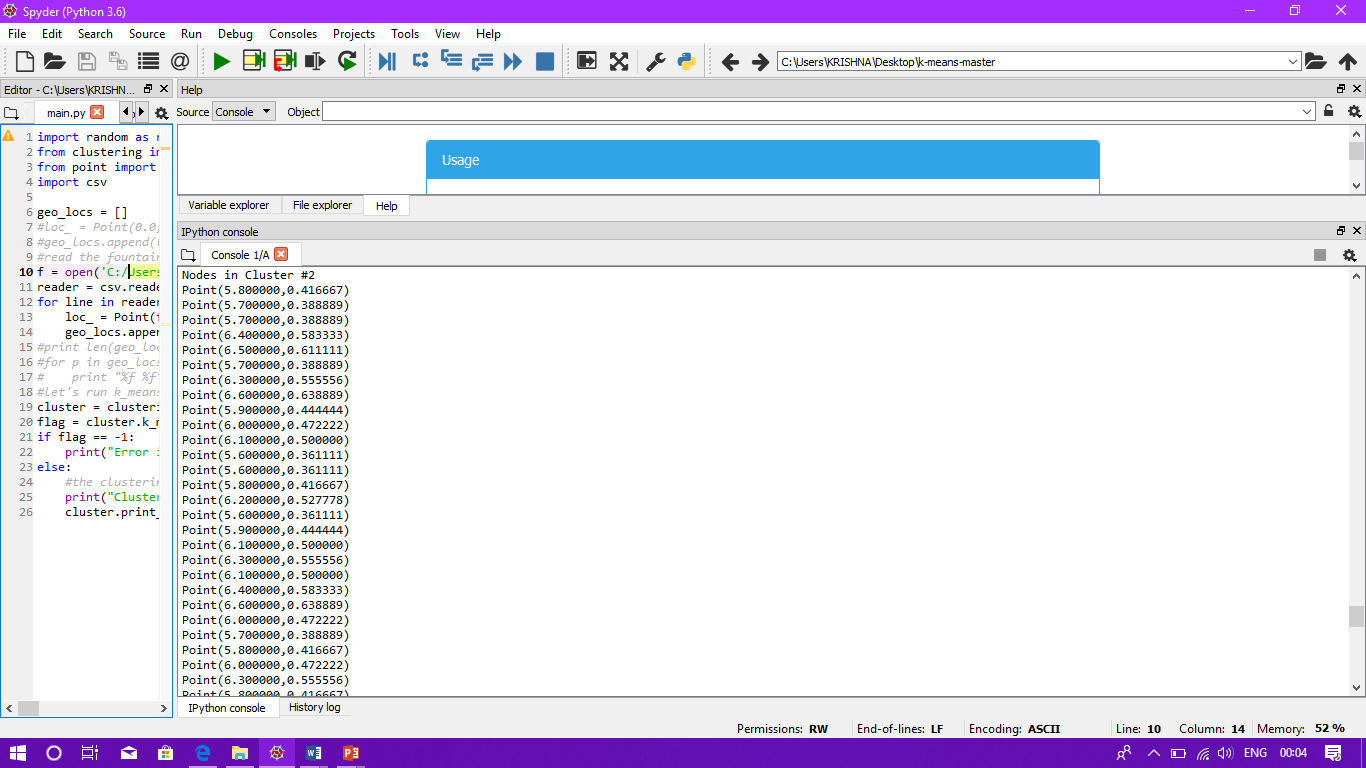
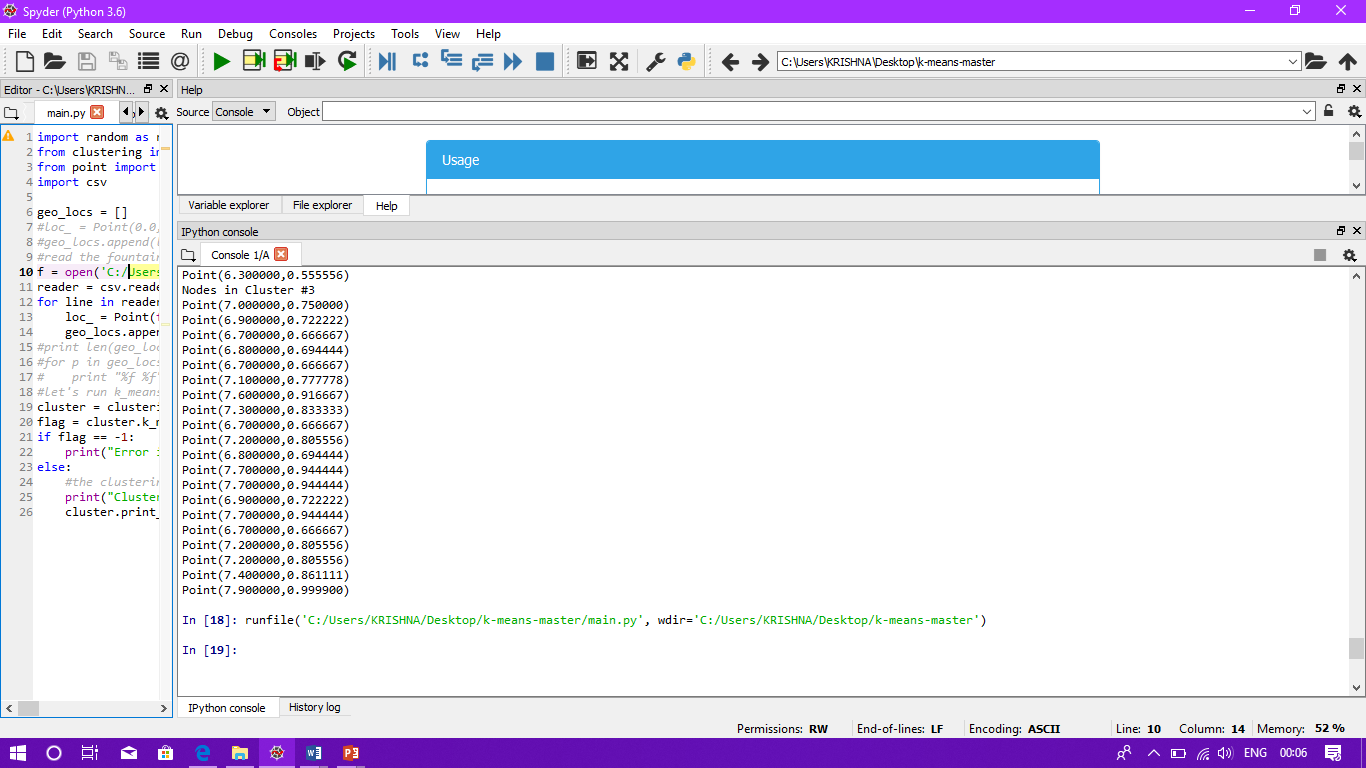


Fig 6.4 Anomalies in all 3 clusters.

**6.3 SAMPLE SCREENSHOTS FROM TESTING**

The following figures are excerpts from the various input and output instances that were used as part of the system. Jupyter Notebooks was used for demonstrating the various Output instances visually.

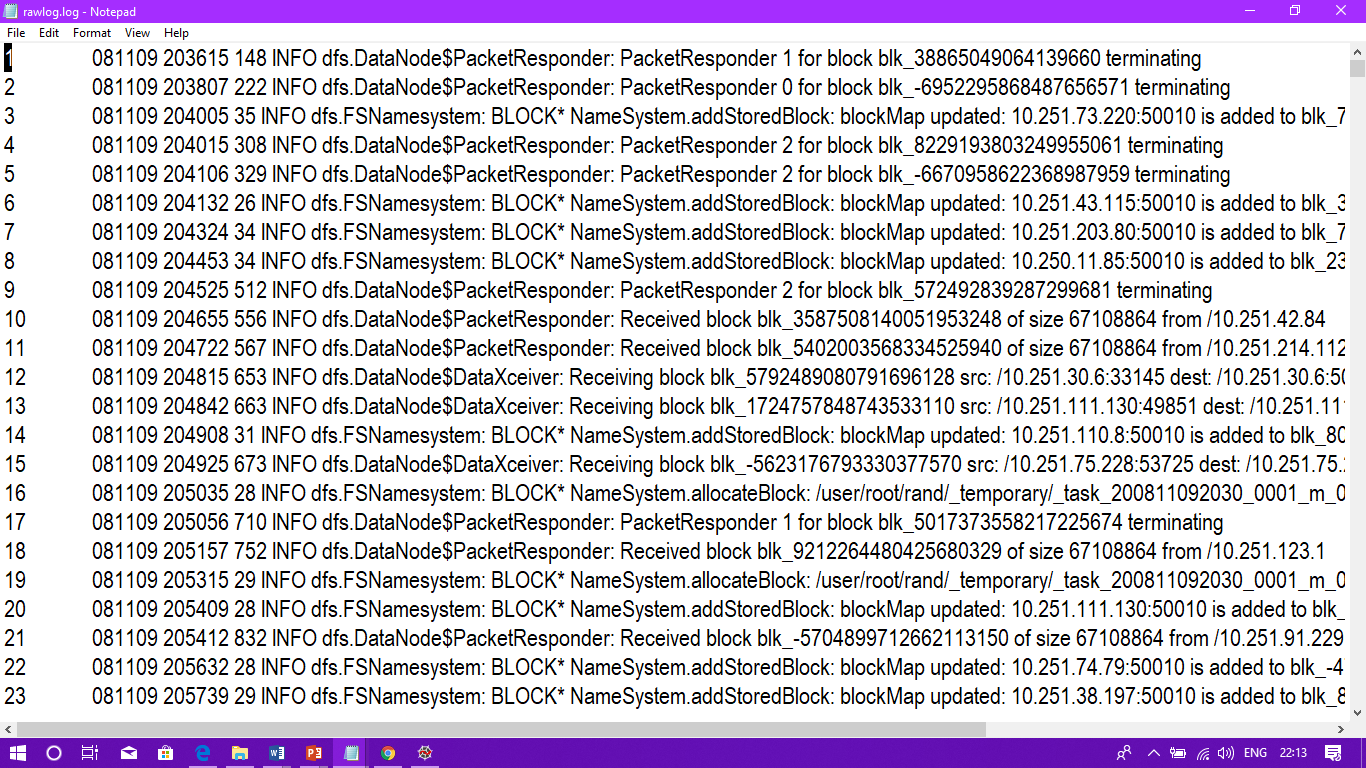
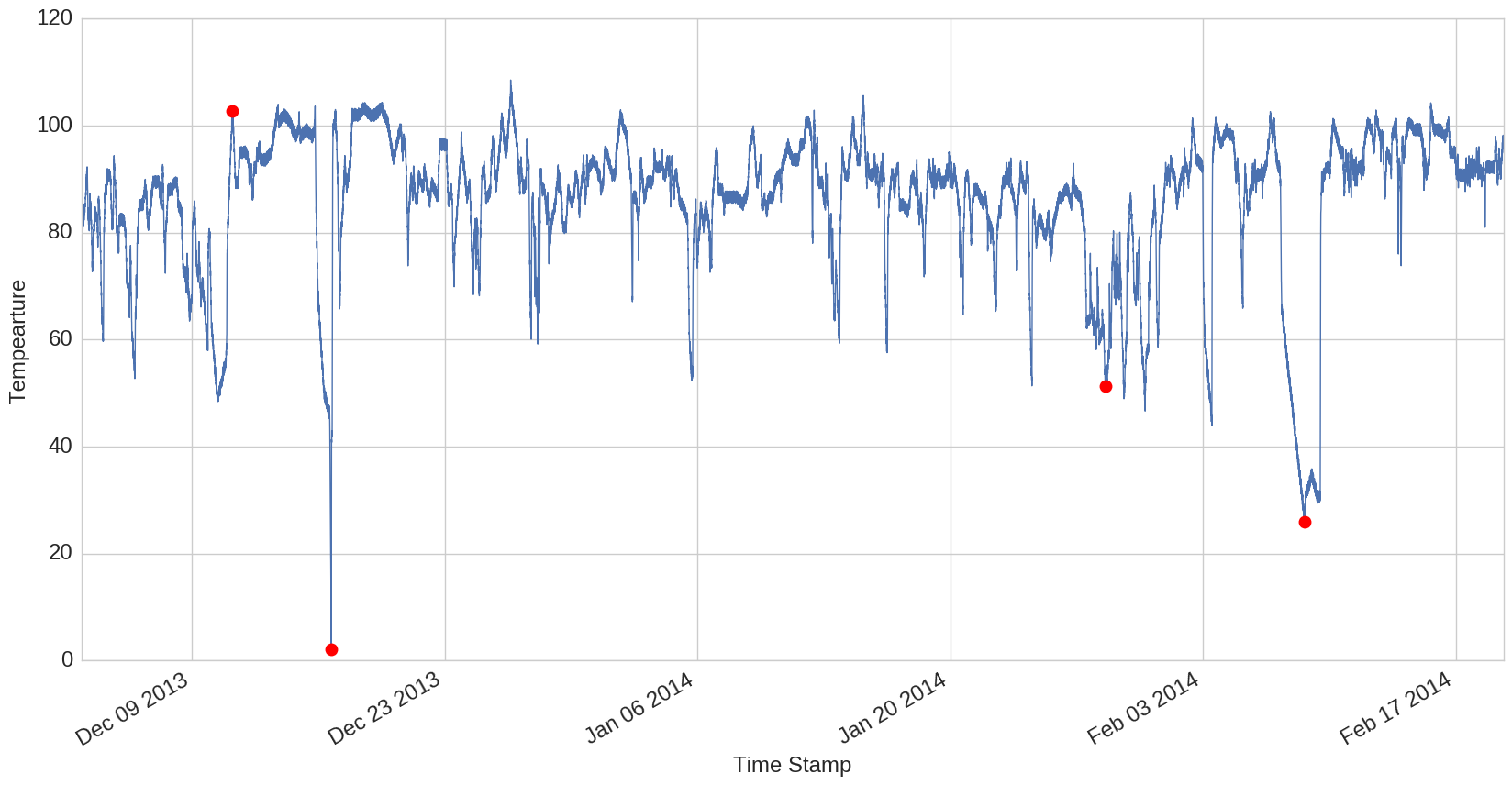
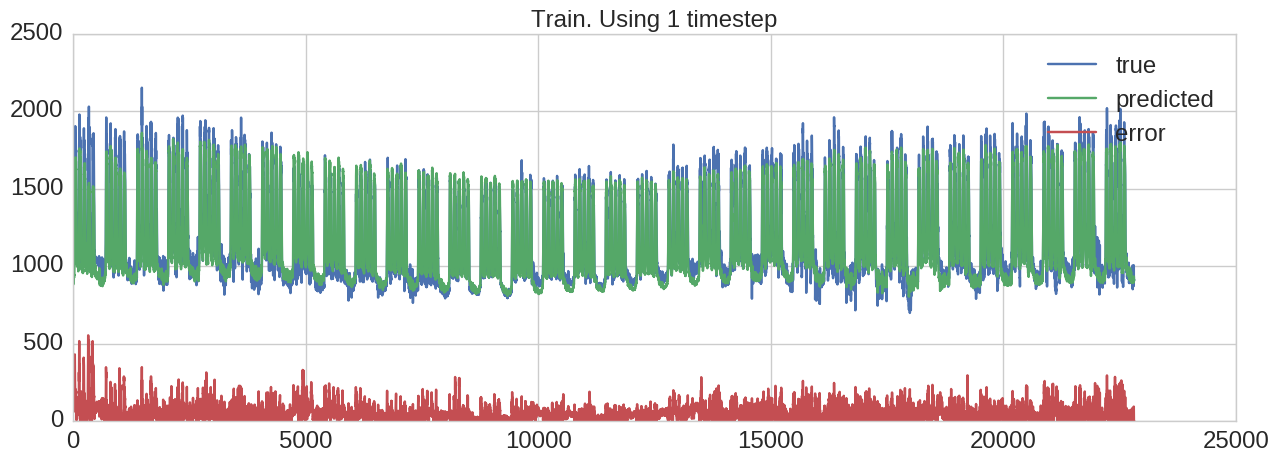


Fig 6.5 Input Log Statements



Fig 6.6 Weight Graph Indicating Detection of anomalies.

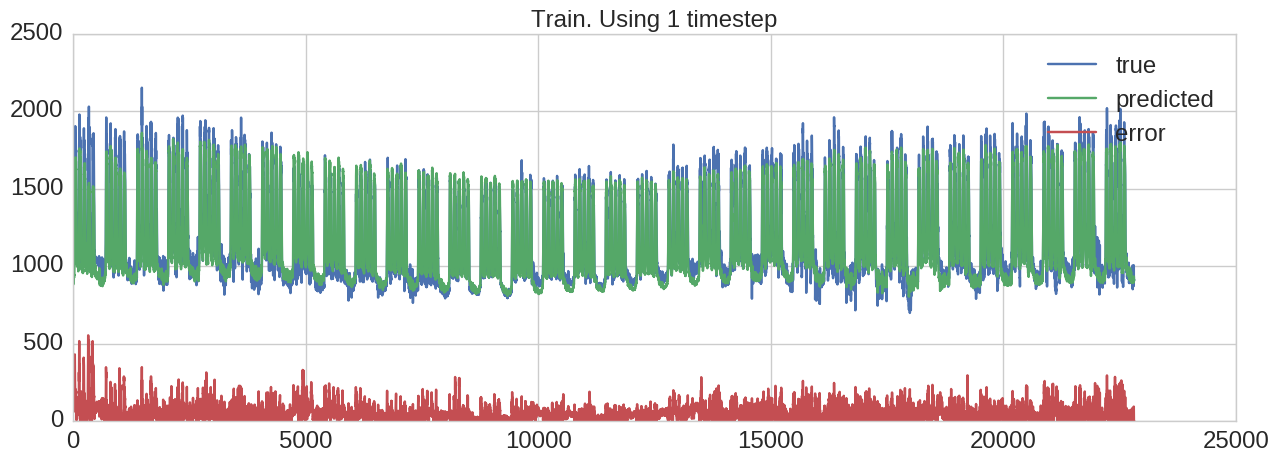


Fig 6.7 Graph indicating the prediction of anomalies.

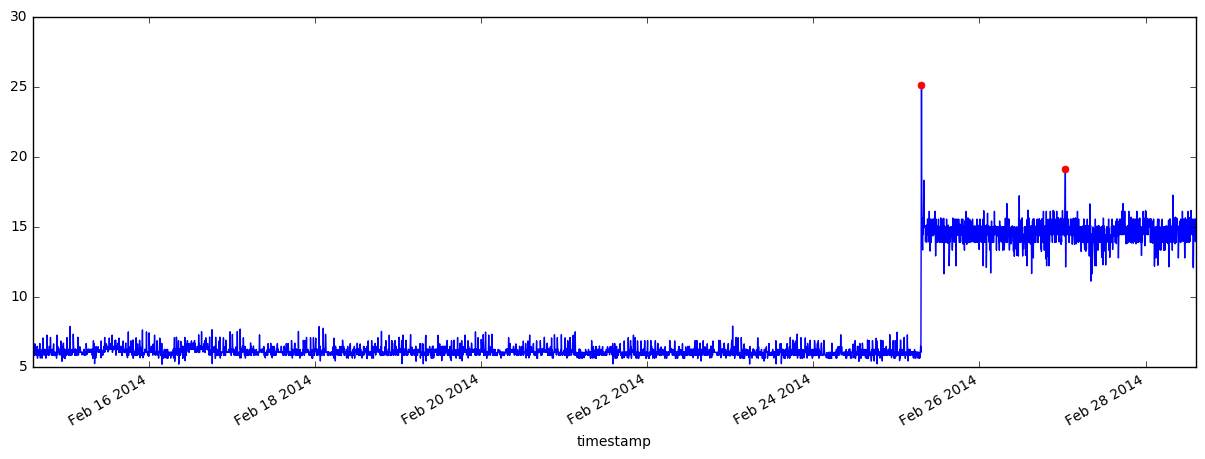


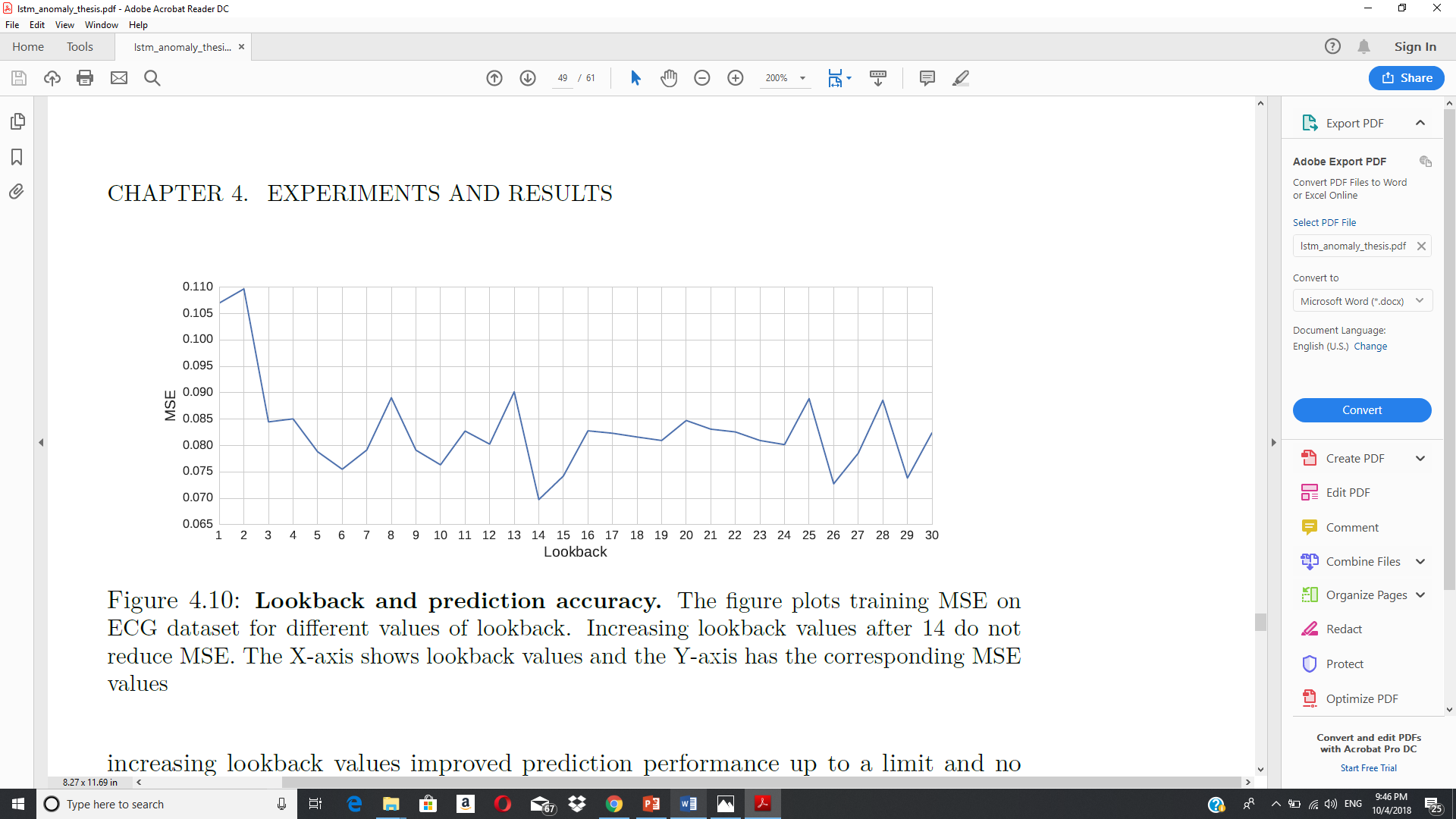
Fig 6.8 Graph indicating the categories of anomalies.

PLEASE ADD KMEAN HERE AND CHANGE NAME OF GRAPH 6.8 IT IS NOT CATEGORIES.

6.4.2 Long Short Term Memory Model

***Lookback Parameter:***

The parameter lookback is the number of time steps for which the RNN is unfolded for back-propagation. There is no recommended value for which the RNN should be unfolded, and the optimal value depends on the nature of data and the temporal correlations present in it. Lookback values ranging from 8 to 200 have been used successfully for different tasks. While lookback value is an important parameter that influences learning, LSTM RNNs can still learn patterns longer than lookback length as the LSTM state can store information from any previous time step.



We experienced a trade-off between prediction performance and the results of anomaly detection. An RNN optimized for minimizing the prediction MSE was not the best for anomaly detection.

One of the assumptions we made was the normal distribution of residuals or prediction errors. We conducted Shapiro-Wilk test for normality on the residuals of each dataset. Based on the outcome of the tests we rejected the hypothesis that

the prediction errors were normally distributed.

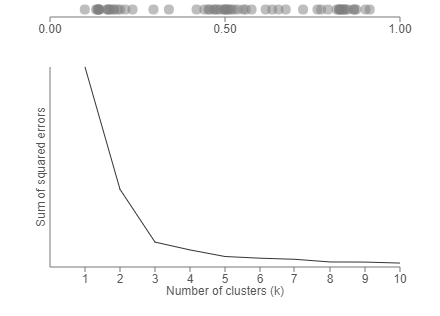
|  |  |  |
| --- | --- | --- |
|  | Train Dataset | Test Dataset |
| F-Measure | 0.973 | 0.957 |
| Precision | 0.977 | 0.978 |
| Recall | 0.971 | 0.952 |

6.4.3 K-Means Clustering for Categorization of Anomalies

Determination of a value of *k* in the K-Means algorithm is by no means a simple task. Choosing too small a value of *k* leads to excess generalization whereas adopting a very high value of *k* leads to overfitting. Thus, there exists a need to choose an appropriate value of *k* to achieve a reasonable value of **Sum of Squared Errors** (SSE) which in turn leads to better categorization of anomalies from unknown test data.

In order to find the appropriate value of *k*, we make use of the Elbow Method.  The idea of the elbow method is to run k-means clustering on the dataset for a range of values of k (say, k from 1 to 10), and for each value of k calculate the sum of squared errors (SSE).

Then, plot a line chart of the SSE for each value of k. If the line chart looks like an arm, then the "elbow" on the arm is the value of k that is the best. The idea is that we want a small SSE, but that the SSE tends to decrease toward 0 as we increase k (the SSE is 0 when k is equal to the number of data points in the dataset, because then each data point is its own cluster, and there is no error between it and the center of its cluster). So our goal is to choose a small value of k that still has a low SSE, and the elbow usually represents where we start to have diminishing returns by increasing k.



Thus, as the graph tapers around 3, we choose three as the appropriate number of clusters.

**CHAPTER 7**

**CONCLUSIONS**

**7.1 SUMMARY**

This application is Anomaly Detection and Categorization system that uses an LSTM model to detect anomalies and classifies the detected anomalies based on their intensity into three broad categories using the K-means clustering algorithm. The Log Parser helps in removing system centric information whilst preserving the important details of every log statement. Once the logs are grouped based on their structural similarities, the LSTM model takes every log template as input and produces a group of log statements that it detects to be an anomaly. Once the anomalies have been detected, they are categorized into one of three categories – low, medium and high based on the level of intensity they possess. The clustering is achieved via the K-means clustering algorithm.

The log parser was designed to accept an input log file from one particular software system or tool for simplicity. The log parser generates 14 different templates by means of regular expressions as 14 has been perceived to be an optimal number for the templates – any number higher or lower has had effects on the efficiency of the model. LSTMs were studied to explain their complex architecture, their ability to learn long-range dependencies, and different ways of maintaining LSTM state. Next, we developed an anomaly detection method based on a summary prediction model. We briefly touched upon the various issues prevalent in anomaly detection research. To circumvent these problems, we selected three real-world datasets that have been used in previous research and contain different types of anomalies. Finally, we conducted experiments on these datasets. Based on our results we conclude that LSTMs are effective time series modelers and anomaly detectors. As categorization of anomalies is a first of its kind approach to this problem statement, we have used three clusters initially which can be expanded in the future to provide better classification. The log parser exhibited an accuracy of 95.30% using F-Measure and an accuracy of 91.03% using Precision Recall.

**7.2 CRITICISMS**

As far as this application is concerned, the log parsing stage employs the usage of a single log set at a time and uses regular expressions to template the logs. This can be enhanced to accept multiple log sets at the same time and introduce a better mechanism for templates’ purpose. Developing a specific anomaly detection approach, should be influenced by the intended purpose and domain of anomaly detection. The performance of the LSTM is impacted by the choice of the definition of an anomaly. For example, in case of the power demand dataset there can be multiple ways to define an anomaly: a day could be an anomaly, an entire week could be treated as an anomaly, or one can even define individual observations to be anomalous. It was found that for the specific datasets the LSTM model was outperformed by feed-forward NNs. It is concluded that for many time series tasks only a few recent time steps are required and LSTM’s ability to remember past information for long time periods does not provide any benefits. The application could do better with a sophisticated categorization algorithm instead of the K-means clustering algorithm which makes use of anomaly weights. Furthermore, the number of clusters being created can be increased from three to a higher number to produce more specific anomaly categorization.

**7.3 FUTURE WORK**

The log parser can be improvised to perform the template process in an efficient manner rather than using regular expressions. The parser can be enhanced to handle a larger volume of logs in a single execution than the current level. It is

widely acknowledged that LSTMs are difficult to train and optimize, as there are a lot of different parameters that are needed to be tuned. By using already trained

models and known values of parameters one can get good results in practice, but

it requires a great deal of effort to understand what makes LSTMs work and what

are their limitations. Studies like have been conducted to understand the purpose of the various LSTM components and to analyze if the different variants of LSTM actually offer any advantage. A new simplified architecture called *gated recurrent unit (GRU)* was proposed in and has become quite popular. This is

another indication that the standard LSTM can be quite overwhelming and we will most likely see more simpler variants in future. The number of clusters can be increased from three to a higher number to produce more specific classifications.

**REFERENCES**

1. L. Amorim, E. Costa, N. Antunes, B. Fonseca, and M. Ribeiro, “Experience report: Evaluating the effectiveness of decision trees for detecting code smells,” *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)*, 2015.
2. Y. Zhang and A. Sivasubramaniam, “Failure prediction in IBM BlueGene/L event logs,” *2008 IEEE International Symposium on Parallel and Distributed Processing*, 2008.
3. Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, “Log clustering based problem identification for online service systems,” *Proceedings of the 38th International Conference on Software Engineering Companion - ICSE 16*, 2016.
4. J. Li and J. Cao, “System Problem Detection by Mining Process Model from Console Logs,” *Lecture Notes in Computer Science Network and Parallel Computing*, pp. 140–144, 2017.
5. W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, “Detecting large-scale system problems by mining console logs,” *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles - SOSP 09*, 2009.
6. S. He, J. Zhu, P. He, and M. R. Lyu, “Experience Report: System Log Analysis for Anomaly Detection,” *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, 2016.
7. A. Pecchia, M. Cinque, G. Carrozza, and D. Cotroneo, “Industry Practices and Event Logging: Assessment of a Critical Software Development Process,” *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015.
8. M. Cheng, Q. Xu, J. Lv, W. Liu, Q. Li, and J. Wang, “MS-LSTM: A multi-scale LSTM model for BGP anomaly detection,” *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, 2016.

.