Royaume du Maroc
Haut Commissariat au Plan
Institut National de Statistique
et d'Économie Appliquée
INSEA

Subject:

An Artificial Intelligence Model for Explosives Detection

Produced by : ZOUHRI Yassine
MOUZAKI Youness
Supervised by : BENELALLAM Imad
ERRAJI Zakariae

Dedication

To our dear parents For all their sacrifices,

. . .

Acknowledgement

We would like to thank first of all our school; The National Institute of Statistics and Applied Economics in Rabat, ...

...

National Institute of Statistics and Applied Economics

Rabat, November 2021

Abstract

Hidden explosives detection is a priority for defence and security for every country as search by humans or dogs is not a good situation as the technology used in explosives is very advanced. In this project, the aim is to design an electronic nose to detect the explosives which will be accurate, reliable, fast, simple and small using data mining techniques. This is a classification problem, for detection of explosives along with deep learning algorithms like LSTM and this project develops the machine learning algorithms to evaluate the proposed work. The proposed work is implemented using software platforms such as Anaconda (Python), Sklearn and Keras. The main objective of proposed work is four folded: 1) Understanding the basic idea of which chemicals are contributing towards an explosive nature 2) Use data from different types of sensors to detect 3) Improve the results of the detection by the sensors.4) Build models using different classification techniques and LSTM and evaluate the performance. The main aim is to achieve accurate detection of explosives using a profound LSTM network. So we should understand how LSTM(s) works with explosives detection, in fact LSTM refers to Long Short Term Memory which means that the past informations are saved and used for the next steps, for an electronic nose the human brain distinguishes between smells based on historical smells saved on the synapses of our brain, as said in the IEEE article : "In these networks a state of what the network has seen so far is kept and used during the learning process", Thus, LSTMs mimic this wonderful natural event, in fact, every time our model finds the same response from the sensors (detectors), it is the same smell and that is why we substitute the convolutional process using LSTM cells in order to have a higher precision when treating time-limited records, the activation functions used are the same as for the convolutional neural network.

Keywords – Decision Tree, Random Forest, KNN, CNN, LSTM

Contents

1 Introduction 1
1.1 Odor recognition 2
1.2 Artificial intelligence 3
1.3 Aims and objectives 4
1.4 Problematic 4
2 Literature review 6
2.1 Odour 6
1.i Characteristics of odours 7
1.ii Odour parameters 7
2.2 Classification of explosive detection techniques 8
2.3 Deep learning based explosion techniques 9
2.4 Summary of progress to date 15
2.5 Dataset: mixed explosives dataset 15
3 Project plan 16
3.1 Tasks involved in the proposed Work 16
3.2 Prototype of the proposed work 18
4 Methodology 19
4.1 Data collection 19
4.2 Technologies tools 20
4.3 Methodology 23
5 Experimental setup and analysis procedure 24
5.1 Data analysis 24
5.2 Explosives analysis 24
5.3 Adopted algorithms 24
3.i Machine learning models 24

3.ii Deep learning models 25
5.4 Algorithms application process 26
5.5 Models efficiency evaluation 27
5.i Accuracy 27
5.ii Precision 28
5.iii Recall 28
5.iv F1-score 28
6 Result and interpretation 29
6.1 Normalization and feature scalling 29
6.2 Data analysis results 30
6.3 Implementation of Machine Learning Models 31
3.i Decision tree 32
3.ii Random Forest 32
3.iii K nearest neighbours 33
6.4 Implementation of Deep Learning Models 34
4.i Convolutional Neural Network 35
4.ii Long Short Term Memory 36
6.5 Model comparaison 38
6.6 Models comparison at various timesteps 39
7 Conclusion 40
References 41

Annexe **58**

List of Figures

1.1 Henning's scented prism 2
4.1 Input data (Sensors Data) 19
4.2 Binary output data 20
4.3 Anaconda Logo 21
4.4 Methods used 23
5.1 Developing Process 27
6.1 Scaled Data 29
6.2 Scaled/Normalized Data 30
6.3 Sensors reaction over time 30
6.4 Scaled/normalized sensors reaction over time 31
6.5 Final Data 31
6.6 Metrics report and Confusion Matrix for Random Forest model 32
6.7 Metrics report and Confusion Matrix for Decision Tree model 33
6.8 Metrics report and Confusion Matrix for knn model 34
6.9 Metrics report and Confusion Matrix for CNN model 35
6.10 Accuracy and loss plots for CNN model 36
6.11 Metrics report and Confusion Matrix for LSTM model 37
6.12 Accuracy and loss plots for LSTM model 38

List of Tables

6.1 Method used 38
6.2 Accuracy over different time intervals 39

1 Introduction

It has always been necessary to identify potential threats. The main threats identified are explosives, chemicals, and biological substances, due to their severity. The main challenges facing security and defense personnel are the detection of hidden explosives and illicit chemicals. Conventional explosives detection techniques are no longer viable solutions due to their own limitations. It is therefore necessary to develop a fast and reliable detection technique that provides accurate results. Among the human senses, the sense of smell is probably one of the most mysterious. The basics of how it works have only recently been elucidated: it was only in 1991 that biologists Linda Buck and Richard Axel, from Columbia University in the United States, discovered the olfactory receptor gene family, a discovery that earned them the Nobel Prize in Physiology or Medicine in 2004. Within our brain, the sense of smell has its own function. "An amazing property is that we manage, from a limited number of olfactory receptors, only 400 different ones, to recognize at least tens of thousands of odors," explains Edith Pajot, director of research at INRA (National Institute for Agricultural Research). In fact, each of these sensors located at the bottom of the nasal cavity is sensitive to several molecules. "The activation map of the different neurons of the olfactory cortex forms a sort of unique signature for this odor. The brain then uses memory to match this signature to those of previously known odors," summarizes the researcher.

The important thing is the idea of the electronic nose which consists in imitating this principle of analysis known as combinatorial. In concrete terms, this involves aligning a variety of sensors specific to different molecules on a support in contact with the surrounding gases. For example, the French company Aryballe Technologies produces an electronic nose called NeOse, which contains a total of 100 miniature sensors. "When this device is placed near a source of odor, the molecules responsible for that odor attach themselves to the sensors and change their physical or electrical state. This change of state can be recorded as a signal, which is the signature of that molecule," explains Thierry Livache, the company's scientific director. Several works have been published on the use of electronic noses in specific applications. However, there is not a large body of work on artificial noses that can be used in many applications. In this paper, our AI

system aims to detect and recognize explosives using an electronic nose. We will discuss some ML algorithms and compare them with one or two deep learning algorithms such as LSTM. We also mention that our e-nose uses one of the existing ML/DL models.

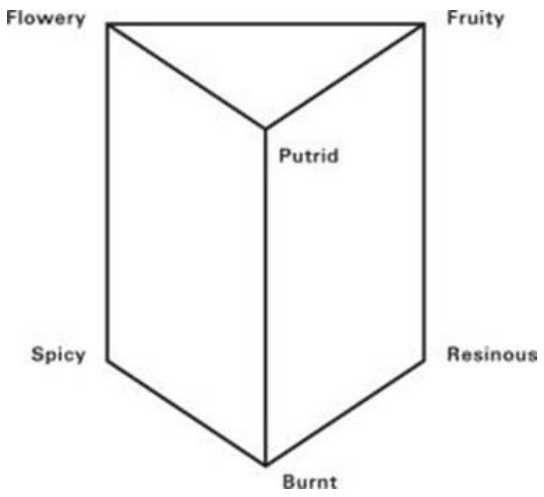
The goal of this research is to develop accurate, fast LSTM neural network models that can accurately detect explosive substances, even when mixed with confounders.

1.1 Odor recognition

An electronic nose is a fast, non-invasive and intelligent on-line instrument. It consists of a set of carefully selected sensors and an appropriate pattern recognition system capable of identifying particular odors. There are different types of e-noses in different application areas, based on commercially available sensors. Nevertheless, there are many challenging questions regarding performance improvement, including improved classification rate, speed of online detection, and accuracy of recognition and prediction.[17]

Odor detection or recognition is one of the most challenging fields in the world today. For example, it is used to ensure airport security by detecting dangerous chemicals, to help manual workers detect explosives or drugs, or to assess the quality of honey in agriculture. Odor detection is very useful to detect odors like floral, fruity, spicy, etc... as shown in the figure below, without using the detection capabilities of the human nose. This can be done in different ways.

Figure 1.1: Henning's scented prism



In practice, an electronic nose consists of three main parts:

- Sampling system.
- Système de détection.
- Data processing system.

Among the techniques that an odor detection system can have, we found:

- Metal oxide sensors (often sensitive to humidity) ;
- Conductive polymers;
- Piezoelectric quartz sensors (CQP, SAW);
- Field effect sensors (MOSFET);
- Fingerprint mass spectrometry;
- Ultra-fast gas chromatography;
- Eectrochemical cells for target gas analysis (NH3, H2S, RSH, CO2, etc.) in the field;
- The photoionisation detector (PID) for the analysis and quantification of Volatile Organic Compounds (VOCs) in the field;
- A biosensor with electronic and chemical properties equipped with mammalian olfactory receptors.

1.2 Artificial intelligence

Artificial intelligence (AI) is a set of technologies that allow machines to simulate some sort of real intelligence. Artificial intelligence is being implemented in a growing number of application areas. The principle was invented in the 1950s by the mathematician Alan Turing. In his book Computing Machinery and Intelligence. Overall he proposed the Turing test, in which a subject communicates with a human and then with a machine which has been adapted to give a reasonable response. If the subject cannot give the same answer or a better one, then the machine wins and passes the test, we can call the

machine then as 'Intelligent'. [18]

What is artificial intelligence used for ?

Artificial intelligence has many applications. It is present in smartphone cameras. In night mode, it can adapt the colorimetry to the environment, and restore a lit façade to its original brightness so that it can be faithfully reproduced in your shot.

AI is also being introduced in the military (e.g. for decision-making in drones), in the financial sector (assessing the risks of a transaction such as granting a mortgage), in medicine (diagnosing eye diseases), in robotics, in video games (animating non-player characters), in transport (managing traffic in public transport) and in industry (setting up maintenance systems to deal with production problems).

1.3 Aims and objectives

We resumed our objectives into four main objectives which are :

- Understanding the basic idea of which chemicals are contributing towards an explosive nature.
- Use data from different types of sensors to detect.
- Improve the results of the detection by the sensors.
- Build models using different classification techniques and LSTM and evaluate the performance.

1.4 Problematic

The large size of the e-nose is usually manifested by a very large number of gas sensors or by a complex sampling system. These factors also influence the power consumption and the complexity of the learning algorithm.
Furthermore, it is very difficult to delimit the application areas of e-nose, as odours are everywhere and can be a decisive indicator for several problems. Therefore, we need a

learning and recognition system that can be adapted to our application. Indeed, the systems created before give good performances in the field of explosives, but may not be effective in other untested applications.

As we all know, there are odors that cannot be detected using the human nose because the products under study are likely to be harmful to human health, sometimes with a high mortality risk if exposed to any of these chemicals, so our E-nose detector will help us not only to avoid this risk but also to better detect explosives, and this is the important part because it's a real challenge for humans to ensure a high level of security using those kinds of algorithms.

2 Literature review

A lot of scientific researchers have proposed their works concerning odour recognition through a converted signal as an input. The proposed models identify the odours by taking the smell signal as an input. However, the widely used research of interest is Odour detection using LSTM (s).

2.1 Odour

The smell is an anatomical response stimulated from the contact between the sensory system and a mixture of light and small molecules which are always in low concentration in the air . [15]

The human sensory organ responsible for odour detection is the nose. The stimulation of sensors in the nose by vapours defines the sensation of smell. Odour is the sensation resulting from this stimulation ("smell perception"). According to Henning's odour prism in Figure 1, all complex odours can be mixtures of the primary odours located at the corners of the prism. Thus, mixtures of two odours would be defined by the edges of the prism. The triangular and square surfaces would be occupied by odours consisting of three and four components respectively. A pleasant (fragrant, spicy) or unpleasant (putrid, burnt) sensation, all are induced by the inhalation of volatile materials. Volatile organic compounds (VOCs) are the sources of odour. Hundreds of thousands of materials produce odours (i.e. VOCs). For example, the aromatic series (benzene) and compounds containing oxygen (such as alcohol and ketone) and nitrogen are among the VOCs that can be perceived as odours by humans. Fluorine, Sulphur, Ammonia and Hydrogen Sulphide are generally unpleasant odours.

Volatile organic compounds (VOCs) have molecular weights between 30 and 300 g/mol (Yuwono Lammers, 2004). The heavier molecules cannot be active at room temperature, as they generally have a low vapour pressure. Therefore, they do not occur as VOCs. The volatility of molecules is determined both by their molecular weight and by their intermolecular interaction. [16]

1.i Characteristics of odours

In combination with one or more other substances, the inherent odor of this substance can be transformed so as to be unidentifiable. [23]
Odours have characteristics that are common to all:

- Substances of similar or different chemical composition can have the same odour;
- The nature and strength of the odour may vary with product dilution;
- The strong odour always dominates the weak odours;
- The mixing of two odours of equal strength produces a combination, one or both of which may be difficult to recognise;
- The individual quickly loses awareness of the sensation of exposure to an odour of constant intensity. And they only notice it when the intensity changes;
- The cause of a complaint is often an unknown smell;
- The smell travels downwind;
- The person can smell an odour from a distance;
- Many animals have a more acute sense of smell than humans;
- he response to a scent (taste and disgust) often depends on the association of the scent with a pleasant or unpleasant experience.

1.ii Odour parameters

The human have a strong smell sens and can identify even low concentration odours. [23] In order to quantify an odour, there are several parameters that are generally human-dependent: detection threshold, intensity, hedonic tone and odour characterisation.

- The detection threshold: is the lowest concentration of the odour, which can be detected by a certain percentage of the population (usually 50%). This concentration corresponds to one unit of the odour;

- Intensity: is a force related to the concentration of the odour. It is divided into categories ranging from "not perceptible" to "extremely strong";
- Hedonic tone: this is the degree of satisfaction associated with the perception of an odour.

This measure depends on previous experience and the emotions felt by a person when perceiving an odour;

- Characterisation: this is basically what the smell smells like. The character of an odour can vary with intensity. For example, nitrogen dioxide (NO2) has an irritating and pungent smell. It is difficult to assess this parameter.

2.2 Classification of explosive detection techniques

The explosive detection techniques were classified into bulk(x-ray,CT technology, nuclear resonance absorption,ElectromagneticImaging,) or vapour and trace detection methods such as colorimetry,Mass Spectrometry(MS),AtmosphericPressure Chemical Ionization(APCI),Ion Mobility Spectrometry(IMS) are existing[1].

In bulk, explosive detection involves imaging or nuclear property of the explosives. The trace detection involves a small amount of collection of explosive vapour or substances [2]. These techniques are less reliable than trace explosion detection methods. The literature this research involves metal oxide sensor for e-sniffer implementation. A few researches use principal component analysis to identify sub-micron level changes among explosives [3].

Chemical substance sensing is a complex method, since it will try to imitate the complex biological system which consists of thousands of Neurons and receptors [4]. The volatile organic compounds (VOC) generated substances are compared with past odour that are stored in memory of humans and animals as synaptic weight [5].

2.3 Deep learning based explosion techniques

John T. Becker [6] applied a DL model to increase the accuracy of the CFAR : (constant false alarm rate) pre-screened. The authors used 3 DL architectures which are DBNs : (deep-belief-networks), SDAEs (stacked-denoising-autoencoders) and CNNs : (convolutional-neural-networks).

In conclusion, John said that in looking at the results, some important lessons can be learned. On the face of it, it appears that deep learning approaches do very well in single lane training experiments and, in contrast, shallow learning approaches appear to perform better when multiple lanes are used for training. In contrast, shallow learning algorithms seem to do best when multiple paths are employed for training. In a broad sense, this is more or less true.

Ali JameelAl-Mousawi et al[7] proposed Magnetic Explosives Detection Systems (MEDS) based on wireless sensor network and machine learning. Mariusz Chmielewski et al (2012) [8] proposed a system using Sensor Amplified Perception for Explosives Recognition (SAPER). This project runs over a smartphone platform in order to make use of the features of built-in chipset sensors.Srinivas Chakravarthy Thandu (2014) [9] used accelerometer sensor for the explosive detections.

Besaw, Lance E. et al (2016) [9] developed a project to detect buried explosive hazards with handheld GPR and deep learning. The authors applied deep learning artificial neural networks (ANN) to detect BEHs and discriminate them from harmless clutter. The authors demonstrated that deep learning ANNs can automatically extract meaningful information from complex GPR signatures compared with other state of art classification techniques.

Derek T. Anderson et al (2015) [10] developed a project by applying Computational intelligence in explosive hazard detection. The authors first extracted the features using the Convolutional Neural Network (CNN) and then fused the features using multiple kernel learning. After fusion, they classified the information using a Support vector Machine (SVM).

Francisco et al (2009) [11] applied artificial neural networks in drug and explosive detection through tomographic images with thermal neutrons. The authors used Multilayer Perceptron model, the ack-Propagation training algorithm and the Cross-validation

interruption criterion and achieved 97% accuracy.

Huayuan Ma et al(2020) [12] identified Explosive Vibration Based on EEMD Energy Entropy and Multi Classification SVM. The authors proved that Compared with BP (backpropagation) neural network algorithm, SVM (support vector machine) algorithm has higher training efficiency.

Indeed for Huayuan Ma work, he adopted the EEMD and SVM algorithms, applied on four types of explosion vibration signals passing by training and testing the new predictions, as a conclusion we found these relevant ideas:

The entropy is seen as a significant feature which contributes really to the wich results for the recognition of vibration signals.

The EEMD method has more advantages in training phase and accuracy when predicting if we compare it to classic Back Propagation neural network algorithms.

Also made a conclusion saying that his method doesn't have some obvious advantages compared with the permutation or approximate entropy method.

The work made by Huayuan can help in monitoring, specially preventive measures to ensure helping managers to detect those types of accident before it happens.

And finally he mentioned the data augmentation factor where he increased the number of samples collected (Same distribution of the data but more data) to achieve a better accuracy because it was insufficient at first.

Summary of the existing work is tabulated in Table 1 and included in appendices.

Algorithm	Objectives	Accuracy	Author
Convolutional Neural Networks Stacked Denoising Autoencoders	Improve the performance of a Constant False-Alarm-Rate Minimize the average reconstruction error	90% 97%	John T.Becker
Convolutional Neural Networks Support vector Machine	Magnetic Explosives Detection System (MEDS) based on wireless sensor network and machine learning. Use of magnetic sensors to identify magnetic IEDs. Use of the WSN approach to link the magnetic captors to the network server. Test, validate and analyse the results of the new detection system.		Ali Jameel Al-Mousawi and al
Artificial neural network	Detect buried explosive hazards with handheld GPR and deep learning Detect BEHs and discriminate them from harmless clutter		Besaw,Lance E. and al (2016)
Convolutional Neural Networks Support vector Machine			Derek T.Anderson and al (2015)
Multilayer Perceptron model	Experimental setup for neutron tomography Ensure a good digital processing of the image tomography Explosive detection through tomographic images	97%	Francisco and al (2009)

(continued)

Algorithm	Objectives	Accuracy	Author
SVM EEMD(Energy entropy)	Explosive Vibration Datection Based on EEMD Energy Entropy and Multiclassification SVM Ensuring a high accuracy with a small number of rows via data augmentaion Applying the final model to small technological engine	76.8% 96%	Huayuan Ma and al (2020)
VGG-16 VGG-19 Resnet18 Resnet34 Resnet50	Classifying mixed gas analogous-image matrix [19] Maping the original two-dimensional time series matrix into a three-dimensional analogous-image matrix	63.33%-70% 60%-70% 90%-86.67% 93.33%-90% 96.67%	C.C.Y. ,L.H. (experimental work); K.T.X. (sensor characteristics analysis) and X.Z. completed the data analysis.
LeNet-5 SVM PNN	Implementation of LeNet5 Gas Detection CNN model for Electronic Noses[20] Turning or adujsting parameters in order to best idenify categories while training,perform a validation processus by testing on real word cases.	98.67% 85.70% 93.33%	Guangfen Wei Gang Li Jie Zhao Aixiang He
SVM	Gas Classification Using Deep Convolutional Neural Networks	79.9%	Pai Peng

(continued)

Algorithm	Objectives	Accuracy	Author
MLP	Pre-processing, extracting features, implementation.[21]	82.3%	Xiaojin Zhao
GasNet	Prooving that GasNet algo perform better than SVM or MLP	95.2%	Xiaofang Pan Wenbin Ye
CNN-LSTM	Implementation of CLSTMNN to Sensors Network data in order to locate the gas in the environement [22]	95.0%	Christian Bilgera Akifumi Yamamoto
LSTM		85.0%	
CNN-DNN	Find the best model that gives the highest number of values prediction wich are true postivies but with limitation regarding the other values like TN, FN or FP.	90.0%	Maki Sawano Haruka Matsukura Hiroshi Ishida
DNN		91.1%	

Autor	**Approach**
John T. Becker [6]	deep learning approach to improve the performance of a constant false-alarm-rate (CFAR)
Ali JameelAl-Mousawi and al[7]	Magnetic Explosives Detection System (MEDS) based on wireless sensor network and machine learning
Mariusz Chmielewski and al(2012)[8]	using Sensor Amplified Perception for Explosives Recognition (SAPER).
Besaw, Lance E. and al (2016) [9]	applied deep learning artificial neural networks (ANN) to detect BEHs and discriminate them from harmless clutter
Derek T. Anderson and al (2015) [10]	extracted the features using Convolutional Neural Network (CNN) and then fused the features using multiple kernel learning. After fusion they classified the information using Support vector Machine (SVM).
Francisco and al (2009) [11]	applied artificial neural networks in drug and explosive detection through tomographic images
Huayuan Ma and al(2020) [12]	identified Explosive Vibration Based on EEMD Energy Entropy and Multiclassification SVM

For the LSTM based approach, two major elements make us think that LSTM may be the best under our conditions, the first thing that comes to mind is that the DL models should be easily implemented in an embedded system that would be a centerpiece in an accurate, mobile, fast and cheap electronic nose for explosive detection. Secondly, our time is limited and the data looks like a time series as the sensors take data for 6, 5, 4 or 3 minutes etc.... and this is how RNN(s) like LSTM work. But when more time-related data is available, CNN works better. At this stage we can't really conclude which algorithm will be better, so we have to go deep into the practice of our case to compare and analyse the results at the end.

2.4 Summary of progress to date

In this project we have datasets from different chemical sensors. At first, we try to use simple classification techniques like logical regression, binary classification, and decision tree and then gradually jump to the deep learning part and get the result using LSTM networks. Then we can compare the results in the different models for different types of sensor data and conclude the most efficient model to be used.

A similar project has been implemented in [6].In this project explosive markers are detected using zeolite modified gas sensors and the data was then tested for different conditions like high humidity and high temperature; the detection is using static data so here it's a classic classification. Here when the SVM was used it was able to classify data with higher than 85% accuracy when tested using various factors, like humidity and variable gas exposure times.

The proposed project work improves the existing work [6] by adding LSTM networks to the learning model and also uses all possible classification techniques to analyze different results.

After doing a literature review the following steps have been arrived to detect the explosives using LSTM techniques that is listed on next section.

2.5 Dataset: mixed explosives dataset

The data set used in this project was downloaded from the IEEE Data Port by providing login credentials in this **link**. This data set was created by Ana Guaman Seok-Bum Ko et al [1] and is available to the public for research purposes. The research work using this data set has to cite the data set in their reference. The proposed work cited the data set as reference [14].

3 Project plan

In this research problem I am trying to classify whether something is explosive or not using artificial nose hardware which has sensors capable or reading values of substances and accordingly the value read by generated by the sensor will be the input to Machine Learning and Deep Learning Model which will identify whether it can be classified as an explosive material or not. This research problem has been treated many times by applying few famous Supervised learning Classification Algorithms such as Decision Tree, Random Forest, Logistic Regression, KNN, RNN and LSTM.

3.1 Tasks involved in the proposed Work

- **Dataset** : are available at the ieee-dataport site or Kaggle dataset were taken from there or use in the proposed project.
- **Data Understanding** : Get the information of data attributes and their key characteristics. Summarize the data and identify the outliers, missing values.
- **Data Pre-Processing** : This stage mainly includes 4 steps: cleaning, blending, decomposition and transformation. After doing the pre-processing steps, the proposed research work can effectively handle the missing and outliers, scaling and features reduction.
- **Data Visualization** : In order to visualize the data and the proposed work uses some libraries like Seaborn, Matplotlib etc.
- **Splitting into training and testing** : In order to train the proposed model, the data set is split into train and test data and then fit into according to Machine learning algorithms.
- **Modelling** : The proposed approach uses basic and traditional classification modelling like logistic regression, binary classifiers and decision tree first and then comparing them with RNN and LSTM models and increasing the efficiency of the model.

- **Testing** : the model with data from different types of sensors and making the required amendments.
- **Questions Answered** : After the model is ready finding out the answers to the required question which were discussed above.
- **Model Evaluation** : Now in this last step we are trying to evaluate the performance of our model for which we have different kind of performance metrics like accuracy score, confusion matrix and classification report.
- Making sure that the evaluation has been able to completely cope up with all the questions which need to be answered.
- Evaluate the performance.

In order to provide better clarity on the proposed work, it is divided into three parts namely :

Data pre-processing contains the following steps

- Data Understanding
- Data Pre-Processing
- Data Visualization

Training the module contains the following steps

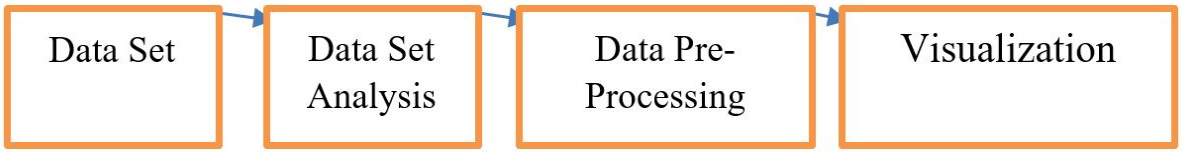
- Splitting into training and testing
- Modelling
- Testing
- Questions answered

Evaluate the performance contains the following steps

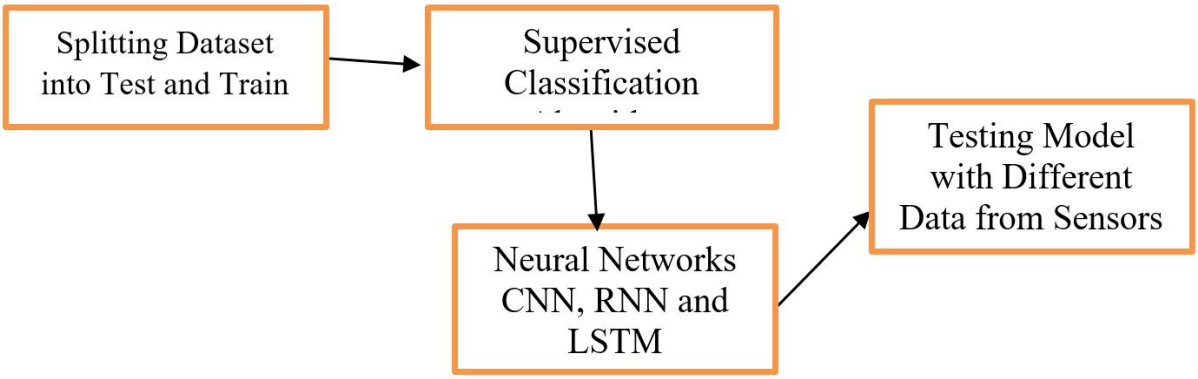
- Model Evaluation
- Making sure that the evaluation has been able to completely cope up with all the questions which need to be answered
- Evaluate the performance

3.2 Prototype of the proposed work

Phase 1 : Data Pre-Processing



Phase 2 : Training the model



Phase 3 : Evaluate the performance

4 Methodology

Concerning this section we'll have three subsections, first part will be about the Data Collection process, then we'll follow up on both technologies and tools used to finaly conclude with the method adopted.

4.1 Data collection

In this section we have two files which are our input data, the files were downloaded from kagle website as on of their projects, the first files is "sensors_data_augmented.h5" and it contains the response of each sensor from the six sensors to the explosive over a certain period, the second file is the "binary_labels_augmented.txt" , this one contains only the binary label classification which is either Explosives or No_explosives and it refers to our output. The both file serve to apply prediction models in the field of explosive.

Figure 4.1: Input data (Sensors Data)

```
[[[ 4.   0.   0.  ... 14.  15.  14. ]
  [ 4.   1.   0.  ... 14.  14.  15. ]
  [ 5.   0.   0.  ...  4.   4.   4. ]
  [ 0.   1.  15.  ... 18.  18.  17. ]
  [ 5.   0.   0.  ...  6.   6.   5. ]
  [ 4.   0.   0.  ... 13.  14.  13. ]]

 [[ 0.   0.   0.  ... 14.2 14.2 14. ]
  [ 0.   0.   0.  ... 14.8 14.6 14.6]
  [ 0.   0.   0.  ...  4.4  4.   4. ]
  [ 0.   0.   0.  ... 19.2 18.8 18.2]
  [ 0.   0.   0.  ...  6.   6.   5.6]
  [ 0.   0.   0.  ... 13.4 13.6 13.2]]

 [[ 0.   0.   0.  ... 14.8 14.7 14.5]
  [ 0.   0.   0.  ... 15.3 15.1 15. ]
  [ 0.   0.   0.  ...  4.6  4.5  4.4]
  [ 0.   0.   0.  ... 20.3 19.9 19.4]
  [ 0.   0.   0.  ...  6.2  6.2  6. ]
  [ 0.   0.   0.  ... 13.6 13.7 13.6]]

 [[ 0.   0.   1.  ... 11.   6.   5. ]
  [ 0.   1.   2.  ... 15.   7.   6. ]
  [ 2.   2.   1.  ... 14.   5.   3. ]
  [ 0.   3.  12.  ...  9.   2.   1. ]
  [ 2.   2.   0.  ... 12.   5.   3. ]
  [ 0.   0.   1.  ... 14.   5.   5. ]]]
```

The shape of our data is : (420, 6, 300), this means that the data contains 420 samples and each sample has a shape of (6, 300) wich also means six lines and 300 columns.
The output looks like this :

Figure 4.2: Binary output data

```
 1 No_explosives
 2 No_explosives
 3 No_explosives
 4 Explosives
 5 Explosives
 6 Explosives
 7 No_explosives
 8 No_explosives
 9 No_explosives
10 No_explosives
11 No_explosives
12 No_explosives
13 Explosives
14 Explosives
15 Explosives
```

4.2 Technologies tools

Tools :
The explosive detection work was carried out using Python 3.7.10. In this framework, Python 3 is employed like a tool for analysis processing and program implementation (Kim, et al., 2017). The reason for this choice of Python 3.7.10 is described below:

- Python is not payable because it's an open-source langage.
- Python 3 contains several libraries for Data Pre-processing, Data mining, Machine Learning or Deep Learning.
- The python communities take good care of their users and anyone could find documentation on any existing function in python.

Environment and IDE

Figure 4.3: Anaconda Logo

The work will be implemented using Anaconda which is an open source distribution that contains programming languages such as Python and R that are used in different fields such as: data collection and processing, machine learning, statistical analysis.
With ANACONDA we can install modules (packages) easily since there is a dependency between the modules that must be taken into consideration. Anaconda manages this point, so the user will just install them by typing commands that often start with "conda". In addition, Anaconda allows us to create working environments. This helps to organize the work and make it clearer and more structured, that is, when you are working on several projects at the same time and each project uses different modules than the other projects, you can create an environment for each project.

The Anaconda browser has several applications or editors like :

- Jupyter Notebook
- Spyder
- RStudio

As said, it's an open source distribution, hence no ethical, legal, professional and social issues with this proposed work.

Librairies :

Numpy : is a python library dedicated to arrays (very used in data science) it is open-source and was created in 2005, written partially in Python (the parts requiring fast calculations are written in C or C++).

Pandas : is an open-source library based on NumPy providing easy-to-handle data structures and data analysis tools. Pandas is more suitable because it provides many easy-to-use tools for manipulating many kinds of data in Python.

H5py : is an open-source library used to support the HDF5 format in Python.

Scickit-learn : is a python module incorporating traditional machine learning algorithms. It intends to provide smart and efficient solutions for learning issues, available to any user and adaptable to various contexts.

Matplotlib : is a library which enables you to do 2D data visualization in an incredibly practical way. It is not necessary, of course, but as soon as we deal with Machine Learning projects it can be very useful.

Seaborn : is a library that harnesses the power of matplotlib to create beautiful visualisations in a few lines of code. The main difference is the styles and colour palettes offered by Seaborn, which are designed to be more aesthetically pleasing and modern.

Keras : is an open source software library designed to simplify the creation of machine learning models. Keras is coded in Python and can be deployed on other artificial intelligence technologies such as TensorFlow, Microsoft Cognitive Toolkit (CNTK) and Theano. Keras is known for its ease of use, modularity and scalability. It is perfect if you need a machine learning tool that allows for simple and fast prototyping, supports convolutional and recurrent networks, and runs optimally on both CPUs and GPUs.

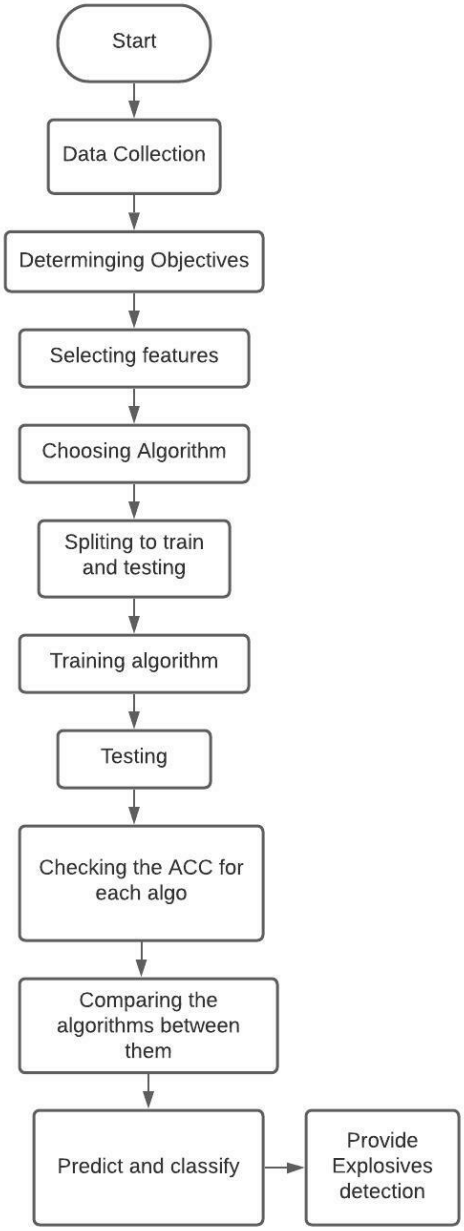
Tensorflow : is an open source machine learning framework that is easy to use

and deploy. It is one of the best maintained and most widely used machine learning frameworks. Created by Google to support its research and production goals, TensorFlow is now widely used by several companies and allows you to develop neural networks.

4.3 Methodology

The approach used is shown in the process bellow :

Figure 4.4: Methods used



5 Experimental setup and analysis procedure

The experimental procedure for the recognition of explosives and the analysis approach will be discussed in this section. The analysis results obtained using machine learning and deep learning approaches will be elaborated in the following section.

5.1 Data analysis

Firstly, the data is imported using pandas ans h5py libraries. The sensors data is stored in a .h5 format for each sample there is six metal oxid chemical sensors the first two sensors are T-GS2610 and two of T-GS822 sensor, and finally the last two chemical sensors of type T-GS825 and T-GS826, each sensors has 300 observations. Please do not forget that we already verified the baance of the data.

5.2 Explosives analysis

In order to better achieve insights from this part we studied a random sample from our input data, the simple as an array of shape (6,300), so six sensors and 300 observations over time, the visualization of the six sensors reactions or evolution over time is necessary. The sensors data is neither sclaed neither normalized, so we need to apply feature scaling and mean normalization before processing our data, indeed when we do this, our observations will have same scale and mean around zero.

5.3 Adopted algorithms

We will implement Three Machine Learning classification algorithms which are : **Decision tree, Random forest , KNN**, and finally two Deep Learning models; **CNN ,** and **LSTM**.

3.i Machine learning models

Decision Tree :

It is one of the most popular supervised learning methods for data classification problems.

In practice, the decision tree concept is to establish a hierarchy of checks to predict something. Checks are nothing else than tests. There are two main types of decision trees:

- Regression trees predict an actual quantity, a numerical value (example, Salary prediction);
- Classification trees deal with classification problems to predict for example, if someone will purchase or not based on the age and salary estimation.

The possible choices are located at the ends of the branches and are obtained by taking into account the choices made at each stage. A decision tree operates by a complete application of an iterative process composed of many logical rules.

Random Forest :

The Random Forest is a classification or regression model that aims to reduce the variance of predictions for a single decision tree, which means increasing its efficiency. It achieves this by merging a lot of decision trees in a bagging method.

KNN :

The kNN algorithm considers that similar items exist in near distance between them. In other words, similar objects are close to each other.
The kNN model is based on the idea of similarity which refers to distance or proximity. The distance is the same distance that we learned when we were childrens.
Other methods of measuring distance exist. The choice of which method to use is related to the issue at hand. Nevertheless, the Euclidean distance is generally considered as a good distance formula to use !

3.ii Deep learning models

CNN :

Convolutional networks are widely used in image analysis. The two main characteristics of convolutional networks are that they use filters (kernels) and implement pooling. The filters analyse the images area by area. Each filter is specialised to recognise patterns. For example, one filter may specialise in edge detection, while another will recognise certain shapes. Convolution has the effect of increasing the depth of the image matrix, as

each filter will add a layer to it. An image that has a depth of 3 layers (the number 3 corresponding to the 3 RGB channels) could thus result in a matrix with a depth of 5, if the convolutional network consists of 5 filters. Pooling reduces the size of an image by keeping only the most important pixels. This has the effect of distorting the image by losing the precise positioning of the pixels. This effect is actually beneficial, as it limits the risk of overlearning. For example, a face detection system will benefit from learning that a face consists of two eyes, a nose and a mouth, but it is preferable that it does not memorise the pixel spacing between these facial features, as their position may vary from person to person. There are other techniques than pooling, notably those brought about by the capsule networks, a new technique which will not be discussed in this tutorial on convolutional networks.

LSTM :

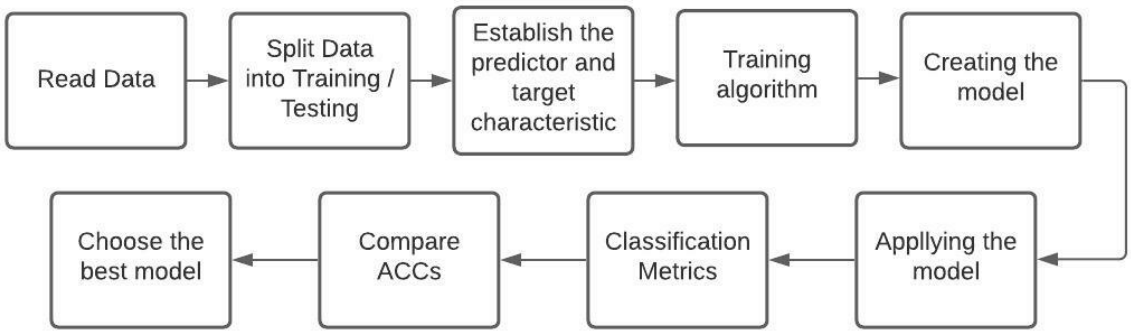
Long Short Term Memory (LSTM) is a recurrent neural network (RNN) architecture used in the field of deep learning. Unlike forward propagation neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also complete sequences of data (such as speech or video).

A basic LSTM unit consists of a cell, an input gate, an output gate and a forgetting gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. LSTM networks are well suited for classification, processing and forecasting based on time series data, as there may be time lags of unknown duration between important events in a time series. LSTMs have been developed to cope with the gradient problems that can be encountered when training traditional RNNs.

5.4 Algorithms application process

Our models will be applied to the data using a fixed strategy step by step which make it a process to respect. This process looks like this :

Figure 5.1: Developing Process



5.5 Models efficiency evaluation

Evaluation metrics are used to measure the quality of the statistical or ML model. Evaluating machine learning models or algorithms is essential for any project. There are many different types of evaluation metrics available to test a model. These include classification accuracy, logarithmic loss; confusion matrix, and others. Classification accuracy is the ratio of the number of correct predictions to the total number of input samples, which is usually what we refer to when we use the term accuracy. Logarithmic loss, also called log loss, works by penalizing the false classifications. A confusion matrix gives us a matrix as output and describes the complete performance of the model. There are other evaluation metrics that can be used which are important to us for our case : Accuracy, Precision, Recall and F1-Score. Evaluation metrics involves using a combination of these individual evaluation metrics to test a model or algorithm.

5.i Accuracy

If we ask ourselves how much correct indentification we have, the accuracy can answer to this question. So It give us the % of correct predictions. It's also the result of devision between the sum of Trues and the total number of observations.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$$

5.ii Precision

The precision corresponds to the number of documents correctly attributed to class i compared to the total number of documents predicted as belonging to class i (total predicted positive).

$$Precision = \frac{TruePositive}{TruePositive+FalsePositive}$$

5.iii Recall

It's the ratio between the True Positives(TP) and all the data points(TP and FN), which means of all positive data point how many were actually positive which is predicted by our model.

$$Recall = \frac{TruePositive}{TruePositive+FalseNegative}$$

5.iv F1-score

F1-score : It is nothing more but the weighted avg of Recall and Precision. Therefore, this score takes both False Positives (FP) and False Negatives(FN) into account.

$$F1 = 2 * \frac{Precision*Recall}{Precision+Recall}$$

6 Result and interpretation

In this chapter we'll debate about results obtained and compare the models in order to conclude on the best one.

6.1 Normalization and feature scalling

After reading the data in the working environment (Jupyter Notebook). The data contains all observations after the data augmentation process. From the analysis of the evolution graph and the data of each observation it could be concluded that the data needs feature scaling and normalization in order to have reduced centered data.

In order to establish the feature scaling, we call the 'Preprocessing' sub-library of the parent library "sklearn" and precisely the Standardscaler object, the instance created from this object allows us to execute the task thanks to the "fit_transform" function for the train and just "fit" in the testing part. For the normalization, the goal is to a mean which is around zero. This is done using the instance "Normalize" of the same sub-library Pre-processing.

A random observation after feature scaling look like this :

Figure 6.1: Scaled Data

```
[[-0.4472136  -1.29986737  0.         ... -0.04319342  0.08567059
  -0.02119996]
 [ 2.23606798  0.92847669  1.22474487 ... -0.30235395 -0.29984706
  -0.1483997 ]
 [-0.4472136   0.92847669 -1.22474487 ... -0.69109474 -0.6853647
  -0.78439843]
 [-0.4472136  -1.29986737 -1.22474487 ...  2.15967106  2.14176468
   2.14119572]
 [-0.4472136   0.92847669  0.         ... -0.820675   -0.81387058
  -0.78439843]
 [-0.4472136  -0.18569534  1.22474487 ... -0.30235395 -0.42835294
  -0.40279919]]
```

And after normalization it look like figure bellow :

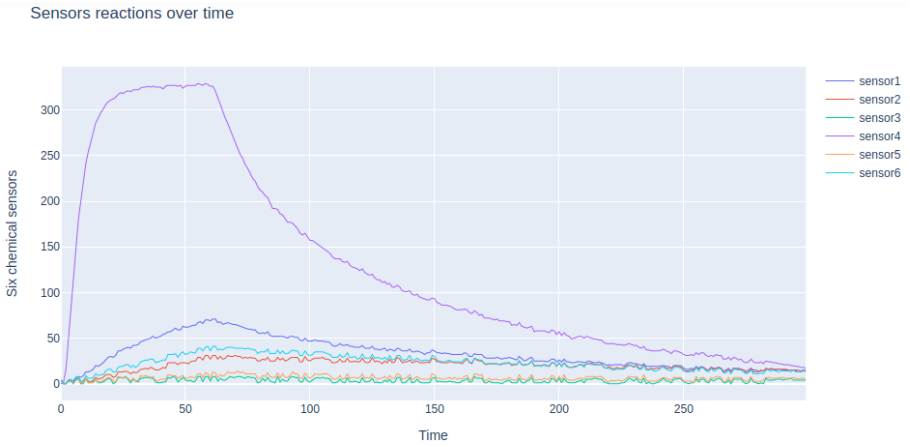
Figure 6.2: Scaled/Normalized Data

```
[[-0.61924801 -1.88266744 -0.2644138  ...  0.52864785  0.88424159
   0.5678098 ]
 [ 2.21150141  0.97997187  1.1976524  ... -0.14670443 -0.14158791
   0.25140939]
 [-0.43815309  0.80415755 -1.48188654 ... -0.8869929  -0.82708317
  -1.05453697]
 [-0.15419002 -0.37643325 -0.56796311 ...  1.8881202   1.74262709
   1.78547663]
 [-0.42878489  0.78248483 -0.2644138  ... -1.13358325 -1.05542188
  -1.01268637]
 [-0.5711254  -0.30751355  1.38102484 ... -0.24948747 -0.60277573
  -0.53747246]]
```

6.2 Data analysis results

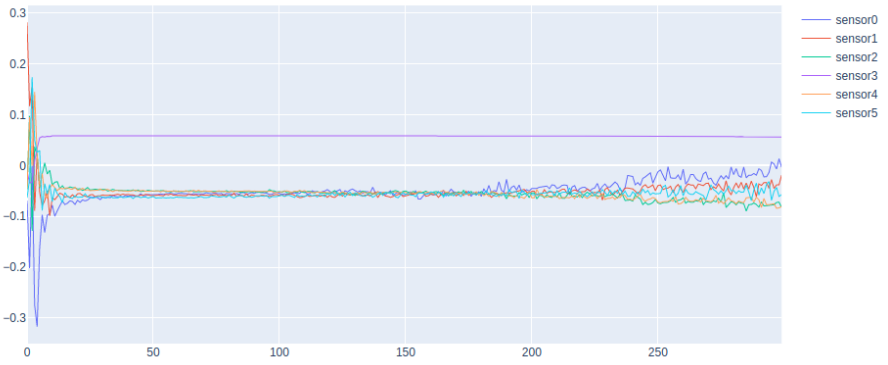
Each sensor of the six chemical sensors data contains 300 observations over time. The results show us at first the sensors reaction over time, time is splited into 300 units which make the 300 observastions mentionned above. This graph looks like this :

Figure 6.3: Sensors reaction over time



Then after scalling and normalizing our data, the mean become approximately zero and all observation will have same sclae as show in the figure bellow :

Figure 6.4: Scaled/normalized sensors reaction over time



6.3 Implementation of Machine Learning Models

Fabien Benureau (2015): "Learning is a modification of behaviour on the basis of experience".

This notion encompasses any method allowing to build a model of reality from data, either by improving a partial or less general model, or by creating the model completely. There are two main trends in learning, the one coming from artificial intelligence and qualified as symbolic, and the one coming from statistics and qualified as numerical.

The final data after scaling and normalizing is structured in this dataframe :

Figure 6.5: Final Data

Date	1	2	3	4	5	6	7	8	9	10	...	291	292	293	29
0	-0.069422	-0.201781	0.000000	-0.274414	-0.317085	-0.161841	-0.096973	-0.131783	-0.097872	-0.094698	...	-0.006359	-0.009609	-0.022213	-0.02204
0	0.281893	0.117050	0.154399	-0.089142	0.023410	-0.052573	-0.078753	-0.069250	-0.049677	-0.098879	...	-0.036147	-0.039017	-0.033502	-0.03325
0	-0.046947	0.097468	-0.128569	0.037115	0.019494	-0.043778	-0.016395	0.005242	-0.016547	-0.009149	...	-0.081700	-0.071477	-0.079400	-0.07881
0	-0.011705	-0.034022	-0.032056	0.009254	0.034022	0.054575	0.057227	0.056203	0.057757	0.057025	...	0.056822	0.056703	0.057250	0.05682
0	-0.045784	0.095054	0.000000	0.144782	0.019011	-0.042694	-0.015989	-0.056237	-0.064547	-0.044610	...	-0.067096	-0.082380	-0.064877	-0.07685
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
314	0.000000	0.000000	0.000000	0.000000	-0.015624	-0.023488	-0.030239	-0.033897	-0.037154	-0.037692	...	-0.081201	-0.070346	-0.084683	-0.05678
314	0.000000	0.000000	0.000000	0.000000	-0.050033	-0.044737	-0.041228	-0.039517	-0.038652	-0.036035	...	-0.018666	-0.009241	-0.019466	-0.02088
314	0.000000	0.000000	0.000000	0.000000	-0.022491	-0.021325	-0.020671	-0.019764	-0.018911	-0.018222	...	0.059181	0.060938	0.061719	0.05827
314	0.000000	0.000000	0.000000	0.000000	-0.039602	-0.041541	-0.041149	-0.043090	-0.043857	-0.046548	...	-0.109954	-0.101088	-0.090097	-0.12303
314	0.000000	0.000000	0.000000	0.000000	-0.053502	-0.055262	-0.056390	-0.056541	-0.057330	-0.059150	...	0.009980	-0.009881	-0.020816	-0.02233

Please note thet the data was already splited into trainning and testing with a 80/20 % of split between Training/Testing. All the models we will use are aplied to the training set and in order to compare between them we will use the test set.

3.i Decision tree

Decision tree is a predictor model used for both classfication and regression situations. It's simple to implement.

Confusion Matrix

	precision	recall	f1-score	support
0	0.55	0.71	0.62	59
1	0.39	0.24	0.30	46
accuracy			0.50	105
macro avg	0.47	0.48	0.46	105
weighted avg	0.48	0.50	0.48	105

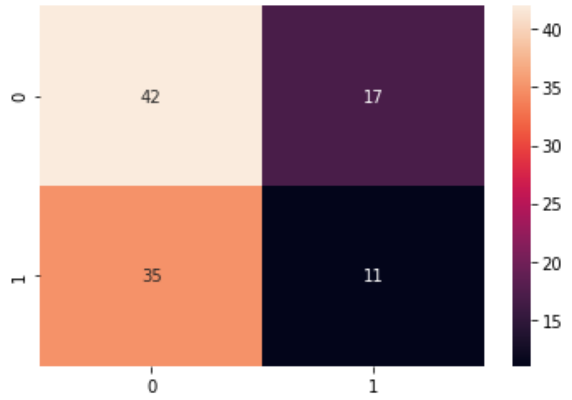


Figure 6.6: Metrics report and Confusion Matrix for Random Forest model

Evaluation Metrics

The evaluation of the metrics derived from this model are written bellow

Accuracy : 50% | **Precision** : 48% | **Recall** : 50% | **F1-score** : 48%

The accuracy obtained here isn't good at all and also for the other parameters. So this model is and will not be the best one for our case.

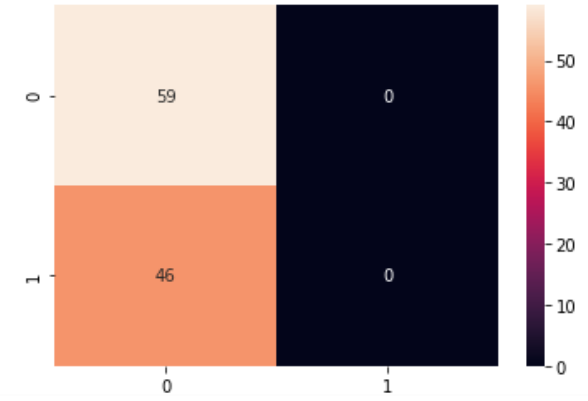
3.ii Random Forest

Random Forest is a predictor model used for both classfication and regression situations. It's simple to implement and it refers to a number of trees why it's called a forest.

Confusion Matrix

Figure 6.7: Metrics report and Confusion Matrix for Decision Tree model

	precision	recall	f1-score	support
0	0.56	1.00	0.72	59
1	0.00	0.00	0.00	46
accuracy			0.56	105
macro avg	0.28	0.50	0.36	105
weighted avg	0.32	0.56	0.40	105



Evaluation Metrics

The evaluation of the metrics consists of several significant variables which allow us to measure the effectiveness of the agorithm like Accuracy, Precision, Recall and F1-Score.

Accuracy : 56% | **Precision** : 32% | **Recall** : 56% | **F1-score** : 40%

The accuracy obtained here isn't good at all and also for the other parameters. So this model is and will not be the best one for our case.

3.iii K nearest neighbours

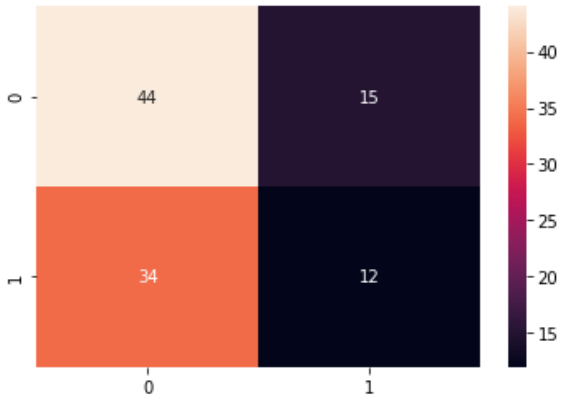
To summarise, Knn is a predictor model used for classfication situations. It's simple to implement and it refers to the k groups chosen. Then each group is defined by a centroids and surrounded with a number of points (Those points are the nearst ponits to the centroid)

Confusion Matrix

Evaluation Metrics

Figure 6.8: Metrics report and Confusion Matrix for knn model

	precision	recall	f1-score	support
0	0.56	0.75	0.64	59
1	0.44	0.26	0.33	46
accuracy			0.53	105
macro avg	0.50	0.50	0.49	105
weighted avg	0.51	0.53	0.50	105



Accuracy : 53% | **Precision** : 51% | **Recall** : 53% | **F1-score** : 50%

The accuracy obtained here isn't good at all too and also for the other parameters. So this model is and will not be the best one for our case. For the Machine Learning it's not an accepted score in accuracy.

6.4 Implementation of Deep Learning Models

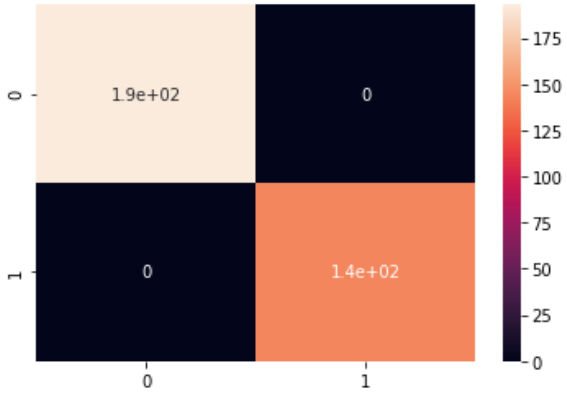
As you already know after implementing the classical Machine Learning modes, we didn't get an accceptable accuracy which makes the ML algorithms not well adapted to our problem. We will implement Deep Learning in order to see if they would perform better in indentifying explosives. The higher accuracy we got when dealing with ML models is 56% which is not acceptable. In this section we will take on the Convolutional Neural Network and Long Short Term Memory models to get a higher prediction.

4.i Convolutional Neural Network

Confusion Matrix

Figure 6.9: Metrics report and Confusion Matrix for CNN model

	precision	recall	f1-score	support
0	1.00	1.00	1.00	193
1	1.00	1.00	1.00	144
accuracy			1.00	337
macro avg	1.00	1.00	1.00	337
weighted avg	1.00	1.00	1.00	337



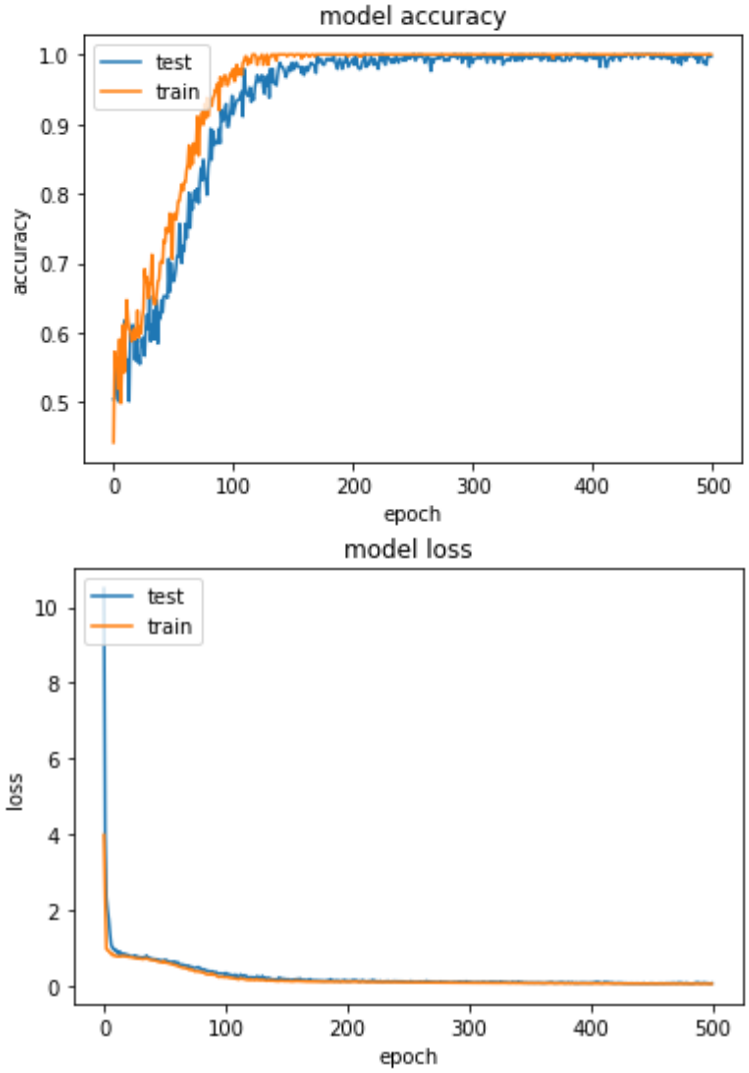
Evaluation Metrics

Accuracy : 100% | **Precision** : 100% | **Recall** : 100% | **F1-score** : 100%

The accuracy obtained here is perfect and also for the other parameters. So this model is the best one for our case after what we saw the classical Machine Learning models. In deep Learning models, it's a great accuracy.

Loss and Accuracy plots

Figure 6.10: Accuracy and loss plots for CNN model

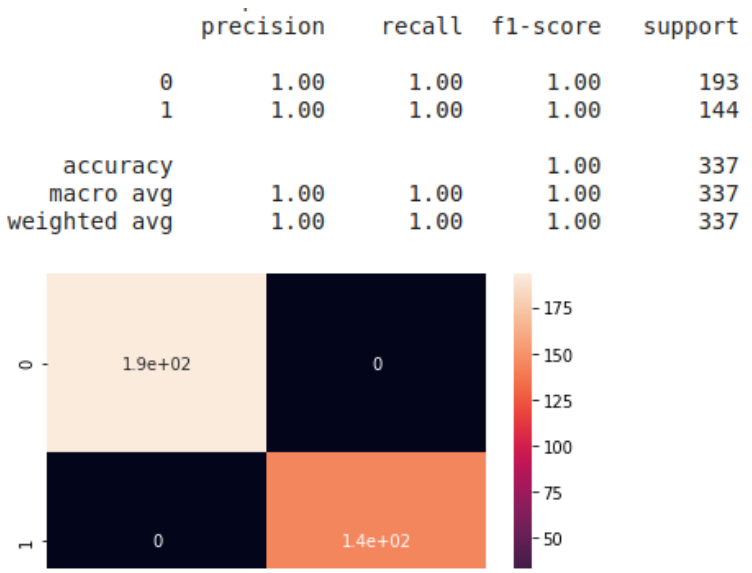


The accuracy is increasing and the loss is decreasing over epochs to the end. We had a perfect accuracy which is 100%. Maybe the memory algorithms can also work for this case, that's why we will try the LSTM model in the next part.

4.ii Long Short Term Memory

Confusion Matrix

Figure 6.11: Metrics report and Confusion Matrix for LSTM model



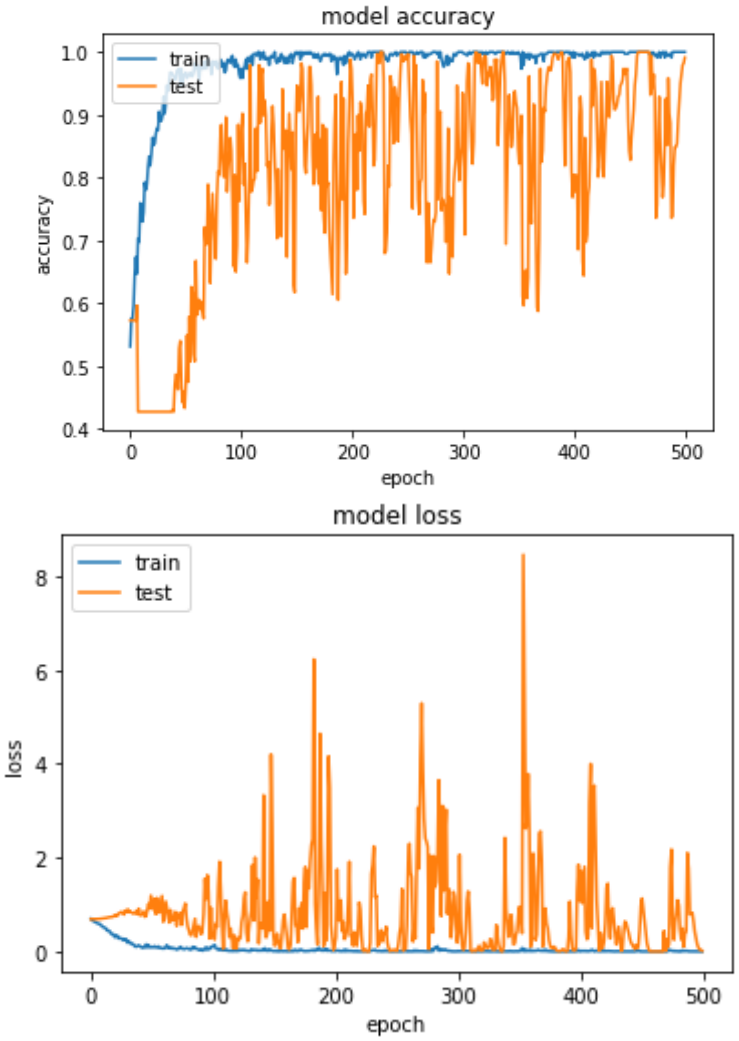
Evaluation Metrics

Accuracy : 100% | **Precision** : 100% | **Recall** : 100% | **F1-score** : 100%

This is the best we can do, we can call it as a perfect prediction, because here we obtained an accuracy of 100%, also the recall and precision is at their best, so we ca conclude that using lstm(s) netwroks we can achieve the best results possible,either in terms of the true positives and the true negatives. Finally, we can have a better time computation using less observations which means less data.

Loss and Accuracy plots

Figure 6.12: Accuracy and loss plots for LSTM model



The accuracy is increasing and the loss is decreasing over epochs to the end. We had a perfect accuracy which is 100%. Maybe the memory algorithms can also work for this case, that's why we will try the LSTM model in the next part.

6.5 Model comparaison

The models comparison at 300(s) is shown bellow :

Model	Decision Tree	Random Forest	KNN	CNN	LSTM
Accuracy	53%	56%	53%	100%	100%

Table 6.1: Method used

6.6 Models comparison at various timesteps

This is the table showing the accurracy at diverous time intervals :

Model	Decision Tree	Random Forest	KNN	CNN	LSTM
N(s)	**Accuracy**				
300	53%	56%	53%	100%	100%
240	55%	55%	50%	100%	100%
180	48%	57%	55%	100%	98%
120	44%	50%	57%	100%	96%
60	52%	48%	58%	99%	92%

Table 6.2: Accuracy over different time intervals

What models (Machine vs Deep Learning) works efficiently on the less data points?

As we can deduce from the table bellow the Deep Learning models works efficiently on less data points despite N(s).

How does the low data influence the behaviour of the Machine Learning and Deep Learning models?

If we discuss the time compute, well there is a progress in it because the speed is higher while computing due to less data points. For Machine Learning or Deep Learing models, often the less data points give us less informations and ML models need badly the maximum of informations existing in most of cases, but in some specific tasks it's not the case as in our case the best accuracy obtained for "decision tree" as an example is at 240s and not 300s, also for "Knn" where a prediction at 60s is the best one so low data may affect negatively our model as well as positively, in the other hand regarding DL models if we don't reduce too much data it doesn't affect most of the cases and it can serve us to have a model which doesn't require a lot of data, so it can be deployed on small devices.

7 Conclusion

Explosives detection places particular emphasis on gas or image recognition and the type of explosive that may be encountered. The methodology can also be applied in different fields such as the recognition of fragrances and the detection of chemicals that are harmful to health by detecting their reactions against well-defined gases.

In this research paper, the detection of explosives was done to recognise explosives that a human being alone cannot identify. Firstly we should know that we can do this either using image detection or gas sensors detection. The goals and objectives were defined to carry out the research. The research concept was derived from the past researches where the implementation of deep learning was observed in the explosives database. Previous researchers have implemented several models of Machine Learning Deep Learning to detect exolosives.

By implementing machine learning approaches in explosives databases, meaningful accuracy has been reached. In these cases, the classification models were applied, the accuracy was computational or compared to others.

On the other side, A lot of other researchers have implemented Deep Learning models such as Convolutional Neural Network or Long Short Term Memory. In these cases, the accuracy was most of the cases higher than 92%. Indeed, there is a necessity to improve the accuracy by making the detection more effective with less data. Just like all the dataset that have been used by previous researchers, the Kaggle database is choosen for the Machine Learning and Deep Learning applications. This dataset was chosen so that the deep learning models will be applied to achieve a satisfying results in the explosives detection domain.

This dataset was then used to identify explosives using classical machine learning algorithms such as Decision Tree, Random Forest or KNN and deep learning models such as Convolutional Neural Network or Long Short Term Memory. The findings of these algorithms were compared with those of the machine learning models to detect the most efficient way to correctly identify those explosives. Finally, by comparing them, it was shown that the lstm network managed to have 100% accuracy which is better than the CNN and incredibly better than the classical ML models.

Bibliography

[1] J. Torres-Tello, A. V. Guaman and S. -B. Ko, "Improving the Detection of Explosives in a MOX Chemical Sensors ArrayWithLSTM Networks," in IEEE Sensors Journal, vol. 20, no. 23, pp. 14302-14309, 1 Dec.1, 2020, doi: 10.1109/JSEN.2020.3007431.

[2] William J. Peveler, Russell Binions, Stephen M.V. Hailes and Ivan P. Parkin, "Improving the Detection of Explosives in a MOX Chemical Sensors Array with LSTM Networks", IEEE SENSORS JOURNAL. DOI 10.1109/JSEN.2020.3007431

[3] R.G. Ewing , D.A. Atkinson , G.A. Eiceman , G.J. Ewing, "Detection of Explosive Markers Using Zeolite Modified Gas Sensors", DOI: 10.1039/b000000x

[4] Lisa Thiesan, David Hannum, Dale W. Murray, John E. Parmeter, "Survey of Commercially Available Explosives Detection Technologies and Equipment 2004", Document No.: 208861, (2005) Award Number: 96-MU-MU-K011.

[5] SumithManiath, Aravind Ashok, PrabaharanPoornachandran, Sujadevi VG, Prem Sankar A U, Srinath Jan, "A critical review of ion mobility spectrometry for the detection of explosives and explosive related compounds", Talanta 54 (2001) 515–529.

[6] John T. Becker, " deep learning methods for multiband explosive hazard detection using l-band and x tion using l-band and x-band forw -band forwardlooking ground-penetrating radar ", dissertations, master's theses and master's reports - open, 2014. https://discovery.ucl.ac.uk/id/eprint/1393189/1/1393189a.pdf

[7] Ali Jameel Al-Mousawi " Magnetic Explosives Detection System (MEDS) based on wirelesssensornetwork and machine learning", Measurement 151 (2020) 107112

[8] Mariusz Chmielewski, Saper - Sensor Amplified Perception for Explosives Recognition, University of Military Technology, Warsaw, Poland, 2013.

[9] Besaw, Lance E., "Detecting buried explosive hazards with handheld GPR and deep learning", Proceedings of the SPIE, Volume 9823, id. 98230N 11 pp. (2016). DOI: 10.1117/12.2223797

[10] Derek T. Anderson, Stanton R. Price, Timothy C. Havens and Anthony J. Pinar, "Computational intelligence in explosive hazard detection", SPIE, 24 December 2015

[11] Francisco J. O. Ferreira, Verginia R. Crispim, Ademir X. Silva, " USE OF ARTIFICIAL NEURAL NETWORKS IN DRUG AND EXPLOSIVE DETECTION THROUGHTOMOGRAPHIC IMAGES WITH THERMAL NEUTRONS", 2009nternational Nuclear Atlantic Conference - INAC 2009 Rio de Janeiro.RJ, Brazil, September27 to October 2, 2009 ASSOCIACAO BRASILEIRA DE ENERGÍA NUCLEAR - ABEN ISBN: 978-85 99141-03-8

[12] Huayuan Ma, Xinghua Li, "Research on Identification Technology of Explosive Vibration Based on EEMD Energy Entropy and Multiclassification SVM", Hindawi Shock and Vibration Volume 2020, Article ID 7893925, 10 pages https://doi.org/10.1155/2020/7893925

[13] William J. Peveler, Russell Binions, Stephen M. V. Hailes and Ivan P. Parkin, Detection of explosive markers using zeolite modified gas sensors" J. Mater. Chem. A, 2013, 1,2613

[14] https://ieee-dataport.org/documents/mixed-explosives-dataset

[15] Craven MA, Gardner JW, Bartlett PN. Electronic noses—Development and future prospects. Trends Anal. Chem. 1996;15:486–493.

[16] (Yuwono Lammers, 2004) Odor Pollution in the Environment and the Detection Instrumentation [1

7]*Zh*ang, L.; Tian, F.; Zhang, D. Book Review and Future Work. In Electronic Nose: Algorithmic Challenges; Springer: Singapore, 2018; pp. 335–339.

[18] COMPUTING MACHINERY AND INTELLIGENCE, A. M. TURING , Mind, Volume LIX, Issue 236, October 1950, Pages 433–460.

[19] A New Method of Mixed Gas Identification Based on a Convolutional Neural Network for Time Series Classification, Lu Han, Chongchong Yu, Kaitai Xiao, Xia Zhao

[20] Development of a LeNet-5 Gas Identification CNN Structure for Electronic Noses. Guangfen Wei, Gang Li, Jie Zhao and Aixiang He

[21] Gas Classification Using Deep Convolutional Neural Networks. Pai Peng, Xiaojin Zhao, Xiaofang Pan and Wenbin Ye

[22] Application of Convolutional Long Short-Term Memory Neural Networks to Signals Collected from a Sensor Network for Autonomous Gas Source Localization in Outdoor Environments. Christian Bilgera, Akifumi Yamamoto, Maki Sawano, Haruka Matsukura and Hiroshi Ishida.

Annexe

```
# # Decision Tree Classification
# ## Importing the libraries
# In[2]:
from keras.models import Sequential, Input, Model
from keras.layers import Embedding, GlobalMaxPooling1D, SpatialDropout1D
from sklearn.metrics import confusion_matrix, classification_report
from keras.models import load_model
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.optimizers import Adam
import time
from keras.callbacks import ModelCheckpoint
from keras.models import Sequential
from keras.layers.core import Dense, Activation, Dropout, Flatten
from matplotlib import pyplot as plt
import matplotlib.dates as md
import seaborn
import matplotlib
from keras.optimizers import Adam
from keras.layers.recurrent import LSTM
from tensorflow.keras.callbacks import ModelCheckpoint
import tensorflow as tf
from keras.regularizers import l2
from keras.layers.advanced_activations import LeakyReLU
from keras.layers.normalization import BatchNormalization
from keras.layers import Conv1D, MaxPooling1D
from keras.layers import Dense, Dropout, Flatten, Activation
import keras
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.ensemble import RandomForestClassifier
import seaborn as sns
from sklearn.metrics import confusion_matrix, accuracy_score,
↪  classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import h5py
import plotly.graph_objects as go
from sklearn.metrics import plot_confusion_matrix
# ## Importing the dataset
# In[3]:
# Reading the H5 file
filename = "mixed_explosives_dataset/sensors_data_augmented.h5"
h5f = h5py.File(filename, 'r')
sensors_data = h5f['dataset_1'][:]
h5f.close()
# The H5 file contains also the information of a temperature sensor,
↪  that you can discard
sensors_data_final = sensors_data[:, [
    1, 2, 3, 4, 5, 6], :]  # Not using temperature
# In[4]:
with open('mixed_explosives_dataset/binary_labels_augmented.txt') as f:
    lines = f.readlines()
# In[5]:
X = sensors_data_final
```

```
y = lines
# In[6]:
le = LabelEncoder()
y = le.fit_transform(y)
# In[7]:
df = pd.DataFrame([list(l) for l in X]).stack().apply(
    pd.Series).reset_index(1, drop=True)
df.index.name = 'Date'
df.columns = [str(i) for i in range(1, 301)]
df
# In[8]:
time = [i for i in range(0, 300)]
len(time)
# In[9]:
fig = go.Figure()
for i in range(0, len(X[0])):
    A = X[0][i].tolist()
    fig.add_trace(go.Scatter(x=time, y=A,
                             mode='lines',
                             name='sensor'+str(i+1),
                             line=dict(width=1)))
fig.update_layout(title='Sensors reactions over time',
                  xaxis_title='Time',
                  yaxis_title='Six chemical sensors')
fig.show()
# ## Splitting the dataset into the Training set and Test set
# In[10]:
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=0)
# In[11]:
print(X_train)
```

```
# In[12]:
print(y_train)
# In[13]:
print(X_test)
# In[14]:
print(y_test)
# ## Feature Scaling
# In[15]:
sc = StandardScaler()
for i in range(len(X_train)):
    X_train[i] = sc.fit_transform(X_train[i])
print(X_train[0])
for i in range(len(X_test)):
    X_test[i] = sc.transform(X_test[i])
for i in range(len(X_train)):
    X_train[i] = preprocessing.normalize(X_train[i])
# print(X_train)
for i in range(len(X_test)):
    X_test[i] = preprocessing.normalize(X_test[i])
fig = go.Figure()
for i in range(0, len(X_train[0])):
    A = X_train[0][i].tolist()
    fig.add_trace(go.Scatter(x=time, y=A,
                             mode='lines',
                             name='sensor'+str(i),
                             line=dict(width=1)))
fig.show()
# In[16]:
len(X_train)
# In[17]:
df = pd.DataFrame([list(l) for l in X_train]).stack().apply(
```

```
    pd.Series).reset_index(1, drop=True)
df.index.name = 'Date'
df.columns = [str(i) for i in range(1, 301)]
df
# ## Training the Decision Tree Classification model on the Training set
# In[18]:
nsamples, nx, ny = X_train.shape
d2_train_dataset = X_train.reshape((nsamples, nx*ny))
# In[19]:
classifier = DecisionTreeClassifier(criterion='entropy', random_state=0)
classifier.fit(d2_train_dataset, y_train)
# ## Predicting the Test set results
# In[20]:
nsamples, nx, ny = X_test.shape
X_test_resh = X_test.reshape((nsamples, nx*ny))
y_pred = classifier.predict(X_test_resh)
print(np.concatenate((y_pred.reshape(len(y_pred), 1),
↪  y_test.reshape(len(y_test), 1)), 1))
# ## Making the Confusion Matrix
# In[21]:
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))
sns.heatmap(cm, annot=True)
print(classification_report(y_test, y_pred))
# In[23]:
y_pred
# ## Random forest
# In[236]:
nsamples, nx, ny = X_test.shape
X_test_resh = X_test.reshape((nsamples, nx*ny))
```

```
153 classifierrand = RandomForestClassifier(
154     n_estimators=100, criterion='entropy', random_state=0)
155 classifierrand.fit(d2_train_dataset, y_train)
156 y_pred = classifierrand.predict(X_test_resh)
157 print(np.concatenate((y_pred.reshape(len(y_pred), 1),
    ↪  y_test.reshape(len(y_test), 1)), 1))
158 cm = confusion_matrix(y_test, y_pred)
159 print(cm)
160 print(accuracy_score(y_test, y_pred))
161 sns.heatmap(cm, annot=True)
162 print(classification_report(y_test, y_pred))
163 # ## KNN
164 # In[237]:
165 knn = KNeighborsClassifier(n_neighbors=2, metric='minkowski', p=2)
166 knn.fit(d2_train_dataset, y_train)
167 y_pred = knn.predict(X_test_resh)
168 print(np.concatenate((y_pred.reshape(len(y_pred), 1),
    ↪  y_test.reshape(len(y_test), 1)), 1))
169 cm = confusion_matrix(y_test, y_pred)
170 print(cm)
171 print(accuracy_score(y_test, y_pred))
172 sns.heatmap(cm, annot=True)
173 print(classification_report(y_test, y_pred))
174 # ## Linear SVM
175 # In[238]:
176 svm = SVC(kernel='linear', random_state=0)
177 svm.fit(d2_train_dataset, y_train)
178 y_pred = classifier.predict(X_test_resh)
179 print(np.concatenate((y_pred.reshape(len(y_pred), 1),
    ↪  y_test.reshape(len(y_test), 1)), 1))
180 cm = confusion_matrix(y_test, y_pred)
```

```
print(cm)
print(accuracy_score(y_test, y_pred))
sns.heatmap(cm, annot=True)
print(classification_report(y_test, y_pred))
--------------------------------------------------------------------------------
CNN
# Reading the H5 file
filename = "sensors_data_augmented.h5"
h5f = h5py.File(filename, 'r')
sensors_data = h5f['dataset_1'][:]
h5f.close()
# The H5 file contains also the information of a temperature sensor,
↪  that you can discard
sensors_data_final = sensors_data[:, [
    1, 2, 3, 4, 5, 6], :]  # Not using temperature
with open('binary_labels_augmented.txt') as f:
    lines = f.readlines()
X = sensors_data_final
y = lines
le = LabelEncoder()
y = le.fit_transform(y)

binary_labels_augmented = np.array(y)
print(binary_labels_augmented.shape)
binary_labels_augmented = np.reshape(binary_labels_augmented, (420, 1))
print(binary_labels_augmented.shape)
y_train = binary_labels_augmented[:337, :]
y_test = binary_labels_augmented[337:, :]
print(X.shape)
X = np.reshape(X, (420, 300, 6))
print(X.shape)
```

```
X_train = X[:337, :]
X_test = X[337:, :]
n_timesteps, n_features, n_outputs = X_train.shape[1], X_train.shape[2], y_train.shape[1]
model = Sequential()
model.add(Conv1D(filters=20, kernel_size=2, kernel_regularizer=l2(0.005), input_shape=(n_timesteps, n_features)))
model.add(BatchNormalization())
model.add(Activation('relu'))
# model.add(Dropout(0.1))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(filters=30, kernel_size=2, kernel_regularizer=l2(0.005)))
model.add(BatchNormalization())
model.add(Activation('relu'))
# model.add(Dropout(0.1))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(120, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(n_outputs, activation='sigmoid'))
opt = Adam(learning_rate=0.0003, decay=0.0000001)
model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])
print(model.summary())
hist = model.fit(X_train,
                 y_train,
                 batch_size=50,
                 epochs=500,
                 validation_data=(X_test, y_test),
```

```
                shuffle=True,
                callbacks=[ModelCheckpoint("model.hdf5",
                                           monitor='val_accuracy',
                                           save_best_only=True,
                                           save_weights_only=False,
                                           save_freq='epoch')])
model.load_weights('model.hdf5')
model.evaluate(X_test, y_test)
y_pred = []
model.load_weights('model.hdf5')
Y_pred = model.predict(X_test)
for elt in Y_pred:
    if elt < 0.5:
        y_pred.append(0)
    else:
        y_pred.append(1)
# print(Y_pred)
print('Confusion Matrix')
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True)
print('Classification Report')
class_labels = y_test
report = classification_report(y_test, y_pred)
print(report)
# summarize history for accuracy
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
```

```
plt.show()
# summarize history for loss
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
--------------------------------------------------------------------------------------


# Reading the H5 file
filename = "sensors_data_augmented.h5"
h5f = h5py.File(filename, 'r')
sensors_data = h5f['dataset_1'][:]
h5f.close()
# The H5 file contains also the information of a temperature sensor, that you can discard
sensors_data_final = sensors_data[:, [
    1, 2, 3, 4, 5, 6], :]  # Not using temperature
with open('binary_labels_augmented.txt') as f:
    lines = f.readlines()
X = sensors_data_final
# print(X.shape)
#X = np.reshape(X,(420, 300, 6))
# print(X.shape)
y = lines
le = LabelEncoder()
y = le.fit_transform(y)
binary_labels_augmented = np.array(y)
```

```
print(binary_labels_augmented.shape)
binary_labels_augmented = np.reshape(binary_labels_augmented, (420, 1))
print(binary_labels_augmented.shape)

sc = StandardScaler()
for i in range(len(X)):
    X[i] = sc.fit_transform(X[i])
for i in range(len(X_train)):
    X[i] = preprocessing.normalize(X[i])
X_train = X[:337, :]
X_test = X[:337, :]
y_train = binary_labels_augmented[:337, :]
y_test = binary_labels_augmented[:337, :]

opt = Adam(learning_rate=0.0005)


def build_classifier(optimizer):
    model = Sequential()
    model.add(LSTM(
                20,
                input_shape=[X_train.shape[1], X_train.shape[2]],
                recurrent_dropout=0.3,
                return_sequences=True
                )
                )
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(LSTM(30,
                    recurrent_dropout=0.3
                    )
```

```
            )
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    # model.add(Flatten())
    model.add(Dropout(0.1))
    model.add(Dense(120, activation='relu'))
    model.add(Dropout(0.1))
    model.add(Dense(84, activation='relu'))
    model.add(Dense(y_train.shape[1], activation='sigmoid'))
    from tensorflow.keras.optimizers import Adam
    opt = Adam(learning_rate=0.005)
    model.compile(loss='binary_crossentropy',
                  optimizer=opt, metrics=['accuracy'])
    return model


model = KerasClassifier(build_fn=build_classifier)
parameters = {'batch_size': [10, 20, 30, 40, 50],
              'epochs': [100, 200, 300, 400, 500],
              #  'learning_rate' :[0.1,0.01,0.001,0.0001,0.0005],
              'optimizer': ['adam', 'Adadelta', opt]}

grid_search = GridSearchCV(estimator=model,
                           param_grid=parameters,
                           scoring='accuracy',
                           cv=2)
grid_result = grid_search.fit(X, y)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_,
 ↪  grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
```

```
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))

model = Sequential()
model.add(LSTM(
          20,
          input_shape=[X_train.shape[1], X_train.shape[2]],
          recurrent_dropout=0.2,
          return_sequences=True
          )
          )
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(LSTM(30,
               recurrent_dropout=0.2
               )
          )
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dense(120, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(84, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(y_train.shape[1], activation='sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='Adam', metrics=['accuracy'])
print(model.summary())
hist = model.fit(X_train,
                 y_train,
```

```
                    batch_size=50,
                    epochs=500,
                    validation_data=(X_test, y_test),
                    shuffle=True,
                    callbacks=[ModelCheckpoint("model.hdf5",
                                               monitor='val_accuracy',
                                               save_best_only=True,
                                               save_weights_only=False,
                                               save_freq='epoch')])

model.load_weights('model.hdf5')
model.evaluate(X_test, y_test)

model.evaluate(X_train, y_train)

y_pred = []
model.load_weights('model.hdf5')
Y_pred = model.predict(X_test)
print(Y_pred)
for elt in Y_pred:
    if elt < 0.5:
        y_pred.append(0)
    else:
        y_pred.append(1)
# print(Y_pred)
print('Confusion Matrix')
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True)
print('Classification Report')
class_labels = y_test
report = classification_report(y_test, y_pred)
```

```
423 print(report)
424
425 # summarize history for accuracy
426 plt.plot(hist.history['accuracy'])
427 plt.plot(hist.history['val_accuracy'])
428 plt.title('model accuracy')
429 plt.ylabel('accuracy')
430 plt.xlabel('epoch')
431 plt.legend(['train', 'test'], loc='upper left')
432 plt.show()
433 # summarize history for loss
434 plt.plot(hist.history['loss'])
435 plt.plot(hist.history['val_loss'])
436 plt.title('model loss')
437 plt.ylabel('loss')
438 plt.xlabel('epoch')
439 plt.legend(['train', 'test'], loc='upper left')
440 plt.show()
```