BIG DATA ANALYTICS FOR AIR QUALITY MONITORING ASSESSMENT BASED ON IoT PLATFORM

Desislava Ivanova[1], Angel Elenkov[2]

Technical University of Sofia,
Faculty of Applied Mathematics and Informatics[1],
Faculty of German Engineering and Economics Education[2]
e-mails: d_ivanova@tu-sofia.bg[1], angel.elenkov@fdiba.tu-sofia.bg[2]
Bulgaria

Abstract: The paper is proposed a machine learning algorithm for IoT platform, Paspberry Pi to predict the values of various pollutants in the air based on big data analytics. The conceptual model of the proposed system and experimental framework is designed for the case of verification and validation of the proposed machine learning model for air quality monitoring and assessment using the datasets for the last five years provided by the Executive Environment Agency (EEA), Ministry of Environment and Water, Bulgaria.

Key words: IoT, Big Data Analytics, Air Quality Monitoring

1. INTRODUCTION

The goal of this paper is to develop an intelligent system for air quality monitoring assessment using the Raspberry Pi platform and machine learning algorithm developed using the Python programming language. The paper fully encompasses the idea of the Internet of things (IoT) where all the devices are interconnected via Internet. In the case of this paper, the Raspberry Pi has Internet connection, will receive data from sensors, and can send it over the internet if we want to [1, 2]. Currently the collected and predicted data is saved locally, but can be sending so it conforms to the definition [3, 4]. The system will take as inputs the measured values from sensors for temperature, humidity, and air pressure, and using a neural network regressor outputs predicted values for these pollutants: nitric oxide, nitrogen dioxide, ozone, and PM10 (the fraction of particles with an aerodynamic diameter smaller than 10 μm) [5].

The pollution problem is getting more and more up-to-date when we check the European air quality index showing the current pollution in a wide range of European countries (European Air Quality Index). The statistics has been shown that the air in

all biggest cities in Europe is with heavily polluted. This shows that the problem is very serious and needs caution.

In the next sections of the paper will be summarized the conceptual model of the proposed system and the developed machine learning algorithm. Finally, the experimental result will be presented and analyzed.

2. CONCEPTUAL MODEL OF THE PROPOSED SYSTEM FOR AIR QUALITY MONITORING BASED ON BIG DATA ANALYTICS

The conceptual model of the system is subdivided in four levels. The bottommost layer contains the Raspberry Pi with the GrovePi+ board (Hardware level), which is needed to connect the sensors to the system, and the sensors themselves. The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It can do everything a normal desktop computer can. It is also a great IoT platform, on which people can learn to program in Scratch or Python. As it can be seen on the above image, it has all the connectors that a normal desktop computer has, which allows for a realization of desktop specific tasks on this small machine. Its size makes it easily mobile. On the figure below a block diagram can be found, which shows how the individual components of the system are interconnected.

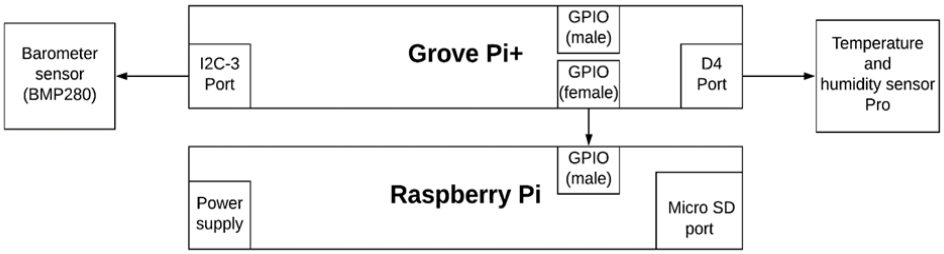

Fig. 1. Connectivity of individual components of the proposed system

One level above of the designed system sits the operating system. The Raspbian for Robots operating system is used for the system. It manages the communication between the system and the software, and also the resources of the system. This OS is a version of Raspbian, which is made by Dexter Industries. It is a free operating system based on Debian optimized for the Raspberry Pi hardware. It is also the official operating system supported by the Raspberry Pi foundation. So, the Raspbian for robot's OS is simply a customized version of the officially supported OS. The biggest advantage of using the "for robots" version is that it has all the software you need built in to connect to your GoPiGo, BrickPi, GrovePi (Grove Pi+) or Arduberry and get you programming in just a few minutes. For the goal of the paper, the GrovePi+ needs all of these software configurations of the OS.

Above the operating system level, sits the software one. It includes all the scripts, written in Python, which accomplish the goal of the system. The scripts can be divided into three main parts: *data processing scripts*, *sensors I/O scripts'*, *machine learning model scripts.*

First are the data processing scripts, which are written to transform the data to be easily fed into the machine learning algorithm, fig. 2. For that purpose, it is necessary the following parameters: date, NO, NO2, AirTemp, Press, and UMR columns from the data file, containing hourly measures to be concatenated with the corresponding date, O3, and PM10 from the file, containing daily measures.

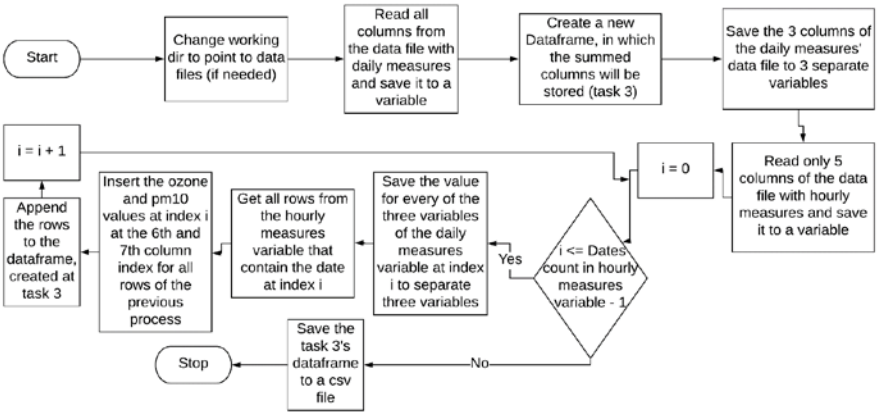

Fig. 2. Flowchart of data processing script

The sensors I/O scripts are bound to reading data from the sensors and managing this data. The needed packages for this script are: *Grovepi*, *Math*, *Adafruit_BME280.* Grovepi is the package, which allows for direct reading of the Temperature and humidity sensor pro's values. Math is needed for mathematical functions. The last import – Adafruit_BME280, is the library needed to create the barometer sensor as an object and to read from it.

Topmost of the conceptual model is the user interface level. There the scripts are located. They realize the two user interfaces. The first one acts as a monitor, which shows rows of information about measured values, and their predicted outputs. The second UI can be used to enter the three inputs (Temperature, Humidity, and Pressure) to check the machine learning model's prediction about them.

The datasets that are used for training and validation of the machine learning algorithm in the designed system has been provided by the Executive Environment Agency (EEA), Ministry of Environment and Water, Bulgaria [6]. When collecting and processing multiple data, it is taken into account the challenges of the contemporary digital age for the individuals' privacy and personal data protection [7, 8].

3. INTELLIGENT ALGORITHM BASED ON MACHINE LEARNING FOR AIR QUALITY MONITORING ASSESSMENT AND UI

The system uses a Multi-layer Perceptron (MLP) regressor as a machine learning model. This model can learn a non-linear function to solve classification and regression problems. In the case of this system, regression is used, because the main difference is that classification is the task of predicting a discrete class label, where regression is the task of predicting a continuous quantity [9, 10, 11].

This model can learn a non-linear function to solve classification and regression problems. In the case of this paper, regression is used, because the main difference is that classification is the task of predicting a discrete class label, where regression is the task of predicting a continuous quantity. For the realization of this paper, the output values are not discrete. The outputs that are used for the model for the realization of the intelligent system can vary. They are random quantities and cannot be made discrete.

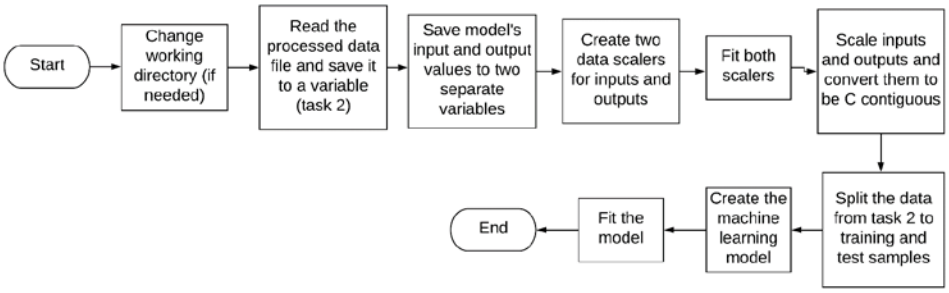

Fig. 3. Flowchart of machine learning algorithm script

The Multi-layer Perceptron regressor implementations from scikit-learn trains using backpropagation, and has no activation function at the output layer. The network outputs some values for each of the 4 pollutants and these values do not need to pass any activation. The activation function is simply the likelihood of a certain perceptron firing. By this, we mean if the outputs were discrete – say 0 and 1 (this means 2 perceptrons at the output layer), and after giving an input (after the training of the network is complete), this input propagates forward and at the output two different activations of the 2 output neurons are calculated. If the activation for 0 for example is bigger, this means that the network thinks that for this input the 0 value corresponds as output. The main purpose of the MLP network (our model) is to minimize the cost function. This function simply states how big the difference between the predicted output values and their true values is. The flowchart of the script is presented on fig. 3.

For the software implementation of the code, the following libraries are needed: *Os, Pandas, Numpy, StandardScaler, train_test_split, MLPRegressor*. Os is again needed to read and use files that are not in the current working directory. It is optional to use it. Don't if the NONANSSUMMEDANDREADY.csv file is in the same

directory as the script. Pandas is again used for file manipulations, where numpy for array transformations. Some new libraries that were not used until this time are the last three. They are actually subscripts, so to say, from sklearn [12].

The StandardScaler library is needed to scale the input and output values. This is done, because MLP networks are sensitive to scaling. In other words, big differences in values from different inputs can lead the network to train inaccurately, because it needs to figure relationships between inputs and outputs, and big differences can influence the relationships' establishment. To use the scaler, it needs to be fit on some data. The fitting calculates the mean and standard deviation, so the scaler can be used later to scale data. After fitting the inputs and outputs, both n-dimensional arrays containing the values for inputs and outputs are scaled and converted to C contiguous. The values are scaled around 0. The conversion to C contiguous is done to speed up the reading from memory. C contiguous means that every row of the n-dimensional array is stored in the next memory cell [10, 12].

The package train_test_split is used to generate train and test values for the model. The parameters that are passed to the function specify the inputs array, outputs array, percentage of the samples left for test (20% in the case of this paper), and the random_state parameter. This random_state parameter is only used to tell the function that we need every split to be exactly the same (at every run of the script). This helps to tune up the model.

Next, the MLP regressor is defined. It is easily done by creating an MLPRegressor object. The parameters passed depend hardly on the task. Note: The chosen parameters will change with time, because the model will be tuned as more data comes. The new data that will come will be partially fitted by the partial_fit method. It allows improving the model by feeding more data to the already trained model. Because of this incremented learning and future validating of the intelligent system the parameters passed as seen in the code are not final. For now, the network has 10 perceptrons in the hidden layer. We need to pass only those parameters, which we do not need as default. This script can be used to tune the model and also to save it. The random_state parameter is here again, to make every execution the exact same.

The main job of next script is to implement a function that reads the sensors' values and return a row that contains the read values from the sensors and predicted values for those read values. The libraries that are needed for this script are simply our Sensors_read.py file, time for some time operations, joblib's load function, numpy, and math.

At the beginning of the script, our model with our two scalers is loaded from files to variables, so we can use them. These models were dumped to files after executing the last script. Next is the definition of our function ReadAllSensors. It starts with storing the temperature and barometer sensors measured values. The values of temperature, humidity, and pressure are stored directly in a numpy array, which becomes of shape 1, 3 (1 row, 3 columns). This shape is exactly the same as this of the inputs, with which the intelligent system is trained with (AirTemp, UMR,

and Press are passed as x to the machine learning model). Before using the predict function of our model, the array's values are scaled and saved to a new array. After the scaling the output values (NO, NO2, O3, PM10) are predicted and stored in a variable (pr in this case). After applying inverse transform (because the predicted values are scaled, they need to be returned to non-scaled state) the print variable holds the predicted values for NO, NO2, O3, and PM10.

The next row formats a list, which has as elements the time (we need to know when the measurement happened), temperature, humidity, pressure, and the three predicted values, each rounded to 2 decimals after the decimal point. At the end we simply return this list, which is used by the UI.

At the top of our conceptual model sits the user interface, which has two scripts, each one being a different UI. The first one acts as a monitor, which shows rows of information about measured values, and their predicted outputs. The second UI can be used to enter the three inputs (Temperature, Humidity, and Pressure) to check the machine learning model's prediction about them. The UI is created using the tkinter python library, which is entirely for building UIs. The topmost and bottommost rows are embedded in the main window, because scrolling of the data is implemented. The inner window, where the 2nd and 3rd row can be seen is a canvas object, with a frame created on top of it. This is needed to implement scrolling. For the realization of this intelligent system, we used only vertical scrolling. The scrollbar can be seen on the top right corner under the close button. All elements that can be seen on this UI are positioned using the grid() method of tkinter, with which a position for an object can be specified by giving a row and a column numbers. The second UI proposed in the paper is PredictValueUI, fig. 4.

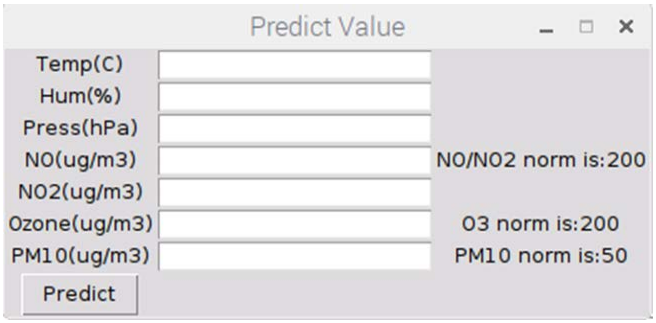

Fig. 4. Predict Value UI

This UI simply allows the user to use the machine learning model to predict values for NO, NO2, O3, and PM10 by inputting values for temperature, humidity, and pressure.

4. RESULT ANALYSIS AND CONCLUSION

The performance of the trained model is realized by the r^2 (r squared) score, the coefficient of determination. It describes how much of the output variable's variation

is described by the variation of the input variable. It can take negative values, but mainly it is between 0 and 1. Negative values mean a bad model. Basically, the r squared score shows how good the model is.

The r^2 score is calculated by the function score of our object regr by calculating 1 - u/v. Here u is the residual sum of squares ((y_true - y_pred) ** 2).sum() (summed) and v is the total sum of squares ((y_true - y_true.mean()) ** 2).sum().

Our model has an r^2 score of about 0.65. R^2 score returns the coefficient of determination. R^2 simply gives a measure of how far our predicted values are from the true ones. In other words, the bigger the r^2 score, and the better the model is.

The achieved results mean that the model is good, but the expectations are that with future incremented learning using the partial_fit method and tuning the parameters more if needed will improve the overall r^2 score.

The results for the MLP model are obtained by a lot of parameter tuning. This includes trying to adjust every different parameter and check what its influence to the output result of the model will be. Further data feeding into the model by using incremental learning can improve the coefficient of determination, but only further tests can prove this.

The goal of this paper is fully corresponds to the goals of IoT intelligence with ML and brings a system to the real world, which can help in preventing air pollutions. This paper realizes an intelligent system based on IoT platform using the machine learning algorithm for big data analytics with respect to air quality monitoring and assessment to predict the quantities of NO, NO2, O3, and PM10 air pollutants using temperature, humidity, and pressure as inputs, which are received from sensors. The results for the MLP model are obtained by a lot of parameter tuning. This includes trying to adjust every different parameter and check what its influence to the output result of the model will be.

We plan on improving the machine learning algorithm with new data, which will be obtained from the Executive Environment agency as the training data was. The predictions are also going to be implemented in real scenarios and validated with data from the Executive Environment agency with time to see if the model is getting better. The inputs and outputs of the machine learning algorithm can change with time, some outputs or inputs can get dropped out, but the basic idea of how such a system should work is described in detail in this paper.

REFERENCES

[1] What is PM10 and PM2.5?; *http://www.irceline.be/en/documentation/faq/what-is-pm10-and-pm2.5*

[2] Brownlee J., Difference Between Classification and Regression in Machine Learning, S. et al. *Monitoring the dynamic, 2017; URL: https://machinelearningmastery.com/ classification-versus-regression-in-machine-learning/*

[3] Khot R., Chitre V., Survey on air pollution monitoring systems; *Proceedings of the 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*; Coimbatore, India. 17–18 March 2017; Piscataway, NJ, USA: IEEE; 2017. pp. 1–4.

[4] Yi W.Y., Lo K.M., Mak T., Leung K.S., Leung Y., Meng M.L., A survey of wireless sensor network based air pollution monitoring systems, *Sensors. 2015; 15:31392–31427*, DOI: 10.3390/s151229859.

[5] Alvear O., Zamora W., Calafate C.T., Cano J.C., Manzoni P., EcoSensor Monitoring environmental pollution using mobile sensors; *Proceedings of the 2016 IEEE 17th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Coimbra, Portugal. 21–24 June 2016.

[6] Executive Environment Agency http://eea.government.bg/en

[7] Romansky, R. Opportunities of the Digital Space and Challenges for Privacy and Individual's Security". *Proceedings of the 31st International conference on Information Technologies*, Bulgaria, 20-21 Sep. 2017, pp. 169-178.

[8] Romansky, R. A Survey of Digital World Opportunities and Challenges for User's Privacy. *International Journal on Information Technologies and Security, ISSN 1313-8251*, No. 4, vol. 9, 2017, pp. 97-112.

[9] Nielsen M., 2015, Neural Networks and Deep Learning, http://neuralnetworksanddeeplearning.com/

[10] Riley A., 2017, https://stackoverflow.com/questions/26998223/what-is-the-difference-between-contiguous-and-non-contiguous-arrays

[11] Oakley B., 2017, https://stackoverflow.com/questions/3085696/ adding-a-scrollbar-to-a-group-of-widgets-in-tkinter/3092341#3092341

[12] sklearn.neural_network.MLPRegressor: https://scikit-learn.org/stable /modules/generated/sklearn.neural_network.MLPRegressor.html

Information about the authors:

Desislava Ivanova – She is a vice head of Informatics Department, Faculty of Applied Mathematics and Informatics, Technical University of Sofia. Her areas of research are HPC, Big Data, Cloud Computing and IoT.

Angel Elenkov – He is a master student at Faculty of German Engineering and Economics Education, Technical University of Sofia.

Manuscript received on 30 April 2019