ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΔΙΑΤΜΗΜΑΤΙΚΟ ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ
ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΙΑΤΡΙΚΗ
ΚΑΤΕΥΘΥΝΣΗ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΛΗΡΟΦΟΡΙΚΗ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗΝ ΑΣΦΑΛΕΙΑ, ΔΙΑΧΕΙΡΙΣΗ
ΜΕΓΑΛΟΥ ΟΓΚΟΥ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΠΡΟΣΟΜΟΙΩΣΗ»

Ευφυής Διαχείριση Δεδομένων στο Διαδίκτυο των Πραγμάτων

ΑΝΔΡΕΑΣ ΔΕΔΟΥΣΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Dr. Κωνσταντίνος Κολομβάτσος
Assistant Professor

Λαμία 2021

UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF INFORMATICS AND COMPUTATIONAL BIOMEDICINE

Intelligent Data Management in Internet of Things(IoT)

ANDREAS DEDOUSIS

FINAL THESIS

ADVISOR

Dr. Konstantinos Kolomvatso
Assistant Professor

Lamia 2021

ΠΕΡΙΛΗΨΗ

Σε αυτή τη διατριβή, μελετάμε την έξυπνη διαχείριση δεδομένων που παράγεται από συσκευές IoT. Αυτή η έρευνα επικεντρώνεται σε ένα σενάριο σε εφαρμογές Edge Computing για το Internet of Things (IoT). Οι κόμβοι IoT υποστηρίζουν την αυτόματη εκμάθηση μηχανών (Machine Learning) για τη δημιουργία ενός μοντέλου χωρίς ανθρώπινη αλληλεπίδραση. Στην περίπτωσή μας, χρησιμοποιήσαμε μη εποπτευόμενη εκμάθηση χαρακτηριστικών (unsupervised feature learning) για να προβλέψουμε το μελλοντικό μοντέλο χρησιμοποιώντας μερικές προτεινόμενες μεθόδους όπως το Principal Component Analysis (PCA), Locally Linear Embedding (LLE), Multidimensional Scaling (MDS) συγκεντρώνοντας και συγκρίνοντας τα αποτελέσματα αυτών των αλγορίθμων σε κάθε διάσταση και στοιχείο για να βρούμε τις πιο σημαντικές παραμέτρους. Τα αποτελέσματα που συλλέγονται μπορούν να εφαρμοστούν σε μοντέλα Machine Learning όπως το Transfer Learning για να κατανοήσουν πώς αλλάζει το περιβάλλον και παρέχονται λεπτομερείς πληροφορίες για αυτές τις μεθόδους που αναφέρονται παραπάνω. Ο στόχος είναι επίσης, η μελέτη νέων τεχνικών επεξεργασίας και παρακολούθησης δεδομένων και η ανάπτυξη των απαιτήσεων για το παραπάνω μοντέλο.

ABSTRACT

In this thesis, we are studying the intelligent data management produced by IoT devices. This research focuses on a scenario in Edge Computing for Internet of Things (IoT) applications. IoT nodes support auto Machine Learning to create a model without human interaction. In our case we used unsupervised feature learning to predict the future model by using some proposed methods like Principal Component Analysis (PCA), Locally Linear Embedding (LLE), Multidimensional Scaling (MDS) aggregating and comparing the results of these algorithms in each dimension and component to find the most important parameters. The results gathered can be applied in Machine Learning models like Transfer Learning to understand how the context changes and provide detailed insights into these methods mentioned above.The goal is also to study new data processing and tracking techniques and develop the requirements for the model above

LIST OF TABLES

3.2 Table of Input Feeding Component

3.3.1 Table showing PCA algorithm

3.3.2 Table showing LLE algorithm

3.3.3 Table showing MDS Algorithm

4.3.1 PCA results

4.3.2 MDS results

4.3.3 LLE results

5.1 Common Results plot

LIST OF FIGURES

2.3 Figure Machine Learning

2.3.1 Figure Transfer learning

3.1 Figure Showing the workflow

3.3 Figure showing the results of PCA, LLE, MDS

6.3.1 Figure: Running Cycle

Table of Contents

ΠΕΡΙΛΗΨΗ 1
ABSTRACT 3
LIST OF TABLES 5
3.2 Table of Input Feeding Component 5
3.3.1 Table showing PCA algorithm 5
3.3.2 Table showing LLE algorithm 5
3.3.3 Table showing MDS Algorithm 5
4.3.1 PCA results 5
4.3.2 MDS results 5
4.3.3 LLE results 5
5.1 Common Results plot 5
LIST OF FIGURES 0
2.3 Figure Machine Learning
2.3.1 Figure Transfer learning 0
3.1 Figure Showing the workflow 0
3.3 Figure showing the results of PCA, LLE, MDS 0
6.3.1 Figure: Running Cycle 0

Table of Contents 2

Introduction 5
1.1 Objectives 5
1.2 Presentation of Chapters 6

Preliminaries 8
2.1 Internet of Things (IoT) 8
2.1.1 Processing IoT Data 9
2.2 Machine Learning 10
2.3 Transfer Learning 13
2.3.1 Transfer Learning with Language Data 14

Methodology 16
3.1 Workflow 16
3.2 Input Feeding 17
3.3 Unsupervised Feature Learning 18
3.4 Common Parameters 22

Model 24

4.1 Libraries 24
4.2 Data Collection 26
4.3 Apply Unsupervised Feature Learning 27
4.4 Fitting Common Data 29

Conclusions & Future Work **32**
5.1 Conclusions 32
5.2 Lessons Learned 33
5.3 Future Work 33

Bibliography **35**

Algorithm **37**
Libraries 37
Main program 38
Support functions 42
Running Cycle 45

Chapter 1

Introduction

The Internet of Things is a network of physical things that are integrated with electronics, software, sensors, actuators, and connections, allowing them to communicate with one another. This infrastructure permits the development of applications and services that facilitate the lives of citizens around the world and bring us closer to the convergence of the physical and digital worlds.

Smart-grids, smart-homes, intelligent transportation, and smart-cities are among the domains where the applications created give solutions. Some examples include smart-heating, Phillips Hue for smart-lighting, Fitbit, Xiaomi. Withings for personal fitness trackers, smart cars like Tesla, as well as the personal assistant like Alexa from Amazon or Siri from Apple.

The amount of data that can generate a single sensor device installed in a house or a wearable device worn by a person can be overwhelming for the device itself and in many cases needs to be offloaded to a data-processing application from which the end-users can access it and connect it to a more meaningful use.

1.1 Objectives

In order to provide intelligent data management solutions it is important to be able to integrate the data that are generated by all of these smart devices in a common base. For this reason we introduce the meaning of edges. Edges will collect data generated from IoT devices and send them in batches on the cloud in order to deload the volume of the data on the IoT devices themself. So we need to be able to process them, view them and operate on them using a common methodology and also be able to acquire common knowledge from them.

The scenario we focus on a scenario in edge computing, where we assume having N nodes and these nodes accept multidimensional vectors <x1, x2, x3,...,xM> with values from actual IoT nodes.Processing these data and extracting the significance of them will help us achieve the build of a ML model describing the important parameters in each dimension of those IoT devices.

The first goal of our work is to set up the environment representing the data IoT devices produce. We want to be clear and understandable what is

an IoT device and what is a Node-Edge in our experiment. For this reason we parametrize everything to be configurable in order to have more flexibility on our handling.

Afterwards, we want to analyze the values of these data to have dimensionality reduction with some Unsupervised Feature Learning Algorithms like PCA, LLE, MDS where we are analyzing them in detail later in this thesis.

Finally we extract the results knowing the importance of them and find common results in each dimension which will help us fulfill our goal which is to produce a model suitable for Transfer Learning.

1.2 Presentation of Chapters

This section presents a brief description of the structure of the dissertation which consists of the following chapters.

In Chapter 2 presents the theoretical background needed to understand basic terminologies like Internet of Things (IoT), Machine Learning (ML) and finally Transfer Learning. These terminologies are necessary in order to go deeper into the area of modeling and unsupervised learning.

We study what the internet of things is, how it started and the way it will spread in our future and in our everyday life. We also present a selected list of research applications that leverageIoTtechnologies in the areas ofSmart Buildings Smart Cities focusing on data collection, as well as the challenges in analyzing and validating such data. Also we make a small introduction on how we manage and process Iot Data and the steps needed to be done before the processing.

Afterwards, we continue with machine learning by making a small introduction on what machine learning is and how it's been used.

Finally at the last part of this chapter we are talking about transfer learning on its uses, how they are applied and also the two approaches used to represent transfer learning. The last part concerns the Transfer

Learning with language- data namely what we are using in this master thesis and 2 well known transfer learning techniques.

In Chapter 3 we are describing the basic methodology we used to represent our scenario by showing how each component is connected and analyzing the flow of the workflow we made.

This is made by splitting the workflow in 3 steps. The input feeding step which is how we collected the input data and what decisions we made in order to feed the next step which is the unsupervised Feature Learning. This step is analyzing the input data with 3 algorithms PCA, MDS, LLE and performing dimensionality reduction preparing them for the last part.

The last part is useful because we are aggravating the data finding the most important we need and finding the common important values of the algorithms mentioned above.This has as a result to have the data ready for any ML model needed for further analysis.

In Chapter 4 we are presenting the actual algorithm and the supporting functions we used to accomplish this thesis as well as the libraries and the language used. Also we make a reference to the running cycle of the program on how it will execute and what is the output.

Chapter 5 presents the general conclusions of the dissertation and suggests specific fields for future extensions. Finally, the bibliography used to write this dissertation and the appendices are listed.

Chapter 2

Preliminaries

This chapter presents essential background information related to the topic described in this thesis. We present the main concepts to (i) Internet of Things(IoT),(ii) Machine Learning (ML), (iii) Transfer Learning.

2.1 Internet of Things (IoT)

While there is a huge interest on the Internet of Things (IoT) there is not a specific definition explaining exactly what IoT is. The Internet of Things (IoT) is a networked system made up of computing devices, analog and digital machinery, products, animals, or people with unique identifiers Also Internet of Things presents the capacity to transfer data across a network without requiring human-to-human or human-to-computer contact.

Any natural or man made object that can be assigned with adn IP address and have the ability to transfer insights and data over the network are examples of Internet of Things. As very good examples we can give a person with a monitor implanted or weather sensors in agriculture ,a smart home controlling basic house operations or a car with sensors about tires or weather.All these are the Internet of Things.Experts anticipate that the Internet of Things will have 17.6 billion items by 2020, a number far less than the original predictions of 1 trillion or 50 billion devices, but still quite high1(when compared to the number of devices

currently connected to the Internet). It is also estimated that the global market value of IoT reached $7.1 trillion by 2020.

IoT devices allow items to remotely operated through existing network infrastructure, allowing for more direct physical world integration with computer-based systems as a result, there is more efficiency, precision, and economic gain, as well as less human interaction.The technology falls within the broader category of cyber-physical systems, which includes smart grids, virtual power plants, smart homes, intelligent transportation, and smart cities that's happening when IoT devices are combined with sensors and actuators.

The Data IoTs devices generate do not follow the typical characteristics of data processing technologies and methodologies past computer science uses as their volume and size usually exceeds the size a single machine can process in real time. The first attempts to manipulate such data were made using big data technologies like Hadoop. We can say that Hadoop is an open-source software for storing data and running applications on clusters of commodity hardware. It provides massive storage for any kind of data, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs.

2.1.1 How IoT Data are being Processed

Nowadays the quantity of data generated by IoT sensors and devices is massive so we must process them.This processing will happen in order to make data ready to be used.

Nonetheless, sincere may see data originated from a variety of sources-devices and in a variety of forms and formats, there are a few things we must do before processing or applying any sort of analytics to it:

- We must do a standardization or transfotion to the data to a uniform format,so that it will ensure that this format is compatible with any application.

- We must handle all the new transformed data by storing and creating backups.

- To increase accuracy, filter away any data that is repeated, old, or unnecessary.

- Finally, to assist improve your present data collection, include more structured (or unstructured) data from other sources.

2.2 Machine Learning (ML)

Machine learning as a term or ML, is a technology focusing on the use of data and algorithms in a way is imitating the way a human gradually learns and improves the accuracy in learning something. Artificial intelligence (AI) is strictly connected with machine learning and computer science as ML is a branch of them.

ML is a key part of the data science as it develops. ML is using algorithms and statistical methods for predictions,classifications and in general learning. Through this machine learning ML uncovers insights for data mining.

Following that, these insights drive decision-making within applications and enterprises, with the goal of influencing important growth KPIs.

How machine learning actually works lets see:

1. **Decision Processing**: We use Machine learning algorithms in order to make predictions or classifications on some input. The chosen algorithm will take this input and it will try to produce an estimation pattern in the data based on the input data we gave.These data can be labeled or unlabeled.

2. **An Error Function**: The model will be generated by the step above will give a prediction and this will be evaluated using an error function. If there are reported cases, error functions can be used to analyze and determine the accuracy of the model we have.

3. **An Model Optimization Process**: The model produced will have some weights at its nodes. Theseweights will be constantly evaluated and optimized in order to be adjusted to the the maximum level of accuracy for the model. This will be done to minimze the difference between the old known example and the new one because we want to fit the model better into the training set.

Classifiers are used in machine learning. These classifiers can be divided in 3 groups Supervised machine learning,Unsupervised machine learning,Semi-supervised learning , as shown below:

Supervised machine learning

Supervised learning, often is also called as supervised machine learning, is distinguished because it uses the datasets with labels to train algorithms that reliably categorize the data or divine, predict some results. Organizations may use supervised learning to tackle a range of actual issues at scale, such as spam classification in a distinct folder from your email inbox.As more input data is introduced into the model, the weights are updated until the model is well fitted. A well fitted model is very important to avoid some consequences. This happens throughout the validation process in order to verify that the model does not overfit, underfit. The supervised learning is using some common methods like linear-logistic regression, support vector machine, neural networks, naïve bayes.

Unsupervised machine learning

Unsupervised learning, often known as unsupervised machine learning,analyzes with the help and use of machine learning algorithms. Unsupervised machine learning can also cluster unlabeled datasets. Without the need for human interaction, these algorithms identify hidden patterns or data groupings. Because of its capacity to find similarities and

contrasts in data, it's perfect for some in depth approaches like data analysis, cross-selling techniques, consumer segmentation, picture and pattern recognition.This method is also utilized in the dimensionality reduction process to lower the amount of features in a model. This is succeeded with some algorithms-approches like principal component analysis (PCA),singular value decomposition (SVD),local linear embedding (LLE), Multidimensional Scaling (MDS). Also to end with for unsupervised learning there are approaches-methods for deep neural networks, clustering with k-means and probabilistic.

Semi-supervised learning

Semi-supervised learning stands good between the other two mentioned above and it's a very good compromise. It assists classification and feature extraction from a wider, unlabeled data set using only a smaller labeled data set during training.

Let's take a closer look at why Machine Learning is so important. The same dynamics affecting data mining and Bayesian analysis becoming more popular than ever are also pushing machine learning to the same direction. ThoseFactors include expanding data quantities- data variety, the ability to have cheaper and more powerful computing processing for modeling, and the availability in larger data storages.

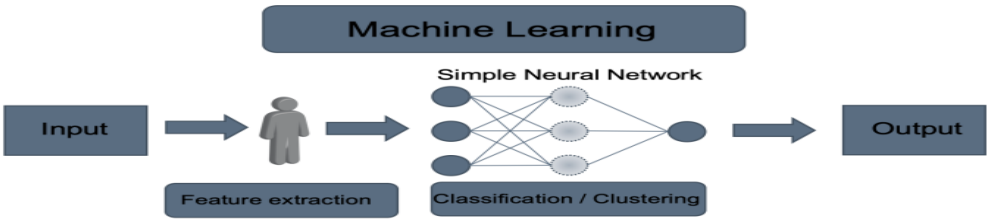


Figure 2.2: Machine learning

2.3 Transfer Learning

With the term Transfer learning we can say that it is a ML method which is reusing as the beginning point a model which is developed for another task to a next task. Deep learning techniques are using Transfer Learning as an approach in pre-trained models which are needed to begin NLP tasks. Given the enormous computational and time resources necessary to create neural network models for these challenges, as well as the enormous leaps they provide on related tasks,. We can see there are two common approaches on transfer learning for predictive modeling problems as shown below the Develop Model Approach and Pre-trained Model Approach. The second approach is common in deep learning.

- Develop Model Approach

1. **Source Task**. A large variety of data must be chosen-selected carefully with a relevant predictive modeling issue. These data must have a relationship between them either on input to output data or during the connection developed during the mapping from input to output.

2. **Develop Source Model**. In this step the model is being developed to be routed for the first piece of work. This model have to be enough better than any other model to indicate some type of feature learning has taken place to it.

3. **Reuse Model**. Depending on the model technique used, the model fit on the first step can be utilized later as beginning node to another task.Either by using all of the model or some parts of it.

4. **Tune Model**.Occasionally, there is a possibility for an enhancement or converting to be done in the model on the input-output pair data available.

- Pre-trained Model Approach

1. **Select the Source-Model**. A pre-trained source model is chosen to be used as input from some available models. It is commonly used by many scientific research institutes to publish their models on large datasets they have developed that may be perfect applicants models to choose from for any experiment.

2. **Reuse Model**. The model from the previous step mentioned can afterwards be utilized again as a beginning node for a model on the next task. According to the modeling methodology employed, this may include utilizing all or sections of the model.

3. **Tune Model**. On the input-output pair data available for the job of interest, the model will may be need to be altered or enhanced.

2.3.1 Transfer Learning with Language Data

The transfer learning is usually performed in natural language problems (NLP) where they use this text as input or output.Word embeddings are used for NLPs, a word embedding is a mapping of words to a high-dimensional continuous vector space where different words with a similar meaning have a similar vector representation. There have been implemented, many efficient algorithms in order to support the learning process of these representations and it is quite common, pre trained models to be publicly available from many research institutes to be used with a permissive licence.

Models of this type:

- Google's word2vec Model
- Stanford's GloVe Model

The reason we choose to use transfer learning models is for better optimization. This will affect time and better performance. The benefits can be listed as follows:

A. Higher start.
B. Higher slope.
C. Higher asymptote.

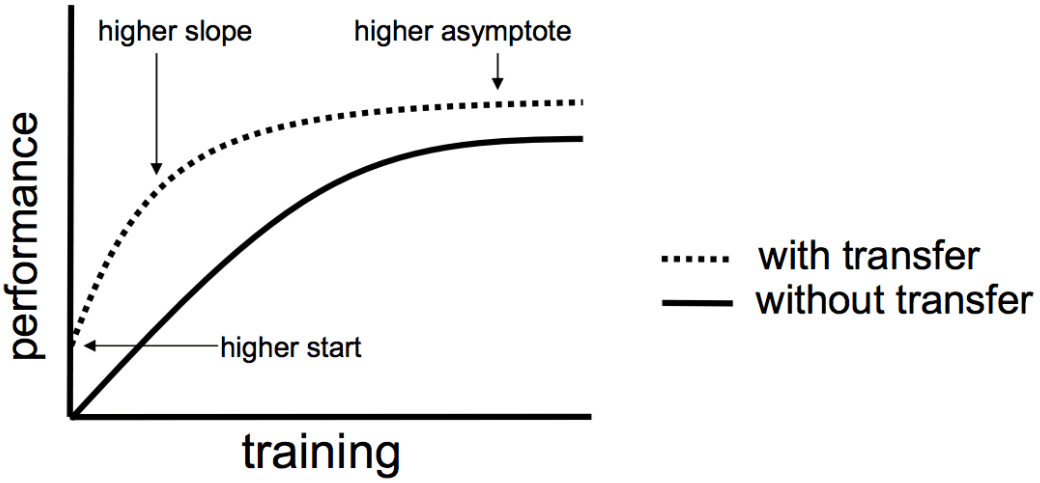


Figure 2.3.1 : Transfer learning

Chapter 3

Methodology

This chapter will present our approach and methodology for extracting a Machine Learning model representing a scenario in edge computing, as well as the main components of our work and experiment.

3.1 Workflow

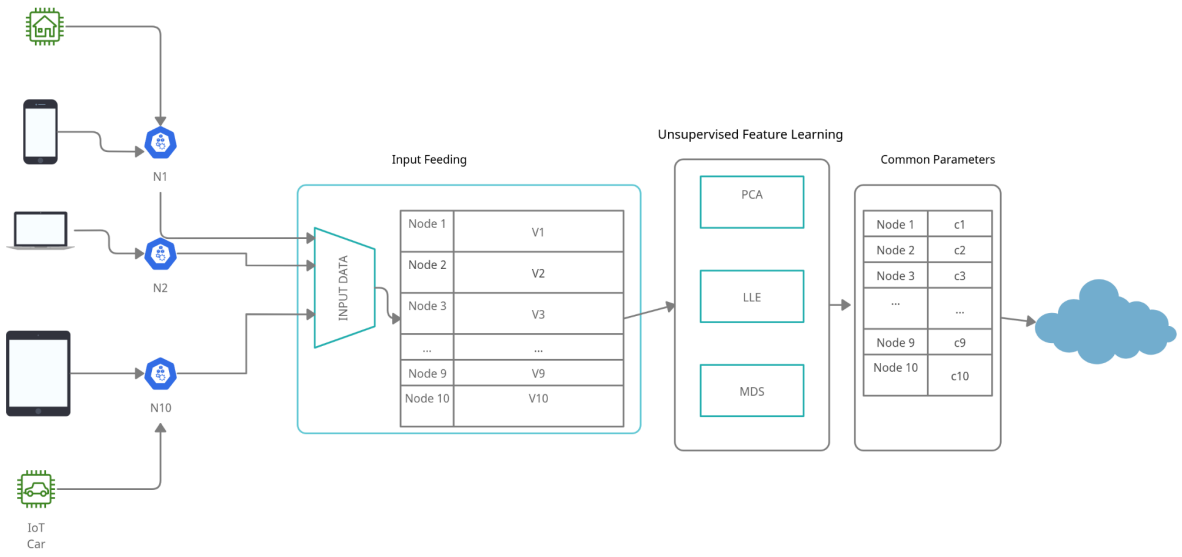


Figure 3.1: Workflow

To enable the extraction of the ML model for a large amount of data from different IoT devices we propose the system above. Figure 3.1, which presents in abstract level the workflow we follow in our approach.

We have three main steps which we will analyze further below and some additional. The main steps are The Input Feeding, The unsupervised Feature Learning and the common Parameters all these combined will help us find the Machine Learning model we want and share this model with the other IoT nodes and finally send it on the cloud.

3.2 Input Feeding

In order to begin the Input Feeding initial step we make some assumptions. Firstly we have the IoT devices connected to N nodes.The data we have are constructed from x Nodes and y dimensions. We fed our input data with configurable values because we needed different values in our experiments for Nodes and Dimensions.

Assuming we have N nodes and these nodes accept multidimensional vectors <x1, x2, x3,...,xM> with values from actual IoT nodes.The main experiment we run had 10 Nodes, each node had 5000 dimensions and with values from [0,1]

Each Node accepts data and stores them locally for this reason we accepted the data being at this range. We define the input data as follows at the table below. Table 3.2.

	Dimensions[1, 2, 3, …,4998, 4999, 5000]
Node 1	[0.69975656 0.24474762 0.9198572 ... 0.98530185 0.14884073 0.38440713]
Node 2	[0.7076864 0.7795434 0.40902424 ... 0.01022692 0.7136915 0.3037559]
Node 3	[0.47044194 0.92595315 0.09213836 ... 0.48703355 0.78172064 0.9407172]
..	...
Node 8	[0.7840641 0.41346282 0.42621794 ... 0.8503634 0.10175344 0.24615629]
Node 9	[0.99341273 0.59468806 0.07666443 ... 0.17012808 0.41915613 0.21318962]
Node 10	[0.5234284 0.6672067 0.8624287 ... 0.22812828 0.25042918 0.31817374]

Table 3.2: Table showing an example of the input data feeding component

In this work we chose to have random input data and do not use an existing dataset in order to achieve a better value variance. The data we constructed will be used in the next component after some aggregation.

3.3 Unsupervised Feature Learning

In our scenario,we continue in the second component of Figure 3.1 which is the Unsupervised Feature Learning component. Here we chose to have 3 methods in this master thesis in order to proceed with dimension reduction.

With the term Dimensionality reduction or dimension reduction is a procedure of reducing the number of random variables by obtaining a set of principal variables. Dimension reduction strategies are used to condense the original p-dimensional data space into a reduced k-dimensional components subspace.

These three methods are **Principal Component Analysis (PCA), Locally Linear Embedding (LLE) and Multidimensional Scaling (MDS)**.

- **PCA**: Principal Component Analysis, is a dimensionality-reduction algorithm that is used in large datasets to reduce the dimensionality without the loss of important information. This is achieved by transforming sets of variables into smaller ones. When reduction of variables is performed it comes with the cost of accuracy but its better losing accuracy over simplicity of data.The advantage of having smaller data sets is that these are easier to manipulate them by analyzing or visualization. Also when the datasets are small and simple there is not the cost of slower machine learning and expensive processing power.

1. Compute the mean feature vector

   $\mu = \frac{1}{p}\sum_{k=1}^{p} x_k$, where, x_k is a pattern (k = 1 to p), p = number of patterns, x is the feature matrix
2. Find the covariance matrix

   $C = \frac{1}{p}\sum_{k=1}^{p}\{x_k - \mu\}\{x_k - \mu\}^T$ where, T represents matrix transposition
3. Compute Eigen values λ_i and Eigen vectors v_i of covariance matrix

   $Cv_i = \lambda_i v_i$ (i = 1, 2, 3,....q), q = number of features
4. Estimating high-valued Eigen vectors
   (i) Arrange all the Eigen values (λ_i) in descending order
   (ii) Choose a threshold value, θ
   (iii) Number of high-valued λ_i can be chosen so as to satisfy the relationship

   $\left(\sum_{i=1}^{s}\lambda_i\right)\left(\sum_{i=1}^{q}\lambda_i\right)^{-1} \geq \theta$, where, s = number of high valued λ_i chosen

   (iv) Select Eigen vectors corresponding to selected high valued λ_i
5. Extract low dimensional feature vectors (principal components) from raw feature matrix.

   $P = V^T x$, where, V is the matrix of principal components and x is the feature matrix

Table 3.3.1: Table showing the PCA algorithm

- **LLE**: **LLE** is a topology preserving learning method. All manifold learning algorithms assume that the dataset lies on a smooth nonlinear manifold of low dimension and a mapping ***f:*** $\boldsymbol{R^D \rightarrow R^d}$ (D>>d) will be found by preserving one or more properties of the upper dimension space. Unlike distance preservation methods like MDS as we'll describe below, topology preservation is more logical. The points which are situated nearby in higher dimension should after LLE be drawn in lower dimension. LLE creates the lattice supporting the knowledge contained within the dataset.

LLE ALGORITHM

1. Compute the neighbors of each data point, $\vec{X}_i$.
2. Compute the weights W_{ij} that best reconstruct each data point $\vec{X}_i$ from its neighbors, minimizing the cost in eq. (1) by constrained linear fits.
3. Compute the vectors $\vec{Y}_i$ best reconstructed by the weights W_{ij}, minimizing the quadratic form in eq. (2) by its bottom nonzero eigenvectors.

Table 3.3.2: LLE algorithm

- **MDS: Multidimensional scaling**, also referred as Principal Coordinates Analysis (PCoA). The data used for multidimensional scaling (MDS) are dissimilarities between pairs of objects. MDS has as main object to show these dissimilarities as distances. These distances should correspond as closely as possible to the dissimilarities.

The MDS algorithm

- ***Input:*** n_xn distance matrix **D**
- Random **n** points in the **k**-dimensional space $(x_1,\ldots,x_n)$
- **stop = false**
- **while not stop**
  - **totalerror = 0.0**
  - For every **i,j** compute
    - $D'(i,j)=d(x_i,x_j)$
    - **error** = $(D(i,j)-D'(i,j))/D(i,j)$
    - **totalerror +=error**
    - **For** every dimension **m**: $x_{im} = (x_{im}-x_{jm})/D'(i,j)*error$
  - **If totalerror** small enough, **stop = true**

Table 3.3.3: Table showing MDS Algorithm

We used each algorithm PCA, LLE MDS given the input data with 3 components as parameters. But lets see what components are. Components are vital to determine the dimensionality reduction.

We need to find a way to find the correct number of the components to use. So this can be easily determined by looking at the cumulative explained variance.This number if it is used as function can quantify how much of the total 64-dimensional variance is contained within the first *N* components. In our experiment, we saw that with two-dimensional projection we have loss of a lot of information as measured by the explained variance so we used 3 components instead at the first part of this thesis.

The result of PCA,LLE, MDS is an array of 10 Nodes with 3 Components each. In order to be easier to plot these data we decided to aggregate these results choosing the 2 bigger values. In this way we have a reduction on 2 Components without losing vital information

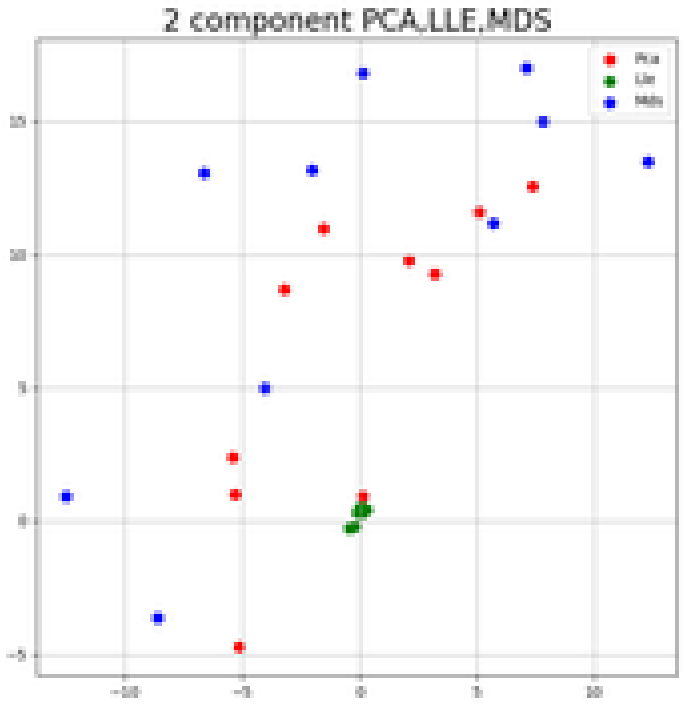


Figure 3.3 : Plot showing in 2 dimensions the results of PCA,LLE,MDS

As we can see in Figure 3.3 the variance we have in results of PCA . MDS is much larger than this in LLE results. LLE algorithm is responsible for reducing and transforming the data to a more reduced dimension space. This will happen by using a linear combination of the original variables. The main goal of LLE is to reorder the original variables by a smaller set of underlying variables. Thas because we are seeing the green dots range between 0.

Also as we said before PCA algorithm is a linear dimensionality reduction technique.PCA is not so accurate to represent nonlinear data but is quite efficient.As a result, PCA is not good at discriminating data and for this reason may not lead to an interesting viewpoint for data clustering.

Whereas PCA takes a set of points in R^n and gives a mapping of the points in R^d, Multidimensional Scaling (MDS) takes a matrix of pairwise distances and gives a mapping into R^d.

MDS algorithm has some main advantages which will be reported below.MDS is quite simple to implement and very useful in visualization. It can discover hidden structures inside the data.

On the other hand MDS has some disadvantages like the difficulty in selecting the best dimension of the map ,the hardness to visualize small distances corresponding to the local structure whereas the PCA algorithm can do it easily. Another weakness is MDS's high computational and memory complexity, $O(n^3)$ and $O(n^2)$, respectively.

3.4 Common Parameters

In the last component of our workflow of Figure3.1 we have the common parameters component. This component's use is to find the common values produced earlier of the 3 algorithms.

Finding the common values is a task that was done in each Node of the workflow for each component.In order to be more clear i will give a simple example of this. We assume having one dimension:

1. PCA has 2 components with important parameters[2,7]

2. LLE has 2 component with important parameters [7,8]

3. MDS has 2 component with important parameters [10,7]

So now we merge the three lists in each dimension holding only the important common parameters of PCA, LLE, MDS. These parameters are the one they will eventually feed the model for Transfer Learning.

Chapter 4

Model

Below we will present what we used to implement each component of the workflow and in detail we will present the model we implemented to support our methodology and our experiments.

4.1 Libraries

In order to achieve all the things mentioned above we used Python programming language.Python is a scripting language with huge capabilities in Machine Learning ,in DataAnalysis and visualization.

Python programming language can offer concise and readable code. The developers using python's simplicity for Machine Learnings (ML) and Artificial Intelligence (AI) can write reliable systems with libraries for complex algorithms and versatile workflows.

Python is quite easy to learn because it is very flexible in variables or compiling and running so many developers choose Python for their implementation. Python is a language that it can do a set of complex machine learning tasks and enable the developer to build prototypes really quickly. This allows the developer to test the product easier for machine learning subjects.

Also we used the numpy python library to have the control of reshaping the data. NumPy is a Python library which is frequently used for working with linear algebra arrays. The object array in Numpy is called ndarray and this python library is providing many supporting functions in order to make multiple complex operations with arrays which make it very easy.Numpy and Arrays are frequently used in data science, where speed and resources are very important.

Furthermore in order to implement PCA, LLE, MDS we used sklearn library. Scikit-learn library is the most useful library in Python for machine learning (ML) because it contains many tools which increase

efficiency in machine learning,regression,clustering and dimensionality reduction. Scikit-learn has many features where they will be analyzed below in order to have a better understanding on the spread of this library:

- **Supervised learning algorithms:** It is very frequent that many known supervised machine learning algorithms will be part of scikit-learn. A good example of these algorithms is Linear Regression, Support Vector Machines (SVM), Decision Trees to Bayesian methods. Scikit-Learn is commonly utilized due to its spread in machine learning (ML).

- **Cross-validation:** Scikit-Learn includes a high variety of methods to check supervised models accuracy.

- **Unsupervised learning algorithms:**LLE,PCA, clustering are some of the algorithms Scikit-Learn has for unsupervised learning.

- **Datasets:** Many academic datasets for training are available in scikit-learn like IRIS dataset, Boston House prices dataset.

- **Feature extraction:** Features from images and text can finally be extracted from scikit-learn library.

In the end we used Pandas library to make some tables and the plots for the results of our experiments. Pandas library in python is very useful because it gives the opportunity to the developper to easily manipulate data and analyze them.

4.2 Data Collection

The total data we chose for our model concern the workflow described in the previous chapter.

According to the above the data we collected are from imaginary IoT devices and we chose in our experiment random values between [0,1] to

represent the data. This data might be data from health deviced like smartwatches, smart houses, smart cars ,smartphones etc. We actually decided to have this range ([0,1]) to prevent loss of generality.

Loss of generality is a term that is used in proofs. This will indicate that an assumption is being made which does not introduce new restrictions to the problem. Without loss of generality is often pronounced **WLOG** or **WOLOG**. WLOG means that it is ok to assume a value for a variable, or other such unknown, in order to solve the problem. This can be often worn out problems concerning ratios, or the other value that is still constant irrespective of what's assumed.

For our model creation we used N Nodes in the edge representing the actual nodes the IoT devices will use to send their data. In our experiments we used 10 Nodes and we want to apply an ML model for our data in each node and then combine them all together to send the new model created on the cloud.

A main part of this thesis is the data reshaping and preprocessing. We wanted to reshape the data in order to fit our unsupervised feature learning algorithms and not have some crucial information loss , we also pre-processed them by rounding either to the first decimal or the second. This is done to have a more clear view of the input data and dont have unnecessary noise.

4.3 Apply Unsupervised Feature Learning

In this subsection we will analyze in detail what we made for our data to produce the final model

We used 3 unsupervised Feature Learning Algorithms PCA, MDS and LLE on our data of each Node.We got a result as we can see in the three tables below Table4.3.1,Table 4.3.2, Table 4.3.3 for 3 components.

Table 4.3.1: PCA results

component 1	component 2	component 3
-13.26588024087868	12.71288867313977	-3.8488004163666725
7.850791912658227	8.752964558197734	-14.831639091064714
-9.765112870815162	-2.348475966112154	-15.823039688344606
-9.868950949697908	4.756897823841077	15.40988917066247
3.287023387893008	-15.737480543466456	-10.140439686309866
5.35722969235703	17.990730056627466	3.3586857104460814
3.2895243646733925	-15.666321374563319	10.698366791355786
10.75481508483118	2.7041920788255887	15.03739911048357
18.09337745545815	-3.072431522044172	-2.9715340712358183
-15.732817836479228	-10.092963784445535	3.111112170373764

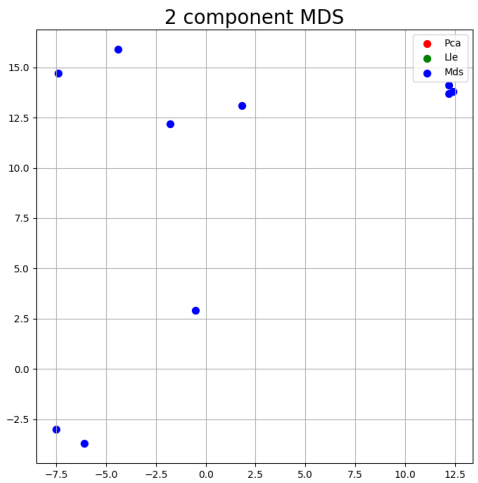
Table 4.3.3: LLE results

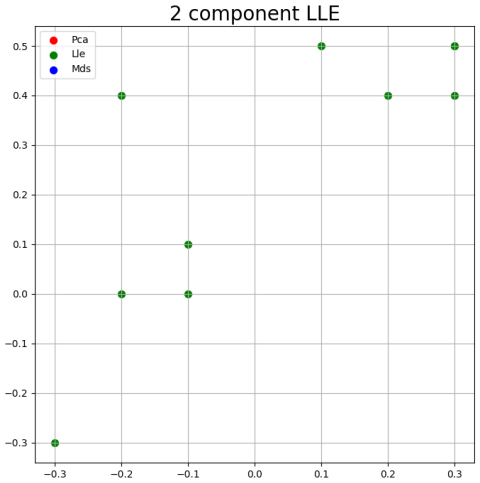
component 1	component 2	component 3
-0.5069296618269916	0.018331994865336848	-0.15370091678547435
-0.056306624516116344	-0.47518627256075857	0.05549254128061582
-0.08822163570239538	0.48510632395159403	0.11629168929666184
0.3174459502749084	-0.24914699773920154	0.5269589795839017
-0.2573649996217606	0.29146278177328294	0.4630518739700428
0.23724164966414335	0.35329799685186425	-0.5745994601450761
-0.35472295906084206	-0.27769811284263063	-0.27153589761021035
0.41658391970368874	-0.3250697537930366	-0.24261091885432257
-0.14256150265786205	-0.09480422169878597	0.011504862003174032
0.4348358637432258	0.27370626119233543	0.06914724726068706

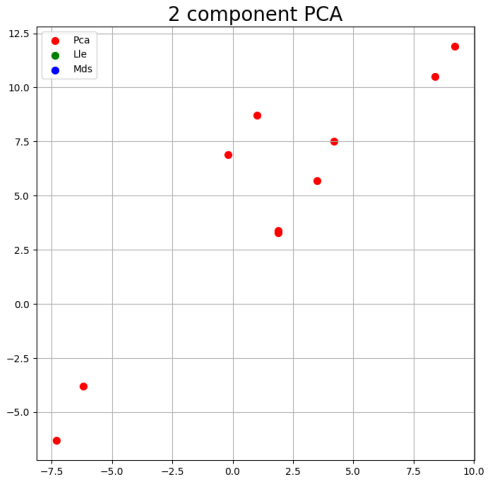
Table 4.3.2:MDS results

After we got the results we chose to aggregate them by finding the 2 most significant values in each dimension. This is done by searching for the 2 bigger values for each algorithm and in each dimension.

The next step was to round this results at 1 decimal places to achieve a more accurate result to support oure plots. After the tables transformation and aggregation the results are entered into some functions to plot them and show the variance of each algorithm. We achieved that with the pyplot library as mentioned above.







4.4 Fitting Common Data

After the supervised learning we applied a technique to find the common values generated by PCA, LLE, MDS in each dimension in order to find the most important values and construct the model from these values. We tested the model analyzed above running the python script 50 times to have a better understanding of the results and we found that with precision to one decimal we have common results in about 75% of the total

runs. Our goal is to have as many as common results we can to have data to analyze.

These results can be used for ML models as input to calculate the transfer learning to see when the context changes. This means that these parameters are important for a particular context. The plot in Table 5.1 show as an exportation

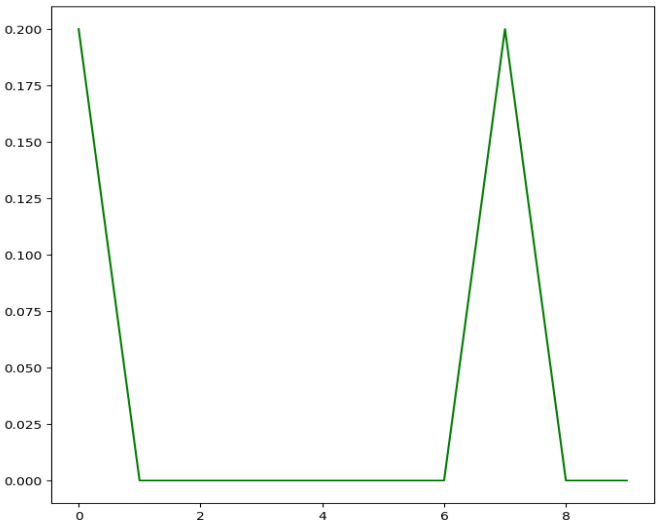


Table 5.1: Common results

Common
0.2
0.0
0.0
0.0
0.0
0.0
0.0
0.2
0.0
0.0

Table 5.2: Common Results table

of the common results data showing that we have common results in dimension 1 and in dimension 7. These common results are the one that will fit in the model and are those results we found that are the most important between PCA, LLE, MDS algorithms.

Chapter 5

Conclusions & Future Work

In this chapter is presented the main conclusions from our work in the context of this thesis, the lessons we learned and some possible future extensions

5.1 Conclusions

In the context of this thesis, we are dealing with the problem that comes from the introduction of more and more smart and connected devices in our everyday environments. This increasing use of IoT devices is what makes an improved data management methodology essential for making use of the data that are generated.Our work was based on a scenario produced in the edge computing providing us with a testing environment to apply and validate our solution with the edge nodes.

As a result, we can now present an idea capable of efficiently collecting and analyzing data from a huge number of IoT devices and providing solutions on how these data can be processed. We also proved that our workflow design can handle various data types and inputs with minimal changes by feeding it with random numbers every time.

The algorithm we used on our model has an accuracy prediction of about 75% on our 50 runs experiment. This means that in real time data whose variance would be much lower this percentage would be over 85%. So with the help of unsupervised feature learning and the use of the results to make a model ready for an machine learning algorithm we achieved to efficiently manage IoT devices data.

5.3 Future Work

Knowing the amount of data of the IoT devices and having in mind that in the future years IoT devices will become a greater part in our everyday life we have to find a way to manage all this information and data even more efficiently.

An important extension to the work presented in this thesis would be the input data feeding would be from a real time scenario. This means having collected multiple IoT devices sending data to some nodes and collecting these data and fit them to our model. With this the results would be more accurate.

Also another extension this thesis can be improved is by applying the Transfer Learning layer and producing the ML model to compute and analyze all the above data and finally send them to the Cloud.

Bibliography

- https://builtin.com/data-science/step-step-explanation-principal-component-analysis
- https://blog.paperspace.com/dimension-reduction-with-lle/
- https://medium.datadriveninvestor.com/the-multidimensional-scaling-mds-algorithm-for-dimensionality-reduction-9211f7fa5345
- https://scialert.net/fulltext/?doi=jai.2010.119.134
- https://steelkiwi.com/blog/python-for-ai-and-machine-learning/
- https://www.analyticsvidhya.com/blog/2015/01/scikit-learn-python-machine-learning-tool/
- https://en.wikipedia.org/wiki/Pandas_(software)
- http://www.sciencedirect.com/science/article/pii/S0747563216302990
- https://www.sas.com/en_in/insights/big-data/hadoop.html
- https://www.toolbox.com/tech/iot/blogs/iot-data-how-to-collect-process-and-analyze-them-032619/
- https://www.ibm.com/cloud/learn/machine-learning#toc-what-is-ma-qhM6PX35
- https://www.sas.com/en_us/insights/analytics/machine-learning.html

- https://machinelearningmastery.com/transfer-learning-for-deep-learning/
- https://code.google.com/archive/p/word2vec/
- https://nlp.stanford.edu/projects/glove/
- https://en.wikipedia.org/wiki/Internet_of_things
- https://en.wikipedia.org/wiki/Big_data
- https://artofproblemsolving.com/wiki/index.php/Without_loss_of_generality#Advanced_Level

Algorithm

Libraries

In order to achieve all the things mentioned above we used Python programming language.Python is a scripting language with huge capabilities in Machine Learning ,in DataAnalysis and visualization.

Python programming language can offer concise and readable code. The developers using python's simplicity for Machine Learnings (ML) and Artificial Intelligence (AI) can write reliable systems with libraries for complex algorithms and versatile workflows.

Python is quite easy to learn because it is very flexible in variables or compiling and running so many developers choose Python for their implementation. Python is a general-purpose language with this term we mean that it can do a set of complex machine learning tasks and enable you to build prototypes quickly. This allows the developer to test the product easier for machine learning purposes.

Also we used the numpy python library to have the control of reshaping the data. NumPy is a Python library which is frequently used for working with linear algebra arrays. The object array in Numpy is called ndarray and this python library is providing many supporting functions in order to make multiple complex operations with arrays which make it very easy.Numpy and Arrays are frequently used in data science, where speed and resources are very important.

Furthermore in order to implement PCA, LLE, MDS we used sklearn library. Scikit-learn library is the most useful library in Python for machine learning (ML) because it contains many tools which increase efficiency in machine learning,regression,clustering and dimensionality reduction. Scikit-learn has many features where they will be analyzed below in order to have a better understanding on the spread of this library:

- **Supervised learning algorithms:** It is very frequent that many known supervised machine learning algorithms will be a small piece of scikit-learn. A good example of these algorithms is Generalized linear models (e.g Linear Regression), Support Vector Machines (SVM), Decision Trees to Bayesian methods. Scikit-Learn has high usage due to its spread in machine learning (ML).

- **Cross-validation:** Scikit-Learn includes a high variety of methods to check supervised models accuracy.

- **Unsupervised learning algorithms:** Clustering, factor analysis, principal component analysis to unsupervised neural networks are some of the algorithms Scikit-Learn has for unsupervised learning.

- **Datasets:** Many academic datasets for training are available in scikit-learn like IRIS dataset, Boston House prices dataset.

- **Feature extraction:** Features from images and text can finally be extracted from scikit-learn library.

In the end we used Pandas library to make some tables and the plots for the results of our experiments. Pandas library in python is very useful because it gives the opportunity to the developper to easily manipulate data and analyze them. More accurately Pandas is offering many data structures and some very useful functions for visualization tables and time series.

Main program

With all mentioned above we implemented an algorithm as shown below in Python:

```
import random ;
import numpy as np;
import os;
```

```
    from sklearn.decomposition import PCA;
    from sklearn.manifold import LocallyLinearEmbedding;
    from sklearn.manifold import MDS;
    import pandas as pd;
    import function;

    dimensions = 5000;
    nodes = 10;
    nodesListValue = [];
    components=3;

    target_pca= ["Pca", "Pca", "Pca", "Pca", "Pca", "Pca", "Pca", "Pca", "Pca",
"Pca"]
    target_lle= ["Lle", "Lle", "Lle", "Lle", "Lle", "Lle", "Lle", "Lle", "Lle", "Lle"]
    target_mds= ["Mds", "Mds", "Mds", "Mds", "Mds", "Mds", "Mds", "Mds",
"Mds", "Mds"]
    target_common=['common','common','common','common','common','common'
,'common','common','common','common']

    url= '/home/andreas/Documents/thesis/'

    runs=50
    path = url+'run'+str(runs)+'/';
    os.mkdir(path)
    for n in range(0, nodes):
        values = [];
        for i in range(0,dimensions):
          values.append(random.random());
        nodesListValue.append(values);
    npList=None;
    npList =np.asarray(nodesListValue,dtype = np.float32); # cast list to np
array in order to reshape the data

    print("Input Data:\n",npList);
    pca =None;
    lle = None;
    mds = None;

    pca = PCA(n_components =components);
    lle = LocallyLinearEmbedding(n_components=components)
    mds = MDS(n_components=components)
```

```
    function.project_table_3_Components(pca.fit_transform(npList),'pca',path)
    function.project_table_3_Components(lle.fit_transform(npList),'lle',path)
    function.project_table_3_Components(mds.fit_transform(npList),'mds',path)
    precision =1;

    pca_result=None;
    lle_result= None;
    mds_result = None;
    pca_result = function.round(pca.fit_transform(npList),precision);
    lle_result = function.round(lle.fit_transform(npList),precision);
    mds_result = function.round(mds.fit_transform(npList),precision);

    pca_result = function.find_top_values(pca_result);
    lle_result = function.find_top_values(lle_result);
    mds_result = function.find_top_values(mds_result);

    # print(pca_result);
    # print(lle_result);
    # print(mds_result);

    print("PCA TABLE");

    # print('explained variance ratio (first two components): %s'
    #       % str(pca.explained_variance_ratio_))

    pca_result_panda=None;
    final_target_pca =None;
    pca_result_panda =function.project_table(pca_result,'PCA',path)
    final_target_pca= pd.DataFrame(data=target_pca, columns=['target']);
    # print(pca_result_panda);
    pd_concat_pca=None;
    pd_concat_pca = pd.concat([pca_result_panda, final_target_pca[['target']]],
axis=1);
    # print(pd_concat_pca);
    function.project_2D(pd_concat_pca,'PCA',path);



    print("LLE TABLE");
    lle_result_panda = function.project_table(lle_result,'LLE',path);
    # print(lle_result_panda);
```

```
    final_target_lle= pd.DataFrame(data=target_lle, columns=['target']);
    pd_concat_lle = pd.concat([lle_result_panda, final_target_lle[['target']]],
axis=1);
    # print(pd_concat_lle);
    function.project_2D(pd_concat_lle,'LLE',path);


    print("MDS TABLE");
    mds_result_panda = function.project_table(mds_result,'MDS',path)
    # print(mds_result_panda);

    final_target_mds= pd.DataFrame(data=target_mds, columns=['target']);
    pd_concat_mds= pd.concat([mds_result_panda, final_target_mds[['target']]],
axis=1);
    # print(pd_concat_mds);
    function.project_2D(pd_concat_mds,"MDS",path);

    #Project everything PCA LLE MDS
    pdList = [pd_concat_pca,pd_concat_mds,pd_concat_lle]  # List of your
dataframes
    pca_lle_mds_dataframe = pd.concat(pdList)
    # print(pca_lle_mds_dataframe)
    function.project_2D(pca_lle_mds_dataframe,"PCA,LLE,MDS",path);

    common_results = function.find_match(pca_result, lle_result, mds_result);
    print("Common results between PCA LLE and MDS")
    # print(common_results);

    final_results=[];
    for dimension in common_results:
        tmp=[]
        if not dimension:
          tmp.append(0);
        else:
          tmp.append(max(dimension))
        final_results.append(tmp)

    # print (final_results)
    final_results=np.asarray(final_results,dtype = np.float32);
```

```
    common_result_panda=function.project_common_results_table(final_result
s,path);
    # print(common_result_panda)

    function.project_scatter(common_result_panda,path)
    final_target_common= pd.DataFrame(data=target_common,
columns=['target']);
    pd_concat_common= pd.concat([common_result_panda,
final_target_common[['target']]], axis=1);
    # print(pd_concat_common);
    # function.project_2D(pd_concat_common,"Common");
```

Support functions

We also have another file with some support functions. These functions are used to round or aggregate data and make the tables and the plots.

```
import numpy as np;
import pandas as pd;
import matplotlib.pyplot as plt;


def round(list,precision):
        return np.round(list,precision);


def find_top_values(list):
        new_array =[]
        for sub_list in list:
        sorted=np.argsort(sub_list);
        sorted_array=sub_list[sorted];
        rslt = sorted_array[-2:];
        new_array.append(rslt.tolist())
        return np.asarray(new_array,dtype =np.float32);
```

```
def find_match(pca, lle, mds):
        print("\n\n\n")
        matching_list=[];
        for i in range(0,len(pca)):
        first = np.intersect1d(pca[i],lle[i]);
        second = np.intersect1d(pca[i],mds[i]);
        third  = np.intersect1d(lle[i],mds[i]);
        result = np.concatenate((first,second,third));
        matching_list.append(result.tolist());
        return matching_list;


def project_table(data,name,path):
        df = pd.DataFrame(data=data, columns=['component 1',
'component 2']);
        figure, axes = render_table(df, header_columns=0, col_width=4.0)
         #save

        fig.savefig(path+name+'_table.png')
        return df


def project_table_3_Components(data,name,path):
        df = pd.DataFrame(data=data, columns=['component 1',
'component 2', 'component 3']);
        figure, axes = render_table(df, header_columns=0, col_width=4.0)
         #save
        figure.savefig(path+name+'_table_3.png')
        return df
def project_common_results_table(data,path):
        df = pd.DataFrame(data=data, columns=['Common']);
        figure, axes = render_table(df, header_columns=0, col_width=4.0)
         #save

        fig.savefig(path+'common_table.png')
        return df
```

```
def concat_tables(data,target):
        pd_concat=pd.concat([data, target], axis=1);
        print(pd_concat)
        return pd_concat

def project_2D(data,name,path):

        figure = plt.figure(figsize=(10,8))
        ax = figure.add_subplot(2, 1, 1)
        ax.set_title('2 component '+name, fontsize=20)
        targets = ['Pca', 'Lle', 'Mds','common']
        colors = ['r', 'g', 'b']
        for target, color in zip(targets, colors):
        indicesToKeep = data['target'] == target
        ax.scatter(data.loc[ indicesToKeep,'component 1']
                , data.loc[ indicesToKeep,'component 2']
                , c=color
                , s=50,
                label=target)
        ax.legend(loc='best', shadow=False, scatterpoints=1)
        ax.grid()
        plt.savefig(path+name+'.png');

def project_scatter(data,path):
        plt.clf()
        figure = plt.figure(figsize=(8, 8))
        axes = fig.add_subplot(1, 1, 1)
        plt.plot(data,color='g');
        plt.savefig(path+'common_results.png')
```

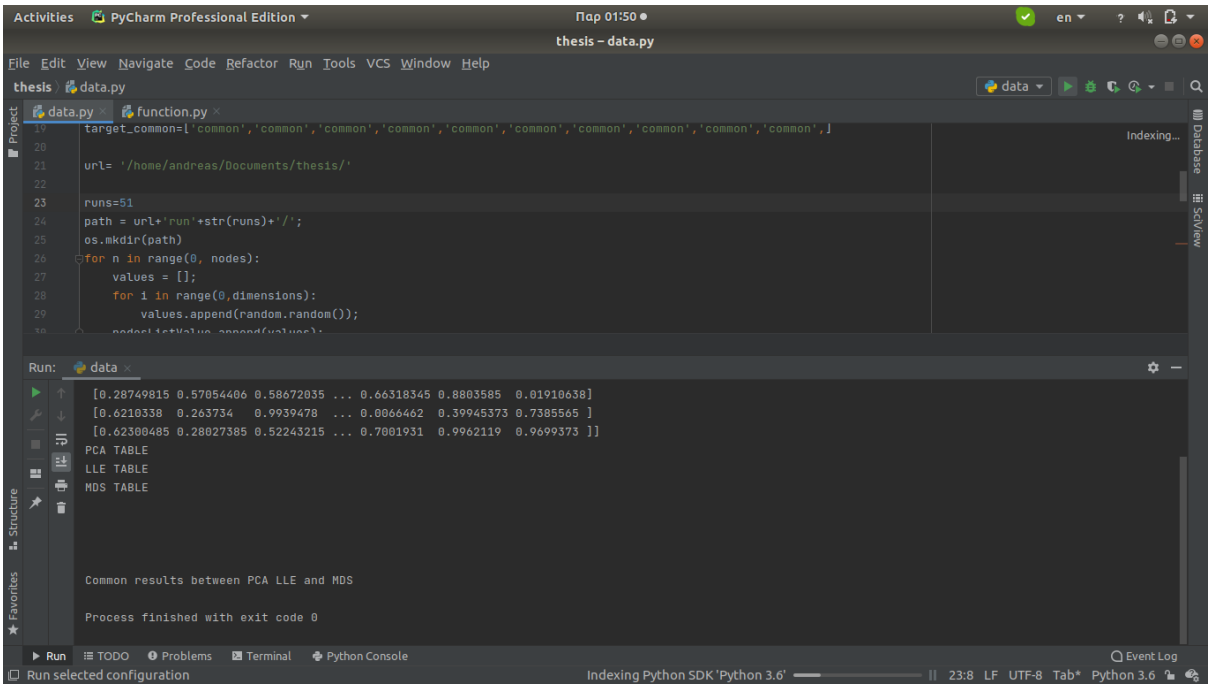
Running Cycle

We tested our algorithm in Pycharm we chose PyCharm as a tool for this master Thesis because we wanted an efficient environment and this is an integrated environment for Python. It is developed by the Czech

company JetBrains and offers a variety of useful IDE for our application development.

In addition we may run the program in a terminal with the command python <filename>. Both running methods can be applied when every library has been firstly installed in our operating system.

Finally after the whole algorithm has been run it produces many tables and plots that either are used to display data in this report or are used to extract some results. Below I will cite some examples of running.



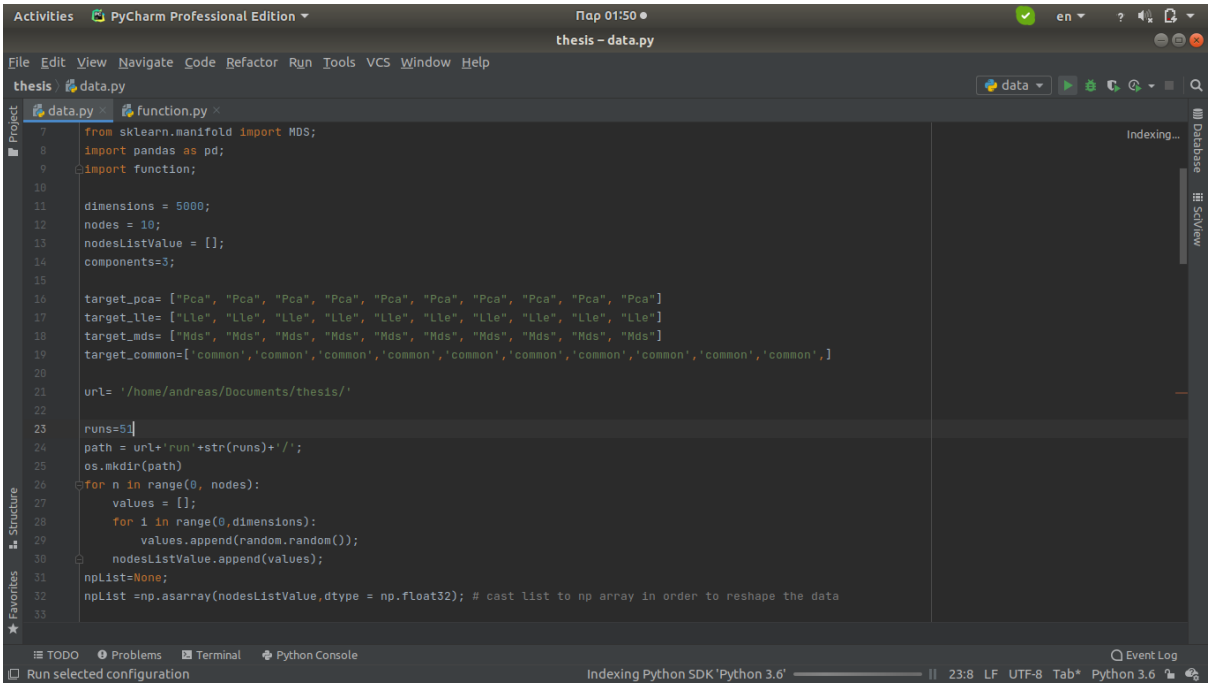


Figure 6.3.1: Running Cycle