Chapter 1

Introduction

1.1 Purpose

- Anomaly detection is the identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data. Typically the anomalous items will translate to some kind of problem such as bank fraud, a structural defect, medical problems or errors in a text. Anomalies are also referred to as outliers, novelties, noise, deviations and exceptions.
- Anomaly detection is applicable in a variety of domains, such as intrusion detection, fraud detection, fault detection, system health monitoring, event detection in sensor networks, and detecting ecosystem disturbances. It is often used in preprocessing to remove anomalous data from the dataset.
- Our project aims in detection of anomalies in the operational time-series data received from the devices and identify whether the device is in good working condition or not i.e, system health monitoring.

1.2 Scope

- The main objective is to detect anomalies in the operational data and thereby the health condition of the device.
- Early detection of bad condition of device is encouraged.
- The model should be able to detect the anomalies in an intelligent manner and also visualize the result in a way that a common man can easily understand.

1.3 Literature Survey

A variety of methods were proposed in recent years for anomaly detection like ensemble learners mentioned by Igor Franc et al.,[5]. This paper analyses the performance of ensemble learners on incomplete IoT intrusion datasets, represented by point anomalies. Another efficient method called Multidimensional scaling (MDS) in the paper presented by Bosch[4]. The paper describes how they used MDS

algorithm to identify the anomalies and later visualize the results obtained. Multidimensional scaling (MDS) is one of several multivariate techniques that aim to place objects in N-dimensional space such that the between object distances are preserved as well as possible. They used cosine distance as the dissimilarity measure. When the data is complex then MDS fails to detect anomalies appropriately. Instead we can use one-class SVM or we can detect anomalies by fitting the data in to ellipse in multidimensional space.

Another study by Lingjuan Lyu et al., [2] used Hyperellipsoidal clustering to detect anomalies. This algorithm is not a good choice for time-series data since it cannot handle continuous data arriving from the device to obtain anomalies. Further, glyphs were used by Nan Cao et al., [3]. But, there is still lack of understanding in creation of generic glyphs for generic applications. Glyphs can be used only for domain-specific applications.

Considering the analysis that was done, we want to implement our project using one-class SVM because it is very efficient for time-series data and it works very well when the data is complex.

1.4 Existing Systems

The health condition of the device is usually not monitored continuously. Even if we collect the data, a person has to manually check on if the device is in healthy condition or not. The anomalies in the device is not detected at early stages itself. The anomalies are detected at severe stages when the device is bad working condition.

Automated systems to identify anomalies in the device would solve this problem and this is exactly what our project does.

1.5 Proposed System

The proposed model automates the detection process. The anomalies detected are then visually shown to the user so that it is easily understood by him/her rather than the complicated representation that is generally used to show the anomalies detected.

The proposed system uses one-class SVM for detection of anomalies. The model is trained on the unlabeled data. This trained model is then used to identify if anomalies exist in new input given.

1.6 Statement of the problem

Detection of anomalies is a classic problem that has been around in the industry for years. But with the ample amount of data available these days we are trying to leverage machine learning to detect anomalies in order to deduce the anomalies that exist in the data. This is would help us to identify the working condition of a device. Our projects aims to detect the anomalies that exist is the data and also visualize the response obtained after detection for easy identification of anomalies.

Three broad categories of anomaly detection techniques exist.

- Unsupervised anomaly detection techniques detect anomalies in an unlabeled test data set under the assumption that the majority of the instances in the data set are normal by looking for instances that seem to fit least to the remainder of the data set.
- Supervised anomaly detection techniques require a data set that has been labeled as ”normal” and ”abnormal” and involves training a classifier (the key difference to many other statistical classification problems is the inherent unbalanced nature of outlier detection).
- Semi-supervised anomaly detection techniques construct a model representing normal behavior from a given normal training data set, and then test the likelihood of a test instance to be generated by the learnt model.

Our project uses one-class SVM which is an unsupervised anomaly detection technique in order to identify whether anomalies are existing in the operational time-series data obtained from the device or not.

1.7 Summary

This chapter gave a brief introduction on what exactly the proposed system is. It also covered the future scope and the demand of this product in the market. Also how this product can help any technician understand whether the device is in healthy condition or not. It even marked the main features of the device which helps in overcoming the drawbacks of the existing system.

Chapter 2

System Requirements Specifications

2.1 Software Requirements Specifications

Requirement specification is the movement of interpreting the data assembled amid investigation into prerequisite report.
Software requirements specifications are the detailed enlisting of all necessary requirements that arise in the project. The aim of having these requirements is to gain an idea of how the project is to be implemented and what is to be expected as a result of the project. The sections in this chapter deal with the various kinds of software, hardware and other functional and non functional requirements of the project. A brief description of the various users of the system is also mentioned.

2.1.1 Operating Environment

This section gives a brief about the hardware and software prerequisites for the project.

Hardware Requirements

- **Processor**: 1.6GHz or faster processor
- **RAM**: 1 GB(32 bit) or 2GB(64 bit)
- **Storage**: 250GB of available hard disk space
- Other general hardwares such as a mouse and keyboard for inputs and a monitor for display.

Software Requirements

- **Operating system**: Ubuntu 16.04 and above
- **Programming languages**: Python, Java
- **Software tools**: Android Studio 3.4.0
- **Documentation**: Overleaf

2.1.2 Functional Requirements

Functional requirements are a formal way of expressing the expected services of a project. We have identified the functional requirements for our project as follows:

- The system should be able to gather data.
- The system should be able to detect the anomalies in the data.
- The system should be able to have the capacity to decide the contribution of each attribute towards the decision made by the predictor.

2.1.3 Non-Functional Requirements

Non functional requirements are the various capabilities offered by the system. These have nothing to do with the expected results, but focus on how well the results are achieved.

- Usability: The detection system is highly user-friendly and conveniently usable because of the easy-to-use graphical user interface.
- Reliability :The detection of anomalies in the data help the technicians to identify whether a device is in healthy condition or not.
- Security : The system must provide necessary security and must secure the whole process from crashing.
- Performance : The software system should be able to detect the anomalies with a good probability.
- Re-usability : The degree to which existing applications can be reused in new application. The predicted output could be reused in many fields.

2.1.4 User Characteristics

There is only one type of user associated with the system:

- Technician: He/She may use this project to identify if any anomaly present in the system and thereby the health condition of the device can therefore be deduced.

2.1.5 Applications

The main advantage of using this project is to identify the health condition of the device. The app that we would build can help the user to monitor his/her device so as to find anomalies. If anomalies are present then the device is not in a good working condition. The user can obtain this information at early stages itself than when the device is completely damaged.

2.2 Summary

This chapter discussed the basic software and hardware requirements. More importantly it discusses the functional and non-functional requirements along with the details of different applications of our project and regarding the users who would use our project.

Chapter 3

High Level Design

This section mainly covers the design technique of the entire system which involves the implementation of 2 modules which are as follows:

- Training the machine learning model to detect the anomalies
- Using the trained model to visualize the anomalies and correct data points in the form of a graph using Android Studio as UI interface

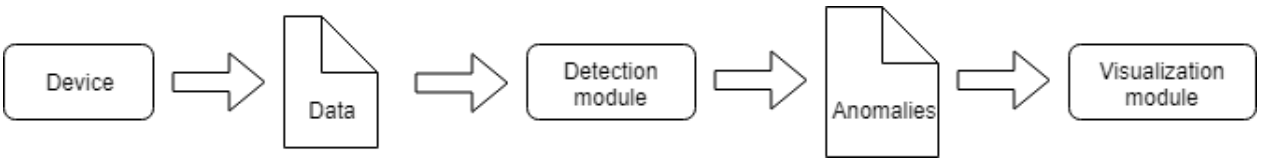

Figure 3.1: High Level Design

3.1 Design Approach

Here are two methodologies for software designing:

- Top-down Design:It takes the entire programming framework as one entity and after that disintegrates it to accomplish in excess of one subsystem or some components based on few attributes.
- Bottom-up Design: The model begins with most particular and essential components. It accedes with making more elevated amount out of subsystems by utilizing essential or lower level components.

As mentioned above the project requires two main modules to be implemented. Each module has its own components to be developed. We use bottom-up design strategy in this product design phase as we start designing the basic components in each module and finally we interlink both the modules to get the final product. So, we would build detection module and visualization module first after which we will integrate both of these modules.

3.2 System Architecture

The architecture diagram design outline gives a review of a whole framework, distinguishing the primary segments that would be created for the item and their interfaces.

Figure 3.2: System Architecture

3.3 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the ”flow” of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated.

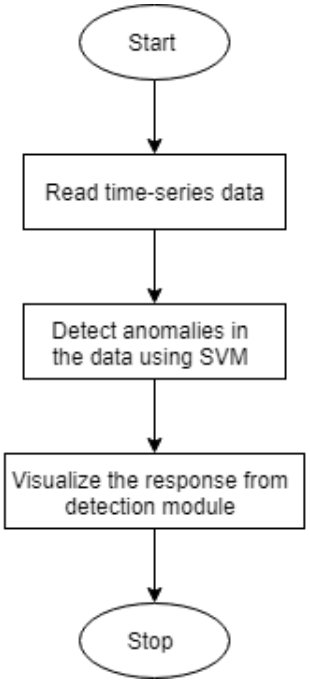

Figure 3.3: Data Flow Diagram

3.4 Sequence Diagram

A sequence diagram is the representation of interactions of components among each other in order. It shows the interactions between the objects in the system with the time order that particular interaction takes place. Since it shows the time order of the interactions it is called as sequence of events and hence the sequence diagram.

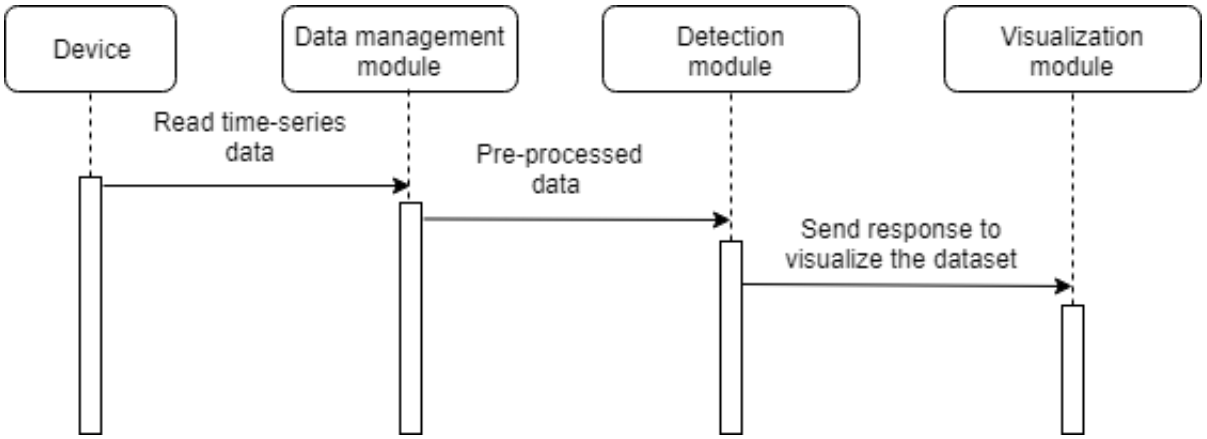

Figure 3.4: Sequence Diagram

3.5 Summary

In this chapter we discussed the different design patterns which can be used in any product development cycle. For our project mainly sequence diagram is used. We also discussed the flowchart which shows the data flow between various components. This chapter even described the high level design.

Chapter 4

Detailed Design

4.1 Purpose

The purpose of the detailed design is to plan our system to meet the requirements specified at the start. In the detailed design we see what is the input data for each model, how the model implementation is carried out and how the output is interpreted. The basic purpose of the project is to detect anomalies and show the anomalies visually in Android application.

4.2 Module 1: Detection

The following is the detailed design of detection module.

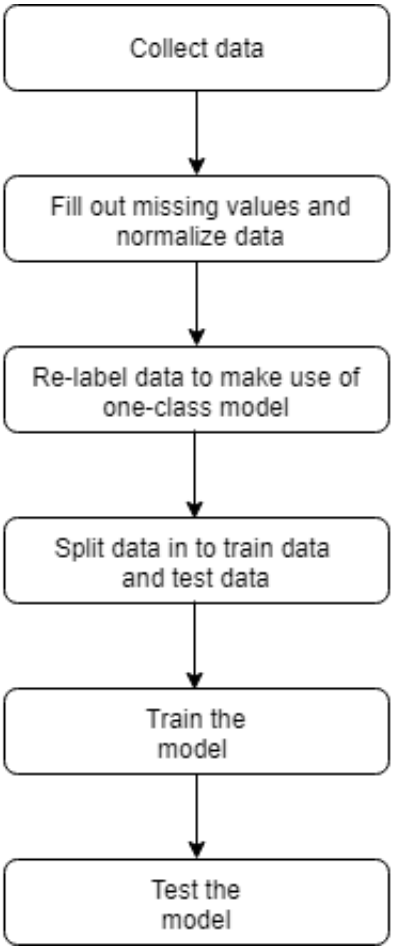

Figure 4.1: Detailed Design

The dataset we used is obtained from the link https://data.world/cityofchicago/beach-water-quailty-automated-sensors. In the pre-processing phase the missing values are handled. This obtained dataset is split into training and test data. The model is trained with the help of train data and the obtained model which is then tested against the test data achieved.The model detects if anomalies exist in the data or not. This ends the detection phase.

4.3 Module 2: Visualization of data

Data visualization is a general term that describes any effort to help people understand the significance of data by placing it in a visual context. Patterns, trends and correlations that might go undetected in text-based data can be exposed and recognized easier with data visualization software. The response from the detection module is used to visualize the data through which the anomalies and the normal data points can be differentiated. This visualization is finally shown in the android application.

The following diagram indicates how the project was constructed:

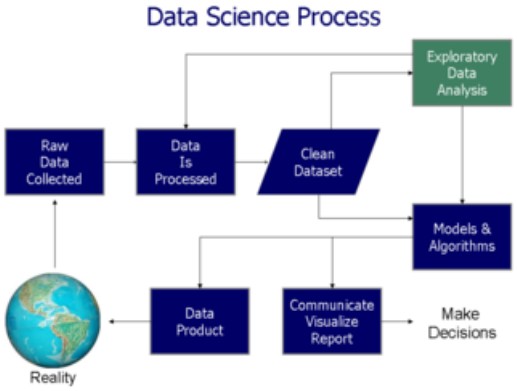

Figure 4.2: Procedure undertaken in the project

4.4 Historic operational Dataset

The historic operational dataset contains several features such as water temperature, turbidity, transducer depth, wave height, wave period, battery life. The dataset contains the information of IoT devices used in different beaches. Each beach contains all the above features.These features are mentioned below in the figure 4.2.The approach of this project was from a hypothetical perspective so data

preprocessing has been done. This was done to provide the classifier with the highest quality of data also can be used to test maximum amount of real time data.

	Beach Name	Measurement Timestamp	Water Temperature	Turbidity	Transducer Depth	Wave Height	Wave Period	Battery Life	Measurement Timestamp Label	Measurement ID
2	Montrose Beach	08/30/2013 08:00:00 AM	20.3	1.18	0.891	0.08	3	9.4	8/30/2013 8:00 AM	MontroseBeach201308300800
3	Ohio Street Beach	05/26/2016 01:00:00 PM	14.4	1.23		0.111	4	12.4	05/26/2016 1:00 PM	OhioStreetBeach201605261300
4	Calumet Beach	09-03-2013 16:00	23.2	3.63	1.201	0.174	6	9.4	09-03-2013 16:00	CalumetBeach201309031600
5	Calumet Beach	05/28/2014 12:00:00 PM	16.2	1.26	1.514	0.147	4	11.7	5/28/2014 12:00 PM	CalumetBeach201405281200
6	Montrose Beach	05/28/2014 12:00:00 PM	14.4	3.36	1.388	0.298	4	11.9	5/28/2014 12:00 PM	MontroseBeach201405281200
7	Montrose Beach	05/28/2014 01:00:00 PM	14.5	2.72	1.395	0.306	3	11.9	5/28/2014 1:00 PM	MontroseBeach201405281300
8	Calumet Beach	05/28/2014 01:00:00 PM	16.3	1.28	1.524	0.162	4	11.7	5/28/2014 1:00 PM	CalumetBeach201405281300
9	Montrose Beach	05/28/2014 02:00:00 PM	14.8	2.97	1.386	0.328	3	11.9	5/28/2014 2:00 PM	MontroseBeach201405281400
10	Calumet Beach	05/28/2014 02:00:00 PM	16.5	1.32	1.537	0.185	4	11.7	5/28/2014 2:00 PM	CalumetBeach201405281400
11	Calumet Beach	05/28/2014 03:00:00 PM	16.8	1.31	1.568	0.196	4	11.7	5/28/2014 3:00 PM	CalumetBeach201405281500
12	Montrose Beach	05/28/2014 03:00:00 PM	14.5	4.3	1.377	0.328	3	11.9	5/28/2014 3:00 PM	MontroseBeach201405281500
13	Calumet Beach	05/28/2014 04:00:00 PM	17.1	1.37	1.52	0.194	4	11.7	5/28/2014 4:00 PM	CalumetBeach201405281600
14	Montrose Beach	05/28/2014 04:00:00 PM	14.4	4.87	1.366	0.341	3	11.9	5/28/2014 4:00 PM	MontroseBeach201405281600
15	Calumet Beach	05/28/2014 05:00:00 PM	17.2	1.48	1.525	0.203	4	11.7	5/28/2014 5:00 PM	CalumetBeach201405281700
16	Montrose Beach	05/28/2014 05:00:00 PM	14.1	5.06	1.382	0.34	4	11.9	5/28/2014 5:00 PM	MontroseBeach201405281700
17	Montrose Beach	05/28/2014 06:00:00 PM	14.2	5.76	1.415	0.356	3	11.9	5/28/2014 6:00 PM	MontroseBeach201405281800
18	Calumet Beach	05/28/2014 06:00:00 PM	17.1	1.8	1.501	0.188	4	11.7	5/28/2014 6:00 PM	CalumetBeach201405281800
19	Calumet Beach	05/28/2014 07:00:00 PM	17	1.82	1.487	0.194	4	11.7	5/28/2014 7:00 PM	CalumetBeach201405281900
20	Montrose Beach	05/28/2014 07:00:00 PM	14.2	6.32	1.386	0.346	3	11.9	5/28/2014 7:00 PM	MontroseBeach201405281900
21	Montrose Beach	05/28/2014 08:00:00 PM	14.4	6.89	1.401	0.38	4	11.9	5/28/2014 8:00 PM	MontroseBeach201405282000
22	Calumet Beach	05/28/2014 08:00:00 PM	17	1.83	1.488	0.196	4	11.7	5/28/2014 8:00 PM	CalumetBeach201405282000

Figure 4.3: Historic operational dataset

This dataset can be used as a training grounds to best type of fit for all the further incoming data of similar nature.As mentioned before, the isolation of features as required by Anomaly detection is necessary to initiate the right model.

4.5 One-class SVM - Unsupervised learning algorithm

After the Data Assessment is done in the previous section it now falls heavily on us to choose the classifier.There are multiple ways we can approach this problem such as one-class SVM, Ensemble method, Bayesian networks. We have chosen one-class SVM in order to do our project. Unsupervised machine learning is machine learning without labelled data (where data hasn't been labelled beforehand to say what it is – in our case, whether a data point is an anomaly or not).

Unsupervised machine learning can be useful in information security problems because a) we don't always have accurately labelled data, and b) we want to be able to identify novel (or anomalous) observations in data without having necessarily seen an example of that behaviour in the past.

Support vector machines (SVMs) are a type of learning model used for classification and regression analysis. In an SVM data points are represented as points in space in such a way that points from different categories are separated by a plane. You can think of this like a line through data points that separates data of different classes just like the how the above diagram describes.

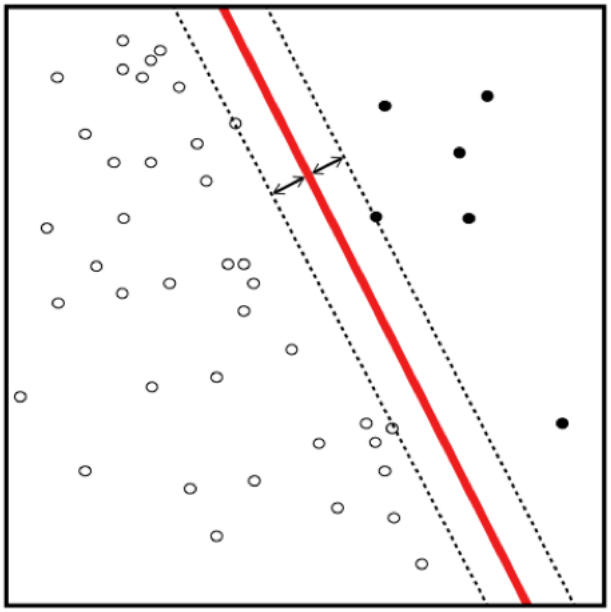

Figure 4.4: one-class SVM - unsupervised learning

New data are mapped into the same space and their location relative to the plane is used to predict which categories each point belongs, with the plane being referred to as the decision boundary (i.e. determining to which class the data belongs). In the case where the decision boundary needs to be non-linear (i.e. where classes cannot be separated by a straight line), SVMs also have the ability to project space through a non-linear function, lifting the data to a space with a higher dimension where a linear decision boundary does separate classes.

One-class SVMs are a special case of support vector machine. First, data is modelled and the algorithm is trained. Then when new data are encountered their position relative to the ”normal” data (or inliers) from training can be used to determine whether it is ”out of class” or not - in other words, whether it is unusual or not. Because they can be trained with unlabelled data they are an example of unsupervised machine learning.

The main steps followed in one-class SVM is:

- SVM implementations work better with normalized data. This yields both better accuracy and reduces numerical instability that is inherent in their implementation. Thus, the data that we collected is extracted and normalized.
- The data must be mapped to +1 or -1 to indicate whether the data is an ”inlier” or ”outlier” respectively. This also transforms our data from multi-class (multiple different labels) to one-class (boolean label), which is a prerequisite for using a one-class SVM.

- The data obtained is then split to training set and test set.
- The model is fit using the training data.
- The accuracy is obtained using the test data.
- When new data is given as input, the model is able to detect anomalies. The model will predict the new record is either 1(inlier i.e, not an anomaly) or -1(outlier i.e, anomaly).

4.6 Visualization of anomalies and other data points

Data visualization is viewed by many disciplines as a modern equivalent of visual communication. It involves the creation and study of the visual representation of data. To communicate information clearly and efficiently, data visualization uses statistical graphics, plots, information graphics and other tools. Numerical data may be encoded using dots, lines, or bars, to visually communicate a quantitative message.Effective visualization helps users analyze and reason about data and evidence. It makes complex data more accessible, understandable and usable. Users may have particular analytical tasks, such as making comparisons or understanding causality, and the design principle of the graphic (i.e., showing comparisons or showing causality) follows the task. Tables are generally used where users will look up a specific measurement, while charts of various types are used to show patterns or relationships in the data for one or more variables.

In our project we have used scatter plots in order to show the visualization of the data points.

4.7 Summary

In this chapter we discussed the different modules in our project. We also discussed the algorithm and visualization method that we propose to use in our project. The details about the dataset is also specified. This chapter gives the detailed design of our project.

Chapter 5

Implementation

Implementation is the phase of the undertaking when the hypothetical plan is transformed out into a working framework. Subsequently it can be thought to be the most basic stage in accomplishing fruitful new framework and in giving the client, certainty that the new framework will work and be efficient.

5.1 Programming Language Selection

This project is implemented in Python,Java, Flask and Android Studio as they are:

- Simple: The languages was designed to be easy for the professional programmer to learn and use effectively. They inherit the C/C++ style and many object oriented features of C++.Android Studio provides fastest tools to create apps for any android device.
- Object-oriented: The object model in Flask is simple and easy to extend, while primitive types are kept as non-objects for performance reasons.
- Robust: Python frees you from worrying about a portion of the basic programming mistakes. It is an entirely composed dialect and checks the code both at arrange time and run time.
- Multi threaded: Java was intended to meet this present reality prerequisite of making intelligent system programs. To accomplish this, it supports server-side programming on the client side.
- Interpreted and High-performance: Web development enables the creation of cross-platform programs by running on the browser so that the requests from client(mobile phone) can be handled.

5.2 Platform Selection

5.2.1 Linux (Ubuntu 16.04)

Linux is one of famous form of UNIX working System. It is open source as its source code is uninhibitedly accessible. It is allowed to utilize. Linux was composed considering UNIX compatibility. Its usefulness list is very like that of UNIX. Virtual conditions have the preferred standpoint that they

never introduce the required dependencies system wide so we have a superior control over the environment in which our application is running. We can choose only to install the required libraries and packages and keep the environment clean. Ubuntu aims to be secure by default. User programs run with low privileges and cannot corrupt the operating system or other users' files. For increased security, the sudo tool is used to assign temporary privileges for performing administrative tasks, which allows the root account to remain locked and helps prevent inexperienced users from inadvertently making catastrophic system changes or opening security holes.

Virtual environment is also important as we may have multiple applications on one system with conflicting requirements.

5.2.2 Flask

Flask is a Python framework, based on Werkzeug, Jinja2 and inspired by Sinatra Ruby framework, available under BSD license. Some of the important features of Flask are:

- built-in development server and fast debugger
- integrated support for unit testing
- RESTful request dispatching
- Jinja2 template
- support for secure cookies (client side sessions)
- WSGI 1.0 compliant
- Unicode based

5.2.3 Python

Python is a translated, object oriented programming language like PERL, that has picked up prevalence in light of its reasonable syntax and comprehensibility. Python is said to be relatively simple to learn and convenient, which means its statements can be translated in various working frameworks, including UNIX-based frameworks, Mac OS, MS-DOS, OS/2, and different renditions of Microsoft Windows 98. Python was made by Guido van Rossum, a previous occupant of the Netherlands, whose

most loved satire gathering at the time was Monty Python's Flying Circus. The source code is unreservedly accessible and 'open for modification and reuse. Python has a critical number of clients. A remarkable highlight of Python is its indenting of source explanations to make the code less demanding to peruse. Python offers dynamic information compose, instant class, and interfaces to numerous framework calls and libraries. It can be broadened, utilizing the C or C++ dialect.

5.2.4 Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

Some of the important features of Android Studio are:

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- ProGuard integration and app-signing capabilities
- Template-based wizards to create common Android designs and components
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine

5.3 Libraries Required

- **scikit-learn**: scikit learn is a wide python library which practices machine learning, cross validation of data, and preprocessing of data. It is built on NumPy and SciPy.

- **NumPy**: NumPy's main object is an multidimensional array, it holds an array data structure. NumPy is a core python library which contains a collection of tools and techniques. One of these tools is multi dimensional object.
- **Pandas**: Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.
- **gspread**: Google spreadsheets python API that can be used to open a google spreadsheet by title, url or key.
- **pickle**: The pickle module implements binary protocols for serializing and de-serializing a Python object structure.
- **oauth2client**: oauth2client is the client framework for Oauth version 2.0. python-oauth2 is a framework that aims at making it easy to provide authentication via OAuth 2.0 within an application stack.
- **matplotlib**: Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits.
- **base64**: This module provides data encoding and decoding as specified in RFC 3548. base64 was used to encode the image to be sent to the android client for display.
- **io**: The io module provides Python's main facilities for dealing with various types of I/O. There are three main types of I/O: text I/O, binary I/O and raw I/O. The io module was used to send the image of graph in the form of bytes to the android client.

5.4 Flowchart

A flowchart is simply a graphical representation of steps. It shows steps in a sequential order, and is widely used in presenting flow of algorithms, workflow or processes. The following is the flowchart that gives the basic idea of the flow of our project:

The data set is preprocessed in such a way that the missing values in the data set are filled and the values are normalized such that the range of the values in small. Once the data is preprocessed, it has to be divided with respect to each beach and further divide with respect to each sensor. This data is then moved to the model for training. Once the training is completed, it can be used further for detection by the users.

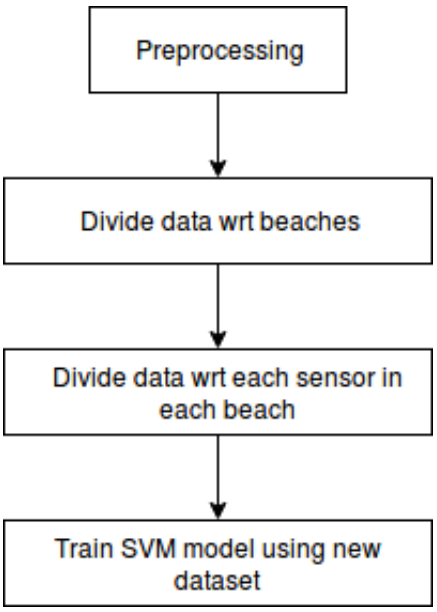

Figure 5.1: Flowchart

5.5 Summary

This chapter dealt with the various techniques used in the development of the project,starting with the language and platform selection along with the details of the libraries used to finally explain the entire process of implementation steps.

Chapter 6

Testing

6.1 Software Testing

To deliver evidence about the excellence of the product or system under test software testing used.From an unbiased view, it tries to establish the quality of the product.

The resolution of software testing is to find the bugs or errors or in the program. Since it is an iterative process, a bug identification can lead to illumination of another bug.

6.1.1 White Box Testing

White box testing is a method of software testing that tests the working of the program or an application rather than functionality.Path testing is an obvious example. It is used in spaces where a black box testing cannot reach.

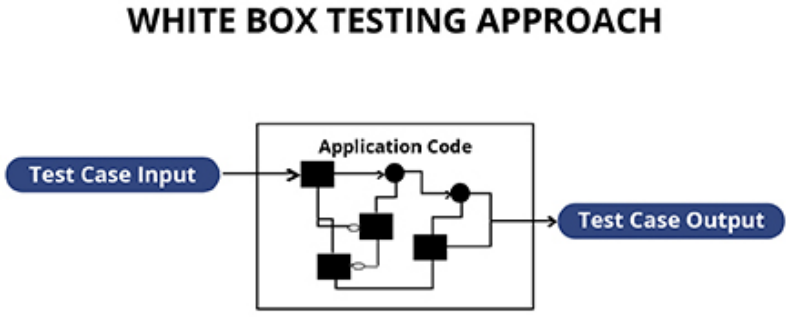

Figure 6.1: White box testing

6.1.2 Black Box Testing

In black box testing the function of the black box is understood completely in terms of inputs and outputs in-spite of not knowing the content or implementation of black box . Specification testing is additionally called as functional testing in light of the fact that the program is considered as a capacity that maps esteems from its info area to values in its yield extend.

Specification based testing as 2 specific points of interest:

- They are not needy of how the product is actualized. On the off chance that if the implementation changes, the experiments are as yet valuable.
- Reduction in the overall project development interval because both the test case development and implementation can occur in parallel.

6.2 Levels of Testing

From the point of view of customers there are 2 different levels:

- **Low-level testing** Low level testing is a set of tests for different level components of software application.
- **High-level testing** High level testing is a set of tests for the application as a whole.

There are normally 3 levels of testing: Unit testing, System testing and Integration testing.

6.2.1 Unit Testing

In the method individual units of source code,program module associated control data, other procedures are tested to find their fit for use. Specific functionality of the developed subunits are tested. The more appropriate testing in unit testing is structural testing. Unit testing is followed by integration testing.

6.2.2 Integration Testing

Integration testing is the second level of testing. Integrated testing combines all the modules of unit testing and tests them. The goal of this form of testing is to test whether there are any problems in the integration of different modules.

6.2.3 System Testing

System testing evaluates whether a system meets the requirements specified before.

It tests the behaviour of system as specified by the customer.It also tests the requirements in software/hardware specification but also tests beyond that. System testing is followed by acceptance testing.

6.2.4 Acceptance Testing

Acceptance testing is used to test the system compliance for the requirements. Hence, the system is tested for acceptability. It is the last test that is performed before making the system.

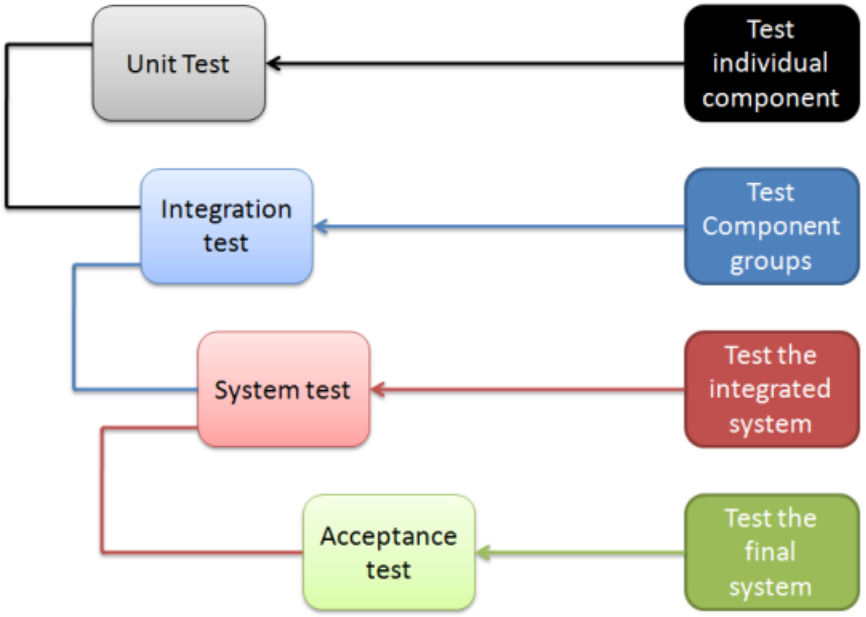

Figure 6.2: Levels of testing

6.3 Unit Testing of Module

Execution of the prediction system is tested for various conditions and the test cases are tabulated as follows:

Test Case ID	Unite Test Case P 1
Description	To test the predictor with input values
Input	Correct device ID: IOT03
Expected Output	New screen must be generated in the android client
Actual Output	New screen was generated in the android client
Remarks	The system performed as expected.

Unit Test Case 1

Test Case ID	Unite Test Case P 1
Description	To test the predictor with input values
Input	Wrong device ID: IOTat
Expected Output	New screen must not be generated in the android client
Actual Output	New screen was not generated in the android client
Remarks	The system performed as expected.

Unit Test Case 2

6.4 Summary

Software testing is a process to show the customers the quality of the product. Unit testing is a method used to test individual modules. Integration testing aggregates all modules that are unit tested and tests them using Integration testing methods. System testing basically tests the system to check whether the system meets all the specified requirements. The chapter also describes the test cases used to test the working of our project.

Chapter 7

User Interface

7.1 Graphical User Interface

A mobile based user interface was implemented for this project.The UI is an android-based application used to design and deal with the server apparatus from a remote client.The application provides a clear description of all the work done in the project. Users will feed the data to the trained model and the results of the prediction are presented on the screen of the mobile.

Following are the technologies used for implementing the UI:

- Android Studio
- Java
- Flask

7.1.1 Flask

Flask is a miniaturized scale web structure written in Python and in light of the Werkzeug toolbox and jinja2 layout engine.Flask is considered and used for the following advantages:

- It’s easy to set up
- It’s well documented
- It’s very simple and minimalistic, and doesn’t include anything you won’t use
- It’s flexible enough that you add extensions,if you need more functionality

Jinja is a layout engine for the python programming dialect and is authorized under a BSD License. It gives Python like articulations while guaranteeing that the layouts are assessed in a sandbox. It is a content based format dialect and along these lines can be utilized to produce any increase and in addition source code.

7.1.2 Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

Android studio was considered and used for the following advantages:

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- ProGuard integration and app-signing capabilities
- Template-based wizards to create common Android designs and components
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine

7.2 Snapshots

The snapshots of the user interface of the project are depicted in the following figures

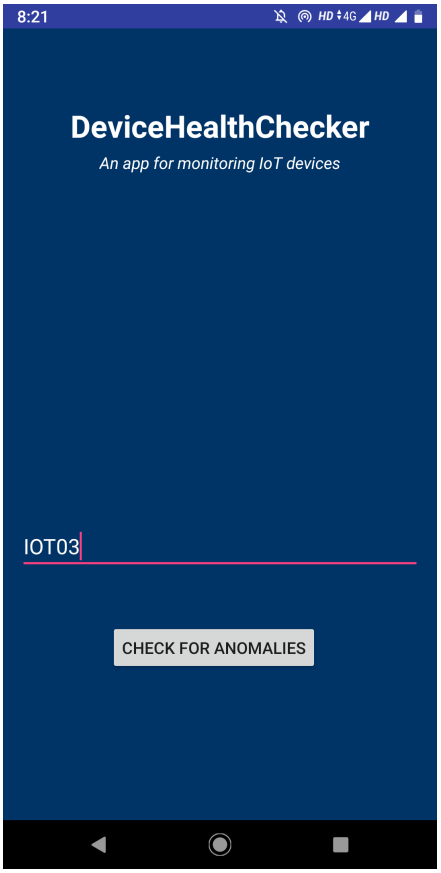

Figure 7.1: Snapshot

By giving the appropriate input values(i.e, device id of IoT devices) would redirect the page to the below figure 7.2 if all the sensors are working correctly otherwise the output would be like figure 7.3 if any of the sensors are not working properly

Figure 7.2 and 7.3 as shown includes the performance of the sensors present in the IoT device depending on the details entered by the user.

In order to see the visualization of the data points, visualize button in the second screen must be clicked(i.e, figure 7.2). The visualization is shown in Figure 7.4.

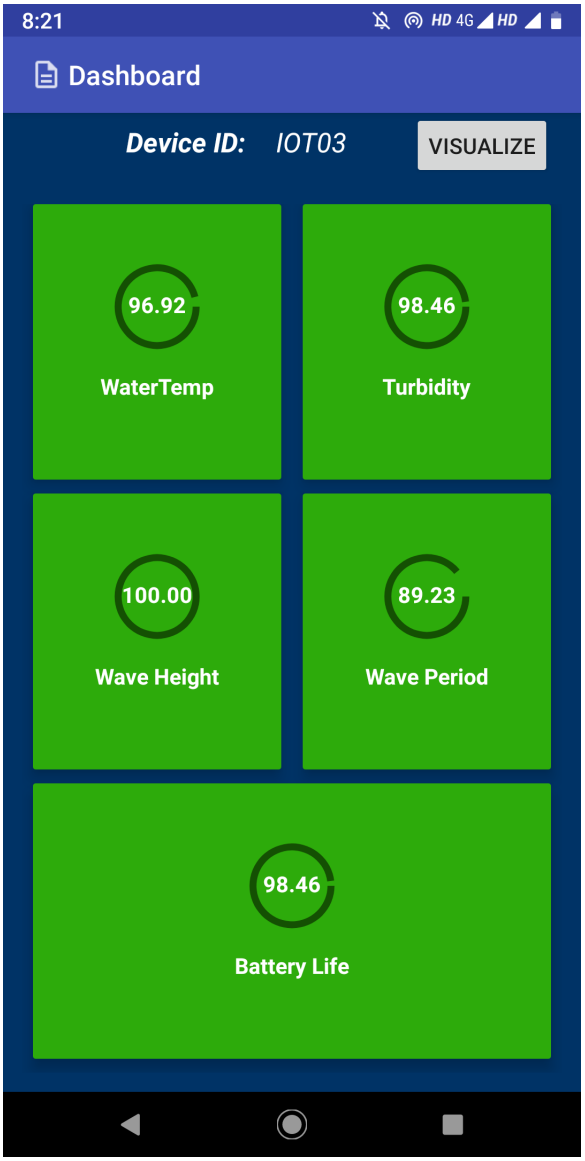

Figure 7.2: Snapshot

Figure 7.3: Snapshot

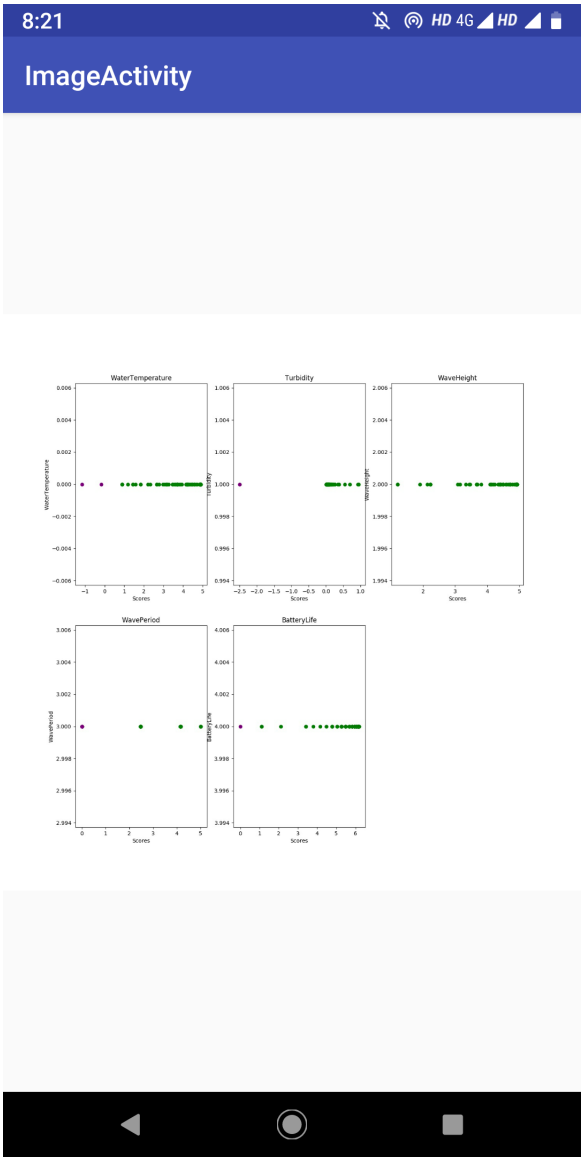

Figure 7.4: Snapshot

Chapter 8

Analysis & Conclusion

8.1 Analysis

As part of detection stage in our system, we initially split the dataset into training and test dataset. This dataset is stored in the google cloud for simulation purpose. The dataset is filtered so that the missing values are filled and the values are normalized such that there is no much difference in the range of the values present in the dataset. Whenever the user enters the device id at a particular time, the records from the cloud are fetched randomly in order to identify which of these are outliers and which of them are not outliers. Depending on the number of outliers present in these selected records, the performance of the sensor is determined. Since we are monitoring the device, we also compare this performance with the earlier performance and thereby we conclude whether is working better or not. Once the detection is completed, the data points are plotted in the form of a scatter plot.

The summary of our analysis is as follows:

- Number of outliers detected indicate whether the device is in working condition or not.
- When the number of outliers detected are small in number, the device is said to work better.
- When the number of outliers are high in number, the device is said to work abnormally.
- By comparing the current performance and earlier performance, we can identify if the device is working better compared to earlier time or it is deteriorating.
- The output that we have produced is in such a way that if the device is working better then the sensor is shown in green card otherwise it is shown in red card so that it is easier even for a common man to understand without any difficulty.

8.2 Conclusion

The project outlines the analysis done on the historic operational dataset of the IoT devices used in the beaches. The main objective of the project was to detect anomalies in this dataset and thereby identify if the device is in the good working condition or not. The algorithm - One class SVM was used in order to detect the anomalies present in the dataset. Once the output from detection is obtained, the visualization of data points is done in the form of a scatter plot. This project can be applied to any such IoT devices in order to detect anomalies to identify their working condition. The

model should be updated periodically and include more additional features for it to make more accurate analysis of anomalies.

8.3 Limitations

- The results of the study were limited to the area in which research data was collected.
- This research takes into consideration only those predictors that are in the range of our study.

8.4 Future Work

- Future researchers can expand the model by including several other IoT devices which need continuous monitoring.
- Future work can also consider including more factors from the database that could have more effect on deciding representative factors.

8.5 Summary

In this project we see the analysis of the project which we created.The analysis is done by taking in the historical operational data set and splitting it into training and test data. This data is taken which is further filtered so that missing values can be filled and the values can be normalized and One-class SVM algorithm is applied. The thus modeled classifier is able to detect the anomalies and identify the healthy condition of the device. The output from the classifier is then visualized in the form of a scatter plot.

The limitations of the project are mentioned in the next section and future scope shows what development can be made to get better results.

Bibliography

[1] Lingjuan Lyu, et al., (2017) *, Fog-Empowered Anomaly Detection in IoT Using Hyperellipsoidal Clustering*. IEEE Internet of Things Journal (Volume: 4, Issue: 5, Oct. 2017)

[2] Nan Cao et al., (2017) *, Z-glyphs: Visualizing outliers in multivariate data,* Information Visualization, SAGE publications 2017

[3] Software Innovations - Bosch IoT Suite (2016) *, Anomaly detection with event data in the Internet of Things*.

[4] Igor Franc, et al., (2016) *, Detecting Malicious Anomalies in IoT: Ensemble Learners and Incomplete Datasets,* The Eight International Conference on Business Information Security (BISEC), At Metropolitan University, Belgrade

[5] Deris Stiawan, et al., (2016) *, Anomaly detection and monitoring in Internet of Things communication,* 2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE)

[6] Goverdhan Reddy Jidiga et al., (2014) *, Anomaly detection using machine learning using a case study,* IEEE International Conference on Advanced Communications, Control and Computing Technologies, 2014

[7] Daniel A. Keim (2002) *, Information Visualization and Visual Data Mining,* IEEE transactions on visualization and computer graphics, vol. 7, no. 1, January-March 2002