Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

MASTER THESIS

TITLE: Study and application of machine learning techniques to the deployment of services on 5G optical networks.

MASTER DEGREE: Master's degree in Applied Telecommunications and Engineering Management (MASTEAM)

AUTHOR: Héctor Delás Castellá

ADVISOR: SPADARO, Salvatore

DATE: October,22nd2019

Abstract

The vision of the future 5G corresponds to a highly heterogeneous network at different levels; the increment in the number of services requests for the 5G networks imposes several technical challenges.

In the 5G context, in the recent years, several machine learning-based approaches have been demonstrated as useful tools for making easier the networks' management, by considering that different unexpected events could make that the services cannot be satisfied at the moment they are requested. Such approaches are usually referred as cognitive network management.

There are too many parameters inside the 5G network affecting each layer of the network; the virtualization and abstraction of the services is a crucial part for a satisfactory service deployment, being the monitoring and control of the different planes the two keys inside the cognitive network management.

In this project it has been addressed the implementation of a simulated data collector as well as the study of several machine learning-based approaches. This way, possible future performance can be predicted, giving to the system the ability to change the initial parameters and to adapt the network to future demands.

Introduction

Introduction and scope.

The upcoming generation of mobile communication systems will have around 8 million of subscriptions according to the forecasts. As a consequence, it has to be considered the overwhelming increase of smart devices and the irruption of the Internet of Things (IoT) whose main aim is to connect everything.

From predecessors, 5G systems implies a big jump, since key performance indicators (KPIs) have to be taken into account, making a basic requirement the self-organization of the whole network, including end-to-end network behaviour intelligence to ensure a profitable business model. Here is where the self-organizing network (SON) is introduced as the engine which will enable the exploitation of Artificial Intelligence (AI) mechanisms for the efficient self-management.

In legacy systems, the network observes the environment, makes a diagnosis of the problem and executes the compensating action which involves to spend a valuable operation time which is not compatible anymore with the 5G targeted latency requirements. It is here where the big data must introduce the capabilities which make that the network can adapt its structure making the future SONs distinct from legacy cellular systems.

In this the following objectives have been settled down and finally they have been finally accomplished:

- Review the 5G architecture and the study of machine learning techniques as valuable tools for the autonomy management of 5G networks.
- Benchmark different machine learning techniques to infer the most suitable solution.
- Analyse the results of the ML techniques and study their implementation.

CHAPTER 1. Overview of 5G-based networks systems

1.1 Architecture and network slicing

Nowadays due to the fact that a wide range of new services are growing up and the industry automation is becoming a necessity, the significant increment of the traffic makes that networks need to adapt in order to satisfy these new requirements.

Then, it is planned a network slicing for satisfy these aims, including two fundamental enablers, that is, software-defined and virtualization in order to managed and orchestrate network in an efficient way. However, the network will have other needs like the scalability which will take a relevant position determining how the resources must be distribute in order to satisfy future upcoming demands.

The customization of the network will introduce new elements like the virtual network function where the network abstraction is particularly relevant to accomplish the network slicing concept.

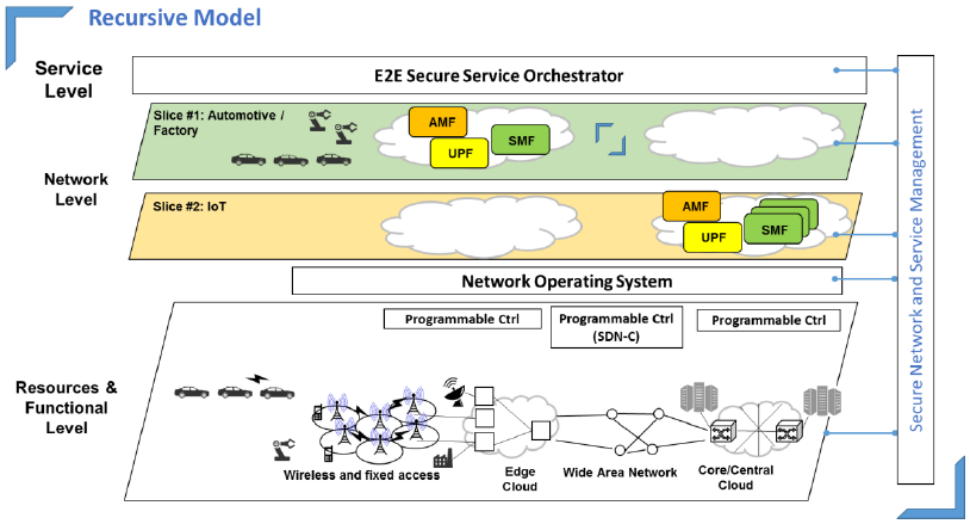

Fig. 1.1 E2E 5G vision per layers [2]

As we see in the figure 1.1, at the service level, the structure is repeated like in network level, it can be just a specific part of the network or only a resource which needs to be repeated.

Regarding to the network slicing, as it is shown in Fig 1.1, it will provide to the operators the ability for fitting networks depending on the wide range of uses cases (Personalized), by means of virtualization and software defined networking. Summing up, the network slicing pretends to match the cloud

infrastructure and network functions to meet the specific requirements of different use cases.

In order to compute all this orchestration for the wide range of different use cases and covered the totality of network segments it will be necessary to predict and to analyse huge amounts of information, achieving not only the network management but also the isolation of resources and the disjunctive rate of each slice.

The adoption of network sharing and the introduction of multi-tenancy will allow to increase the network flexibility relying on virtualization mechanisms and software-based capabilities; in this regard, the network function virtualization (NFV) takes a relevant sense inside of the network management architecture. In particular, it will be responsible of identifying requirements, interfaces and procedures which could be re-used or even extended.

The network slicing will allow dedicated networks where each one will be assigned to serve different types of users based on client demand and assuring the total isolation and independent scaling, making the 5G network slice broker the key element which will allocate a collection of shared network resources and VNFs among several concrete slices that fulfil the requirements of services.

The other new characteristic that will be part of the 5G network will be the mobile edge computing (MEC), making possible that the service could be envisioned, allowing low latencies, offering flexible services and if it is combined with VNFs will enable a joint optimization of these services.

We can differentiate between different layers as we can see in the diagram of the 5G layers (Fig 1.2) where the network is clearly divided into 4 different layers, where they are splitted among independent elements.

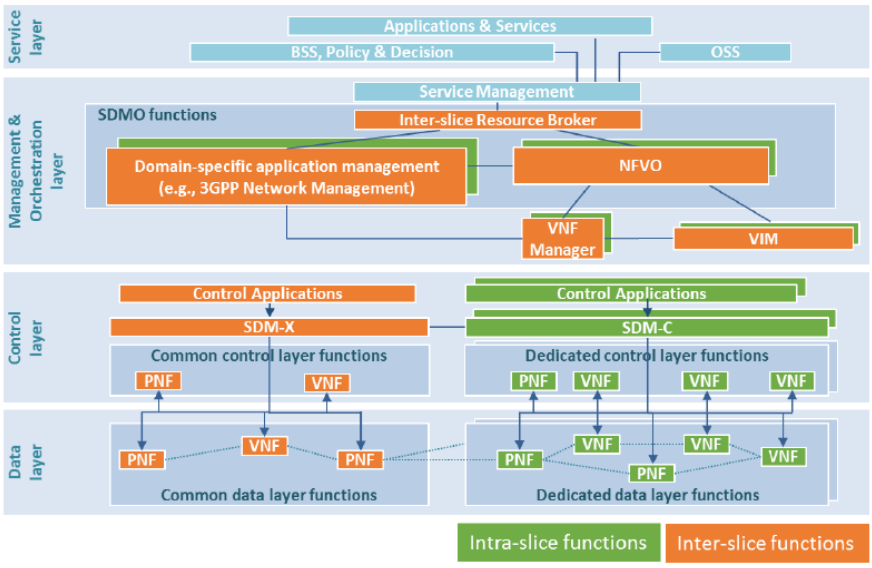

Fig. 1.2 Diagram of 5G layers [2]

Basically the service layer is the layer which is closer to the real users of the network and provide the applications and services. The management orchestration will transform the consumer-facing service descriptions into resources and vice versa. The control layer will translate decisions of control, which will be processed in the data layer in order to decide what it will be the best choice, into commands.

Regarding to the network slicing Life-cycle part we must separate the management plane and the instance level (Fig 1.2). Every instantiation will correspond to a segment of the network and all of them will have an embedded common softwarization which will organize the resources; this way every slice will be independent from the rest.

Summing up, network slicing basically “enables the operator to create logically partitioned networks at a given time customized to provide optimized services for different market scenarios” [2]. Therefore, the 5G can be reduced in three simpler concepts: massive machine to machine communications connecting billions of devices at the same time; ultra-reliable low latency communications in order to control in real-time industry devices, transport networks etc.; And finally enhanced mobile broadband: providing significantly faster data speeds and greater capacity keeping the world connected.

1.1.1 Understanding the radio access network 5G.

The 5G technology introduce a new combination of challenges, emphasizing the integration of the new radio access technology with the previous ones. These problems are solved by means of centralizing hardware, reducing power consumption, and with flexibility to adopt intermediate solutions.

One of the keys is SDMC, which allows to control and program the full network. The different slices in the network enable sharing network functions between them and reuse resources and slices. This control function is run on the top of a controller for slice-dedicated and another coordinating controller in order to shared functions. Furthermore, this programmable control and coordination function provides a new level of abstraction in the lowest layer resources in RAN simplifying in this way its control.

Other problem which appears because of the heterogeneous mobile networks is optimal resource utilisation, for solving this issue the software defined network (SDN) in radio access network (RAN) could give us the benefit of global optimisation and programmability. But creating this software defined for the radio access resources when it must be determined the optimal solution makes that delay becomes in a truly problem between the centralized control entity and the individual radio elements. The benefit of global optimisation and programmability comes at the expense of scalability and latency.

In order to face up these problems in several papers has been proposed the next solutions:

1. Centralised controller for network-wide control and coordination.

2. Local controller for network functions requiring real-time operation.

The first solution deals with the idea of a centralised controller making decisions which affects network states while local controller handles just latency decisions in low layers without interfering in other ones; at the end it will improve the levels of scalability and latency. The second solution will focus on different sub-controllers in order to deal with the requirements in real time, adding a degree of complexity to manage the RAN.

One of the main characteristics of 5G network is the integration of different technologies in the radio access network by means of resources abstraction which will be able to manage the complexity and simplifies the implementation of all these functions and physical layer resources. Moreover, 5G wants to cope huge diversity of application service requirements and deployment scenarios whose features could be totally different.

The flexible implementation of 5G RAN will be impacted by application performance requirements and capabilities of computing platform. Therefore, it is crucial to understand this part of the network because, at the end, the abstraction will play an essential role. The machine learning techniques will be applied in a huge range in this part of the network. For this reason this section aims to show how the virtualization would work.

Now it is more interesting that 5G and enhanced LTE will be integrated on the RAN level in a way of tight interworking, this means that it must be dual connectivity, the user equipment (UE) has to be able to connect to eLTE and the CP and UP simultaneously (Dual connectivity, DC).

Table 1.1. Advantages and disadvantages of dual connectivity

Advantages of DC	Disadvantages of DC
DC can increase UE throughput	Resource efficient than connect to the best cell
Connection more reliable via CP diversity	

In the table 1.1 is presented which additions brings the dual connectivity to the connection between the client part and the core network and how the implementation of new radio access technologies could bring some points which help to the performance of the network. However, and despite the advantages that dual connectivity brings, it will implies other problems too. Right now the dual connectivity is considering as the best choice for this issue but still there are some points which has not been solved yet.

1.1.2 Understanding the core network in 5G

The core network regards the part of the 5G network where is the mobile exchange and data network that manages all of the mobile voice, data and internet connections. For 5G, the ‘core network’ is being redesigned to better integrate with the internet and cloud based services and also includes distributed servers across the network improving response times (reducing latency).

This part of the 5G network is where the project will work, the core network is where the new advanced features of 5G will be managed, like the function virtualization and the network slicing for different applications and services. In the figure 1.4 [18] we can observe the segment which corresponds with the core network where the virtualization, abstraction and management functions work. In the illustration we can see the introduction of the software defined network (SDN) and local cloud servers; where the services applications will run providing, thanks to the abstraction and virtualization, faster content and low latency applications.

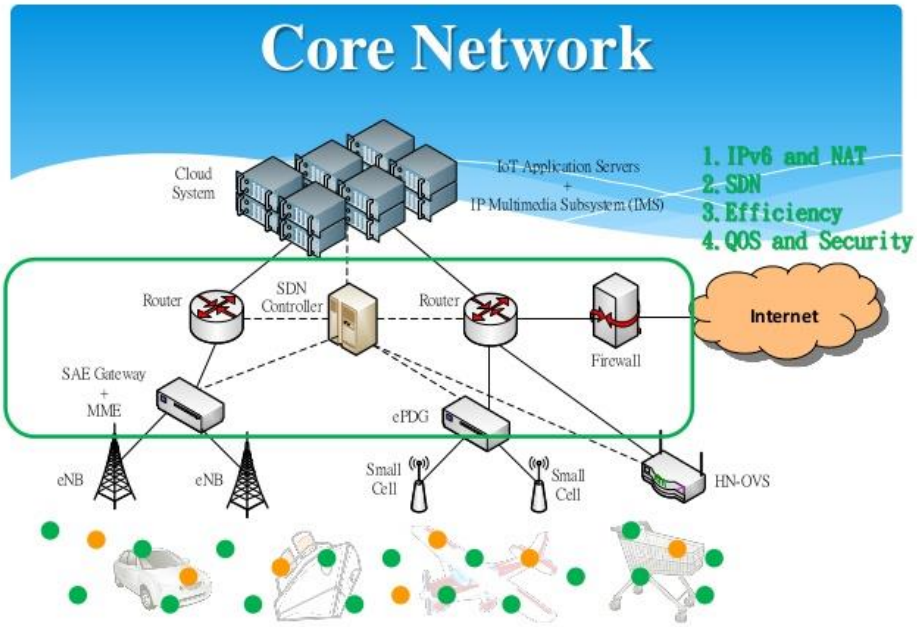

Fig 1.3. 5G network architecture [18]

In this section we are not going too deeper in the core network architecture but we will introduce some concepts that they will be used later as well as divided the core architecture in the most relevant elements just for clarify how it works.

One of the most important elements inside the core network and an element that it will be recursive is the network Function Virtualization (NVF); consisting in the ability to instantiate network functions in real time at any desired location within the operator’s cloud platform. Network functions that used to run on dedicated hardware, for example a firewall and encryption at business premises, can now

operate on software on a virtual machine. NVF is crucial to enable the speed efficiency and agility to support new business applications and it is an important technology for a 5G ready core.

The software defined network (SDN) is the other concept which has appeared in previous sections, as an architecture controller or central software program, called a controller where it dictates the overall network behaviour. SDN allows to separate the data plane where the network devices become simple packet forwarding devices, while the “brain” or control logic is implemented in the controller will be the control plane.

This project has the aim of investigating this services virtualization due to the fact that it could suppose a real challenge to manage thousands of different metrics and data in order to provide a low latency services. There will be a huge amount of parameters, which a datacenter will have to face up. Consequently, the project will work on the idea of how could be managed and the introduction of the helpful machine learning techniques in order to predict possible future demands inside the core network.

1.2 Programmability & Softwarization

The dynamic programming refers to execute different codes in the execution environments of the network. It means program personalized application services in the network. This is strictly related to the management and the orchestration part.

In this part, the operator will decide how is going to carry out the request, placing the network function (NFs) to compose the slice and the assignation of the resources, referring to placement algorithms that can be designed by using Integer Linear Programming (ILP)-based approaches, multi-provider embedding, greedy heuristics with backtracking.

The operator wants to fit as much as possible the number of resources. From an end-to-end perspective it is considered a cooperative ecosystem among operators in order to complete the service. Network programmability is an essential part of the new 5G network in order to support the rapid deployment of new use cases combining cloud-based services with mobile network infrastructure, where the use cases in critical scenarios could benefit from QoS programmability. In fact, cellular network’s connectivity requirements, including latency, throughput, service lifetime and cost, could vary widely across different use cases where the last beneficiary is the consumer getting a unique and personalized experience (also defined as Quality of Experience, QoE).

The authors in [2] outline that: “The agents analyse on the information collected during the monitoring phase and (re)configure policies to enact a desired alteration in the network infrastructure.” Clearly it can be understood how the softwarization assumes the important role of reconfiguration in the network when it is necessary.

As it has been mentioned, 5G must support several use cases and business models; this brings the problem of having many different actors that have to be independently managed. This could be a critical part from security point of view. Now in 5G, new nodes can be added and removed, so it is a must to define model attack vectors in this dynamic environment and to be able to offer strong network protection and define security control points in order to avoid possible threats and attacks. In spite of the fact that this project has not the aim of considering the security inside the 5G layers, there are several points where the application of machine learning techniques could be an important part of the system security, making easier to capture and predict possible future threats for the network just visualising different metrics.

It is important to consider that three different domain types appear: 1) infrastructure(physical) domain; 2) tenant(management) domain used to virtualize and getting logical functions and, 3) compound domain, which constitute the relationship between other domain and ours. The latter one implies for example, mapping 3G/4G with our network into our security architecture in a simple way. Additionally, inside of this domain, we can highlight the slice domain, which bring trust issues between the actors that control a domain and other actors controlling at the same time operating slices in the same domain. But it is important to stand out the strict isolation between domains and slices belonging to different actors.

1.2.1 Softwarization of 5G Service Management and Orchestration

In order to execute the different services as efficient and fast as possible we need to monitor the performance of 5G networks, therefore we can assume this problem from different points of view but we should just understand how the softwarization will be in general terms implemented.

We have to assure that each service component could support desired level of performance, avoiding bottlenecks, this can be achieved by observing and analysing how the infrastructure is used and what it the relation with the service performance. In this regards, structures like TALE have been introduced. It is based on a full stack telemetry to collect data metrics related to throughput in order to identify possible issues and develop improvements/actions for solving them.

Now we are going to focus on the management plane that supports the slicing concept. The final aim of slice the 5G network is to provide an independent and customized service to different uses cases. As every customer will have different requirements, it is necessary that some entity manage the resources and deal with them in order to achieve an agreement and a balance resource distribution.

For the above reasons it has been proposed the NFV management and network orchestration (MANO) architecture which is stackable (Fig 1.5).

We can divide NFV MANO into three functional blocks:

- **NFV Orchestrator**: Responsible for adding new network services and virtual network function (VNF) packages; NS lifecycle management; global resource management; validation and authorization of network functions virtualization infrastructure (NFVI) resource requests.
- **VNF Manager**: supervise lifecycle management of VNF instances; coordinates and adapts configuration and event reporting between NFV infrastructure (NFVI) and Element/Network Management Systems.
- **Virtualized Infrastructure Manager (VIM):** Controls and manages the NFVI compute, storage, and network resources.

The management plane can be simplified including Self Organizing networking (SON) management mechanisms like configuration, self-optimization, self-healing and self-protection.

The slicing business model is basically based on multi-tenancy support which provides multiple simultaneously services. Four important aspects in network slicing have to be taken into account:

1. Infrastructure sharing: it means that in order to reuse and optimize the physical resources the network has to be able to use a scalable infrastructure which can provide dedicated hardware for several services without wasting any of the infrastructure resources. If two users can share the same physical resource it will be more efficient than use one for each of them.
2. Spectrum shared: the spectrum in a network is a significant issue due to the fact that it is very expensive and it is limited. So its usage must be optimised.
3. RAN sharing: depending on customer requests it will be necessary more eNB which at the same time will require more physical resources. [Section 1.2].
4. Network sharing: regarding to the software defined mobile orchestration enables the coordination between different inter-slice services; the main responsibilities are to map the slice templates representing requirements in contrast with available network resources, by deciding which network can be shared and the placement of the virtual and physical resources.

The implementation of multi-tenancy in the radio access network can rely on the current 3GPP and LTE implementation; by implementing effectively schedule resource blocks between different mobile virtual network operators (MVNO). This can be made by means of real time controller which helps to make policies shared in RAN resources in order to keep information in real time and for synchronising it.

The softwarization of the networks includes virtualization technologies like VNF which can be used to cover different kind of functions, also the concept of reusable functional block like the virtual network function (VNF) generalization

allowing an abstraction of platforms and hardware. Softwarization and slicing bring the implication that the infrastructure must be programmable. RAN coordination and programmability are central concepts in 5G that are aimed to improve service quality, resource usage, and management efficiency, while addressing the limitations of the current LTE and WLAN systems caused by distributed control among them.

Inside of softwarization it is essentially to include network graph abstractions which are created from distributed data sources by means of collecting information accessible directly from infrastructure entities. In conclusion, it is created a hierarchical architecture instantiating the implementation capabilities in order to control the operations locally in a centralised way via virtualized infrastructure manager (VIM).

The orchestration of the network regulates network resources and management decision. Some of the objectives that orchestration is focused on are: Provisioning, security, QoS, fault tolerance, energy efficiency, etc. The difficulty about orchestration is in essential the cognitive network management across heterogeneous networks with their own characteristics and requirements.

This project is focused on the study of how the machine learning techniques and several algorithms can contribute to provide to 5G networks this autonomous management. Therefore, the orchestration is based on taking management decisions, optimise and adapt the network over time trough in order to obtain self-configuration. The main problem for achieving this self-management is the heterogeneity of the networks because everyone has their particularities and specific requirements like edge networks or computing clouds.

In new mobile service models due to the fact that are new requirements in terms of data and functions for the network management based on 5G it will be necessary to take care of them like the virtualization function in the network or the multiple network services, the scalability referring to the capacity of the network for adapting to the demand, the security which we have talked in previous sections or the QoS. The next step is the encapsulation of networking function virtualization (NFV) applications standardized by the European Telecommunications Standards Institute (ETSI) as management and network orchestration virtual network function manager (MANO VNFM) functionalities towards a multi-tenant management of VNF lifecycle. The network functions virtualization (NFV) defines standards for compute, storage, and networking resources that could be used to build virtualized network functions. The virtual network functions manager (VNFM) is a key component of the NFV management and organization (MANO) architectural framework.

The VNFM works with the other two NFV-MANO functional blocks, the virtualized infrastructure manager (VIM) and the NFV orchestrator (NFVO), to help standardize the functions of virtual networking and increase the interoperability of software-defined networking elements.

Finally, there are other elements that the network need to manage, due to the fact that nowadays there are several operators that will have to share the same ecosystem; then for achieving a multi-operator interaction, additional work is needed for the inter-operator orchestration. Therefore, in order to achieve this inter-operator coexistence, the components which are responsible for this capabilities are divided in three groups; the first one is dedicated to the resource acquisition; the second one to service request deployment and the third is in charge of service assurance and service level agreement (SLA) management.

1.3 End to End virtualization

In order to satisfy the needs of providing services with different requirements, offering at the same time a huge amount of different network functions is a must. If the 5G networks want to satisfy these demands it will be necessary to virtualize some infrastructures and functions, as well as making them scalable which means that we can add as resources as the demand requires them.

For this reason, it has appeared the network function virtualization (NFV) explained previously, whose function is providing a solution to reduce the installation deployment and the infrastructure cost as well as making a flexible function placement and service customization as we can check in Fig 1.5.

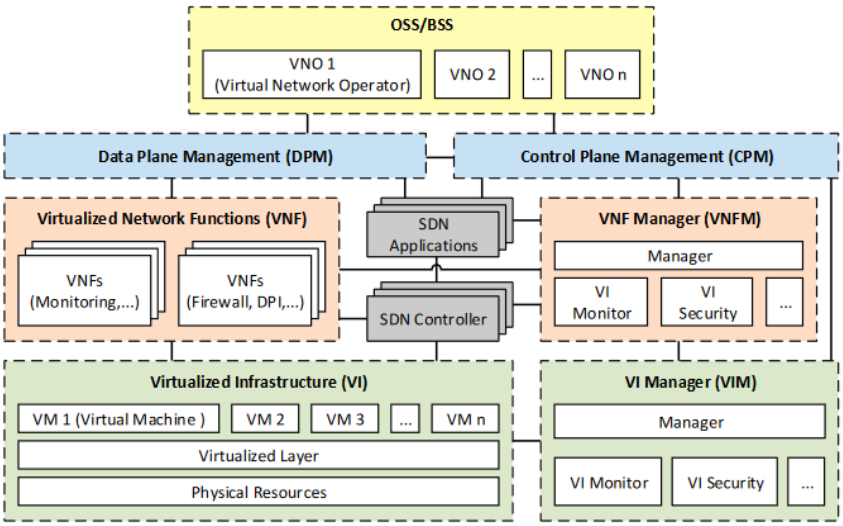

Fig 1.4 Data and control plane virtualized structure manager [1]

Going through the structure of the virtualization elements, one of the main parameters which play a fundamental role is the NFV nodes. They have a general purpose as programmable servers, each NFV node is virtualized as a virtual CPU cores. In this regard it will be possible to apply different algorithms in order to optimize and predict the future usage requirements. Then finally the virtualization will work in a quite similar way as virtual machines by means of programmable

virtual network functions (VNF Fig. 1.5) on VMs of different NFV nodes at different network locations in order to achieve high and optimize resource utilization.

Finally, the backbone core network will consist in different network switches and NFV nodes connected, the NFV nodes will host and operate VNFs. Consequently, a set of VNFs and the virtual links making the proper interconnection will constitute a logic VNF chain, therefore these VNF chains will be managed by a virtualization controller (VNF manager Fig 1.5) which means that will require a E2E service provisioning.

The 5GPPP multi-domain architecture as we have explained in section 1.1 is composed by a layered model which compose the infrastructure layer, multi-domain orchestration and application layer. In the infrastructure layer is where cellular network operators will add their resources therefore where E2E multi-domain orchestrator will be placed. The E2E service orchestrator uses the resource slices in order to integrate services slices which finally will finish into apps.

There are other requirements that the multi-domain architecture will have to support in 5G networks, such as the division between the control and user plane. As we can observe in figure 1.5 between the VNF and VNF manager coexists the software defined networking (SDN) which allows that control and data planes of network devices being decouple, enabling fast innovation as well as global network programmability and independent evolution of control and data planes.

Summing up, NFV enables that the resources of an infrastructure provide can be virtualized and shared among tenants, with one or more tenants requesting services can reach the efficient utilization of its resources by dynamically slicing them. The Network Virtualization function (NFV) proposes to move the processing of dedicated hardware middleware packages to the hardware running on the base servers; this way NFV brings the possibility of outsourcing enterprise Network Function (NFs) processing to the cloud. When an enterprise outsources its NFs to a Cloud Service Provider (CSP), the CSP is responsible for deciding: where initial Virtual NFs (VNFs) should be instantiated and what, when and where additional VNFs should be instantiated to satisfy the traffic changes (scalability) with minimal impact on network performances.

It is remarkable than the VNFs can be re-scaled vertically in order to offer an infrastructure resources based on client demands, consequently will provide better management, flexibility, lower cost, energy efficiency and better resource location and utilization. Most of the proposed heuristics are based on greedy algorithms using simple rules, such as First Fit Decreasing (FFD) and Best Fit Decreasing(BFD). In this project we are going to explored alternatives algorithms for the virtualization optimization and proposed a known structure which could be implemented in this kind of network regarding to 5G as well as explore different heuristic algorithms which scales to large datacenters without a significant decrease in performance.

1.3.1 Abstraction in Software Defined Networking (SDN) for DataCenters

The abstraction is an important part of the new generation of 5G networks, where those elements that in previous generations were physically working in the network; now are became into virtual machines or virtual processes that finally are executed in a datacenter, where several hosts will allocate the resources and it can be migrated depending on the needs.

Now, in this section we are going to analyse the part of the 5G core network dedicated to virtualize all those services which are used for achieving low latency and satisfy the demand of different network services and how it can be managed, introducing in this way the objective of this project which is the investigation of machine learning algorithms in the 5G network and why they are necessary.

A total effective abstraction of resources allocation requires a high precision for measuring which services are going to be attended first, depending on the priority and the available resources in each moment.

The SDN allows to organize the resources in the 5G network; summarizing its functionality, it can be said that computing the shortest paths are functions of the control plane and data plane functions handle packets and routing them from input port to output port, therefore control planes consist of logic that controls the packet forwarding behaviour of the routers and switches. It contains configuration procedure for middle boxes too, such as firewalls, and load balancers. The data plane is responsible for forwarding packets in the network, so it contains routing tables and hardware pertaining to the network.

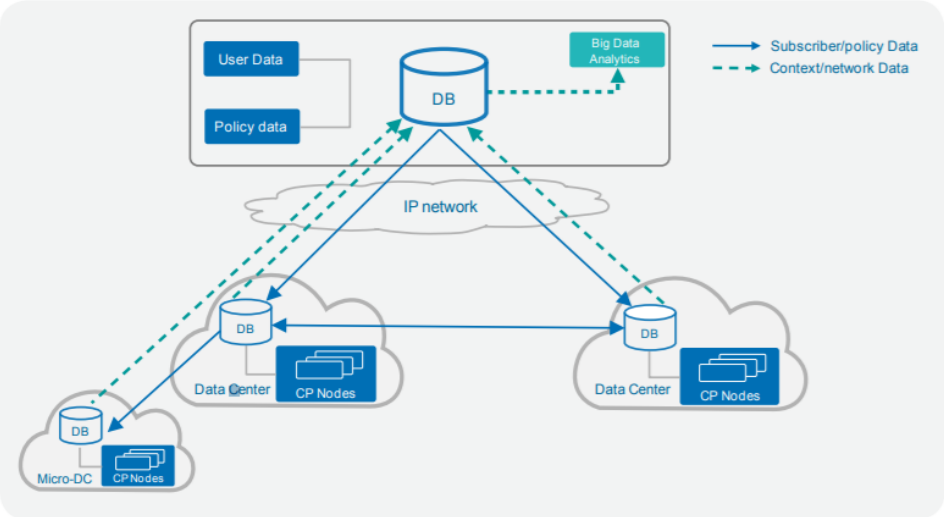

Fig 1.5 Core network abstraction **[16]**

We can say that the abstraction comes from becoming the legacy network into intelligent network, thus means that we need to analyse every parameter in order to optimize the resources abstraction. Here it comes the necessity of introducing big data analytics as we see in the figure 1.6, where the resources abstraction

will be blind if there is not any analysis and the policies can be changed in order to adapt the network to possible future problems.

In the figure 1.6 the problematic of virtualization appears. Due to the fact that the policies must change depending on the demand or the overwhelmed (due to a poor prevision), here it appears a new concept inside the 5G network called self-organize network (SON). The network must be able to adapt itself to possible future inconvenient by means of big data and analytics approaches, without that it would risk to make totally useless the improvements introduced in the new generation.

1.3.2 Policies

The orchestration and management architecture allows mobile virtual networks operators MVNOs to solve network slice requests, allowing programmable policies. This section is dedicated to explain the last part of the virtualization in the network management presented previously in the figure 1.5. Firstly, it is important to highlight that this project will not carry out the implementation of this part of the network which includes policies and actuators. Due to the fact that everything will be sent to the part of actuators we need to understand it. Therefore, our results in a real environment will be sent to the policy manager where it would be designed a policies strategy in order to be prepared to face up the future demand.

The policies manager or policy engine will be constant along this project, largely due to the fact that we have to consider always this part, all the predictions will be reflected in the policies engine, where depending on the results the actuators should change some resources allocation in order to optimize the performance.

The policy engine will have to include old (previously defined) and the new ones, those defined to support again the committed service requirements.

Furthermore, there will be a module in charge of making the convenient recommendation in order to evaluate the different possibilities and calculate the success rate we; can call this module as policy recommender (Fig 1.4). It could be improved always so there should be a policy optimizer where the policies are updated and installed in order to finally distribute and apply them into the network, taking an advantage from the predictions made in the smart engine.

In conclusion the policies are an indispensable part of the 5G virtualization, just not because they are the laws which will help to orchestrate and manage the network abstraction but also the essential part for making the network totally programmable.

1.4 Cognitive Self-Organize Network Architectures

The use case which better fits in order to explain the propose of applying machine learning techniques in 5G is the cognitive network model. The key aspect about cognitive network is the use of these models, collecting previously data and metrics from the NFVI and the control plane, analysing them and by means of

policies we can obtain a high level of abstraction in the network which means a better performance and a softer deployment in different parts of the network layers, as we have explained in the section 1.3.1.

In order to automatize the network management there is other element that can applied strategies previously designed called 5G self-organise network (SON). As SONs can be also defined those networks which have an autonomous management vision extended to the end to end network. In literature and also in some instances of products available in the market, Machine Learning (ML) has been identified as the key tool to implement autonomous adaptability and take advantage of experience when making decisions. In this project we will investigate how 5G network management can benefit from the ML solutions. The concept of SON is an approach to the future network generation requirements where they will be totally autonomous.

The cloud computing plays a relevant role inside the cognitive networks, where a high capability of computation is necessary in order to distribute efficiently the resources, clean data, apply machine learning techniques and allocate them when it was necessary. For that reason, the idea behind this structure is that the analysis and the application of machine learning algorithms will be run in the cloud (datacenters).

The new 5G networks look forward to achieve this autonomic network management, but there are some crucial elements in order to give the dynamics looked. The first important element will be the machine learning which will suppose one of the key technologies used in Cognitive network management in order to predict and facilitate the adaptation of the network, starting from the data collected from the different elements of the network.

Also it would be necessary to build a representative knowledge base (KB) in order to process input data from different sources through learning classification, prediction and clustering models in order to bring all this data to the self-organizing network (SON).

One of the most important points in the cognitive network is the autonomic monitoring which will have to identify the important features and captures information just to send it to the last important element, the automatic planning and execution (ANM) which by means some policies will have to adapt the resources to the needs of the network following this steps:

1. Knowledge.
2. Strategy.
3. Purposefulness.
4. Degree of adaptation autonomy.
5. Stimuli.
6. Adaptation rate.
7. Temporal scope and spatial scope.
8. open/closed adaptation and security.

Following this structure proposed in the [8], this project would try to reproduce the behaviour and making emphasis in each step for giving the Cognitive management.

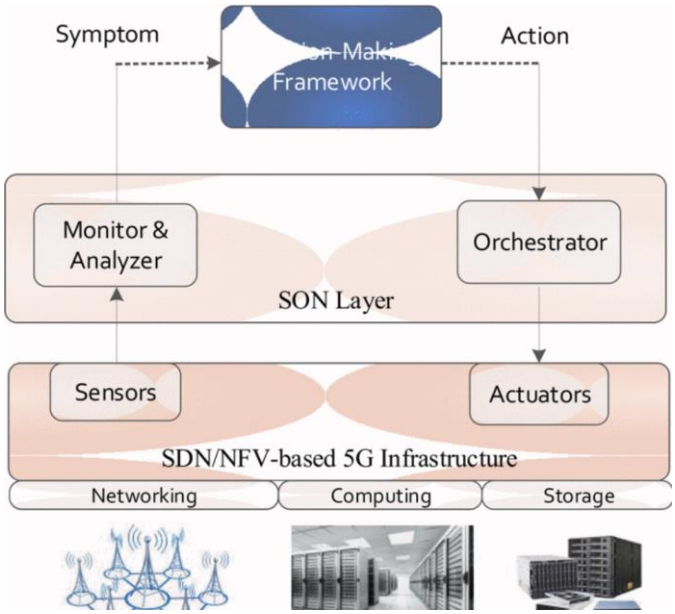

Fig 1.6 Cognitive self-organize network structure [8]

Following with the structure in the figure 1.7 we must highlight that there will be a common structure independently of the model or the use case; consequently, the characteristics finally are quite similar in a Cognitive network, therefore the diagram showed in the Fig 1.7 will be helpful for making the implementation of a real use case establishing the relation between the cognitive part and the self-organize network and how it would work.

The figure 1.7 show us the general structure of a cognitive network, where the SDN infrastructure is the layer where the metrics are collected and where it would apply the policies in the network; the policies are built in the SON layer where all the metrics are analyzed and sent to the smart engine where the machine learning techniques are applied, for example, to detect anomalies.

The new 5G networks look forward to achieve this autonomic network management, but there are some crucial elements in order to achieve the self-automation pursued. The softwarization and orchestration of the new 5G networks brings new opportunities that can be explored towards to the efficiency and optimization, therefore, from the project's point of view it is interesting analyse how the programmability of a network can evolve enough for making a network self-automatized.

Following the structure of the cognitive network, in this project has been proposed to take the Cognet structure in order to study the impact of machine learning techniques application, basing on the structure proposed by the authors in the paper [8], where they explain how and why the Cognet solution has been proposed achieving enough self-automation in the 5G network. Consequently,

our project would try to reproduce the behaviour and making emphasis in each step for analyse this Cognitive management.

Accordingly, the use of this structure together with ML techniques will not just enable cognitive network management but also it will bring new challenges like: automaticity, network function virtualization (NFV), software defined network (SDN), network slicing and finally knowledge based radio resource management. In order to manage these challenges and make a structure which provide the fundamental elements that will be able to provide self-management we have based on CogNet (Fig 1.8) structure which will be explain in the next section.

1.4.1 Cognitive networking

The general idea behind the Cognet structure is too keep the QoS and cost management in a proportional and optimized way, this means that the resources dedicated to the demand must be done according to the previous analysis and service provision in order to save resources for future demands.

In the figure 1.8 is clearly described how the structure works and which components constitute it. In the previous section we have explained the general idea about the Cognet project, but there are still missing some details about how it works and how it is going to introduce into our project.

The fact that 5G networks have not been implemented yet, makes impossible to implement a real data collector. For this reason, we decided to simulate the module and generate data inside of a data center, discarding the mobility and focus on the virtualized elements. The network function virtualization is included inside the Cognet structure (Fig 1.8/1.9), recording different kind of data, from dynamic resource allocation, going through performance degradation to demand prediction. To reduce complexity we only focus on data center resources within the Cognet network.

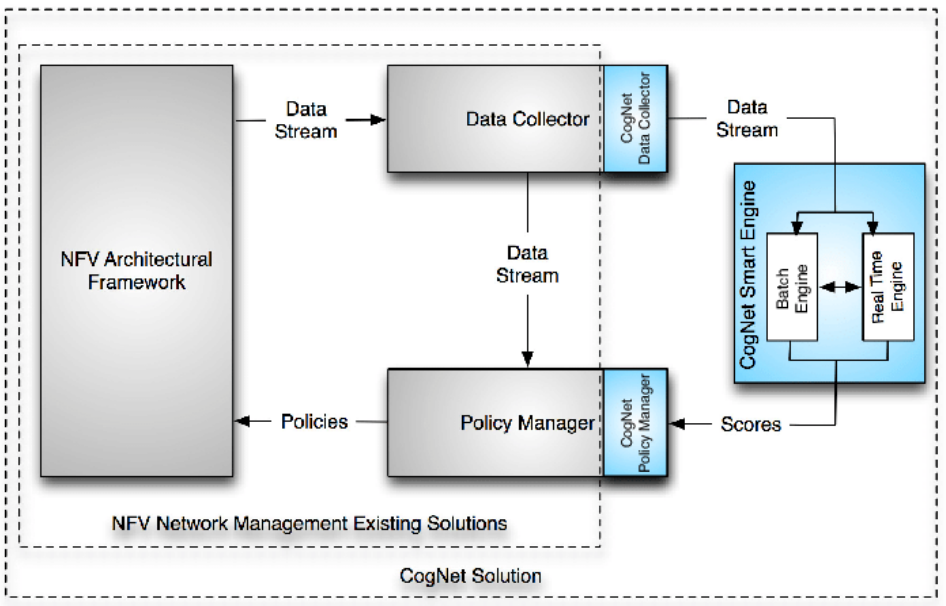

Fig 1.7 Cognet-based architecture

As we can observe in the figure 1.8 we can differentiate two main parts in the architecture; on the left one it is described the connection between the VM, control and orchestration level, from the network function virtualization framework which implies all the physical devices and functions that are running there; then all the key metric data will be sent to the data collector, where they are received and sent to the part of the Cognet functionality which manages all the data analysis.

The next step after applying all the algorithms, clean the data and obtain all the scores, it is when the policy manager will receive the results, where the orchestration level will decide which policies must be applied in order to improve, solve or prevent the architectural framework issues or optimizations.: .

As the data, they can be very uncorrelated. Therefore we should be cautious about the data since there are a lot of metrics which can be analysed. In this project, we will be focused on the enhancement of the virtualization services and the abstraction inside the hosts. The basic idea is to collect all the data which come from servers and physical machines, meaning that we receive a part of measurable data which is from machines like the RAM, CPU usage, total memory to allocate a resource or number of virtual machines required in a specific moment depending on a concrete demand. The data structure will be explained more widely in the next sections.

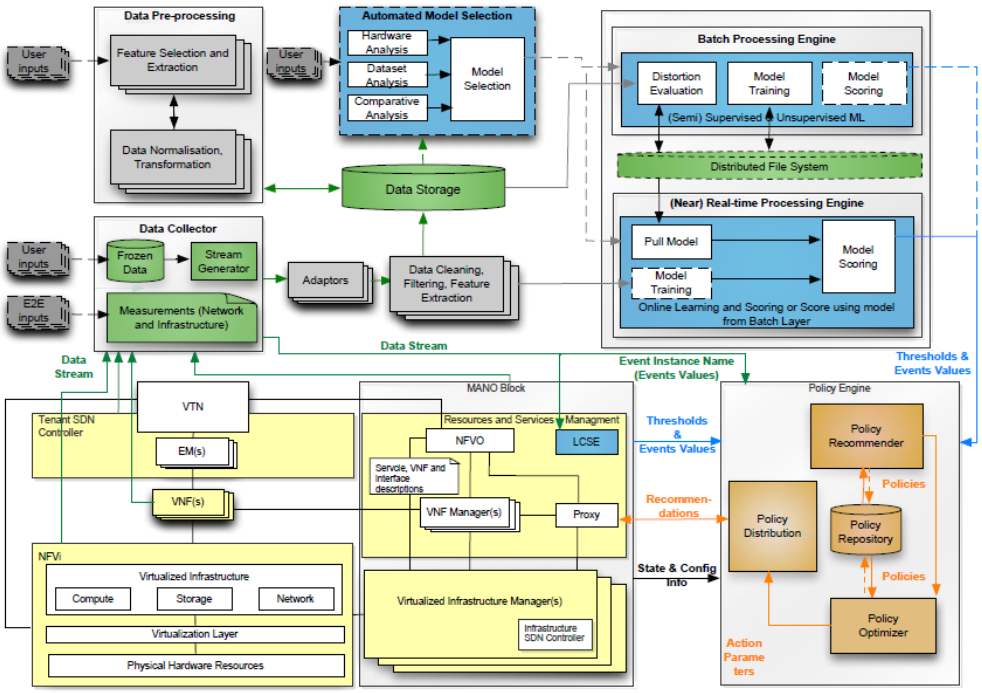

Fig 1.8 Cognet and SON structure within 5G [4]

According with the figure 1.9 presented in [4] we are going to analyse deeply the smart mechanism of CogNet, concretely the smart engine, where the first element that we will find, it is the data collector. Data collector will be implemented in our project as a simulator of virtual datacenter environments providing simulated

data. The data should be always pre-processed because it could be problematic to deal directly with it, consequently it needs to be clean and selected according with the network requirements.

The structure in our project will be limited by the data and the actuators, mainly due to the lack of real data. According with the figure 1.9 the data collected is sent to a data storage where it is sent and received from the pre-processing and automated model selection modules, therefore in the data storage it is stored after and before all the models have been chosen. Once, when the model has been selected the data will go through the batch processing engine, where it will be applied the model, usually in real time which has been selected, making more efficient the policies definition and the actuators performance.

As we have explained, without a data collector, the cognitive management will be useless. Once the data are collected from the network, the next step will be to clean the data and storage, where they can be consulted by the CSE components.

The pre-processing is the part of the network where the feature selection is applied in order to reduce the dimensionality and prepared for Batch processing engine; the feature selection is done by neural networks. It is important to remark that the feature selection is an important functionality to overall flexibility of the architecture.

When we have the variables which we are going to be used, we must choose the algorithm that will be applied for satisfy the requirements. However, sometimes an algorithm could not adapt properly so the batch processing engine evaluates current model and if the model stale or is not available it will generate a new one.

Finally, when the model has been applied can be saved optionally in the distributed file system because maybe the batch processing engine will generate models directly, being consumed by real-time processing engine which scores the data even in real time. As a conclusion, in this section we have explain the parts of our use case in order to understand how we are going to study the use of the machine learning techniques inside of the 5G environment.

Chapter 2. Machine learning

Machine Learning (ML) is a new approach guided by Artificial Intelligence that suggests that machines should be able to learn by themselves. Instead of providing them with repeatable knowledge, just giving some raw data from certain patterns can be extracted and learn from them as the human mind would.

We have mentioned in previous section why machine learning is not only a useful application for different areas within 5G, but also a requirement if we want to achieve the QoS levels or latency and bandwidth expected. In order to analyse the whole service given by some determined slice schemas, or due to the amount of data generated by several resources, it will be necessary the full optimization of the radio resource as well as the classification of the data traffic.

Therefore, network providers are looking for the minimum cost solutions to deploy total programmable network infrastructure. In order to maximize the profit of an infrastructure provider, the orchestrator will need to accept as many slices asked as possible; this obviously implies that, it will become in more incomes but they should match the variation of service requirements scaling the network. In fact, if a slice cannot be scaled up due to resource contention among the slices, the InP needs to pay a penalty corresponding to service degradation (see [4]).

For all these reasons, managing and predicting the variation of different services is a complicated task. The orchestrator can rely on ML techniques and Big Data analytics (BDA), as depicted in the figure 2.1.

The project is aimed to analyse how the machine learning techniques can improve the performance of the 5G networks and how they should be implemented or which infrastructures we should design in order to take benefit of apply them. The main architecture can be observed in the figure where the orchestrator should be equipped with different elements based on BDA; we can differentiate the typical processes which are followed in data analysis and data science where we need a module where all the data is picked up RD, other where this one is analysed by BDA.

The next step is to decide regarding to the slice requirements presented for the MSP and CSP (fig 2.1), about the acceptance of an income slice request depending on the resource availability. In order to have a continuous evaluation we need to monitoring the current state of the network resources by means of resource monitoring (RM).

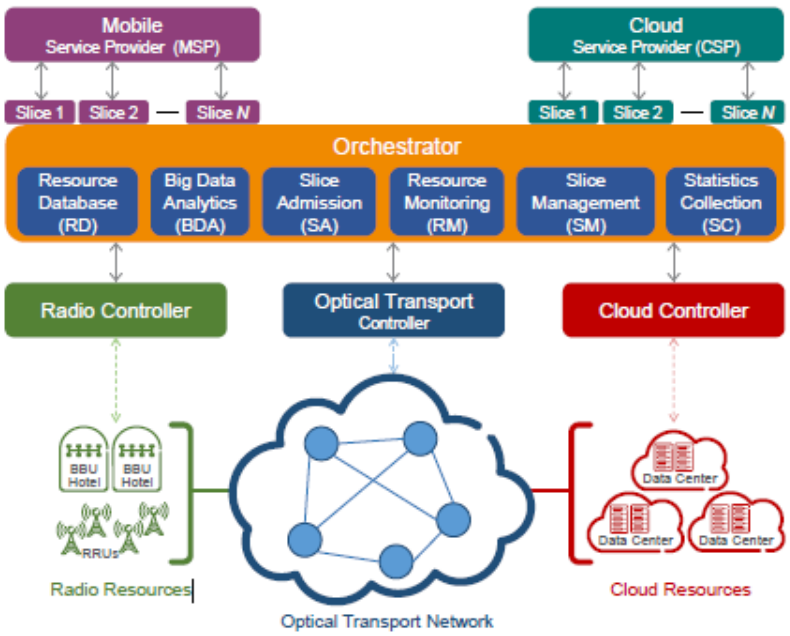

Fig 2.1 System architecture in resource Orchestration [4]

Consequently, the slice management whose performance have been explained in previous sections is where the policies will be implemented, deciding which slices need to be scaled up or down, sending the necessary requests to domains

controllers. Finally, the last module is about statistics collection where are extracted different statistics from slices.

In the figure 2.1 it is explained in a general view and according to the authors of [4] how the ML algorithms could be implemented for the optimization of slices, but it is not the only solution, because they could be applied equally to the cluster and forecast traffic behaviours of cells; as an example a clustering model will explore and identify areas of the network where the traffic is suffering issues or irregularities, then it should be compared with similar cells with same traffic patterns and extract them; pointing that when we have the clusters extracted we could apply ML algorithms.

2.1 Machine learning algorithms

In this section we are going to review what algorithms are available currently, when and where we could apply them and which one will fit better. According to [5] we can separate among several areas within 5G networks what kind of algorithm will be more suitable. In this section we will make a general overview of the main techniques known and the studies that have developed about applying ML algorithms in 5G, as well as explaining the main algorithms and classifying them.

The networking and distributed computing system is the key infrastructure to provide efficient computational resources for machine learning. Networking itself can also benefit from this promising technology. There a wide range of algorithms which can be implemented but our objective is not focused on develop new algorithms, but try to find out which are the most efficient so we will develop how the different techniques could be apply.

2.1.1 Supervised learning

In the supervised learning each training set is composed by labelled data, basically it consists in training a model by means of data which you can prove that it is reliable, then you can use the model in order to recognize the optimal solutions. It can be explained as a basic function in a linear system where you have input variables (X) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output, $Y = f(X)$

The goal is to approximate the mapping function that can allow to predict a target numeric value given several features, therefore, when you have new input data (X) you are able to predict the output variables (Y) for that data. Therefore, the final aim is to split the dataset into two different datasets, the training and the test; with the first we could train the algorithm in order to make it more reliable and with the second one we will check the results.

However, supervised learning often brings some problems:

- Incomplete data cannot be modelled (Missing data), but it should be valuable the source of unknown-ness (forgotten or lost) or the value does not care or is not applicable.
- Feature selection is not as effective as in the other learning techniques, due to the fact that many features will depend on a high grade from the others, which make unduly the influences into the accuracy of supervised ML classification models.
- The influence of the algorithm or the technique used will affect in a huge way the results, consequently it is crucial to choose the adequate technique.

We should differentiate between two different kind of techniques; regression and classification, regression tries to model the relationship between the parameters which are iteratively redefined using a measure error in the predictions. There exist several regression algorithms that we can classify according to the variables or the shape of the regression line, like linear, logistic, polynomial, stepwise, ridge, lasso or elastic net regression.

Table 2.1 Supervisee learning algorithms

AI Technique	**Learning model**	**5G-based Apps**
Supervised Learning	ML and statistical logistic regression techniques	Dynamic frequency and BW allocation in self-organize LTE dense small cell deployments
	Support vector Machines (SVM)	Path loss prediction model for urban environments
	Neural-Network-based approximation	Channel learning to infer unobservable CSI from an observable channel
	Supervised ML frameworks	Adjustment of the TDD UL-DL configuration to maximize the network performance based on the ongoing traffic conditions
	Artificial neural networks (ANN) and Multi-Layer Perceptron	Modelling and approximations of objective functions for link budget and propagation loss for next generation wireless networks

Regression or prediction which can be used when the variable is real or continuous, allows us to predict with a probability of failing. Like in the small cells which are being deployed in 5G networks to cope with the high demand of traffic,

they have an unpredictable and dynamic interference so that the SONs will be able to learn dynamically and adapt to the conditions in the current environment.

Classification is a technique for determining to which class belongs and to assign label to each class depending on the one or more independent variables. Regarding to the main algorithms that can be used, in the table 2.1 we can see summarized some supervised learning algorithms, where there are some relevant ones which will be applied in this project. However, we are not going to explore the theoretical background behind each one but, we will analyse how they act in every case with different data and how they can suit for 5G use cases.

2.1.2 Unsupervised learning

The main key in unsupervised learning is that the data is unlabelled, therefore the objective is training the data but without guidance, which can have supposed to be quite useful when the differences between the data groups are not very high. One of the main techniques within unsupervised learning is the clustering (table 2.2), which uses ML to situate data inside several groups of them considering the similarities among the features of the data.

The problem with clustering is that the no labelled data set cannot be compared, consequently at the end, the error which it could make, it is also undetermined in terms of comparison. Therefore, is difficult to evaluate if it is not in the specific case where it is properly to apply it.

The performance of the technique is described by a vector of N features which could be used to represent it in N dimensional spaces which could be useful in heterogeneous cellular networks.

Table 2.2 Unsupervised learning algorithm

AI technique	Learning model	5G – based Apps
Unsupervised learning	K-means clustering, Gaussian mixture model (GMM) and expectation maximization (EM)	Cooperative spectrum sensing. Relay node selection in vehicular networks
	Hierarchical clustering	Anomaly/Fault/Intrusion detection in mobile wireless networks
	Unsupervised soft-clustering ML framework	Latency reduction by clustering fog nodes to automatically decide which LPN is upgraded to HPN
	Affinity propagation clustering	Data driven resource management for Ultra-Dense Small cells

The most common techniques are: k-means clustering: which consists into k clusters where the data samples are place in groups.

Every cluster chooses Hierarchical clustering: Mini- Batch K-Means, Mean-Shift clustering, DBSCAN, Agglomerative Clustering, etc., can be used to associate the users to a certain base station in order to optimize the user equipment (UE) and base stations (BS) transmitting/receiving power according with the authors in [5].

2.2 State of the art

The objective of this section is to give some details of the main techniques and technologies used in machine learning and then, we review the literature on cognitive networking where we can study these techniques.

The aim behind this project is to show a high well-designed technology solution to be used in the paradigm of 5G network, where it will be essential in order to manage the millions of data generated at every second every day. Otherwise, it could not be even imagined the implementation of low latency services with a high QoS, at each moment that a client needs to use the service.

When we talked about the computing aspect inside the network we must take care about the possible variables which will enter inside our modules. Consequently, we should distinguish between two different algorithms, learning and optimization algorithms. We will focus on the first one, but the optimization algorithm will turn on the deterministic or stochastic techniques which corresponds more to the policies part.

2.2.1 Machine learning techniques in network management

We have presented in the previous section a summary of the most common ML techniques used at this moment. In spite of the fact that there are techniques like the reinforcement and the clustering, which are widely used in the network management, we should focus on those algorithms which are more oriented to measure and predict the parameters inside of a network. In this project the aim is to analyse the different techniques but not which platform is going to be more powerful, so we will use python as the main technology, without going deeper inside other technologies like Kafka, Spark, R or IBM InfoSphere Streams [19].

Therefore, we are going to start with the algorithms we should use to perform analysis built on the premise that this project aims to achieve the demonstration of how the ML algorithms can help in the performance of the new 5G network performance where it is represented in the figure 2.2.However, as addressed in the Recommendation ITU-T G.1000 on the QoS framework, the great challenge to success when deploying a cognitive network management model (Fig 2.2) is to deal with all the different QoS-related. This may become a difficult task when dealing with next-generation wireless ecosystems, where many unpredictable events may have an influence on the user´s experience. In view of this, some

recent studies suggest using big data analysis and ML algorithms for modelling possible future demands upcoming. ML techniques might be useful to infer rules from big data analysis and identify the KPIs/KQIs that will lead to automatically estimating the quality of a service. Selecting the most suitable learning algorithm may be critical to obtaining reliable results.

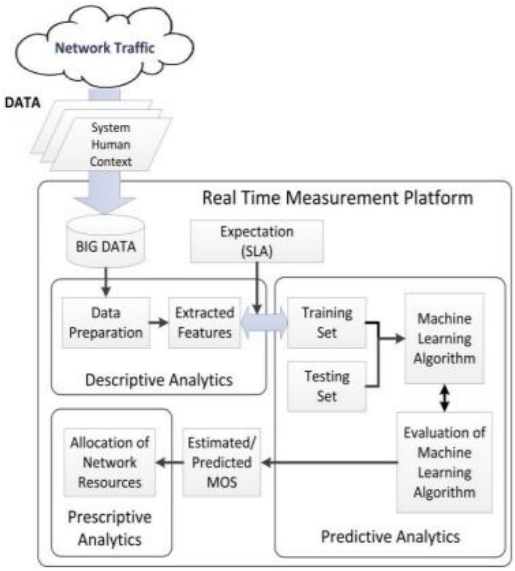

Fig 2.2 Modelling network metrics using a big-data analytics approach [17]

There are other techniques which can be more efficient in comparison with the typical supervised and unsupervised algorithm for the complexity of the network but their efficiency is still being investigated such as the Q-learning and the neural networks, as it is suggested in [15]. The authors discuss about the possibility of the implementation of this new techniques since a wide range of RL problems (including the ones of RRM) can be modelled as Markov decision processes. But in our case we will deal with the warehouse and databases problem.

The congestion avoidance and control is other issue which a network usually must deal with, then the reserved resources based on the forecast traffic of cells should be calculated or even the abnormal problems inside the network have to be analysed, where some algorithms are especially useful since they are exclusively design for it that depending if the data is labelled or unlabelled they will be apply like K-Nearest Neighbours Classifier.

In conclusion, for the mobile traffic there are other considerations that the network must realize where the feature selection plays an important role reducing the possible variable from hundreds to a short range, furthermore is quite usual that k-means method could be used as efficient technique extracting common patterns and in order to weekly traffic behaviour can be predicted, even the hourly traffic forecasting where is also common to see auto-regressive algorithms or Gaussian process due to the fact that as the weather it could imply random patterns which make more difficult the training and the correct predictions of them.

CHAPTER 3. IMPLEMENTATION AND SIMULATION.

3.1 Introduction of the use case

We have described in the first part of the project, the potential benefit of the application of the machine learning techniques for 5G networks. Then, we have identified a management architecture which can help us in the implementation and the enhancement of the performance applying the algorithms.

The next step in order to show how it will work this architecture, it is the necessity of obtaining the data which will be received in a real environment, but here we found several problems:

1. The first one is that the companies saved this data and it is impossible to get it. This can be explained by two main factors, this technology is still in a process of development and on the other hand, the companies would not share the metrics until the technology was well implemented like 4G.
2. Due to the first problem it was thought that in order to get the most reliable data we will need to reduce the global vision, meaning that, instead of obtaining data from the 5G we should think about getting it from 4G but it appears another problem; there will be a lot of parameters missing due to the similarity between technologies but finally, they are different in many aspects like the slicing, which is not contemplated, consequently we would be predicting something unreal and without sense.
3. The third one appears when we want to focus on one part of the wide 5G structure, due to the fact that it would be too much if we try to analyse several parts (clustering, forecast traffic behaviours of cells or managing handovers among other parameters) [9] so we have to add the lack of real data to we cannot match uncorrelated data, because we will need much more amounts of data in order to get a notable performance, therefore we have to choose a specific 5G field.
4. The final problem is about how we will simulate the data. To do that we must implement a sophisticated simulator and generated artificial metrics based on mathematical and theoretical models which can approach a real environment in order to get a good accuracy**.**

Therefore, these are the problems which we had to face up. Therefore, we thought that the best choice would be to use a free-source software simulator and see what kind of metrics we could obtain from it. But still we did not know in what part of 5G we should focus on, so that we would try to obtain the data performance from a datacenter, due to the fact that the datacenter will be an important part of 5G network and the abstraction services.

Then we would try to improve the performance inside a datacenter, where every host has an important cost not only in terms of money, but also in time when it is a must to satisfied a service as quickly as possible and achieve a good optimization of the structure, in order to not waste resources.

3.2 Implementation of the use case

The first step was to try the different simulators which currently exist and how they gave us the data. In particular, we tested different simulators which are very common in the market, even installing a virtual machine in Linux; among others there were tested: “DeepMIMO_simulator, ns-allinone, NYUSIM, omnetpp, Nemo_outdoor, GNS3 etc”. Additionally, we have performed the review of studies about mathematical models in order to derive the end-to-end latency by changing the parameters. Nevertheless, at the end, it resulted in a too complex task.

Finally, we found a simulator based on java which has been used following the recommendations in [10], within cloud environments, which provides the most important features that are included into a datacentre called CloudSim**,** reported in the paper [11]. CloudSim is a simulator which allows to simulate according to several parameters and requirements, varying along the time, the different conditions which can appear inside of cloud computing scenario. It also includes the customization interfaces for the implementation of policies and providing techniques for the allocation of VMs under an inter-networked cloud which can simulate the data centre segment in a 5G ecosystem.

The cloud environments will approach the next generation of datacenters, purchasing dynamic and flexible application provision by means of virtual services, where the users will be able to access to them at every moment from anywhere, enabling the application deployment according to the demand and QoS requirements. [11] Therefore, we have a simulator which can provide us the use case in order to simulate the part of the data collector, so we do not need to use any monitoring software.

CloudSim generates the part of the physical requirements inside hosts and datacentre, this means that every service which appears at every moment, it will have several characteristics of memory, speed, time and CPU usage. Basically the idea is that behind of a 5G network, there are a lot of important parameters to be care. As an example, if a host is too saturated, it will suppose that a service it could not be satisfied properly, implying a possible delay and making slower the network.

As we can observe in the figure 1.9 the part of the virtual network function and management and network orchestration will be directly implemented by the metrics which are given in the simulator, whose parameters will be manually controlled in order to obtain different situations. We will be the management part and the actuator in order to focus on how the techniques can help inside of the 5G network.

3.2.1 CloudSim structure

Previously to with the definition of the simulation scenario, we must to explain what it is the structure of the simulator that we have used in order to understand what kind of data we are going to lead with. Obviously, the simulation will give us an approximation of the cloud environment which it could appear inside of a

datacenter, where the 5G services are allocated. However, it will allow us to demonstrate how the ML techniques can help to the 5G networks and not just predict possible fails but also optimize and improve the performance in the whole network.

We must understand the terminology used inside the CloudSim in order to clarify with what kind of data we are going to lead. The most important element that we need to understand is the **cloudlet.** It is quite abstract because it defines a service which it could be provide in a 5G network.

Policies decide the list of potential actions to be taken to guarantee the VNF Monitoring provision, in accordance with CRI of SDN and NVF elements as well as 5G aspects like, for example, the number of active users consuming network services which in our case it would be cloudlets, the network infrastructure's location, or the users' mobility. Policies actions influence the behaviour of the components and layers of the proposed architecture as it is explained in the section 1.3.2 [12]

As we have observed in the figure 1.9, there are two clear sections for the radio resources, where it is contemplated the part of mobility and user allocation, which is intrinsically connected by means of the optical transport network to the cloud services. In this part is where we are simulating; therefore, every service requirement it will have this two parts mobile and cloud, so summing up everything a cloudlet will represent the application services. It encapsulates the number of instructions that will be executed, and the amount of disk transfer to maintain the task.

The other elements which we have to understand are the hosts machine, virtual machines, CPU, RAM and BW usage, datacenter and service broker explained below [14].

- Datacenter: it is a set of typical hosts or servers; it is possible to have several datacenter as in the real environments and make the communication among them.
- Hosts: It operates the physical machine and joins information provided by the processing unit, main memory virtualization monitor specification, and disk and network bandwidth. It also specifies the information about the policies for control and processing unit, disk, network and main memory to virtual machines.
- Virtual machine: it is an application software which emulates the instructions of a real computer, every virtual machine it will have determined several parameters which cannot exceed the requirements of the hosts, it will be used to separate the tasks, in order to make it more efficient and work in parallel satisfying different tasks/services at the same time.
- CPU, RAM and BW usage: the usage will measure the different requirements and how they are evolving along every simulation getting

directly the metrics from these parameters and applying directly the ML algorithms according to these data.

- Service Broker: it is the element which choose which virtual machine will allocate and provide the service request.
- Scheduling interval: Each task is represented by an *interval* describing the time in which it needs to be executed.
- Host PES and VM PES: Indicates the number of cores inside of a host or inside of a virtual machine typically are from 2 to 8 inside of a VM and depending of the number of these VMs it would change the number inside the host.

CloudSim follows several mathematical models in order to estimate the different parameters like capacity, usage, etc. [11]. The models have not been changed, and we have chosen the stochastic model which provides a more variety of values in order to force the algorithm as much as it can.

Finally, the structure of this simulator can be observed in the figure 3.1 where it can be found the elements mentioned above. Thus we could find that every user it would suppose a service request, then every request will have different characteristics like the length in terms of memory, the size, RAM etc. There will be some parameters that in a datacenter are always stable like the bandwidth between hosts and inside these ones, so here appears a possible case for applying machine learning algorithms in order to optimize the different hosts. Moreover, the storage capacity and the RAM are usually equal in every host, so we must make it as efficient as possible distributing every cloudlet in different virtual machines.

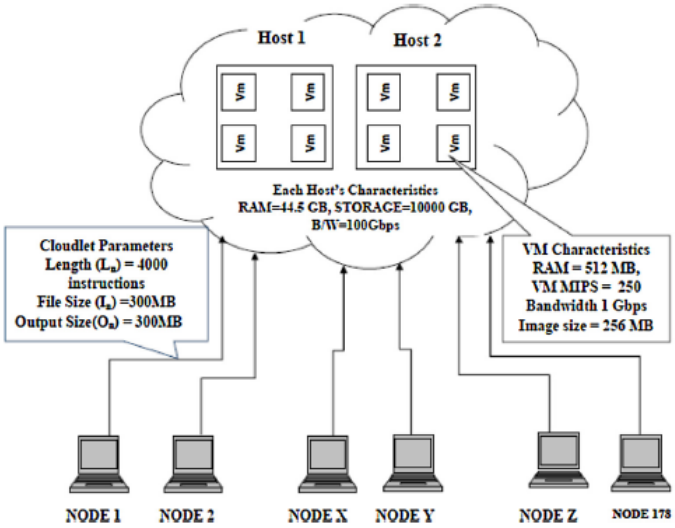

Fig 3.1 Operational CloudSim environment

In conclusion, the simulator will represent a scenario previously defined by us, with a limited number of hosts which will be the main constraint in order to create the VMs. We can increase (or introduce) the amount of cloudlets. Depending on the amount of demands there will be a moment when the time in order to satisfy

the demand it would unacceptable so we should try to find the better solution. This can be extrapolated to the real environment where several users are asking different services with different needs and you have limited resources in order to satisfy them; here is where we must apply ML techniques in order to find the most optimized path.

3.2.2 Control Simulation

In this section we are going to present the interface of the software used and what modification we have included in order to adapt the simulator to our use case.

We have needed several modules from CloudSim API like CloudSim plus and EDGE-CloudSim, the libraries imported from java API are included when you download the program so the only part which have been modified has been the example module. In the figure 3.2 we can see that we have created a specific class in order to reproduce as similar as to a real example for our COGNET case, we want to simulate the real conditions inside of a datacenter but in the conditions of a 5G network.

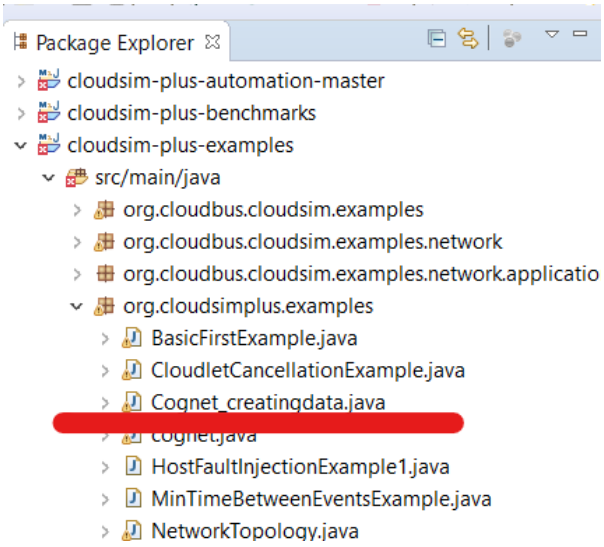

Fig 3.2 CloudSim modules

Therefore, we can observe (Figure 3.2) that there is not any interface which can help us to use the simulator, thus we have to understand how the program works from the code part, after understand what data is generated we have created a module adapting it to our requirements.

We have to establish several constraints which are not included in the simulator (Figure 3.3)**,** like the maximum number of RAM which cannot exceed the Host Ram, also the virtual machines which can be created per host and making a little modification we can get that generates a simulation with similar conditions but changing the number of cloudlets which are arriving in order to see how it would be the performance of the datacenter.

After defining the constraints, we will be able to control every parameter inside of the datacenter, there will be some parameters which never change trying to make it as real as possible. We will establish the typical values inside of a datacenter.

```
numsimul=consultlastsimul();
System.out.println("este es el que vale" + numsimul);
printinfile("N_simul,Ram_host,Storage_host,Vm_Ram,Num_cloudlets,storage_vm,N_VMS,CLOUDLETS_LENGTH",true);
for (CLOUDLETS_INITIAL_LENGTH=10000; (CLOUDLETS_INITIAL_LENGTH*CLOUDLETS)<(storage_host*HOSTS); CLOUDLETS_INITIAL_LENGTH= CLOUDLETS_INITIAL_LENGTH+
for (i=25; i<35; i++) {
    numsimul++;
    if((VM_RAM*VMS<=HOSTS*ram_host)  && ((VM_PES*VMS) <= HOST_PES) ) { //VM_PES*(i-2)
        VMS=i-1;
    } else if (HOST_PES >= VM_PES*(HOSTS*ram_host)/VM_RAM) {
        VMS=(int) ((HOSTS*ram_host)/VM_RAM);   //Maximum value posible without exceding the HOST RAM
    } else {
        VMS=HOST_PES/VM_PES; //Maximum number of VM that can be created
     }

storage_vm=storage_host/VMS;  //The storage of every VM will change depending on the number of VM in order to adapt
CLOUDLETS=i*3;
```

Fig 3.3 Parameter definition inside the code

The program has been modified in order to generate the data conveniently, we will have two different datasets whose relationship will be established by the number of simulation, the first one [Annex A] gives us what has been the state of every cloudlet which have entered into the datacentre. The cloudlets are identify by an id number, each cloudlet will be attended by only one virtual machine allocated in a host, so, for instance, if the cloudlet 57 has been attended by the virtual machine 6 inside the host 0 cannot be at the same time attended by other one. Furthermore, as we have mentioned previously every service (cloudlet) will be probably different from the others, it is difficult that two different services have the same characteristics, consequently the size needed will differ and it would need different times to be satisfy.

The other dataset [Annex B] will concern to the part of each moment where the execution is occurring. In the figure it is shown how evolve the different usage along the time until the last cloudlet it has been attended as well as how many resources they have to spend in order to satisfy cloudlets at every moment.

However, we needed another dataset [Annex B] which indicates what are the initial conditions in every simulation in order to reinforce the algorithm and how has evolved the simulation, where it will be included the number of simulations and the initial conditions presented in [Annex A]**.**

Finally, we have to match the different datasets in only one; by means of python we have created the cognet_dataset where it has been integrated all the information needed for applying the machine learning algorithms. The dataset (Fig 3.4) is composed by all the simulations made. In order to get the most important features and getting the most reasonable values and realistic instead of using the metric provided at every moment we have got the average and the maximum and minimum. Therefore, for each virtual machine in every simulation we will get the data which has been allocated there, number of cloudlets total size, the maximum usage used as well as the average. The explanation for doing it, comes from the necessity of accuracy, we could extrapolate it to a real time monitoring but it would have cost much more time and we want to demonstrate

how a simple application of machine learning algorithms can improve the performance of the network.

The main point here is that we can have as much data as we want, so it will depend on how many simulations and which parameters you change but finally we have made 3 different burst, with 250, 500 and with 750 simulations, in order to see how the accuracy can increase when more data is added. It could be applied to the dataset which stores the data at every moment but there it was not other important data which should be analyse too.

	N_simul	Host_id	Vm_id	Ram_host	Storage_host	Vm_Ram	Num_cloudlets	storage_vm	Max_CpuUsage	Max_RamUsage	...	Mean_BwUsage
0	1	0.0	0.0	32768	2000000	2048	25	500000	95.219339	100.0	...	72.588079
1	1	0.0	1.0	32768	2000000	2048	25	500000	80.909234	100.0	...	80.237086
2	1	0.0	2.0	32768	2000000	2048	25	500000	83.097853	100.0	...	89.166887
3	1	0.0	3.0	32768	2000000	2048	25	500000	88.416054	100.0	...	63.355629
4	2	0.0	0.0	32768	2000000	2048	30	400000	84.949036	100.0	...	87.877049

Min_CpuUsage	Min_RamUsage	Cloudletlen_mean	Max_Clodlets	Mean_ExecTime	Mean_FinishTime	NumCloudlet	max_Cloudletlen	min_Cloudletlen
0.0	0.0	13907.000000	24.0	33.714286	43.714286	7.0	28000.0	2666.0
0.0	0.0	15555.500000	21.0	38.000000	45.500000	6.0	29000.0	3333.0
0.0	0.0	17253.333333	22.0	35.666667	44.500000	6.0	40000.0	3076.0
0.0	0.0	13920.500000	23.0	28.833333	39.000000	6.0	27000.0	2857.0
0.0	0.0	15383.500000	25.0	42.000000	50.666667	6.0	30000.0	2857.0

Fig 3.4 Dataset from CloudSim environment

3.3 Parameters inside CloudSim

In order to understand what each parameter means and its importance, in this section it will be explained their characteristics, meaning and how has been calculated in a simple way.

Inside of a data analysis there is always an important background related to the data, a lot of times just for making the predictions we do not need to understand what the data means, but it is deeply helpful that we identify the parameters before we proceed with the classification.

As we have explained the simulation will provide us three different datasets that we will combine and making the feature selection. In all the databases it is a must that exists an *id* in order to identify the rows and making that a parameter can search. In our implementation, the id will be given by the simulation, virtual machine and host; this way, every simulation will provide different measures. The number of host will be established from the beginning due to the fact that it will determine the result of every simulation.

We can distinguish between input parameters, output parameters and identifying or categorical parameters. The number of hosts also will determine the maximum number of virtual machine which can exists at the same time if we have established that the RAM of a VM is 2 Gb and there two host of 8 Gb it could build

as maximum 8 VM, 4 in each host. The creation of this VMs it would depend on the requirements in the simulation, if they have to satisfy more resource request or no. Every virtual machine has the same parameters but they could change depending the cloudlet which they are attending in that moment. Some of these parameters will be the total storage, the number of cloudlets which will be attended in every simulation, the maximum of CPU, RAM and Bandwidth as well as the meaning indicating the typical and the strange parameters, how it is the mean size of the cloudlets in order to make the estimations, it can be understood better looking at the table 3.1.

Table 3.1 CloudSim parameters extracted

Input parameters	Output parameters	Categorical parameters
Ram Host	Maximum CPU,BW and RAM usage	Number of simulation
Storage host	Mean CPU,BW and RAM usage	Host id
Total number of cloudlets	Number of the cloudlet	Virtual machine id
Cloudlets length	Mean execution time	
Total number of virtual machines	Mean finish time	
Virtual machines parameters (dependent on the host)	Maximum and minimum size of the cloudlet attended	

Cloudlet	Status	DC	Host	Host PEs	VM	VM PEs	CloudletLen	CloudletPEs	StartTime	FinishTime	ExecTime
44	SUCCESS	1	1	32	10	3	5111	1	18	30	12
45	SUCCESS	1	1	32	11	3	4600	1	20	30	10
46	SUCCESS	1	1	32	12	3	4181	1	22	31	9
43	SUCCESS	1	0	32	9	3	5750	1	16	33	18
47	SUCCESS	1	1	32	13	3	3833	1	24	34	10
41	SUCCESS	1	0	32	7	4	7666	1	12	35	23
48	SUCCESS	1	1	32	14	3	3538	1	26	36	11
49	SUCCESS	1	1	32	15	3	3285	1	28	36	9
50	SUCCESS	1	1	32	16	3	3066	1	30	36	7
42	SUCCESS	1	0	32	8	4	6571	1	14	40	27
53	SUCCESS	1	0	32	2	3	2555	1	36	41	5

Fig 3.5 Time parameters

There are two output parameters which need to be explained because of their ambiguity, that is the times given by the simulator. One of them is the finish time which usually is greater than the execution time due to the fact that it represents the time needed for satisfying a cloudlet request. The main difference between them is that the finish time will depend always on the start time being the execution time the difference ExecTime= FinishTime - StartTime, concluding that the execution time it is how long it takes to complete a cloudlet request.

Consequently, this two times are important in order to estimated when it would be the next virtual machine free for attending the next service request and how long it takes complete the service request.

CHAPTER 4. DEVELOPMENT AND RESULTS

After dealing with the problems related to the simulator, modifying the code in order to give us the implementations for the generation of different cloud centre scenarios, they would change depending on the number of resources which have to be satisfy at every moment. Therefore, once we have obtained all the data and the metrics we have to develop the analysis program, typically there are several steps which are followed when we have to applied the techniques.

- Reinforcement: Previously of cleaning the data we can achieve a better performance if the data is supported by other dataset in order to give it more sense inside of our predictions. In our case, we have mixed three different datasets (data at real time, final organization of the cloudlets and the initial conditions) due to the fact that every single one will not give us the required performance as we have explained previously in the section 1.5

- Missing data: The missing data is a typical problem which can appear inside of a dataset or in a real environment due to the fact that not always the data will have sense or it could be collected; the missing data appear when an error occurs too, so that it is important to consider it. There are several techniques which can calculate the approximate value which is lack in the cell but in our case will be substituted by ceros just for making it simplified. Feature selection: when we talk about the data a lot of times is spoken that more data is better but not always is as easy as it seems. In the datasets there will be always parameters which will add more information and they will be more useful than others, but in our case we have to eliminate those columns which will not change the final result in order to streamline the process. This is the reason why we have removed several columns like the minimum usage of the different process as RAM, CPU or BW because it is logical that it would be 0 for all of them. In this case it is easy to see that those parameters are useless but in other cases we have to proceed to further analyse the data.

- Clean data: The next logical step will be to remove those columns which will not be useful in a future, usually those parameters which are repeated. They probably do not improve the prediction or otherwise, they are not correlated with the others parameters, because sometimes to have more data it could have a high price, reducing the accuracy. Other data like the number of cores, storage host or Ram host will keep immutable, so it has to be removed.

- Categorical data: this kind of data usually supposed a problem in terms of how it is going to lead with it, although it is important data, it must be

transform previously, first into numerical and after, indicating that this data is categorical to the algorithm. In the project the id of the host or virtual machines and the number of simulation belong to this type of data, but it is important to know it, because indicates more than only a numerical value, so that, these parameters will be considered as a categorical data.

- Normalization, finally when we have all the elements which we want to include into the prediction, cleaned and organized, this step usually helps to improve much more the prediction. In this project, the data has been normalized between 0 and 1, due to the fact that we are not going to predict categorical data but the all the numerical data is continuous, this means that data will have very disperse values making more complicated their management.

4.1 Predictions

In this section we are explaining how the prediction was performed. Generally speaking, after making all the adjustments, fixing and cleaning all the data, we have to choose which parameters we want to predict, from the point of view of a 5G network. Since we focus on the cloud part we should predict those parameters which makes sensible that the service can be offered in a properly way.

Table 4.1 Algorithms accuracy in the use case

Algorithms	Legend	Accuracy 250 simulations	Accuracy 500 simulations	Accuracy 750 simulations
LogisticRegression	LR	0.088356	0.119051	0.116883
LinearDiscriminantAnalysis	LDA	0.481827	0.720334	0.445292
KNeighborsClassifier	KNN	0.166602	0.127659	0.164109
DecisionTreeClassifier	CART	0.212278	0.214519	0.280696
GaussianNB	NB	0.112309	0.130283	0.155449
Support vector machine	SVM	0.070757	0.133273	0.172473

As we have explained in the section 2.1, there are several techniques that we could apply basing on the type of the data; we can use regression, classification, clustering or anomaly detection but at the first sight we can observe that we are leading with continuous data which it means there are several values for the result. We can discard the last two options, the first one, anomaly detection, in this case we do not want to find errors or fails inside the cloud centre, consequently the values will be among a regular range.

About clustering, whose main objective looks for detection of data groups forms, is disposable, largely due to the values will depend on the initial variable conditions, making quite difficult that the output could be similar in different

moments. Therefore, at the end, we will keep the other two options, in order to see which, one give us a better performance.

However, there is a step which is quite common when we are going to apply ML techniques, that consists in splitting the dataset into two different datasets, the train set and the test set. The train set will be used to train the algorithm in order to make the algorithm which results (Validation set in the figure 4.1) gives with certain components; the test set will be used to prove if the algorithm works properly, so that when there was more data the training will be better and the prediction will earn more reliability.

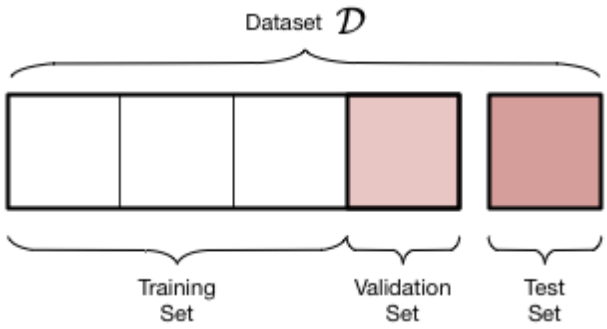

Fig 4.1 Dataset splitted

4.1.1 Classification.

We are going to start analysing how it is the performance of the classification techniques. As we have explained, we are leading with continuous data, then the classification could not work with it, we need to transform it into exact integer without decimals, losing a wide range of values as well as accuracy in the prediction of the parameters, making classification at the first sight, quite vague.

Then proceeding with the analysis we have to encode those values in a range of 0.1 so that, if we have 0.423 it would change to 0.4. When we have normalized dataset encoded we must choose which parameter will be predicted. In our case the most efficient would be to make the estimation of several parameters but first we have to select the algorithm which suits more properly. We decided to start with the mean execution time, due to the fact is one of the most important parameters that we have to take into account, as we have explained previously it will suppose the difference between satisfying a service in the time required or not.

We have chosen six different ML algorithms in order to test how the classification will work with this dataset, between them it would be one algorithm of regression in order to compare the efficiency. The algorithms will be the typical ones inside of classification and their results can be seen in the table 4.1 represented in the figure 4.2.

We can check that the regression does not work properly with the encoded values, so the result is quite logical due to the fact it is an algorithm made for

disperse values which are not between a short range. Regarding to the other algorithms we can see that the classification will not work properly, our suppositions were correct. In spite of the fact that they are not accurate the linear discriminant analysis curiously it has almost a 50% of probability of predict right but this is not enough, it can be explained because LDA is also closely related to principal component analysis (PCA) and factor analysis in that they both look for linear combinations of variables which best explain the data.

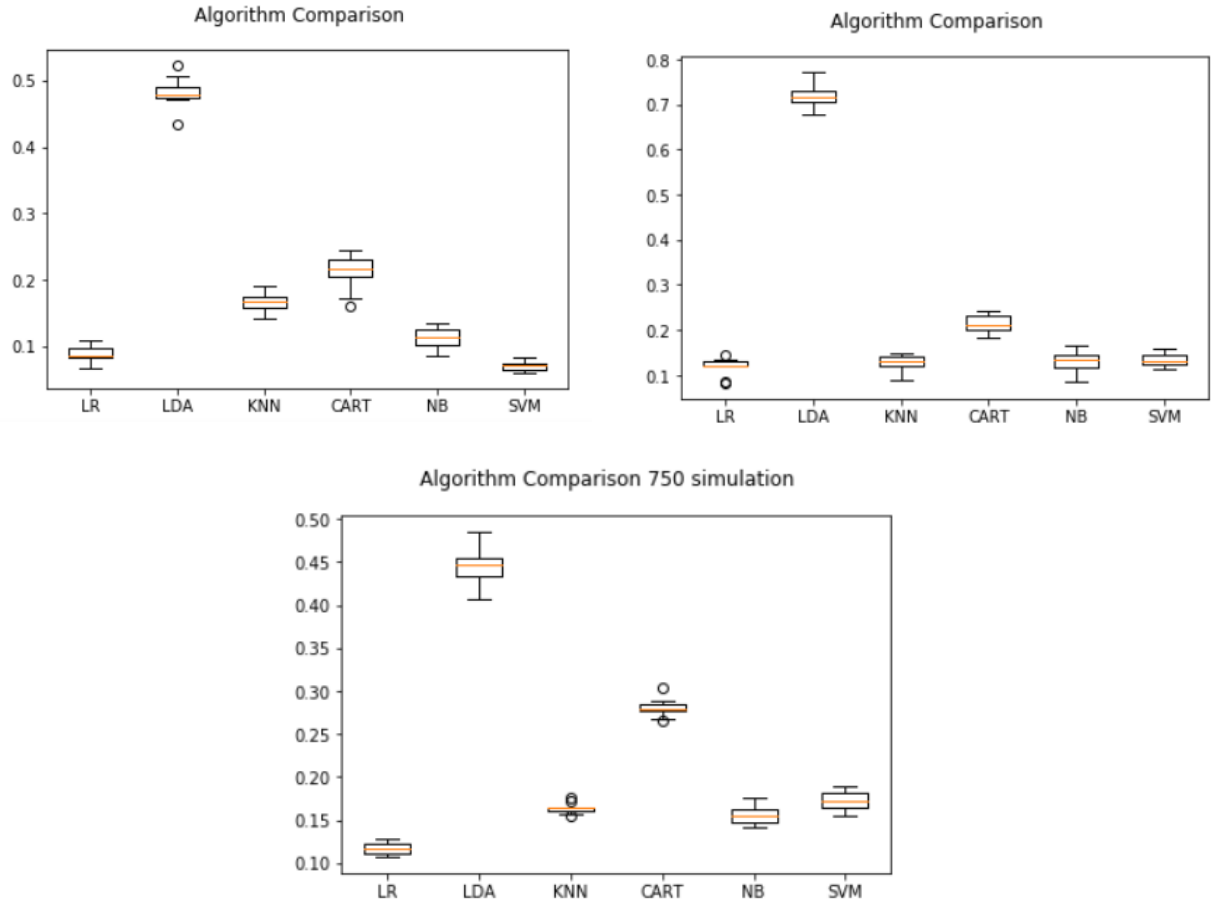

Fig 4.2 Algorithm comparison with 250,500 and 750 simulations

4.1.2 Regression

Once we have analysed the performance of the classification, we must try it with regression. Previously we have explained that we will test the efficiency of these algorithms with three different datasets, made by 250, 500 and 750 simulations in order to work in the case of the last one with 23 columns and more than 12000 rows which it will give almost 300.000 cells of data. Usually the numbers are even greater but in our case we want to measure and extract the conclusion of how the machine learning can help to predict possible over consumption, or saturation

of the resources making unsustainable the time needed with the current VMs available.

Making a first sight into regression we test what result gives the linear regression this time without encoding the data and we compare the validation set with the prediction set in order to see if it could suit.

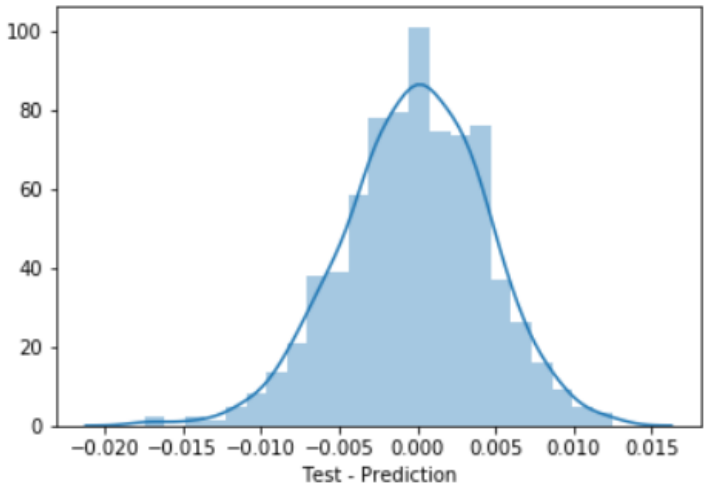

Fig 4.3 Plotting prediction vs test data with 250 simulations

The fig 4.3 indicates in a simple way how the predictions are according to the validation set, as we can observe it plots a typical normal distribution, which makes that the most of the values will be predicted correctly because the 80% of values are situated between the -0.01 and 0.01 of the deviation. The x axis indicates if the deviation between the prediction and the validation set it is too great, so in this case we can see that the values in a range of 0.02 is a satisfactory outcome.

Consequently, we can continue applying different regression algorithms in order to check the performance in every case. Previously, we have explained that the first parameter which would be evaluated it will be the execution time, this parameter will be evaluated individually due to its importance. In order to see the improvement into the performance the algorithms will be tested with the datasets of 250,500 and 750. Applying in each of them the same algorithms and trying to predict the execution time. Finally, if the regression techniques suit properly and we achieve the performance desired we will test how the multi target regression algorithm can be applied.

4.2 Results

In this section we are going to explain the results achieved. The first algorithm which has been tested is the linear regression model which can be observed in the figure 4.4 where is one of the simplest algorithms but it works quite well with the data, giving well outcomes and the percentage of being well predicted is quite high. So we can deduce that these techniques it is a good candidate if we want to predict only one target, but commonly in the real 5G scenarios, this could have

a high cost, due to the fact that it will require a high level of computation in order to calculate which resources are needed in each case to accomplish the expectations.

In the figure 4.4 we can see two plots.; The first one represents the deviation of the results; if the line is straighter, the performance and the accuracy will be better; on the left, plot indicates what are the normalized values in every virtual machine, so there we can see that the times of the execution are quite similar to the validation dataset.

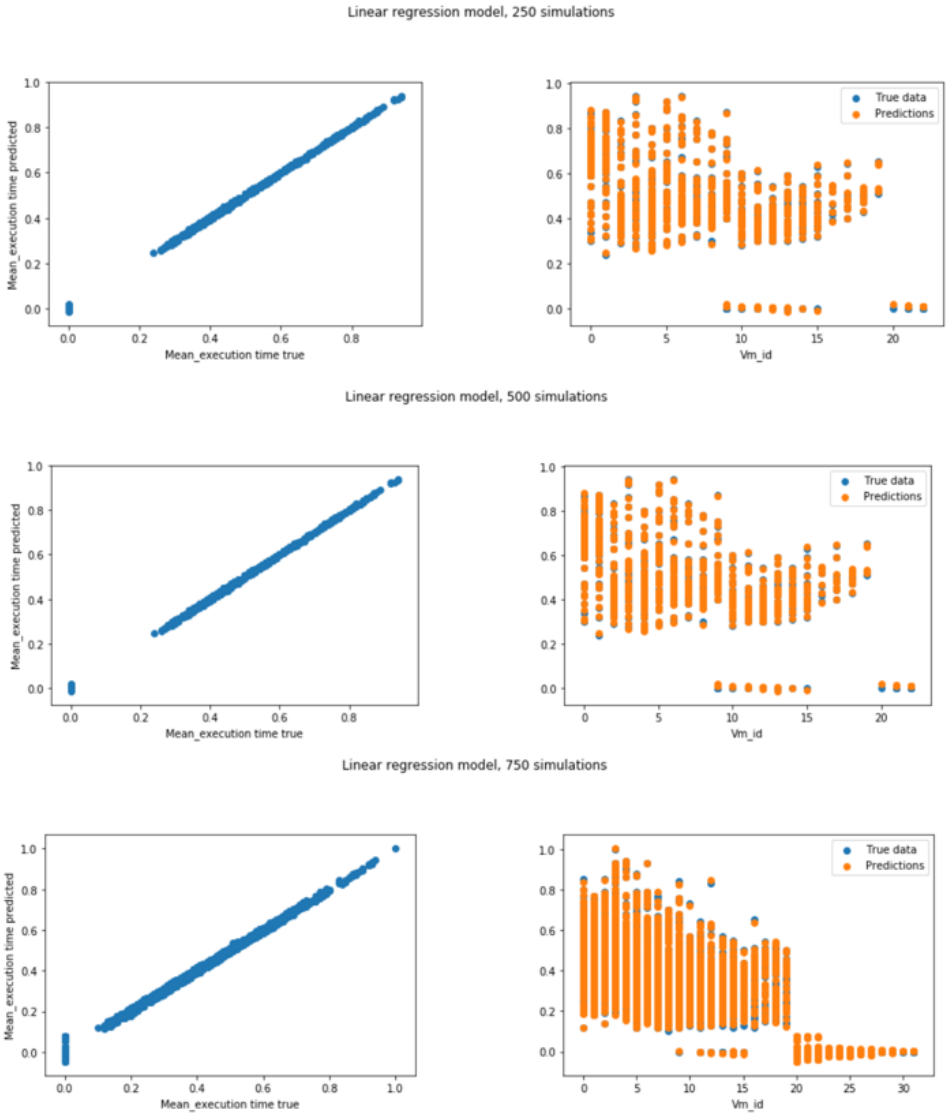

Fig 4.4 Linear regression model comparison

In the table in the annex C is indicated the response in each case, where the error which is obtained from the linear regression is lesser than in the other cases.

After test the linear regression with the three datasets it was the time of support vector regression. This algorithm creates a line or a hyperplane which separates the data into classes, consequently the general idea is that the algorithm takes the data as an input and outputs a line that separates those classes, but it would

be taken different lines until it gets the best candidate. Easily we can observe that this algorithm can be used with classification and regression but in our case the data is too much disperse in order to get a good performance.

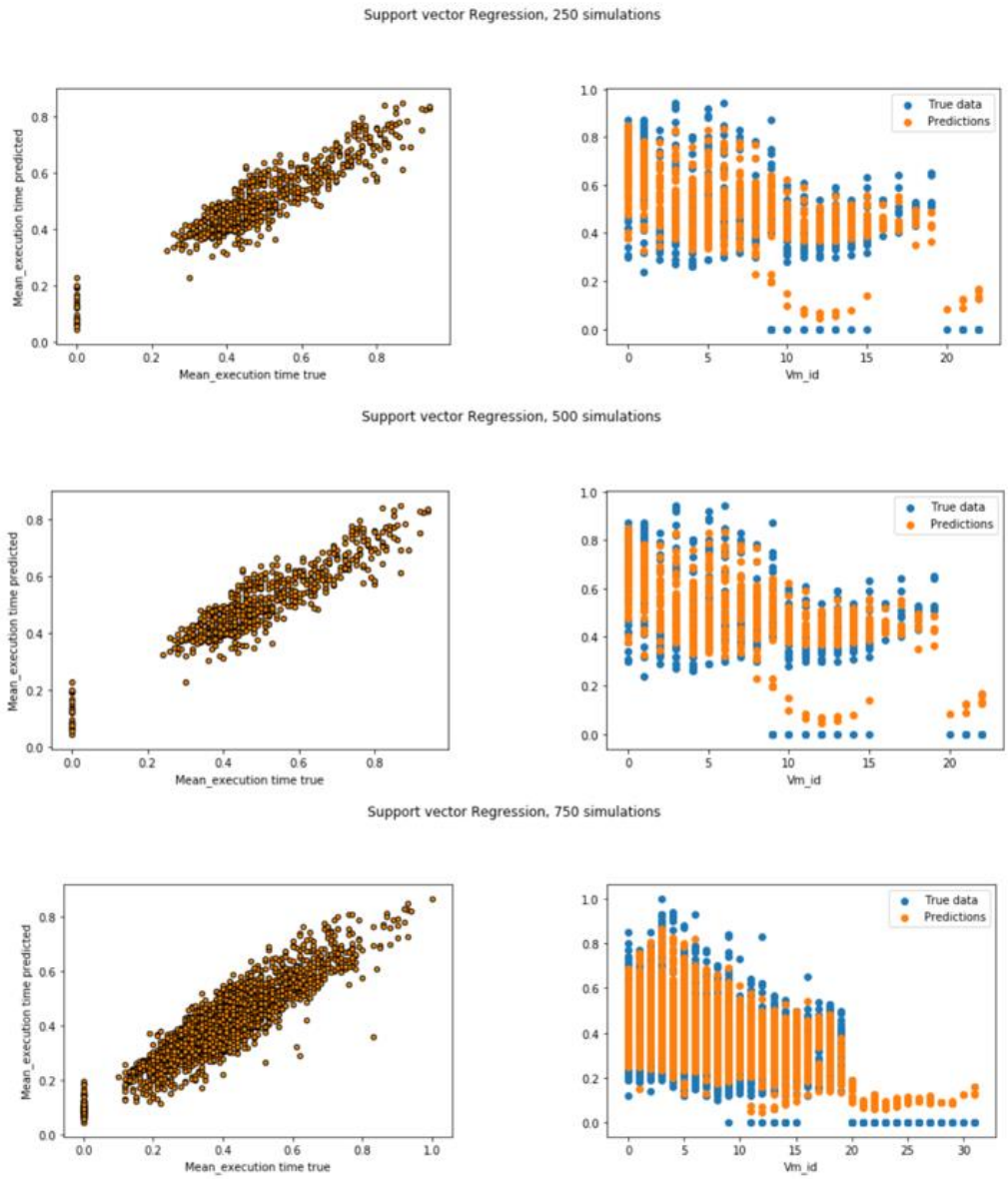

Fig 4.5 Support vector machine model comparison

In the figure 4.5, **it is** shown how the dispersion from 250 to 750 simulation increases instead of reducing it; the error introduced by this algorithm makes that it must be discard for the used inside the cloud environment in the 5G network.

Meanwhile, we checked the last algorithm for predictions of only one target, which is the decision tree. The general idea is to break down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. This algorithm was chosen because it can handle with categorical and numerical data, furthermore it is highly extended and quite known so that makes easy to work with it.

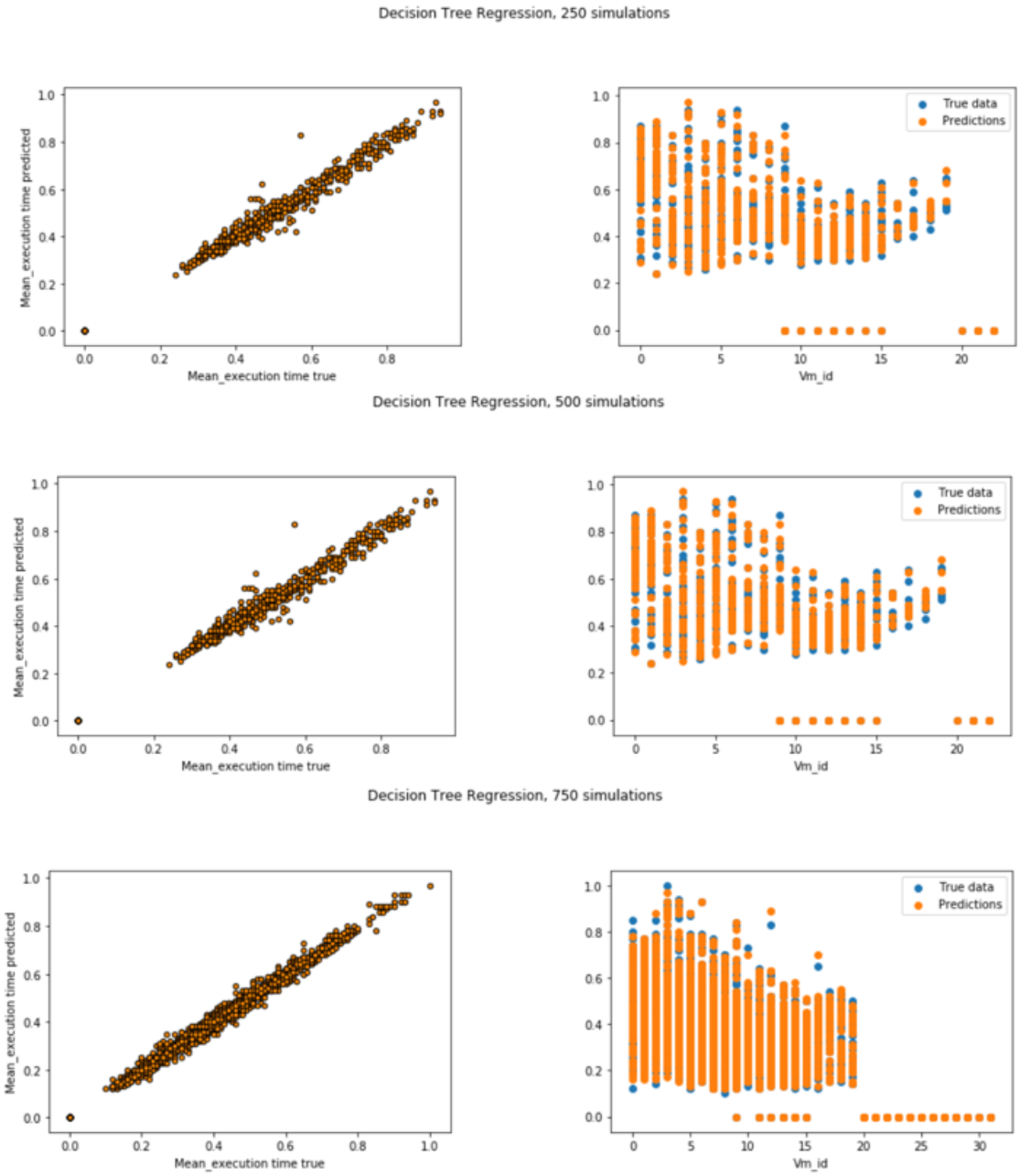

Fig 4.6 Decision Tree model comparison

In the figure 4.6 we see that the performance it is quite similar to the linear regression giving good performance too; this can be explained firstly because this algorithm can be used perfectly for regression cases and maybe needs a bit more time of processing but the result is quite accurate.

After checking the algorithms dedicated to predictions of only one target we thought that in a real environment it will be very tedious, even not considering that it will have a high cost to make every time the predictions just with one parameter. Eventually it was thought that with the linear regression it is possible to handle with several targets, then it would call multi target linear regression.

Multi target regression is the term used when there are multiple dependent variables. If the target variables are categorical like in this case, then it is called multi-label or multi-target classification. This algorithm can be used with decision trees but we are just tested it with the simple linear regression.

In this case, to begin with the MTR, we have to choose more than one variable which we want to predict, therefore we are going to predict those ones which will impact directly in the final performance of the network and those which will consume more resources, they are the RAM and CPU usage, the execution time and the finish time explained in the section 3.3.

Firstly, we will apply the algorithm for 250 simulations figure 4.7, but due to the fact that we are working with several targets in the same predictions the performance needs more data in order to extract conclusions, in the case of using 250 simulations we have that the times are quite precise, just in the case of the CPU the predictions are more disperse, so it will need more predictors in order to fix it.

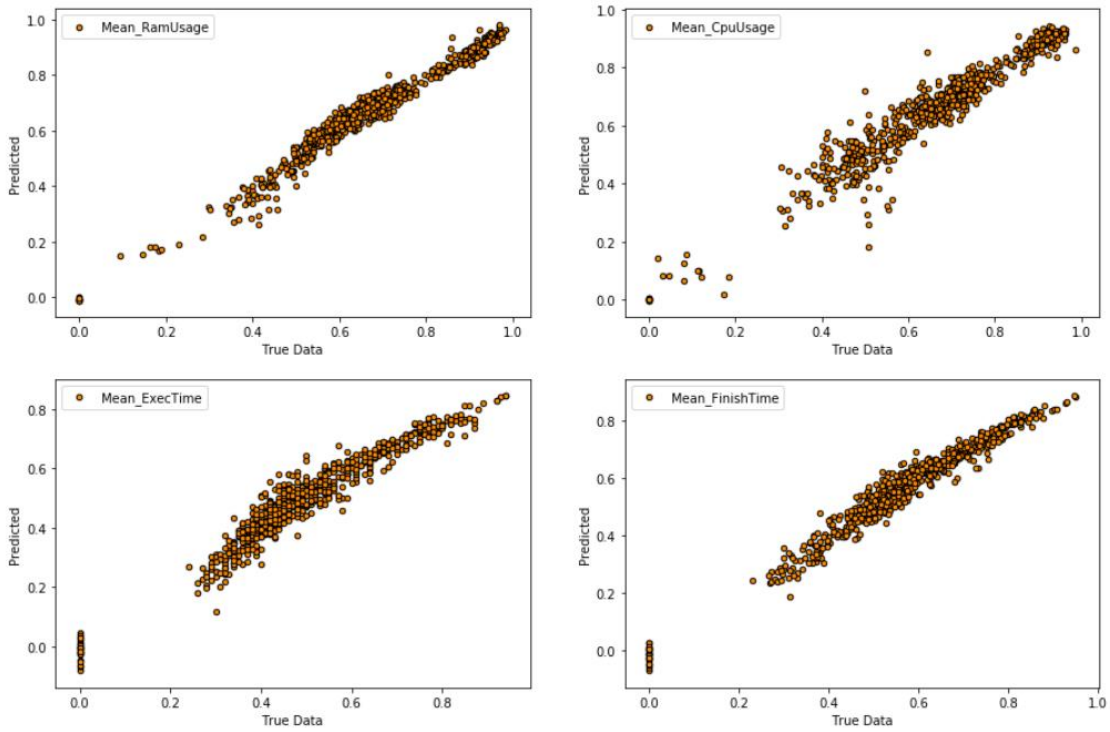

Fig 4.7 Multi target comparison prediction vs test data with 250 simulations

When we analysed the plots related to the network features we can see that the first results indicates that we are able to predict in every virtual machine or in future events (number of simulations) what could happen, as we observe the 10 first VMs are very saturated it overpasses the 0.8 of usage or time which it means that this data it will be situated very close to the maximum value, so knowing previously this situation we can modify the policies and distribute the resources in order to optimized and not overpass the red line in order to reduce the times and distribute the resources.

Consequently, we can extract the first conclusion from here that for example we could be able to see than in the time when the simulation 150 is being occurring the performance in 100 simulations after of the network will be too similar and it will be over saturating the first 10 VMs so it can be changed adding more priority to the other 15 VMs by means of policies avoiding this situation.

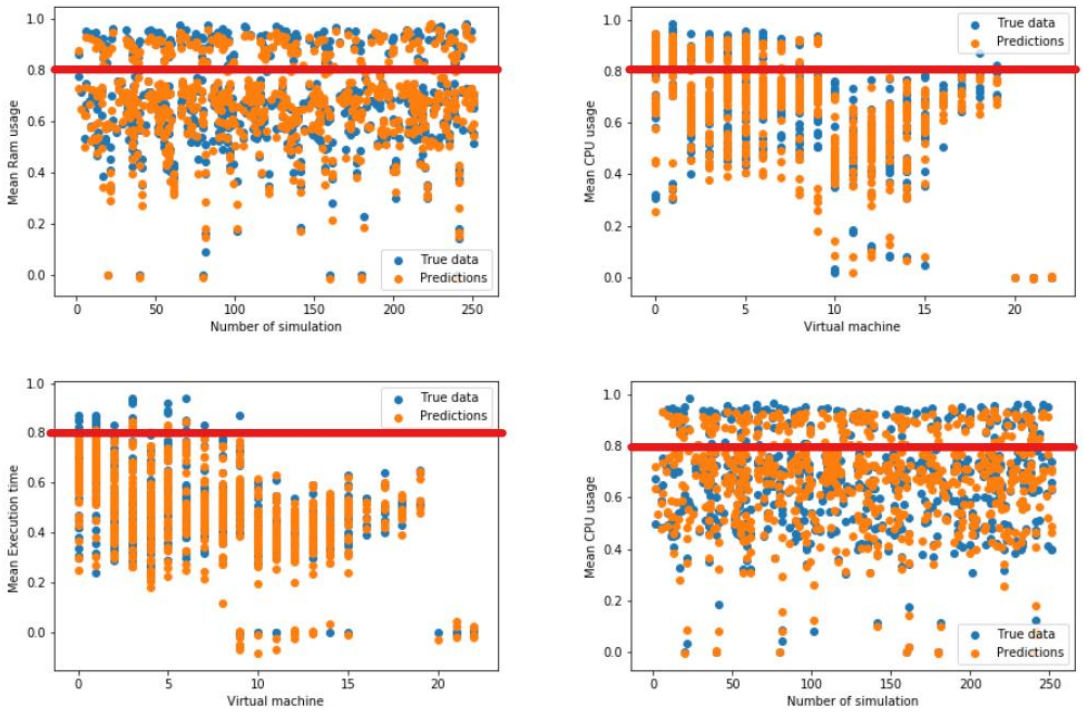

Fig 4.8 Multi target prediction with 250 simulations

But it is quite simple with just 250 simulations, this time we do not analysed the case with 500 simulations due to is very similar. Let's go with the case of 750 simulations, it is the most similar to a real environment.

The first result which we can obtain comparing the different algorithm is that depending how it is the behaviour of our scenario will change the techniques and the way of applying the algorithms, this means that now for a multiple random targets, like the times for finishing a service or the usage which it will be implied at every moment in a VM, we must have a strong predictions with enough accuracy, looking at the figure 4.9**,** there is a greater dispersion from the regression line but it does not mean something wrong, in this case if we see the table located in the annex C the error introduced into the prediction along more than 750 simulations it is quite sure that more than 90% of the predictions will be right, making the plots quite reliable in order to improve the performance. The parameter in both cases with 250 and 750, which introduces more error and uncertainty is the CPU usage, maybe the dispersion between the data makes more difficult for the regression techniques predict the values but it should be analysed as future work.

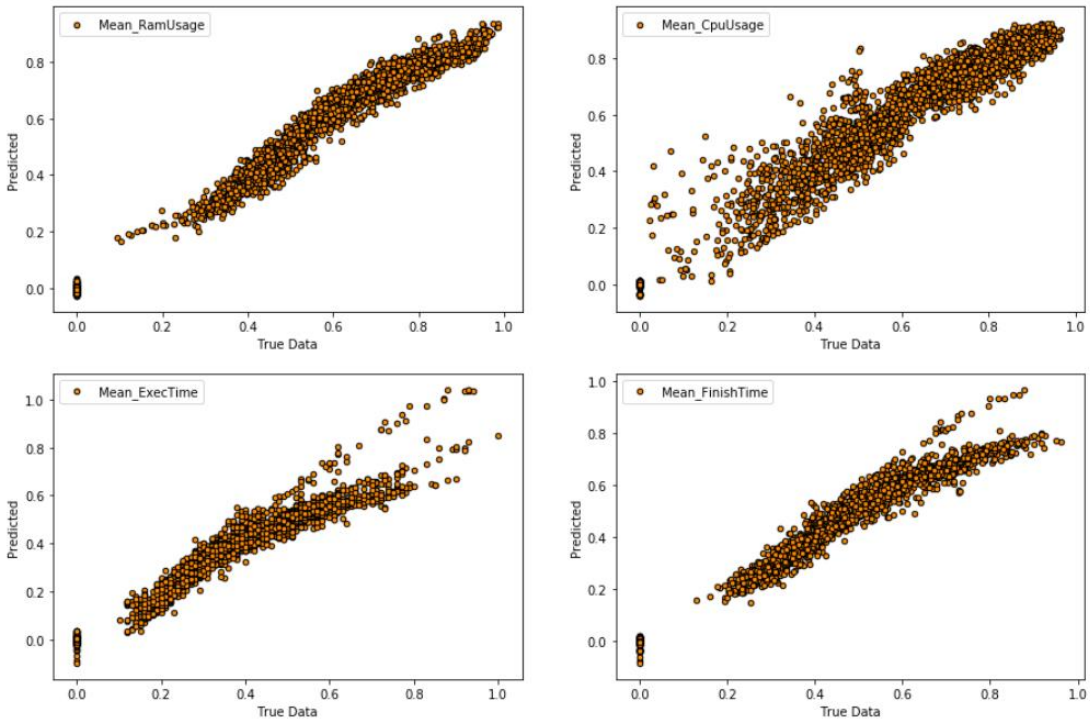

Fig 4.9 Multi target comparison prediction vs test data with 750 simulations

Once, when we have the results from the data along the 75 simulation we can realize that with 250 simulations we are able to train the algorithm and it will be able to predict the behaviour of the cloud scenario, In the figure 4.10 we can observe how the predictions works, finally we have a test set called Y_validation where it must be predicted, if instead of having this Y_validation we want to know what would happen in the next 500 simulations that in the real environment will be translated to real time, we are able to predict with more than 90% of probability what would happen.

The second result that we can extract in the case of a real scenario as we have explained for 250 simulations would be that once when we can trust on the prediction, now by means of actuators and policies it should be defined which it is the best plan for the optimization.

In the figure 4.10 we can divided in two plots those which are talking in terms along the time (number of simulations) and those which are focus on in each virtual machine, the first case it can be used to change that the RAM usage and the CPU usage are close to the maximum values along of the different scenarios, so it can be understood because only the half of virtual machines are doing almost all the work, just when they are totally saturated it uses the rest as we observed in the CPU usage vs VM plot. This can be extrapolated to the time needed for every VM to satisfied the services which is near to the maximum values, but the criteria here should be analysed thinking what parameters must be change in order to avoid this situation.

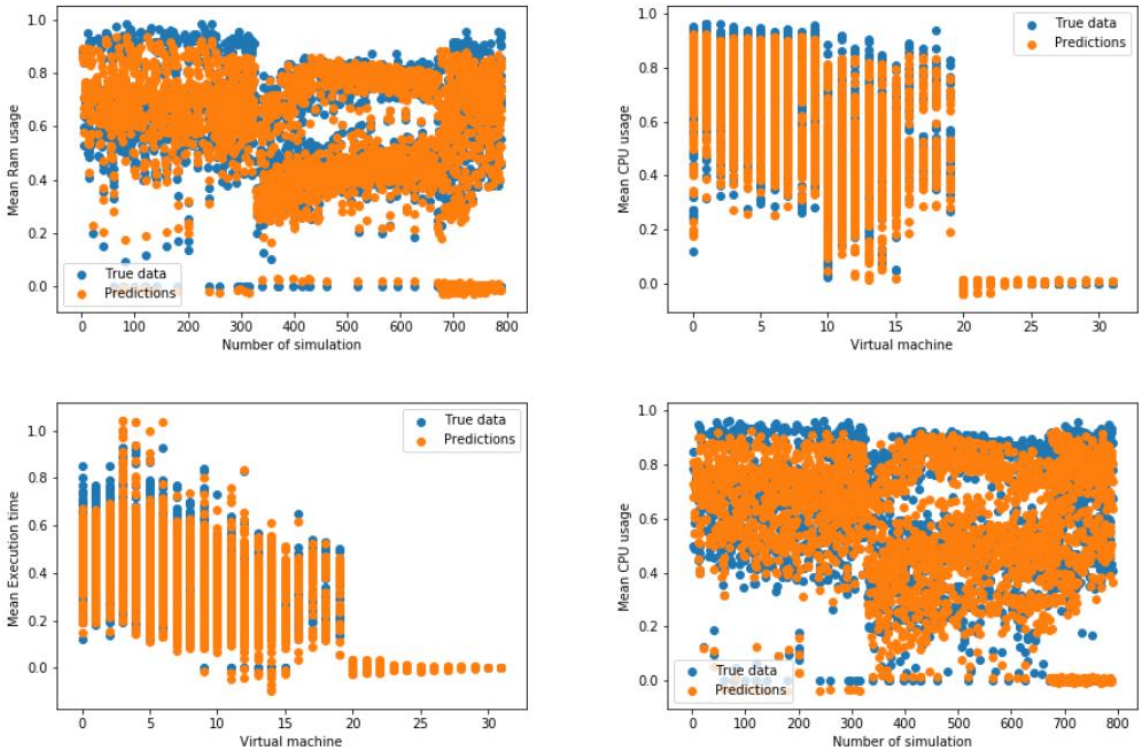

Fig 4.10 Multi target prediction with 250 simulations

The annex C has been represented into the fig 4.11 indicates the evolution in every case of the error and the accuracy with every algorithm, the R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, continuing with the graph we can see that the performance of the algorithms chosen is relevant in terms of accuracy, being the linear regression and the multi target regression those that have the minimum error and more probability of predict correctly.

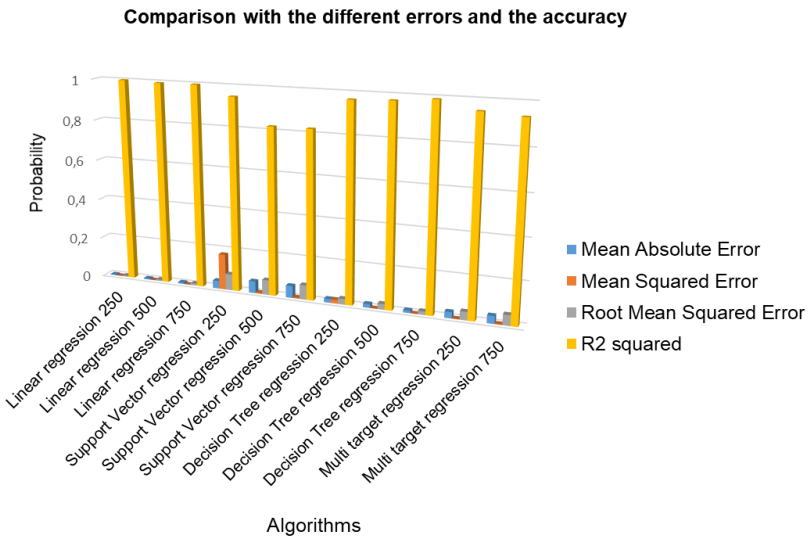

Fig 4.11 Comparative table about each technique performance

4.3 Environmental impact

In this section we are going to analyse the final impact that this project can produce into different environments. On the one hand, we have to highlight that this project has been developed from the point of view of softwarization inside the 5G network, so the maximum impact could be the server's necessity, where they will carry out all the performance.

On the other hand, we should see beyond this, the main reason is the deployment of new datacenters and base stations, in order to satisfy the increase in demand. First, regarding to datacenters, the increment of huge amounts of data will make necessary to deploy new datacenters, obviously where, this huge amount of metrics must be computed. Here it comes a relevant impact where there will be necessary a planned infrastructure where the servers can be installed.

Meanwhile these metrics have to come from several base stations implemented in the streets, where we can see a clear environmental impact. The 5G and the implementation of machine learning algorithms will need a real time streaming of hundreds of parameters in each moment, the multi massive antennas implementation needed for making MIMO (Multiple-input Multiple-output) technology a reality will have a clear impact in the streets, as well as, the new base stations where it will store data and send it to the closest datacenter.

Consequently, our project will not have in a direct way a clear impact in the environment, however and despite of being indirect impact, it connects directly with the direct impact produce by the implementation and deployment of the 5G infrastructure, like the new fibre infrastructure connection in order to connect the base stations with the core network.

Finally, in order to analyse the total impact, there is one more field, which we have to analysed, called the impact in the social environment, where the people life is changed by the action of our project, in this case the 5G has searched to make easier the life and connected in a better way the whole network of existing devices, consequently, the most important impact in this part is an advantage, but they should buy devices adapted to this new generation.

CHAPTER 5. CONCLUSIONS AND FUTURE WORK.

5.1 Conclusions

In this thesis, we addressed the problem of studying different techniques for being applied into the 5G optical networks. In particular, we investigated how the part of cognition inside of the structure of 5G works and the proposal of an Autonomic Network Management based on Machine Learning as a key technology for 5G networks to reach the vision of automated management of telecoms network infrastructures. We have introduced the part of Smart Cognet related to the collection of data streams and their analysis in order to apply different ML techniques, giving a simple example of how the actuator part and policies can be introduced.

We have solved the problem related with the data collector by means of the implementation of a simulator of cloud environments which could represents the real situation into the 5G structure due to currently it does not exist any repository with the needed data.

A discussion on different ML algorithms performance has been provided as well as the different solutions in order to deal with the data and provided the most reliable solution. In particular, we have proposed the possible solution which can suit better with the structure of the network.

The main focus of our thesis was on the optimization itself. A new approach based on estimation of ML algorithms was introduced for solving cognition problem, acting on the real application which can be extracted as well as looking into the best option for the satisfaction of services request.

Finally, another contribution relies in the parallelization of the different prediction by the introduction of multi target regression which gives a complete view of the future parameters and how could be the performance of the network being able to adapt the conditions of the whole network to possible increments in the demand of services in punctual moments. From an experimental point of view, our contribution lies in the comparison of different machine learning techniques which could be apply in the Cognet structure inside of the 5G network.

5.2 Future work

Many different adaptations, tests, and experiments have been left for the future due to lack of time as well as the data problem which has been common along the project (i.e. the experiments with real data are usually very time consuming, requiring even days to finish a single run)

From this project it can be extract several future works like the simulation with real data and the monitoring in real time, furthermore one of the most complicated

parts in the Cognet network have not been implemented which it counts with actuators and policies management, indeed without this part we cannot know if the data predicted has a real impact into the network. Therefore, this opens a new opportunity of investigation, applying directly the algorithms and defining the policies but if there is not real data could bring too many problems.

Obviously, the use of other types of ML techniques and data treatment functions could be investigated, Concerning the results for our Cognet structure (data collector, smart engine and policy manager), we can also expect to improve them by having better data, with more attributes and even going deeper inside this structure explaining the different parts and developed them.

Annexes

Annexe A Cloudlet dataset

Cloudlet	Status	DC	Host	Host PEs	VM	VM PEs	CloudletLen	CloudletPEs	StartTime	FinishTime	ExecTime
44	SUCCESS	1	1	32	10	3	4888	1	18	28	10
45	SUCCESS	1	1	32	11	3	4400	1	20	28	9
46	SUCCESS	1	1	32	12	3	4000	1	22	30	9
42	SUCCESS	1	0	32	8	3	6285	1	14	32	18
41	SUCCESS	1	0	32	7	3	7333	1	12	33	21
47	SUCCESS	1	1	32	13	3	3666	1	24	33	9
48	SUCCESS	1	1	32	14	3	3384	1	26	34	9
43	SUCCESS	1	0	32	9	3	5500	1	16	35	20
49	SUCCESS	1	1	32	15	3	3142	1	28	37	9
50	SUCCESS	1	1	32	16	3	2933	1	30	38	8
40	SUCCESS	1	0	32	6	3	8800	1	10	39	29
39	SUCCESS	1	0	32	5	3	11000	1	8	41	34
53	SUCCESS	1	0	32	2	3	2444	1	36	41	6
51	SUCCESS	1	0	32	0	3	2750	1	32	46	15
54	SUCCESS	1	0	32	3	3	2315	1	38	46	9
52	SUCCESS	1	0	32	1	3	2588	1	34	47	13
56	SUCCESS	1	0	32	5	3	2095	1	42	48	6
55	SUCCESS	1	0	32	4	3	2200	1	40	49	9
57	SUCCESS	1	0	32	6	3	2000	1	44	51	7
38	SUCCESS	1	0	32	4	3	14666	1	6	52	46

Annexe B System dataset at every time

```
Vm_id,Host_Id,Time,CPU_usage,Ram_Usage,Bw_Usage
0, 0, 0.1, 79.06954893876078, 0.0,0.0
1, 0, 0.1, 65.73365304305594, 0.0,0.0
2, 0, 0.1, 25.224994140987977, 0.0,0.0
3, 0, 0.1, 70.75571728074344, 0.0,0.0
4, 0, 0.1, 40.21866397504017, 0.0,0.0
5, 0, 0.1, 49.05344783716264, 0.0,0.0
6, 0, 0.1, 23.075037897456607, 0.0,0.0
7, 0, 0.1, 76.10355222066865, 0.0,0.0
8, 0, 0.1, 56.49906916224078, 0.0,0.0
9, 0, 0.1, 33.50323075470922, 0.0,0.0
10, 1, 0.1, 68.29141544793157, 0.0,0.0
11, 1, 0.1, 39.042955356446, 0.0,0.0
12, 1, 0.1, 44.316429127340136, 0.0,0.0
13, 1, 0.1, 72.80795114956632, 0.0,0.0
14, 1, 0.1, 87.8515867710081, 0.0,0.0
15, 1, 0.1, 54.22742292742304, 0.0,0.0
16, 1, 0.1, 75.39550672834235, 0.0,0.0
0, 0, 1.1, 0.0, 100.0,33.300000000000004
1, 0, 1.1, 32.33723370491745, 100.0,33.300000000000004
2, 0, 1.1, 36.19966664736648, 75.732421875,22.2
3, 0, 1.1, 38.00301798394526, 100.0,22.2
4, 0, 1.1, 9.760713112534274, 56.8359375,22.2
5, 0, 1.1, 34.55116895013897, 86.23046875,22.2
6, 0, 1.1, 10.147476247986301, 99.0234375,22.2
7, 0, 1.1, 9.644423713483425, 100.0,22.2
8, 0, 1.1, 24.74452595380162, 83.251953125,22.2
9, 0, 1.1, 58.411697424139234, 94.140625,22.2
```

```
N_simul,Ram_host,Storage_host,Vm_Ram,Num_cloudlets,storage_v
1,32768,2000000,2048,25,500000,4,20000,2
2,32768,2000000,2048,30,400000,5,20000,2
3,32768,2000000,2048,35,333333,6,20000,2
4,32768,2000000,2048,40,285714,7,20000,2
5,32768,2000000,2048,45,250000,8,20000,2
6,32768,2000000,2048,50,222222,9,20000,2
7,32768,2000000,2048,55,200000,10,20000,2
8,32768,2000000,2048,60,181818,11,20000,2
9,32768,2000000,2048,65,166666,12,20000,2
10,32768,2000000,2048,70,153846,13,20000,2
11,32768,2000000,2048,75,142857,14,20000,2
12,32768,2000000,2048,80,133333,15,20000,2
13,32768,2000000,2048,85,125000,16,20000,2
14,32768,2000000,2048,90,117647,17,20000,2
15,32768,2000000,2048,95,125000,16,20000,2
16,32768,2000000,2048,100,105263,19,20000,2
17,32768,2000000,2048,105,125000,16,20000,2
```

Annexe C Algorithm accuracy

Algorithm	Mean Absolute Error	Mean Squared Error	Root Mean Squared Error	R2 squared
Linear regression 250	0,0036	0,000020866	0,004567	0,99931
Linear regression 500	0,00457	0,00006047	0,00667	0,99457
Linear regression 750	0,00652	0,00009005	0,00948	0,9969
Support Vector regression 250	0,03894	0,17676	0,0801	0,95009
Support Vector regression 500	0,05953	0,00539	0,0734	0,8225
Support Vector regression 750	0,06002	0,00514	0,07174	0,82402
Decision Tree regression 250	0,01877	0,01877	0,0281	0,9679
Decision Tree regression 500	0,01845	0,00077	0,02787	0,9744
Decision Tree regression 750	0,012183	0,00028	0,017005	0,99011
Multi target regression 250	0,03053	0,00168	0,04104	0,94931
Multi target regression 750	0,03804	0,00266	0,0516	0,9363

Annexe D CloudSim code in Java

```
2
3  package org.cloudsimplus.examples;
4
5  import org.cloudbus.cloudsim.allocationpolicies.VmAllocationPolicySimple;
6  import org.cloudbus.cloudsim.brokers.DatacenterBroker;
7  import org.cloudbus.cloudsim.brokers.DatacenterBrokerSimple;
8  import org.cloudbus.cloudsim.cloudlets.Cloudlet;
9  import org.cloudbus.cloudsim.cloudlets.CloudletSimple;
10 import org.cloudbus.cloudsim.core.CloudSim;
11 import org.cloudbus.cloudsim.core.Simulation;
12 import org.cloudbus.cloudsim.datacenters.Datacenter;
13 import org.cloudbus.cloudsim.datacenters.DatacenterSimple;
14 import org.cloudbus.cloudsim.hosts.Host;
15 import org.cloudbus.cloudsim.hosts.HostSimple;
16 import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
17 import org.cloudbus.cloudsim.provisioners.ResourceProvisionerSimple;
18 import org.cloudbus.cloudsim.resources.Bandwidth;
19 import org.cloudbus.cloudsim.resources.Pe;
20 import org.cloudbus.cloudsim.resources.PeSimple;
21 import org.cloudbus.cloudsim.resources.Processor;
22 import org.cloudbus.cloudsim.resources.Ram;
23 import org.cloudbus.cloudsim.resources.ResourceManageable;
24 import org.cloudbus.cloudsim.schedulers.cloudlet.CloudletSchedulerTimeShared;
25 import org.cloudbus.cloudsim.schedulers.vm.VmSchedulerTimeShared;
26 import org.cloudbus.cloudsim.util.MathUtil;
27 import org.cloudbus.cloudsim.utilizationmodels.UtilizationModel;
28 import org.cloudbus.cloudsim.utilizationmodels.UtilizationModelDynamic;
29 import org.cloudbus.cloudsim.utilizationmodels.UtilizationModelFull;
30 import org.cloudbus.cloudsim.utilizationmodels.UtilizationModelPlanetLab;
31 import org.cloudbus.cloudsim.utilizationmodels.UtilizationModelStochastic;
32 import org.cloudbus.cloudsim.vms.Vm;
33 import org.cloudbus.cloudsim.vms.VmSimple;
34 import org.cloudsimplus.autoscaling.HorizontalVmScaling;
35 import org.cloudsimplus.autoscaling.VerticalVmScaling;
36 import org.cloudsimplus.autoscaling.VerticalVmScalingSimple;
37 import org.cloudsimplus.autoscaling.resources.ResourceScaling;
38 import org.cloudsimplus.autoscaling.resources.ResourceScalingGradual;
39 import org.cloudsimplus.autoscaling.resources.ResourceScalingInstantaneous;
40 import org.cloudsimplus.builders.tables.CloudletsTableBuilder;
41 import org.cloudsimplus.builders.tables.CsvTable;
42 import org.cloudsimplus.builders.tables.TextTableColumn;
43 import org.cloudsimplus.listeners.EventInfo;
44 import org.cloudsimplus.listeners.EventListener;
45
46 import java.util.ArrayList;
47 import java.util.Arrays;
48 import java.util.Comparator;
49 import java.util.HashMap;
50 import java.util.List;
51 import java.util.Map;
52 import java.util.Set;
53 import java.util.TreeMap;
54 import java.util.stream.Collectors;
55
56 import static java.util.Comparator.comparingDouble;
57
58 import java.io.BufferedReader;
59 import java.io.BufferedWriter;
60 import java.io.File;
```

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintStream;
import java.io.PrintWriter;

public class Cognet_creatingdata {

    private static final int SCHEDULING_INTERVAL = 1;
    private static final int HOSTS = 2;
    private static final int HOST_PES = 32;
    private static int VMS = 0;
    private static final int VM_PES = 2;
    private static int VM_RAM = 2048;
    private static int CLOUDLETS = 25;
    private static int CLOUDLETS_INITIAL_LENGTH = 20000;
    public static long ram_host =32768; //in Megabytes
    public static long bw_host = 10000; //in Megabits/s
    public static long storage_host = 2000000; //in Megabytes
    public static long bw_vm = 1000; //in Megabits/s
    public static long storage_vm = 10000; //in Megabytes
    public static int i;
    //private static final long CLOUDLET_LENGTHS[] = {20_00};//, 30_00, 40_00, 50_00, 10_00};

    private final CloudSim simulation;
    private DatacenterBroker broker0;
    private List<Host> hostList;
    private List<Vm> vmList;
    private List<Cloudlet> cloudletList;
    private int createsVms;
    static int numsimul=0;

    public static void main(String[] args) {
        numsimul=consultlastsimul();
        System.out.println("este es el que vale" + numsimul);

        printinfile("N_simul,Ram_host,Storage_host,Vm_Ram,Num_cloudlets,storage_vm,N_VMS,CLOUDLETS_LENGTH",tru
        e);
        for (CLOUDLETS_INITIAL_LENGTH=10000; (CLOUDLETS_INITIAL_LENGTH*CLOUDLETS)<(storage_host*HOSTS);
        CLOUDLETS_INITIAL_LENGTH= CLOUDLETS_INITIAL_LENGTH+2500) {
        for (i=25; i<35; i++) {
            numsimul++;
            if((VM_RAM*VMS<=HOSTS*ram_host)  && ((VM_PES*VMS) <= HOST_PES) ) { //VM_PES*(i-2)
                VMS=i-1;
            } else if (HOST_PES >= VM_PES*(HOSTS*ram_host)/VM_RAM) {
                VMS=(int) ((HOSTS*ram_host)/VM_RAM);   //Maximum value posible without exceding the HOST RAM
            } else {
                VMS=HOST_PES/VM_PES; //Maximum number of VM that can be created
             }

        storage_vm=storage_host/VMS;  //The storage of every VM will change depending on the number of VM in
        order to adapt
```

```
        order to adapt
        CLOUDLETS=i*3;
        //ram_host=4096*(1+i/5);
        //VM_RAM = 2048*(1+i/5);
        System.out.println("este es el que vale 2" + numsimul);
        printinfile(numsimul +","+ram_host +","+ storage_host +","+ VM_RAM + ","
        + CLOUDLETS +","+ storage_vm + "," + VMS + "," + CLOUDLETS_INITIAL_LENGTH+","+HOSTS,true);

        new Cognet_creatingdata();

        }

        }
    }

    private Cognet_creatingdata() {


        hostList = new ArrayList<>(HOSTS);
        vmList = new ArrayList<>(VMS);
        cloudletList = new ArrayList<>(CLOUDLETS);//CLOUDLET_LENGTHS.length);
        simulation = new CloudSim();
        printinfile("Vm_id,Host_Id,Time,CPU_usage,Ram_Usage,Bw_Usage",false);
        simulation.addOnClockTickListener(this::onClockTickListener);
        createDatacenter();
        broker0 = new DatacenterBrokerSimple(simulation);

        vmList.addAll(createListOfScalableVms(VMS));
        createCloudletListsWithDifferentDelays();
        broker0.submitVmList(vmList);
        broker0.submitCloudletList(cloudletList);
        simulation.start();
        printSimulationResults();
        printcsv();

    }


    private void onClockTickListener(EventInfo evt) {


        vmList.forEach(vm -> {

            String pal3= String.valueOf(evt.getTime());
            String pal0=String.valueOf(vm.getCpuPercentUsage()*100.0);
            String pal1=String.valueOf((vm.getRam().getPercentUtilization() * 100.0));
            String pal2=String.valueOf(vm.getBw().getPercentUtilization()*100);
            String linea=pal3 + ", " +pal0+", "+pal1+","+ pal2;
            if (vm.getHost().getId()==-1)
                printinfile(""+ vm.getId() + ", " + 1 + ", "+linea,false );
            else
            printinfile(""+ vm.getId() + ", " + vm.getHost().getId()+ ", "+linea,false);
        });
    }
```

```java
    private void printSimulationResults() {
        final List<Cloudlet> finishedCloudlets = broker0.getCloudletFinishedList();
        final Comparator<Cloudlet> sortByVmId = comparingDouble(c -> c.getVm().getId());
        final Comparator<Cloudlet> sortByStartTime = comparingDouble(Cloudlet::getExecStartTime);
        finishedCloudlets.sort(sortByVmId.thenComparing(sortByStartTime));
        //new CloudletsTableBuilder(finishedCloudlets).build();

    }

    /**
     * Creates a Datacenter and its Hosts.
     */
    private void createDatacenter() {
        for (int i = 0; i < HOSTS; i++) {
            hostList.add(createHost());
        }

        Datacenter dc0 = new DatacenterSimple(simulation, hostList, new VmAllocationPolicySimple());
        dc0.setSchedulingInterval(SCHEDULING_INTERVAL);
    }

    private Host createHost() {
        List<Pe> peList = new ArrayList<>(HOST_PES);
        for (int i = 0; i < HOST_PES; i++) {
            peList.add(new PeSimple(1000, new PeProvisionerSimple()));
        }


        final int id = hostList.size();
        return new HostSimple(ram_host, bw_host, storage_host, peList)
            .setRamProvisioner(new ResourceProvisionerSimple())
            .setBwProvisioner(new ResourceProvisionerSimple())
            .setVmScheduler(new VmSchedulerTimeShared());
    }


    private List<Vm> createListOfScalableVms(final int numberOfVms) {
        List<Vm> newList = new ArrayList<>(numberOfVms);
        for (int i = 0; i < numberOfVms; i++) {
            Vm vm = createVm();
            vm.setPeVerticalScaling(createVerticalPeScaling(vm));
            newList.add(vm);
        }

        return newList;
    }


    private Vm createVm() {
        final int id = createsVms++;

        final Vm vm = new VmSimple(id, 1000, VM_PES)
            .setRam(VM_RAM).setBw(bw_vm).setSize(storage_vm)
            .setCloudletScheduler(new CloudletSchedulerTimeShared());

        vm.getUtilizationHistory().enable();
        return vm;
    }
```

```
220
221 ▼    private void printcsv() {
222 ▼    try {
223          CsvTable csv = new CsvTable();
224          csv.setPrintStream(new PrintStream(new java.io.File("C:\\Users\\delas\\Documents\\Master\\TFM-MASTER
             _THESIS\\Datasets\\cloudsim\\simulation"+ numsimul +".csv")));
225          new CloudletsTableBuilder(broker0.getCloudletFinishedList(), csv).build();
226 ▼        } catch (IOException e) {
227          System.err.println(e.getMessage());
228      }
229      }
230
231
232 ▼    /**
233       * Creates a {@link VerticalVmScaling} for scaling VM's CPU when it's under or overloaded.
234       *
235       * <p>Realize the lower and upper thresholds are defined inside this method by using
236       * references to the methods {@link #lowerCpuUtilizationThreshold(Vm)}
237       * and {@link #upperCpuUtilizationThreshold(Vm)}.
238       * These methods enable defining thresholds in a dynamic way
239       * and even different thresholds for distinct VMs.
240       * Therefore, it's a powerful mechanism.
241       * </p>
242       *
243       * <p>
244       * However, if you are defining thresholds in a static way,
245       * and they are the same for all VMs, you can use a Lambda Expression
246       * like below, for instance, instead of creating a new method that just returns a constant value:<br>
247       * {@code verticalCpuScaling.setLowerThresholdFunction(vm -> 0.4);}
248       * </p>
249       *
250       * @see #createListOfScalableVms(int)
251       *
252       */
253 ▼    private VerticalVmScaling createVerticalPeScaling(Vm vm) {
254          //The percentage in which the number of PEs has to be scaled
255          final double scalingFactor = 0.1;
256          VerticalVmScalingSimple verticalCpuScaling = new VerticalVmScalingSimple(Processor.class,
             scalingFactor);
257          VerticalVmScalingSimple verticalRamScaling = new VerticalVmScalingSimple(Ram.class, 0.1);
258
259 ▼        /* By uncommenting the line below, you will see that, instead of gradually
260           * increasing or decreasing the number of PEs, when the scaling object detects
261           * the CPU usage is above or below the defined thresholds,
262           * it will automatically calculate the number of PEs to add/remove to
263           * move the VM from the over or underload condition.
264          */
265          //verticalCpuScaling.setResourceScaling(new ResourceScalingInstantaneous());
266
267 ▼        /** Different from the commented line above, the line below implements a ResourceScaling using a
             Lambda Expression.
268           * It is just an example which scales the resource twice the amount defined by the scaling factor
269           * defined in the constructor.
270           *
271           * Realize that if the setResourceScaling method is not called, a ResourceScalingGradual will be used,
```

```
272         * which scales the resource according to the scaling factor.
273         * The lower and upper thresholds after this line can also be defined using a Lambda Expression.
274         *
275         * So, here we are defining our own {@link ResourceScaling} instead of
276         * using the available ones such as the {@link ResourceScalingGradual}
277         * or {@link ResourceScalingInstantaneous}.
278         */
279        //vm.setRamVerticalScaling(verticalRamScaling);
280        verticalCpuScaling.setResourceScaling(vs -> 2*vs.getScalingFactor()*vs.getAllocatedResource());
281
282        verticalCpuScaling.setLowerThresholdFunction(this::lowerCpuUtilizationThreshold);
283        verticalCpuScaling.setUpperThresholdFunction(this::upperCpuUtilizationThreshold);
284
285        return verticalCpuScaling;
286    }
287
288    /**
289     * Defines the minimum CPU utilization percentage that indicates a Vm is underloaded.
290     * This function is using a statically defined threshold, but it would be defined
291     * a dynamic threshold based on any condition you want.
292     * A reference to this method is assigned to each Vertical VM Scaling created.
293     *
294     * @param vm the VM to check if its CPU is underloaded.
295     *        <b>The parameter is not being used internally, which means the same
296     *        threshold is used for any Vm.</b>
297     * @return the lower CPU utilization threshold
298     * @see #createVerticalPeScaling()
299     */
300    private double lowerCpuUtilizationThreshold(final Vm vm) {
301        return 0.3;
302    }
303
304    private double lowerRamUtilizationThreshold(Vm vm) {
305        return 0.5;
306    }
307
308
309    /**
310     * Defines a dynamic CPU utilization threshold that indicates a Vm is overloaded.
311     * Such a threshold is the maximum CPU a VM can use before requesting vertical CPU scaling.
312     * A reference to this method is assigned to each Vertical VM Scaling created.
313     *
314     * <p>The dynamic upper threshold is defined as 20% above the mean (mean * 1.2),
315     * if there are at least 10 CPU utilization history entries.
316     * That means if the CPU utilization of a VM is 20% above its mean
317     * CPU utilization, it indicates the VM is overloaded.
318     * If there aren't enough history entries,
319     * it defines a static threshold as 70% of CPU utilization.</p>
320     *
321     * @param vm the VM to check if its CPU is overloaded.
322     *        The parameter is not being used internally, that means the same
323     *        threshold is used for any Vm.
324     * @return the upper dynamic CPU utilization threshold
325     * @see #createVerticalPeScaling()
```

```
326      *
327      */
328
329      private double upperCpuUtilizationThreshold(final Vm vm) {
330          final List<Double> history = new ArrayList<>(vm.getUtilizationHistory().getHistory().values());
331          return history.size() > 10 ? MathUtil.median(history) * 1.2 : 0.7;
332      }
333
334      /**
335       * Creates lists of Cloudlets to be submitted to the broker with different delays,
336       * simulating their arrivals at different times.
337       * Adds all created Cloudlets to the {@link #cloudletList}.
338       */
339
340
341
342
343
344
345      private void createCloudletListsWithDifferentDelays() {
346          final int initialCloudletsNumber = (int)(CLOUDLETS/2.5);
347          final int remainingCloudletsNumber = CLOUDLETS-initialCloudletsNumber;
348          //Creates a List of Cloudlets that will start running immediately when the simulation starts
349          for (int i = 0; i < initialCloudletsNumber; i++) {
350          /*  for (long length: CLOUDLET_LENGTHS) {
351                  int j = (int)length;*/
352              cloudletList.add(createCloudlet(CLOUDLETS_INITIAL_LENGTH+(i*1000), 2));//+(i*1000), 2
353              }
354
355
356          /*
357           * Creates several Cloudlets, increasing the arrival delay and decreasing
358           * the length of each one.
359           * The progressing delay enables CPU usage to increase gradually along the arrival of
360           * new Cloudlets (triggering CPU up scaling at some point in time).
361           *
362           * The decreasing length enables Cloudlets to finish in different times,
363           * to gradually reduce CPU usage (triggering CPU down scaling at some point in time).
364           *
365           * Check the logs to understand how the scaling is working.
366          */
367          for (int i = 1; i <= remainingCloudletsNumber; i++) {
368              cloudletList.add(createCloudlet(CLOUDLETS_INITIAL_LENGTH*2/i, 1,i*2));
369          }
370      }
371
372      /**
373       * Creates a single Cloudlet with no delay, which means the Cloudlet arrival time will
374       * be zero (exactly when the simulation starts).
375       *
376       * @param length the Cloudlet length
377       * @param numberOfPes the number of PEs the Cloudlets requires
378       * @return the created Cloudlet
379       */
380      private Cloudlet createCloudlet( long length, final int numberOfPes) {
381          return createCloudlet(length, numberOfPes, 0);
382      }
```

```
383
384     /**
385      * Creates a single Cloudlet.
386      *
387      * @param length the length of the cloudlet to create.
388      * @param numberOfPes the number of PEs the Cloudlets requires.
389      * @param delay the delay that defines the arrival time of the Cloudlet at the Cloud infrastructure.
390      * @return the created Cloudlet
391      */
392     private Cloudlet createCloudlet(final long length, final int numberOfPes, final double delay) {
393         /*
394         Since a VM PE isn't used by two Cloudlets at the same time,
395         the Cloudlet can used 100% of that CPU capacity at the time
396         it is running. Even if a CloudletSchedulerTimeShared is used
397         to share the same VM PE among multiple Cloudlets,
398         just one Cloudlet uses the PE at a time.
399         Then it is preempted to enable other Cloudlets to use such a VM PE.
400          */
401
402
403         final UtilizationModel utilizationCpu = new UtilizationModelStochastic();
404         UtilizationModelDynamic bwUtilizationModel = new UtilizationModelDynamic(0.1);
405         bwUtilizationModel.setUtilizationUpdateFunction(this::utilizationUpdate);
406         final UtilizationModel utilizationModelDynamic = new UtilizationModelDynamic(1.5/CLOUDLETS);
407         final UtilizationModel RamUtilizationModel = new UtilizationModelStochastic();
408         /**
409          * Since BW e RAM are shared resources that don't enable preemption,
410          * two Cloudlets can't use the same portion of such resources at the same time
411          * (unless virtual memory is enabled, but such a feature is not available in simulation).
412          * This way, the total capacity of such resources is being evenly split among created Cloudlets.
413          * If there are 10 Cloudlets, each one will use just 10% of such resources.
414          * This value can be defined in different ways, as you want. For instance, some Cloudlets
415          * can require more resources than other ones.
416          * To enable that, you would need to instantiate specific {@link UtilizationModelDynamic} for each
             Cloudlet,
417          * use a {@link UtilizationModelStochastic} to define resource usage randomly,
418          * or use any other {@link UtilizationModel} implementation.
419         */
420
421         Cloudlet cl = new CloudletSimple(length, numberOfPes);
422         cl.setFileSize(1024)
423           .setOutputSize(1024)
424           .setUtilizationModelBw( bwUtilizationModel)
425           .setUtilizationModelRam(RamUtilizationModel)
426           .setUtilizationModelCpu(utilizationCpu)
427           .setSubmissionDelay(delay);
428         return cl;
429     }
430
431     private double utilizationUpdate(UtilizationModelDynamic utilizationModel) {
432         return utilizationModel.getUtilization() + utilizationModel.getTimeSpan() * 0.01;
433     }
434         FileWriter fichero = null;
435         PrintWriter pw = null;
436         private static int consultlastsimul() {
```

```
436         private static int consultlastsimul() {
437             int lastnumsimul=0;
438             File tempFile = new File("C:\\Users\\delas\\Documents\\Master\\TFM-MASTER
                _THESIS\\Datasets\\cloudsim\\DC_Features.txt");
439                 if (tempFile.exists()) {
440                  lastnumsimul = readinfile();
441                 }
442                 return lastnumsimul;
443         }
444 
445         private static void printinfile(String Linea,Boolean datos ) {
446             String dir,lastline="",st,dircheck;
447 
448             BufferedWriter bw = null;
449             FileWriter fw = null;
450 
451             System.out.println("Ultima simulacion " + numsimul);
452                   try {
453 
454                       if (datos == false) {
455                           dir = "C:\\Users\\delas\\Documents\\Master\\TFM-MASTER
                            _THESIS\\Datasets\\cloudsim\\data" + numsimul +".txt";
456                       } else  {
457                           dir="C:\\Users\\delas\\Documents\\Master\\TFM-MASTER
                            _THESIS\\Datasets\\cloudsim\\DC_Features.txt";
458                       }
459                         File file = new File(dir);
460                        if (!file.exists()) {
461                           file.createNewFile();
462                       } else if (Linea.contains("N_simul")) {
463                           return;
464                       }
465                       fw = new FileWriter(file.getAbsoluteFile(), true);
466                       bw = new BufferedWriter(fw);
467                       bw.write(Linea);
468                       bw.newLine();
469 
470                   } catch (IOException e) {
471                       e.printStackTrace();
472                   } finally {
473                       try {
474                                       //Cierra instancias de FileWriter y BufferedWriter
475                           if (bw != null)
476                               bw.close();
477                           if (fw != null)
478                               fw.close();
479                       } catch (IOException ex) {
480                           ex.printStackTrace();
481                       }
482                   }
483                   return;
484 
485            }
486 
487         private static int readinfile() {
488             BufferedReader br=null;
489             int lastsimul=0;
490 
491             try {
492                 String sCurrentLine;
493                br = new BufferedReader(new FileReader("C:\\Users\\delas\\Documents\\Master\\TFM-MASTER
```

```
493                 br = new BufferedReader(new FileReader("C:\\Users\\delas\\Documents\\Master\\TFM-MASTER
                    _THESIS\\Datasets\\cloudsim\\DC_Features.txt"));
494                   String lastLine = "";
495
496                   while ((sCurrentLine = br.readLine()) != null)
497                   {
498
499                       lastLine = sCurrentLine;
500                   }
501                   System.out.println("Ultima linea "+ lastLine);
502                   String partes[] = lastLine.split(",");
503                   List<String> list = Arrays.asList(partes);
504
505                   if(list.contains("N_simul")) {
506                       return 0;
507                   }else {
508                   System.out.println(partes[0]);
509                   lastsimul= Integer.valueOf(partes[0]);
510                   }
511                   //Character.toString(char);
512                   System.out.println("Este es el numero que aparece " +lastLine.charAt(0));
513                   System.out.println("Este es el numero que aparece " + lastsimul);
514               } catch (IOException e) {
515                   e.printStackTrace();
516               } finally {
517                   try {
518                       if (br != null)br.close();
519                   } catch (IOException ex) {
520                       ex.printStackTrace();
521                   }
522               }
523           return lastsimul;
524            }
525   }
```

```java

    /**
     * Creates a single Cloudlet.
     *
     * @param length the length of the cloudlet to create.
     * @param numberOfPes the number of PEs the Cloudlets requires.
     * @param delay the delay that defines the arrival time of the Cloudlet at the Cloud infrastructure.
     * @return the created Cloudlet
     */
    private Cloudlet createCloudlet(final long length, final int numberOfPes, final double delay) {
        /*
        Since a VM PE isn't used by two Cloudlets at the same time,
        the Cloudlet can used 100% of that CPU capacity at the time
        it is running. Even if a CloudletSchedulerTimeShared is used
        to share the same VM PE among multiple Cloudlets,
        just one Cloudlet uses the PE at a time.
        Then it is preempted to enable other Cloudlets to use such a VM PE.
         */


        final UtilizationModel utilizationCpu = new UtilizationModelStochastic();
        UtilizationModelDynamic bwUtilizationModel = new UtilizationModelDynamic(0.1);
        bwUtilizationModel.setUtilizationUpdateFunction(this::utilizationUpdate);
        final UtilizationModel utilizationModelDynamic = new UtilizationModelDynamic(1.5/CLOUDLETS);
        final UtilizationModel RamUtilizationModel = new UtilizationModelStochastic();
        /**
         * Since BW e RAM are shared resources that don't enable preemption,
         * two Cloudlets can't use the same portion of such resources at the same time
         * (unless virtual memory is enabled, but such a feature is not available in simulation).
         * This way, the total capacity of such resources is being evenly split among created Cloudlets.
         * If there are 10 Cloudlets, each one will use just 10% of such resources.
         * This value can be defined in different ways, as you want. For instance, some Cloudlets
         * can require more resources than other ones.
         * To enable that, you would need to instantiate specific {@link UtilizationModelDynamic} for each
         Cloudlet,
         * use a {@link UtilizationModelStochastic} to define resource usage randomly,
         * or use any other {@link UtilizationModel} implementation.
        */

        Cloudlet cl = new CloudletSimple(length, numberOfPes);
        cl.setFileSize(1024)
          .setOutputSize(1024)
          .setUtilizationModelBw( bwUtilizationModel)
          .setUtilizationModelRam(RamUtilizationModel)
          .setUtilizationModelCpu(utilizationCpu)
          .setSubmissionDelay(delay);
        return cl;
    }

    private double utilizationUpdate(UtilizationModelDynamic utilizationModel) {
        return utilizationModel.getUtilization() + utilizationModel.getTimeSpan() * 0.01;
    }
        FileWriter fichero = null;
        PrintWriter pw = null;
        private static int consultlastsimul() {
```

Annexe E: Python code

```
import pandas as pd
import numpy as np
import os
import sys
import scipy
import matplotlib
import sklearn
import csv,operator
from sklearn import metrics
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn import svm
import seaborn as sns
import pandas as pd
import numpy as np
import os
import sys
import scipy
import matplotlib
import sklearn
import csv,operator


def txt_to_csv(numsimul):
   with open("cloudsim\\data"+ str(numsimul)+".txt", 'r') as in_file:
        stripped = (line.strip() for line in in_file)
        lines = (line.split(",") for line in stripped if line)
        with open('cloudsim\\data'+ str(numsimul)+'.csv', 'w') as out_
file:
            writer = csv.writer(out_file)
            writer.writerows(lines)


def parameters_to_csv():
   with open("cloudsim\\DC_Features.txt", 'r') as in_file:
        stripped = (line.strip() for line in in_file)
        lines = (line.split(",") for line in stripped if line)
        with open('cloudsim\\DC_Features.csv', 'w') as out_file:
            writer = csv.writer(out_file)
            writer.writerows(lines)

def make_tables(df):

    dfObj = pd.DataFrame()
    for j in range(0,df['Host_Id'].max()+1):
        for i in range(0,df['Vm_id'].max()+1):
            df2=df.loc[(df['Vm_id'] == i) & (df['Host_Id'] == j)].desc
ribe()
```

```
            if (df2.isnull().values.any()):
                continue
            else:
                dfObj = dfObj.append({'Host_id':int(j) ,'Vm_id':int(i)
, 'Mean_CpuUsage':df2.iloc[1][3] , 'Max_CpuUsage':df2.iloc[7][3],
                'Min_CpuUsage':df2.iloc[3][4],'Mean_RamUsage':df2.iloc
[1][4],'Max_RamUsage':df2.iloc[7][4],
                'Max_BwUsage':df2.iloc[7][5] ,'Mean_BwUsage': df2.iloc
[1][5],'Min_RamUsage':df2.iloc[3][4]}, ignore_index=True)
    return  dfObj

def make_tables_simulation(dfcloud):
    dfObjcloud=pd.DataFrame()

    for j in range(0,dfcloud['Host'].max()+1):
        for i in range(0,dfcloud['VM'].max()+1):
            dfcloud2=dfcloud[(dfcloud['VM'] == i) & (dfcloud['Host'] =
=j)].describe()

            if (dfcloud2.isnull().values.any()):
                continue
            else:
                dfObjcloud = dfObjcloud.append({'Host_id':int(j),
                'Vm_id':int(i) , 'Mean_FinishTime':dfcloud2.iloc[1][9]
, 'Mean_ExecTime':dfcloud2.iloc[1][10], 'NumCloudlet':dfcloud2.iloc[0]
[0],
                'Cloudletlen_mean':dfcloud2.iloc[1][6],'max_Cloudletle
n':dfcloud2.iloc[7][6],
                'min_Cloudletlen':dfcloud2.iloc[3][6], 'Max_Clodlets':
dfcloud2.iloc[7][0]}, ignore_index=True)
    return  dfObjcloud




parameters_to_csv()
df = pd.read_csv("cloudsim\\DC_Features.csv",index_col=False )
dfFeatures_csv = pd.read_csv("cloudsim\\DC_Features.csv",index_col=Fal
se )
dfFeatures_final=pd.DataFrame()
dfFinal=pd.DataFrame()

for numsimul in range(1, df['N_simul'].max() + 1):
    txt_to_csv(numsimul)
    dfsimul = pd.read_csv("cloudsim\\data" + str(numsimul) + ".csv")
    dfcloud = pd.read_csv("cloudsim\\simulation" + str(numsimul) + ".c
sv", delimiter=';')
    #print(numsimul)
    dfObj = make_tables(dfsimul)
    dfObjcloud = make_tables_simulation(dfcloud)
    df5 = pd.concat([dfObj, dfObjcloud], axis=1)
    df5 = df5.loc[:,~df5.columns.duplicated()]
    dfFeatures = pd.DataFrame(np.repeat(dfFeatures_csv.values,len(df5.
index),axis=0))
```

```
    dfFeatures.columns = dfFeatures_csv.columns
    dfFeatures=dfFeatures.loc[dfFeatures['N_simul']==numsimul]
    dfFeatures_final=dfFeatures_final.append(dfFeatures,ignore_index=T
rue)
    dfFinal=dfFinal.append(df5,ignore_index=True)

df6 =dfFeatures_final.join(dfFinal, how='outer')

df6=df6[['N_simul','Host_id', 'Vm_id', 'Ram_host', 'Storage_host', 'Vm
_Ram', 'Num_cloudlets',
       'storage_vm',  'Max_CpuUsage', 'Max_RamUsage','Max_BwUsage',
       'Mean_CpuUsage', 'Mean_RamUsage','Mean_BwUsage', 'Min_CpuUsage'
, 'Min_RamUsage',
       'Cloudletlen_mean', 'Max_Clodlets', 'Mean_ExecTime',
       'Mean_FinishTime', 'NumCloudlet', 'max_Cloudletlen', 'min_Cloud
letlen']]
df6.to_csv('cloudsim\\cognet_dataset.csv',index=False)


from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
import matplotlib.pyplot as plt

url='cloudsim\\cognet_dataset.csv'
'''names = ['N_simul','Host_id', 'Vm_id', 'Ram_host', 'Storage_host',
'Vm_Ram', 'Num_cloudlets',
       'storage_vm',  'Max_CpuUsage', 'Max_RamUsage','Max_BwUsage',
       'Mean_CpuUsage', 'Mean_RamUsage','Mean_BwUsage', 'Min_CpuUsage'
, 'Min_RamUsage',
       'Cloudletlen_mean', 'Max_Clodlets', 'Mean_ExecTime',
       'Mean_FinishTime', 'NumCloudlet', 'max_Cloudletlen', 'min_Cloud
letlen']'''
dataset = pd.read_csv(url)#, names=names)
dataset.fillna(0, inplace=True)
suma=dataset.isnull().sum()
dataset.to_csv('cloudsim\\cognet_dataset.csv',index=False)

dataset.fillna(0, inplace=True)
suma=dataset.isnull().sum()
suma


dataset.hist()
plt.rcParams['figure.figsize'] = (24, 18)
plt.show()
```

png

png

```
scatter_matrix(dataset)
plt.rcParams['figure.figsize'] = (34, 24)
plt.show()
```

png

png

```
array = dataset.values
X = array[:,0:300]
Y = array[:,22]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train,Y_validation = model_selection.train_te
st_split(X,
                                        Y, test_size=validation_size, rand
om_state=seed)
seed = 7
scoring = 'accuracy'
```

```
models = []
#models.append(('LR', LogisticRegression(solver='liblinear', multi_cla
ss='ovr')))
#models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
#models.append(('SVM', SVC(gamma='auto')))
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_train, Y_tra
in, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
KNN: 0.046046 (0.010447)
CART: 0.793339 (0.030894)
NB: 0.633490 (0.040810)
```

```
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

png

png

```
knn = KNeighborsClassifier()
knn.fit(X_train, Y_train)
predictions = knn.predict(X_validation)
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
```

```
from sklearn import preprocessing
dataset = dataset.astype({"N_simul":'category',"Host_id":'category',"V
m_id":'category'})
dataset.dtypes
#dataset_norm=dataset.loc[:, dataset.columns != 'N_simul',"Host_id","V
m_id"]
dataset_norm_incomplete=dataset[dataset.columns.difference(['N_simul',
"Host_id","Vm_id"])]
dataset_categorical=dataset[['N_simul',"Host_id","Vm_id"]]
```

```
x = dataset_norm_incomplete.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
dataset_norm_incomplete = pd.DataFrame(x_scaled)
names=['Cloudletlen_mean', 'Max_BwUsage', 'Max_Clodlets', 'Max_CpuUsag
e',
       'Max_RamUsage', 'Mean_BwUsage', 'Mean_CpuUsage', 'Mean_ExecTime
',
       'Mean_FinishTime', 'Mean_RamUsage', 'Min_CpuUsage', 'Min_RamUsa
ge',
       'NumCloudlet', 'Num_cloudlets', 'Ram_host', 'Storage_host', 'Vm
_Ram',
       'max_Cloudletlen', 'min_Cloudletlen', 'storage_vm']
dataset_norm_incomplete.columns = [names]
```

```
dataset.head()
```

5 rows × 23 columns

```
print(dataset.describe())
dataset.shape
(12706, 23)
```

```
Calcular las predicciones en funcion de los datos categoricos
We should represent it according to N_simul, VM and HOST
Test what does it have more accuracy
```

```
dataset_norm=dataset_categorical.join(dataset_norm_incomplete, how='ou
ter')
namecolumns=['N_simul',"Host_id","Vm_id",'Cloudletlen_mean', 'Max_BwUs
age', 'Max_Clodlets', 'Max_CpuUsage',
       'Max_RamUsage', 'Mean_BwUsage', 'Mean_CpuUsage', 'Mean_ExecTime
',
       'Mean_FinishTime', 'Mean_RamUsage', 'Min_CpuUsage', 'Min_RamUsa
ge',
       'NumCloudlet', 'Num_cloudlets', 'Ram_host', 'Storage_host', 'Vm
_Ram',
```

```
       'max_Cloudletlen', 'min_Cloudletlen', 'storage_vm']
dataset_norm.columns = [namecolumns]

# Split-out validation dataset
#MinMaxScaler(copy=True, feature_range=(0, 10))
#target_train=scaler.transform(target_train)
#print(target_train)

target_train=dataset_norm['Mean_ExecTime']
from sklearn.model_selection import train_test_split
data=dataset_norm.drop(['Mean_ExecTime'],axis=1)
#array = dataset_norm.values
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(data,t
arget_train, test_size=validation_size, random_state=seed)
scoring = 'accuracy'

#est = KBinsDiscretizer(n_bins=10, encode='ordinal', strategy='uniform
')
#est.fit(Y_train)
Y_train.Mean_ExecTime =round(Y_train,2)
Y_validation.Mean_ExecTime=round(Y_validation,2)
```

```
c:\users\delas\appdata\local\programs\python\python37-32\lib\site-pack
ages\pandas\core\generic.py:4405: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  self[name] = value
```

```
import seaborn as sns
import matplotlib.pyplot as plt
model = LinearRegression()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
sns.distplot(Y_validation - predictions, axlabel="Test - Prediction")
plt.show()
```

png

png

```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train,Y_train)
predictions = lm.predict(X_validation)

plt.figure(figsize=(15,10))
plt.subplots_adjust(hspace=0.4, wspace=0.4)
plt.suptitle('Linear regression model, 250 simulations')
plt.subplot(2, 2,1)
plt.xlabel('Mean_execution time true')
plt.ylabel('Mean_execution time predicted')
```

```
plt.scatter(Y_validation,predictions)

plt.subplot(2, 2,2)
plt.xlabel('Vm_id')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['Vm_id'],Y_validation,label="True data")
plt.scatter(X_validation['Vm_id'],predictions,label="Predictions")
plt.legend()

plt.subplot(2, 2,3)
plt.xlabel('Host_id')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['Host_id'],Y_validation,label="True data")
plt.scatter(X_validation['Host_id'],predictions,label="Predictions")
plt.legend()

plt.subplot(2, 2,4)
plt.xlabel('Number of simulation')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['N_simul'],Y_validation,label="True data")
plt.scatter(X_validation['N_simul'],predictions,label="Predictions")
plt.legend()

plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```

png

png

```
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_validation
,predictions))
print('Mean Squared Error:', metrics.mean_squared_error(Y_validation,p
redictions))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(Y
_validation,predictions)))
```

```
Mean Absolute Error: 0.003600919583487267
Mean Squared Error: 2.086629223845936e-05
Root Mean Squared Error: 0.004567963686201912
```

```
clf = svm.SVR()
clf.fit(X_train,Y_train)
predictions=clf.predict(X_validation)

plt.figure(figsize=(15,10))
plt.subplots_adjust(hspace=0.4, wspace=0.4)
plt.suptitle('Support vector Regression, 250 simulations')
plt.subplot(2, 2,1)
plt.xlabel('Mean_execution time true')
plt.ylabel('Mean_execution time predicted')
plt.scatter(Y_validation,predictions,s=20,edgecolor="black",c="darkora
nge")
```

```
plt.subplot(2, 2,2)
plt.xlabel('Vm_id')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['Vm_id'],Y_validation,label="True data")
plt.scatter(X_validation['Vm_id'],predictions,label="Predictions")
plt.legend()

plt.subplot(2, 2,3)
plt.xlabel('Host_id')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['Host_id'],Y_validation,label="True data")
plt.scatter(X_validation['Host_id'],predictions,label="Predictions")
plt.legend()

plt.subplot(2, 2,4)
plt.xlabel('Number of simulation')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['N_simul'],Y_validation,label="True data")
plt.scatter(X_validation['N_simul'],predictions,label="Predictions")
plt.legend()

plt.show
<function matplotlib.pyplot.show(*args, **kw)>
```

png

png

```
print(np.sqrt(metrics.mean_squared_error(Y_validation, predictions)))
print(np.sqrt(metrics.mean_absolute_error(Y_validation, predictions)))
print(metrics.r2_score(Y_validation, predictions))

0.03894161752648024
0.17676675125092087
0.9500909638076308

from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
regressor.fit(X_train, Y_train)
X_columns = pd.DataFrame(X_train.columns)
X_columns.T
coeff_df = pd.DataFrame(regressor.coef_.T, X_columns, columns=['Coeffi
cient'])
print(coeff_df)

                         Coefficient
((Cloudletlen_mean,),)  1.361957e-01
((Max_BwUsage,),)      -6.369338e-02
((Max_Clodlets,),)     -9.706228e-01
((Max_CpuUsage,),)     -2.012091e-03
((Max_RamUsage,),)      4.284801e-01
((Mean_BwUsage,),)     -3.494495e-03
((Mean_CpuUsage,),)     1.010490e-03
((Mean_FinishTime,),)   1.309856e+00
```

```
((Mean_RamUsage,),)     6.615824e-02
((Min_CpuUsage,),)      0.000000e+00
((Min_RamUsage,),)      1.110223e-16
((NumCloudlet,),)       1.774677e-01
((Num_cloudlets,),)     4.046876e-01
((Ram_host,),)         -5.551115e-17
((Storage_host,),)      0.000000e+00
((Vm_Ram,),)            0.000000e+00
((max_Cloudletlen,),)  -1.542416e-01
((min_Cloudletlen,),)   2.803162e-02
((storage_vm,),)       -7.900223e-02
```

```
from sklearn.tree import DecisionTreeRegressor

# create a regressor object
regressor = DecisionTreeRegressor(random_state = 0)
# fit the regressor with X and Y data
regressor.fit(X_train, Y_train)
predictions = regressor.predict(X_validation)

plt.figure(figsize=(15,10))
plt.subplots_adjust(hspace=0.4, wspace=0.4)
plt.suptitle('Decision Tree Regression, 250 simulations')
plt.subplot(2, 2,1)
plt.xlabel('Mean_execution time true')
plt.ylabel('Mean_execution time predicted')
plt.scatter(Y_validation,predictions,s=20,edgecolor="black",c="darkora
nge")

plt.subplot(2, 2,2)
plt.xlabel('Vm_id')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['Vm_id'],Y_validation,label="True data")
plt.scatter(X_validation['Vm_id'],predictions,label="Predictions")
plt.legend()

plt.subplot(2, 2,3)
plt.xlabel('Host_id')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['Host_id'],Y_validation,label="True data")
plt.scatter(X_validation['Host_id'],predictions,label="Predictions")
plt.legend()

plt.subplot(2, 2,4)
plt.xlabel('Number of simulation')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['N_simul'],Y_validation,label="True data")
plt.scatter(X_validation['N_simul'],predictions,label="Predictions")
plt.legend()

plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```

png

png

```
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_validation
,predictions))
print('Mean Squared Error:', metrics.mean_squared_error(Y_validation,p
redictions))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(Y
_validation,predictions)))
print('R2 Score:',metrics.r2_score(Y_validation, predictions))
```

```
Mean Absolute Error: 0.018772455089820356
Mean Squared Error: 0.0007940119760479043
Root Mean Squared Error: 0.02817821811342769
```

```
#Encoding the Y_train in order to handle with the continuous data

lab_enc = preprocessing.LabelEncoder()
training_scores_encoded = lab_enc.fit_transform(Y_train)
training_scores_encoded_validation=lab_enc.fit_transform(Y_validation)


from sklearn.svm import SVR
from sklearn import svm
clf = svm.SVC()
clf.fit(X_train,training_scores_encoded)

svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, training_scores_encoded)
y_pred = svclassifier.predict(X_validation)


print(confusion_matrix(training_scores_encoded_validation,y_pred))
print(classification_report(training_scores_encoded_validation,y_pred

#Polynomial Kernel
svclassifier = SVC(kernel='poly', degree=8)
svclassifier.fit(X_train, training_scores_encoded)
y_pred = svclassifier.predict(X_validation)
print(confusion_matrix(training_scores_encoded_validation,y_pred))
print(classification_report(training_scores_encoded_validation,y_pred)
)
```

```
c:\users\delas\appdata\local\programs\python\python37-32\lib\site-pack
ages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: P
recision and F-score are ill-defined and being set to 0.0 in labels wi
th no predicted samples.
  'precision', 'predicted', average, warn_for)
```

```
#Gaussian Kernel
svclassifier = SVC(kernel='rbf')
svclassifier.fit(X_train, training_scores_encoded)
y_pred = svclassifier.predict(X_validation)
print(confusion_matrix(training_scores_encoded_validation,y_pred))
print(classification_report(training_scores_encoded_validation,y_pred)
)
```

```
#Sigmoid Kernel
svclassifier = SVC(kernel='sigmoid')
svclassifier.fit(X_train,training_scores_encoded)
y_pred = svclassifier.predict(X_validation)
print(confusion_matrix(training_scores_encoded_validation,y_pred))
print(classification_report(training_scores_encoded_validation,y_pred)
)

c:\users\delas\appdata\local\programs\python\python37-32\lib\site-)

#Logistic regression
from sklearn.linear_model import LogisticRegression

# instantiate the model (using the default parameters)
logreg = LogisticRegression()

# fit the model with data
logreg.fit(X_train,training_scores_encoded)
y_pred=logreg.predict(X_validation)
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(training_scores_encoded_validati
on, y_pred)
cnf_matrix

import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
# evaluate each model in turn
results = []
names = []
scoring = 'accuracy'
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_train,traini
ng_scores_encoded, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
# boxplot algorithm comparison
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
```

```
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
LR: 0.088356 (0.011503)
LDA: 0.481827 (0.021984)
KNN: 0.166602 (0.013082)
CART: 0.212278 (0.026045)
NB: 0.112309 (0.015819)

SVM: 0.070757 (0.006964)
```

```
target_train=dataset_norm[['Mean_RamUsage','Mean_CpuUsage','Mean_ExecT
ime','Mean_FinishTime']]
from sklearn.model_selection import train_test_split
data=dataset_norm.drop(['Mean_RamUsage','Mean_CpuUsage','Mean_ExecTime
','Mean_FinishTime'],axis=1)
#array = dataset_norm.values
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(data,t
arget_train, test_size=validation_size, random_state=seed)
scoring = 'accuracy'
Y_train.Mean_ExecTime =round(Y_train,2)
Y_validation.Mean_ExecTime=round(Y_validation,2)

model = LinearRegression()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)

predictions_table = pd.DataFrame()
for row in predictions:
    predictions_table = predictions_table.append(pd.DataFrame([row]),
ignore_index=True)
names=['Mean_RamUsage','Mean_CpuUsage','Mean_ExecTime','Mean_FinishTim
e']
predictions_table.columns = [names]
#final
[668 rows x 4 columns]
```

```
plt.figure(figsize=(15,10))
plt.suptitle('Multi target linear regression, 250 simulations')
plt.subplots_adjust(hspace=0.3, wspace=0.3)
plt.subplot(2, 2,1)
plt.xlabel('Number of simulation')
plt.ylabel('Mean Ram usage')
plt.scatter(X_validation['N_simul'],Y_validation['Mean_RamUsage'],labe
l="True data")
plt.scatter(X_validation['N_simul'],predictions_table['Mean_RamUsage']
,label="Predictions")
plt.legend()
plt.subplot(2, 2,2)
plt.xlabel('Virtual machine')
plt.ylabel('Mean CPU usage')
plt.scatter(X_validation['Vm_id'],Y_validation['Mean_CpuUsage'],label=
```

```
"True data")
plt.scatter(X_validation['Vm_id'],predictions_table['Mean_CpuUsage'],l
abel="Predictions")
plt.legend()
plt.subplot(2, 2,3)
plt.xlabel('Virtual machine')
plt.ylabel('Mean Execution time ')
plt.scatter(X_validation['Vm_id'],Y_validation['Mean_ExecTime'],label=
"True data")
plt.scatter(X_validation['Vm_id'],predictions_table['Mean_ExecTime'],l
abel="Predictions")
plt.legend()
plt.subplot(2, 2,4)
plt.xlabel('Number of simulation')
plt.ylabel('Mean CPU usage')
plt.scatter(X_validation['N_simul'],Y_validation['Mean_CpuUsage'],labe
l="True data")
plt.scatter(X_validation['N_simul'],predictions_table['Mean_CpuUsage']
,label="Predictions")
plt.legend()
plt.show()
```

png

png

```
plt.figure(figsize=(15,10))
plt.suptitle('Multi target linear regression, 250 simulations')
plt.subplots_adjust(hspace=0.2, wspace=0.2)

plt.subplot(2, 2,1)
plt.scatter(Y_validation['Mean_RamUsage'],predictions_table['Mean_RamU
sage'],label='Mean_RamUsage',s=20,edgecolor="black",c="darkorange")
plt.xlabel('True Data')
plt.ylabel('Predicted')
plt.legend()
plt.subplot(2, 2,2)
plt.scatter(Y_validation['Mean_CpuUsage'],predictions_table['Mean_CpuU
sage'],label='Mean_CpuUsage',s=20,edgecolor="black",c="darkorange")
plt.xlabel('True Data')
plt.ylabel('Predicted')
plt.legend()
plt.subplot(2, 2,3)
plt.scatter(Y_validation['Mean_ExecTime'],predictions_table['Mean_Exec
Time'],label='Mean_ExecTime',s=20,edgecolor="black",c="darkorange")
plt.xlabel('True Data')
plt.ylabel('Predicted')
plt.legend()
plt.subplot(2, 2,4)
plt.scatter(Y_validation['Mean_FinishTime'],predictions_table['Mean_Fi
nishTime'],label='Mean_FinishTime',s=20,edgecolor="black",c="darkorang
e")
plt.xlabel('True Data')
plt.ylabel('Predicted')
```

```
plt.legend()
plt.show()
```

png

png

```
import seaborn as sns
import matplotlib.pyplot as plt
model = LinearRegression()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
sns.distplot(Y_validation - predictions, axlabel="Test - Prediction")
plt.show()
```

png

png

```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train,Y_train)
predictions = lm.predict(X_validation)

plt.figure(figsize=(15,10))
plt.subplots_adjust(hspace=0.4, wspace=0.4)
plt.suptitle('Linear regression model, 500 simulations')
plt.subplot(2, 2,1)
plt.xlabel('Mean_execution time true')
plt.ylabel('Mean_execution time predicted')
plt.scatter(Y_validation,predictions)

plt.subplot(2, 2,2)
plt.xlabel('Vm_id')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['Vm_id'],Y_validation,label="True data")
plt.scatter(X_validation['Vm_id'],predictions,label="Predictions")
plt.legend()

plt.subplot(2, 2,3)
plt.xlabel('Host_id')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['Host_id'],Y_validation,label="True data")
plt.scatter(X_validation['Host_id'],predictions,label="Predictions")
plt.legend()

plt.subplot(2, 2,4)
plt.xlabel('Number of simulation')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['N_simul'],Y_validation,label="True data")
plt.scatter(X_validation['N_simul'],predictions,label="Predictions")
plt.legend()

plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```

png

png

```
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_validation
,predictions))
print('Mean Squared Error:', metrics.mean_squared_error(Y_validation,p
redictions))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(Y
_validation,predictions)))
print('R2 Score:',metrics.r2_score(Y_validation, predictions))
```

```
Mean Absolute Error: 0.003600919583487267
Mean Squared Error: 2.086629223845936e-05
Root Mean Squared Error: 0.004567963686201912
R2 Score: 0.999313253436645
```

```
clf = svm.SVR()
clf.fit(X_train,Y_train)
predictions=clf.predict(X_validation)

plt.figure(figsize=(15,10))
plt.subplots_adjust(hspace=0.4, wspace=0.4)
plt.suptitle('Support vector Regression, 500 simulations')
plt.subplot(2, 2,1)
plt.xlabel('Mean_execution time true')
plt.ylabel('Mean_execution time predicted')
plt.scatter(Y_validation,predictions,s=20,edgecolor="black",c="darkora
nge")

plt.subplot(2, 2,2)
plt.xlabel('Vm_id')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['Vm_id'],Y_validation,label="True data")
plt.scatter(X_validation['Vm_id'],predictions,label="Predictions")
plt.legend()

plt.subplot(2, 2,3)
plt.xlabel('Host_id')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['Host_id'],Y_validation,label="True data")
plt.scatter(X_validation['Host_id'],predictions,label="Predictions")
plt.legend()

plt.subplot(2, 2,4)
plt.xlabel('Number of simulation')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['N_simul'],Y_validation,label="True data")
plt.scatter(X_validation['N_simul'],predictions,label="Predictions")
plt.legend()

plt.show
```

png

png

```
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_validation
,predictions))
print('Mean Squared Error:', metrics.mean_squared_error(Y_validation,p
redictions))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(Y
_validation,predictions)))
print('R2 Score:',metrics.r2_score(Y_validation, predictions))
```

```
Mean Absolute Error: 0.05953956952186667
Mean Squared Error: 0.0053917670540499915
Root Mean Squared Error: 0.07342865281380281
R2 Score: 0.8225474151102468
```

```
from sklearn.tree import DecisionTreeRegressor

# create a regressor object
regressor = DecisionTreeRegressor(random_state = 0)
# fit the regressor with X and Y data
regressor.fit(X_train, Y_train)
predictions = regressor.predict(X_validation)

plt.figure(figsize=(15,10))
plt.subplots_adjust(hspace=0.4, wspace=0.4)
plt.suptitle('Decision Tree Regression, 500 simulations')
plt.subplot(2, 2,1)
plt.xlabel('Mean_execution time true')
plt.ylabel('Mean_execution time predicted')
plt.scatter(Y_validation,predictions,s=20,edgecolor="black",c="darkora
nge")

plt.subplot(2, 2,2)
plt.xlabel('Vm_id')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['Vm_id'],Y_validation,label="True data")
plt.scatter(X_validation['Vm_id'],predictions,label="Predictions")
plt.legend()

plt.subplot(2, 2,3)
plt.xlabel('Host_id')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['Host_id'],Y_validation,label="True data")
plt.scatter(X_validation['Host_id'],predictions,label="Predictions")
plt.legend()

plt.subplot(2, 2,4)
plt.xlabel('Number of simulation')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['N_simul'],Y_validation,label="True data")
plt.scatter(X_validation['N_simul'],predictions,label="Predictions")
plt.legend()

plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```

png

png

```
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_validation
,predictions))
print('Mean Squared Error:', metrics.mean_squared_error(Y_validation,p
redictions))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(Y
_validation,predictions)))
print('R2 Score:',metrics.r2_score(Y_validation, predictions))
```

```
Mean Absolute Error: 0.01845808383233533
Mean Squared Error: 0.0007770958083832337
Root Mean Squared Error: 0.027876438229860603
R2 Score: 0.9744244032573669
```

```
#Encoding the Y_train in order to handle with the continuous data

lab_enc = preprocessing.LabelEncoder()
training_scores_encoded = lab_enc.fit_transform(Y_train)
training_scores_encoded_validation=lab_enc.fit_transform(Y_validation)


from sklearn.svm import SVR
from sklearn import svm
clf = svm.SVC()
clf.fit(X_train,training_scores_encoded)

svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, training_scores_encoded)
y_pred = svclassifier.predict(X_validation)


import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
# evaluate each model in turn
results = []
```

```
names = []
scoring = 'accuracy'
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_train,traini
ng_scores_encoded, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
# boxplot algorithm comparison
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

```
LR: 0.119051 (0.019042)
LDA: 0.720334 (0.025901)
KNN: 0.127659 (0.017453)
CART: 0.214519 (0.019078)
NB: 0.130283 (0.021463)
SVM: 0.133273 (0.014411)
```

png

png

```
target_train=dataset_norm[['Mean_RamUsage','Mean_CpuUsage','Mean_ExecT
ime','Mean_FinishTime']]
from sklearn.model_selection import train_test_split
data=dataset_norm.drop(['Mean_RamUsage','Mean_CpuUsage','Mean_ExecTime
','Mean_FinishTime'],axis=1)
#array = dataset_norm.values
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(data,t
arget_train, test_size=validation_size, random_state=seed)
scoring = 'accuracy'
Y_train.Mean_ExecTime =round(Y_train,2)
Y_validation.Mean_ExecTime=round(Y_validation,2)

model = LinearRegression()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)

predictions_table = pd.DataFrame()
for row in predictions:
    predictions_table = predictions_table.append(pd.DataFrame([row]),
ignore_index=True)
names=['Mean_RamUsage','Mean_CpuUsage','Mean_ExecTime','Mean_FinishTim
e']
predictions_table.columns = [names]
#final
```

```
plt.figure(figsize=(15,10))
plt.suptitle('Multi target linear regression, 500 simulations')
plt.subplots_adjust(hspace=0.3, wspace=0.3)
plt.subplot(2, 2,1)
plt.xlabel('Number of simulation')
plt.ylabel('Mean Ram usage')
plt.scatter(X_validation['N_simul'],Y_validation['Mean_RamUsage'],labe
l="True data")
plt.scatter(X_validation['N_simul'],predictions_table['Mean_RamUsage']
,label="Predictions")
plt.legend()
plt.subplot(2, 2,2)
plt.xlabel('Virtual machine')
plt.ylabel('Mean CPU usage')
plt.scatter(X_validation['Vm_id'],Y_validation['Mean_CpuUsage'],label=
"True data")
plt.scatter(X_validation['Vm_id'],predictions_table['Mean_CpuUsage'],l
abel="Predictions")
plt.legend()
plt.subplot(2, 2,3)
plt.xlabel('Virtual machine')
plt.ylabel('Mean Execution time ')
plt.scatter(X_validation['Vm_id'],Y_validation['Mean_ExecTime'],label=
"True data")
plt.scatter(X_validation['Vm_id'],predictions_table['Mean_ExecTime'],l
abel="Predictions")
plt.legend()
plt.subplot(2, 2,4)
plt.xlabel('Number of simulation')
plt.ylabel('Mean CPU usage')
plt.scatter(X_validation['N_simul'],Y_validation['Mean_CpuUsage'],labe
l="True data")
plt.scatter(X_validation['N_simul'],predictions_table['Mean_CpuUsage']
,label="Predictions")
plt.legend()
plt.show()
```

png

png

```
plt.figure(figsize=(15,10))
plt.suptitle('Multi target linear regression, 500 simulations')
plt.subplots_adjust(hspace=0.2, wspace=0.2)

plt.subplot(2, 2,1)
plt.scatter(Y_validation['Mean_RamUsage'],predictions_table['Mean_RamU
sage'],label='Mean_RamUsage',s=20,edgecolor="black",c="darkorange")
plt.xlabel('True Data')
plt.ylabel('Predicted')
plt.legend()
plt.subplot(2, 2,2)
plt.scatter(Y_validation['Mean_CpuUsage'],predictions_table['Mean_CpuU
sage'],label='Mean_CpuUsage',s=20,edgecolor="black",c="darkorange")
plt.xlabel('True Data')
```

```
plt.ylabel('Predicted')
plt.legend()
plt.subplot(2, 2,3)
plt.scatter(Y_validation['Mean_ExecTime'],predictions_table['Mean_Exec
Time'],label='Mean_ExecTime',s=20,edgecolor="black",c="darkorange")
plt.xlabel('True Data')
plt.ylabel('Predicted')
plt.legend()
plt.subplot(2, 2,4)
plt.scatter(Y_validation['Mean_FinishTime'],predictions_table['Mean_Fi
nishTime'],label='Mean_FinishTime',s=20,edgecolor="black",c="darkorang
e")
plt.xlabel('True Data')
plt.ylabel('Predicted')
plt.legend()
plt.show()
```

png

png

```
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_validation
,predictions))
print('Mean Squared Error:', metrics.mean_squared_error(Y_validation,p
redictions))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(Y
_validation,predictions)))
print('R2 Score:',metrics.r2_score(Y_validation, predictions))
```

```
Mean Absolute Error: 0.03053023291998854
Mean Squared Error: 0.0016847407807049147
Root Mean Squared Error: 0.0410455939255959
R2 Score: 0.9493117158337983
```

```
import seaborn as sns
import matplotlib.pyplot as plt
model = LinearRegression()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
sns.distplot(Y_validation - predictions, axlabel="Test - Prediction")
plt.show()
```

png

png

```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train,Y_train)
predictions = lm.predict(X_validation)

plt.figure(figsize=(15,10))
plt.subplots_adjust(hspace=0.4, wspace=0.4)
plt.suptitle('Linear regression model, 750 simulations')
plt.subplot(2, 2,1)
plt.xlabel('Mean_execution time true')
```

```
plt.ylabel('Mean_execution time predicted')
plt.scatter(Y_validation,predictions)

plt.subplot(2, 2,2)
plt.xlabel('Vm_id')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['Vm_id'],Y_validation,label="True data")
plt.scatter(X_validation['Vm_id'],predictions,label="Predictions")
plt.legend()

plt.subplot(2, 2,3)
plt.xlabel('Host_id')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['Host_id'],Y_validation,label="True data")
plt.scatter(X_validation['Host_id'],predictions,label="Predictions")
plt.legend()

plt.subplot(2, 2,4)
plt.xlabel('Number of simulation')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['N_simul'],Y_validation,label="True data")
plt.scatter(X_validation['N_simul'],predictions,label="Predictions")
plt.legend()

plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```

png

png

```
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_validation
,predictions))
print('Mean Squared Error:', metrics.mean_squared_error(Y_validation,p
redictions))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(Y
_validation,predictions)))
print('R2 Score:',metrics.r2_score(Y_validation, predictions))
```

```
Mean Absolute Error: 0.006524309598998842
Mean Squared Error: 9.005736979335923e-05
Root Mean Squared Error: 0.009489856152406065
R2 Score: 0.996920870360047
```

```
clf = svm.SVR()
clf.fit(X_train,Y_train)
predictions=clf.predict(X_validation)

plt.figure(figsize=(15,10))
plt.subplots_adjust(hspace=0.4, wspace=0.4)
plt.suptitle('Support vector Regression, 750 simulations')
plt.subplot(2, 2,1)
plt.xlabel('Mean_execution time true')
plt.ylabel('Mean_execution time predicted')
```

```
plt.scatter(Y_validation,predictions,s=20,edgecolor="black",c="darkora
nge")

plt.subplot(2, 2,2)
plt.xlabel('Vm_id')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['Vm_id'],Y_validation,label="True data")
plt.scatter(X_validation['Vm_id'],predictions,label="Predictions")
plt.legend()

plt.subplot(2, 2,3)
plt.xlabel('Host_id')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['Host_id'],Y_validation,label="True data")
plt.scatter(X_validation['Host_id'],predictions,label="Predictions")
plt.legend()

plt.subplot(2, 2,4)
plt.xlabel('Number of simulation')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['N_simul'],Y_validation,label="True data")
plt.scatter(X_validation['N_simul'],predictions,label="Predictions")
plt.legend()

plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```

png

png

```
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_validation
,predictions))
print('Mean Squared Error:', metrics.mean_squared_error(Y_validation,p
redictions))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(Y
_validation,predictions)))
print('R2 Score:',metrics.r2_score(Y_validation, predictions))
```

```
Mean Absolute Error: 0.060022034988154414
Mean Squared Error: 0.005146782674973025
Root Mean Squared Error: 0.07174108080432734
R2 Score: 0.8240276046117104
```

```
from sklearn.tree import DecisionTreeRegressor

# create a regressor object
regressor = DecisionTreeRegressor(random_state = 0)
# fit the regressor with X and Y data
regressor.fit(X_train, Y_train)
predictions = regressor.predict(X_validation)

plt.figure(figsize=(15,10))
plt.subplots_adjust(hspace=0.4, wspace=0.4)
```

```
plt.suptitle('Decision Tree Regression, 750 simulations')
plt.subplot(2, 2,1)
plt.xlabel('Mean_execution time true')
plt.ylabel('Mean_execution time predicted')
plt.scatter(Y_validation,predictions,s=20,edgecolor="black",c="darkora
nge")

plt.subplot(2, 2,2)
plt.xlabel('Vm_id')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['Vm_id'],Y_validation,label="True data")
plt.scatter(X_validation['Vm_id'],predictions,label="Predictions")
plt.legend()

plt.subplot(2, 2,3)
plt.xlabel('Host_id')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['Host_id'],Y_validation,label="True data")
plt.scatter(X_validation['Host_id'],predictions,label="Predictions")
plt.legend()

plt.subplot(2, 2,4)
plt.xlabel('Number of simulation')
#plt.ylabel('Mean_execusion time predicted')
plt.scatter(X_validation['N_simul'],Y_validation,label="True data")
plt.scatter(X_validation['N_simul'],predictions,label="Predictions")
plt.legend()

plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```

png

png

```
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_validation
,predictions))
print('Mean Squared Error:', metrics.mean_squared_error(Y_validation,p
redictions))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(Y
_validation,predictions)))
print('R2 Score:',metrics.r2_score(Y_validation, predictions))
```

```
Mean Absolute Error: 0.012183320220298976
Mean Squared Error: 0.0002891817466561762
Root Mean Squared Error: 0.017005344649732218
R2 Score: 0.9901126571927921
```

```
import warnings
warnings.filterwarnings('ignore')
#Encoding the Y_train in order to handle with the continuous data

lab_enc = preprocessing.LabelEncoder()
training_scores_encoded = lab_enc.fit_transform(Y_train)
```

```
training_scores_encoded_validation=lab_enc.fit_transform(Y_validation)


from sklearn.svm import SVR
from sklearn import svm
clf = svm.SVC()
clf.fit(X_train,training_scores_encoded)

svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, training_scores_encoded)
y_pred = svclassifier.predict(X_validation)

import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
# evaluate each model in turn
results = []
names = []
scoring = 'accuracy'
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_train,traini
ng_scores_encoded, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
# boxplot algorithm comparison
fig = plt.figure()
fig.suptitle('Algorithm Comparison 750 simulation')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

```
LR: 0.116883 (0.007268)
LDA: 0.445292 (0.020541)
KNN: 0.164109 (0.006022)
CART: 0.280696 (0.010406)
NB: 0.155449 (0.010608)
SVM: 0.172473 (0.010660)
```

png

png

```
target_train=dataset_norm[['Mean_RamUsage','Mean_CpuUsage','Mean_ExecT
ime','Mean_FinishTime']]
from sklearn.model_selection import train_test_split
data=dataset_norm.drop(['Mean_RamUsage','Mean_CpuUsage','Mean_ExecTime
','Mean_FinishTime'],axis=1)
#array = dataset_norm.values
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(data,t
arget_train, test_size=validation_size, random_state=seed)
scoring = 'accuracy'
Y_train.Mean_ExecTime =round(Y_train,2)
Y_validation.Mean_ExecTime=round(Y_validation,2)

model = LinearRegression()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)

predictions_table = pd.DataFrame()
for row in predictions:
    predictions_table = predictions_table.append(pd.DataFrame([row]),
ignore_index=True)
names=['Mean_RamUsage','Mean_CpuUsage','Mean_ExecTime','Mean_FinishTim
e']
predictions_table.columns = [names]
#final

plt.figure(figsize=(15,10))
plt.suptitle('Multi target linear regression, 750 simulations')
plt.subplots_adjust(hspace=0.3, wspace=0.3)
plt.subplot(2, 2,1)
plt.xlabel('Number of simulation')
plt.ylabel('Mean Ram usage')
plt.scatter(X_validation['N_simul'],Y_validation['Mean_RamUsage'],labe
l="True data")
plt.scatter(X_validation['N_simul'],predictions_table['Mean_RamUsage']
,label="Predictions")
plt.legend()
plt.subplot(2, 2,2)
plt.xlabel('Virtual machine')
plt.ylabel('Mean CPU usage')
plt.scatter(X_validation['Vm_id'],Y_validation['Mean_CpuUsage'],label=
"True data")
plt.scatter(X_validation['Vm_id'],predictions_table['Mean_CpuUsage'],l
abel="Predictions")
plt.legend()
plt.subplot(2, 2,3)
plt.xlabel('Virtual machine')
plt.ylabel('Mean Execution time ')
plt.scatter(X_validation['Vm_id'],Y_validation['Mean_ExecTime'],label=
"True data")
```

```
plt.scatter(X_validation['Vm_id'],predictions_table['Mean_ExecTime'],l
abel="Predictions")
plt.legend()
plt.subplot(2, 2,4)
plt.xlabel('Number of simulation')
plt.ylabel('Mean CPU usage')
plt.scatter(X_validation['N_simul'],Y_validation['Mean_CpuUsage'],labe
l="True data")
plt.scatter(X_validation['N_simul'],predictions_table['Mean_CpuUsage']
,label="Predictions")
plt.legend()
plt.show()
```

png

png

```
plt.figure(figsize=(15,10))
plt.suptitle('Multi target linear regression, 750 simulations')
plt.subplots_adjust(hspace=0.2, wspace=0.2)

plt.subplot(2, 2,1)
plt.scatter(Y_validation['Mean_RamUsage'],predictions_table['Mean_RamU
sage'],label='Mean_RamUsage',s=20,edgecolor="black",c="darkorange")
plt.xlabel('True Data')
plt.ylabel('Predicted')
plt.legend()
plt.subplot(2, 2,2)
plt.scatter(Y_validation['Mean_CpuUsage'],predictions_table['Mean_CpuU
sage'],label='Mean_CpuUsage',s=20,edgecolor="black",c="darkorange")
plt.xlabel('True Data')
plt.ylabel('Predicted')
plt.legend()
plt.subplot(2, 2,3)
plt.scatter(Y_validation['Mean_ExecTime'],predictions_table['Mean_Exec
Time'],label='Mean_ExecTime',s=20,edgecolor="black",c="darkorange")
plt.xlabel('True Data')
plt.ylabel('Predicted')
plt.legend()
plt.subplot(2, 2,4)
plt.scatter(Y_validation['Mean_FinishTime'],predictions_table['Mean_Fi
nishTime'],label='Mean_FinishTime',s=20,edgecolor="black",c="darkorang
e")
plt.xlabel('True Data')
plt.ylabel('Predicted')
plt.legend()
plt.show()
```

png

png

```
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_validation
,predictions))
print('Mean Squared Error:', metrics.mean_squared_error(Y_validation,p
```

```
redictions))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(Y
_validation,predictions)))
print('R2 Score:',metrics.r2_score(Y_validation, predictions))
```

```
Mean Absolute Error: 0.03804439270685506
Mean Squared Error: 0.002663392505041593
Root Mean Squared Error: 0.051608066278844365
R2 Score: 0.9363153172418959
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(15,10))
plt.xlabel('Host_id')
#plt.ylabel('Mean_execusion time predicted')
#plt.plot(X_validation['Host_id'],Y_validation['Mean_CpuUsage'],label=
"True data")
#plt.plot(X_validation['Host_id'],predictions_table['Mean_CpuUsage'],l
abel="Predictions")
plt.scatter(X_validation['Host_id'],Y_validation['Mean_RamUsage'],labe
l="True data")
plt.scatter(X_validation['Host_id'],predictions_table['Mean_RamUsage']
,label="Predictions")
plt.scatter(X_validation['Host_id'],Y_validation['Mean_CpuUsage'],labe
l="True data")
plt.scatter(X_validation['Host_id'],predictions_table['Mean_CpuUsage']
,label="Predictions")
plt.scatter(X_validation['Host_id'],Y_validation['Mean_ExecTime'],labe
l="True data")
plt.scatter(X_validation['Host_id'],predictions_table['Mean_ExecTime']
,label="Predictions")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1d4ec710>
```

png

png

```
plt.figure(figsize=(15,10))
plt.xlabel('Host_id')
plt.scatter(names,[Y_validation['Mean_RamUsage'],Y_validation['Mean_Cp
uUsage'],Y_validation['Mean_ExecTime'],
                   Y_validation['Mean_FinishTime']])
```

REFERENCES

[1] The 12th International Conference on Future Networks and Communications (FNC-2017) Automatic monitoring management for 5G mobile networks Alberto Huertas Celdrana,_, Manuel Gil P´ereza, F´elix J. Garc´ıa Clementeb, and Gregorio Mart´ınez P´ereza *aDept. Ingenier´ıa de la Informaci´on y las Comunicaciones, University of Murcia, 30071 Murcia, Spain bDept. Ingeniería y Tecnología de Computadores, University of Murcia, 30071 Murcia, Spain*

[2] 5G PPP 5G Architecture White Paper Revision 2.0 – (White Paper version 2.0 Dec 2017) 5GPPP Architecture Working Group.

[3] Data Traffic Model in Machine to Machine Communications over 5G Network Slicing, Mohammed Dighriri, Gyu Myoung Lee, Thar Baker, Ali Saeed Dayem Alfoudi, Liverpool John Moores University.

[4] Demonstration of Resource Orchestration Using Big Data Analytics for Dynamic Slicing in 5G Networks, M. R. Raza(1), C. Natalino(1), A. Vidal(2), M. A. S. Santos(2), P. Öhlen(3), L. Wosinska(1), P. Monti(1). KTH Royal Institute of Technology, Electrum 229, SE-164 40 Kista, Sweden

[5] Morocho Cayamcela, Manuel Eugenio & Lim, Wansu. (2018). Artificial Intelligence in 5G Technology: A Survey. 860-865.10.1109/ICTC.2018.8539642.

[6] Machine Learning at the Edge:A Data-Driven Architecture with Applications to 5G Cellular Networks Michele Polese, Student Member, IEEE, Rittwik Jana, Member, IEEE, Velin Kounev, Ke Zhang, Supratim Deb, Senior Member, IEEE, Michele Zorzi, Fellow, IEEE

[7] Statistical QoS-Driven Power Allocation for Cooperative Caching Over 5G Big Data Mobile Wireless Networks Jingqing Wang and Xi Zhang Networking and Information Systems Laboratory Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843, USA E-mail: fwang12078@tamu.edu, xizhang@ece.tamu.edu

[8] Xu, Lei & Assem, Haytham & Grida Ben Yahia, Imen & Buda, Teodora & Martin, Angel & Gallico, Domenico & Biancani, Matteo & Pastor, Antonio & Aranda Gutiérrez, Pedro & Smirnov, Mikhail & Raz, Danny & Uryupina, Olga & Mozo, Alberto & Ordozgoiti, Bruno & Corici, Marius-Iulian & O'Sullivan, Pat & Mullins, Robert. (2016). CogNet: A network management architecture featuring cognitive capabilities. 325-329. 10.1109/EuCNC.2016.7561056.

[9] Applying Big Data, Machine Learning, and SDN/NFV to 5G Traffic Clustering, Forecasting and management. Luong-Vy Le ; Do Sinh ; Bao-Shuh Paul Lin National Chiao Tung University, Deparment of Computer Science, Hsinchu, Taiwan; Li-Ping Tung, 018 IEEE International Conference on Network Softwarization (NetSoft 2018) - Technical Sessions.

[10] CloudSimSDN: Modeling and Simulation of Software-Defined Cloud Data Centers, Jungmin Son∗, Amir Vahid Dastjerdi∗, Rodrigo N. Calheiros∗, Xiaohui Ji∗, Young Yoon†, and Rajkumar Buyya∗∗Cloud Computing and Distributed Systems (CLOUDS) Laboratory Department of Computing and Information Systems The University of Melbourne, Australia. 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

[11] CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Rodrigo N. Calheiros1, Rajiv Ranjan2, Anton Beloglazov1, Cesar A. F. De Rose3 and Rajkumar Buyya1, Cloud Computing and Distributed Systems (CLOUDS) Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, Australia School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia Department of Computer Science, Pontifical Catholic University of Rio Grande do Sul, Porto Alegre, Brazil

[12] Automatic monitoring management for 5G mobile networks Alberto Huertas Celdrana,, Manuel Gil Pereza, Felix J. García Clemente, and Gregorio Martınez Pereza Dept. Ingenierıa de la Informacion y las Comunicaciones, University of Murcia, 30071 Murcia, Spain Dept. Ingenieria y Tecnologia de Computadores, University of Murcia, 30071 Murcia, Spain.

[13] 2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD). Optimized Cloudlet Management in Edge Computing Environment Efthymios Oikonomou Department of Digital Systems University of Piraeus Piraeus, Greece oikonomouef@unipi.gr Angelos Rouskas Department of Digital Systems University of Piraeus Piraeus, Greece arouskas@unipi.gr

[14] STUDY ON FUNDAMENTAL USAGE OF CLOUDSIM SIMULATOR AND ALGORITHMS OF RESOURCE ALLOCATION IN CLOUD COMPUTING Ekta Rani, Harpreet Kaur, Department of Computer Engineering Punjabi University, Patiala *Assistant Professor, Department of Computer Engineering, Punjabi University, Patiala

[15] Calabrese, Francesco Davide & Wang, Li & Ghadimi, Euhanna & Peters, Gunnar & Hanzo, L. & Soldati, Pablo. (2018). Learning Radio Resource Management in 5G Networks: Framework, Opportunities and Challenges. IEEE Communications Magazine. 56. 10.1109/MCOM.2018.1701031.

[16] Implementation of Virtualization in Software Defined Networking (SDN) for Data Center Networks. Nader F. Mir, Jayashree N. Kotte, and Gokul A. Pokuri nader.mir@sjsu.edu Department of Electrical Engineering San Jose State University San Jose, CA, 95195

[17] A MACHINE LEARNING MANAGEMENT MODEL FOR QoE ENHANCEMENT IN NEXT-GENERATION WIRELESS ECOSYSTEMS Eva

Ibarrola1 , Mark Davis2 , Camille Voisin3 , Ciara Close3 , Leire Cristobo1 1University of the Basque Country (UPV/EHU), Spain 2Dublin Institute of Technology (DIT), Ireland 3OptiWi-fi, Ireland.

[18] The Way to 5G Dr. Whai-En Chen Associate Professor and Director Library and Information Center Dept. of Computer Science and Information Engineering National Ilan University Email: wechen@niu.edu.tw

[19] A scalable machine learning online service for big data real-time analysis. Alejandro Baldominos ; Esperanza Albacete ; Yago Saez ; Pedro Isasi. Published in: 2014 IEEE Symposium on Computational Intelligence in Big Data (CIBD). INSPEC Accession Number: 14855680. Electronic ISBN: 978-1-4799-4540-5. Publisher: IEEE.