Politecnico di Torino

FACULTY OF ENGINEERING
Cybersecurity LM-32

Understanding and Predicting VM Costs in the Multi-Cloud Landscape

Candidate:
Lorenzo Canciani

Supervisor:
Prof. Alessio Sacco

Correlator:
Prof. Guido Marchetto

Academic year 2024-2025

Abstract

The current trend for ICT infrastructure is largely based on new architectures and paradigms, such as Cloud Computing and Virtual Machines (VMs). However, the complicated and obfuscated nature of the price structure across different Cloud Service Providers (CSPs) creates significant challenges for organizations in the quest for improved cost effectiveness and vendor lock-in avoidance. Existing comparison tools currently possess set price data but lack dynamic forecasting capabilities for custom VM builds.

Organizations also need a systematic way to forecast virtual machine expenses across various cloud services based on specific technical specifications. The disjointed nature of the cloud's pricing complicates infrastructure decisions, making informed choices difficult. This lack of clarity creates vendor lock-in situations, making the costs of migration excessively high, as organizations have become deeply embedded in the provider's environment.

This thesis establishes a common system for generating hourly VM costs in prominent CSPs. The process involves retrieving price information from 20 providers using API-based and web scraping methods, normalizing inhomogeneous technical details, and configuring forecasting models to approximate costs for any random VM setting.

The thesis provides a wide-ranging dataset consisting of virtual machines from European, American, and Asian providers and contains detailed specifications such as CPU architecture, memory and storage, GPU capabilities, and locations. The paper describes the methodology used for acquiring the data, the complexities encountered during standardization, an exploratory data analysis of the discovered price trends, and the development of machine learning models for price prediction. The study reveals significant price differences by provider and geography, with European providers typically offering better CPU-intensive configurations. Memory capacity is the most powerful predictor of the price of VMs, followed by vCPU and storage capacity. Geographic concentration reveals that European providers tend to have more similar price strategies than hyperscale providers, such as AWS and Azure, which offer more varied regional pricing.

The work outlines how machine learning can efficiently abstract complex cloud pricing schemes, providing organizations with the necessary cost estimation tools across various suppliers and enabling informed infrastructure decisions. The patterns extracted at the provider-based and geographic levels reveal how mindful provider selection, based on workload definition and geographic coverage, can result in effective cost reductions. The developed analysis tool facilitates the comparison of different provider offerings and addresses a fundamental challenge in multi-cloud cost optimization.

Future research should expand provider coverage, incorporate temporal pricing dynamics through time-series analysis, and develop real-time price forecasting capabilities. Integration with cloud orchestration platforms can facilitate auto-cost-aware resource provisioning, and working with CSPs can enhance the data quality and completeness of the features.

Contents

1 Introduction **12**
- 1.1 Brief overview on Elemento . . . 12
- 1.2 Vendor lock-in . . . 12
  - 1.2.1 Vendor lock-in and VM's pricing . . . 13
- 1.3 Contributions . . . 13

2 Background **15**
- 2.1 VMS in the Cloud . . . 15
- 2.2 Cost Factors in the Cloud . . . 16
- 2.3 How providers expose prices . . . 17
- 2.4 Introduction to Machine Learning . . . 18

3 State of the art **20**
- 3.1 Cloud Mercato . . . 20
- 3.2 Cloud Price . . . 20
- 3.3 Vantage . . . 21

4 Methodology **22**
- 4.1 Data Collection . . . 22
- 4.2 Exploratory Data Analysis . . . 23
- 4.3 Model Development and Training . . . 23
- 4.4 Model Evaluation . . . 24

5 Work development **25**
- 5.1 Data Collection . . . 26
  - 5.1.1 Scraping-based Collection . . . 26
  - 5.1.2 API-based Collection . . . 28
- 5.2 Data Standardization . . . 28
- 5.3 Data Analysis . . . 29
- 5.4 Dataset Imputation . . . 43
- 5.5 Model Training . . . 45

6 Work evaluation **47**
- 6.1 Model Evaluation . . . 47
  - 6.1.1 Linear Regression . . . 47
  - 6.1.2 RBFNN . . . 50
  - 6.1.3 Random Forest . . . 52
  - 6.1.4 XGBoost . . . 55

7 Conclusion and Future Work **62**
7.1 Conclusion 62
7.2 Future improvements 62

Bibliography **64**

List of Figures

2.1 Google Cloud Platform (GCP) interactive pricing page 17

4.1 Methodology overview: process pipeline 23

5.1 Comparison between direct extraction of predefined shapes (e.g. CloudFerro) and component-based reconstruction of virtual machines (e.g. IONOS). 27
5.2 Distribution of VMs in the dataset 31
5.3 Log-log distribution of VM counts across providers with fitted power-law curves 32
5.4 Average price in USD for VM based in Asia per provider 38
5.5 Average price in USD for VM based in Africa per provider 39
5.6 Average price in USD for VM based in Europe per provider 39
5.7 Average price in USD for VM based in the Middle-East per provider . 40
5.8 Average price in USD for VM based in North America per provider . 40
5.9 Average price in USD for VM based in South America per provider . 41
5.10 Average price in USD for VM based in Oceania per provider 41
5.11 Average price in USD for VM by continent per provider 42
5.12 Mutual Information Analysis 43

6.1 Set of scatter plots for Linear Regression by price range 58
6.2 Set of scatter plots for RBFNN by price range 59
6.3 Set of scatter plots for Random Forest by price range 60
6.4 Set of scatter plots for XGBoost by price range 61

List of Tables

5.1 Analyzed Providers 25
5.2 Distribution of Virtual Machines Across Cloud Providers 32
5.3 Top 10 Most Common VM Configurations Across Providers 33
5.4 Top providers for the configuration 0 GPU, 32GB RAM, 8 vCPU, ordered by average price per hour. Prices are in USD. 34
5.5 Top providers for the configuration 0 GPU, 4GB RAM, 2 vCPU, ordered by average price per hour. Prices are in USD. 34
5.6 Top providers for the configuration 0 GPU, 4GB RAM, 16 vCPU, ordered by average price per hour. Prices are in USD. 34
5.7 Maximum resources per provider 35
5.8 Top 5 Most Cost-Effective Small Instance Providers (1–2 vCPU, 0–8GB) 36
5.9 Top 5 Most Cost-Effective Medium Instance Providers (2–8 vCPU, 8–32GB) 36
5.10 Top 5 Most Cost-Effective Large Instance Providers (8+ vCPU, 32+GB) 36
5.11 Top 5 Most Cost-Effective GPU Instance Providers 37
5.12 Imputation Accuracy for Categorical Variables 44
5.13 Imputation MSE for Numerical Variables 44

6.1 Linear Regression metrics for Fold 1 by price range 48
6.2 Linear Regression metrics for Fold 2 by price range 48
6.3 Linear Regression metrics for Fold 3 by price range 49
6.4 Linear Regression metrics for Fold 4 by price range 49
6.5 Linear Regression metrics for Fold 5 by price range 49
6.6 Summary of Linear Regression metrics by price range (mean ± std over 5 folds) 49
6.7 RBFNN metrics for Fold 1 by price range 50
6.8 RBFNN metrics for Fold 2 by price range 51
6.9 RBFNN metrics for Fold 3 by price range 51
6.10 RBFNN metrics for Fold 4 by price range 51
6.11 RBFNN metrics for Fold 5 by price range 51
6.12 Summary of RBFNN metrics by price range (mean ± std over 5 folds) 52
6.13 Random Forest metrics for Fold 1 by price range 53
6.14 Random Forest metrics for Fold 2 by price range 53
6.15 Random Forest metrics for Fold 3 by price range 54
6.16 Random Forest metrics for Fold 4 by price range 54
6.17 Random Forest metrics for Fold 5 by price range 54
6.18 Summary of Random Forest metrics by price range (mean ± std over 5 folds) 54

6.19 XGBoost metrics for Fold 1 by price range 56
6.20 XGBoost metrics for Fold 2 by price range 56
6.21 XGBoost metrics for Fold 3 by price range 56
6.22 XGBoost metrics for Fold 4 by price range 56
6.23 XGBoost metrics for Fold 5 by price range 57
6.24 Summary of XGBoost metrics by price range (mean ± std over 5 folds) 57

Chapter 1

Introduction

1.1 Brief overview on Elemento

Elemento [7] is an Italian deep-tech startup revolutionizing the way organizations build and oversee their cloud infrastructure. Elemento is adaptable, straightforward, and long-term scalable, allowing businesses to establish cloud environments customized to their specific needs in public, private, or hybrid cloud settings.
Elemento's platform is built on two patented technologies: the Cloud Network [6] and AtomOS [5], which is a high-performance hypervisor based on KVM. They offer a single, user-friendly interface to control and coordinate compute, storage, and network resources across various environments. IT staff can control, expand, and operate public and private cloud infrastructures through a single, cohesive platform with Elemento. Elemento supports native hybrid and multi-cloud operations, making resources from over 230 global data centers, handled by the largest five public cloud providers, available to organizations. This enables complete integration of on-premise or private infrastructure.
Through cloud abstraction, Elemento makes cutting-edge cloud infrastructure easily accessible, provides a robust, elastic foundation that enables businesses and service providers to migrate legacy systems, expand international applications, or develop cloud-native ones as their needs evolve, avoiding vendor lock-in.

1.2 Vendor lock-in

Vendor lock-in refers to a cloud computing scenario where the cost or complexity of switching to an alternate vendor is so high that the customer has no other option but to stick with the original chosen vendor [9]. This dependence can result in limited other options for the customer to switch to another provider, even when the current provider is performing poorly.
Vendor lock-in may be caused by a cloud provider using proprietary technologies, formats, or interfaces that do not work with other providers effortlessly, thus preventing the customer from moving their data and applications [29]. It may also result from sole license terms, complex integration requirements, or custom installations that tie the customer to the targeted provider's ecosystem. This rigidity and non-portability can compromise the customer's ability to adapt to changing needs or leverage competitive solutions, which can impact cost-effectiveness and innovation.

1.2.1 Vendor lock-in and VM's pricing

The market for virtual machines (VMs) plays a crucial role in cloud computing vendor lock-in dynamics. Although lock-in is often described in terms of technical dependencies on proprietary services, economic ones are equally limiting.
Cloud providers offer multiple VM types with varying combinations of CPU, memory, storage, and network. Pricing by providers is complex, non-uniform, and opaque. Once an enterprise is locked into a provider and weights workloads to specific VM instances, it is challenging to compare or migrate to different providers, requiring a significant amount of work and cost. This is particularly true when long-term discounts or reserved instances are utilized, and they save costs for less flexibility.
Moreover, cloud providers periodically modify their pricing models, and subscribers within a specific ecosystem are often forced to absorb the impact of such changes without having the practical option to shift elsewhere. In addition, billing granularity variability, licensing models, and geographically variable pricing also complicate cross-provider comparison.
In this case, VM expense is a cost control issue as much as a mobility lock-in structural hurdle. The more a user is tied to provider-specific compute offerings, the greater the expense of re-configuring or relocating workloads elsewhere, even if comparable VM choices exist elsewhere at a lower price. Therefore, pricing rigidity contributes to the lock-in effect, converting one-time cost savings into future financial obligations.
It is thus essential to understand and forecast VM costs across providers to control cloud flexibility. It enables companies to foresee cost traps and quantify the economic cost of provider dependency.

1.3 Contributions

Several websites, such as *Cloud Mercato* [8], gather price information from a variety of cloud service providers. While useful for the comparison of available offerings, these types of tools do not usually facilitate dynamic estimation of price based on user-specified virtual machine (VM) parameters. In practice, if a user desires to provision a VM with specific hardware requirements, e.g., 2 CPU and 2GB of RAM, someone has to look up every provider's price webpage or API to discover a suitable option. The process is labor-intensive and error-prone, especially in light of the vast and changing landscape of cloud service costs.
The primary contribution of this thesis is the design and integration of a machine learning model that predicts the hourly cost of a virtual machine based on a full set of configuration and provider-specific variables.
The model input features used are a full array of technical specifications and metadata, including:

- virtual CPU cores and memory size,
- storage type, bus, and capacity,
- presence of GPU, vendor, model, and VRAM capacity,
- CPU attributes, such as vendor, model, architecture, family, and clock rates,

- operating system family, version,
- shared or dedicated CPU,
- geographic and provider-specific areas

By including and analyzing this diverse set of characteristics, the model is able to capture price variations between providers and configurations. This approach offers a scalable and adaptable solution to support decisions in choosing cloud infrastructure and serves as a basis for integration into automated deployment frameworks and orchestration systems.

Chapter 2

Background

2.1 VMS in the Cloud

Cloud computing refers to the delivery of computer resources, such as storage, networking, and virtualized processing power, over the internet on a pay-as-you-go basis [3]. Virtual Machines (VMs) are at the center of the majority of cloud services, providing a simulation of physical computer behavior by software abstraction. VMs are important as they enable Cloud Service Providers (CSPs) to share hardware resources among multiple users, scale workloads, and provide isolated environments for execution.

Different types of VMs

Cloud providers offer various types of virtual machines (VMs) tailored to meet user needs. The categories related to usage are:

1. General-purpose VMs: balanced CPU-to-memory ratio, suitable for web servers, small databases, and development environments.

2. Compute-optimized VMs: high CPU performance relative to memory; ideal for batch processing, scientific modeling, and game servers.

3. Memory-optimized VMs: large amounts of RAM; suitable for in-memory databases, big data analytics, and caching.

4. GPU-enabled VMs: equipped with Graphics Processing Units for deep learning, video rendering, and high-performance parallel workloads.

5. Specialized processors: some VMs use ARM-based CPUs for power efficiency, or FPGA-based instances for custom hardware acceleration.

6. Storage-optimized VMs: designed for workloads requiring high I/O throughput, such as large transactional databases or log processing.

However, it is possible to simplify making a distinction between VMs with GPUs and VMs without GPUs.

Different locations of datacenters, concept of "Region"

VM virtualization is achieved through software called a hypervisor (e.g., AtomOS [5] or VMWare [37]). Therefore, it is possible to say that a VM is a virtualization of hardware. Having all the hardware for the infrastructure in a single geographical area is not an ideal solution, which is why in the cloud, there is the introduction of the concept of *region*.
The *region* of a Cloud Service Provider (CSP) is a specific geographical area where the physical data centers that will host the provider's infrastructure are located. The use of different regions by the same provider is crucial for enhancing the end-customer experience.
The choice of region has an impact on latency [17] as well as on the cost of cooling the machines [1] and regulatory aspects such as GDPR and single-region data residency. Another important aspect to consider is the risks associated with natural and political disasters [4].

2.2 Cost Factors in the Cloud

Prices in the cloud world are of essential importance, as they determine the success or failure of a business.
Standard costs can mainly be categorized as follows:

1. Infrastructure: cost of individual server components, cost of rental space, network cost, cooling cost

2. Data transfer: cost of transferring data from the provider's network to your network (egress traffic)

3. Software licenses: many VMs come with third-party software that requires licenses. An example is AWS pricing for EC2 with and without Linux with SQL Enterprise, which are $0.1664 and $1.6664 respectively [2]

Another distinction that can be made concerns external factors and internal factors specific to individual cloud providers that can impact costs.

External factors External factors include the supply chain, an example of which is the COVID-19 lockdown that blocked many supply chains, including that of semiconductors [26]; energy prices; and the presence of a monopoly by large providers that prevents small providers from scaling.

Internal factors Internal factors, on the other hand, relate more to the management of the individual provider and can be optimized by following a proper business model. Examples of internal factors include egress costs, customer service costs, and proper resource allocation (e.g., correctly balancing the workload among customers).

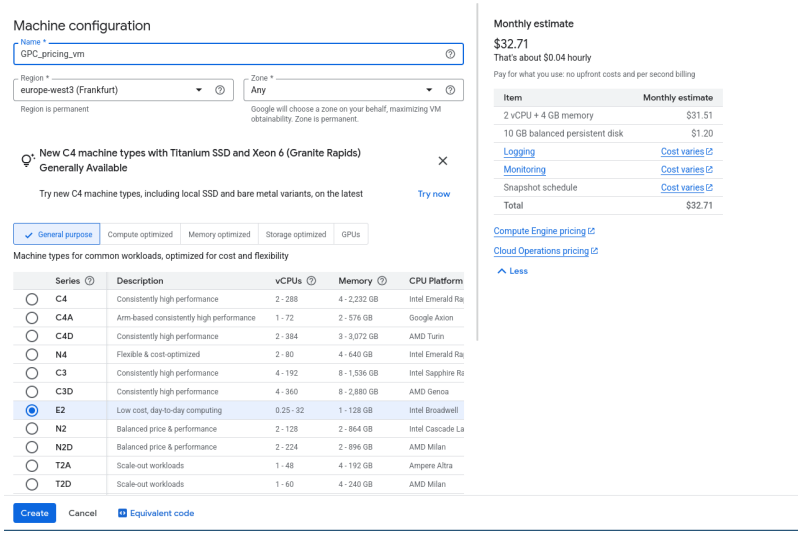


Figure 2.1: Google Cloud Platform (GCP) interactive pricing page

2.3 How providers expose prices

Returning to the issue of monopoly discussed in 2.2, CSPs do not easily disclose the prices of their services. This makes it difficult for competitors to determine the optimal price for a given service to remain competitive, and it also hinders end users from comparing different cloud provider prices.
There are three main ways in which cloud providers display prices for their services: via API (authenticated and unauthenticated), via pricing pages (which can be price tables or JSON files), and via human customer service.

API

When a CSP provides APIs for retrieving its price catalog, it provides endpoints that may or may not be authenticated to which HTTP calls can be made, an example is UpCloud [23]. This approach allows prices for individual services to be obtained quickly and provides good flexibility for the end user because it allows prices to be collected repeatedly and automatically via scripting.
In this category are also included SDKs such as Boto3 [34].

Pricing Pages

The use of pricing pages by providers requires users to manually consult a price catalog, an example of this is the IONOS price catalog [20], or to interactively configure a cloud service to estimate its price, as shown in fig. 2.1 for GCP.

Human Customer Service

The third way in which cloud providers expose their prices is through customer service. Some cloud providers require users to contact their sales department for certain types of services, which includes all the delays that human interactions can cause.

2.4 Introduction to Machine Learning

Since this thesis aims to develop a machine learning model for estimating prices based on specifications for VMs provided by cloud providers, it is important to provide a brief introduction to machine learning.
Machine learning (ML) allows computers to learn and make decisions without being explicitly programmed. In order to find patterns and forecast fresh data, it entails putting data into algorithms.
Without being specifically designed for every situation, a machine "learns" by finding patterns in data and expanding its capacity to carry out particular tasks. Based on the information they are given, this learning process aids robots in making precise predictions or conclusions. In contrast to traditional programming, which uses fixed instructions, machine learning enables models to grow and change over time based on provided data.
There are three different types of machine learning techniques:

- Supervised learning: using labeled data, supervised learning creates a model with known right outputs for each input. By contrasting its predictions with these accurate responses, the model gains knowledge and gets better over time. It is applied to situations involving both regression and classification.

- Unsupervised learning: unlabeled data, which lacks categories or accurate responses, is used in unsupervised learning. Finding the data, hidden patterns, similarities, or groups on its own is the model's responsibility. In situations when data labeling is challenging or impossible, this is helpful. Clustering and association are common uses.

- Reinforcement learning: through interaction with an environment, Reinforcement Learning teaches an agent to make judgments. Agents learn via trial and error rather than being given the solutions. They receive incentives for good behavior and punishments for bad behavior. It gradually devises a plan to optimize benefits and accomplish objectives. This method works well for challenges involving sequential decision-making, such as autonomous systems, robotics, and gaming.

Regression vs Classification

The main distinction between classification and regression, two of the main tasks in supervised machine learning, is the kind of output: classification works with discrete outcomes (such as categories), while regression deals with continuous values (such as price).

Basic Models

There are some basic machine learning models. Below, there is a brief introduction to the main "traditional" models.

- Linear Regression: Linear Regression maps data points with the best linear functions after learning from labeled datasets. This method may be used to make predictions. It presumes a linear relationship between the input and output, indicating that any variation in the input will result in a corresponding change in the output at the same rate. A straight line is used to depict this relationship [12].

- Logistic Regression: Logistic Regression predicts the likelihood that an input belongs to a certain class, as opposed to linear regression, which predicts continuous values. When it comes to binary classification, the result can fall into one of two categories, such as True/False, Yes/No, or 0/1. It transforms inputs into a probability value between 0 and 1 using the sigmoid function [14].

- Decision Tree: a Decision Tree is a structure that resembles a tree, with each internal node standing for a feature-based decision, each branch for an outcome, and each leaf node for a final choice or forecast. The stages a decision tree takes are to divide the data, pick the best feature, create a decision node, create subgroups from the dataset, and recursively repeat the procedure until a stopping condition, such as maximum depth or pure classification, is satisfied [11].

- Random Forest: Random Forest employs a large number of decision trees. It is an ensemble learning approach since each tree examines distinct random portions of the data, and the results are aggregated by voting for classification or averaging for regression. This aids in increasing precision and decreasing mistakes [13].

- Boosting Methods: Boosting is an ensemble learning strategy that builds a strong classifier by successively combining several weak classifiers. It is accomplished by utilizing training data to train a model, which is then assessed. The next model is based on the prior one and attempts to fix its flaws. Until the whole training data set is properly predicted or a certain number of iterations is reached, this process is repeated, and new models are added [35].

Chapter 3

State of the art

As discussed in section 2.3, providers have different ways of exposing prices for their services to users. In general, we have identified three different methods, which involve a lot of work for users who want to compare offers from different cloud providers. To reach this goal, portals have been created that collect prices from different CSPs and display them via their APIs to make the comparison process easier for end users.
However, these kinds of portals collect prices directly from cloud providers and provide them to the end user. The aim of this thesis is slightly different, as it sets out to build an ML model to estimate the prices of IaaS services for an arbitrary number of CSPs in an arbitrary number of regions.
At present, there are no publicly available references to a model for estimating prices for cloud VMs.
The main portals offering standard pricing exposure for cloud services are Cloud Mercato [24], Cloud price [10], and Vantage [36].

3.1 Cloud Mercato

Cloud Mercato [24] is a portal that collects and compares offers from leading cloud service providers. The platform uses cloud service provider (CSP) pricing catalogs to obtain information on the prices and features of Infrastructure as a Service (IaaS) offerings, presenting them in a uniform and easily searchable format.
Of the three solutions mentioned, it is the most comprehensive as it collects data from around fifty providers [25].

3.2 Cloud Price

Users can compare virtual machine instances from Google Cloud Platform (GCP), Microsoft Azure, and Amazon Web Services (AWS) using the CloudPrice platform [10]. The platform makes it possible to compare the technical details and costs of various instances while accounting for different payment models (pay-as-you-go, reserved, and spot) and geographical locations. The platform makes it easier to integrate data into analysis tools or automatic comparison models by enabling the download of the entire catalog of services in JSON format.

3.3 Vantage

Vantage [36] is a platform for comparing virtual machine instances and cloud services from major public cloud providers, with a focus on Amazon Web Services (AWS) and Microsoft Azure. Compared to Cloud Price and Cloud Market, Vantage also covers a wider range of SaaS services, thus offering a more comprehensive overview of the cloud market.
The platform also enables users to link their accounts from different CSPs, allowing them to track their cloud service expenses on a single platform.
Similar to CloudPrice, Vantage allows users to download the entire service catalog in JSON format, facilitating the integration of data into analysis tools or automatic comparison models.

Chapter 4

Methodology

The process of developing and training a model can be divided into five macro areas. The first concerns data collection; data is the basis of every machine learning system, and without good-quality data, it is not possible to build or train any model. The first phase is therefore the collection of data from cloud providers.
Next, the collected data must be analyzed to evaluate any outliers and understand if there are specific patterns, for example, if VMs cost more in certain regions or if there are features that have a greater influence on the price for a given VM configuration. Data analysis is followed by the selection of features to be used for price estimation. Some of the collected features may not be relevant or important in determining the price, so it is essential to make the right choice. Incorrect choices can lead to wasted resources by the model, as well as incorrect predictions.
For the development of the model, it is necessary to figure out the best solution for the created dataset. This means that it is necessary to test different models to find the best fit, which could be expensive in terms of hardware usage and time, but it depends on the number of features and the number of points in the dataset. The optimal solution would be to find already implemented models that have a good fit on the created dataset.
The model evaluation consists of assessing the regression obtained from the model in relation to metrics such as accuracy and the number of errors made. At this stage, different models can also be evaluated to verify which one performs best for the use case.
Some of the phases described above will be analyzed in greater detail below.

4.1 Data Collection

Data collection is a crucial part of the system, as shown above. In fact, cloud providers are reluctant to provide their prices in a transparent and easily accessible manner, but good data is essential for an ML model to work well. To simplify the collection process, the idea is to have a *getter.py* file for each provider analyzed, containing the script to collect prices for that provider. This allows prices to be collected with the same command regardless of the provider.
In addition to collection, it is necessary to find a way to standardize data from different cloud providers and to compare them. Some CSPs expose information that others do not; for example, some providers may disclose details about the hardware underlying their virtual machine, while others may not provide any information,

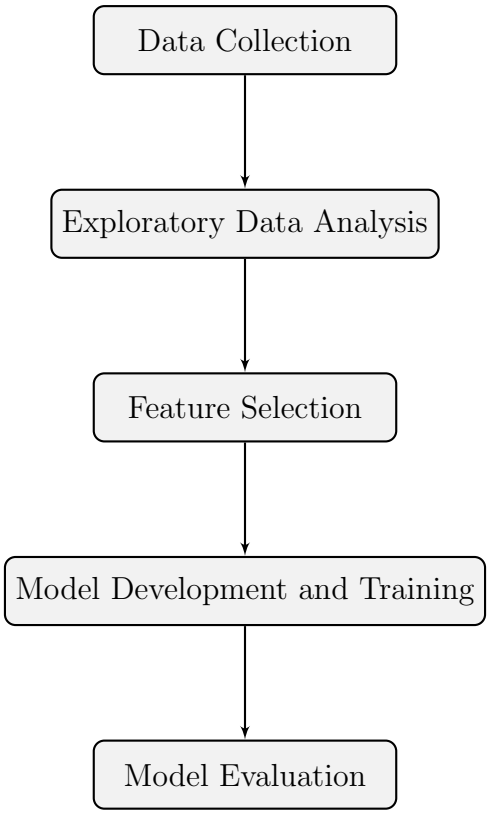


Figure 4.1: Methodology overview: process pipeline

explaining that they may modify the hardware without notifying the user, as long as the service's performance remains unchanged.

4.2 Exploratory Data Analysis

Analyzing data after collecting it allows the elimination of any outliers, identifies specific patterns, and determines whether there are correlations between the different features collected. To perform this assessment, the quickest solution is to plot the individual features to highlight any issues, eliminate outliers, and finally use the mutual information [22] to understand which features have the greatest impact on determining a particular characteristic of the dataset; in the case of this thesis, the price for the cloud virtual machine.
It is also important to understand the distribution of the dataset to avoid having a dataset that is biased towards a particular type of virtual machine, such as a virtual machine with a GPU or a virtual machine without a GPU.

4.3 Model Development and Training

These chosen attributes are subsequently utilized for generating regression models that are appropriate for the prediction of continuous measures like prices. A few different algorithms can be tried to find the best way. Popular picks for such a problem are linear regression, decision tree-based models, random forests, and gradient boosting models. It is important to understand that building from scratch,

rather than using available implementations, might not be the most efficient solution. The construction of a model from scratch requires significant amounts of time and computational power, and it may inject extraneous errors into the algorithm and optimization routine executions. On the other hand, well-respected libraries like `scikit-learn` [30] and `XGBoost` [18] offer optimized, well-proven, and widely used implementations that are well equipped to handle massive datasets efficiently and provide consistent performance.

4.4 Model Evaluation

Following the training process, the models undergo an evaluation process whose aim is to ascertain the performance and reliability of the models. In the case of regression issues, popular measures utilized in measuring predictive precision include Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared [31]. Such measures give useful information regarding the average variation between the predictions and the corresponding measures and the percentage of variance explained by the model.
Comparing against several models enables to determine the algorithm that generalizes best to unseen data.
Using well-established libraries reduces the chance of errors and ensures that the evaluation measures correctly reflect the predictive powers of the model and not the side effects of an unoptimized implementation. Cross-validation techniques are used to reduce overfitting and ensure that performance measures are reliable on different portions of the dataset.

Chapter 5

Work development

Providers included in the analysis

Provider	Nationality	Data collection method	Europe
Alibaba Cloud	China	API	No
Aruba Cloud	Italy	Scraper	Yes
Amazon Web Services	USA	API	No
Azure	USA	API	No
Bluvault	Saudi Arabia	Scraper	No
Catalystcloud	New Zealand	Scraper	No
CloudFerro	Poland	Scraper	Yes
Exoscale	Switzerland	Scraper	Yes
Gigas	Spain	Scraper	Yes
Google Cloud Platform	USA	Scraper	No
Gridscale	Germany	Scraper	Yes
Hetzner	Germany	API	Yes
Hidora	Switzerland	Scraper	Yes
Hostinger	Lithuania	API	Yes
IONOS	Germany	Scraper	Yes
Leafcloud	Netherlands	Scraper	Yes
Leaseweb	Netherlands	API	Yes
Linode	USA	API	No
Scaleway	France	API	Yes
UpCloud	Finland	API	Yes

Table 5.1: Analyzed Providers

The choice of the providers to include in the thesis is related to different features, such as the nationality of the provider; for this reason, most of the CSPs analyzed are based in Europe. The table 5.1 enumerates the providers used during the data collection phase and gives information about the method used for data collection, as discussed in section 2.3.

5.1 Data Collection

The purpose of data collection is to create a dataset of virtual machines available on the cloud providers considered in the thesis.
To achieve this goal, each virtual machine will be managed as a JSON object, so each provider will have a *getter.py* file that will output a vms.json file, a collection of JSON objects, for each region in which it has VMs available.
Now, the implementations of the two main collection methods used will be analyzed: scraping the provider's website and using APIs exposed by the provider.

5.1.1 Scraping-based Collection

Concerning data collection through scraping, it can be divided into two macro areas: direct price collection given a pre-established VM configuration (shape), or price collection for individual components and then arbitrary configuration generation. In both cases, data collection was carried out using the web browser interaction features of the Selenium library [32]; since scraping is customized on the website of each provider, it is understandable that it is not possible to analyze each provider in detail, but rather to analyze only two providers to understand how data collection is carried out in both modes.

Direct Pricing Collection

CloudFerro, a Polish cloud provider, publishes pricing information by means of predefined tables that directly describe the available virtual machine types. Each entry in these tables [33] gives the characteristics of the virtual machine, namely the number of virtual CPUs, the memory, the storage and the corresponding prices in three different charging modalities (per hour, per month, and per year).
The availability of such predefined shapes makes the gathering of such information very simple. The tables can be interpreted as structured datasets in which each row already corresponds to a complete virtual machine configuration. In this way, the fundamental attributes of the offering are readily accessible: instance identifier, computational capacity, memory resources, storage type and size, as well as price levels.
For the creation of the dataset, the values extracted from the provider's tables are harmonized. This involves, for example, expressing memory values in a consistent unit, standardizing storage information according to type and interface, and associating processors with their respective vendor and architecture family. The euro prices are also converted to USD to have a common currency to allow comparison with other providers.
In this way, the predefined shapes of CloudFerro can be easily merged into a larger dataset that is consistent with the data coming from providers who adopt different strategies in how to display their offers. The final result is a unified and comparable view of virtual machine flavors from heterogeneous sources.

Component-based Pricing Collection

IONOS adopts a pricing strategy [21] that differs substantially from providers such as CloudFerro. Instead of offering predefined virtual machine types, its documentation reports unit costs for the main resources that compose a virtual machine. These include the hourly price per virtual CPU (with distinctions across processor families such as AMD EPYC or Intel Xeon generations), the cost per gigabyte of memory, the price of different storage media (HDD SATA, SSD SATA, SSD NVMe), and additional fees related to operating systems or enterprise software licences.
Such a structure requires an indirect reconstruction of the virtual machines. Rather than relying on predefined shapes, the methodology builds synthetic configurations by combining the available components. The process can be described as follows:

1. for each geographical region, the available processor families are identified;
2. realistic combinations of processing power, memory size, and storage capacity are generated within feasible ranges;
3. the total cost of each configuration is derived as the sum of the CPU price, the proportional memory cost, and the cost associated with the chosen storage type and quantity.

By adopting this component-based approach, it becomes possible to approximate the portfolio of virtual machines that IONOS would offer if it published predefined shapes. This strategy enables comparability with providers that disclose explicit instance types. In this way, a uniform representation of different providers can be achieved, despite the heterogeneity in the way pricing information is presented.

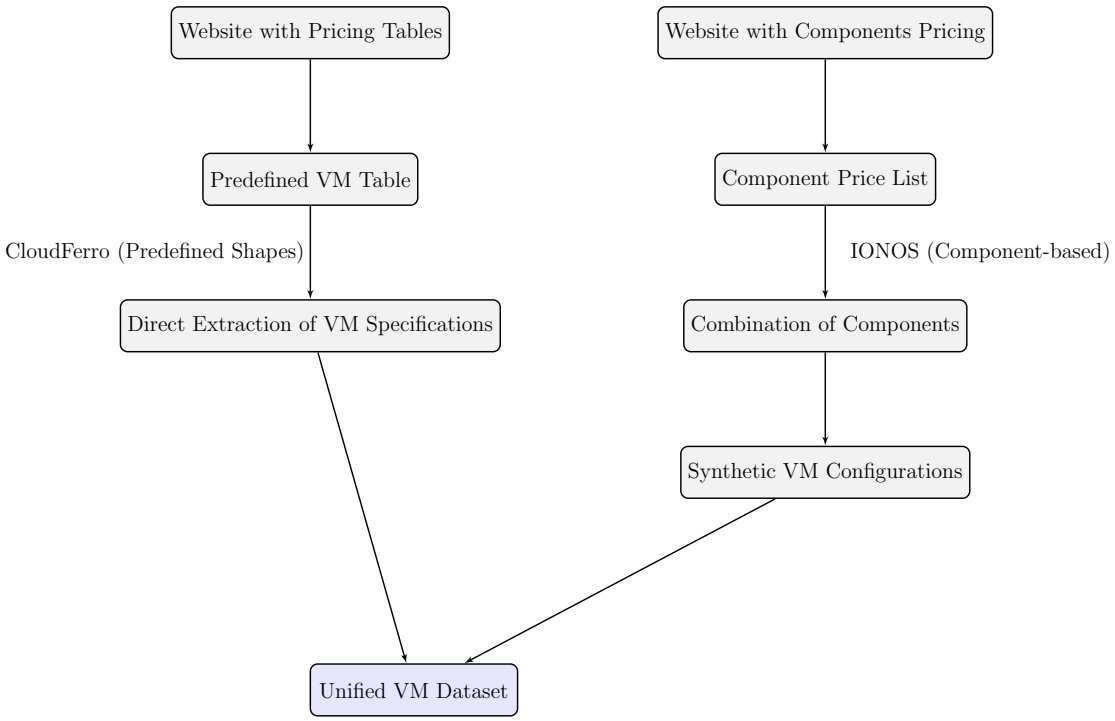


Figure 5.1: Comparison between direct extraction of predefined shapes (e.g. CloudFerro) and component-based reconstruction of virtual machines (e.g. IONOS).

5.1.2 API-based Collection

Opposite to scraping, this process is dependent on the official APIs of the cloud provider, which provide both configuration and price in a formatted way. The flow of work is different for each provider, but for example, for AWS, it gets started by listing all the regions in which services can be offered because instance type and price tend to vary significantly by location. After identifying regions, the system requests each region for a detailed description of the instance types offered. The details include processor type, like model, architecture, and clock speed; memory and disk specifications; network attributes; GPU numbers and types; and pre-installed software that tends to affect licensing.
Information on prices and configurations is collected, and the raw features from these are then normalized and harmonized to ensure uniformity throughout various geographical regions, correcting for differences in units and nomenclatures. Particular attention is given to details such as the classification of GPU suppliers and models into uniform categories. The careful process ensures that the output dataset is both accurate and possible to compare.
The workflow can be thought of as a three-stage process. All relevant price information is first accrued. Technical specifications for each of the instance types are next obtained. The two streams of information are finally integrated such that there is an extensive aggregation of all offerings. Certain use cases, such as instances with more than one GPU, unconventional storage offerings, and region-based constraints, have particular attention given to them, such that the dataset accurately mirrors the actual capability and cost of each resource.
The outcome is a dataset of virtual machine information that binds virtual machine shapes with prices.

5.2 Data Standardization

Different providers expose different specifications; for this reason, it is essential to find a way to standardize the output, otherwise, it is not possible to compare and use the data to train a machine learning model.
To reach this goal, a good way is to have a class with a super set of all the attributes exposed from all the providers and map all the service's data to that class. If there is no information for that feature, the class field will be filled with a default null value.
When it comes to GPUs, providers are prone to use different names for their GPUs without following any convention. To make virtual machines comparable, it is necessary to find a mapping that takes into account the number of slices of the GPU dedicated to the machine, the GPU vendor, and the GPU model. The solution to this situation is a mapping based on PCI ID.
Mapping based on PCI ID is a suitable solution because it is based on a mechanism that is simple and future-proof. Each PCI or PCIe device is identified by the firmware and operating system based on two basic numbers: Vendor ID, i.e., the manufacturer, and Device ID, i.e., the model of the individual device. These identifiers are standardized within the PCI specification. The Device IDs are chosen by the vendor company for every respective device within their own namespace; on the other hand, the Vendor IDs are assigned and managed globally by the central

administration of the PCI-SIG consortium, which ensures their uniqueness on the international level among vendors; therefore, the combination of the Vendor ID and the Device ID provides a reliable and unique identity of every PCI or every PCIe device.
The mapping used in this thesis for GPUs can be found on Elemento's GitHub [19].

5.3 Data Analysis

The dataset has a total of 361168 virtual machines collected across 20 providers. Below is the structure of a single record with an example of mapping to numerical or categorical values.

Dataset Features

Provider alibabaclooud=0, azure=1, arubacloud=2, – Cloud service provider

Service Type vm=0 – Type of service

Instance Type t2.micro, n1-standard-4 – Specific instance name based on provider

Global Region Europe_United-Kingdom=17, Asia_China=0 – Macro geographic area

Provider Region eu-west-1, us-west-2 – Specific provider datacenter region

OS Family linux=0, windows=1 – Operating system family

OS ubuntu=0, fedora=1, windows_server=7 – Specific distribution

OS Version 2016standard=12, 20.04=1, stream9=4 – Operating system version

Pre-installed Software sqlstd=0, sqlent=2 – Pre-installed software packages

OS License included=1 – Operating system is included or not in VM price

CPU Vendor intel=0, amd=1, ampere=2 – CPU Vendor

CPU Generation skylake=0, milan=8, genoa=9 – CPU Generation

CPU Family xeon=0, epyc=1, altra=2 – CPU macro family

CPU Model platinum=0, gold=1, silver=2 – CPU model line

CPU Architecture 64, 32 – CPU Architecture

CPU ISA arm=1 – CPU instruction set

CPU Frequencies (GHz) Max: 3.5, 4.0; Min: 1.0, 1.5; Avg: 2.5, 3.0 – CPU frequency, each tier is a feature

CPU Shared No=0, Yes=1 – CPU shared or not

vCPU 4, 8, 16 – Number of CPU cores

RAM (GB) 8, 16, 32 – Size of RAM memory in GB

Network Speed (Mbps) 100, 1000 – Network interface speed

Included Traffic (TB) 1, 10 – Traffic included in the price of the vm

Storage Capacity (GB) 100, 500 – Storage capacity included in GB

Storage Bus sata=0, nvme=1 – Storage Bus

Storage Type ssd, hdd – Storage Type

Storage Speed (GB/s) 0.5, 1.0, 2.5 – Stoeage speed in GB/s

GPU Count 0, 1, 2 – Number of GPUs attached to VM

GPU Vendor nvidia=1, amd=2, habana=3 – GPU Vendor

GPU Model a100=8369, t4=7864, v100=7606 – GPU Model mapped to *int(PCIid,16)*

GPU VRAM (GB) 16, 32, 40 – GPU VRAM in GB

IPv4 Included=1, Not Included=0 – Availability of IPv4

IPv6 Included=1, Not Included=0 – Availability of IPv6

Hourly Price (USD) 0.05, 0.20 – VM price per hour in USD

Monthly Price (USD) 10, 20, 100 – VM price per month in USD

Yearly Price (USD) 1 year: 100, 200; 2 years: 180, 360; 3 years: 250, 500 – VM price per year in USD, each category is a feature

Distribution of VMs in the dataset The breakdown of the dataset between virtual machines with GPUs and virtual machines without GPUs is shown in Figure 5.2. This data clearly indicates that few providers offer machines with GPUs.

This allocation aligns with the inherent patterns of cloud workloads, preferring CPU consumption to GPU usage. Academic studies uncover that CPU-intensive applications rank among the most utilized types in cloud environments [27, 28], such as traditional enterprise services, web services, and database activities, which act as the foundation of most cloud infrastructures. The common presence of these workloads, which benefit more from CPU scalability than from GPU acceleration, explains the motivation of cloud providers' resource allocation policy, accounting for the observed shortage of virtual machines with GPU support.

Number of VMs per provider An analysis of the generated dataset indicates that the instance count varies by four orders of magnitude, with a value ranging from Amazon Web Services' sizable 183,174 virtual machines down to niche providers such as Hidora with a mere 28 instances, as seen from Table 5.2. Such strong distribution bias is also illustrated in Figure 5.3, where providers were ordered by their VM count with logarithmic axes. There is a first part comprising the first seven providers who closely follow a power-law distribution [15] with an exponent of -1.18, but after the seventh provider, a much steeper curve with an exponent of -6.29 is observed.

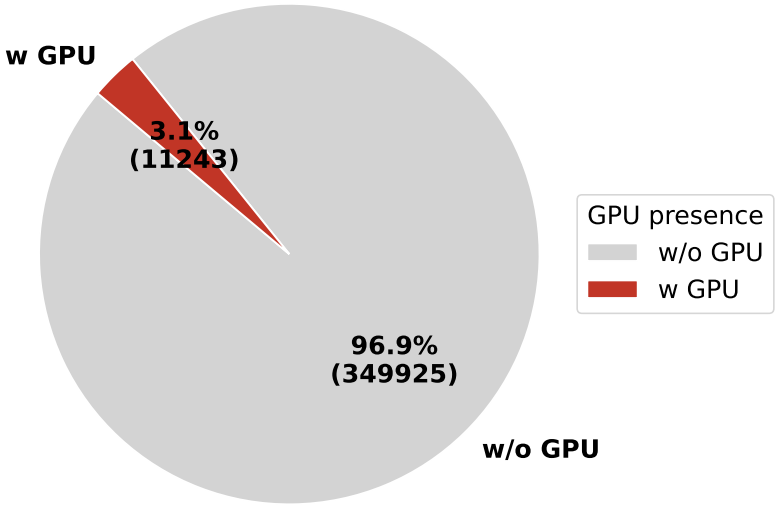


Figure 5.2: Distribution of VMs in the dataset

Such behavior is a close match to Zipf's distribution [16], which suggests that a high number of empirical distributions have their frequency or size to be inversely proportional to their rank. Under such a scheme, a few leading providers contain a dominant majority of VMs, but such a long tail is formed by the remaining providers with much smaller infrastructures Table 5.2.

Cloud Provider	VM Count
AWS	183,174
Scaleway	39,643
Leaseweb	37,296
Hetzner	23,166
Alibaba Cloud	22,396
UpCloud	20,332
Google Cloud	13,776
Azure	12,536
Leafcloud	3,069
Ionos	1,750
Gridscale	1,600
Linode	1,117
Catalyst Cloud	565
Bluvault	91
Arubacloud	208
Exoscale	169
CloudFerro	159
Hostinger	48
Gigas	45
Hidora	28

Table 5.2: Distribution of Virtual Machines Across Cloud Providers

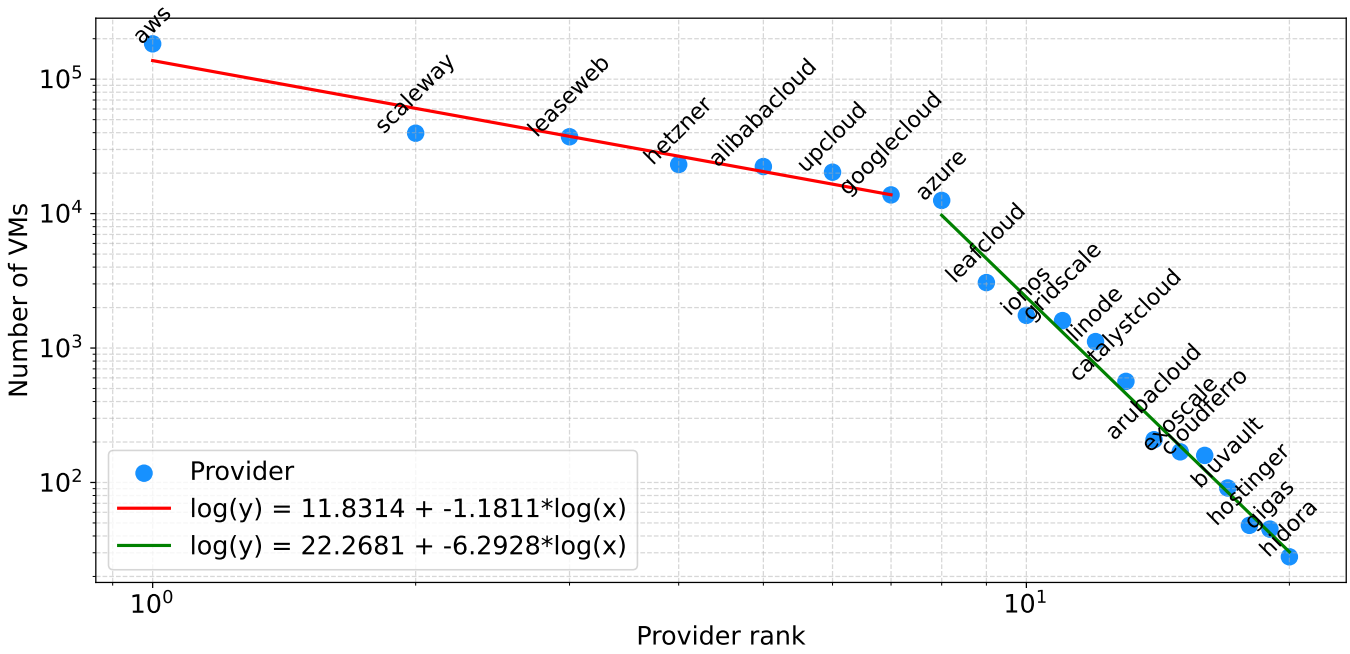


Figure 5.3: Log-log distribution of VM counts across providers with fitted power-law curves

Most Common Hardware Configurations Table 5.3 summarizes the ten most common VM configurations across providers. Most widely available instances do not include GPUs, reflecting a focus on CPU and RAM-centric workloads. Configurations such as 32 GB RAM with 8 vCPUs and 4 GB RAM with 2 vCPUs are offered by the largest number of providers, indicating standardization of mid-range and entry-level VMs.
Hourly prices for identical configurations vary substantially; this variation can be partly explained by differences in included storage and the pre-installed software: instances with larger storage allocations tend to be more expensive, as machines with extra software such as SQL Enterprise (section 2.2), although provider-specific pricing policies and regional availability also contribute.

GPU	RAM	vCPU	# Providers	Min Price	Max Price	Avg Price
0	32	8	17	0.0744	11.5570	0.9459
0	4	2	17	0.0051	1.5249	0.2036
0	16	4	16	0.0372	11.5142	0.5273
0	8	4	16	0.0093	11.4801	0.4966
0	64	16	14	0.1489	11.6426	1.7003
0	8	2	14	0.0186	11.4972	0.2734
0	32	16	14	0.0372	11.5380	1.5674
0	16	8	13	0.0186	11.4962	0.9037
0	64	32	13	0.4302	16.2608	3.6082
0	128	32	12	0.2977	19.5704	3.2663

Table 5.3: Top 10 Most Common VM Configurations Across Providers

Provider-Specific Pricing Analysis for Standard Configurations After identifying the most common VM configurations on most of the cloud providers considered, it is interesting to analyze which providers offer the lowest prices for these configurations. Specifically, we will analyze the three most common configurations, i.e., the first three rows of table 5.3.
The results are shown in tables 5.4, 5.5, and 5.6, respectively, and show the minimum, maximum, and average prices in USD. These prices are influenced not only by the number of CPU cores and RAM, but also by the region and the amount of storage that the provider makes available on the configuration chosen by the user. This explains the price variation within the same provider, so in this case, it is significant to evaluate the average price.

Provider	Min Price	Max Price	Avg Price
Hostinger	0.0822	0.0822	0.0822
Arubacloud	0.1528	0.2977	0.2253
Scaleway	0.1736	0.7245	0.2720
Linode	0.2880	0.4030	0.2936
Upcloud	0.2262	0.6860	0.3539
Google Cloud	0.2680	0.4032	0.3565
Exoscale	0.3171	0.4242	0.3796
Cloudferro	0.3804	0.4743	0.4111
Hidora	0.3823	0.4642	0.4232
Gridscale	0.2099	0.7160	0.4254

Table 5.4: Top providers for the configuration 0 GPU, 32GB RAM, 8 vCPU, ordered by average price per hour. Prices are in USD.

Provider	Min Price	Max Price	Avg Price
Arubacloud	0.0294	0.0657	0.0475
Upcloud	0.0263	0.1194	0.0517
Linode	0.0360	0.0910	0.0525
Cloudferro	0.0374	0.0623	0.0526
Exoscale	0.0544	0.0544	0.0544
Hidora	0.0478	0.0751	0.0614
Google Cloud	0.0335	0.0851	0.0698
Azure	0.0426	0.1010	0.0807
Alibabacloud	0.0254	0.2240	0.0853
Scaleway	0.0282	0.2566	0.1068

Table 5.5: Top providers for the configuration 0 GPU, 4GB RAM, 2 vCPU, ordered by average price per hour. Prices are in USD.

Provider	Min Price ($/h)	Max Price ($/h)	Avg Price ($/h)
Hostinger	0.0411	0.0411	0.0411
Arubacloud	0.0838	0.1562	0.1200
Scaleway	0.0869	0.3649	0.1373
Upcloud	0.1185	0.2548	0.1528
Google Cloud	0.1340	0.2016	0.1779
Exoscale	0.1586	0.2129	0.1903
Cloudferro	0.1902	0.2366	0.2057
Hidora	0.1911	0.2321	0.2116
Alibabacloud	0.0940	2.2814	0.2441
Leafcloud	0.1585	0.2979	0.2528

Table 5.6: Top providers for the configuration 0 GPU, 4GB RAM, 16 vCPU, ordered by average price per hour. Prices are in USD.

Provider Diversity and Market Positioning The above configurations all lack GPUs, which may mean that virtual machines with GPUs are not widely used among

cloud providers. For this reason, it is worth analyzing the providers considered for this thesis more broadly and evaluating the maximum amount of vCPU, RAM, and GPU they make available per virtual machine. This analysis is presented in Table 5.7. The analysis shows that only 10 of the 20 providers analyzed offer GPUs to their customers for use on virtual servers without the need to purchase bare metal servers.

Provider	Max vCPU	Max RAM (GB)	Max GPU
IONOS	62	230	0.00
gridscale	64	224	0.00
aws	896	32768	8.00
googlecloud	360	5952	16.00
alibabacloud	256	3072	8.00
azure	832	15200	8.00
upcloud	80	512	0.00
catalystcloud	64	1280	0.25
scaleway	128	960	8.00
cloudferro	128	880	4.00
leaseweb	96	715	0.00
linode	56	300	4.00
exoscale	96	448	8.00
leafcloud	100	680	8.00
bluvault	96	256	0.00
arubacloud	32	64	0.00
hetzner	48	192	0.00
hidora	12	64	0.00
gigas	10	16	0.00
hostinger	8	32	0.00

Table 5.7: Maximum resources per provider

Market Segmentation Analysis by Usage Tier Tables 5.8 through 5.11 summarize the top five most efficient providers in each use tier. The findings reveal systematic patterns across market tiers in the forms of price dispersion, geographic clustering, and provider specialization.

In the small instance category, the variation in price is small, from $0.009980 to $0.012169 per hour. Such a relatively small dispersion testifies to the highly competitive environment. The least expensive offering is that of Scaleway in France. Others, including Alibaba Cloud in China, AWS in the USA, Linode in New Zealand, and UpCloud in Australia, provide the same levels of costs, which means that the geographical location is not the determining factor in this category. What is more, both x64 and the ARM CPU architecture are present among the highest-cost-effective offers, demonstrating the pluralism of the hardware choices at competitive levels.

Provider	vCPU	RAM	GPU	Storage	Price ($/h)	CPU	Region
Scaleway	2	2	0.0	10	0.009980	x64	France
Alibaba Cloud	1	1	0.0	0	0.010000	x64	China
AWS	2	1	0.0	0	0.010000	arm	USA
Linode	1	1	0.0	25	0.010500	x64	New-Zealand
UpCloud	1	1	0.0	25	0.012169	x64	Australia

Table 5.8: Top 5 Most Cost-Effective Small Instance Providers (1–2 vCPU, 0–8GB)

The medium instance segment shows high price dispersion, from $0.010868 to $0.044800 per hour. The 311% spread indicates the less-standardized market in this case compared to the small instances. Hetzner dominates in cost-efficiency with the cheapest observed price offering of 4 vCPU, 8 GB RAM, and 80 GB of storage. The remaining providers consist of both the European (Scaleway in France) and the non-European locations (Hostinger and AWS in India, UpCloud in Australia), indicating that cost efficiency is not confined to the traditionally low-cost geographies. Of interest is the fact that AWS and Hostinger Indian configurations are more expensive than Hetzner's German one, indicating that the efficiency of operations, along with the specific provider prices, prevail over the geography-based cost advantages of the labor in final prices.

Provider	vCPU	RAM	GPU	Storage	Price	CPU	Region
Hetzner	4	8	0.0	80	0.010868	arm	Germany
Hostinger	2	8	0.0	0	0.024644	N/D	India
UpCloud	2	8	0.0	3	0.041959	x64	Australia
Scaleway	4	8	0.0	10	0.043180	x64	France
AWS	2	8	0.0	0	0.044800	arm	India

Table 5.9: Top 5 Most Cost-Effective Medium Instance Providers (2–8 vCPU, 8–32GB)

In the large instance size category, Hetzner displays strong economies of scale: a 16 vCPU / 32 GB instance with 320 GB disk is priced at $0.037204 per hour, lower than numerous medium-level plans. The market structure here is more concentrated in Europe, with Scaleway in France and Aruba Cloud in Italy, also among the cheapest cost-effective providers. Non-European providers such as Hostinger and AWS in India are significantly more expensive. The durability of cost-effective pricing in the European regions translates into a structural efficiency advantage compared to the non-European markets. The CPU architecture diversification (ARM, x64, and N/D) highlights flexibility in hardware choice in this category.

Provider	vCPU	RAM	GPU	Storage	Price	CPU	Region
Hetzner	16	32	0.0	320	0.037204	arm	Germany
Hostinger	8	32	0.0	0	0.082178	N/D	India
Aruba Cloud	8	32	0.0	120	0.152833	x64	Italy
Scaleway	8	32	0.0	10	0.173580	arm	France
AWS	8	32	0.0	0	0.179200	arm	India

Table 5.10: Top 5 Most Cost-Effective Large Instance Providers (8+ vCPU, 32+GB)

The GPU instance segment (Table 5.11) reflects the highest price premiums, with the baseline GPU options starting at $0.270000 per hour, more than five times the highest price for large CPU-only instance. The cost leadership of this segment is concentrated in Asia, with Alibaba Cloud in China offering the least expensive option. The remaining European providers, such as Cloudferro in the Netherlands and LeafCloud in France, follow with competitively priced fractional or full GPU allocations. The other suppliers, such as AWS in South Korea and Linode in India, enter at the higher prices. The fractional GPU allocations observed (0.083 through Alibaba Cloud and Cloudferro) reflect the tactics based on server virtualization with the objective of entering the GPU-dependent workload barrier, with full GPU allocations (1.0) being offered in the high-performance offerings.

Provider	vCPU	RAM	GPU	Storage	Price	CPU	Region
Alibaba Cloud	4	8	0.083	0	0.270000	x64	China
Cloudferro	4	16	0.083	0	0.278497	x64	Netherlands
AWS	4	16	1.000	150	0.378530	x64	South-Korea
LeafCloud	8	32	1.000	0	0.453560	N/D	France
Linode	4	16	1.000	524	0.520000	x64	India

Table 5.11: Top 5 Most Cost-Effective GPU Instance Providers

Overall, there are a few key patterns; first, geographic clustering shows that European providers tend to dominate the CPU-heavy tiers, while Asian providers, especially Alibaba Cloud, lead when it comes to GPU options. Second, the concept of inverse scaling economics means that when users go for larger configurations, the cost per unit of compute generally drops, pointing to some solid economies of scale in infrastructure. Third, storage pricing doesn't seem to change much with capacity, except in cases where there's an unusually large amount bundled together. Providers tend to specialize: Hetzner, for example, keeps a cost advantage across CPU segments from its German data centers, whereas hyperscale companies like AWS and Azure mainly focus on the GPU market rather than competing on basic compute pricing.

Geographic Pricing Distribution Analysis Geographical pricing distribution, as evidenced by the regional heatmaps of mean hourly prices per provider (Fig. 5.4 through 5.11) reveals significant regional differences that validate and reinforce the identified patterns of the preceding configuration-level analyses.

The observable clustering of competitive pricing in Europe is specifically validated in the data, with providers such as Hostinger having a uniform pricing of $0.04 per hour for all regions in which they operate.

At the same time, Aruba Cloud offers especially low rates ($0.24 per hour) in a selection of European nations. Such geographic uniformity in European pricing supports the hypothesis of structural economies of scale. It indicates a more mature, standardized market in the region, in accordance with the European dominance presented in Tables 5.9 and 5.10.

On the other hand, hyperscale providers demonstrate greater geographic variability: AWS establishes prices ranging from $8.27 per hour in the United Arab Emirates to $12.89 per hour in South Africa, underscoring the major effect of local operating

costs and regional demand in setting pricing strategies. Also, the variability of price for Azure ranged from $3.39 per hour in Indonesia to $13.20 per hour in Singapore, suggesting that hyperscale operators practice differentiated pricing strategies in consideration of local market forces.
Alibaba Cloud, in contrast, continues to retain a leadership in Asia with prices ranging from $1.45 per hour in the UAE to $4.27 per hour in Saudi Arabia, and hence upholds a market penetration strategy through low price aggression in both origin region and adjacent regions, previously noted in Table 5.11. The sparse but stable presence of specialized operators such as Hetzner in a selection of markets (Germany, USA, Singapore, Finland) with very competitive pricing ($0.47-$3.34 per hour) reveals a way in which geographic specialization secures major price benefits, hence serving economies of scale patterns outlined in tier-level analysis.

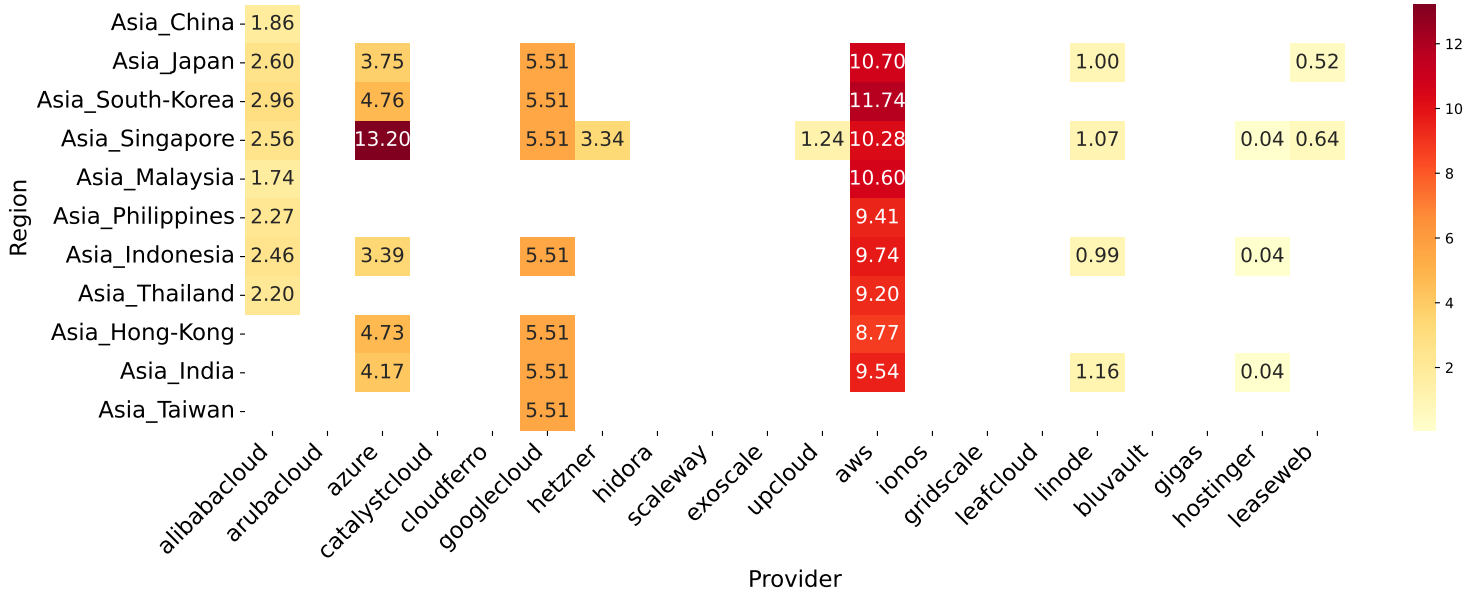


Figure 5.4: Average price in USD for VM based in Asia per provider

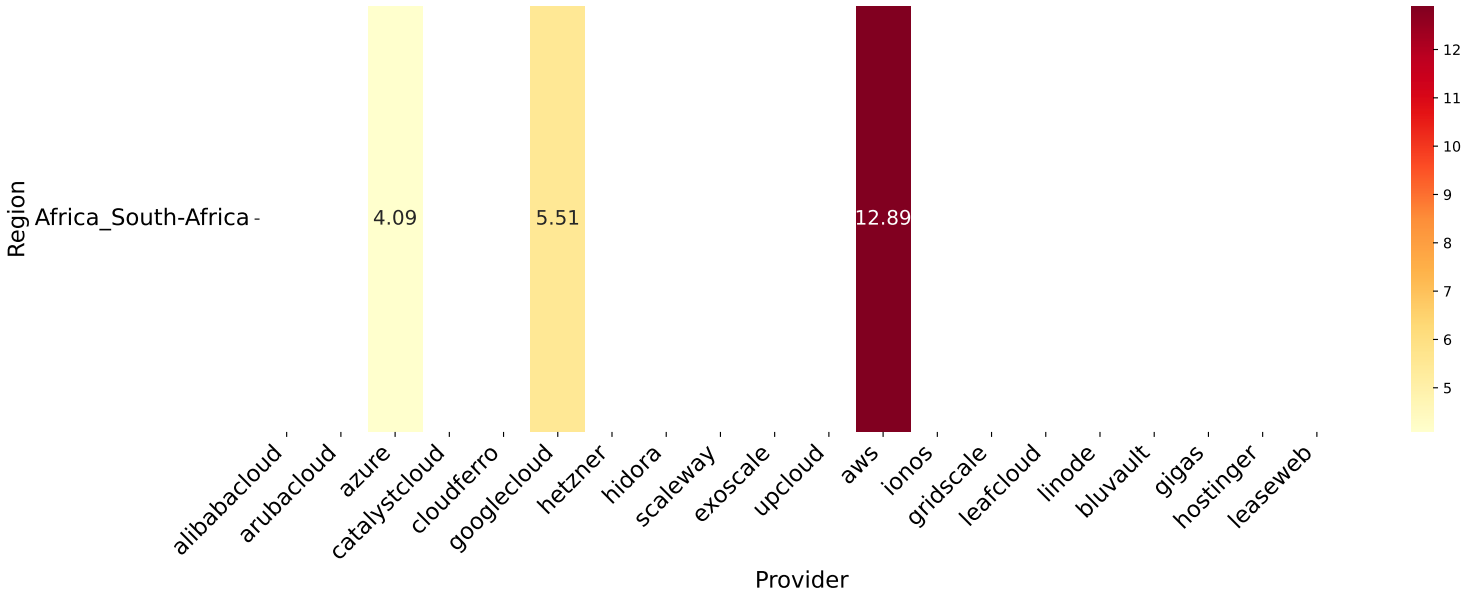


Figure 5.5: Average price in USD for VM based in Africa per provider

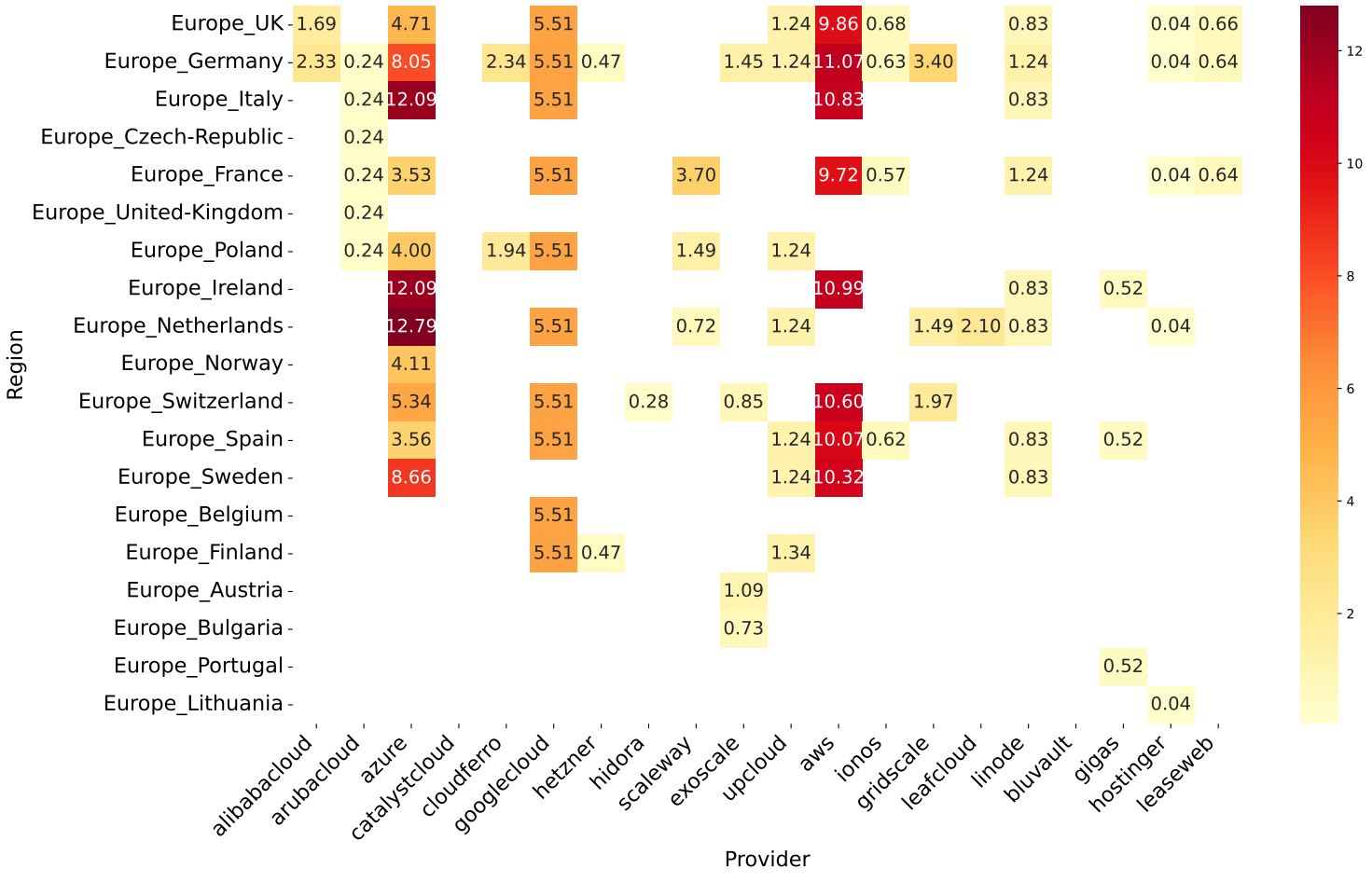


Figure 5.6: Average price in USD for VM based in Europe per provider

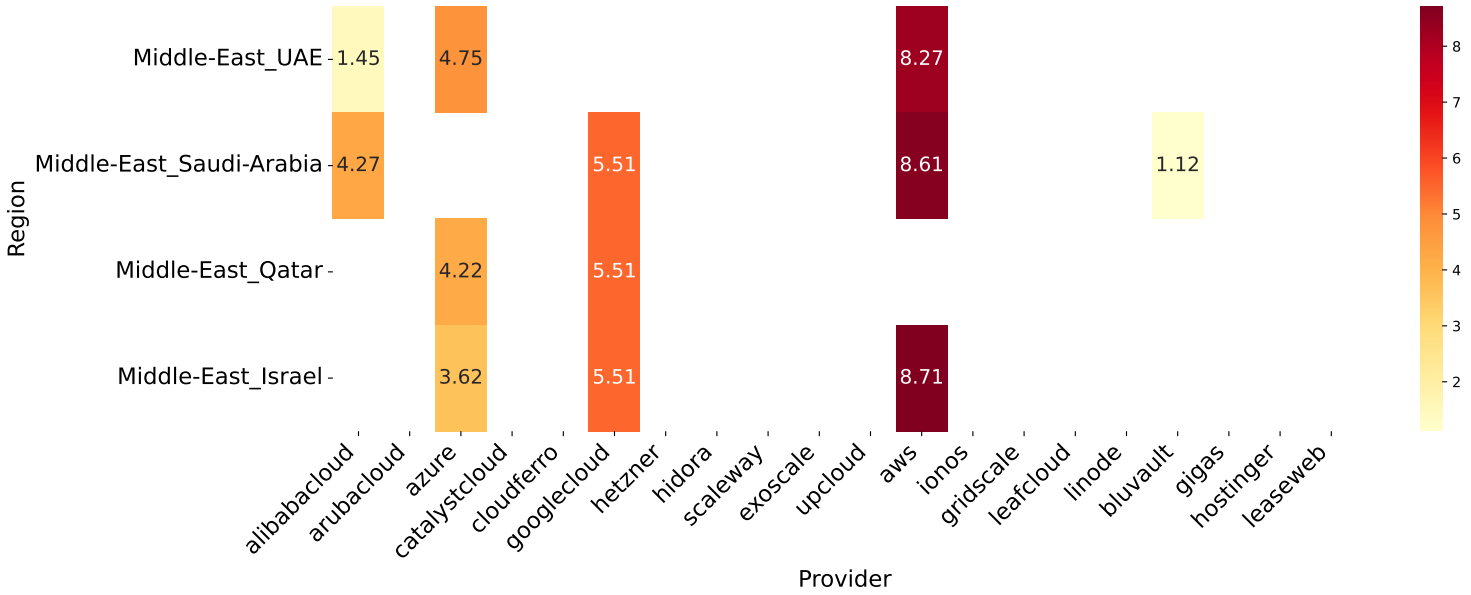


Figure 5.7: Average price in USD for VM based in the Middle-East per provider

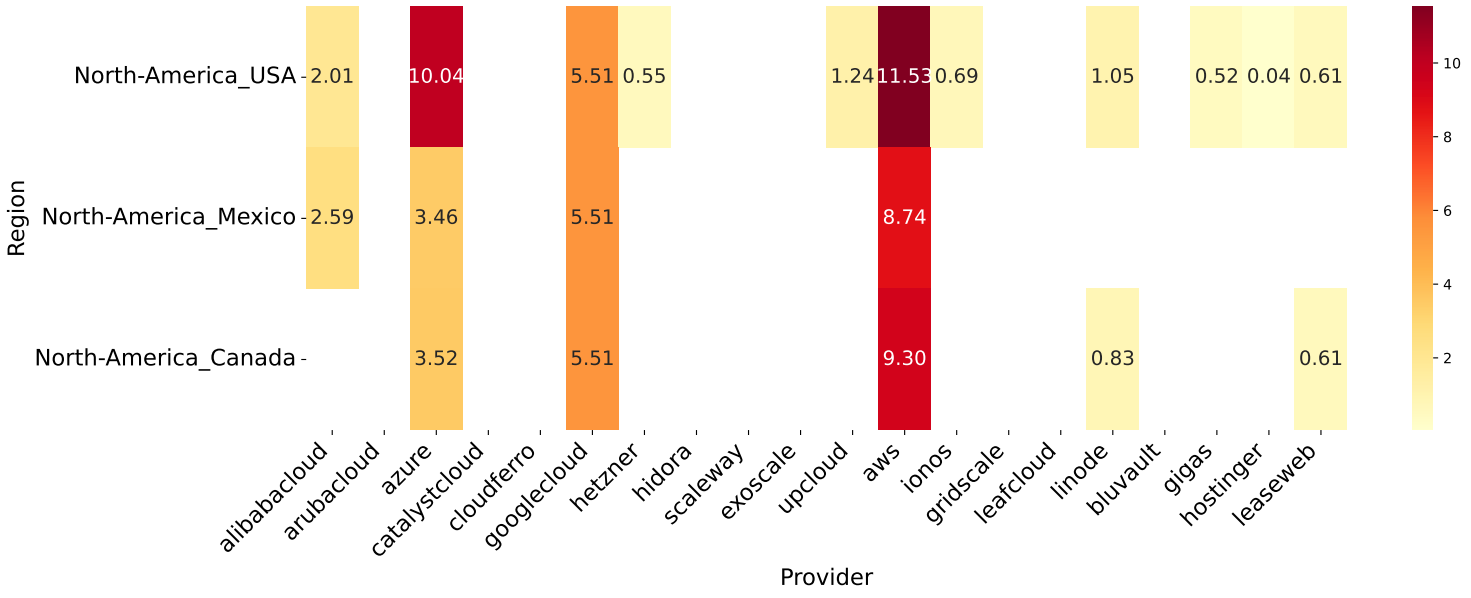


Figure 5.8: Average price in USD for VM based in North America per provider

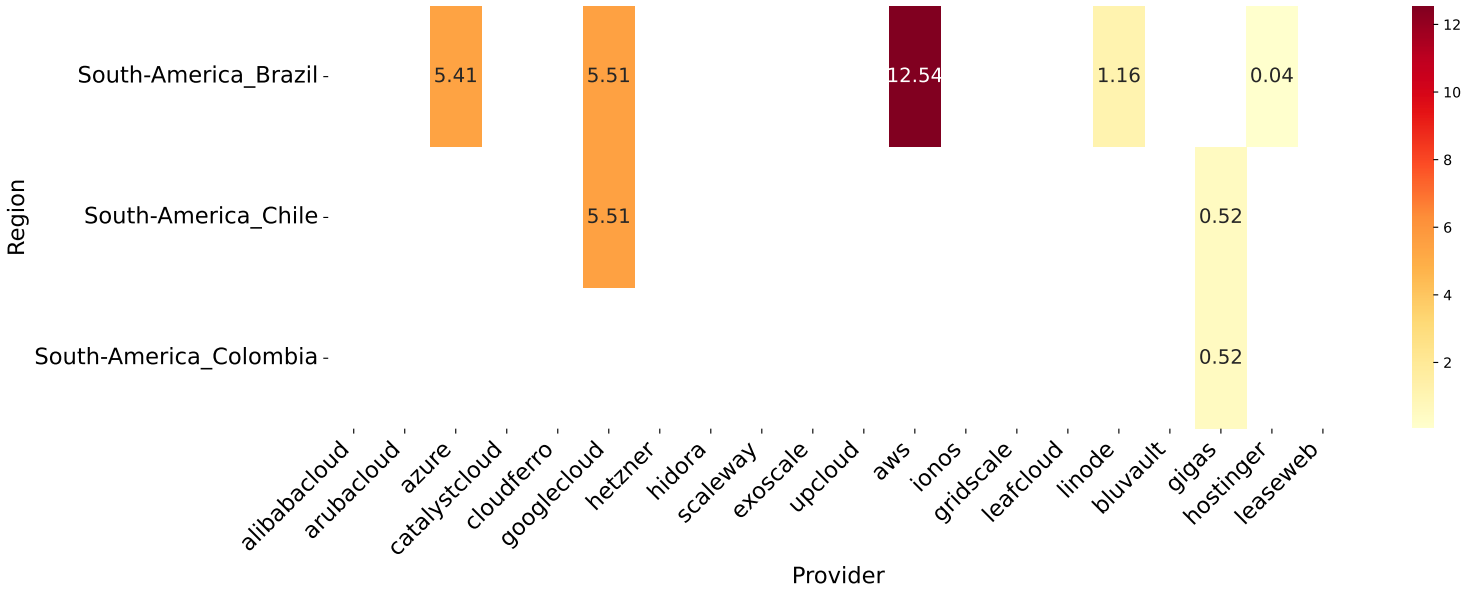


Figure 5.9: Average price in USD for VM based in South America per provider

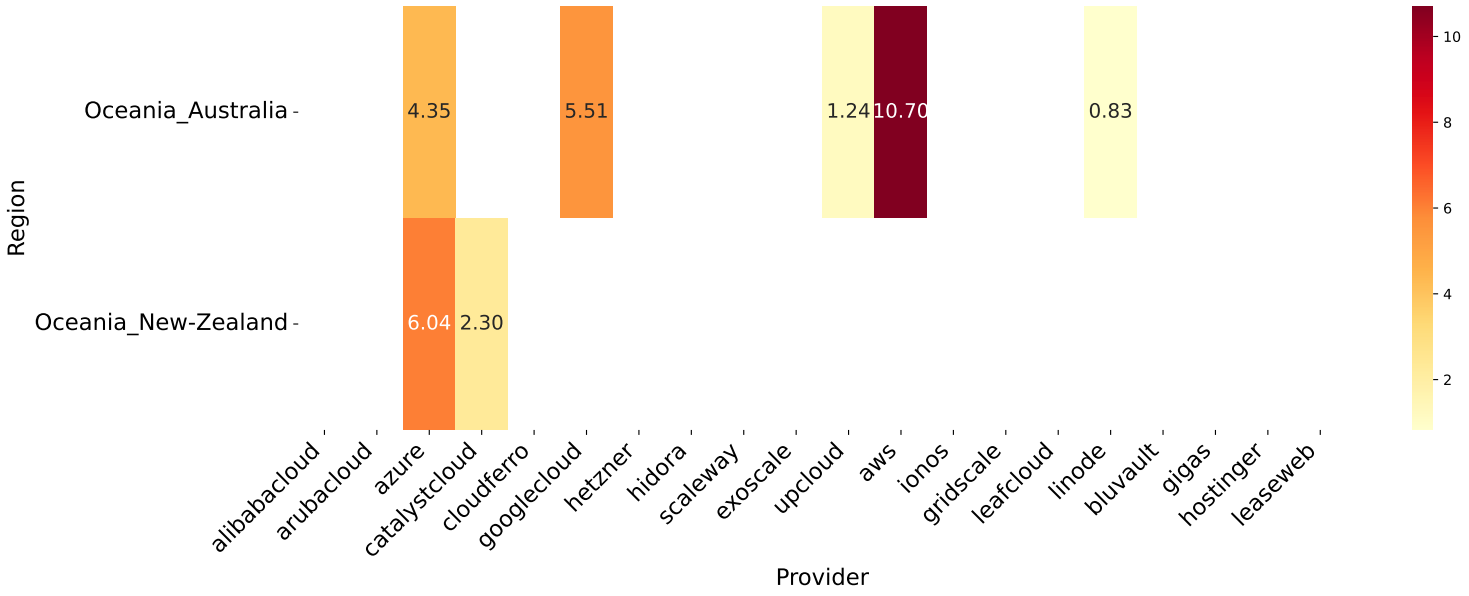


Figure 5.10: Average price in USD for VM based in Oceania per provider

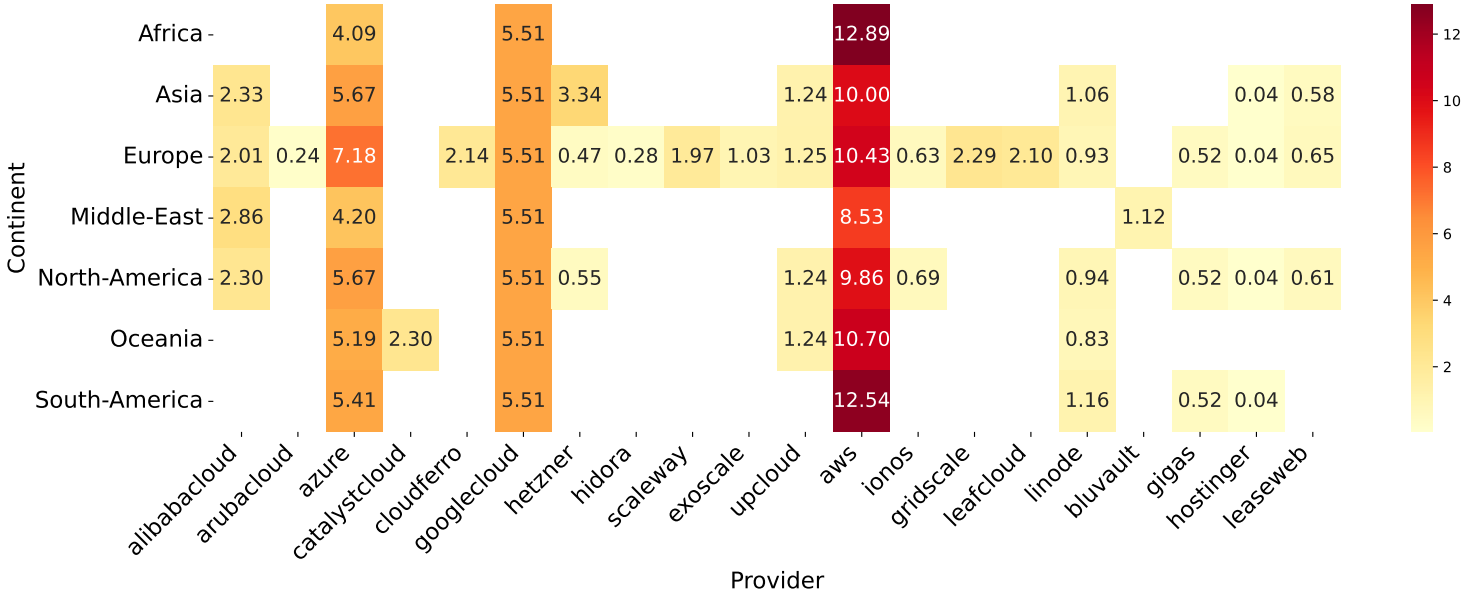


Figure 5.11: Average price in USD for VM by continent per provider

Feature Importance Analysis via Mutual Information Following the analysis just completed on the types of virtual machines available and the costs based on geographical areas, it is interesting to analyse what has the greatest impact on the price of the service.

The mutual information analysis for hourly pricing, presented in Figure 5.12, indicates a clear set of features that contribute to price composition.

Memory capacity (memoryGB) is the top predictor of pricing, followed by the number of virtual CPUs (vcpu), thereby supporting the CPU-RAM focused nature of cloud pricing models seen in Tables 5.8 through 5.10.

Storage capacity (storageGB) takes third position, suggesting a prominent but secondary pricing-determining role for storage. It is interesting to note that the GPU-related features are at the very end of the ranking (GPU = 0.041, GPUVendor = 0.024, GPUModel = 0.046, GPUVramGB = 0.044), seemingly against the high premium pricing seen for GPU instances in the Table 5.11. Low mutual information in this case likely indicates sparsity of GPU instances in the dataset rather than low price determination impact, owing to the data set distribution where we observe the majority of the configurations having no GPUs (Table 5.3).

Low determination for CPU architecture supports the little price determination effect for architectural differences (x64 versus ARM), owing to hardware diversity seen in the case of cost-effective configurations in the tiered analysis.

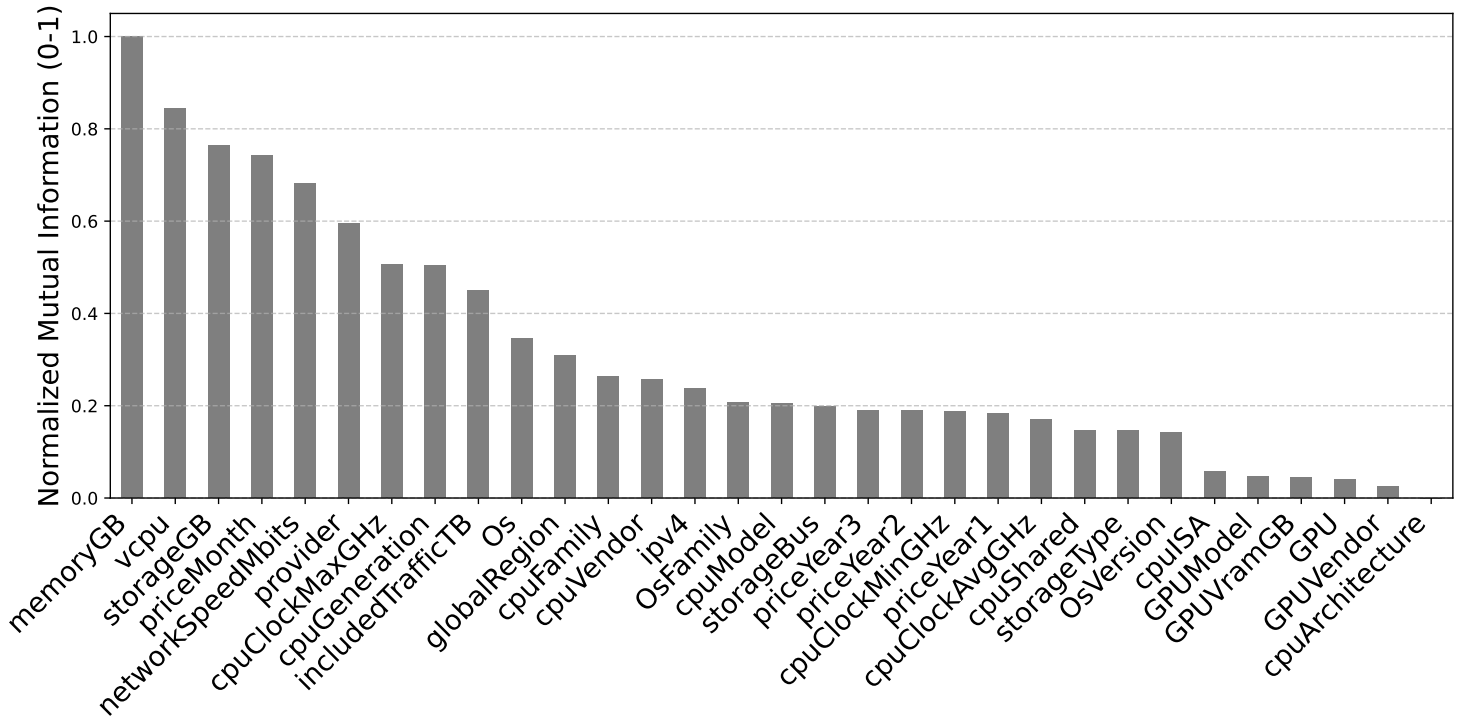


Figure 5.12: Mutual Information Analysis

5.4 Dataset Imputation

As discussed in the previous sections, not all providers disclose all the information about their virtual machines. For example, some providers may not disclose information about the underlying hardware architecture or may not declare the speed of the network to which the machine has access. In this case, this feature will be set to `None` by default for that record. For this reason, the dataset contains many `None` values. To try to fill in the "gaps", imputation can be used.

Importance of imputation Imputation is crucial to prevent the loss of information in the dataset.
If all rows with missing values were removed, the dataset would shrink significantly, and useful patterns could be lost.
By imputing missing values, the data is preserved and the statistical quality improves, since missing values may otherwise introduce bias and distort distributions. Moreover, imputation is essential for the final objective of this thesis, which is to estimate the prices of virtual machines in the cloud using a machine learning model. Many models, such as linear regression and random forests, cannot handle `None` values, making the imputation of missing data a necessary step.

Results of imputation with Random Forest and XGBoost Tables 5.12 and 5.13 outline how Random Forest and XGBoost deal with missing data in the given dataset. More concretely, Table 5.12 sets out various accuracies for categorical variables, while Table 5.13 documents mean squared errors for numerical variables.
Over almost all categorical features, the results empower XGBoost to distinctly outperform Random Forest, which, in fact, gets close to one hundred percent accuracy

for the mentioned features like `cpuFamily`, `cpuModel`, and `preInstalledSoftware`. So, the point where the most considerable jump for accuracy happens is in those features that are complex, such as `OsVersion` and `OsFamily`, where XGBoost profits from a huge accuracy increment.
As for the numerical features, XGBoost is still the best solution. The difference is noticeable on CPU clock features and `includedTrafficTB`, where the error is quite a bit lower than that of Random Forest. Unfortunately, the two algorithms are equally ineffective when it comes to dealing with elements that have a lot of variations, such as `networkSpeedMbits`. However, the error is still lower if XGBoost is used.
The analyses carried out show that XGBoost produces better results. For this reason, it was decided to use the dataset imputed with XGBoost to proceed with the development and training of the models.

Feature	**Random Forest**	**XGBoost**
OsFamily	0.8078	**0.9349**
Os	0.8648	**0.9679**
OsVersion	0.3888	**0.9141**
cpuVendor	0.9985	**0.9998**
cpuGeneration	0.9767	**0.9907**
cpuFamily	0.9999	**1.0000**
cpuModel	1.0000	**1.0000**
cpuArchitecture	1.0000	1.0000
cpuISA	0.9848	**1.0000**
storageBus	0.9348	**0.9877**
storageType	0.9967	**0.9983**
preInstalledSoftware	0.8745	**0.9991**
OsLicense	1.0000	1.0000
ipv4	0.9939	**1.0000**
ipv6	1.0000	1.0000
cpuShared	0.9340	**0.9958**
Average	0.9397	**0.9816**

Table 5.12: Imputation Accuracy for Categorical Variables

Feature	**Random Forest**	**XGBoost**
networkSpeedMbits	4.82e7	**1.44e7**
includedTrafficTB	1.21e3	**1.27**
cpuClockMaxGHz	0.0843	**0.0034**
cpuClockMinGHz	0.0005	**0.0000**
cpuClockAvgGHz	0.0037	**0.0000**

Table 5.13: Imputation MSE for Numerical Variables

5.5 Model Training

For the regression problem, four different models were selected, and their performance was evaluated on the constructed dataset.
Linear Regression was chosen as a baseline model. This model is simple and interpretable, but its performance may be unsatisfactory if the relationship between inputs and outputs is not linear.
The second model tested was the RBFNN (Radial Basis Function Neural Network), a type of neural network that uses radial basis functions as activation functions. It is well-suited for datasets with non-linear relationships, as it constructs small local models around the centroids. However, on large datasets, it can be computationally intensive because the distance between each point to be predicted and all centroids must be calculated for every new prediction.
The third model used was the Random Forest, which, as described in Section 2.4, trains multiple trees on subsets of the dataset and makes predictions by averaging the outputs of all trees.
The final model selected was XGBoost. The decision to use this last model was motivated by the quality of the previous imputation. In fact, XGBoost proved capable of accurately estimating most of the missing values within the dataset. Another advantage of XGBoost is the large number of hyperparameters available for fine-tuning the model.

Linear Regression The baseline model uses the LinearRegression class provided as part of scikit-learn; the choice to use a default implementation was motivated by its efficiency and accuracy, which facilitated reliable results and a focus on methodological interpretation rather than implementation.
The data was pre-processed by removing non-numeric attributes unsuitable for regression, then split into training and test sets.
The input features were also transformed into z-scores to promote numerical stability and comparison between variable dimensions.
The model was trained with standardized training data and has the limitation of only making non-negative predictions to avoid unrealistic price predictions; linear regression, in itself, does not guarantee the absence of negative values out of the box.

Radial Basis Function Neural Network The RBFNN was developed using custom implementations as it is not directly available in the standard scikit-learn libraries.
The model organizes data using RBFs as active functions, forming complex and targeted models through the MiniBatchKMeans centroid-based cluster model.
In the training phase, the centroids are the centers of the RBFs, while the RBF variance σ is determined as the average distance between the centroids; once the activation matrix is formed, the output weights are acquired via the pseudoinverse, allowing the model to bypass complex matrix multiplication.
The data used is transformed via z-score normalization to standardize the model in relation to the dimensions of the cross-feature.
The number of centroids is determined through a candidate search procedure based

on validation error.

Random Forest The Random Forest model was implemented using the RandomForestRegressor class from scikit-learn: this choice was motivated, as for linear regression, by the need to use a widely used and tested implementation of the model in order to focus on the results rather than the implementation. The dataset was pre-processed by removing non-numeric features, and the remaining variables were standardized using z-score normalization to ensure consistent scaling between inputs.

XGBoost XGBoost (Extreme Gradient Boosting) was chosen as the fourth model to be tested, thanks to the good results obtained during the imputation phase and the possibility of fine-tuning the model's hyperparameters to improve the regression output.
Also in this case, the choice to use an existing implementation of the model is made to minimize implementation errors that could lead to incorrect performance.

Chapter 6

Work evaluation

6.1 Model Evaluation

To evaluate model performance in estimating the hourly price of a VM, a 5-fold cross-validation was employed. The dataset is divided into five subsets, each used once for testing, while the others serve for training, providing a robust assessment of the model.
The following metrics were analyzed:

- RMSE (Root Mean Squared Error): average magnitude of prediction errors, with higher weight on larger errors.
- MAE (Mean Absolute Error): average absolute difference between predicted and actual values, less sensitive to outliers than RMSE.
- R^2 (Coefficient of Determination): proportion of variance in the target explained by the model; values near 1 indicate a good fit, negative values indicate poor performance.
- MAPE (Mean Absolute Percentage Error): prediction error expressed as a percentage of actual values.

6.1.1 Linear Regression

Evaluation of the Linear Regression model on the basis of the 5-fold cross-validation shows strong difficulty in forecasting VM hourly prices across varying cost ranges. As shown in Tables 6.1 through 6.5 and tabulated as a summary in Table 6.6, the performance of the model significantly depends on the range of the price under consideration, offering important insight into the diverse nature of VM price schemes. The most significant finding is the model's inadequate efficacy in the lower pricing tiers. For virtual machines (VMs) priced between \$0-1 per hour, which represent the majority of the dataset, the model records an average R^2 of -31.823 $\pm$ 2.874. This negative coefficient of determination suggests that the Linear Regression model performs markedly worse than a simplistic predictor that relies solely on the average price for all forecasts. Equally troubling is the Mean Absolute Percentage Error (MAPE) of 497.86%, indicating that the predictions diverge from actual values by nearly 500% on average. This trend is evident in the \$1-3 and \$3-5 price brackets, where R^2 values remain significantly negative (-46.030 and -109.010, respectively),

coupled with MAPE values surpassing 120%.
Model performance shows gradual improvement as the price of VM increases. In the \$5-15 range, although the R^2 is still negative (-7.480), the absolute value declines significantly, as does the MAPE, which drops to a more manageable 84.80%. This extends to the \$15-50 range, where the R^2 closes on zero (-0.943), meaning that the model prediction is nearly the same as would be the case where the mean was used. It is only in the most expensive range (\$50-800) that the positive R^2 value (0.532 ± 0.046) is obtained, which suggests the model is capable of rendering about 53% of the variation in premium VM price.
This consistency across the five folds used for cross-validation, as indicated by the small standard deviations, substantiates the perception that these performance trends are systematic rather than an artifact of the actual data partitions. This consistency adds to the believability of the results and implies that the poor results on the lower price ranges are an inherent nature of the linearity assumptions of the model rather than being a result of issues related to the quality of the data or overfitting. They point out significant shortcomings of Linear Regression as a suitable predictive instrument for the price of virtual machines. Linearity among target variables and features emerges as a flawed assumption, since sophisticated, non-linear pricing methodologies would likely predominate the less populated price bands.
This analysis identify Linear Regression as an important milestone, demonstrating the failure of simple linear techniques for this predictive task. The pervasive failure evidenced across multiple price ranges provides strong motivation to pursue sophisticated modeling techniques capable of adequately representing the nonlinear forces acting upon VM pricing within the cloud.

Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	1.463	0.838	-28.41	498.17	32509
(1, 3)	3.661	2.963	-44.06	164.41	14638
(3, 5)	5.561	4.889	-107.59	129.13	6739
(5, 15)	8.171	6.985	-7.39	84.16	10750
(15, 50)	13.515	11.467	-0.93	43.38	6015
(50, 800)	46.503	38.190	0.50	45.65	1583

Table 6.1: Linear Regression metrics for Fold 1 by price range

Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	1.485	0.841	-29.56	490.11	32553
(1, 3)	3.642	2.947	-43.62	163.67	14727
(3, 5)	5.539	4.902	-107.51	129.85	6696
(5, 15)	8.273	7.112	-7.57	85.14	10653
(15, 50)	13.472	11.417	-0.90	43.18	6118
(50, 800)	49.155	39.765	0.60	45.52	1489

Table 6.2: Linear Regression metrics for Fold 2 by price range

Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	1.629	0.834	-35.52	494.84	32387
(1, 3)	3.779	2.968	-47.41	164.56	14699
(3, 5)	5.754	4.923	-115.03	129.86	6728
(5, 15)	8.160	7.020	-7.42	84.40	10863
(15, 50)	13.781	11.682	-0.95	44.43	6062
(50, 800)	47.570	39.037	0.53	46.20	1495

Table 6.3: Linear Regression metrics for Fold 3 by price range

Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	1.578	0.833	-33.37	510.01	32393
(1, 3)	3.770	2.964	-47.31	163.67	14685
(3, 5)	5.567	4.919	-108.61	130.46	6787
(5, 15)	8.275	7.121	-7.66	85.28	10851
(15, 50)	13.828	11.638	-1.01	44.48	6019
(50, 800)	46.723	38.872	0.49	45.89	1494

Table 6.4: Linear Regression metrics for Fold 4 by price range

Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	1.556	0.831	-32.25	496.16	32365
(1, 3)	3.784	2.993	-47.75	165.56	14761
(3, 5)	5.608	4.892	-106.31	129.09	6710
(5, 15)	8.201	7.070	-7.37	85.03	10770
(15, 50)	13.461	11.424	-0.92	43.76	6113
(50, 800)	48.064	38.531	0.54	45.41	1516

Table 6.5: Linear Regression metrics for Fold 5 by price range

Price Range	RMSE	MAE	R^2	MAPE (%)	N_total
(0,1)	1.542 ± 0.068	0.835 ± 0.004	-31.823 ± 2.874	497.86 ± 7.41	162207
(1,3)	3.727 ± 0.070	2.967 ± 0.017	-46.030 ± 2.013	164.38 ± 0.78	73510
(3,5)	5.606 ± 0.086	4.905 ± 0.016	-109.010 ± 3.461	129.68 ± 0.57	33660
(5,15)	8.216 ± 0.055	7.062 ± 0.059	-7.480 ± 0.125	84.80 ± 0.49	53887
(15,50)	13.611 ± 0.178	11.526 ± 0.125	-0.943 ± 0.044	43.85 ± 0.59	30327
(50,800)	47.603 ± 1.073	38.879 ± 0.593	0.532 ± 0.046	45.74 ± 0.32	7577

Table 6.6: Summary of Linear Regression metrics by price range (mean ± std over 5 folds)

For completeness of analysis, Fig. 6.1 shows the scatter plots with the actual values and the values predicted by the model. The further the points deviate from the red line, the more inaccurate the model is in the regression.

6.1.2 RBFNN

Radial Basis Function Neural Network (RBFNN) shows, again, specific performance limitations on the VM pricing data, as Tables 6.7 through 6.11, and Table 6.12, indicate. Although there is a theoretical potential for neural networks to capture complex nonlinear relationships, the RBFNN does not reach acceptable predictive fidelity on most price intervals. This limitation could be due to the small number of centroids set up at 500, a result of the larger estimation time needed, combined with the high hardware resources necessitated as the number of centroids increases. It also reveals the inferior performance during the lower price intervals. In the $0-1 segment that holds 162,207 observations, the RBFNN's R^2 stands at -120.569 $\pm$ 10.967, while the MAPE is 1669.115%, meaning the forecasts vary significantly, even averaging over a 1600% deviation from real values. This is the essence of model failure, significantly underperforming even the naive mean prediction. Is possible to see similar trends hold within the $1-3 ($R^2$ = -49.969) and $3-5 ($R^2$ = -67.393) intervals, as the coefficient of determination becomes consistently negative across all the middle-range intervals.

Performance-based baseline levels only happen within high-end ranges. Within the $15-50 category, the value of R^2 is -0.110 $\pm$ 0.063, reaching the mean predictor performance approximations, while only the $50-800 range reaches positive explanatory potential (R^2 = 0.354 $\pm$ 0.070). It still considerably lags behind ensemble methodologies that were previously analyzed.

This consistent underperformance throughout the validation folds confirms systematic instead of partition-specific model shortcomings. Broad standard deviations during the lower price bands suggest learning instability as much as convergence problems.

These deficiencies stem from several factors inherent to the RBFNN architecture applied to VM pricing data. The radial basis function methodology struggles with optimal center placement and bandwidth parameter selection in high-dimensional feature spaces. Additionally, distance-based similarity measures through Gaussian functions exhibit poor compatibility with heterogeneous VM configuration features, where Euclidean metrics fail to capture meaningful relationships between categorical and mixed-type variables.

System failure of the RBFNN validates that architectural complexity does not translate to empirical dominance.

Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	3.047	2.030	-126.489	1798.331	32509
(1, 3)	4.040	2.822	-53.869	164.529	14638
(3, 5)	4.697	3.415	-76.455	90.883	6739
(5, 15)	6.117	4.715	-3.700	55.543	10750
(15, 50)	10.253	7.894	-0.112	30.726	6015
(50, 800)	55.567	34.891	0.282	35.655	1583

Table 6.7: RBFNN metrics for Fold 1 by price range

Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	2.909	2.012	-116.296	1556.843	32553
(1, 3)	3.760	2.618	-46.543	153.726	14727
(3, 5)	4.329	3.106	-65.298	83.332	6696
(5, 15)	6.170	4.595	-3.766	53.121	10653
(15, 50)	10.106	7.719	-0.067	30.470	6118
(50, 800)	62.216	35.569	0.362	32.859	1489

Table 6.8: RBFNN metrics for Fold 2 by price range

Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	3.162	2.133	-136.578	1722.293	32387
(1, 3)	3.716	2.558	-45.789	151.148	14699
(3, 5)	4.133	2.958	-58.869	79.058	6728
(5, 15)	5.673	4.314	-3.070	50.088	10863
(15, 50)	10.098	7.679	-0.048	30.518	6062
(50, 800)	50.899	30.605	0.465	30.739	1495

Table 6.9: RBFNN metrics for Fold 3 by price range

Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	2.893	1.952	-114.533	1499.950	32393
(1, 3)	4.065	2.651	-55.182	154.812	14685
(3, 5)	4.743	3.329	-78.570	89.038	6787
(5, 15)	6.120	4.733	-3.735	55.233	10851
(15, 50)	10.292	7.866	-0.115	31.030	6019
(50, 800)	54.258	35.184	0.306	34.789	1494

Table 6.10: RBFNN metrics for Fold 4 by price range

Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	2.830	2.024	-108.946	1768.159	32365
(1, 3)	3.811	2.697	-48.461	157.528	14761
(3, 5)	4.150	3.160	-57.776	84.334	6710
(5, 15)	5.660	4.283	-2.987	49.653	10770
(15, 50)	10.683	8.296	-0.209	33.153	6113
(50, 800)	57.108	34.590	0.352	34.172	1516

Table 6.11: RBFNN metrics for Fold 5 by price range

Price Range	RMSE	MAE	R^2	MAPE (%)	N_total
(0, 1)	2.968 ± 0.134	2.030 ± 0.065	-120.569 ± 10.967	1669.115 ± 132.812	162207
(1, 3)	3.878 ± 0.163	2.669 ± 0.099	-49.969 ± 4.298	156.348 ± 5.114	73510
(3, 5)	4.411 ± 0.293	3.194 ± 0.181	-67.393 ± 9.703	85.329 ± 4.714	33660
(5, 15)	5.948 ± 0.258	4.528 ± 0.217	-3.452 ± 0.388	52.728 ± 2.774	53887
(15, 50)	10.286 ± 0.238	7.891 ± 0.245	-0.110 ± 0.063	31.180 ± 1.125	30327
(50, 800)	56.010 ± 4.157	34.168 ± 2.024	0.354 ± 0.070	33.643 ± 1.916	7577

Table 6.12: Summary of RBFNN metrics by price range (mean ± std over 5 folds)

For completeness of analysis, Fig. 6.2 shows the scatter plots with the actual values and the values predicted by the model. The further the points deviate from the red line, the more inaccurate the model is in the regression.

6.1.3 Random Forest

Random Forest shows significantly enhanced performance over Linear Regression, as shown by the results in Tables 6.13 through 6.17, summed up in Table 6.18. Capturing non-linearity as well as interactions among features is the prowess of the ensemble technique, which pays off very significantly in predictive accuracy on most price ranges, with some problems still unresolved in targeted ranges.

Most notable progress is also made in the lowest price bin ($0-1), where Random Forest registers a positive R^2 value of 0.403 ± 0.011, a good improvement over Linear Regression's results (-31.823). This would mean that the model is now capable of interpreting about 40% of the variance within the low-cost VM price, a remarkable gain given that this bin holds the largest number of instances (162,207). The MAPE also significantly improves to 79.743%, although it is still fairly high, indicating that percentage errors are still very large, albeit very low-cost VMs, where small absolute errors correspond to very high relative deviations.

In the middle-range price bands($1-3 and $5-15), Random Forest still significantly dominates Linear Regression. In the $1-3 category, the model scores an R^2 of 0.081 $\pm$ 0.045 with a MAPE of 19.800%, the $5-15 category showing very strong performance with an R^2 of 0.781 $\pm$ 0.007 and a MAPE of only 11.626%. These indicate the increasing strength of the model as the price paid for VMs goes up, most likely the result of more consistent pricing behaviour higher up the value range.

The model's performance reaches its peak in the premium segments ($15-50 and $50-800), where R^2 values exceed 0.94, indicating that Random Forest can explain over 94% of the variance in expensive VM pricing. The MAPE values in these ranges drop to below 6%, with the highest price segment achieving an exceptional 3.497% MAPE. This superior performance in premium segments suggests that high-value VMs follow more predictable pricing patterns that the ensemble method can effec-

tively capture.

But there exists a serious anomaly presented by the model around the $3-5 price range, where the R^2 is persistently negative (-1.257 $\pm$ 0.085) even as gains elsewhere are being recorded. This apparently anomalous result suggests that the price range itself contains arguably complex pricing structures that cannot be captured properly by the Random Forest algorithm. The RMSE is 0.803 while the MAE is 0.554, both values that do not meaningfully deviate from other bands, indicating that the R^2 being negative may be an indicator of high variance among the target values.

Consistency among the results obtained on the five folds used for cross-validation, testified to by the relatively small standard deviations, validates the measurements of the performance. The small standard deviations on R^2 values obtained on most segments (usually below 0.05) prove that Random Forest is stable in its prediction, no matter the data split that is used to train as well as to test.

These results also indicate Random Forest's remarkable capability to handle the complex non-linearities present in VM pricing. By averaging together many decision trees, as the ensemble method does, feature interactions, as well as non-linearity that cannot be captured by individually linear models, are adequately conveyed. The strong improvements over most price ranges substantiate the hypothesis that VM pricing possesses a complex structure that requires sophisticated modeling techniques, while the challenges remaining in the $3-5 range indicate opportunities to enhance the modeling or to undertake feature engineering in future work.

Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	0.206	0.116	0.416	86.305	32509
(1, 3)	0.524	0.350	0.077	19.595	14638
(3, 5)	0.793	0.548	-1.210	14.082	6739
(5, 15)	1.300	0.946	0.788	11.483	10750
(15, 50)	1.884	1.304	0.962	5.379	6015
(50, 800)	13.909	4.159	0.955	3.511	1583

Table 6.13: Random Forest metrics for Fold 1 by price range

Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	0.210	0.116	0.386	74.289	32553
(1, 3)	0.542	0.371	0.013	20.628	14727
(3, 5)	0.792	0.550	-1.218	14.203	6696
(5, 15)	1.352	0.985	0.771	11.884	10653
(15, 50)	1.860	1.308	0.964	5.423	6118
(50, 800)	22.548	5.300	0.916	3.615	1489

Table 6.14: Random Forest metrics for Fold 2 by price range

Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	0.209	0.116	0.401	75.854	32387
(1, 3)	0.522	0.357	0.077	19.897	14699
(3, 5)	0.824	0.566	-1.378	14.508	6728
(5, 15)	1.316	0.950	0.781	11.566	10863
(15, 50)	1.927	1.335	0.962	5.513	6062
(50, 800)	18.224	4.547	0.931	3.566	1495

Table 6.15: Random Forest metrics for Fold 3 by price range

Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	0.207	0.116	0.407	77.313	32393
(1, 3)	0.513	0.347	0.104	19.409	14685
(3, 5)	0.808	0.556	-1.309	14.294	6787
(5, 15)	1.327	0.960	0.777	11.651	10851
(15, 50)	1.913	1.336	0.961	5.548	6019
(50, 800)	12.539	3.810	0.963	3.261	1494

Table 6.16: Random Forest metrics for Fold 4 by price range

Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	0.209	0.117	0.403	84.956	32365
(1, 3)	0.504	0.348	0.135	19.472	14761
(3, 5)	0.797	0.551	-1.169	14.110	6710
(5, 15)	1.302	0.948	0.789	11.544	10770
(15, 50)	1.962	1.368	0.959	5.699	6113
(50, 800)	14.371	4.182	0.959	3.531	1516

Table 6.17: Random Forest metrics for Fold 5 by price range

Price Range	RMSE	MAE	R^2	MAPE (%)	N_total
(0, 1)	0.208 ± 0.002	0.116 ± 0.001	0.403 ± 0.011	79.743 ± 5.500	162207
(1, 3)	0.521 ± 0.014	0.355 ± 0.010	0.081 ± 0.045	19.800 ± 0.499	73510
(3, 5)	0.803 ± 0.013	0.554 ± 0.007	-1.257 ± 0.085	14.239 ± 0.172	33660
(5, 15)	1.319 ± 0.022	0.958 ± 0.016	0.781 ± 0.007	11.626 ± 0.156	53887
(15, 50)	1.909 ± 0.039	1.330 ± 0.026	0.962 ± 0.002	5.512 ± 0.124	30327
(50, 800)	16.318 ± 4.072	4.400 ± 0.567	0.945 ± 0.020	3.497 ± 0.138	7577

Table 6.18: Summary of Random Forest metrics by price range (mean ± std over 5 folds)

For completeness of analysis, Fig. 6.3 shows the scatter plots with the actual values and the values predicted by the model. The further the points deviate from the red line, the more inaccurate the model is in the regression.

6.1.4 XGBoost

XGBoost model represents a breakthrough improvement in predictive performance, as the Tables 6.19 through 6.23 and Table 6.24 show. The complex approach of the gradient boosting framework to consecutive error adjustment and modeling of interactions among features results in outstanding improvement on almost all price ranges, making XGBoost the most effective analyzed model. The most significant accomplishment is evident in the lowest price category ($0-1), wherein XGBoost achieves an R^2 of 0.9751 ± 0.0005, thereby accounting for over 97% of the variance in low-cost virtual machine pricing. This marks a substantial enhancement compared to the inadequate results of Linear Regression (-31.823) and the moderate performance of Random Forest (0.403). The Mean Absolute Percentage Error (MAPE) experiences an important decrease to 8.522%, suggesting that prediction inaccuracies are now restricted to single-digit percentages even within this category. The exceptionally low Root Mean Square Error (RMSE) of 0.0425 and Mean Absolute Error (MAE) of 0.0229 further validate the model's effectiveness in forecasting prices for the most populous segment of the dataset.

Performance is uniform throughout the mid-range price segments. Within the $1-3 category, XGBoost attains an R^2 of 0.8704 ± 0.0027, coupled with a MAPE of 6.249%, signifying significant enhancements relative to both previous models. The $5-15 segment exhibits comparably remarkable outcomes, achieving an R^2 of 0.9259 ± 0.0038 and a MAPE of 6.140%, thereby suggesting that the model is capable of reliably forecasting prices across the complete array of commonly utilized VM configurations.

The model is remarkably effective across the high-end segments (15-50$ and 50-800$), achieving R^2 values of 0.9817 and 0.9167, respectively. These results suggest that XGBoost is effective in explaining over 91% of the variance, even among the most expensive VM categories. Moreover, the values of MAPE in the above segments fall to less than 4.5%, showing that the forecasts tend to be very close to actual pricing, deviating normally by no more than a few percentage points.

Noticeably, XGBoost significantly alleviates the fluctuating performance observed in Random Forest in the $3-5 range. Even as this range was the most challenging, with an R^2 value of 0.4708 ± 0.0060, the performance obtained is positive and significantly higher than the negative results realized by Random Forest (-1.257). A Mean Absolute Percentage Error (MAPE) value of 7.066% within this range, even as being higher than the adjacent ranges, remains reasonable for practical applications. This uniformity among the cross-validation folds, reflected by low standard deviations (typically lower than 0.01 for the values of R^2), demonstrates the stability and reliability of the model. This uniformity is significant, given the complexity of the algorithm that underlies the procedure, and suggests that the regularization methods used by XGBoost effectively reduce overfitting while retaining high predictive performance.

The capability of the model to correct prediction defects through boosting, as well as to uncover rich nonlinear relationships, allows it to adequately describe the complex

pricing dynamics that are common to the market for cloud computing (see 5.3). It is found that XGBoost stands as the most accurate model among those evaluated, yielding the required accuracy and trustworthiness for real-world applications of VM price prediction. Improvements realized across multiple price ranges affirm the effectiveness of advanced ensemble methodologies in coping with complex, real-world price datasets and suggest that XGBoost is the best compromise between the model's complexity and forecasting effectiveness for this particular prediction task.

Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	0.043	0.023	0.975	8.514	32509
(1, 3)	0.196	0.116	0.870	6.226	14638
(3, 5)	0.391	0.274	0.464	7.052	6739
(5, 15)	0.771	0.531	0.925	6.187	10750
(15, 50)	1.328	0.887	0.981	3.599	6015
(50, 800)	18.467	6.126	0.921	4.478	1583

Table 6.19: XGBoost metrics for Fold 1 by price range

Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	0.042	0.023	0.976	8.586	32553
(1, 3)	0.198	0.116	0.868	6.232	14727
(3, 5)	0.383	0.271	0.481	7.001	6696
(5, 15)	0.749	0.528	0.930	6.098	10653
(15, 50)	1.372	0.900	0.980	3.669	6118
(50, 800)	25.617	7.330	0.892	4.497	1489

Table 6.20: XGBoost metrics for Fold 2 by price range

Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	0.043	0.023	0.975	8.426	32387
(1, 3)	0.193	0.114	0.874	6.175	14699
(3, 5)	0.389	0.274	0.470	7.075	6728
(5, 15)	0.785	0.522	0.922	6.056	10863
(15, 50)	1.341	0.899	0.982	3.638	6062
(50, 800)	21.525	6.586	0.904	4.426	1495

Table 6.21: XGBoost metrics for Fold 3 by price range

Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	0.043	0.023	0.975	8.511	32393
(1, 3)	0.194	0.114	0.872	6.220	14685
(3, 5)	0.387	0.273	0.469	7.067	6787
(5, 15)	0.783	0.531	0.923	6.159	10851
(15, 50)	1.290	0.888	0.982	3.627	6019
(50, 800)	17.098	6.196	0.931	4.528	1494

Table 6.22: XGBoost metrics for Fold 4 by price range

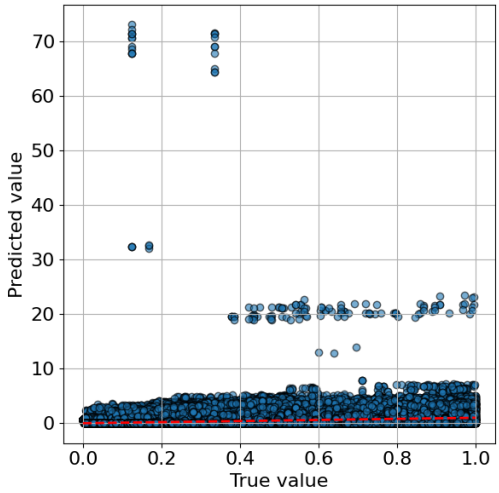
Price Range	RMSE	MAE	R^2	MAPE (%)	N
(0, 1)	0.043	0.023	0.975	8.571	32365
(1, 3)	0.197	0.118	0.868	6.394	14761
(3, 5)	0.394	0.277	0.470	7.137	6710
(5, 15)	0.750	0.531	0.930	6.201	10770
(15, 50)	1.281	0.893	0.983	3.663	6113
(50, 800)	18.035	5.882	0.935	4.131	1516

Table 6.23: XGBoost metrics for Fold 5 by price range

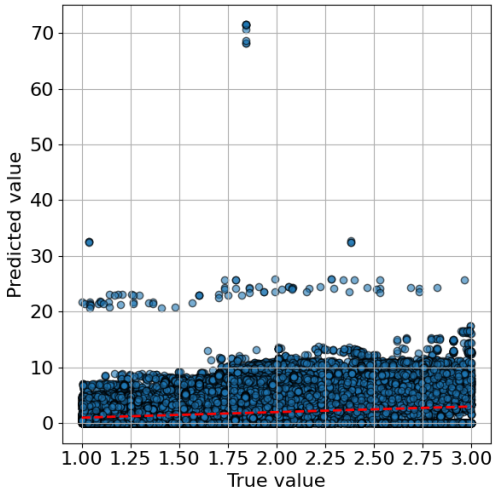
Price Range	RMSE	MAE	R^2	MAPE (%)	N_total
(0, 1)	0.0425 ± 0.0005	0.0229 ± 0.0002	0.9751 ± 0.0005	8.522 ± 0.063	162207
(1, 3)	0.1957 ± 0.0022	0.1155 ± 0.0014	0.8704 ± 0.0027	6.249 ± 0.084	73510
(3, 5)	0.3888 ± 0.0040	0.2737 ± 0.0023	0.4708 ± 0.0060	7.066 ± 0.049	33660
(5, 15)	0.7675 ± 0.0173	0.5286 ± 0.0041	0.9259 ± 0.0038	6.140 ± 0.061	53887
(15, 50)	1.3222 ± 0.0376	0.8932 ± 0.0059	0.9817 ± 0.0009	3.639 ± 0.028	30327
(50, 800)	20.149 ± 3.4784	6.424 ± 0.5662	0.9167 ± 0.0183	4.412 ± 0.161	7577

Table 6.24: Summary of XGBoost metrics by price range (mean ± std over 5 folds)

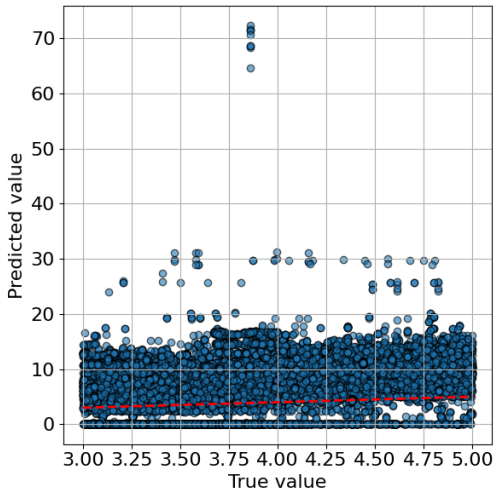
For completeness of analysis, Fig. 6.4 shows scatter plots with actual values and values predicted by the model. The further the points deviate from the line drawn, the more inaccurate the model is in the regression. Here is possible to observe graphically that XGBoost is more accurate than the models analyzed previously if the Fig. 6.4 is compared with Fig. 6.1, Fig. 6.2, and Fig. 6.3.



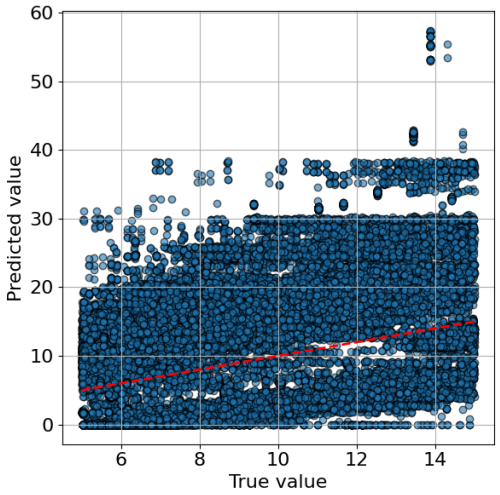
(a) Scatter plot range (0, 1) Linear Regression



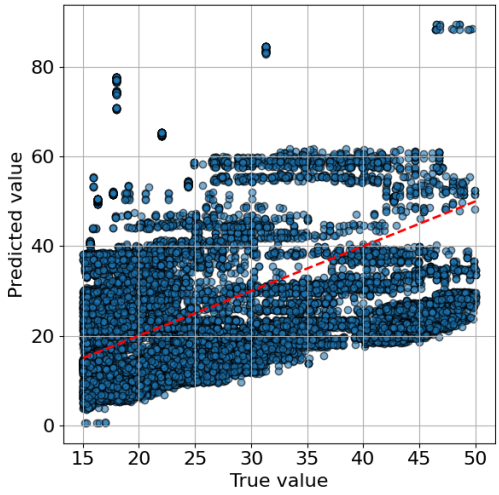
(b) Scatter plot range (1, 3) Linear Regression



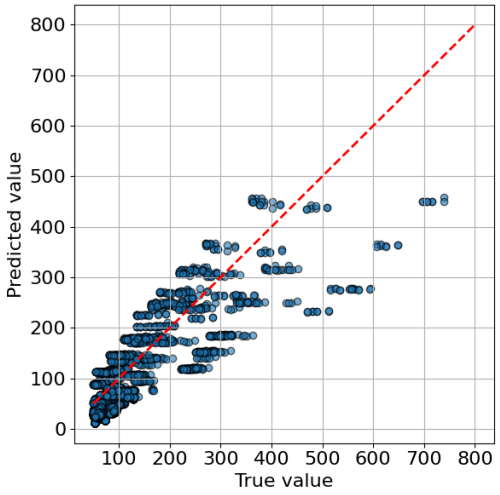
(c) Scatter plot range (3, 5) Linear Regression



(d) Scatter plot range (5, 15) Linear Regression

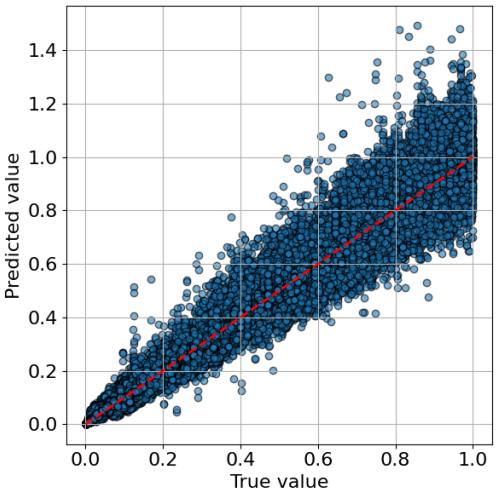


(e) Scatter plot range (15, 50) Linear Regression

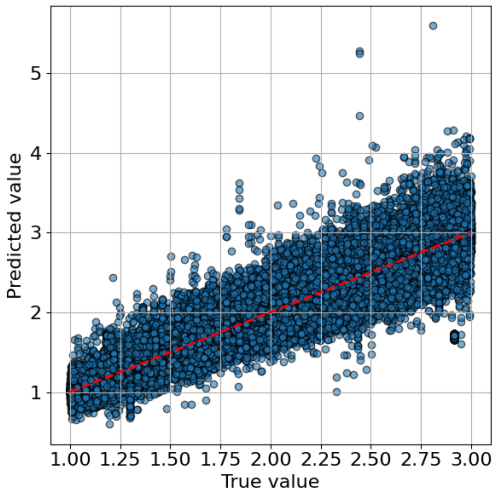


(f) Scatter plot range (50, 800) Linear Regression

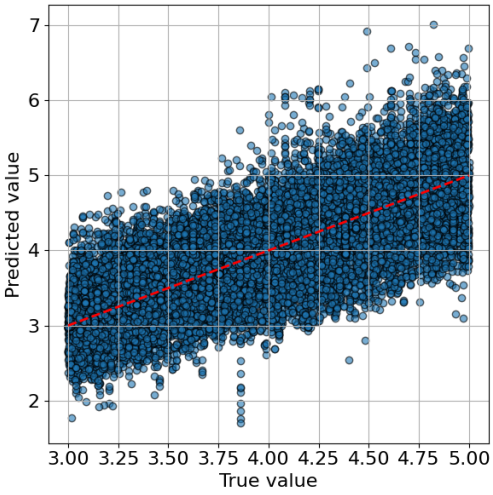
Figure 6.1: Set of scatter plots for Linear Regression by price range



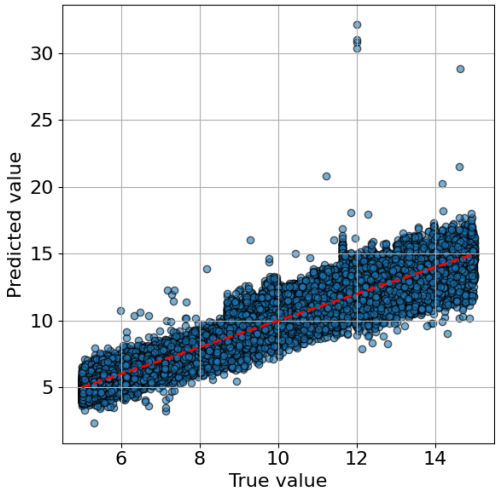
(a) Scatter plot range (0, 1) RBFNN



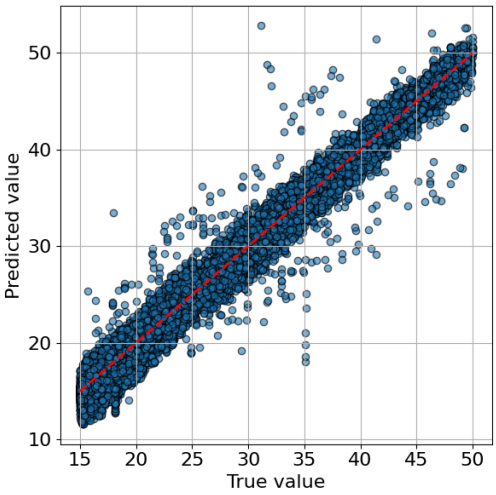
(b) Scatter plot range (1, 3) RBFNN



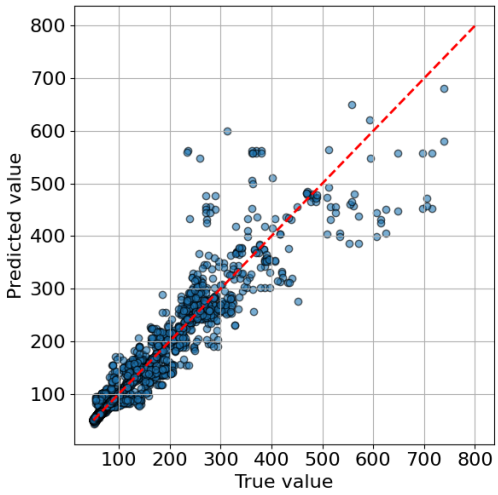
(c) Scatter plot range (3, 5) RBFNN



(d) Scatter plot range (5, 15) RBFNN

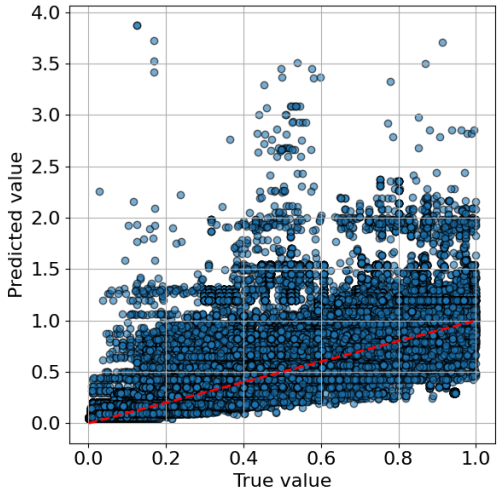


(e) Scatter plot range (15, 50) RBFNN

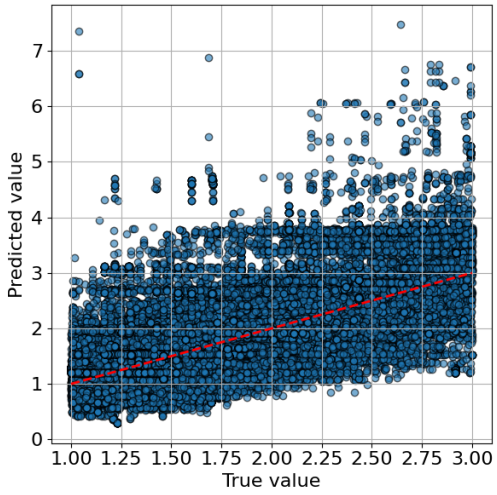


(f) Scatter plot range (50, 800) RBFNN

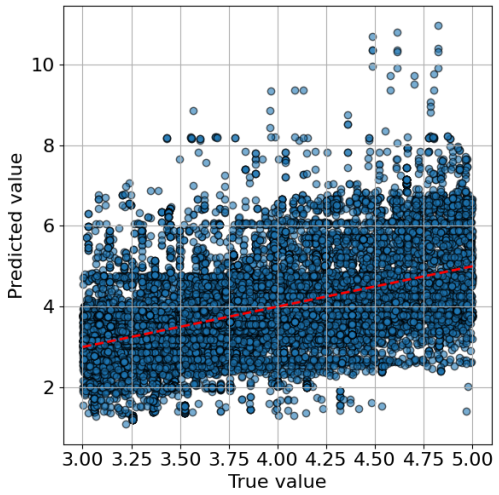
Figure 6.2: Set of scatter plots for RBFNN by price range



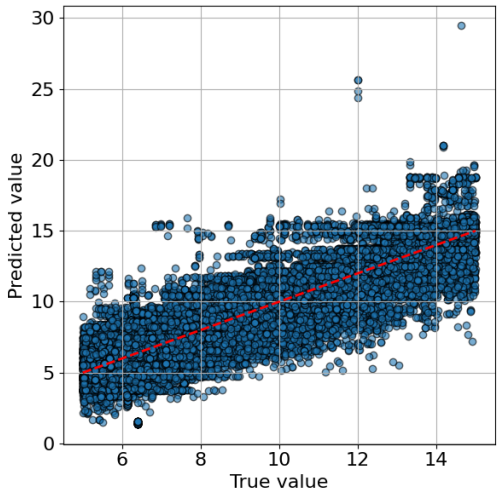
(a) Scatter plot range (0, 1) Random Forest



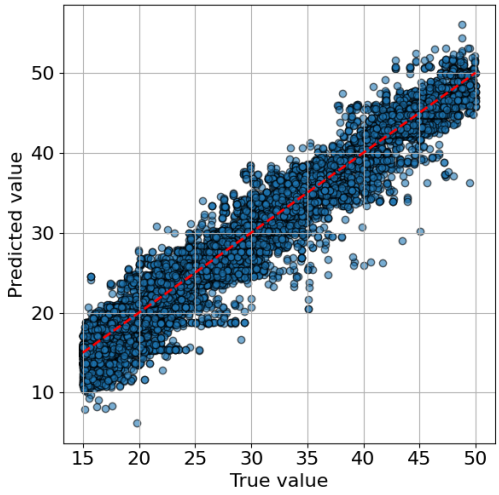
(b) Scatter plot range (1, 3) Random Forest



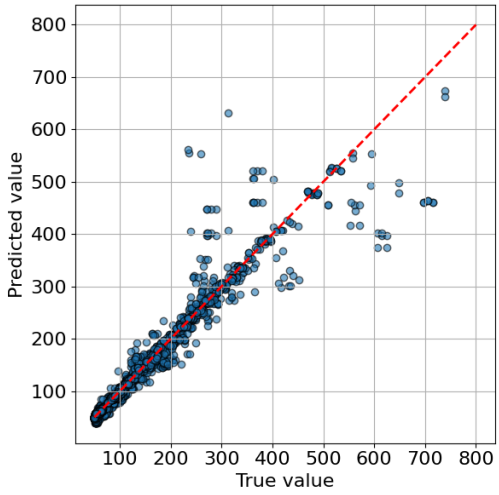
(c) Scatter plot range (3, 5) Random Forest



(d) Scatter plot range (5, 15) Random Forest

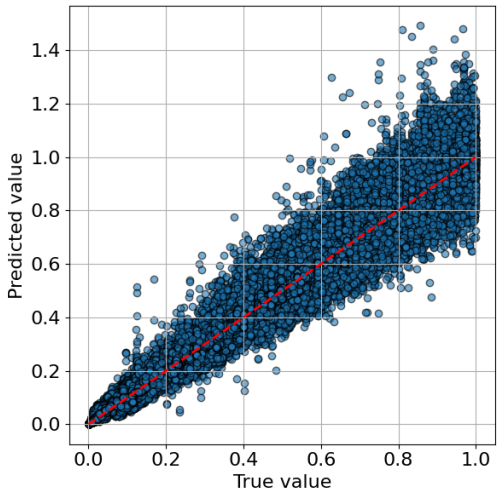


(e) Scatter plot range (15, 50) Random Forest

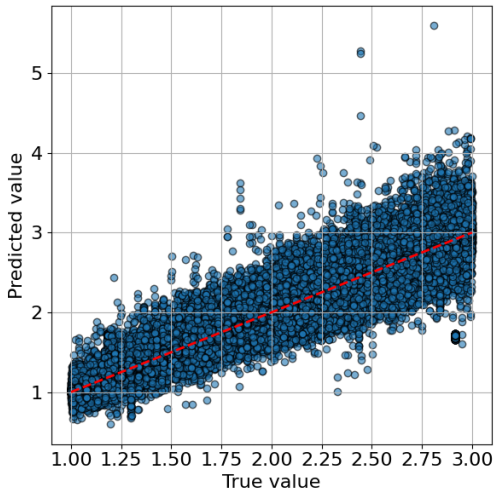


(f) Scatter plot range (50, 800) Random Forest

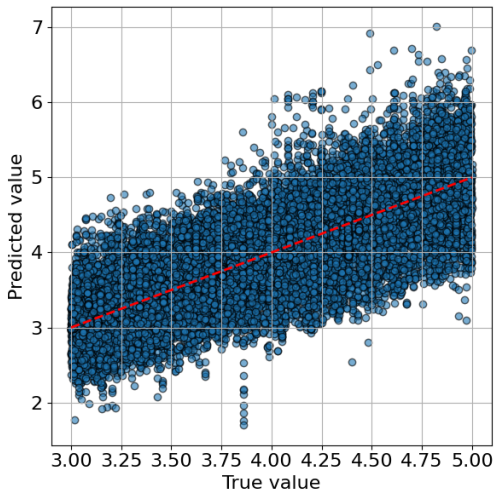
Figure 6.3: Set of scatter plots for Random Forest by price range



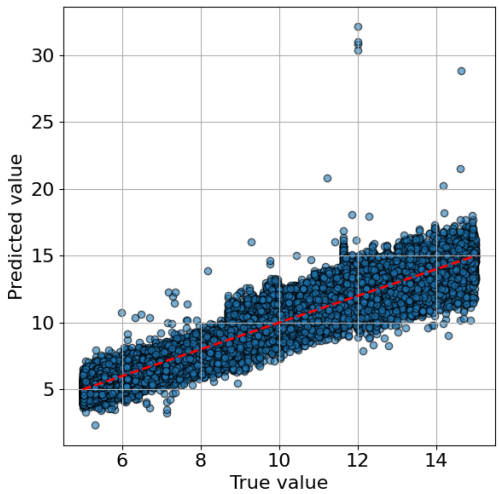
(a) Scatter plot range (0, 1) XGBoost



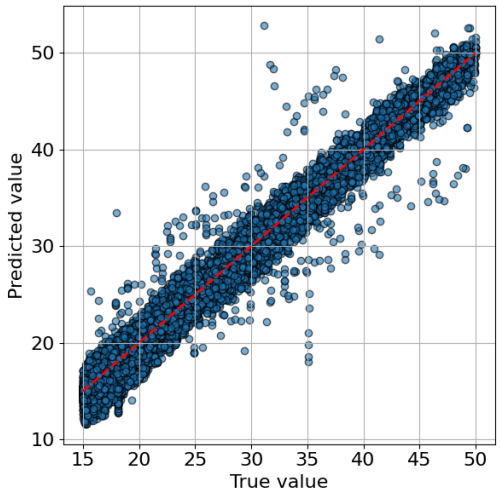
(b) Scatter plot range (1, 3) XGBoost



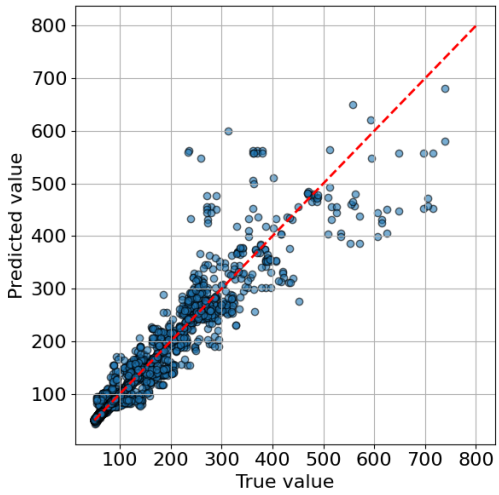
(c) Scatter plot range (3, 5) XGBoost



(d) Scatter plot range (5, 15) XGBoost



(e) Scatter plot range (15, 50) XGBoost



(f) Scatter plot range (50, 800) XGBoost

Figure 6.4: Set of scatter plots for XGBoost by price range

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis conducts a study on cloud virtual machine pricing, starting with data collection, analyzing and identifying specific patterns, and concluding with training a machine learning model to estimate prices for an arbitrary configuration that takes into account, among other features, the number of CPU cores and quantity of RAM, and the region in which the resource is provisioned.
The analysis highlighted how the cloud virtual machine pricing landscape is extremely fragmented and difficult to interpret. Differences are not limited to different providers, but also emerge within the same provider, depending on the Region where the service is provisioned.
The data collection process has proven complex due to the lack of standardization in price catalogs. Each provider adopts different formats and methods, making it difficult to maintain automatic procedures based on scraping.
From a comparative perspective, it emerged that European providers tend to offer more competitive prices than larger competitors, despite having smaller infrastructures. This result suggests a strategic positioning aimed at fostering market competitiveness. Another important aspect to note is how few providers offer instances with GPUs, which demonstrates a still immature approach to cloud graphics power; this pattern is also highlighted by the major use cases of cloud services, which primarily represent CPU-centric processes.
Finally, the analysis of predictive models showed that VM pricing does not follow a linear dynamic. Linear regression models, therefore, provided limited performance, while boosting approaches, particularly XGBoost, proved more effective in capturing the complexity and discontinuities inherent in the pricing structure. These findings pave the way for future investigations using advanced machine learning techniques to further refine the forecasting and analysis capabilities of the cloud computing market.

7.2 Future improvements

The study was carried out on a set of 20 providers, most of which were European. A natural extension of this work would be to expand the set of providers considered, thereby obtaining a broader and more representative picture of the market.
However, increasing the number of providers also implies a significant effort in terms

of data collection and management.
To address the collection limitation, a possible solution could be the establishment of commercial agreements with the providers themselves, granting access to regularly updated price lists (e.g., on a monthly basis). This approach would make it possible to avoid the maintenance of a dedicated data extraction tool for each individual provider, while at the same time ensuring greater consistency, reliability, and timeliness of the dataset used for future analyses.
In addition, it would be useful not only to extend the analysis to virtual machines, which form the core of cloud computing, but also to integrate other services, such as managed databases, object store solutions, and even serverless solutions. Within this broader context, it may be possible to conduct a more comprehensive analysis of cloud service portfolios that encompass a wider range of use cases and provide richer insights into marketplace dynamics and the strategies of providers.

Bibliography

[1] Ahmed A. Alkrush et al. "Data centers cooling: A critical review of techniques, challenges, and energy saving solutions". In: *International Journal of Refrigeration* 160 (2024), pp. 246–262. ISSN: 0140-7007. DOI: https://doi.org/10.1016/j.ijrefrig.2024.02.007. URL: https://www.sciencedirect.com/science/article/pii/S0140700724000458.

[2] Inc. Amazon Web Services. *EC2 On-Demand Instance Pricing*. 2025. URL: https://aws.amazon.com/ec2/pricing/on-demand/ (visited on 08/20/2025).

[3] Michael Armbrust et al. *A view of cloud computing*. Vol. 53. 4. ACM, 2010, pp. 50–58.

[4] Mariam Arzumanyan et al. "Geospatial suitability analysis for data center placement: A case study in Texas, USA". In: *Sustainable Cities and Society* (2025), p. 106687. ISSN: 2210-6707. DOI: https://doi.org/10.1016/j.scs.2025.106687. URL: https://www.sciencedirect.com/science/article/pii/S221067072500561X.

[5] Elemento Cloud. *AtomOS*. 2025. URL: https://www.elemento.cloud/en/technology/atomos (visited on 07/18/2025).

[6] Elemento Cloud. *Cloud Network*. 2025. URL: https://www.elemento.cloud/en/technology/cloud-network (visited on 07/18/2025).

[7] Elemento Cloud. *Elemento*. 2025. URL: https://www.elemento.cloud/en (visited on 07/18/2025).

[8] *Cloud Mercato*. 2025. URL: https://www.cloud-mercato.com/ (visited on 07/23/2025).

[9] Cloudflare. *What is vendor lock-in?* 2025. URL: https://www.cloudflare.com/learning/cloud/what-is-vendor-lock-in/ (visited on 07/22/2025).

[10] CloudPrice.net. *CloudPrice.net: Cloud Instance Price and Performance Comparison*. 2025. URL: https://cloudprice.net/ (visited on 08/25/2025).

[11] GeeksforGeeks contributors. *Decision Tree in Machine Learning*. Last Updated: 30 Jun, 2025. 2025. URL: https://www.geeksforgeeks.org/machine-learning/decision-tree/ (visited on 09/15/2025).

[12] GeeksforGeeks contributors. *Linear Regression in Machine Learning*. Last updated: 29 Jul, 2025. 2025. URL: https://www.geeksforgeeks.org/machine-learning/ml-linear-regression/ (visited on 09/15/2025).

[13] GeeksforGeeks contributors. *Random Forest Algorithm in Machine Learning*. Last Updated: 01 Sep, 2025. 2025. URL: https://www.geeksforgeeks.org/machine-learning/random-forest-algorithm-in-machine-learning/ (visited on 09/15/2025).

[14] GeeksforGeeks contributors. *Understanding Logistic Regression in Machine Learning*. Last updated: 02 Aug, 2025. 2025. URL: https://www.geeksforgeeks.org/machine-learning/understanding-logistic-regression/ (visited on 09/15/2025).

[15] Wikipedia contributors. *Power law*. Accessed: 2025-08-31. 2025. URL: https://en.wikipedia.org/wiki/Power_law (visited on 08/31/2025).

[16] Wikipedia contributors. *Zipf's law*. Accessed: 2025-08-31. 2025. URL: https://en.wikipedia.org/wiki/Zipf%27s_law (visited on 08/31/2025).

[17] Lorenzo Corneo. *Surrounded by the Clouds: A Comprehensive Cloud Reachability Study*. 2021. URL: https://labs.ripe.net/author/lorenzo-corneo/surrounded-by-the-clouds-a-comprehensive-cloud-reachability-study/ (visited on 08/18/2025).

[18] DMLC. *XGBoost: A Scalable Tree Boosting System*. Accessed: 2025-09-15. 2025. URL: https://github.com/dmlc/xgboost (visited on 09/15/2025).

[19] Elemento-Modular-Cloud. *models.json — elemento-pciid-mapper*. Accessed on the GitHub repository of the elemento-pciid-mapper project. 2025. URL: https://github.com/Elemento-Modular-Cloud/elemento-pciid-mapper/blob/master/models.json (visited on 09/05/2025).

[20] IONOS Inc. *Cloud Price List*. 2025. URL: https://docs.ionos.com/support/general-information/price-list/ionos-cloud-inc (visited on 08/22/2025).

[21] IONOS Inc. *IONOS Cloud Inc. Price List*. Last updated on site 2 months prior to access, net prices without VAT. 2025. URL: https://docs.ionos.com/support/general-information/price-list/ionos-cloud-inc (visited on 08/31/2025).

[22] Matthew Kowal. *Understanding Mutual Information*. 2020. URL: https://mkowal2.github.io/posts/2020/01/understanding-mi/ (visited on 08/29/2025).

[23] UpCloud Ltd. *API Documentation: Pricing*. 2025. URL: https://developers.upcloud.com/1.3/4-pricing/#list-prices (visited on 08/21/2025).

[24] Cloud Mercato. *Public Cloud Reference*. 2025. URL: https://pcr.cloud-mercato.com/ (visited on 08/25/2025).

[25] Cloud Mercato. *Public Cloud Reference: Providers*. 2025. URL: https://pcr.cloud-mercato.com/providers (visited on 08/25/2025).

[26] *Global Disruption of Semiconductor Supply Chains During COVID-19: An Evaluation of Leading Causal Factors*. Vol. Volume 2: Manufacturing Processes; Manufacturing Systems. International Manufacturing Science and Engineering Conference. June 2022, V002T06A011.

[27] Author Name and Co-Author Name. "Modeling for CPU-Intensive Applications in Cloud Computing". In: *IEEE Conference Proceedings* (2015). Available at: https://ieeexplore.ieee.org/document/7336138/. DOI: 10.1109/HPCC-CSS-ICESS.2015.128.

[28] Author Name and Co-Author Name. "Resource Optimization Strategy for CPU Intensive Applications in Cloud Computing Environment". In: *IEEE Conference Proceedings* (2016). Available at: https://ieeexplore.ieee.org/document/7545907. DOI: 10.1109/CSCloud.2016.29.

[29] Justice Opara-Martins, Reza Sahandi, and Feng Tian. "Critical review of vendor lock-in and its impact on adoption of cloud computing". In: *International Conference on Information Society (i-Society 2014)*. 2014, pp. 92–97. DOI: 10.1109/i-Society.2014.7009018.

[30] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[31] Vagelis Plevris et al. "Investigation of performance metrics in regression analysis and machine learning-based prediction models". In: June 2022. DOI: 10.23967/eccomas.2022.155.

[32] Selenium Project. *SeleniumHQ: Browser Automation*. 2025. URL: https://www.selenium.dev/ (visited on 08/31/2025).

[33] CloudFerro S.A. *Virtual Machines (VM) — Pricing Tables, WAW3-1 Cloud*. 2025. URL: https://cloudferro.com/pricing/pricing-tables/waw3-1-cloud/virtual-machines-vm/ (visited on 08/31/2025).

[34] Amazon Web Services. *Boto3 Documentation: Pricing Client - list_price_lists*. 2025. URL: https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/pricing/client/list_price_lists.html (visited on 08/22/2025).

[35] Abhay Singh. *Mastering XGBoost: The Ultimate Guide to Extreme Gradient Boosting*. 2025. URL: https://medium.com/@abhaysingh71711/mastering-xgboost-the-ultimate-guide-to-extreme-gradient-boosting-ac7fa2828047 (visited on 09/22/2025).

[36] Vantage. *Amazon EC2 Instance Comparison*. 2025. URL: https://instances.vantage.sh/ (visited on 08/25/2025).

[37] VMware. *VMware*. 2025. URL: https://www.vmware.com/ (visited on 08/18/2025).