UPPSALA
UNIVERSITET

IT 16 016
Examensarbete 30 hp
Mars 2016

Data analysis for predicting air pollutant concentration in Smart city Uppsala

Varun Noorani Subramanian

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Data analysis for predicting air pollutant concentration in Smart city Uppsala

Varun Noorani Subramanian

Pollution concentrations in urban areas are primarily from vehicular exhaust, factories, and small scale industries. Recent studies conducted by the Swedish Meteorological and Hydrological Institute (SMHI) says that 3000-5000 premature deaths [2] occur every year as a result of inhaling a high level of pollution concentrations like PM10, PM2.5, CO, Nitrogen Oxides (NO+NO2). A sustainable lifestyle in an urban city-like environment is thus possible only through smart city style urban management.
Foreseeing the future, the Uppsala Municipality along with the help of IBM, Ericsson, and the Uppsala University has initiated a smart city project in Uppsala. The thrust of this initiative would be deploying pollution detection sensors all over Uppsala city and monitoring pollution concentrations continuously throughout the day. The data collected will then be passed to a knowledge discovery process that would forecast pollution concentration for the future, and will be presented in a user-friendly format in real-time using an Android application. This application will provide users with real-time pollution concentration level along with the predicted value of the location thereby helping in raising awareness of its causes and consequences.
The main focus of this thesis will be in exploring the suitable data mining technique that will help in better forecasting of the pollution concentration. In addition to the data model, it also focuses on the design and implementation of an Android application targeted towards the people of Uppsala community.

Handledare: Edith Ngai
Ämnesgranskare: Philipp Rümmer
Examinator: Justin Pearson
IT 16 016
Tryckt av: Reprocentralen ITC

Acknowledgments

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Edith Ngai for giving me the opportunity to work on this project. Her advice and feedback throughout the project has helped in the successful completion of the project. I would also like to thank the Uppsala Municipality for being willing to take the time to meet with us and provide us with the inputs for this project.

I am also thankful to my reviewer Prof. Philipp Rümmer for his constant feedback and ideas throughout the project.

Last but not the least; I would like to thank God, my parents, brother and my friends for supporting me spiritually and mentally throughout the thesis and my life in general. Without them, I would never have accomplished this feat.

Table of Contents

List of Figures

List of Tables

Code Snippets

Chapter 1

Introduction

This chapter introduces the problem and gives an overview of the thesis. This is followed by the thesis objectives and the structure of the report.

1.1 Background

The population of the world is on the rise and by 2020 is predicted to reach more than 7 billion [1]. Currently, the population in Sweden is around 10 million and is expected to increase to 14 million by 2020. The increasing population is bound to lead to a significant rise in the number of vehicles on the road that in turn will lead to higher emissions of harmful particulates into the atmosphere. The common particulates emitted into the atmosphere are PM10, PM2.5, CO, Nitrogen Oxides (NO+NO2) and Ozone. Inhaling these particulates will affect the normal lung development and lead to respiratory problems such as asthma, heart diseases, etc. Recent studies in Sweden performed by the Swedish Meteorological and Hydrological Institute (SMHI) have found that 3000-5000 premature deaths occur every year because of inhaling particulate matters present in the atmosphere [2]. Also, SMHI pointed out that it would be difficult to reduce the pollution levels unless the emissions caused by on road traffic are restricted [3].

A possible solution to reduce the pollution concentration is by creating awareness to the people on the causes and the harmful effect of the pollutant concentrations. The technologies available today play a vital role in our day to day life and the dependence on it has greatly increased over the years. Therefore, incorporating the available technologies for creating awareness to the people is one of the possible solutions. One such technology that has been widely used in pollution-related projects is the use of pollution sensors that have the capacity to detect and distinguish each pollution particulate separately.

In recent years, the smart cities initiative is on the rise to mitigate the effect of pollution. This initiative comprises of many projects involved in protecting the city environment and also in reducing the pollution level of the city.

The *EKOBUS* project in Serbia involved sensors being placed on the rooftop of the buses to give real-time pollution data of the bus route [4]. This was done in collaboration with Ericsson. Another smart city project named *RESCATAME* was carried out in Salamanca, Spain where the sensors were placed in various parts of the city to identify various pollution sources that helped in developing a traffic control system [5]. All this has paved the way for implementing a similar smart city project in Uppsala, Sweden in association with the Uppsala University, Uppsala Municipality, Ericsson, and IBM.

The thesis is part of a bigger project where the following aspects will be implemented over the course of time in Uppsala. The following are the primary modules for the project:

- Wireless Sensors

  Sensors have been widely used for measuring temperature, pressure, and several other parameters. The addition of wireless capabilities to sensors has increased their utility multifold and hence has led to the creation of several wireless sensor mesh networks. A Wireless Sensor Network (WSN) consists of nodes that are connected to the sensors and pass the collected data through the network [6]. Similar wireless sensors are being developed at Uppsala University for detecting and measuring the temperature, humidity and AQI (Air Quality Index) level of the PM2.5 pollutant concentration. PM2.5 is the finely suspended particulates present in the air which includes dust, smoke, and liquid droplets, etc and its primary source is from vehicle exhaust, burning plants, and metal processing [45]. The sensors developed will be deployed in the city and will be used for collecting data from various parts of the city. In addition to detecting PM2.5, other pollution detection sensors will also be added over the course of time.

- Data Analysis and Forecasting

  Data analysis is the method where the data is pre-processed, transformed, and modeled with the goal of finding out information and drawing conclusions in a decision-making process [7]. The data collected from the sensors will be passed through a Knowledge Discovery (KD) process where the pollution concentrations (PM2.5) pattern will be identified and will be later used for the forecasting.

- Android Application

  Android is an open source software designed for handheld devices such as tablets, mobile phones, etc. It is based on a UNIX operating system and is written in Java programming language using the Android Software Development Kit (SDK). An Android application will be developed that will provide the users with real-time pollution concentration of a location and also will provide the users with hourly forecasted pollution concentration. Also, the application will suggest the users with navigation for the less polluted route.

In this smart city initiative, the wireless sensors will be deployed in various parts of the city which will detect the pollution concentration of PM2.5. The collected data will be passed to a Knowledge Discovery (KD) process that will be used for creating a data model for forecasting the pollution concentration of PM2.5. Finally, an Android application will act as a medium for the users to provide the users with real-time pollution concentration from various locations.

This project mainly focuses on creating the data model for forecasting and also in developing the user interface for the Android application. The developments of the wireless sensors are being done by another group of students from Uppsala University closely working with Upwis AB.

1.2 Research Questions

The research questions that can be derived and will be answered in this thesis are:

- What are the existing learning algorithms used for forecasting the pollution concentration?
- What are the possible external factors that affect the concentration of the pollution?
- How will the model perform when noise is present or induced during and after data collection?
- What will be the performance of the model if its functionality is transferred to another pollutant concentration?
- How user-friendly is the application for a common person?

- Can an alternate vehicle navigation route be provided to the users based on current pollution levels?

1.3 Thesis Objective

The overall aim of this project is to create a learner algorithm that will be able to predict the hourly pollutant concentration. Also, an Android application will be developed that will provide the users about the real-time pollution concentration of PM2.5 along with the hourly forecasted value of the pollutant concentration from the learner algorithm. The Android application will also suggest information of the less polluted navigation route between source and destination based on the Google driving navigation

1.4 Limitations

The project has been completed according to its requirements, but there were some unavoidable limitations. The sensors that were supposed to be developed were not completed on time; therefore, the real-time data were not used in developing the data model. The datasets used for building the data model were from 2012 to 2014 rather than 2015 because of unavailability of the latest dataset. Also, the smart route in the Android application makes use of real-time pollution concentration for suggesting the users with the less polluted route. Because of the delay in the deployment of the sensors, only a theoretical result has been presented for the same.

1.5 Thesis structure

The document is structured as follows: Chapter 2 introduces the readers to detailed background knowledge of Air Quality Index (AQI) followed by the data mining and Android application platform. Chapter 3 talks about the existing and related works similar to the data model and the Android application. The system architecture is introduced in Chapter 4, followed by the implementation of the data mining module and Android application module in Chapter 5 and Chapter 6 respectively. Finally in Chapter 7, the results are presented for the conducted experiments followed by the conclusion and future work.

Chapter 2

Background

In this chapter, a brief introduction to the Air Quality Index (AQI) and the system architecture of the data mining and Android application module are discussed.

2.1 Air Quality Index

Air Quality Index (AQI) is an index that provides the public with the level of pollution associated with its health effects. The AQI focuses on the various health effects that people might experience based on the level and hours of exposure to the pollutant concentration [17]. The AQI values are different from country to country based on the air quality standard of the country. The higher the AQI level greater is the risk of health related problems. To understand the different classifications of AQI, consider the following table:

Table 2-1: AQI Level Classification for Europe [17]

Pollution Level	Index Value	Index Color
Good	0-50	Green
Moderate	51-100	Yellow
Unhealthy for Sensitive Groups	101-150	Orange
Unhealthy	151-200	Red
Very Unhealthy	201-300	Purple
Hazardous	301-500	Maroon

There are five different categories, and each category corresponds to different health concerns.

- Good, AQI is 0 – 50

  In this category, the air quality is "*Satisfactory*" meaning it does not possess any risk to human health.

- Moderate, AQI is 51 – 100

  In this category, the air quality is considered "*Acceptable*." However, people sensitive to Ozone may experience certain respiratory problems otherwise, it only possess moderate health concern.

- Unhealthy for Sensitive Groups, AQI is 101 – 150

  In this category person with lung diseases, elderly people and children are at a greater risk from exposure to Ozone. Also, people suffering from heart and lung diseases possess a greater risk of the presence of particulates in the atmosphere.

- Unhealthy, AQI is 151– 200

  In this category, every person will start experiencing some adverse effects and members of the sensitive group will be affected even more.

- Very Unhealthy, AQI is 201 – 300

  In this category, everyone will start experiencing serious health effects.

- Hazardous, AQI is 301 – 500

  This is the highest possible level, with the entire population suffering from serious health effects.

The AQI is calculated from the pollutant concentration data using the following formula [18]:

$$\boldsymbol{I_P = \frac{I_{Hi} - I_{Lo}}{BP_{Hi} - BP_{Lo}}\left(C_p - BP_{Lo}\right) + I_{Lo}}$$

$\boldsymbol{I_P}$ → Index of the pollutant P

$\boldsymbol{I_{Hi}}$ → AQI corresponding to BP_{Hi}

$\boldsymbol{I_{Lo}}$ → AQI corresponding to BP_{Lo}

$\boldsymbol{BP_{Hi}}$ → Breakpoint[1] that is higher than or equal to C_p

$\boldsymbol{BP_{Lo}}$ → Breakpoint that is lesser than or equal to C_p

$\boldsymbol{C_p}$ → Concentration of pollutant P

[1] Breakpoint is the concentration point which separates each AQI classification level

In the below sections, the data mining followed by the Android application modules are presented.

2.2 Data Mining: An Introduction

Data mining or Knowledge Discovery (KD) is the process of reading and analyzing large datasets and then finding/extracting patterns from the data. It is used for predicting the future trends or forecast patterns over a period. Data mining algorithms are usually based on well-known mathematical algorithms and techniques [19]. There are two types of data mining learning algorithms: 1) Supervised algorithms and 2) Unsupervised algorithms.

2.2.1 Unsupervised Learning Algorithm

The Unsupervised algorithm is the process in which the training dataset contains only the input set and not the corresponding target vectors. The main criterion is to find groups or patterns of similar examples within the dataset, called as clustering [20].

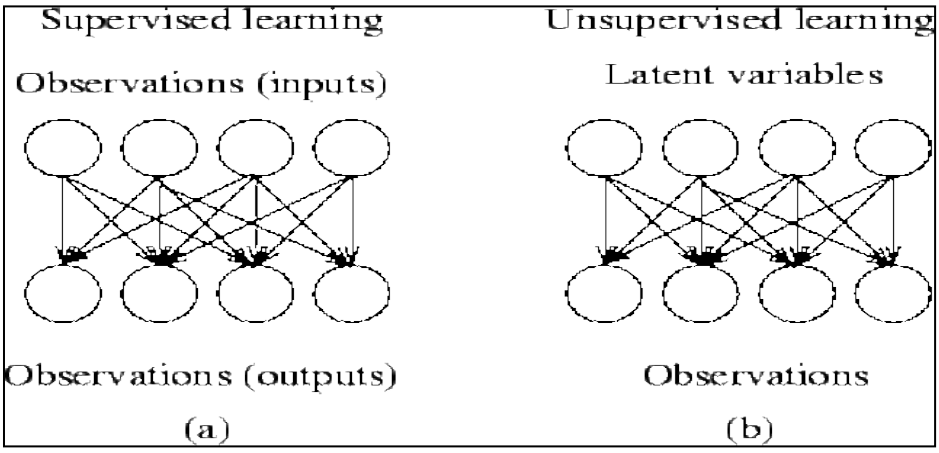


Figure 2-1: Supervised and Un-Supervised Learning [21]

2.2.2 Supervised Learning Algorithm

The Supervised algorithm is the process in which the training data comprises of both the training and the corresponding output target vectors [20].

In this project, a supervised learning algorithm called Artificial Neural Network (ANN) has been used for training, validation and testing the dataset. In addition, to the ANN, a Multiple Linear Regression (MLR) model has been used for comparing the performance against the ANN. The below section introduces the processes of Artificial Neural Network (ANN) and Multiple Linear Regression (MLR).

2.2.2.1 Artificial Neural Network

Artificial Neural Network (ANN) is supervised learning algorithm that consists of many modules, among which the most commonly used algorithm is the Back-Propagation (BP) Neural Network (NN). In a Neural Network (NN), the hidden layers play a critical role in the Back Propagation (BP) process. As Hornik describes, "a network with a single hidden layer with a sufficiently large number of neurons can approximate any smooth, measurable function between input and output vectors by selecting a suitable set of connecting weights and transfer functions" [34]. Therefore, the Neural Network (NN) architecture considered for this project consists of only one hidden layer. The below Figure 2-2 shows the Neural Network (NN) architecture containing the input, hidden and the output layers respectively from left to right. The different layers of the Neural Network (NN) (input, hidden and the output layer) will be explained in the later sections of the report.

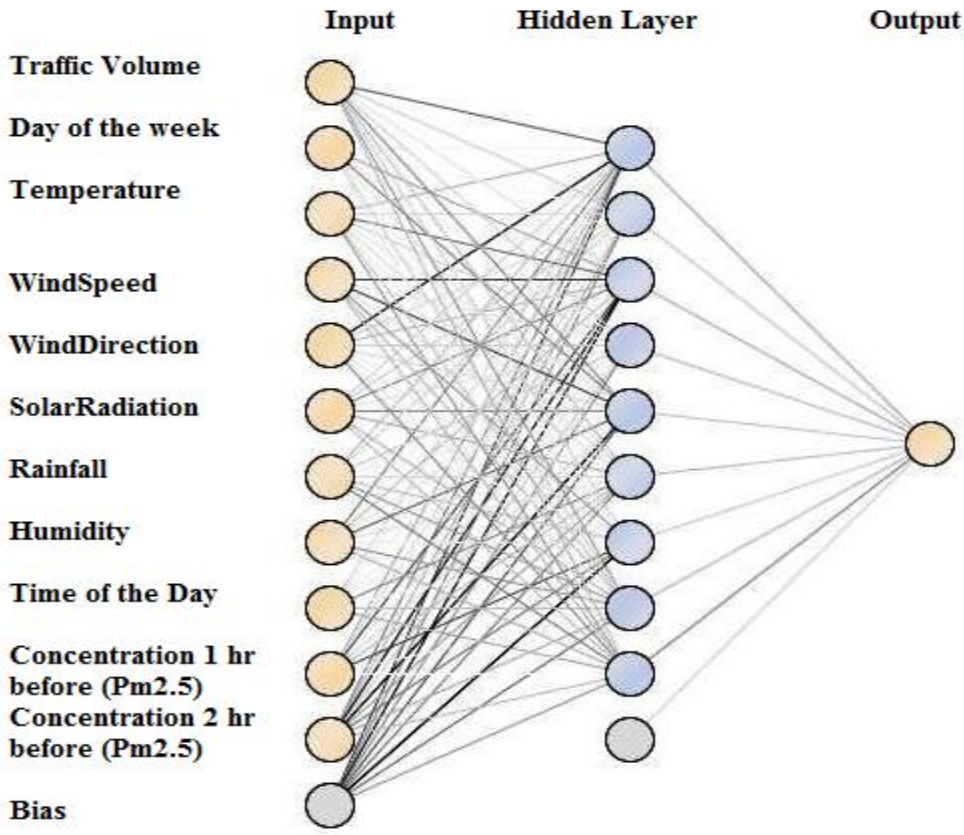


Figure 2-2: Architecture of the Neural Network Model

The Neural Network (NN) process used for the data model consists of two different phases:

- Phase 1: Feed Forward Propagation

  The input is passed in a feed-forward manner through hidden layer to the output layer. The feed-forward method maps the Multilayer Perceptron (MLP) values to that of the output values using an activation function.

  The activation function, also known as the transfer function, introduces nonlinearity to the network because without nonlinearity the Neural Network (NN) will fail to converge. The non-linearity function is introduced to every other layer except the input nodes. In this project, the tangent function is used as the activation function [35], whose output ranges from [-1, 1] instead of the sigmoid function which has a range from [0, 1]

$$\sigma(X) = \boldsymbol{tanh}(z) = \frac{e^{Z} - e^{-Z}}{e^{Z} + e^{-Z}}$$

where $\sigma(X)$ is the tangent/activation function.

$$X = \left(\sum_{i=1}^{m} w_i x_i + w_0 \right)$$

where w_i is the weight of the ith neuron of the input layer to the ith neuron of the hidden layer; x_i is the output of the ith neuron of the input layer in the ith sample while w_0 is the bias invariant neuron of the hidden layer. The purpose of using Biases is to preserve the universal approximation of the Neural Network (NN) [32][34]. Consequently the output for one hidden layer and one output can be expressed as:

$$\sigma(X) = \sigma\left(w_0 + \sum_{i=1}^{h} w_h \, \sigma \left(w_0^h + \sum_{i=1}^{i} x_i \, w_i^h \right) \right)$$

The Multilayer Perceptron (MLP) trains the network by using the Back-Propagation (BP) algorithm, and this network is interconnected with each other in a feed-forward method. The total errors of the Neural Network (NN) are calculated using the error function E,

$$E = \frac{1}{2} \sum_{i=1}^{n} (t_n - a_n)^2$$

where E is the total error for the training set, t_n represents the value of n for target node and a_n represents the activation node and ½ is used for simplifying the derivative [36]. The delta rule that is given by gradient descent on the square error is used in the Back Propagation training method to update the weights [29].

- Phase 2: Weight Update

Back-Propagation (BP) is the process of calculating the error function and updating the synaptic weights of the input nodes to reduce the loss function [29] [32]. If the desired output

is not achieved in the output layers, the error signals are back propagated through the network during which the synaptic weights are adjusted to that of the error signals.

The learning rate parameter determines the weight value for each updating step for the algorithm. To minimize the cost error function E, the weight value $\boldsymbol{w_i}$ is modified in accordance to achieve gradient descent in E.

$$\boldsymbol{w_i} \leftarrow \boldsymbol{w_i} - \eta \frac{\boldsymbol{\partial E}}{\boldsymbol{\partial w_i}}$$

where η is the learning rate, when smaller it takes a longer time to achieve gradient descent and when substantial, larger modification of $\boldsymbol{w_i}$ are performed to achieve gradient descent.

The iterative process or chain rule keeps continuing till the error gets reduced between the desired output and network output, commonly known as the delta rule. The learning rate and the momentum are chosen as 0.3 and 0.2 respectively for this project.

2.2.2.2 Multiple Linear Regression

A Multiple Linear Regression (MLR) is the method where a relationship is established between two or more independent variable x on a dependent variable y. The population regression line p for the x independent variables is defined as $\mu_y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p$. The μ_y line represents the changes of the independent variable to that of the dependent variables [43].

The MLR for n given observation is expressed as follows: [43]

$$\boldsymbol{y_i = \alpha + \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \beta_p X_{ip} + \varepsilon_i \quad for\ i = 1, 2, \ldots, n}$$

where $\boldsymbol{\alpha}$ is the intercept and $\boldsymbol{\beta_i}$ are the parameters for the input variables $\boldsymbol{X_i}$ and $\boldsymbol{\varepsilon_i}$ is the error rate.

2.3 Data Mining Architecture

The data mining architecture consists of several stages to achieve a high rate of prediction accuracy. In the selection procedure, the target data is identified based on the attributes influence on the target. The second step is the pre-processing step where the data is cleaned by removing noise, outliers and normalization.

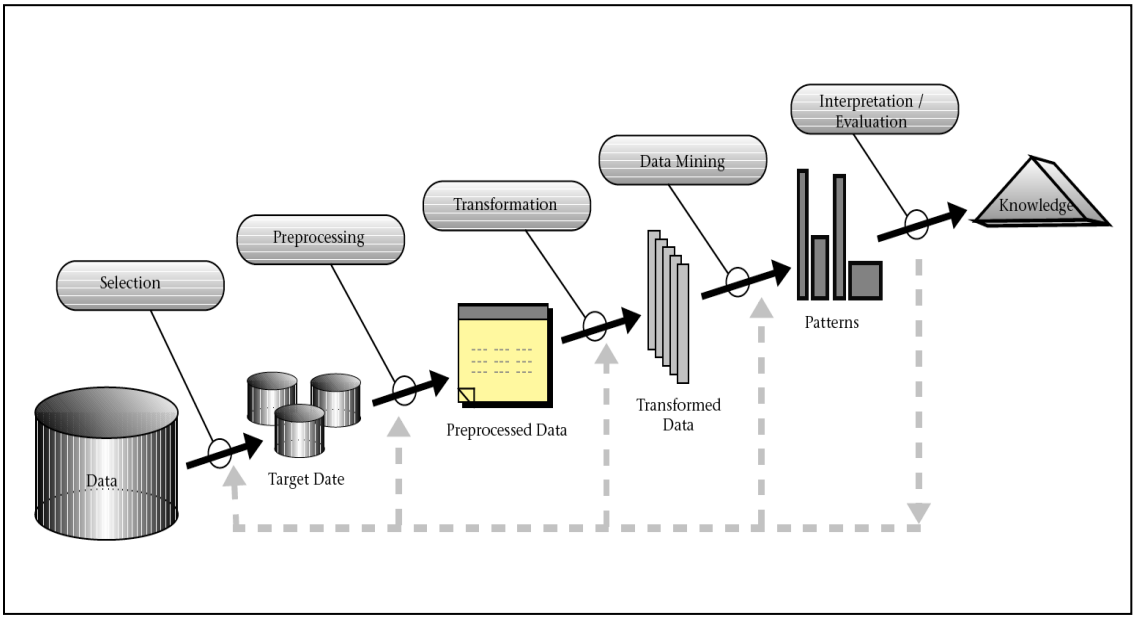


Figure 2-3: An overview of the KDD process [19]

In the third step, methods such as dimensionality reduction, feature selection, and functional transformation are performed. The error free data is passed to the data mining model where a suitable data mining algorithm is applied. This algorithm can either be a supervised or an unsupervised model as seen in Figure 2-1. In the last before step; we interpret the patterns by the target set in the initial step and finally pass the discovered knowledge onto another system [19].

In the below sections, the three different types of phases used during the training, validation and testing of the data model are discussed.

2.3.1 Training Phase

The training dataset is used for training the dataset and in a Neural Network (NN) it is used for adjusting the weight so that the model fits.

2.3.2 Validation Phase

The validation dataset is used to make sure that the model does not suffer from “overfitting” or “underfitting”. The validation set is used to fine tune the model and increase the model's accuracy by testing it against the unseen dataset. In this project, a split validation technique is used for improving the accuracy of the model

2.3.2.1 Split Validation

The Split validation operator randomly splits the training dataset into two separate sets called training and test dataset and then tries to evaluate the model. This is a nested operation and the operation keeps splitting the dataset into two random parts and estimates the performance of the model against the test data. This method is used for finding the optimal number of hidden layers to be used for the Neural Network (NN).

2.3.3 Test Phase

The test dataset is used finally after building the model to check the models predictive performance against unseen data. It also gives an estimate on the error rates of the final predictive model.

2.4 Android Platform: An Introduction

Android is an open-source software platform and Linux-based operating system for mobile devices like phones, tablets, and even netbooks. It was developed by Open Handset Alliance (OHA) [22], partnered by Google and many other companies. It provides the users and developers unlimited access to its resources because of it being published under Apache software license 2.0. The open-source software made it a significant success in the case of Android as it leads the global market share with 82.8% while its competitors share the remaining in the second quarter sales during the year 2015 [8]. In the follow-up section, a brief introduction to the Android system architecture is given.

2.4.1 System Architecture

The system architecture of Android is made of several layers, with each layer having its functionality and the processes. The top most layer is the "Application" layer and is mostly written in the Java programming language. The developers make use of this layer to write and install their applications. It also comes with several pre-installed applications from the manufacturers [24].

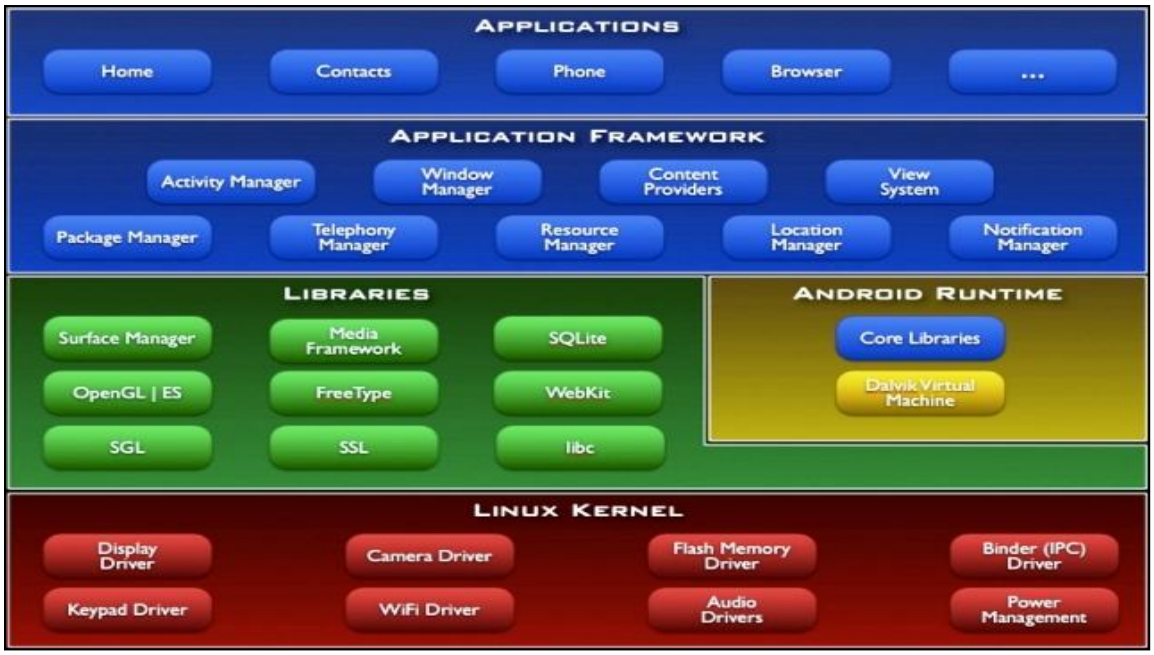


Figure 2-4: Android Architecture [23]

The second layer is the "Application Framework" layer, which contains the Java helper classes for the "Application" layer. Developers make use of these services in the "Application" layer [24]. The third layer is the "Libraries" layer that is written in C or C++ based on the specific hardware. It includes the SQLite library for storage purposes, Webkit library for displaying HTML content, etc. The fourth layer is the "Android Runtime", which consists of Dalvik Virtual Machine and Core Java libraries [23]. The final and the lowest layer is the "Kernel" layer that is the core of the Android operating system. This layer makes it possible for different manufacturers to run Android on various devices with different hardware.

2.4.2 Android Manifest file

The Android Manifest file is the root directory of any Android application and holds all the necessary information regarding the application. It contains information about the application package name, components of the application (activities, services, and content providers), linked libraries, minimum API required to run the application, etc. The Android manifest file is unique and indicates the application which activity should be run when the application starts [26].

2.4.3 LogCat

Logcat is the primary Android logging system which provides the developers with the application's debug output. Logcat is used to print various messages depending on the Log type used. The developers can utilize seven different types of Logcat: verbose, information, debug, warning, and error. Each Log cat type has its property and displays information based on its property [27].

2.5 Message Format

The JSON message format is used by the Android application to retrieve the data from the backend and display it on the user interface. The below section talks about the different types of JSON data structure used during the development of the Android application.

2.5.1 JSON

JSON (JavaScript Object Notation) is a lightweight script that is human readable and understandable format. It is used for exchanging plain text based data [28]. JSON contains two types of data structures:

- JSON Object

  JSON Object is the unordered set of key-value pairs usually seperated by a (:) colon [28].

Code Snippet 2-1: Example of JSON Object

```
{"Key1":"value", "Key2":"value"}
```

- JSON Array

It is an ordered list of values mostly used in the form of an arraylist, vector, and sequence [28].

Code Snippet 2-2: Example of JSON Array

```
Names": [
        {"Key1":"value","Key11":"value"},
        {"Key2":"value","Key12":"value"},
        {"Key3":"value","Key13":"value"}]
```

2.6 Software Development Tools and Technology

This section discusses the various tools and technologies that are used in designing and developing the application.

2.6.1 The Android Application

The tools used for the development of the Android application are described below.

- Eclipse

  Eclipse is an Integrated Development Environment (IDE) and is one of the tools used for Android development. The minimum API for the application was set to 18 (Android 4.3). It can also be integrated with an Android Developer Tools (ADT).

- Android Software Development Kit (SDK)

  The Android SDK enables the users to develop applications for the Android platform. It includes sample source code, libraries, documentation and emulator that are required for building the application.

- Java

Java is the programming language used during the development of the Android application.

- Dalvik Debug Monitoring Service (DDMS)

  The DDMS is the debugging interface between the application and the IDE. It allows the developer to analyze and debug the source code with the help of breakpoints.

- Nexus 5

  Nexus 5 Android phone was used for testing the Android application. It runs the latest Android version of 5.1(Lollipop)

- Robotium

  Robotium is the test automation framework for Android development. In this project, several test scenarios were run using this test framework [9].

2.6.2 Data Model

The tools used for the development of the data model are described below.

- RapidMiner

  RapidMiner is an open source platform used for performing various data mining, machine learning, and text mining tasks. It is mostly used for performing predictive analysis [10].

2.6.3 Libraries

The libraries used for the development of the Android application are described below:

- Google Maps Android API Utility Library

  The Google Maps Android API library provides the users with wide range of features that can be used in Google maps.

- Android Slidingup Panel

  The "Android Slidingup Panel" is an open source library that offers the simple draggable sliding panel. In this project, it helps in creating a similar Google maps style user interface.

- Java Geocalc

  "Java Geocalc" is an open source Java library that helps in calculating the distance between two coordinates.

Chapter 3

Related Work

In this chapter, the related work carried out for forecasting the pollution concentrations along with various applications that provide users with real-time information about pollution concentrations are discussed.

3.1 Data Forecasting

Air pollution is a huge problem all over the world. The most common form of pollution is from vehicular exhaust, industries and also from small scale businesses. The main focus of this project is in identifying the various factors that might influence pollution concentrations and also in building a data model for forecasting the pollution concentration on the identified influential factors.

Forecasting is performed using two different approaches: deterministic and statistical. In the deterministic approach, the future value is predicted with the help of specific data knowledge and in the statistical approach, the future value is predicted with assistance from statistical data collected over time [11].

Over the past few years, there have been many different algorithms used for forecasting the pollutant concentration. The commonly used forecasting method is the statistical model because of its easier implementation and smaller calculation time. The statistical model establishes an underlying relationship between the input and the output variables i.e. a relationship is established between the past values or relevant variables to that of the future values [25].

The statistical model based on Neural Network (NN) has been used widely during the recent years for air quality forecasting. Gardner and Dorling concluded that the Neural Network (NN) ability to handle non-linear behavior led to better results in comparison with other statistical linear methods [12]. They performed tests with and without emission factors to come to a conclusion that the Neural Network (NN) without any external guidance can identify emission

patterns in comparison with other statistical models. Also, they also concluded that the emission rate variation was highly dependent on the time of the day and day of the week [12].

Perez performed a comparative study to identify the suitable forecaster of hourly PM2.5 value from 1994-1995 in Santiago, Chile. He used three different models: multi-layer NN, linear regression, and persistence methods, and concluded that the best hourly predicted concentrations of PM2.5 were obtained from the Neural Network (NN) [13]. In 2001, Kolehmainen [44] performed an evaluation of various statistical models for the hourly concentration of NO2 along with certain meteorological factors and concluded that the Neural Network (NN) produced better prediction results of NO2 than the other linear models.

In 2002, Balaguer-Ballester [14] presented a comparison of several prediction models like the Auto-Regressive-Moving Average with Exogenous Inputs (ARMAX), MultiLayer Perceptrons (MLP) and finite impulse response Neural Network (NN). Their results indicated that MLP Neural Network (NN) was more effective than the remaining two models.

Similar research was performed by Kukkonen [15] in 2003 where they compared five Neural Network (NN) models taking into consideration the flow of traffic and the meteorological aspects for predicting the PM10 and N02.

Finally [16] presents a recurrent Neural Network (NN) for predicting the pollutant concentrations of SO2, O3, PM10, CO two days in advance taking into consideration the meteorological aspects like wind direction, speed, pressure and temperature. They concluded that they were able to create a powerful model that has a correlation coefficient between the ranges of 0.72 to 0.98 for every predicted pollutant thus proving a small difference between the measured values and the forecasted values [16].

3.2 Mobile Applications

There are many different applications available on the market that provides the users with real-time pollution concentration of a city. The most downloaded and higher user rating applications from the market are the CleanAir and the Air Quality China application as shown below in Figure 3-1.

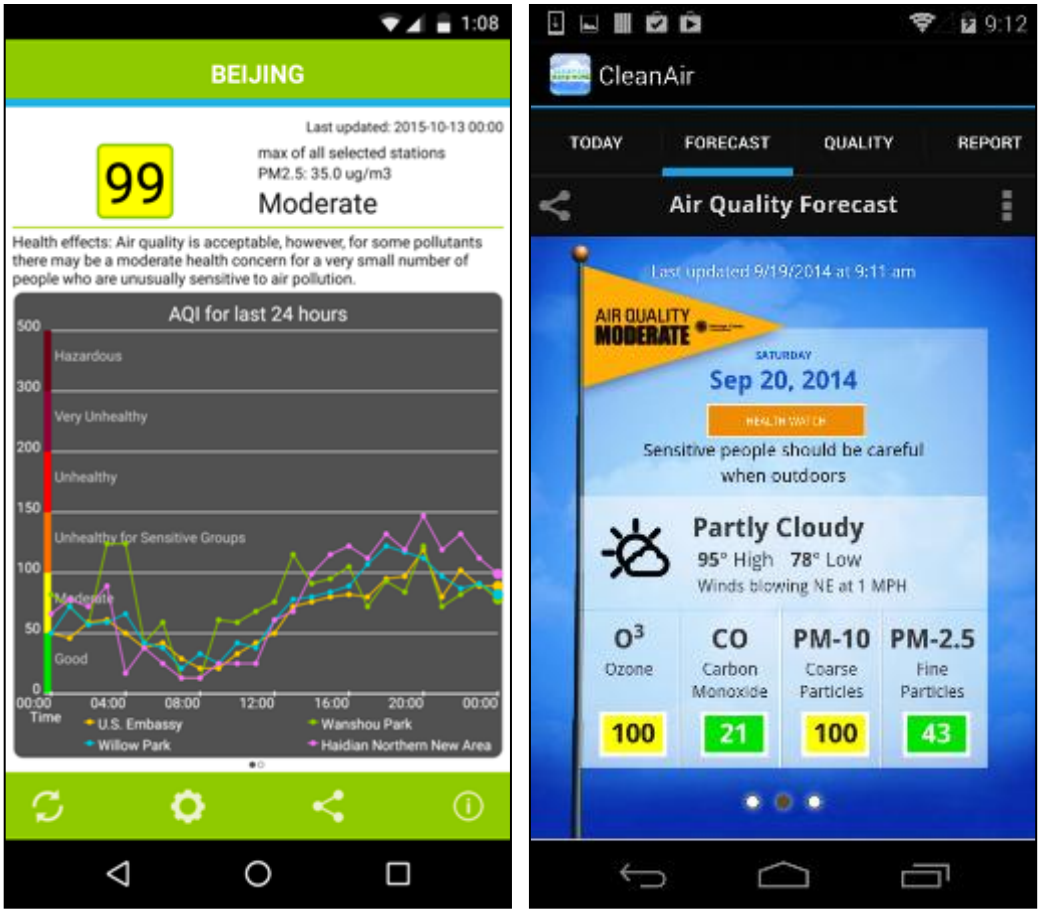


Figure 3-1: (A) Air Quality China (B) Clean Air Application

The Air Quality China application in Figure 3-1(A) provides the people of China with real-time AQI level of PM2.5 over various parts of the country. The CleanAir application in Figure 3-1(B) provides the users with real-time pollution concentration of different pollutants along with the forecast for the following day. This application was designed by the Maricopa County Air Quality Department, Arizona in the public interest.

Chapter 4

System Architecture

This chapter gives a brief overview of the system architecture followed by the data collection from various sources that will be used in the Knowledge Discovery (KD) process.

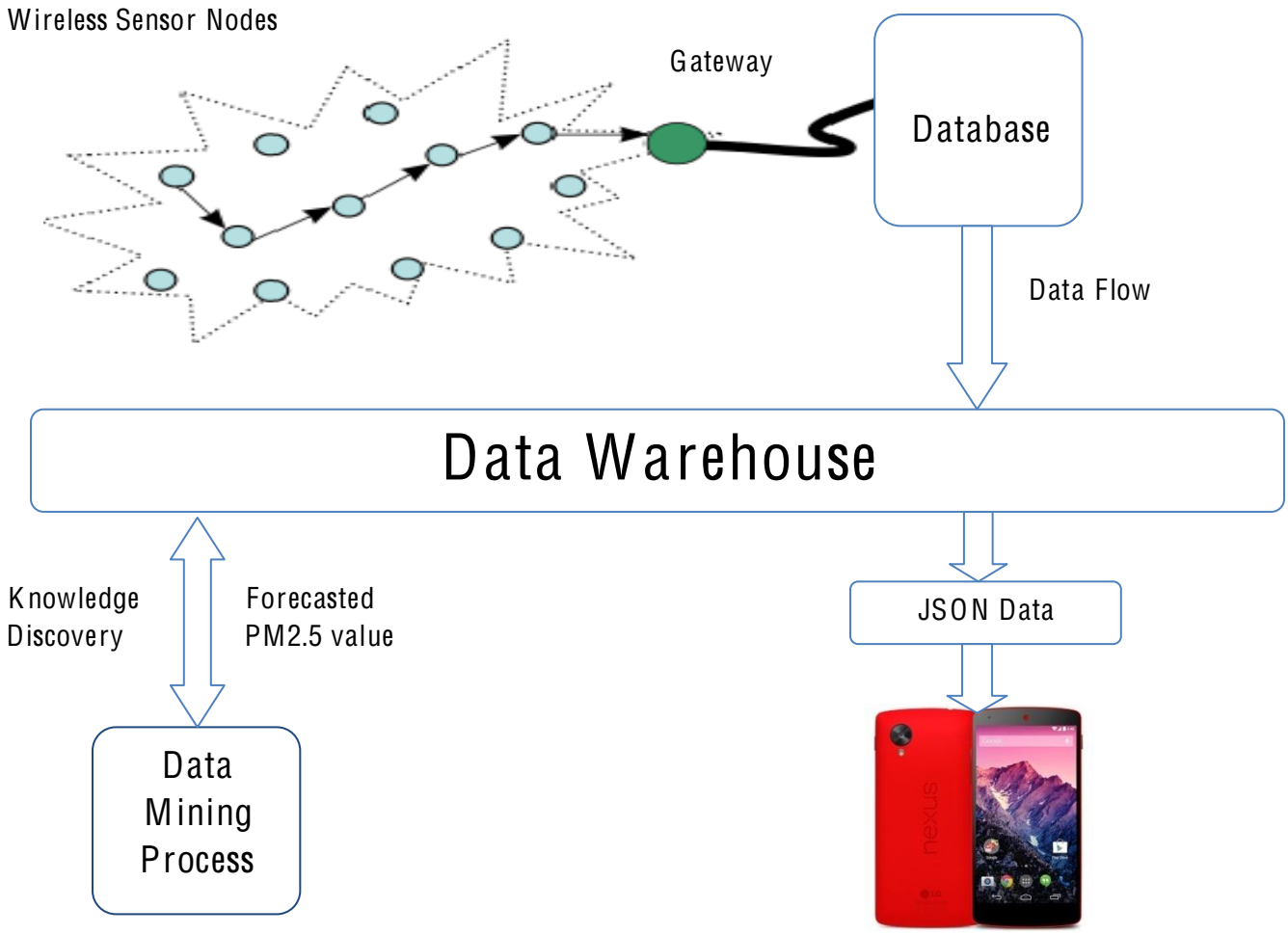


Figure 4-1: Block Diagram of the System Architecture

The Figure 4-1 represents the system architecture, and it consists of four main components: wireless sensors, database and data warehouse, data mining process and the Android application. The wireless sensor development and the data warehouse implementation are being carried out by another group of students from Uppsala University in collaboration with Upwis AB. The

sensors being developed will be able to detect the PM2.5 concentration along with the temperature, pressure and CO2 levels. In this project, the main focus is laid on the data mining process and in developing the Android application. The data mining component will be used for creating a data model using the data collected from the wireless sensors. The model will forecast the pollutant concentration of PM2.5 for the next few hours in advance. The Android application component will fetch the JSON objects from the data warehouse and provide the users with real-time pollution concentration of a location.

The following sections will talk about the various sources from where the data has been collected for the Knowledge Discovery (KD) process.

4.1 Data Collection

The Uppsala Municipality has two monitoring stations one in Uppsala Klostergatan and other in Uppsala Kungsgatan. The monitoring sites are located 3m from the ground and have a range of 100 m radius. The station performs real-time monitoring of PM2.5, PM10, NOx (NO+NO2) concentrations along with meteorological parameters like temperature, pressure, solar radiation, rainfall, etc. In this project, the datasets for the year 2012 to 2014 from the monitoring station at Uppsala Kungsgatan are taken into consideration. In addition, to the pollution concentrations and meteorological factors the numbers of on-road vehicles are also taken into account because of it being the main source of pollution in an urban area. The on-road vehicles are calculated using a metro count system and in this project the one close to the monitoring station (Uppsala Kungsgatan) which is Vaksalagatan is taken into consideration for the same time period i.e. 2012 to 2014.

The below Figure 4-2 shows the location of the monitoring station along with the route where the vehicle counter was placed. To know more information about all the locations where the vehicle counting was performed, refer the Appendix A-1 section that contains a detailed map of the Uppsala city where the vehicle counters were placed.

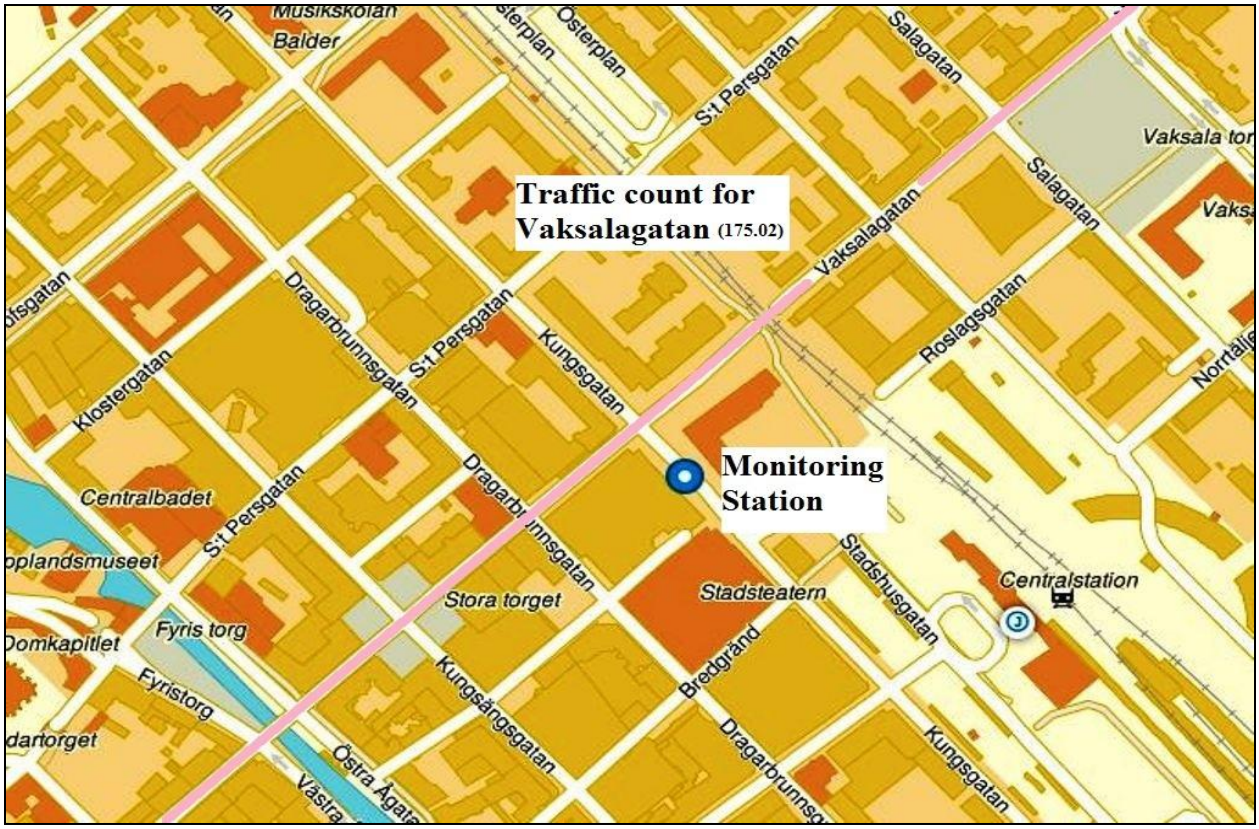


Figure 4-2: Map containing the Location of the Monitoring Station

The blue marker points to the location where the monitoring station (Uppsala Kungsgatan) is present and the pink colored line through Vaksalagatan was the route where the vehicle count was calculated during October from 2012-2014.

The below section talks about the monitoring stations and the vehicle counter system used in the Uppsala County.

4.1.1 Monitoring Stations

The Stockholm-Uppsala County Air Quality Management Association is responsible for monitoring the air quality in Uppsala and Stockholm. This association in collaboration with the Uppsala Municipality and 35 different municipalities to make sure that the air quality is kept under check.

The main function of this association is to maintain a list of the emission sources, measuring air quality and meteorological parameters, and creating dispersion models (Wind model, Gauss

model, Grid model, and Street canyon) which are used to calculate the pollutant concentration from emissions [30].

4.1.2 Vehicle Count

The Uppsala Municipality does not have a permanent vehicle counting system in place. It calculates the vehicle count once or twice every year for a week. The municipality makes use of the MetroCount [31] device for counting the number of vehicles passing through a particular part of the city. The device contains piezoelectric sensors that will generate an electric charge when a vehicle passes through it thereby enabling monitoring of the vehicles. The below Figure 4-3 shows the vehicle count over the course of three years from 3rd to the 9th for the month of October in the Vaksalagatan region, near Kungsgatan. The X-axis corresponds to the dataset chosen for the month of October from 3rd to the 9th for all three years, and the Y-axis represents the vehicle count performed during the same time period in the Vaksalagatan region, near Kungsgatan. It can be seen from the graph that the number of vehicles has increased by two folds during these years thereby confirming the rise in the pollution level over the last three years.

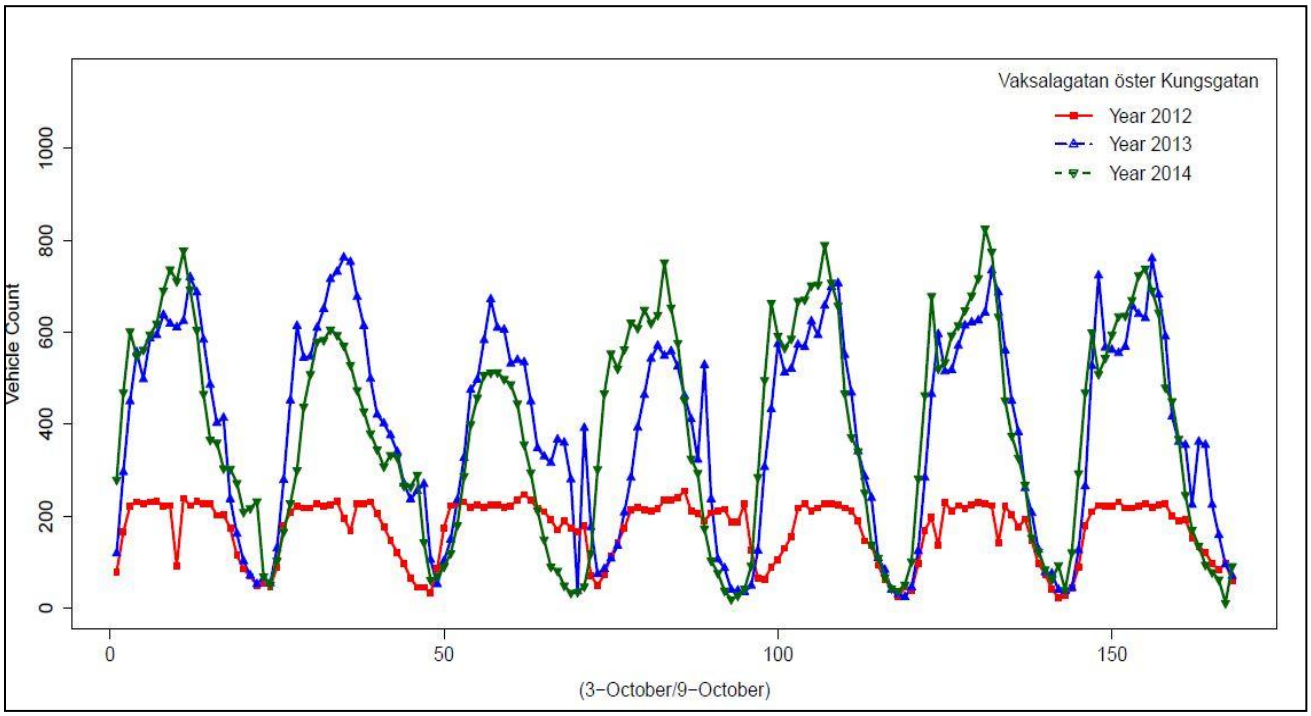


Figure 4-3: Vehicle count for Vaksalagatan

Chapter 5

Knowledge Discovery Process Implementation

5.1 An Introduction

In the light of the previous discussions, a statistical model using Artificial Neural Network (ANN) will be used in designing the data model. Artificial Neural Networks (ANN) is a supervised learning algorithm and consists of many modules among which the most commonly used algorithm is the feed forward method with Back-Propagation (BP).

The reason for using a statistical model is because unlike the deterministic model it does not require lots of information to be fed to the model for accurate prediction. A tangent activation function was used instead of the sigmoid activation function that has been used over the years. The tangent function has a range between -1 and 1 rather than between 0 and 1 unlike sigmoid activation function. The reason for choosing a tangent activation function is because it converges faster in comparison to a sigmoid function.

The below section describes the step by step implementation of the Knowledge Discovery (KD) process used for forecasting the AQI value of pollutant concentration PM2.5.

5.2 Data Selection

The important aspects that need to be considered when it comes to forecasting of the pollutant concentration are its various sources along with the factors that influence its concentration. The SMHI pointed out that it would be difficult to reduce the pollution levels unless the emissions caused by road traffic are restricted [3]. Therefore, the numbers of vehicles on the road are considered as one of the input factors along with the time of the day and the day of the week. These factors combined can be regarded as vehicular factors. The other sources of pollution like industrial pollution or restaurant emission were not taken into consideration for the model design because of its irregular emission times which makes it hard to measure and represent its effects as a valid variable over a period of time [32]. In addition to the vehicular exhaust, meteorological factors play a vital role in distribution and dispersion of the pollutant concentration. The

following meteorological factors are also taken into consideration as input factors: temperature, pressure, precipitation, humidity, solar radiation, wind speed, and wind direction [32]. For an approximation of the prediction results 2 hours prior data of the pollution concentration (PM2.5) are taken into consideration before performing the prediction and are considered as historical information.

In summary, the factors that affect the concentration of the pollution are vehicular, meteorological and historical information that accounts for a total of 11 influential factors for the prediction model.

Table 5-1: Influential Factors used for the Model

Vehicular Factors	Number of vehicles, Date and Time of the Week
Meteorological Factors	Wind Speed Wind Direction Humidity Rainfall Solar Radiation Temperature
Historical Information	Pollutant concentration 1hr before Pollutant concentration 2hr before

The final valid dataset after data transformation consists of a total of 500 training dataset, were 20% of the dataset is used for validation and another 24 datasets for testing. The training dataset is used for training the model according to its output weights, and a split validation is performed for identifying the hidden neurons and fine tuning of the model. Finally, the test dataset is used for checking how well the model performs against unseen data after learning.

5.3 Data Transformation

5.3.1 Normalization of the Samples

The data model is susceptible to overflows in the network because of irregularities in the values or weights. To remove these irregularities the range transformation method of normalizing all the values in the range of [0, 1] is applied [37]. The range normalization function is as below:

$$\boldsymbol{X_{normalization}} = (\boldsymbol{X_i} - \boldsymbol{X_{min}}) \,/\, (\boldsymbol{X_{max}} - \boldsymbol{X_{\min}})$$

where $\boldsymbol{X_{normalization}}$ is the normalized value, $\boldsymbol{X_i}$ is the *ith* value passed, and $\boldsymbol{X_{min}}$ and $\boldsymbol{X_{max}}$ are the minimum and maximum value for $\boldsymbol{X_i}$ value.

5.4 Data Mining Technique

5.4.1 Building Neural Network

The nature of the problem determines the number of input and the output neurons. A total of 11 influential factors are considered as the input to the Neural Network (NN), and the output is the pollutant concentration of PM2.5. As Swingler [33] points out that "the hidden neurons will differ with different instances". To prevent from performance issues like "overfitting" and "underfitting", different numbers of hidden neurons were trained and validated against the same dataset [32]. To evaluate the prediction results, the following error measures were considered: Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Relative Error (RE) [38].

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |\boldsymbol{P_i} - \boldsymbol{M_i}|$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (\boldsymbol{P_i} - \boldsymbol{M_i})^2}$$

$$\mathrm{RE} = \frac{1}{n}\sum_{i=1}^{n} \frac{|\boldsymbol{P_i} - \boldsymbol{M_i}|}{\boldsymbol{M_i}}$$

where n is the number of data in the test dataset, $\boldsymbol{P_i}$ and $\boldsymbol{M_i}$ are the predicted and measure value for the $\boldsymbol{i^{th}}$ hour.

In this project, different layers of Neural Network (NN) models were developed for the pollutant concentration of PM2.5. These models were trained on the same training dataset, and the validation dataset was used for identifying the best performing model. The error functions were calculated for various hidden neurons and are presented in the below Table 5-2. It can be seen from the table that different hidden number of neurons has different error rates and the neuron layer with the lowest error rates will have the smaller difference between the predicted and the measured value leading to best prediction rate. Therefore for this dataset, having 9 hidden neurons will lead to the best prediction results.

Table 5-2: Error Measures for different hidden neurons

Number of Neurons	4	5	6	7	8	9
RMSE	0.157	0.109	0.111	0.112	0.148	0.111
Absolute Error	0.128	0.084	0.085	0.087	0.11	0.082
Relative Error (%)	78.86	43.50	48.38	41.20	65.51	37.23

Number of Neurons	10	12	14	16	24	32
RMSE	0.139	0.116	0.117	0.116	0.117	0.126
Absolute Error	0.105	0.09	0.085	0.086	0.087	0.087
Relative Error (%)	40.13	37.94	41.19	47.26	39.78	39.63

5.4.2 Influence of the Input factors

The influence of input factors plays a vital role during the forecasting of the pollutant concentration of PM2.5. The previous 2 hours pollutant concentrations of PM2.5 are taken into consideration when forecasting the hourly PM2.5 concentration. Therefore during the first forecast, the pollutant concentrations of the previous 2 hours are taken into account. Then for the second forecast, the first forecasted value of PM2.5 and the concentration before the first predicted value are taken into consideration and so on. This method can be used for forecasting for n arbitrary hours in advance.

Different combinations of the inputs based on the influential factors were carried out, and the prediction results are presented in the below tables.

Table 5-3: Influential Factors: Vehicular, Meteorological, Historical Information

Number of Neurons (11-9-1)*	4	6	7	8	9	10
RMSE	0.157	0.111	0.112	0.148	0.111	0.139
Absolute Error	0.128	0.085	0.087	0.11	0.082	0.105
Relative Error (%)	78.86	48.38	41.20	65.51	37.23	40.13

*Structure a–b–c → a: number of inputs, b: number of hidden neurons and c: number of outputs

Table 5-4: Influential Factors: Vehicular, Meteorological

Number of Neurons (9-4-1)*	4	6	7	8	9	10
RMSE	0.335	0.644	0.85	0.506	0.668	0.589
Absolute Error	0.235	0.547	0.694	0.418	0.523	0.485
Relative Error (%)	46.78	104.81	103.75	79.78	99.64	91.57

*Structure a–b–c → a: number of inputs, b: number of hidden neurons and c: number of outputs

Table 5-5: Influential Factors: Meteorological, Historical Information

Number of Neurons (8-7-1)*	4	6	7	8	9	10
RMSE	0.421	0.441	0.253	0.502	0.495	0.414
Absolute Error	0.344	0.358	0.216	0.381	0.334	0.322
Relative Error (%)	64.40	47.29	42.99	46.33	49.23	48.60

*Structure a–b–c → a: number of inputs, b: number of hidden neurons and c: number of outputs

From the above tables, it can be concluded that by using all three influential factors together yields best possible outcome. It can also be noted that the error rates are higher when all three influential factors are not taken into consideration. Thus, we can conclude, to yield the best prediction for the Neural Network (NN) all three influential factors: vehicular, meteorological, and historical information should be taken into account. The final model of the Neural Network (NN) will have one hidden layer with 9 hidden neurons. This architecture will yield the best result because of its lesser error rate when compared to other hidden neuron layers. The architecture of the Neural Network (NN) with 9 hidden neurons is presented in Figure 2-2.

Chapter 6

Android Application Implementation

6.1 An Introduction

The Android application will provide the AQI level of the pollutant concentration (PM2.5) from every location where the sensors are to be placed. It will also provide the users with the hourly predicted value of the pollutant concentration (PM2.5) from each location. Also, it will suggest the users with less polluted vehicle navigation route based on Google driving directions.

In this module, we will look into the implementation of the Android application and its corresponding layouts.

6.2 Requirements

The requirements were based on extensive research carried out on user studies, meeting with the Uppsala Municipality, observations, and testing. In this project, an Android application was created that would provide the users with the real-time pollution data, hourly prediction data and also suggest the users with less polluted vehicle route based on Google driving navigation. The requirements were eventually tested out using a black-box [9] testing before and after the completion of the application.

6.2.1 Design Requirements

The table below represents the design needs of the application along with the status of completion.

Table 6-1: Design Requirements

ID	Description	Completed
1	A navigation drawer should effectively represent each fragment.	Yes
2	The menu in the navigation drawer must be split into categories for easier access.	Yes

3	Icons should be big enough for the users to navigate easily from one menu to another.	Yes
4	A map containing the location of each sensor with its current AQI level.	Yes
5	A menu for the user to request for vehicle navigation direction within the map.	Yes
6	An information page containing the AQI classification level according to the EU standards.	Yes

6.2.2 Functional Requirements

The functional demands of the application are listed in the below table along with its status of completion.

Table 6-2: Functional Requirements

ID	Description	Completed
1	The application should be able to provide the users with current pollution level of the selected location.	Yes
2	The application should also provide the users with the predicted hourly pollution level of the selected location.	Yes
3	The application must provide users with an alternate driving direction based on the AQI level of the sensors location.	Yes
4	The requested navigation route must be provided to the user along with the driving information.	Yes
5	The application must have a built-in database to store the downloaded information for offline viewing of the AQI level of the location.	Yes

6.3 Designing of the layout

The final design of the application was confirmed by carrying out extensive discussion with the project coordinator and testing it with users from different age categories and background. The interviewing of the users followed the same structure: Firstly, the users were given a task (such

as selecting a location, fetching directions) to perform on the application and secondly, an evaluation was done based on how the users performed each task. The final model of the application is made up of a menu (navigation drawer) containing various options (fragments) to choose from based on the user's convenience and feedback. The implementations of different layouts are presented in Appendix B) Implementing the Layouts.

In the below section, the different layouts used during the implementation of the Android application are discussed.

6.3.1 Navigation Drawer

The navigation drawer (Figure 6-1) is the main panel that displays the application's navigation options on the left-hand side of the screen [39]. It appears only when the user swipes his finger on the left corner of the screen or when the user touches the navigation drawer icon on the action bar otherwise it stays hidden.

In this application, the navigation drawer contains the following options: *Location, Go Green, and Tools* with its sub-fragments. In the *Location* options, it contains the names of the places where the sensors are placed and in the *Go Green* section, it contains a *Smart Route* option for vehicle navigation through less polluted route. In the final option *Tools*; it contains a *Help page* and a *Feedback* page for the users.

6.3.2 Fragments

A fragment is a sub-activity [40] of an activity, which represents a part of the user interface. Fragments can be created and destroyed during the run time of activity. Fragments play a significant role in building multi-pane UI by combining multiple fragments together. In this application, the navigation drawer is sub-divided into fragment types called the header fragments and sub-header fragments.

The header fragments consist of the following: *Location, Go Green and Tools*. It does not contain any user interface.

The sub-header fragments consist of the following: *Places, Smart Route, Help and Feedback* and it contains a user interface to interact.

6.4 Designing and implementing the classes and functions

Java is the programming language used in Android programming. Every activity is made up of a class and triggered by events. Every Android application contains its starting activity. For a default project, the starting activity will be the MainActivity class.

6.4.1 Main Activity

The onCreate() method triggers the MainActivity method in an Android application. The syntax for calling the onCreate() method is as follows:

Code Snippet 6-1: MainActivity onCreate() method

```
public class MainActivity extends FragmentActivity {
        @Override
        protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
}
```

This method starts when the application is run, and its layout is set by calling the "*setContentView*" method. The above example calls the *"activity_main"* as its layout. Therefore, when the application starts, it calls the XML layout which triggers the selected option from the navigation drawer list menu. The MainActivity contains several sub-activities (fragments) in its layout that are accessed using the following code:

Code Snippet 6-2: Fragment Transaction

```
android.support.v4.app.FragmentManager fragmentManager = getSupportFragmentManager();
                    fragmentManager.beginTransaction()
                                   .replace(R.id.frame_container, fragment).commit();
```

This command will call the necessary fragment for the application when selected from the navigation drawer. The FrameLayout whose id is "*frame_container*" will be responsible for showing only one fragment at a single time by blocking the other fragments.

6.4.2 Fragments

The onCreateView() methods triggers the UI for the fragment. A return view is necessary if the fragment contains a UI else we must return null. The following syntax below provides the syntax for a fragment creation:

Code Snippet 6-3: Fragment Creation

```
@Override
            public View onCreateView(LayoutInflater inflater, ViewGroup container,
                        Bundle savedInstanceState) {
                  ViewGroup rootView = (ViewGroup) inflater.inflate(
                              R.layout.fragment_home, container, false);
                  return rootView;
      }
```

In the below section, the different types of fragments used for the creation of the SmartCity Android application are discussed. As already mentioned, the fragments are split into two types, one without interface called header fragment and other with an interface called sub-header fragment.

6.4.2.1 Header Fragments

The header fragments in this application contain the following: *Location, Go Green and Tools*. Fragments can be created with or without a user interface [40]. In this fragment, the return value for the ViewGroup will be null. Therefore, the three fragments act as a header in the navigation drawer without any user interface.

6.4.2.2 Sub-Header Fragments

The sub-header fragments are accessed from the MainActivity() contains a UI for the user to interact with. In the below sections, the sub-header fragments implementation along with its functionalities for this android application are discussed.

6.4.2.2.1 Places Fragment

The Places fragment contains the location of all the monitoring stations (e.g.: Polacksbacken). This fragment makes use of the ViewPager library to implement the sliding screen interface. The ViewPager for this application consists of two separate tabs each containing its fragments i.e. when the Sub-Header fragment is selected it calls the ViewPager to trigger the action of calling its two different tabs. Once the fragment is created using the sliding screen interface, it is extended using the FragmentStatePagerAdapter as an abstract class part of the ViewPager. The PagerAdapter implements "*getitem*()" and "*getCount*()" method to supply the screen sliding fragments and also the number of pages the fragment will create respectively. The command below gives an idea of how the ViewPager adapter works:

Code Snippet 6-4: FragmentStatePagerAdapter

```
public static class MyPagerAdapter extends FragmentStatePagerAdapter   {
      private static int NUM_ITEMS = 2;
      public MyPagerAdapter(FragmentManager fragmentManager) {
                  super(fragmentManager);
            }
      // Returns total number of pages
      @Override
      public int getCount() {
            return NUM_ITEMS;
            }
      // Returns the fragment to display for that page
      @Override
      public Fragment getItem(int position) {
      switch (position) {
                  case 0: // Fragment # 0 - This will show FirstFragment
                        return FirstFragment.newInstance(Position1, "String");
                  case 1: // Fragment # 1 - This will show FirstFragment
                        return SecondFragment.newInstance(Position2, "String");
}
```

The ViewPager in this application consists of two separate tabs (fragments), and its functionalities are as follows:

- Tab 1 (Today Tab)

  The Daily tab will provide the users with the real-time PM2.5 concentration level along with temperature, humidity of the location. A custom adapter fetches the JSON object from the data warehouse and passes the values to the corresponding *id* of the XML layout.

- Tab 2 (Hourly Tab)

  The Hourly tab will provide the users with next 12 hours of the predicted PM2.5 concentration level. A custom list adapter fetches the JSON array from the data warehouse and passes the values to the corresponding *id* of the XML layout.

6.4.2.2.2 Smart Route Fragment

The smart route fragment will provide the users with a map view containing all the locations of the sensors along with the pollution concentration. The following are the steps performed when creating the smart route fragment.

- Creating Google Map

  The Google Map is called when the fragment containing the MapView "*Id"* is called from the respective XML layout. The below code snippet shows MapView called from a fragment.

Code Snippet 6-5: MapView Interface

```
    @Override
          public View onCreateView(LayoutInflater inflater, ViewGroup container,
                      Bundle savedInstanceState) {
          handler.postDelayed(runner, random.nextInt(2000));
          View v = inflater.inflate(R.layout.map_test, container, false);
          //Map View is being called with its ID
          mMapView = (MapView) v.findViewById(R.id.map);
          googleMap = mMapView.getMap();
          . . .
    }
```

- Google Driving Directions

  The MapView also provides the users with a vehicle navigation route based on the pollution concentration. When a user requests for a direction between two locations, the application will fetch the Google vehicle navigation route in the JSON format from the following URL.

  "http://maps.googleapis.com/maps/api/directions/json"

The corresponding information regarding the driving directions is obtained from the Google API in the JSON format. The sample response for two locations is shown as below.

Code Snippet 6-6: JSON Driving Directions

```
{
   "geocoded_waypoints" : [
      {
         "geocoder_status" : "OK",
         "place_id" : "ChIJxWca91_JX0YRCoVkEcAWyJU",
         "types" : [ "university", "point_of_interest", "establishment" ]
      },
      {
         "geocoder_status" : "OK",
         "place_id" : "ChIJyS2OFg7MX0YREENCWhvGDho",
         "types" : [ "route" ]
      }
   ],
   "routes" : [
      {
         "bounds" : {
            "northeast" : {
               "lat" : 59.87579499999999,
               "lng" : 17.6861247
            },
            "southwest" : {
               "lat" : 59.8335143,
               "lng" : 17.6397019
            }
         },
        . . .
        ],
         "overview_polyline" : {
            "points" :
"urflJ_xujBGMMGKDINEX?XFTHLJ@JI@C~@vDl@fDTnBj@pH|@zLBxA@x@CDGLCPKHQLmBl@qCv@sCt@@b@Nl
D~@pSn@xNXnHE`@QVmATe@Xe@j@q@dAm@`A{BxCa@j@Wf@}@dBc@fAaBzEYt@qA~C[f@eAh@qAb@gDdAIAg@k
@Mr@Qr@Wl@u@fAqClCuHbHiA`AKPqDpI_@v@SNsDzCwFvEa@`@ACCGCEKEKDGPCZ@T@D{@z@gC”
},
         "summary" : "Luthagsesplanaden",
         "warnings" : [],
         "waypoint_order" : []
      }
   ],
   "status" : "OK"
}
```

- Calculating Coordinates Distance

  The JSON response contains all the necessary navigation coordinates from source to the destination. The coordinates are passed into a function to check if it falls within the sensor radius using the Java Geocalc library. If the coordinates fall within the sensor radius the pollution concentration of that sensor is taken into consideration else ignored. Finally, the

average pollution concentration is found for each different routes received and one with the smallest concentration is chosen as the lesser pollution route.

6.4.2.2.3 Help Page and Feedback Page Fragment

The "Help Page" and "Feedback Page" fragments are created in the same way as above in the

Code Snippet 6-3: Fragment Creation. The "Help Page" fragment provides the users with the classification levels of the Air Quality Index (AQI) and its consequences while the "Feedback Page" fragment helps users interact with the developer.

6.5 Android User Interface

Snapshots of the Android application developed for this project are presented in the following figures: Figure 6-1 presents the Navigation Drawer with its various options available for the user to select. The "Location" option provides the users with the locations where the sensors are placed. The Go Green option provides the users with a MapView containing the sensor locations along with AQI value.

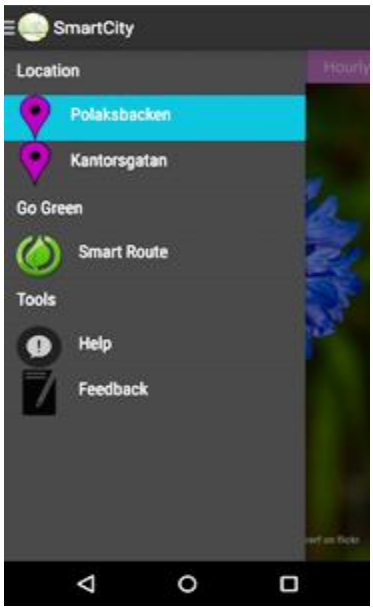


Figure 6-1: Navigation Drawer

Figure 6-2 contains two fragments that are displayed using the ViewPager. The first Tab (A) named "*Today*" shows the user the current AQI level of the location along with meteorological

values. The second Tab (B) named "*Hourly*" provides the users with the list view of the predicted hourly value of the pollutant and its description.

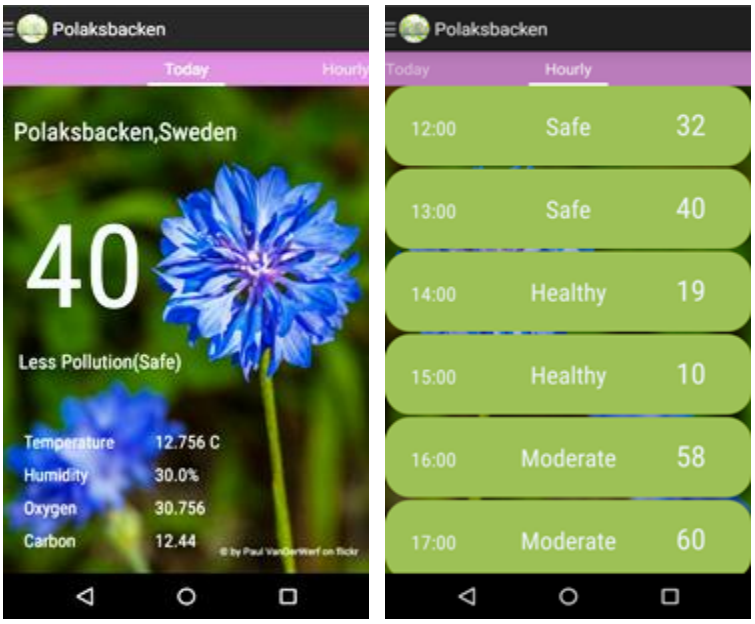


Figure 6-2: ViewPager Sliding Fragments: (A) Today Tab and (B) Hourly Tab

The Figure 6-3 (A) presents the users with the MapView containing the sensors location along with current AQI value of the pollutant as shown in (A). The top right-hand corner contains two buttons called, "*Directions*" and "*Toogle Style*". The "*Directions*" button will allow the user to enter the "From" and "To" location as shown in Figure 6-3 (B). Once the routes are requested, the application will suggest the users with the less polluted route based on the sensor locations. The "*Blue*" color marker is used to indicate the less pollution route along with its driving instructions as shown in Figure 6-3 (C). The "*Toggle Style"* button is used to change the MapView from Earth mode to Terrain mode.

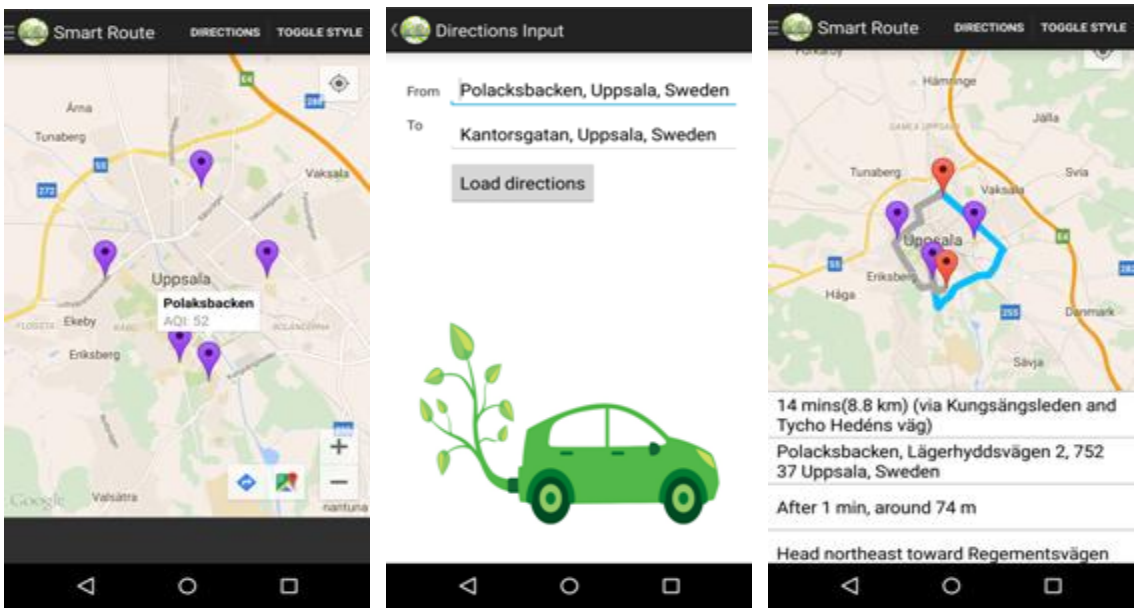


Figure 6-3: MapView: (A) Sensor Locations (B) Directions and (C) Smart Route

The help page option in Figure 6-4 (A) provides the users with necessary knowledge on the level pollution concentration and its consequences. Finally, Figure 6-4 (B) provides the feedback page that the users can use to interact with the developer.

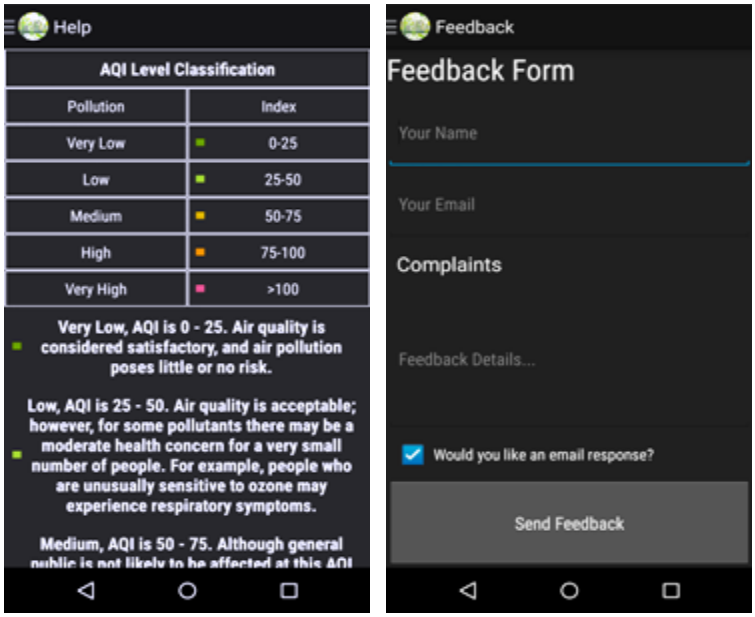


Figure 6-4: Fragments: (A) Help Page and (B) Feedback Page

6.6 Application Functionality

The application could not be tested out in real time because the sensors were not ready on time. The below section will give an idea of the expected results once the application will be deployed in the Google store.

6.6.1 Places Fragments

A file hosting server was setup to make sure the application was able to receive data from the backend. This hosting server contains a fixed data which will be passed every time the corresponding fragments are called. This was done to make sure that the application will be ready to receive data from backend once the sensors are deployed. The Figure 6-2 shows the usernterface (UI) containing the fixed data for the pollution concentration of a location.

6.6.2 Smart Route Fragment

The Smart Route fragment will suggest the users with a route that will have lesser pollution level. Smart Route takes into consideration the pollution concentration parameter when giving the users with the route. It fetches route from the Google API and then calculates the pollution average for on route sensors for different directions and finally gives the route with least pollution concentration.

For better understanding consider the following scenario: Five Sensors in the locations with the following AQI level of the pollutants:

- Polacksbacken with AQI of 52
- Dag Hammarskjöld's vag with AQI of 40
- Luthasgsesplanaden with AQI of 150
- Kantorsgatan with AQI of 92
- Tycho Hedens vag with AQI of 70 as shown in Figure 6-3

Let's say, the user wants to travel from Polacksbacken to Kantorsgatan. On requesting the driving directions, the application will fetch all the possible routes from Polacksbacken to Kantorsgatan from the Google API. Once the directions are fetched, the average pollution concentration will be calculated for on-route sensors if any and then the route with less pollution

level will be suggested to the user. The “Blue” color path as shown in Figure 6-3(C) through Tycho Hedens vag is provided in this scenario because its average pollution concentration is lesser than the remaining two routes.

Chapter 7

Experiments and Evaluations

The overall aim of the project was to develop a learner algorithm and an Android application that will provide the users with information about the level of pollution in each part of the city. Based on the research, observations, and meetings with Uppsala Municipality, the application was developed along with the data model. The final prototype will be available on the Google store once the deployment of the sensors is completed. The results and evaluation of the Neural Network (NN) are presented in the following sections.

7.1 Neural Network Results

The data model was created using the final valid dataset which consisted of a total of 500 training dataset, were 20% of the dataset was used for validation and another 24 datasets for testing.

The meteorological datasets such as wind speed, wind direction, humidity, rainfall, solar radiation, temperature including the pollution concentration: PM2.5, PM10 etc were provided by the SMHI for the monitoring station at Uppsala Kungsgatan (4.1.1 Monitoring Stations). The numbers of on-road vehicles datasets were provided by the Uppsala Municipality (4.1.2 Vehicle Count). A total of 11 influential factors were used for the model development and the results are presented below.

7.1.1 Time Series Analysis

The Neural Network (NN) for the pollutant concentration of PM2.5 was trained and validated against the training dataset, and the prediction performance was compared with the test dataset. The time series graph[2] is plotted for the Neural Network (NN) containing the smallest Relative Error (RE), Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) i.e. 9 hidden neurons from the above Table 5-2.

[2] time series is the sequence of data points especially estimated over a specified interval of time

The time series chart for the measured values and the predicted values for the training and validation set against time are shown in Figure 7-1.

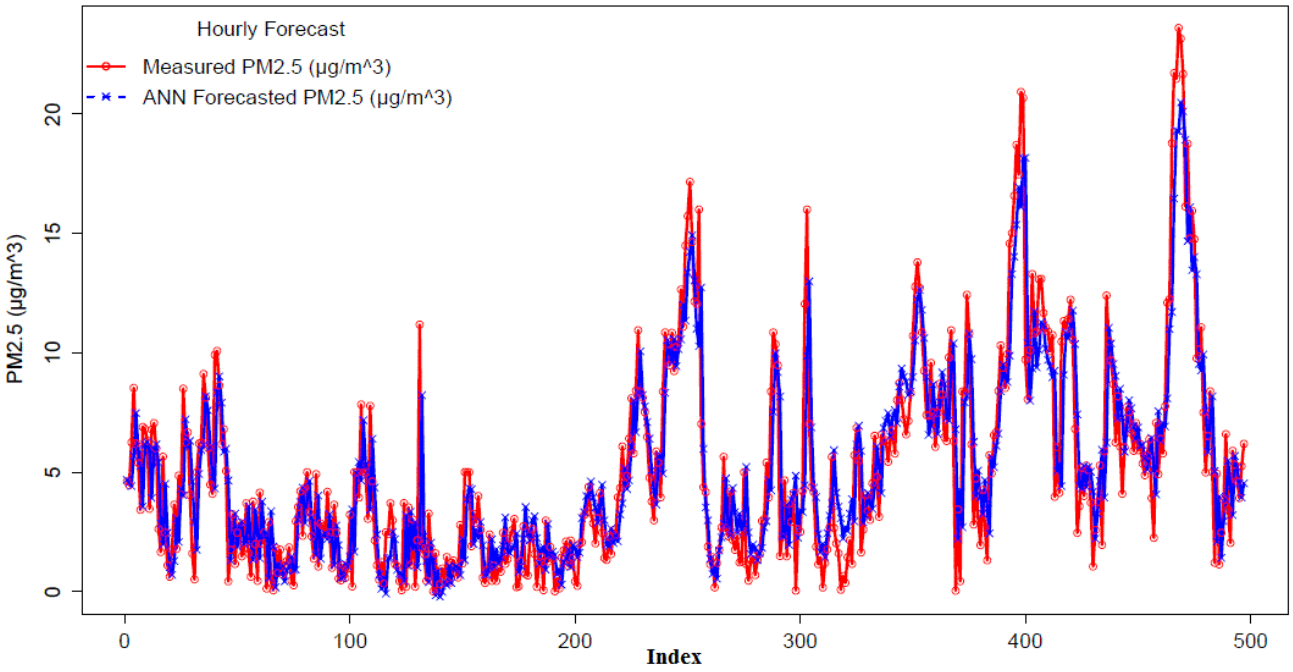


Figure 7-1: Time Series between the Measured and the Predicted values of PM2.5

The graph shows the hourly pollutant concentration of measured and predicted PM2.5 values of the training and validation set from October 3 to October 9 during 2012 to 2014. The X-axis contains the training and validation dataset and the Y-axis consists of the PM2.5 concentration level that has been estimated over a specified time interval. From the graph, we can infer that there is a rising trend in the PM2.5 concentration level from 2012 to 2014.

7.1.2 Prediction of hourly PM2.5 concentration

The Figure 7-2 shows the hourly prediction concentration to that of the measured concentration of PM2.5 for the test dataset for 8-9th October 2014. Inorder check the relationship between measured and the predicted value of the pollutant a correlation check was performed to make

sure that the model had a positive correlation relationship. Also, the prediction trend accuracy[3] was also calculated to make sure the predicted values followed a higher trend of accuracy. The correlation coefficient between the measured and predicted model was around 0.869 which proves that the model has a strong correlation. In addition to the above, the predicted value from the Neural Network (NN) with 9 hidden neurons has a prediction trend accuracy of 0.755, which is fairly a real high level for a relatively small dataset. It can be seen from the graph that the forecasted value of PM2.5 follows a similar pattern as of the measured value of PM2.5.

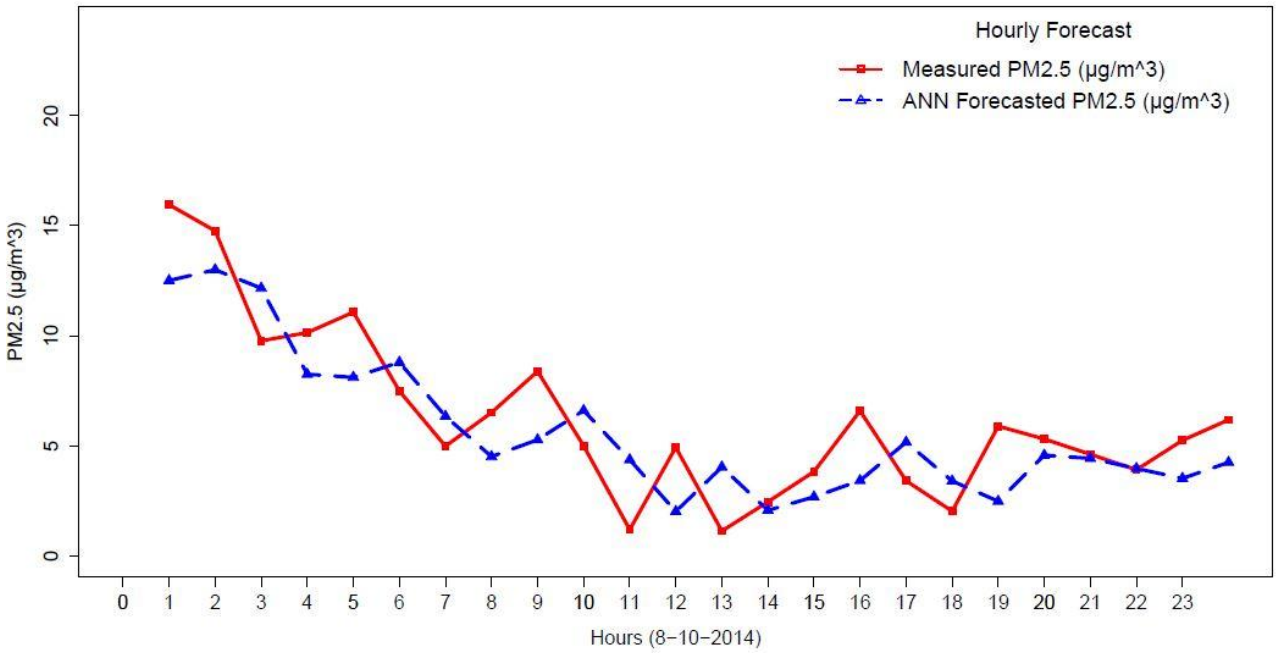


Figure 7-2: Hourly Prediction of PM2.5 for Test Data

The predicted results appear to have a time shift as time increases. This is because during forecasting of future time intervals, the model uses the predicted values as historical information parameters. As time interval increases the results are more based on the predicted values thus leading a small time shift for later time intervals.

[3] measures the average of times the prediction was able to predict the trend correctly

7.1.3 Noise Influenced Analysis

The Noise analysis was performed to take into consideration the fundamental error that might have occurred during different measurements. Different test cases were carried out by adding random error measures to the input factors to check the model efficiency.

A random error of 2%, 5%, and 10% was added and deducted from several input factors and the following test cases results are presented in the below tables.

Table 7-1: Noise Factors: Traffic Volume

Error Percentage	2%	5%	10%
RMSE	0.3	0.312	0.328
Absolute Error	0.249	0.258	0.271
Relative Error (%)	41.34	42.37	43.84

Table 7-2: Noise Factors: Wind Speed, Wind Direction, and Traffic Volume

Error Percentage	2%	5%	10%
RMSE	0.413	0.407	0.424
Absolute Error	0.327	0.338	0.349
Relative Error (%)	54.57	53.81	61.24

It can be seen from the test cases that the model performs well with the different error percentages, and its performance does not decrease rapidly with the increase in the error percentage. Thus, from the test cases, it can be concluded that the model is more robust and can perform effectively up to 10% error percentage.

7.1.4 Comparison with MLR model

The model was compared with a Multiple Regression (MLR) against the same dataset for predicting the concentration value of PM2.5. Therefore, the MLR for pollutant concentration of PM2.5 for its independent variables is the following:

$$PM2.5 = \alpha + \beta_1 * T.V + \beta_2 * D.W + \beta_3 * T.D + \beta_4 * W.S + \beta_5 * W.D + \beta_6 * S.R + \beta_7 * R + \beta_8 * H + \beta_9 * T + \beta_{10} * B.C.1 + \beta_{11} * B.C.2$$

where T.V is the Traffic Volume, D.W is the Day of the Week, T.D is the Time of the Day, W.S is the Wind Speed, W.D is the Wind Direction, S.R is the Solar Radiation, R is the Rainfall, H is the Humidity, T is the Temperature, B.C.1 and B.C.2 are the Background concentration of PM2.5 for 1hr and 2hr respectively. The t-test[4] was performed on the same dataset and the results are presented in the below Table 7-3.

Table 7-3: T and P value for MLR

Attributes	Coefficients	Standard Error	Standard Coefficient	Tolerance	t Stat	P-value
(Intercept)	0.798	0.495	-	-	1.611	0.108
Traffic Volume	0.002	0.001	0.001	0.960	3.695	-
Day of the week	0.109	0.051	0.074	0.991	2.144	0.033
Temperature	0.050	0.033	0.020	0.892	1.508	0.132
Wind Speed	0.089	0.070	0.036	0.988	1.278	0.202
Wind Direction	-0.005	0.002	-0.002	0.785	-3.115	-
Solar Radiation	-0.005	0.001	-0.006	0.950	-3.429	-
Rainfall	-0.033	0.022	-0.118	0.992	-1.470	0.142
Humidity	0.000	0.002	0.000	0.993	-0.221	0.825
Time of the Day	-0.008	0.387	-	-	-0.021	0.983
Pm2.5-2	0.067	0.044	0.058	0.266	1.513	0.131
Pm2.5-1	0.702	0.045	0.607	0.256	15.512	-

The null hypothesis of the t-test proves that contribution made by each variable to that of the dependent variable and from the Table 7-3. It can be concluded that for Multiple Linear Regression (MLR) the traffic volume, wind direction, solar radiation, and humidity does not affect or have an impact on the independent variable (PM2.5).

[4] examines if the means of measured and predicted value are different from each other

A comparison study of the error measures of Multiple Linear Regression (MLR) and the Neural Network (NN) was performed to check the efficiency of each model against the same dataset and the results are presented in the following table.

Table 7-4: Comparison of Error measures between MLR and ANN

	MLR for PM2.5	Neural Network for PM2.5
RMSE	0.275	0.111
Absolute Error	0.223	0.082
Relative Error (%)	40.63	37.23
Correlation	0.816	0.869

It can be seen from the above table that Neural Network (NN) performs better than the Multiple Linear Regression (MLR) models, and there is nearly a 4% difference in the Relative Error between both the models. Therefore, we can conclude that the Neural Network (NN) is more efficient and flexible in dealing with multiple input factors than that of the Multiple Linear Regression (MLR). The hourly predicted results of PM2.5 from both the models on that of the measured values of PM2.5 are depicted in the below Figure 7-3.

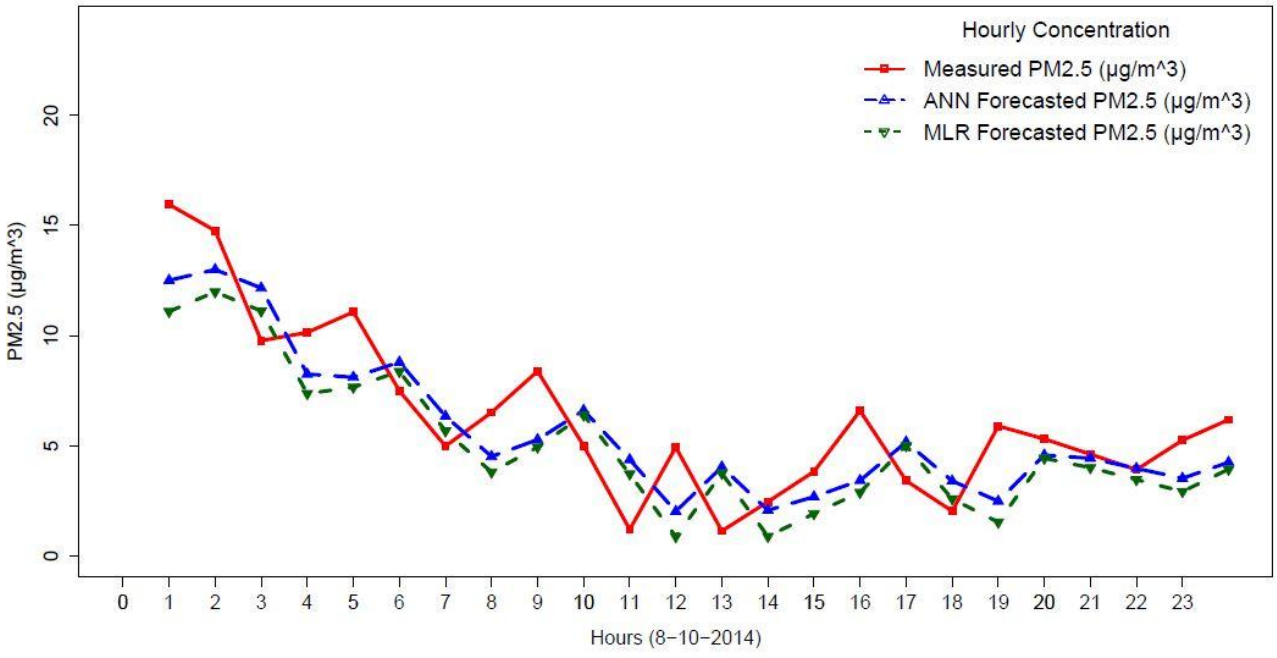


Figure 7-3: Predicted values of ANN and MLR with measure value

7.1.5 Model Transferable Functionality

In this section, an analysis was performed to check how well the model performs for different pollutant concentrations like PM10 and Nitrogen oxides (NO+NO2). The same dataset was used except the pollutant concentrations were changed accordingly to the pollutant concentration chosen.

7.1.5.1 Prediction of hourly concentration for PM10 and NOx

The model was tested against different pollutants that were being monitored by the Stockholm-Uppsala County Air Quality Management [30]. The monitoring station also monitors the pollutant concentration of PM10 and Nitrogen oxides (NO+NO2) other than PM2.5. To extend the feasibility of the model to other pollutants, similar tests were performed for PM10 and Nitrogen oxides (NO+NO2) concentrations. The Figure 7-4 and Figure 7-5 shows the prediction trend of PM10 and Nitrogen oxides (NO+NO2) concentration for every hour respectively.

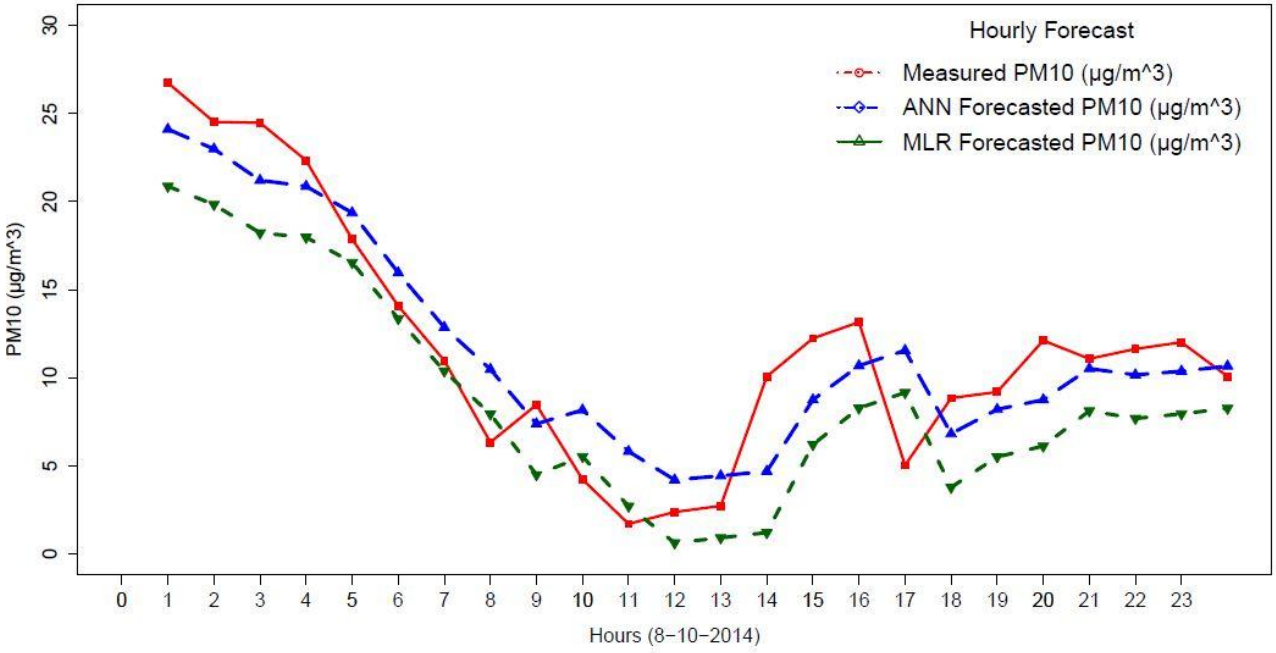


Figure 7-4: Hourly Prediction of PM10 for Test Data

The correlation coefficient for the PM10 and Nitrogen oxides (NO+NO2) was around 0.914 and 0.908 respectively which shows the predicted value and the measured value has a strong

correlation. Also, the Prediction Trend Accuracy for the pollutants PM10 and Nitrogen oxides (NO+NO2) were around 0.777 and 0.691 respectively for nine hidden neurons.

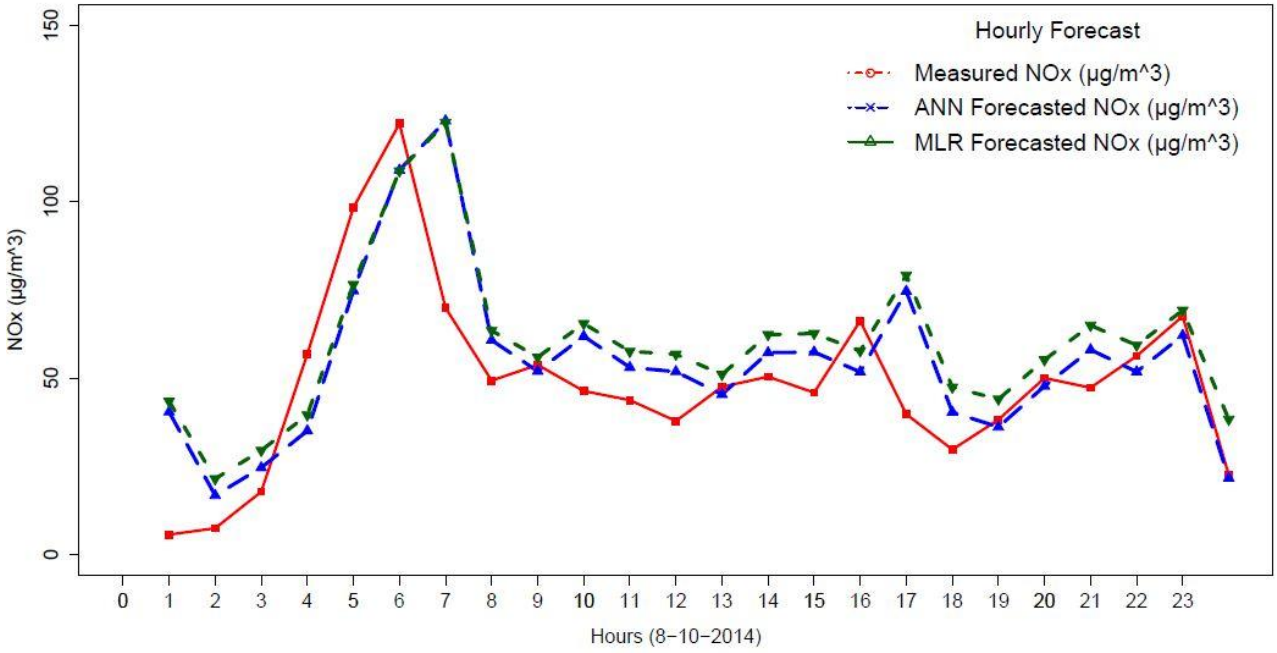


Figure 7-5: Hourly Prediction of Nitrogen oxides (NO+NO2) for Test Data

Thus, from the Figure 7-4 and Figure 7-5 we can conclude that the prediction trend shows a similar pattern on that of the measured value of the pollutant concentrations. Therefore, it proves that the model not only works well with PM2.5 but also with other pollutants like PM10 and Nitrogen oxides (NO+NO2) thus proving its transferability functionality.

Chapter 8

Conclusion

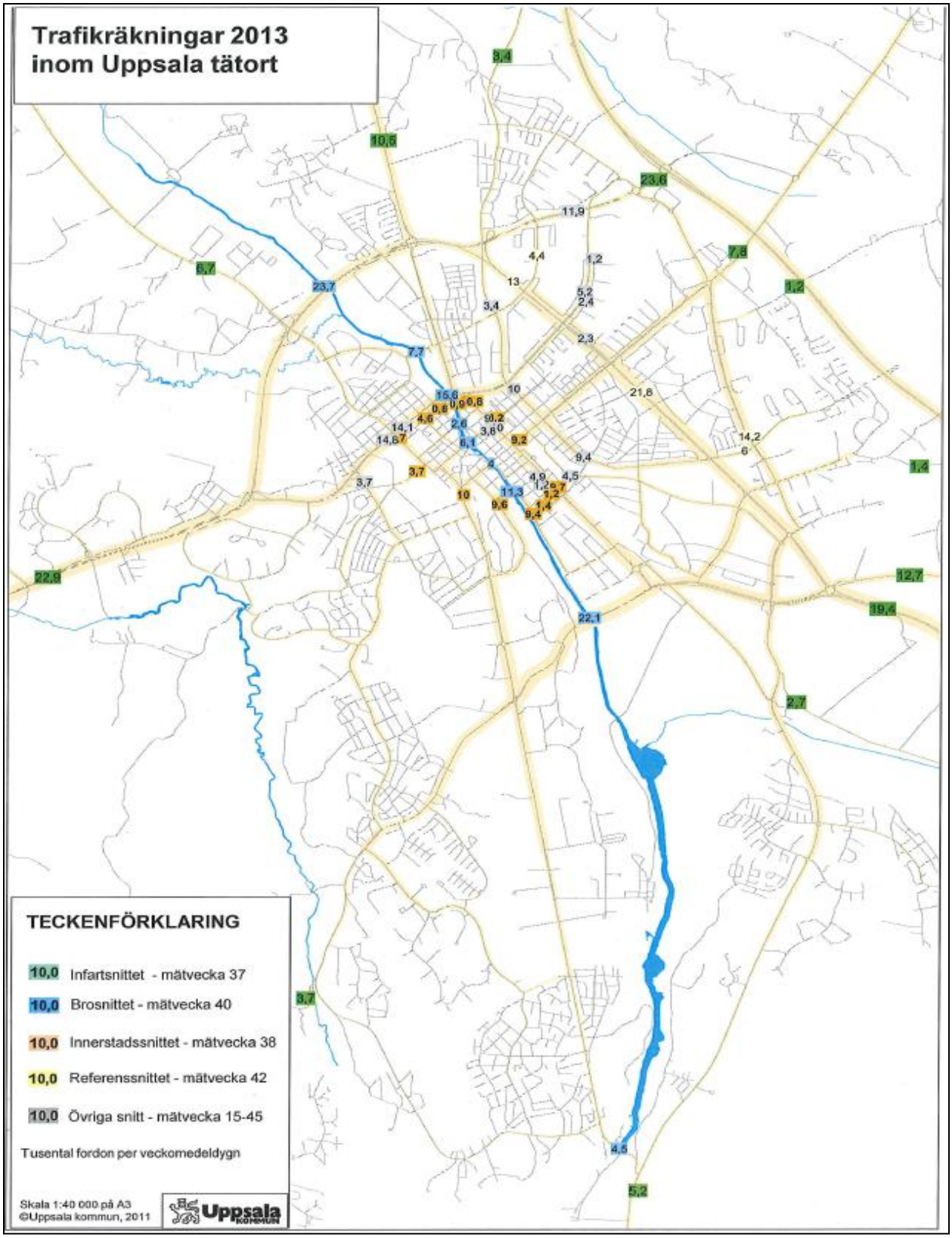
The main purpose of the project was to create a data model and a user interface for the people to interact. The project analyzed various factors that were influential towards the cause of pollution and were separated into three categories namely vehicular, historical and meteorological factors. The input factors of the model were used to forecast not only the PM2.5 pollution concentration but also other pollutants like PM10 and Nitrogen oxides (NO+NO2) thus proving the models transferability function over other pollutant concentrations. Also, various noise analysis tests were performed to check the models feasibility over the different input or measuring errors. The results proved that the models performance deteriorated only when the error rate reached closer to 10 %. Also, the performance analysis proved the Neural Network (NN) was better than the Multiple Linear Regression (MLR) model with a Root Mean Square Error (RMSE) rate of almost 4% in difference and a higher accuracy rate. Thus, the model created is more accurate, easily transferable and has higher resistance to noise in the input factors when predicting the pollution concentration.

In addition to the data model, the developed Android application provided the users with a real-time pollution concentration along with the hourly predicted value of the location. Also, the Android application was able to provide the users with the navigation of a lesser polluted route using the Google driving navigation based on current pollution levels.

For future work, the data model can be bettered if the Uppsala Municipality deploys a permanent vehicle count system over busiest parts of the city thereby making it feasible to make use of more datasets during model creation. Also, once the sensors are deployed, the factors like the height of the sensors placement, the location of the sensors and building aspect ratios surrounding the sensors, etc must be taken into consideration during model creation. The smart route can be further improved if real time on road vehicle data and pollution concentration are available when suggesting the users with an alternate route.

Appendices:

A) Vehicle Count Locations



A-1: Map showing the locations where vehicle counting was performed

Legends

- Infartsvägar (Roads into Uppsala=green) 13 places
- Brosnittet (Bridges=blue) 9 places
- Innerstads/Centrum (Central parts in Uppsala) 14 places
- Reference points 5 places
- Other places around 15 places

Table A-1: Detailed information of the Vehicle count Locations

Nr	Mätpunktsnummer	Snitt	Mätpunktsnamn
1	1391.02	Broarna	Flottsundsbron (1391.02)
2	1183.01	Broarna	Kungsängsbron (1183.01)
3	189.02	Broarna	Islandsbron (189.02)
4	184.02	Broarna	Nybron (184.02)
5	165.02	Broarna	St Olofsbron (165.02)
6	166.02	Broarna	Haglundsbron (166.02)
7	1643.02	Broarna	Luthagsbron västgående 1643.02
8	1643.12	Broarna	Luthagsbron östgående, 1643.12
9	1621.02	Broarna	Fyrisvallsbron (1621.02)
10	534.22	Infarter	Almungevägen Ö Tycho H/Ö Viktoria brandstation (from 96) (534.22)
11	534.03	Infarter	E4 S Gnistarondellen (Kungsängsleden) (534.03) (535.02) Punkt flyttad (-14) pga ombyggd koppling till ICA Maxi
12	553.03	Infarter	Östunavägen S väg 255 (553.03)
13	554.03	Infarter	G:la Stockholmsvägen S Flottsund/S Dag H (554.01,03)
14	1351.03	Infarter	Lurbovägen V Vårdsätrav (1351.03)
15	1003.03	Infarter	Enköpingsvägen V Stenhagsv V infart/V Herrhagsv (1003.03)
16	1003.06	Infarter	1003.06 Enköpingsvägen östgående
17	1512.01	Infarter	Börjegatan N Söderforsgatan (1512.01)
18	1501.01	Infarter	Väg 600 (g:la E4) N Ärnavägen (1501.01)
19	21.01	Infarter	Vattholmavägen N G:la Uppsala/S Vittulsbergsv (21.01)
20	100.02	Infarter	Österleden/Bärbyleden Ö rondellen (100.02) - ny mätpkt 2007
21	100.06	Infarter	Österleden/Bärbyleden östgående
22	102.02	Infarter	Vaksalagatan Ö Österleden (102.02)

23	272.02	Infarter	Alruneg mellan Vitkålsg o Vallby (272.02)
24	542.03	Infarter	Södra Slavstavägen (542.03)
25	13.12	Referenspunkter	Tycho Hedéns väg N Gla Uag (13.12)
26	13.11	Referenspunkter	13.11 Tycho Hedéns väg sydgående
27	236.03	Referenspunkter	Tycho Hedéns väg S Hjalmar Brantingsg (236.03)
28	72.01	Referenspunkter	G:la Uppsalagatan vid von Bahrska häcken (72.01)
29	264.04	Referenspunkter	Fålhagsleden V Fyrislundsg (264.04)
30	264.01	Referenspunkter	Fyrislundsgatan N Fålhagsleden (264.01) (264.08-09 i TDS)
31	1644.03	Innerstaden	Sysslomansgatan S Luthagsespl (1644.03)
32	175.02	Innerstaden	Vaksalagatan Ö Kungsg (175.02)
33	146.02	Innerstaden	S:t Olofsgatan Ö Kungsg (146.02)
34	187.03	Innerstaden	Sjukhusvägen S Munkgatan (187.03)
35	197.01	Innerstaden	Kungsgatan N Strandbodg (197.01)
36	1111.02	Innerstaden	S:t Johannesgatan Ö Rackarbergsg (1111.02)
37	1152.01	Innerstaden	Dag Hammarskjölds väg S Drottningg/N Thunbergsv (1152.01)
38	1591.03	Innerstaden	Svartbäcksgatan S Kungsgatan/S Luthagsesplanaden (1591.03)
39	1592.02	Innerstaden	Kungsgatan N Skolg/S Luthagsesplanaden (1592.02)
40	196.01	Innerstaden	Östra Ågatan N Strandbodg(S Hamnsespl) (196.01)
41	1633.03	Innerstaden	Kyrkogårdsgatan S Luthagsespl (1633.03)
42	1643.03	Innerstaden	Götgatan S Luthagsesplanaden (1643.03)
43	199.01	Innerstaden	Dragarbrunnsgatan N Strandbodgatan (199.01)
44	1930.03	Innerstaden	Kungsängsgatan N Strandbodgatan (1930.03) (fr somm-höst 08)

B) Implementing the Layouts

The Android application uses XML layout to display its contents. The XML view contains several tags with each having its property. The parent tag defines the main view for the document, and there can be several views inside the main view [41]. The XML layouts are accessed in the application by calling it with its "id". Below we will see the different layouts used for the application.

B.1 Main view

The main view of the application contains the navigation drawer with its list items. It is organized with the help of a DrawerLayout statement. The DrawerLayout is linked with the http://schemas.android.com/apk/res/android because it is the parent layout for the main view XML document. The sample for the navigation drawer layout is shown below:

Code Snippet B-1: Navigation Drawer Layout

```
<android.support.v4.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
      android:id="@+id/drawer_layout"
      android:layout_width="match_parent"
      android:layout_height="match_parent" >
      <!-- The main content view -->
      <FrameLayout
      android:id="@+id/frame_container"
      android:layout_width="match_parent"
      android:layout_height="match_parent" />
      <!-- The navigation drawer -->
      <ListView
      android:id="@+id/left_drawer"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_gravity="start"
      android:background="@color/list_background"
      android:choiceMode="singleChoice"
      android:listSelector="@drawable/list_selector" />
</android.support.v4.widget.DrawerLayout>
```

The FrameLayout is the main content and the first child in the DrawerLayout. Its width and height are set to "*match_parent*" because when the Navigation Drawer is absent it needs to fill

up the entire UI. In the List View, the "*android:layout_gravity*" is set to "*start*" to make sure the drawer appears from right to left.

B.2 View Pager

The ViewPager adds screen slide transitions from one page to another [42].ViewPager uses PagerAdapter for passing data created by the fragments. The sample XML code for ViewPager is shown below:

Code Snippet B-2: ViewPager and PagerTab Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
        <!-- activity_screen_slide.xml -->

<android.support.v4.view.ViewPager
        android:id="@+id/vpPager"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

<!--
        This title strip will display the currently visible page title, as well
as the page titles for adjacent pages.
-->
<android.support.v4.view.PagerTabStrip
        android:id="@+id/pager_header"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="top"
        android:background="#B76EB8"
        android:paddingBottom="7dp"
        android:paddingTop="7dp"
        android:textColor="#ffffff" />
</android.support.v4.view.ViewPager>
</LinearLayout>
```

The ViewPager is passed on by calling the ViewPager support library "*android.support.v4.view.ViewPager*" which enables the slide screen activity. The "*PagerTabStrip*" is the child view layout for the "*ViewPager*" in the XML layout and is used for setting the title that can be pinned to the top or bottom by calling "*android:layout_gravity*" to TOP or BOTTOM.

B.3 ListView

ListView is used for displaying a list of items that are scrollable. The items added to the list view are done with the help of an adapter. A custom adapter with two different XML layouts has been used to meet the requirements. One layout holds the ListView *id* and the other layout holds the elements that are to be used for the ListView. The elements are appended with that of the items in the list with the help of Java code. The sample code for ListView is given below:

Code Snippet B-3: ListView Layout

```
<ListView
        android:id="@android:id/list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        tools:listitem="@layout/row" >
</ListView>
```

B.4 MapView

MapView is an Android service that is used for displaying the map with its service layers. The sample code for MapView is given below:

Code Snippet B-4: MapView Layout

```
<com.google.android.gms.maps.MapView
        android:id="@+id/map"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
</com.google.android.gms.maps.MapView>
```

Bibliography

[1] PopulationPyramid.net." PopulationPyramid.net. N.p., n.d. Web. 11 Sept. 2015. <http://populationpyramid.net/world/2020/>

[2] "Frisk Luft." Partiklar I Luft. N.p., n.d. Web. 11 Sept. 2015. < http://www.miljomal.se/Miljomalen/Alla-indikatorer/Indikatorsida/?iid=105&pl=1>

[3] "Air Pollution in Swedish Towns and Cities Must Decrease." Air Pollution in Swedish Towns and Cities Must Decrease. N.p., n.d. Web. 11 Sept. 2015. <http://www.smhi.se/en/research/research-news/air-pollution-in-swedish-towns-and-cities-must-decrease-1.10356>

[4] "Libelium World." Libelium Connecting Sensors to the Cloud RSS. N.p., n.d. Web. 11 Sept. 2015. <http://www.libelium.com/smart_city_environmental_parameters_public_transportation_waspmote/>.

[5] "Libelium World." Libelium Connecting Sensors to the Cloud RSS. N.p., n.d. Web. 11 Sept. 2015. <http://www.libelium.com/smart_city_air_quality_urban_traffic_waspmote/>.

[6] Akyildiz, I.F. et al. "A Survey On Sensor Networks". *IEEE Commun. Mag.* 40.8 (2002): 102-114. Web. 5 Nov. 2015.

[7] Hyndman, Rob J., and George Athanasopoulos. Forecasting: Principles and Practice. Milton Keynes, 2014. Print.

[8] "IDC: Smartphone OS Market Share." Www.idc.com. N.p., n.d. Web. 21 Sept. 2015. <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>.

[9] "About Us – Robotium Tech." About Us – Robotium Tech. N.p., n.d. Web. <http://robotium.com/pages/about-us>

[10] "About Us - RapidMiner." RapidMiner RSS. N.p., n.d. Web. <https://rapidminer.com/us/>.

[11] Armstrong, Scott, J. "PRINCIPLES OF FORECASTING: A Handbook for Researchers and Practitioners." (2001): 1-12. Kluwer Academic Publishers, 2002. Web. 11 Sept. 2015.

[12] Gardner, M., and S. R. Dorling. "Neural Network Modelling and Prediction of Hourly NOx and NO2 Concentrations in Urban Air in London." Atmospheric Environment 33.5 (1999): 709-19.

[13] Pérez, Patricio, Alex Trier, and Jorge Reyes. "Prediction of PM2.5 Concentrations Several Hours in Advance Using Neural Networks in Santiago, Chile." Atmospheric Environment 34.8 (2000): 1189-196.

[14] Ballester, E. Balaguer, G. Camps I Valls, J.l Carrasco-Rodriguez, E. Soria Olivas, and S. Del Valle-Tascon. "Effective 1-day Ahead Prediction of Hourly Surface Ozone Concentrations in Eastern Spain Using Linear Models and Neural Networks." Ecological Modelling 156.1 (2002): 27-41.

[15] Kukkonen, J. "Extensive Evaluation of Neural Network Models for the Prediction of NO2 and PM10 Concentrations, Compared with a Deterministic Modelling System and Measurements in Central Helsinki." Atmospheric Environment 37.32 (2003): 4539-550. Web. 11 Sept. 2015.

[16] Brunelli, U., V. Piazza, L. Pignato, F. Sorbello, and S. Vitabile. "Two-days Ahead Prediction of Daily Maximum Concentrations of SO2, O3, PM10, NO2, CO in the Urban Area of Palermo, Italy." Atmospheric Environment 41.14 (2007): 2967-995. Web. 11 Sept. 2015.

[17] "Air Quality Index (AQI) Basics." Air Quality Index (AQI) Basics. N.p., n.d. Web. 31 Dec. 2015. <http://airnow.gov/index.cfm?action=aqibasics.aqi>.

[18] Mintz, David. Guideline for Reporting of Daily Air Quality: Air Quality Index (AQI). Research Triangle Park, NC: U.S. Environmental Protection Agency, Office of Air Quality Planning and Standards, 2006.

[19] Piatetsky-Shapiro, Gregory, and William Frawley. Knowledge Discovery in Databases. Menlo Park, CA: AAAI, 1991.

[20] Prieditis, Armand, and Stuart J. Russell. Machine Learning: Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, July 9-12, 1995. San Francisco, CA: Morgan Kaufmann, 1995.

[21] Valpola, Harri. "Supervised vs. Unsupervised Learning." Supervised vs. Unsupervised Learning. N.p., n.d. Web. 05 Dec. 2015.

[22] "Android Overview | Open Handset Alliance." Android Overview | Open Handset Alliance. N.p., n.d. Web. 08 Sept. 2015. <http://www.openhandsetalliance.com/android_overview.html>.

[23] "Anatomy & Physiology of an Android - 2008 Google I/O Session Videos and Slides." N.p., 2008. Web. 08 Sept. 2015. <https://sites.google.com/site/io/anatomy--physiology-of-an-android>.

[24] "Android 6.0 Marshmallow." Android Developers. N.p., n.d. Web. 08 Sept. 2015. <http://developer.android.com/>.

[25] Guoqiang, Zhang, Eddy Patuwo B, and Hu Y. Michael. "Forecasting with Artificial Neural Networks:: The State of the Art." International Journal of Forecasting 14.1 (1998): 35-62.

[26] "App Manifest." App Manifest. N.p., n.d. Web. 08 Sept. 2015. <http://developer.android.com/guide/topics/manifest/manifest-intro.html>.

[27] "Logcat." Log cat. N.p., n.d. Web. 08 Sept. 2015. <http://developer.android.com/tools/help/logcat.html>.

[28] "Introducing JSON." JSON. N.p., n.d. Web. 08 Sept. 2015. <http://json.org/>.

[29] Battiti, Roberto. "First- and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method." Neural Computation 4.2 (1992): 141-66. Web. 14 Sept. 2015.

[30] "Stockholm and Uppsala Air Quality Management Association." Stockholm and Uppsala Air Quality Management Association. N.p., n.d. Web. 13 Oct. 2015.

[31] "Portable Traffic Survey - MetroCount." MetroCount. N.p., n.d. Web. 13 Oct. 2015.

[32] Cai, Ming, Yafeng Yin, and Min Xie. "Prediction of Hourly Air Pollutant Concentrations near Urban Arterials Using Artificial Neural Network Approach." Transportation Research Part D: Transport and Environment 14.1 (2009): 32-41. Web. 14 Sept. 2015.

[33] Swingler, K., 1996. Applying Neural Networks: A Practical Guide. Academic Press, London. Web. 14 Sept. 2015.

[34] Hornik, K., Stinchcoombe, M., White, H., 1989. Multi layer feed forward networks are universal approximators. Neural Networks 2, 359–366. Web. 14 Sept. 2015.

[35] Karlik, Bekir, and Vehbi A. Olgac. "Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks". International Journal of Artificial Intelligence and Expert Systems (IJAE), n.d. Web. 14 Sept. 2015.

[36] Gallant, S.i. "Perceptron-based Learning Algorithms." IEEE Trans. Neural Netw. IEEE Transactions on Neural Networks 1.2 (1990): 179-91.

[37] Gardner, M. "Neural Network Modelling and Prediction of Hourly NOx and NO2 Concentrations in Urban Air in London." Atmospheric Environment 33.5 (1999): 709-19. Web. 14 Sept. 2015.

[38] Vong, Chi-Man, Weng-Fai Ip, Pak-Kin Wong, and Jing-Yi Yang. "Short-Term Prediction of Air Pollution in Macau Using Support Vector Machines." Journal of Control Science and Engineering 2012 (2012): 1-11. Web. 14 Oct. 2015.

[39] "Creating a Navigation Drawer." Creating a Navigation Drawer. N.p., n.d. Web. <https://developer.android.com/training/implementing-navigation/nav-drawer.html>.

[40] "Fragments." Fragments. N.p., n.d. Web. <http://developer.android.com/guide/components/fragments.html>.

[41] "Layouts." Layouts. N.p., n.d. Web. <http://developer.android.com/guide/topics/ui/declaring-layout.html>.

[42] "ViewPager." ViewPager. N.p., n.d. Web. <http://developer.android.com/reference/android/support/v4/view/ViewPager.html>.

[43] "Multiple Linear Regression." Multiple Linear Regression. N.p., n.d. Web. 14 Oct. 2015.

[44] Kolehmainen, M., H. Martikainen, and J. Ruuskanen. "Neural Networks and Periodic Components Used in Air Quality Forecasting." Atmospheric Environment 35.5 (2001): 815-25.

[45] Air Info Now: What Is Particulate Matter. N.p., n.d. Web. 31 Dec. 2015. <http://www.airinfonow.org/html/ed_particulate.html>.