# CHAPTER 1

# 1 INTRODUCTION

**1.TITLE OF THE PROJECT”SENSOR DATA QUALITY MANAGMENT”**

## 1.1 Introduction

A new generation of environmental sensors and recent major technological advancements in the acquisition and real-time transmission of continuously monitored environmental data provides a major challenge in providing quality assurance (QA) and quality control (QC) for high-throughput data streams. Deployments of sensor networks are becoming increasingly common at environmental research locations, and there is a growing need to access these large volumes of data in near real-time. However, the direct release of streaming sensor data raises the likelihood that incorrect or misleading data will be made available.

Additionally, as research applications begin to rely on real-time data streams, the continual and consistent delivery of this information will be essential. This increasing access and use of environmental sensor data demands the development of strategies to assure data quality, the immediate application of quality control methods, and a description of any QA/QC procedures applied to the data. Traditional QC systems tend to operate on file-based collections of environmental data from field sheets, field recorders or computers, or downloaded data files. Manually applied tools and techniques such as graphical comparisons are used to provide data validation. Documentation is typically not well-organized and not directly associated with data values.

The application of these systems must balance the need for release without months or years of delay versus the delivery of well-documented, high quality data. However, with increasing deployment of sensor networks, these older systems fail to scale or keep pace with user needs associated with high volumes of streaming data. Comprehensive and responsive QC systems are needed that are designed to reduce potential problems and can more quickly produce high quality data and metadata.

One important aspect of quality monitoring information is the regular feedback of the information to data producers, program managers, airlines, and other interested parties. This is essential in ensuring that issues with data will be corrected when they arise.

## 1.2 Objectives

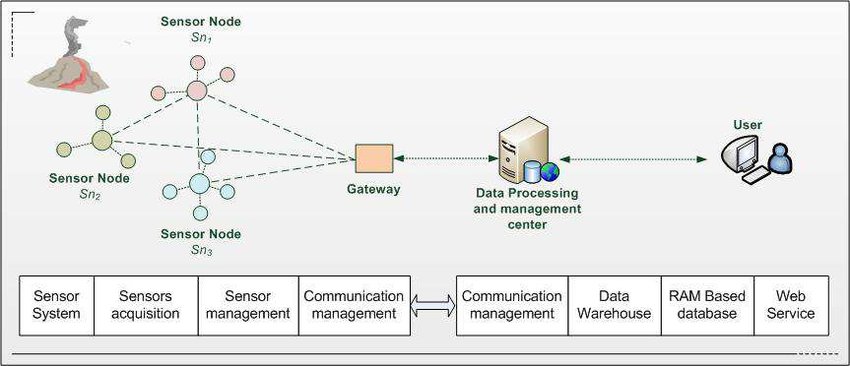
Technological improvement on sensors and sensor networks has opened many opportunities to use and combine geospatial data coming from sensors. Today it is much easier to use geospatial sensor data in all sorts of applications as for environment detecting, weather detecting, animal tracking, environmental monitoring, farmlands monitoring, etc. Despite a strong interest from geospatial domain, the integration of sensors raises new technical and data management challenges, like sensor data uncertainty. In fact, sensor devices have hardware restrictions and perform data collection in hostile environments turning data more imprecise and uncertain. Moreover, the quality of sensor data is often decreased by sensor failures or malfunctions. Thus, deficiencies on sensor data cannot be ignored, but tackled in order to reduce information misunderstanding and assist experts in decision making process by data quality management.

## 1.3 Scope

This data quality management is applied to correct the data by providing default values, formatting numbers and dates, and removing missing values, null values, non-relevant values, duplicates, out of bounds, referential integrity violations, and value integrity violations.

## 1.4 Applications

Our research is motivated by the analysis of data quality especially for environmental monitoring applications. With our approach, we aim to identify and tackle the most transcendental aspects of this problematic. An Environmental Monitoring System (EMS) refers to the activities or processes used to characterize and monitor the environment. Generally, this kind of systems is used to support environmental risk management and evaluate its impact. Through the years, the EMS have became a key of monitoring programs all over the world and applied to control a variety of chemical, biological, or radiological aspects for instance.



**CHAPTER 2**

# 2 LITERATURE SURVEY

Existing System A common classification framework characterized by several quality dimensions, allows us to compare dimensions across different data types. The framework is based on a classification in clusters of dimensions. where dimensions are included in the same cluster according to their similarity. Clusters are defined in the following list, where the first item initalics is the representative dimension of the cluster, thus introducing other member dimensions:

1. Accuracy, correctness, validity and precision focus on the adherence of data to a given reality of interest.

2. Completeness, pertinence and relevance refer to the capability of representing all and only the relevant aspects of the reality of interest.

3. Redundancy, minimality, compactness and conciseness refer to the capability of representing the reality of interest with the minimal use of informative resources.

4. Readability, comprehensibility, clarity and simplicity refer to ease of understanding of data by users.

5. Accessibility and availability are related to the ability of the user to access data from his or her culture, physical status/functions, and technologies available.

6. Consistency, cohesion and coherence refer to the capability of data to comply without contradictions to all properties of the reality of interest, as specified in terms of integrity constraints, data edits, business rules and other formalisms.

7. Trust, including believability, reliability and reputation, catching how much data derive from an authoritative source.

+Data quality has been investigated focusing especially on data as represented in the relational model, traditionally adopted in Data Base Management Systems

We are using the pyspark and seaborn to represent the quality of data as fast as possible in the best way. Quality assurance (QA) refers to preventative measures and activities used to minimize inaccuracies in the data. For example, scheduling regular site visits and maintenance procedures, or continuously monitoring and evaluating site sensor behaviour can prevent sensor failures or lead to early detection of problems. Designing networks with redundant sensor measurements provides an additional means to quality check sensor data and assure continuity of measurement. Of course, the time and expense to conduct high-level maintenance procedures or implement efficient and redundant designs may be limited by project budgets, but may be warranted by the importance of the data.

**CHAPTER 3**

# 3 System Analysis

## 3.1 Existing System

From the point of view of data quality management, it is quite important not to make the mistake of confusing the readings of dysfunctional sensors with inadequate levels of data quality, even when a dysfunctional sensor can produce data without having adequate levels of quality. The reason for making this distinction is that fixing errors due to dysfunctional sensors requires first fixing the sensors; on the other hand, if one can assure that the root cause is not grounded on a dysfunctional sensor, but on the data itself, then, to fix data quality errors, then data quality management techniques should be used since it should not be ignored what data means. The description of the data quality characteristics can be found in the following paragraphs:

**Accuracy**: It is the degree to which data has attributes that correctly represent the true value of the intended attribute of a concept or event in a specific context of use. A low degree of accuracy could be derived from devices that provide values that could differ from the value on the real world. For example, a low degree of accuracy can be when a humidity sensor reads a value of 30% and the real value is 50%. Low levels of accuracy could be directly related to sensor errors such as constant or offset, outlier errors and noise errors. Also, accuracy could be indirectly affected by continuous varying or drifting and trimming error.

**Completeness:** It is the degree to which subject data associated with an entity has values for all expected attributes and related entity instances in a specific context of use. A low degree of completeness could be derived from devices that are reading and sending no values. For example, a low degree of completeness can happen when the records of sensor data have missing values. Low levels of completeness could be related directly to sensor errors such as crash or jammed errors and indirectly, trimming and noise errors.

**Consistency:** It represents the degree to which data has attributes that are free from contradiction and are coherent with other data in a specific context of use. It can be either or both among data regarding one entity and across similar data for comparable entities. A low degree of consistency could happen when two sensors produce contradictory data. For example, for proximity sensors that provide the relative distance to the sensor position, a consistency problem for a single sensor could be a negative distance value, while a consistency problem between two sensors in the same position could be two different values. Thus, low levels of consistency could be related with continuous varying/drifting error and indirectly with constant or offset errors, trimming and noise error.

**Credibility:** It is defined as the degree to which data has attributes that are regarded as true and believable by users in a specific context of use. A low degree of credibility could be derived from a single sensor placed in someplace and the data cannot be validated by another entity or even sensor. For example, a credibility issue could happen when a sensor whose data is compared with another sensor placed near does not match. Low levels of credibility could be related directly to sensor errors such as outlier errors and indirectly with constant or offset error, continuous varying/drifting error and noise error.

**Accessibility:** It is the degree to which data can be accessed in a specific context of use, particularly by people who need supporting technology or special configuration because of some disability. A low degree of accessibility could be derived due to the necessary user is not allowed in the precise moment. For example, data produced by a specific sensor is unreachable due to network issues.

**Compliance:** It refers to the degree to which data has attributes that adhere to standards, conventions or regulations in force and similar rules relating to data quality in a specific context of use. A low degree of compliance could be derived from data sensor that is not being using the standards formats established on the organization. For example, if the organization establishes that for distance sensors the unit for values is meters, and if some sensors produce values expressed in meters and other in miles these data have low compliance levels.

**Confidentiality:** It is the degree to which data has attributes that ensure that it is only accessible and interpretable by authorized users in a specific context of use. A low degree of confidentiality could be derived from an inefficient security management of sensor data. For example, a confidentiality leak might happen when data produced by a sensor placed in a nuclear power plant can be freely accessed from external networks even when this data was marked as sensible and, therefore, confidential in order to prevent possible terrorist acts.

**Efficiency:** It is the degree to which data has attributes that can be processed and provide the expected levels of performance by using the appropriate amounts and types of resources in a specific context of use. For example, a sensor send data about where is placed and send a code and a description every time the sensor sends o stores a record, it has low efficiency because only the code is enough to know all about the place. A low degree of efficiency could be derived from the storage of duplicated data that could take more time and resources to send or manipulate the data.

**Precision:** It is the degree to which data has attributes that are exact or that provide discrimination in a specific context of use. A low degree of precision could be derived from devices that are providing inexact values as in the next example. For example, sensor data that store weight with no decimals and it is required a minimum of three decimals. Low levels of consistency could be related directly with trimming errors, and indirectly with noise errors.

**Traceability**: The degree to which data has attributes that provide an audit trail of access to the data and of any changes made to the data in a specific context of use. A low degree of traceability could be derived from sensor data with no metadata. For example, data logs contain information about who has acceded to sensor data and operations made with them. Low levels of traceability could be related indirectly to crash or jammed errors as well as to temporal delay errors.

**Understandability:** The degree to which data has attributes that enable it to be read and interpreted by users, and are expressed in appropriate languages, symbols and units in a specific context of use. A low degree of understandability could be derived from sensor data represented with codes instead of acronyms. For example, records of data about temperature on a car has an attribute to know the place of the sensor in the car. This attribute can be stored as a code like “xkq1”, but if is stored as “GasolineTank” it is supposed to have a higher level of understandability.

**Availability:** The degree to which data has attributes that enable it to be retrieved by authorized users and/or applications in a specific context of use. A low degree of availability could be derived from the insufficient resources of the system in which sensor data is stored. For example, to assure sensor data availability, sensor replication can be used to make it available even if there is some issue on a sensor. Low levels of availability could be related indirectly with temporal delay error and crash or jammed error.

**Portability**: The degree to which data has attributes that enable it to be installed, replaced or moved from one system to another preserving the existing quality in a specific context of use. For example, sensor data is going to be shared with a concrete system or even other organization or department, data loss can occur. If this happens, for example, due to a data model mismatching or a problem with the data format, the reason is directly related to portability of data. A low degree of portability could be derived from sensor data that does not follow a specific data mode or the format present some problems.

**Recoverability:** The degree to which data has attributes that enable it to maintain and preserve a specified level of operations and quality, even in the event of failure, in a specific context of use. A low degree of recoverability could be derived from devices that does not have a mechanism failure tolerant or backup. For example, when a device has a failure, data stored in that device should be recoverable. Low levels of recoverability could be related indirectly with temporal delay error and crash or jammed errors.

Although our DQ Model considers initially all the DQ characteristics defined in ISO/IEC 25012, it could be necessary to customize the DQ characteristics chosen to adapt them into the specific SCP context. This customization might depend on the concrete organization and how it applies the methodology to specific SCP contexts. The customized model will conform the data quality model for an organization with a specific SCP environment.

## 3.2 Disadvantages

This enumeration is important to the survey, to understand the most basic origins of faults in sensors. The sensor material characteristics or the harshness of the environmental conditions lead to the production of a specific kind of fault. Some sensors strive to perceive an object that is moving in dusty environments, while others experience issues reading a correct level observation in fluids.

The possible types of errors observed in measurement values can be classified as follows:

* Random errors are described by an absence of repeatability in the readings of the sensor, for instance due to measurement noise. These errors tend to happen on a permanent basis, but have a stochastic nature;
* Systematic errors are described through consistency and repeatability in the temporal domain.

There are three types of systematic errors at the sensor level:

* Calibration errors result from errors in the calibration procedure, often in relation to linearization procedures;
* Loading errors emerge when the intrusive nature of the sensor modifies the measurand. Along with calibration errors, loading errors are caused by internal processes;
* Environmental errors emerge when the sensor experiences the surrounding environment and these influences are not considered. In contrast with the previous two types of errors, environmental errors are due to external factors;
* Spurious readings are non-systematic reading errors. They occur when some spurious physical occurrence leads to a measurement value that does not reflect the intended reality. For instance, a light intensity measurement in a room can provide the wrong value if obtained precisely when a picture of the room is taken and the camera flash is triggered.

## 3.3 Proposed System

The process of environment monitoring requires sensor devices to be deployed within the system. These sensors will be the entities responsible for measuring the parameters of interest, like temperature, water level or salinity. A sensor essentially converts a physical quantity in its input to an electrical signal, produced as the output, which is usually proportional to the input. Further to the sensor itself, additional components are needed to perform signal processing functions, store measured values and communicate these values to other systems. It is hence usual to refer to these more complex components as smart sensors or intelligent sensors, typically interconnected to other smart sensors to form wireless sensor networks.

The data is analysed for required values, validate data types, and detect integrity violation. DQM is applied to correct the data by providing default values, formatting numbers and dates, and removing missing values, null values, non-relevant values, duplicates, out of bounds, referential integrity violations, and value integrity violations.

We perform

Data Integrity: Data integrity is the process of guaranteeing the quality of the data in the database.

Data Profiling: Data profiling is the process of discovering and analyzing enterprise metadata to discover patterns, entity relationships, data structure, and business rules. It provides statistics or informative summaries of the data to assess data issues and quality.

Data Cleansing: Data cleansing is the process of identifying incomplete, incorrect, inaccurate, duplicate, or irrelevant data and modifying, replacing, or deleting the dirty data.

Data Transformation: Data transformation deals with converting data from the source format into the required destination format.

Increasing high quality and absoluteness in the data, The importance of evaluating and communicating information quality in emerging geospatial applications like environmental monitoring. As our research work shows, several external factors may impact the quality of sensor data and thus directly impact users’ decision making.

## 3.4 Advantages

The interpretation and modelling of the available information into adequate theoretical frameworks is the main means to characterize the quality of the obtained sensor data.

**Validity** is typically employed when a determined requirement about the quality of data is available, against which it is possible to compare some quality measure and declare if the data are valid.

**Confidence** is an attribute that may be elaborated from the continuous observation of sensor data, without the need for a quality requirement to be available. It is generally used when datasets are available and can be characterized in a probabilistic way, along with model fitting or threshold definition techniques, to yield continuous or multi-level confidence measures.

**Reliability** is a typical dependability attribute, expressing the ability of a system to provide the correct service (or the correct data, for that matter) over a period of time. The term data reliability in sensor networks is often considered when transmissions and/or communications may be subject to faults like omissions or a total crash.

**Trustworthiness** is mostly employed in connection with security concerns, namely when it is assumed that data can be altered in a malicious way. In the context of sensor networks, it characterizes the degree to which it is possible to trust that sensor data have not been tampered with and have thus the needed quality.

**Authenticity** is also used, in particular in a security context, but to express the degree to which it is possible to trust the claimed data origin. This is particularly important when the overall quality of the system or application depends on the correct association of some data to their producer.

This terminology does contain other terms, including other aspects of data quality that are implicit and briefly approached herein, such as timeliness, precision, tunability, completeness, usability, accuracy, throughput, affordability and reusability. We will also describe herein the diverse typologies of data quality and how to obtain a quality parameter, either for each individual sensor or for the global system, according to several studies. Therefore, in terms of applicability, we must differentiate single-sensor validity from multi-sensor fusion validity, when several sensors exist and sensor fusion can be applied.

In single-sensor situations, there are models or related information that allow reasoning about an individual sensor’s data quality without requiring other sensors’ data. The work in tried to identify faulty situations such as noise and outliers in chlorophyll concentration sensors deployed in lake water, by implementing different fault detection methods:

Rule-based methods that use expert knowledge about the variables that sensors are measuring to determine thresholds or heuristics with which the sensors must comply.

Estimation methods that define a “normal” behavior by considering spatial and temporal correlations from sensor data. A sensor reading is matched alongside its forecasted value to assess its validity.

Learning-based methods that define models for correct and faulty sensor measurements, using collected data for building the models.

## 3.5 System Used:

The following are the requirements we used to run proposed system.

**Hardware System Configuration:**

ANY CONTEMPORARY PC

Software System Configuration:

**Environment**: Python 3.6.1

**Operating System**: Windows/Linux and Android

**API:** Python API – PySpark

**Hardware Components:** Sensors

## 3.6 Feasibility Analysis

An important outcome of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resource and time. The different feasibilities that have to be analyzed are

* Operational Feasibility
* Economic Feasibility
* Technical Feasibility

**Operational Feasibility**

Operational feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

**Economic Feasibility**

Economic Feasibility or Cost-benefit is an assessment of the economic justification for a computer based project. As software was installed from the beginning & for lots of purposes thus there is no matter of cost for the project.

**Technical Feasibility**

Technical feasibility is the process of validating the technology assumptions, architecture and design of a product or project. The following are common types of technical feasibility.

**Input design**

Input Design plays a vital role in the life cycle of software development, it requires very careful attention of developers. The input design is to feed data to the application as accurate as possible. So inputs are supposed to be designed effectively so that the errors occurring while feeding are minimized. According to Software Engineering Concepts, the input forms or screens are designed to provide to have a validation control over the input limit, range and other related validations.

**Output design**

Output, or the response to the input, is just as - if not more - important as the input portion of the gameplay experience. There are usually only 2 types of output any standard device can perform, *visual output* and *auditory output*.

# 

# 

# CHAPTER 4

# 4 SYSTEM DESIGN

## 4.1 UNIFIED MODELING LANGUAGE

Unified Modeling Language (UML) is a general purpose modelling language. The main aim of UML is define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering.

UML helps software engineers, businessmen and system architects with modelling, design and analysis. The Object Management Group (OMG) adopted Unified Modelling Language as a standard in 1997. Its been managed by OMG ever since. International Organization for Standardization (ISO) published UML as an approved standard in 2005.

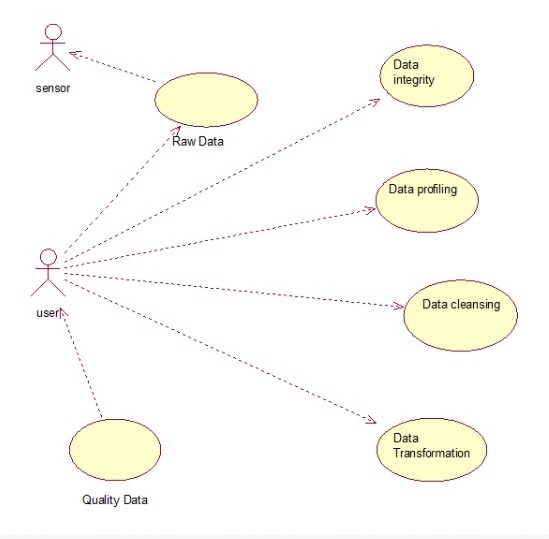
UML is linked with object oriented design and analysis. UML makes the use of elements and forms associations between them to form diagrams. Diagrams in UML can be broadly classified as:

**Structural Diagrams –** Capture static aspects or structure of a system. Structural Diagrams include: Component Diagrams, Object Diagrams, Class Diagrams and Deployment Diagrams.

**Behavior Diagrams –** Capture dynamic aspects or behavior of the system. Behavior diagrams include: Use Case Diagrams, State Diagrams, Activity Diagrams and Interaction Diagrams.

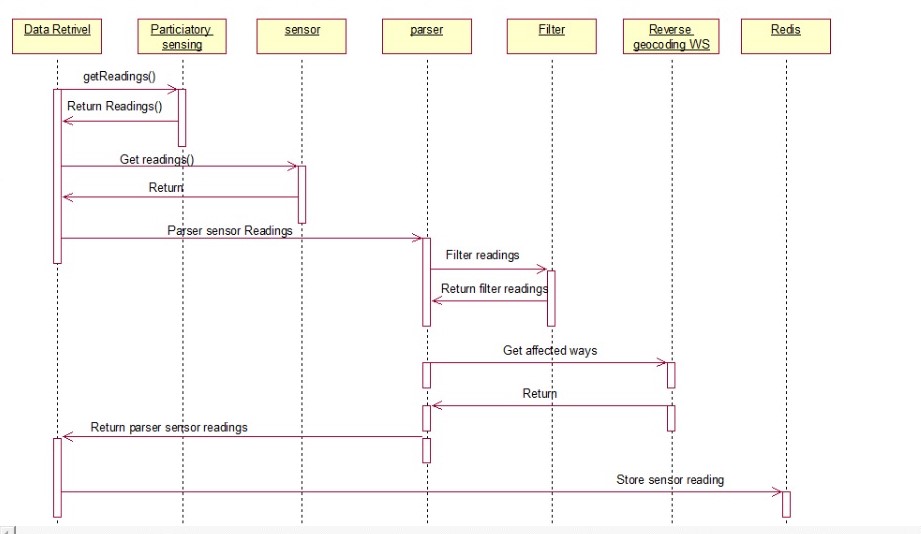
## 4.2 Use Case Diagram:

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

****

## 4.3 Sequence Diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

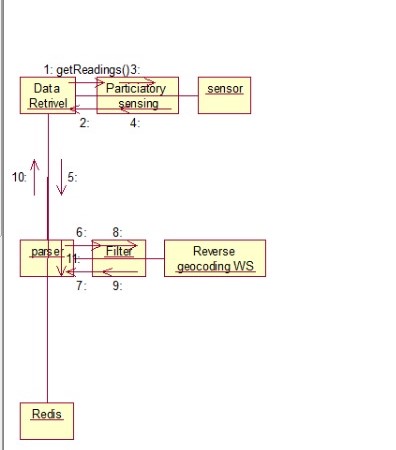
****

## 4.4 Collaboration Diagram

The second interaction diagram is the collaboration diagram. It shows the object organization as seen in the following diagram. In the collaboration diagram, the method call sequence is indicated by some numbering technique. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram.

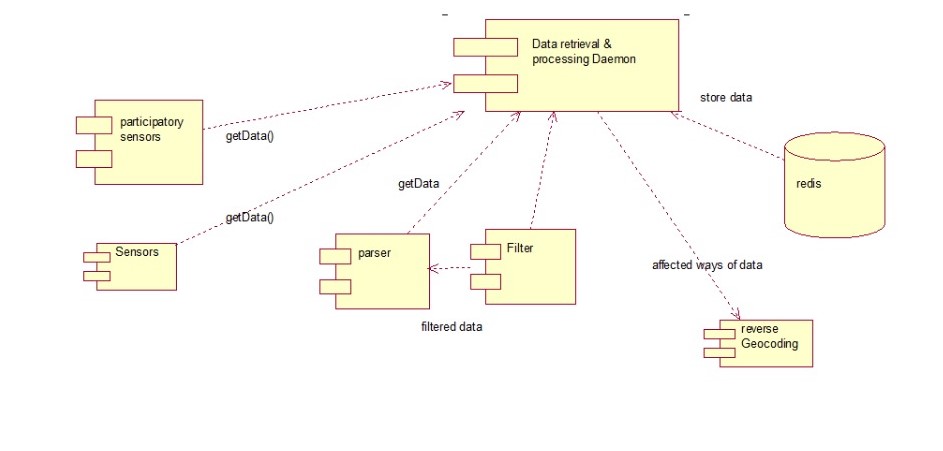
Method calls are similar to that of a sequence diagram. However, difference being the sequence diagram does not describe the object organization, whereas the collaboration diagram shows the object organization.

To choose between these two diagrams, emphasis is placed on the type of requirement. If the time sequence is important, then the sequence diagram is used. If organization is required, then collaboration diagram is used.



## 4.5 Component Diagram

Component diagrams are used to represent the how the physical components in a system have been organized. We use them for modelling implementation details. Component Diagrams depict the structural relationship between software system elements and help us in understanding if functional requirements have been covered by planned development. Component Diagrams become essential to use when we design and build complex systems. Interfaces are used by components of the system to communicate with each other.

****

**CHAPTER 5**

# 5 CODING

**Activity 1: Sample code**

**DQM\_Process\_on\_SensorData\_Using\_Pyspark\_&\_Seaborn.html**

#### **Importing the required libraries for the dataset** #####################

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

from sklearn.preprocessing import Imputer

from sklearn.preprocessing import LabelEncoder,OneHotEncoder

from pyspark.sql.window import Window

from pyspark.sql import SQLContext

from pyspark.sql import Row

import pyspark.sql.functions as F

from pyspark.sql.window import Window

*#****Initializing PySpark***

**from** **pyspark** **import** SparkContext, SparkConf

*# #Spark Config*

conf = SparkConf().setAppName("sample\_app")

sc = SparkContext(conf=conf)

*########################* ***Reading a JSON file of sensor data using pandas DataFrame*** *#############################*

*#/FileStore/tables/9p83uiou1504863670100/sensor\_final\_dataset.json*

df= pd.read\_csv('E:\Desktop\sensor\_data.csv')

print ("############## **DESCRIBE DATASET** ###################")

print (df.describe())

print ("############## **DATA TYPES** ###################")

print (df.dtypes)

############## **DESCRIBE DATASET** ###################

ambient\_temperature humidity photosensor radiation\_level

count 5962.000000 5982.000000 5982.000000 5991.000000

mean 21.169832 79.702565 801.903395 199.604407

std 5.292841 3.835437 50.064696 1.965327

min 1.473000 66.235300 609.380000 194.000000

25% 17.850000 77.140600 767.500000 198.000000

50% 21.000000 79.660350 800.080000 200.000000

75% 24.457500 82.242900 836.500000 201.000000

max 44.450000 92.416500 1020.200000 206.000000

lat lng dewpoint

count 5987.000000 5990.000000 5955.000000

mean 33.944177 -93.140172 17.108798

std 4.648238 9.669680 5.343678

min 27.833300 -105.327200 -2.734880

25% 27.833300 -105.327200 13.840040

50% 34.951300 -92.380900 17.025600

75% 39.064600 -81.717000 20.422660

max 39.064600 -81.717000 40.101140

############## **DATA TYPES** ###################

ambient\_temperature float64

humidity float64

photosensor float64

radiation\_level float64

sensor\_uuid object

sensor\_id object

sensor\_name object

lat float64

lng float64

datetime object

dewpoint float64

dtype: object

*#****Compute Correlation***

print(df.corr())

**import** **seaborn** **as** **sns**

*######* ***correlation matrix*** *########*

plt.clf()

*# Set up the matplotlib figure*

f, ax = plt.subplots(figsize=(10, 8))

*# Compute the correlation matrix*

corr = df.corr()

*# Draw the heatmap with the mask and correct aspect ratio*

sns.heatmap(corr, mask=np.zeros\_like(corr, dtype=np.bool), cmap=sns.diverging\_palette(250, 20, sep=20,as\_cmap=**True**),square=**True**, ax=ax)

plt.title('Sensor Data Correlations Heat Map', fontsize=15, fontweight='bold')

plt.gcf().subplots\_adjust(bottom=0.15)

plt.tight\_layout()

plt.show()

values = df.isnull().sum()

nullvalues = pd.DataFrame({'attribute\_name':values.index, 'nullcount':values.values})

line\_dataframe = sqlContext.createDataFrame(nullvalues[["attribute\_name","nullcount"]])

line\_dataframe.show()

*#* ***Split into sets with known and unknown ambient\_temperature******values***

**from** **sklearn.linear\_model** **import** LinearRegression,LogisticRegression

df\_v2 = df\_v1[["ambient\_temperature","humidity","photosensor","radiation\_level"]]

knownTemperature = df\_v2.loc[(df\_v1.ambient\_temperature.notnull())]

unknownTemperature = df\_v2.loc[(df\_v1.ambient\_temperature.isnull())]

*# All ambient\_temperature values are stored in a target array*

Y = knownTemperature.values[:, 0]

*# All the other values are stored in the feature array*

X = knownTemperature.values[:,1::]

*# Create and fit a model*

linear\_regression = LinearRegression()

linear\_regression.fit(X, Y)

*# Use the fitted model to predict the missing values*

predictedTemperature = linear\_regression.predict(unknownTemperature.values[:, 1::])

*# Assign those predictions to the full data set*

df\_v1.loc[ (df\_v1.ambient\_temperature.isnull()), 'ambient\_temperature' ] = predictedTemperature

df\_v1.show()

*#################* ***Encoding a categorical data using LabelEncoder*** *#######################*

*####* ***Converting a non\_numeric value of sensor\_name into numeric data*** *######*

labelencoder\_X=LabelEncoder()

labelencoder\_X.fit(df\_v1.ix[:,6])

list(labelencoder\_X.classes\_)

df\_v1.ix[:,6] = labelencoder\_X.transform(df\_v1.ix[:,6])

*#### Converting a non\_numeric sensor name into numeric data ######*

labelencoder\_y=LabelEncoder()

labelencoder\_y.fit(df\_v1.ix[:,4])

list(labelencoder\_y.classes\_)

df\_v1.ix[:,4] = labelencoder\_y.transform(df\_v1.ix[:,4])

*#### Converting a non\_numeric value of sensor ID into numeric data ######*

labelencoder\_z=LabelEncoder()

labelencoder\_z.fit(df\_v1.ix[:,5])

list(labelencoder\_z.classes\_)

df\_v1.ix[:,5] = labelencoder\_z.transform(df\_v1.ix[:,5])

*#* ***plot feature importance using built-in function***

**from** **numpy** **import** loadtxt

**from** **xgboost** **import** XGBClassifier

**from** **xgboost** **import** plot\_importance

**from** **matplotlib** **import** pyplot

*# split data into X and y*

X = df\_v1.ix[:,[0,1,2,3,4,5,6,7,10]]

Y = df\_v1.ix[:,[10]]

plt.clf()

*# fit model no training data*

model = XGBClassifier()

model.fit(X, Y)

*# plot feature importance*

plot\_importance(model)

plt.gcf().subplots\_adjust(bottom=0.15)

plt.tight\_layout()

plt.show()

print ("humidity\_mean:",df\_v1['humidity'].mean())

print ("ambient\_temperature:" ,df\_v1['ambient\_temperature'].mean())

print ("photosensor:" ,df\_v1['photosensor'].mean())

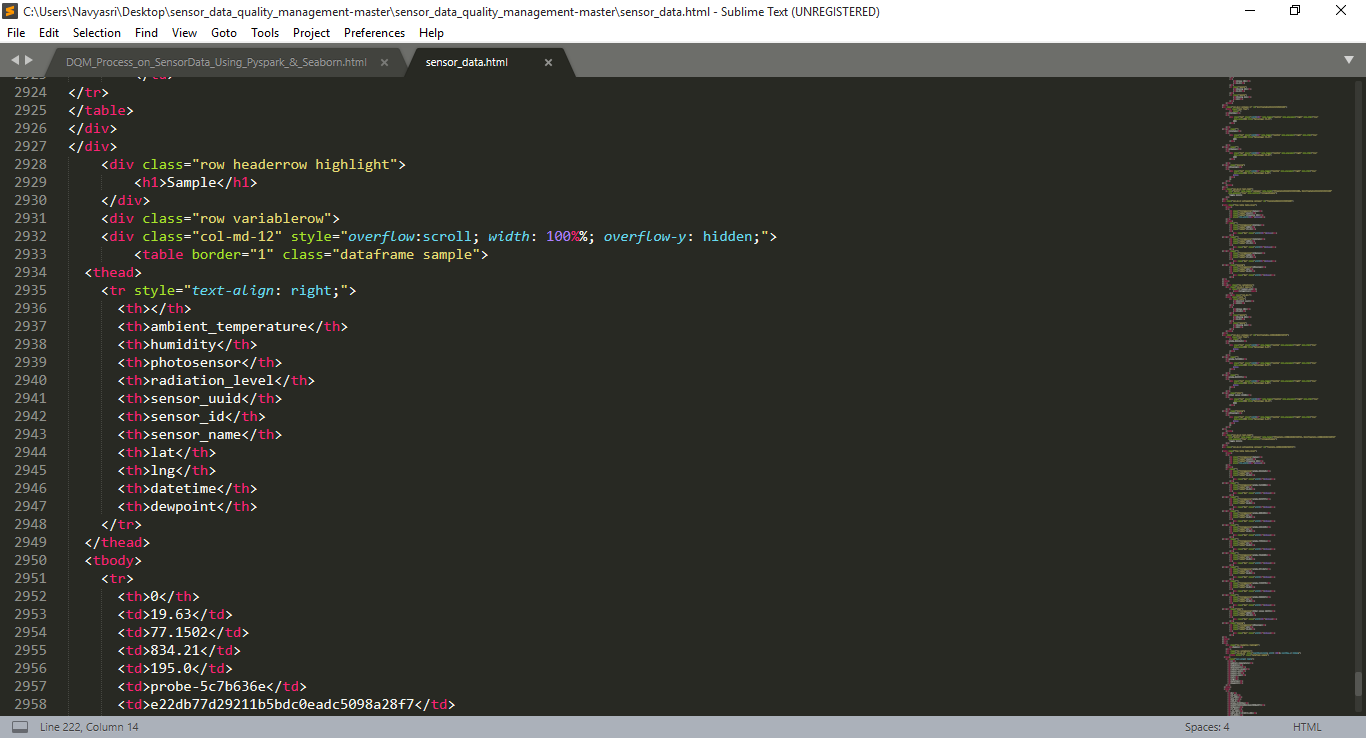
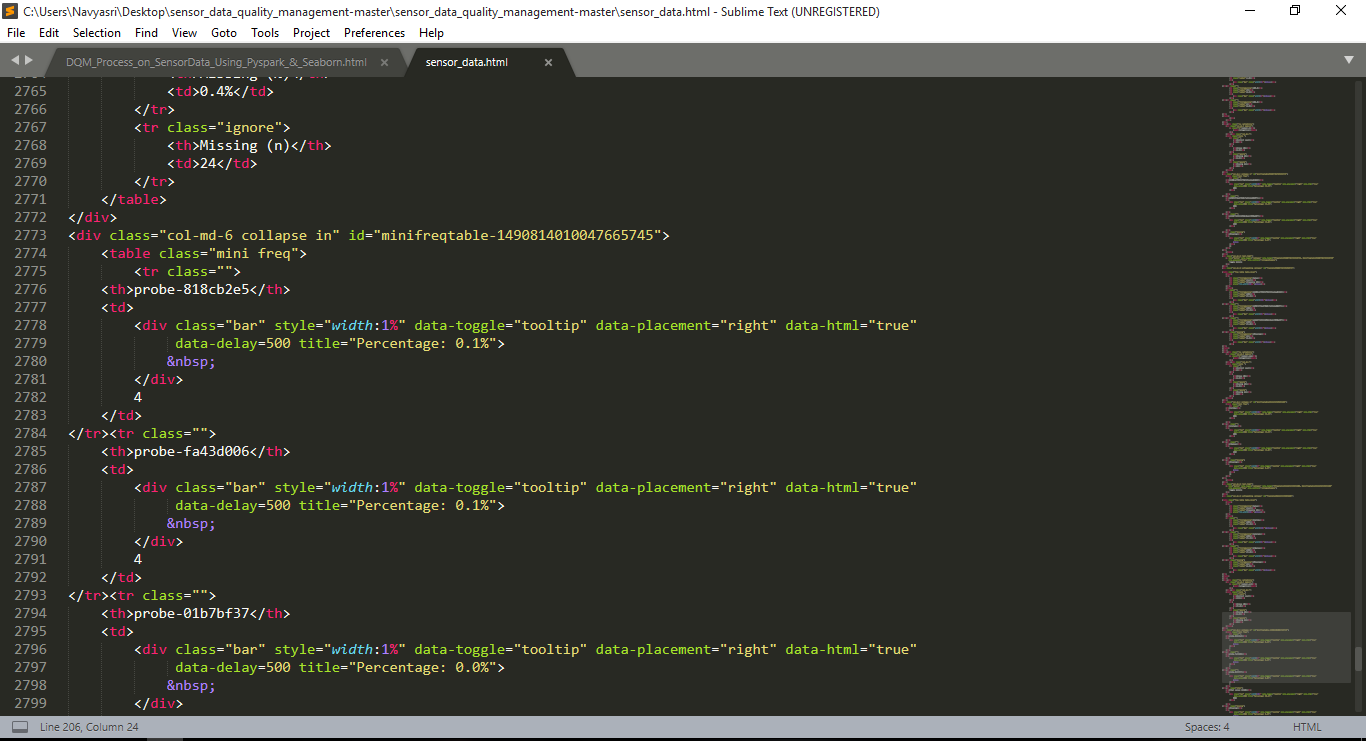
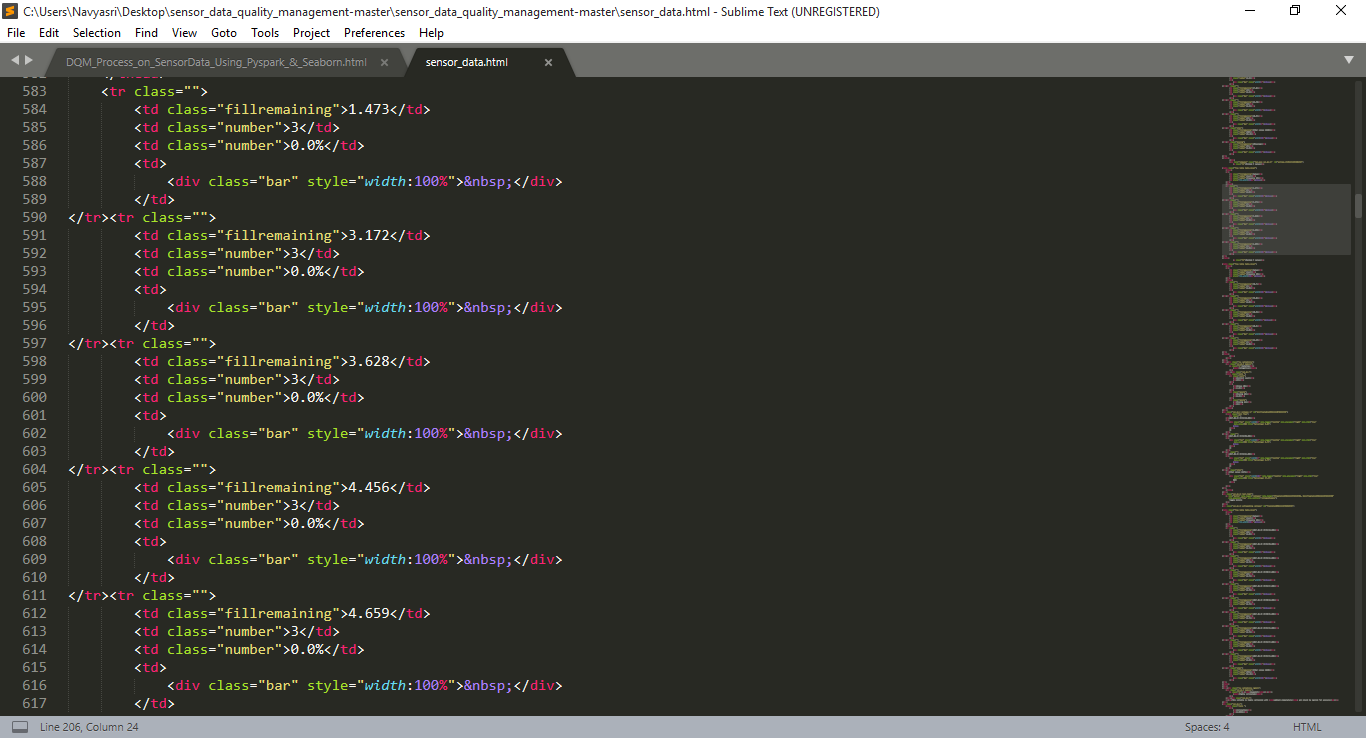
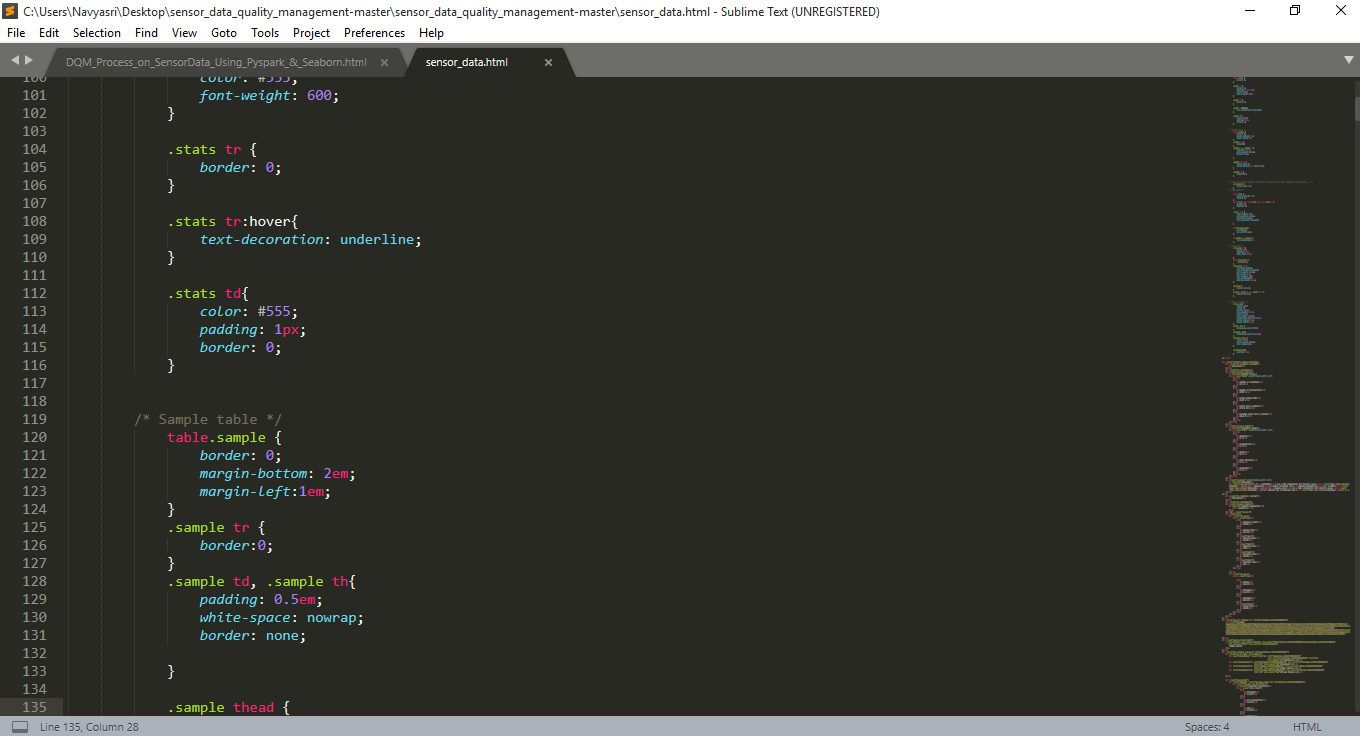
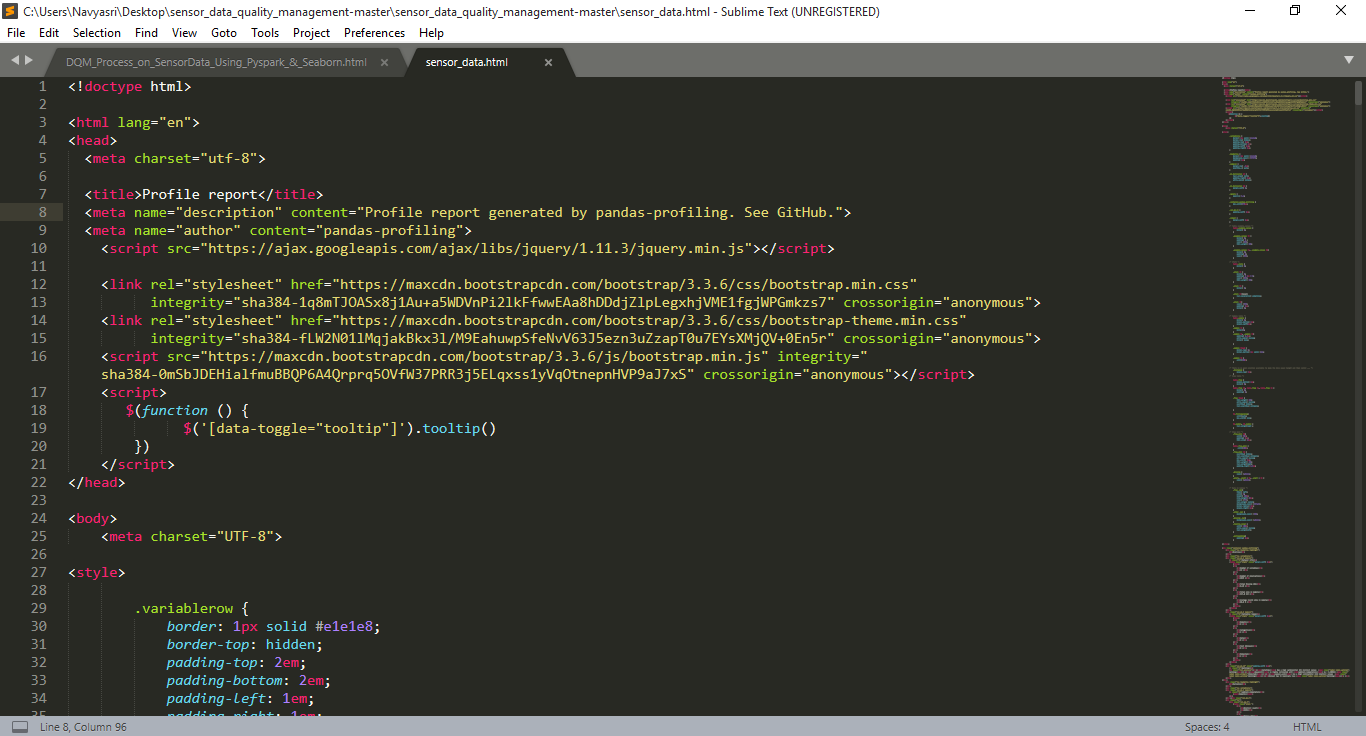
print ("radiation\_level:" ,df\_v1['radiation\_level'].mean())

print ("dewpoint:" ,df\_v1['dewpoint'].mean())

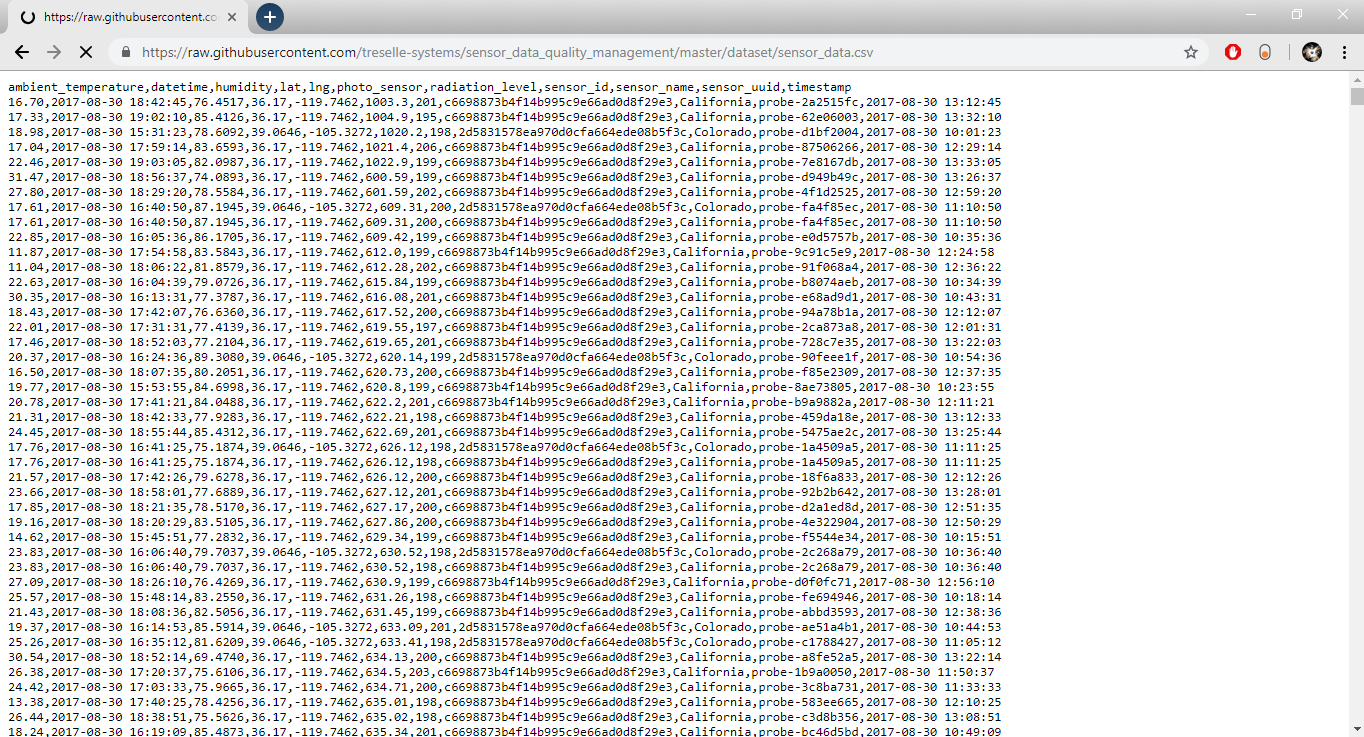
"Displaying the mean values of humidity,ambient temperature , photosensor,radiation level and dew point "

**Activity 2:**

**sensor\_data.html**

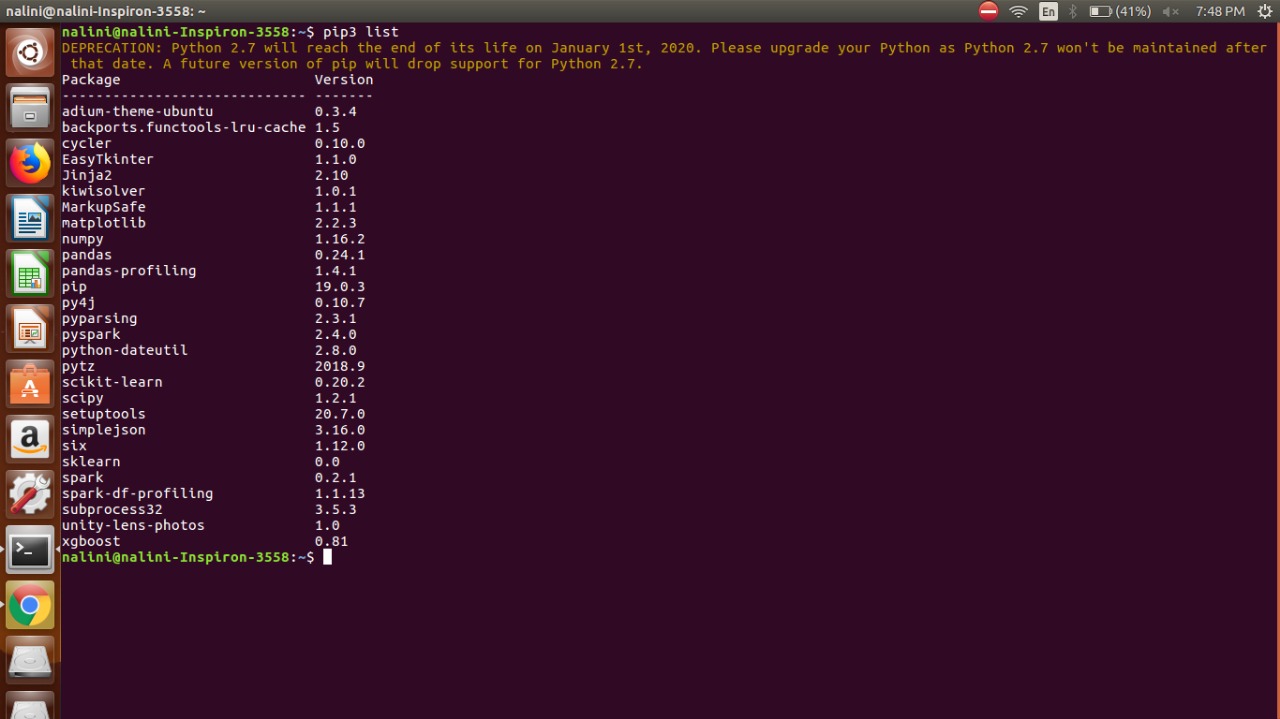


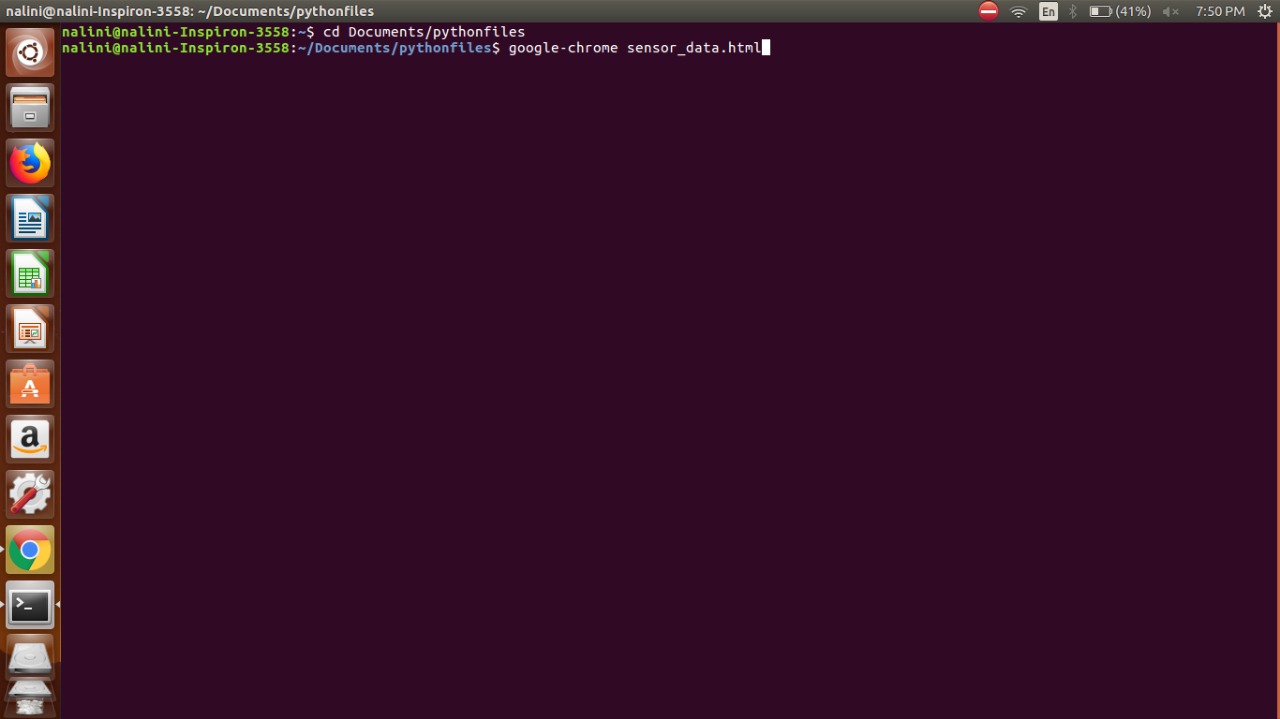
**Dataset:**

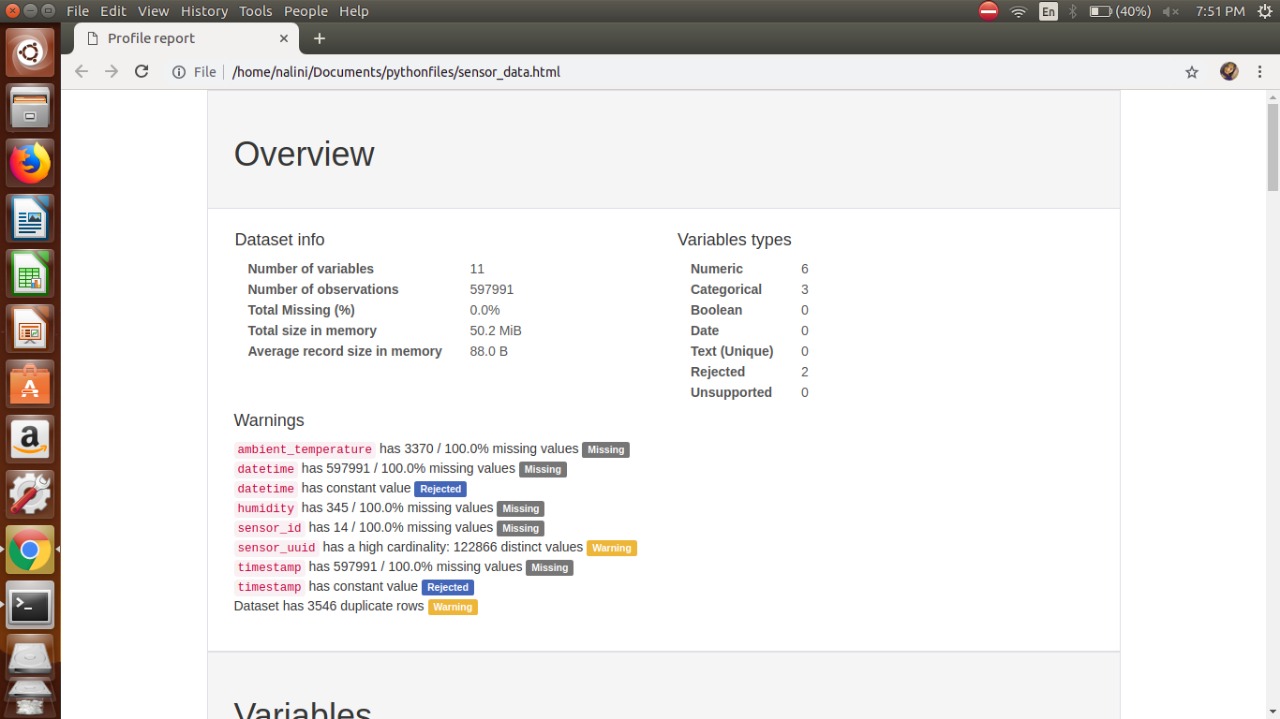


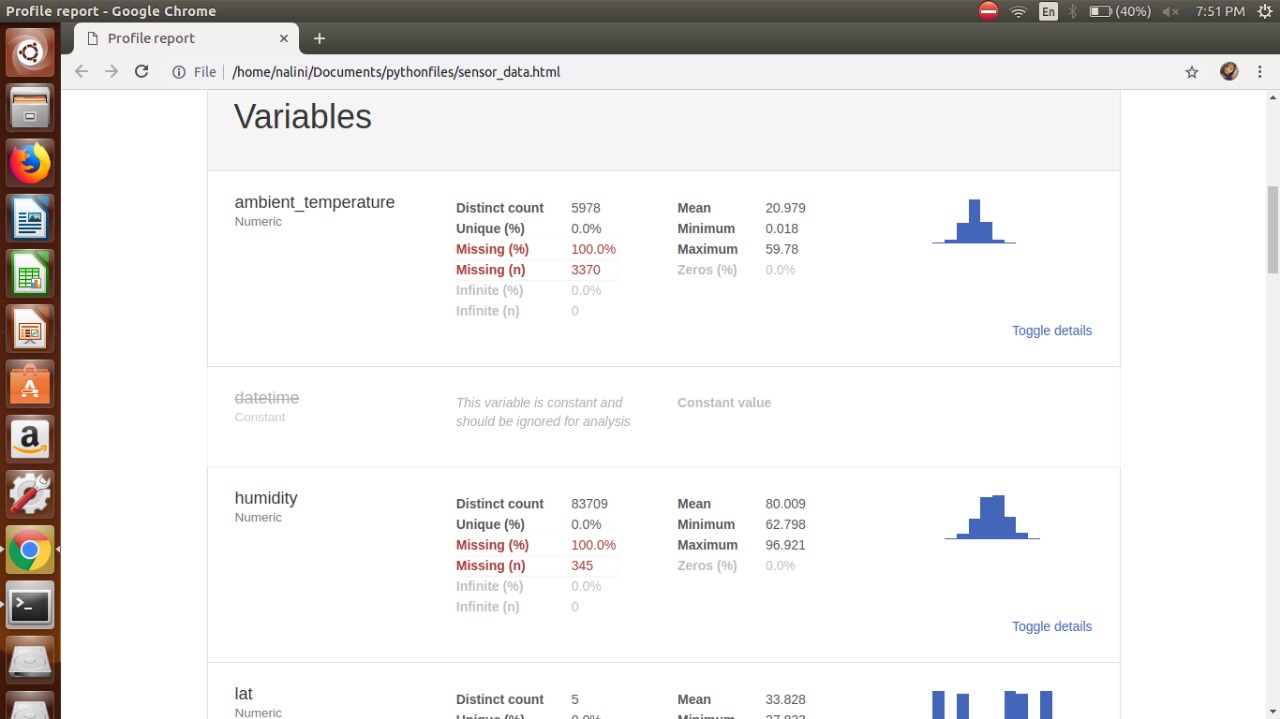
**CHAPTER 6**

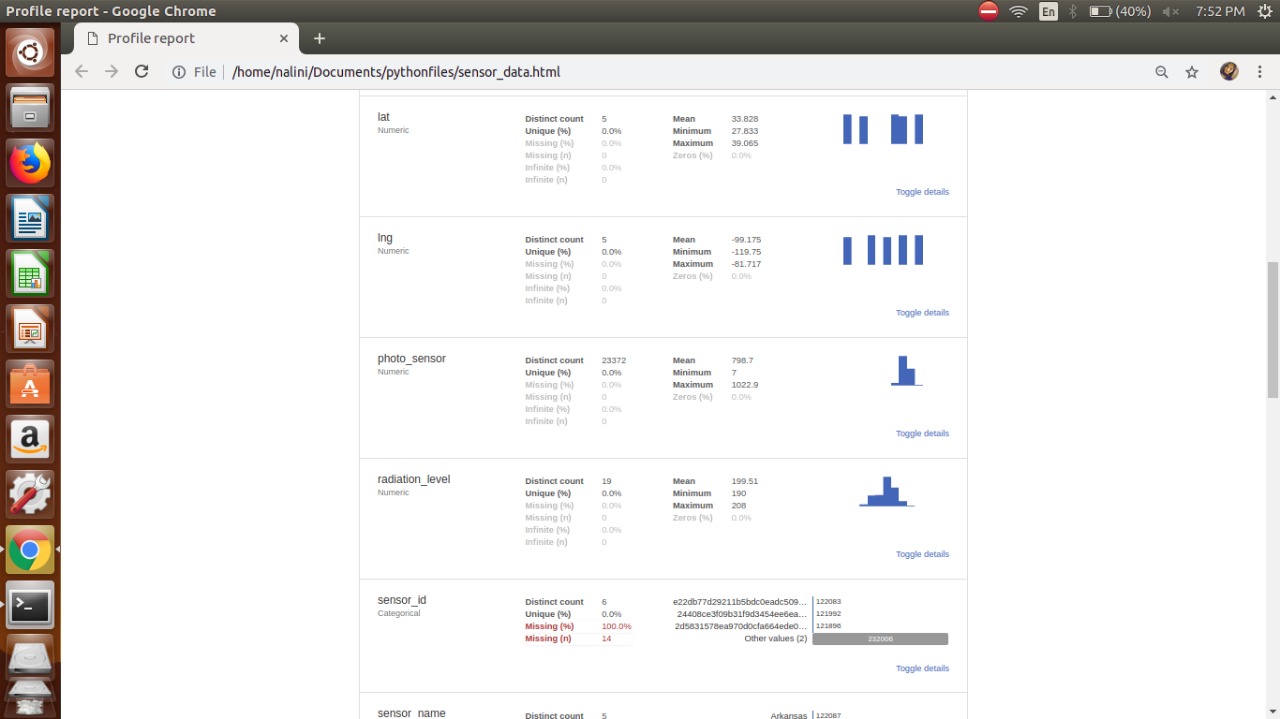
# 6 OUTPUT SCREENS

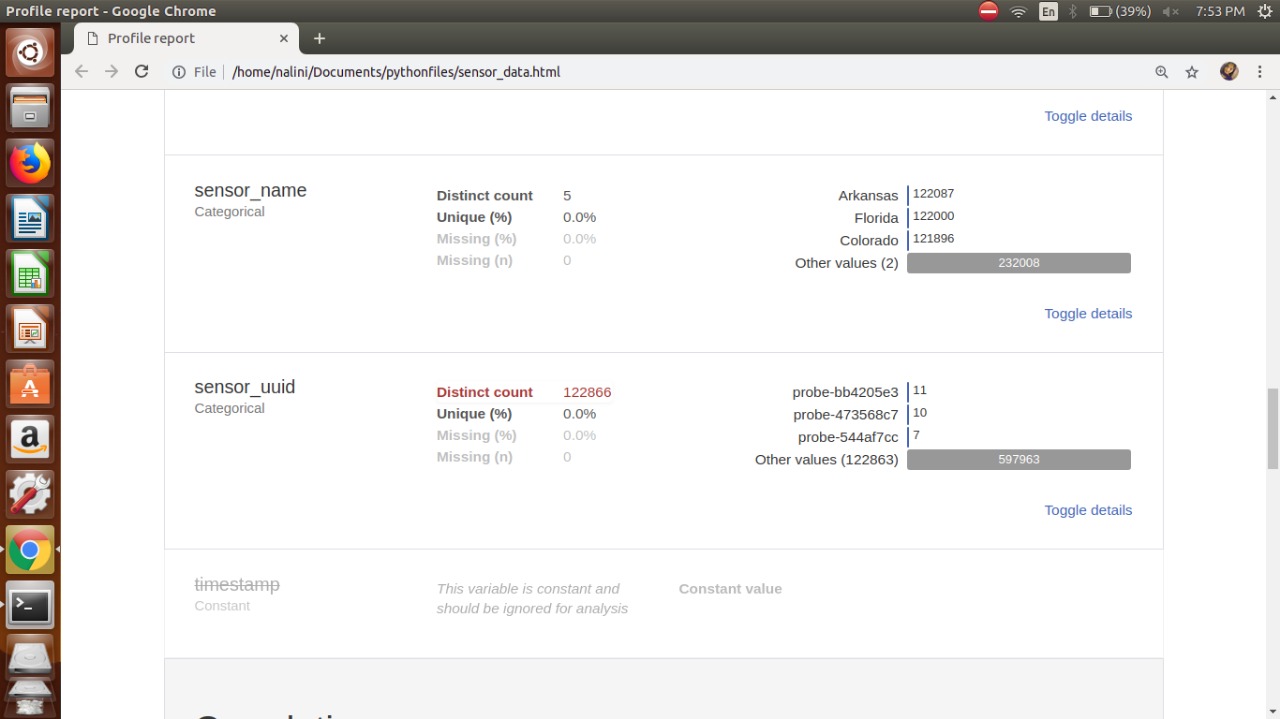


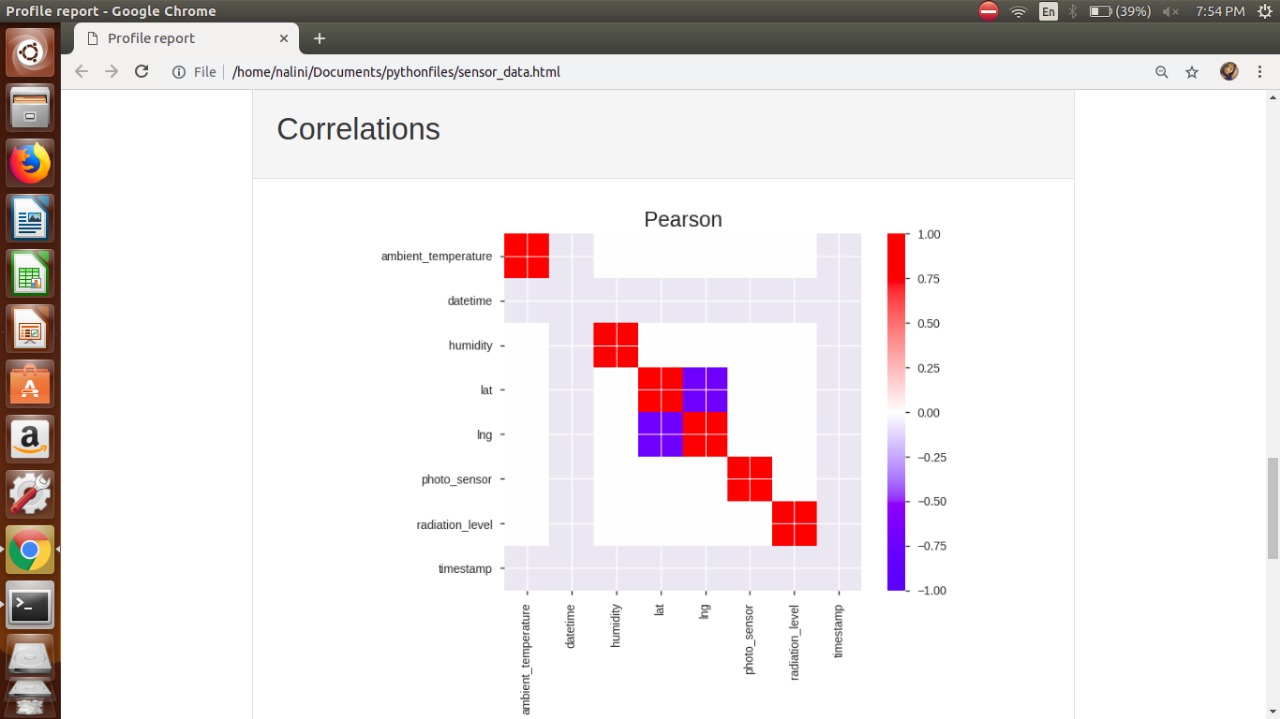


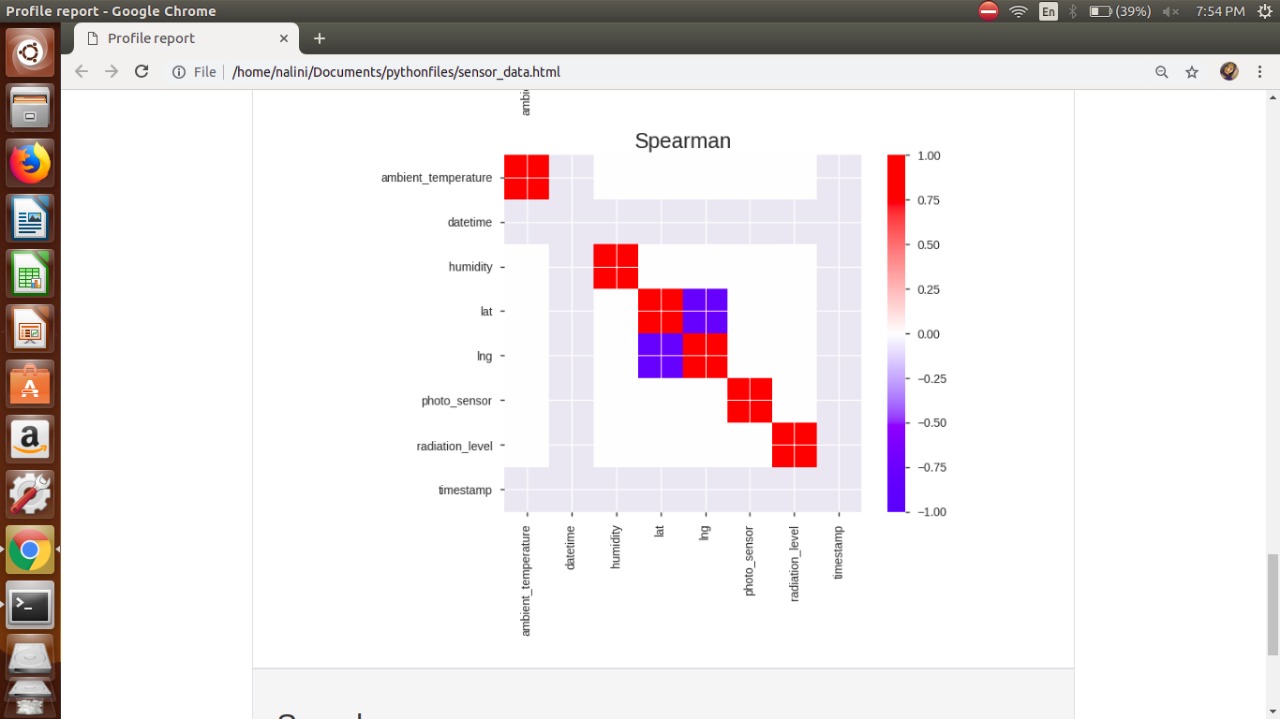
****

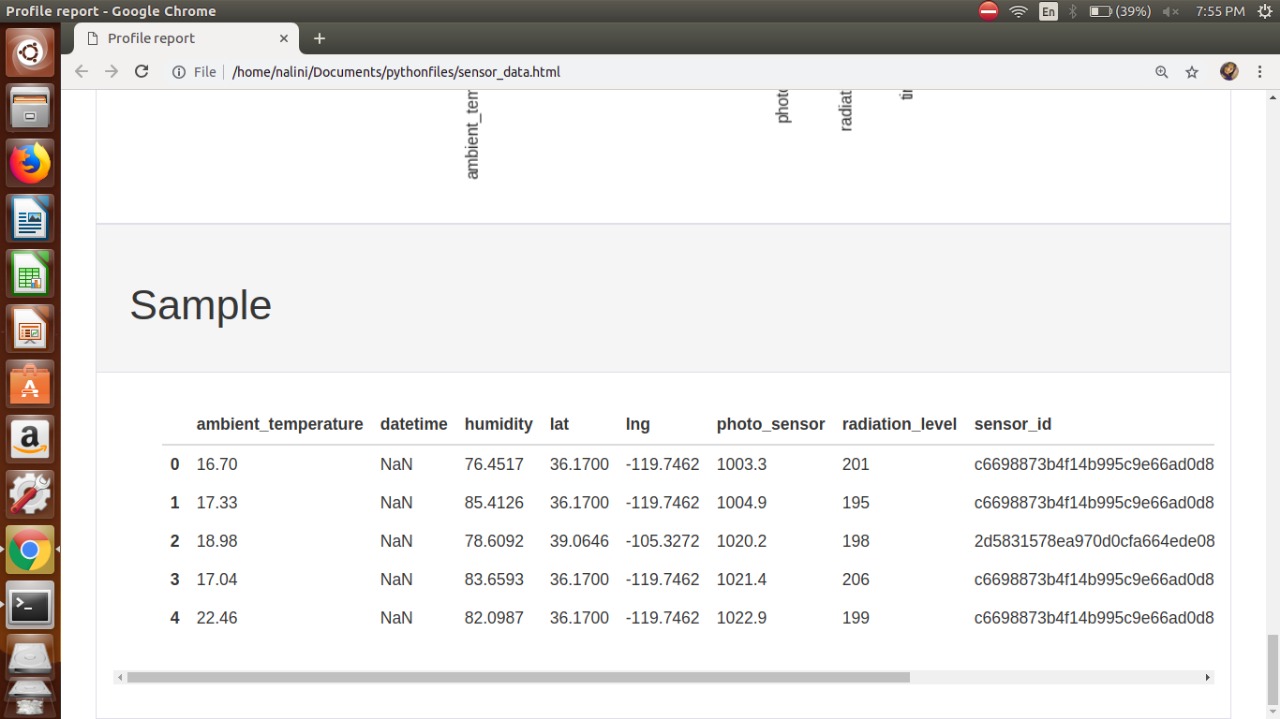
****

****

****

****

****

****

**CHAPTER 7**

# 7 IMPLEMENTATION

## 7.1 Module:

A module is a separate unit of software or hardware. Typical characteristics of modular components include portability, which allows them to be used in a variety of systems, and interoperability, which allows them to function with the components of other systems. The term was first used in architecture.

1) In computer programming, especially in older languages such as PL/1, the output of the language compiler was known as an *object module* to distinguish it from the set of *source* language statements, sometimes known as the *source module*. In mainframe systems such as IBM's OS/360, the object module was then linked together with other object modules to form a *load module*. The load module was the executable code that you ran in the computer.

*Modular programming* is the concept that similar functions should be contained within the same unit of programming code and that separate functions should be developed as separate units of code so that the code can easily be maintained and reused by different programs. Object-oriented programming is a newer idea that inherently encompasses modular programming.

2) In computer hardware and electronics, a module is a relatively compact unit in a larger device or arrangement that is designed to be separately installed, replaced, or serviced. For example, a single in-line memory module is a unit of random access memory (RAM) that you can add to a personal computer.

## 7.2 Module Description:

Module 1

Retrieving the raw data from sensors

As sensors are the major part which detects and produce required data, Sensed data are collected and pre-processed to be transferred (directly or through a gateway) to the data processing and management center for further analysis. Collected data are processed, managed and stored according to users and application requirements.

Unfiltered, raw data, with no quality control tests applied and no data qualifiers (flags) applied - Typically, these are original data streams that are not published but that should be preserved. Data quality flags are not assigned. Conversion of raw measurement values to more meaningful units may be acceptable, e.g., thermocouple table conversions of millivolts to degrees C.

Module 2

We perform different use cases to get the quality data.

**Synopsis:**

* Data Integrity
* Data Profiling
* Data Cleansing
* Data Transformation

**Data Integrity:**

Data integrity is the process of guaranteeing the quality of the data in the database.

* Analyzed input sensor data
* Validated source metadata
* Populated relationships for an entity

**Data Profiling**

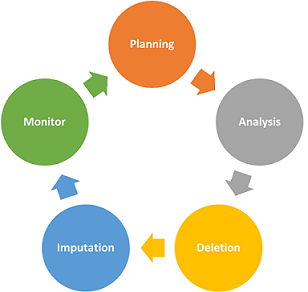
Data profiling is the process of discovering and analyzing enterprise metadata to discover patterns, entity relationships, data structure, and business rules. It provides statistics or informative summaries of the data to assess data issues and quality.

Few data profiling analyses include:

* **Completeness analysis**: Analyze frequency of attribute population versus blank or null values.
* **Uniqueness analysis**: Analyze and find unique or distinct values and duplicate values for a given attribute across all records.
* **Values distribution analysis**: Analyze and find the distribution of records across different values of a given attribute.
* **Range analysis**: Analyze and find minimum, maximum, median, and average values of a given attribute.
* **Pattern analysis**: Analyze and find character patterns and pattern frequency.

**Data Cleansing:**

Data cleansing is the process of identifying incomplete, incorrect, inaccurate, duplicate, or irrelevant data and modifying, replacing, or deleting dirty data.



Analyzed the number of null (NaN) values in the dataset using the command df.isnull().sum().

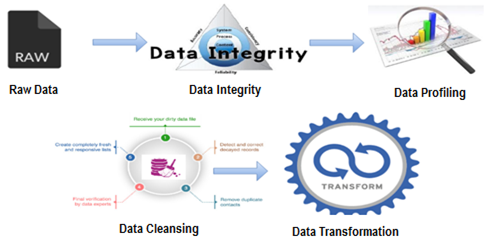
**Data Transformation:**

Data transformation deals with converting data from the source format into the required destination format.

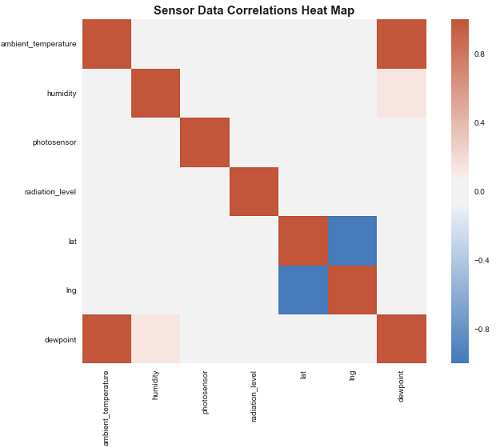


### Module 3

Here the quality data is produced by performing all the test cases.



The process follows the life cycle of performing use cases, and finally the status of the data is represented in the graphical format.



The graphical representation of the quality data after performing all the use cases.

**CHAPTER 8**

# 8 TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .You cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**CHAPTER 9**

# 9 CONCLUSION

In my proposed model design, we underline the importance of evaluating and communicating information quality in emerging applications like environmental monitoring. As our project shows, several external factors may impact the quality of sensor data and thus directly impact users’ decision making. Hence, we tackle this issue with an approach focalized on the definition and evaluation of sensor data quality in monitoring applications. The technologies (pyspark and seaborn) used to fasten the process and produce best quality data. This technologies plays a major role in producing the quality data. We propose a sensor data quality model based on sensor data specificities and attempting to formalize sensor data quality properties (categories, criteria and indicators). This contribution is mainly supported by mechanisms and techniques as metadata to process and manage the quality of information. We also propose in this paper, how to provide users and applications with quality information: visually representing quality indicators and producing quality reports. In order to support our approach, a web-based interface for sensor data quality discovery in real-time has been implemented.

**CHAPTER 10**

# 10 BILIOGRAPHY

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6165119/

https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/spark-data-exploration-modeling

http://wiki.esipfed.org/index.php/Sensor\_Data\_Quality

https://perso.liris.cnrs.fr/sylvie.servigne/2013-IJAEISCGSS.pdf

https://www.researchgate.net/publication/319866075\_DATA\_QUALITY\_IN\_REMOTE\_SENSING

Data Quality Management with Semantic Technologies by Christian Fürber (Author)

Managing Data Quality by Hansen Mark D

Batini, C., & Scannapieco, M. (2006). Data Quality: Concepts, Methodologies and Techniques. Berlin Heidelberg: Springer-Verlag