Simulation Modelling Practice and Theory 115 (2022) 102441

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

A pricing model for Container-as-a-Service, based on hedonic indices

Vasiliki Liagkou, George Fragiadakis *, Evangelia Filiopoulou, Christos Michalakelis, Thomas Kamalakis, Mara Nikolaidou

Harokopio University of Athens, Department of Informatics and Telematics, Greece

ARTICLE INFO

Keywords:
Modeling
Container as a Service
Hedonic price index
Pricing models

ABSTRACT

Container-as-a-service (CaaS) is a business model that facilitates software developers in organizing, running, managing, and deploying applications using container-based virtualization. The CaaS market is a fast-growing one, leading infrastructure-as-a-service (IaaS) providers into offering CaaS solutions built on top of their IaaS platforms. Given the increasing competition among providers, pricing strategies play a key role in determining the business prospects of CaaS.

In this work we propose a methodology to analyze the pricing strategies of the providers, based on a hedonic regression. Following that, we also develop a hedonic price index, based on data collected from popular providers, by identifying a wide set of architectural requirements influencing the pricing strategy. We further analyze the correlations between these requirements and define the subset to be included in the regression analysis, interpreting the results in view of the underlying CaaS market characteristics. The model implementation and the data set are freely available on the web under an open-source license.

1. Introduction

Containerization is increasingly being adopted, allowing organizations to modernize their legacy apps and streamline their infrastructure thus reducing time-to-market due to the more consistent release cycles. Container technologies allow the packaging and isolation of applications with their entire runtime environment and configuration files. Containers share the host operating system (OS) kernel and hence are exceptionally lightweight. Applications can be transferred to different cloud or OS environments without any integration or runtime issues because of the change in environment settings [1]. Compared to conventional virtual machines (VMs) which require a dedicated OS, containers consume lower resources allowing more effective utilization in terms of central processing unit (CPU) and random access memory (RAM) [2]. However, the main drawback between VMs and containers is that because all containers share the same kernel are therefore less isolated than VMs. A compromised kernel leads to compromised containers [3]. These security concerns motivate researchers to study solutions that combine both virtualization technologies by building containers on top of VMs so as to leverage the VM's strong trust boundary and the container's better efficiency.

Given the clear advantages of container technologies for both users and providers, a subscription-based cloud model has emerged, known as Container as a Service (CaaS). CaaS lies between Infrastructure as a Service (IaaS) and Platform as Service (PaaS). Under the CaaS paradigm, users can remotely configure, run, scale, manage and stop containers in the cloud, using the provider's platform

* Corresponding author.
E-mail addresses: vliagkou@hua.gr (V. Liagkou), gfragi@hua.gr (G. Fragiadakis), evangelf@hua.gr (E. Filiopoulou), michalak@hua.gr (C. Michalakelis), thkam@hua.gr (T. Kamalakis), mara@hua.gr (M. Nikolaidou).

https://doi.org/10.1016/j.simpat.2021.102441

Available online 25 November 2021

software [4] (e.g. Kubernetes) allowing managing of cluster elements, automating scaling and failure management and incorporating governance and security considerations [5]. The adoption of CaaS is rapidly spreading and Gartner predicts that by 2022, more than 75% of global organizations will be running containerized applications in production, up from less than 30% today [6]. Given the prospects of CaaS, it is important to understand the pricing strategy of cloud providers for this emerging service. In this paper, we use regression analysis to formulate a model for CaaS pricing, given the data available from major cloud providers and a hedonic price index is constructed. Hedonic pricing is a model that identifies price factors according to the premise that price is determined both by internal characteristics of the good/service being sold and external factors affecting it [7].

Our analysis takes into account several architectural requirements and identifies key constituent characteristics. Using hedonic indices, we analyze the correlation of these characteristics and prioritize them in terms of their regression coefficients. The study of the correlations reduces the number of characteristics that must be accounted for and therefore significantly simplifies the hedonic function model. Both our `Python` implementation and our data are freely available on the web under an open-source license. The rest of the paper is organized as follows: Section 2, provides a literature review on could pricing and other work related to this study. Section 3 outlines key CaaS concepts pertinent to price modeling, while 4 overviews the hedonic function model adopted in this work. Sections 5 and 6 present our full modeling methodology and the obtained results. Some concluding remarks and future outlook are provided in Section 7.

2. Related work

Pricing is a critical component in cloud computing, because it affects directly both cloud providers' revenue and customers' budget [8]. There are several proposed schemes that discuss and examine pricing in cloud computing, as well as approaches that combine pricing and hedonic methods. Pertinent literature includes studies that address cloud computing pricing or discuss cloud pricing based on a hedonic index.

In [9], a genetic algorithm was developed that optimizes cloud price rates, thus increasing the provider's profits. Moreover, an auction approach was proposed, where the users could bid for a price, maintaining optimal QoS and increasing in parallel the provider's revenue [10]. In [11], a stochastic approach was adopted to show that a contract-based model could achieve higher provider revenue. In [12] a statistical model was presented to study the influence of functional/non-functional characteristics on price rates, illustrating the strong influence of functional attributes in IaaS pricing strategies.

Hedonic pricing methods have been applied to the housing [13], personal computer (PC) [14–18], mobile phone [19–21] and telecom markets [22]. In cloud computing, hedonic methods have been applied to a rather limited scope. The work of [23] was one of the first studies introducing hedonic regression to analyze the IaaS offering. In this work, the hedonic function was applied for five major IaaS providers taking into account various VM characteristics (e.g. memory, CPU and storage). A price comparison method was also developed to compare the cloud service providers and find the best possible one in terms of price. The authors in [24] separated the intrinsic from the extrinsic factors including additionally time-dummy variables in their hedonic model, aiming to analyze historical data for Amazon Web Services (AWS) cloud instance prices between 2008 and 2017. A similar approach was followed in [25] for AWS's historical data analysis, extended to include computing, database and storage cloud services. Furthermore, AWS's instance pricing was studied in [26] based on a linear regression method paying particular attention to on-demand instances. Using the same method authors in [27] analyzed the impact of the geographical location factor, while in [28], the authors considered the development of a pricing model according to which customers seek the cheapest service with an ideal quality of service (QoS).

The authors of the current work constructed hedonic price indices based on IaaS services in previous works [29,30]. The IaaS services were initially described by functional attributes (e.g. CPU, RAM) [29] and then non-functional characteristics were also included [30]. In both aforementioned papers data was also collected by major providers. In the current work a hedonic price index is constructed based on CaaS services. Initially, the functional and non-functional attributes of Caas are presented and then the pricing model for CaaS services is provided by constructing a price index. The price index reveals the extent at which both the functional and the non-functional attributes contribute to the shaping of the CaaS service price. Our model can be applied to help users to dimension and evaluate the cost of deploying their container infrastructure in the cloud. Providers can also apply the model in order to make sure that their pricing strategy clearly reflects their infrastructure component costs. Another important contribution of our work is that both our model implementation in `Python` and our data set is freely available on the web under an open-source license, allowing the various cloud actors and researchers to adapt it to their case studies as well.

3. The containers as a service environment

The container-based cluster architecture is illustrated in Fig. 1. Containerized applications are running in container hosts, called "nodes", which are grouped into clusters. Each cluster consists of several nodes which be VMs or even physical servers. Containers communicate each other using container network bridges and overlays. Storage volumes are used in case data persistence is needed.

The clusters need to handle a large numbers of containers and services indicating that the cluster management becomes essential and requires orchestration support. Orchestration frameworks are software solutions that manage containerized applications across distributed cluster of nodes [31]. They provide key features like initialization, monitoring, scaling, load balancing, self-healing and customization throughout the cluster. They typically adopt a master–slave architecture where the master nodes, mostly known control planes, are responsible to maintain the desired state of the cluster while the slave nodes are managed by the master node and run the containers. Administrator and development teams issue commands at the master nodes which relay these instructions to

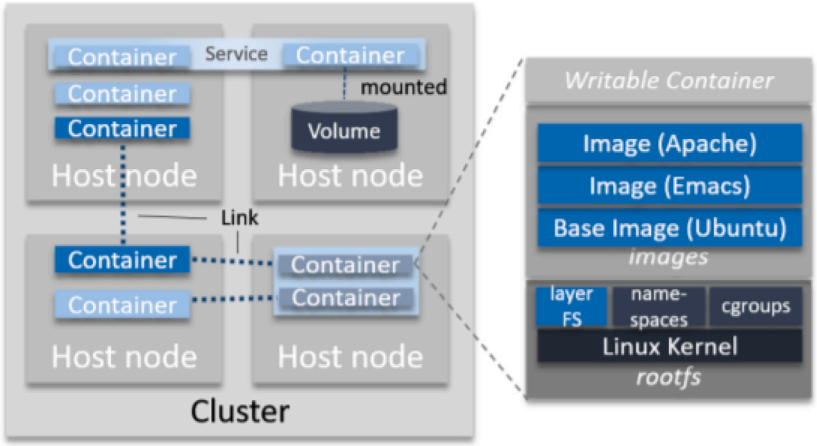

Fig. 1. Container cluster architecture [1].

slave nodes. Commonly used container management tools include Kubernetes, OpenShift and Docker Swarm Manager. At the time of this writing, Kubernetes [32] has become the industry's leading container orchestration tool [33].

The available methods to deploy orchestration platforms on clusters may be divided into three categories [34]: on-premise solutions and deployment on the cloud with or without using managed services. The second option where clusters are deployed with managed services is the scope of the current work while the third one is based on IaaS services where the users build their own orchestration platform in virtual machines or dedicated servers without any management from the provider. In this option CaaS providers bundle the container engine, orchestration tools and underlying compute resources into a unified service. Enterprises can therefore deploy a production-grade cluster in a few minutes minimizing the time spent on environment setup and operating costs. All major cloud providers including Amazon, Microsoft, and Google, currently incorporate CaaS solutions in their portfolios. Most of the CaaS solutions are based on Kubernetes orchestration platform, e.g. Google's Kubernetes Engine (GKE). However, there are a few CaaS solutions which are not based on Kubernetes. For example, the Amazon's Elastic Container Service (ECS) [35] is based on customized orchestration platform.

3.1. Pricing strategies

Cloud vendors have developed different pricing strategies to attract their customers in order to leverage CaaS model. Billing starts when the nodes are first created until the nodes are deleted. Providers may charge an hourly flat fee for cluster management, irrespective of cluster size and topology. Concerning node instances, the customers are billed for each node depending on the pricing strategy which include pay-as-you-go options where users pay for resources they are actually using, based on the hourly rates. In subscription-based options, customers are committed to a consistent amount of usage for a long term period from one to three years [36]. Customers can achieve lower prices if they pay the partial or entire cost upfront. Recently, provider started offering an alternative option to their customers for highly granular pricing [37]. The customers can pay only for the resources their actual containerized workloads consumes, not for the resources that the system containers or the OS itself may consume. Besides the aforementioned costs that are based mainly on infrastructure costs, the final price depends on the cluster architecture.

3.2. Architectural requirements

The providers give the option to the customers to design their own cluster architecture which is certainly not a trivial task. There are many options to select from and many factors to consider which could incur additional costs. The price of the cluster will vary depending on the number of nodes, compute capacity, storage space, etc. Furthermore, the customers may need to adapt the cluster architecture to meet production requirements including high availability, disaster recovery, persistent storage facilities, hybrid and multi-cloud support, security issues etc [31]. Some important architectural requirements are highlighted and described below. They can be grouped in three major categories: resource provisioning, reliability and elasticity.

Resource provisioning

Resource provisioning and allocation are the very first requirements the administrators need to address in order to run the workloads. They need to size the cluster to provide enough capacity, CPU and memory, for the containers. The sizing should be optimized to maximize the infrastructure utilization without service disruptions while at the same time avoiding over-provisioning. The geographical location of the cluster nodes is also one of the factors that need to be considered. Placing the nodes close to end-users can keep latency low but may affect the overall costs since the node prices can vary across different geographical zones. Furthermore, depending on whether Linux or Windows-based containers are required, the node's operating system should be configured accordingly.

Reliability and high availability

These are the key requirements for an orchestration solution that the customers can rely on. High availability implies that the user services will not be disrupted when some cluster components fail. The reliability concept includes many factors that need to be considered and make the decision-making process more difficult:

- *Redundancy* is the foundation of reliable and highly available clusters [38]. The orchestration platform itself replicates only the containers, not the master or the slave nodes of the cluster. Distribution and replication of nodes across different geographical zones in a region or across different regions is a proactive measure the administrators should configure so as to survive the cluster in case of physical data center disaster.
- *Spot or on-demand node instances* is another point that could affect the reliability of the cluster. Although the infrastructure cost could be reduced using spot instances, these instances are unreliable by their nature. Different systems may have heterogeneous requirements for availability and the running applications may not be flexible enough to afford loosing the instances anytime.
- *Data store and protection* is another problem in orchestration frameworks. The containers are fundamentally ephemeral. When the container dies, the created data is lost. The orchestration platform itself is not in charge to manage persistence volumes. It only consumes them [38]. Thus, the administrators need to consume persistent storage as well as add redundancy to prevent data loss.

Elasticity

Elasticity has been boosted by container technology since the latter enables fast bootstrapping. In a micro-service architecture, applications can be decomposed into several containers allowing fine-grain scalability. Orchestration solutions scale specific containers independently to meet the application needs [39]. There are various important situations where clusters need to scale, the most obvious one of which is to ensure that there is no performance degradation when service demand increases. Containers and nodes can scale both horizontally by adding and removing instances and vertically by adjusting the node specifications in terms of CPU and memory. Security requirements including data sensitivity the containers may carry or due to the varying legislation per region may also be addressed with proper scaling. Organization may opt for scaling across multiple cloud providers to avoid vendor lock-in or over-committing to any one cloud vendor [38].

3.3. Identification of architectural requirements

Based on the above, we conclude that there are various factors that could have a significant impact on decision-making process and pricing strategy. We outline these factors that will actually identify the input variables in the regression analysis in this subsection. Table 2 summarizes these factors along with various value options.

Resource provisioning

- *CPU*, defines the size in cores of the entire cluster independent on the number of nodes.
- *RAM*, memory defines the size in Gigabytes (GBs) of the whole cluster independent on the number of nodes.
- *Storage*, defines the storage capacity of the cluster in GBs per month.
- *Disk type*, is the type of storage disk. A 2-level scale was used referring to Standard or solid-state disk (SSD) type.
- *Internal Egress Traffic*, is the number of GBs exchanged between nodes in the cluster per month. Egress traffic is charged based on whether the traffic crosses zone or region boundaries. Data transferred between nodes in the same zone is free. Data transferred across different zones in the same region is charged to the node that is sending the traffic or in each direction depending on the cloud provider.
- *External Egress Traffic*, is the number of GBs sent by all cluster nodes to the internet per month. The incoming data traffic is not charged from providers.
- *OS*, is the operating system the slave nodes are running. A 3-level scale is used distinguish between unlicensed and commercial Linux or Windows-licensed OS.
- *Region*, is the geographical location where the cluster is hosted. A 4-level scale is used to distinguish Europe, US, South America and Asia.

Reliability and high availability

- *Regional redundancy*, determines whether a cluster is regional or zonal. In regional clusters the master and slave nodes are distributed across multiple zones within a single region offering high availability, while in zonal clusters the control plane and the nodes are located in the same zone.
- *Instance type* indicates if the node instance is on-demand, spot or a dedicated host.

Fig. 2. Components of the methodology.

Elasticity

- *Auto-scaling* refers to the scaling feature inside a cluster. Horizontal auto-scaling means when it is possible to scale the number of containers or the nodes in the cluster. Vertical auto-scaling means when it is possible to scale the size of the containers or the cluster nodes. A 2-level scale was used referring to horizontal auto-scaling or to horizontal–vertical auto-scaling.
- *Hybrid and multi cloud support* indicates if the orchestration framework supports management of the slave nodes in multiple cloud providers or on-premise.
- *Vendor agnostic* indicates if the container orchestration framework relies on Kubernetes or on a different container orchestration platform.

Pricing options

- *Payment option*, indicates whether the customer pays nothing upfront, partial amount upfront or all amount upfront in case of commitment usage.
- *Term length*, indicates if the customer specifies a usage commitment which leads to discounts. A 3-level scale is used distinguish between no commitment and 1 year or 3 years commitment.
- *Pay-per-container usage*, indicates if the customer is charged based on the container usage capacity or based on the nodes usage capacity.
- *Cluster management fee*, is a flat fee for cluster management. The fee is charged per hour per cluster. However, some providers charge only the underlying resources of the cluster and do not require to pay the administration fee for the cluster.

4. The hedonic price index framework

Container pricing needs to take into account both the functional and non-functional attributes of the CaaS environment. The attributes that define and compose the CaaS services influence the final price, therefore it is important to examine and evaluate the contribution of each attribute to the pricing policy of the providers. Towards this direction, a hedonic price index modeling is applied. More specifically, CaaS services are described by 17 functional and non-functional requirements, as described in Section 3, each contributing to a different extent. Fig. 2 illustrates the components of the price index methodology framework.

4.1. Hedonic price indices

A hedonic price index is a price index based on a hedonic function, which in turn describes the relation between the price and the attributes of a product [7]. The framework considers that a service is a bundle of characteristics and that consumers purchase bundles of product characteristics, instead of the product itself, and calculates the contributory value of each characteristic [40]. Hedonic indices can be used to construct a quality-adjusted price index of a service. They are fast and easy to implement and can be adopted in order to examine several possible relations between market goods and their price [41]. A hedonic function, $f(\mathbf{X})$, relates the product's characteristics $\mathbf{X} = (X_1, \ldots, X_n)$ with its price P as follows:

$$P = f(\mathbf{X}) + e(\mathbf{X}) \tag{1}$$

where e is the residual errors. The hedonic function can be finally applied to N varieties of the product (or service), as presented in the following equation. If a linear hedonic function is assumed, then it is possible to write the price as a linear combination of the characteristics

$$P = b_0 + \sum_{i=1}^{n} b_i X_i + e(\mathbf{X}) \tag{2}$$

Table 1
VIF values and their impact.

VIF	Multicolinearity level
1	No multicolinearity
4–9	Moderate
10 or greater	Severe

where b_i are the estimated regression coefficients, often called implicit prices, as they indicate the prices charged and paid for an increment of one unit of the corresponding characteristic. They are influenced by market demand and supply and a number of other factors, e.g.the price of substitute and complementary products [42]. Eq. (2) constitutes the basis of the pricing model analyzed in this paper.

4.2. Attribute correlations

Within the proposed hedonic framework, P is a variable dependent on the input variables X_i, which describe the attributes of the CaaS service as presented in Table 2. By using multiple regression the relation between the dependent variable and the input variables can be highlighted. A key goal in constructing a stable price index is to isolate the relationship between each input and the dependent variables and ensure that the input attributes do not impose a high correlation [43]. Given the realizations X_{pk} and X_{qk} of the variables X_p and X_q respectively, their correlation C_{pq} can be calculated by the following formula:

$$C_{pq} = \frac{\sum_{k=1}^{N}(X_{pk} - \bar{X}_p)\sum_{k=1}^{N}(X_{qk} - \bar{X}_q)}{\sigma_p \sigma_q} \tag{3}$$

where $\bar{X}_p$ and $\bar{X}_q$ denote the average values of the variables X_p and X_q respectively, while σ_p and σ_q are the corresponding standard deviations. The correlation R_{pq} is a measure of the relationship between the two variable [44]. Multicollinearity occurs when the multiple linear regression analysis includes several variables that are significantly correlated not only with the dependent variable but also to each other. Multicollinearity reduces the precision of the estimated coefficients and weakens the rigor of the regression model [45].

One method for quantifying multicollinearity is to calculate the variance inflation factor (VIF) [46] given by:

$$\mathrm{VIF}_i = \frac{1}{1 - R_i^2} \tag{4}$$

where R_i^2 is the coefficient of determination obtained when X_i is regressed on all other input variables in the model. A value greater than 10 indicates the presence of strong multicollinearity and the corresponding variable should therefore be excluded from the model [46]. A general rule of thumb in such cases is described in Table 1

Once multicollinear explanatory variables are excluded, the price index can be constructed by employing Ordinary Least Squares (OLS), thus obtaining an estimate on the coefficients b_i.

5. Modeling methodology

Fig. 3 presents the steps of the modeling methodology adopted in this work and is divided into two distinct phases: First comes the analysis of exploratory data and the conversion of characteristics, followed by the adaptation of the regression model to the data, based on the characteristics that emerged from the first phase. A more detailed description of the methodology is given in the following sections.

5.1. Data collection and preparation

Many CaaS providers are currently active in the cloud market. The current data set was derived from six major cloud providers: Google [47], Amazon [35], Microsoft [48], IBM [49], Alibaba Cloud [50], Digital Ocean [51]. In the data collection phase, functional attributes are naturally mapped to numerical values, whereas non-functional ones are represented as categorical variables. Table 2 summarizes the value set for each attribute of the model. The final data set consists of 640 price bundles and is publicly available on the GitHub repository.[1] Fig. 4 illustrates the number of bundles on a provider basis, within the data set.

According to Fig. 3 data preparation includes the process of gathering, combining, structuring and organizing data so it can be analyzed as part of data visualization, analytics and machine learning applications [52]. This entails the detection of any structural errors (strange naming conventions, typos, or incorrect capitalization) and the removal of unwanted observations from the data set consisting of duplicate or irrelevant observations, aiming at conforming data to a standard pattern.

[1] https://raw.githubusercontent.com/gfragi/py_regression/main/cn_pricing_per_provider_upd3.csv.

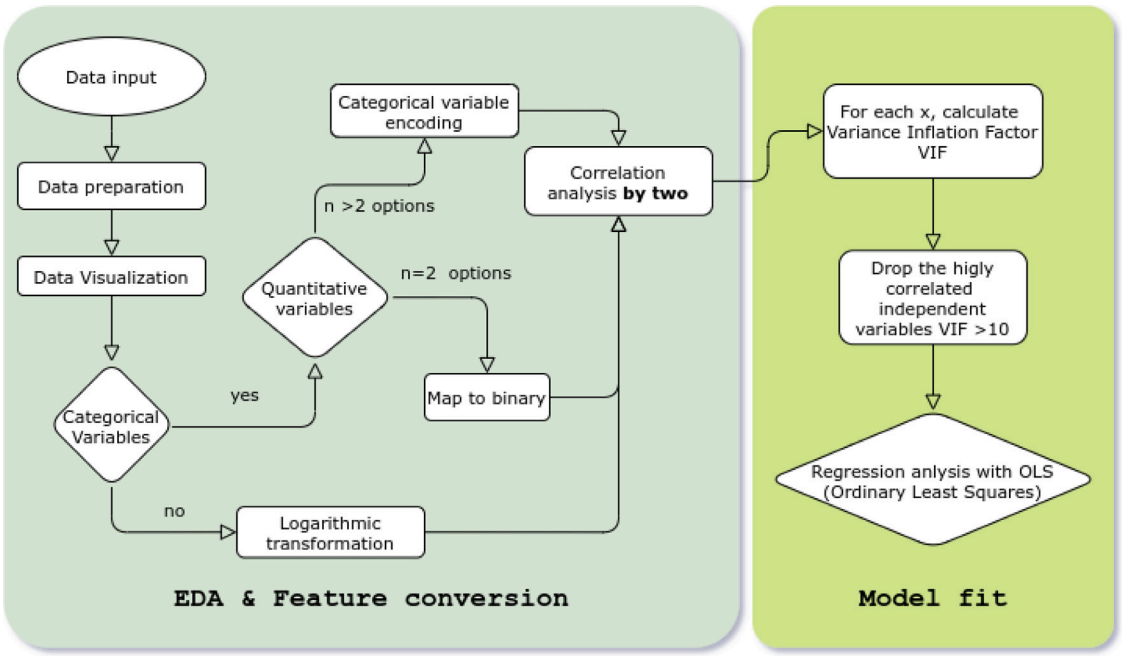

Fig. 3. Proposed price prediction model process.

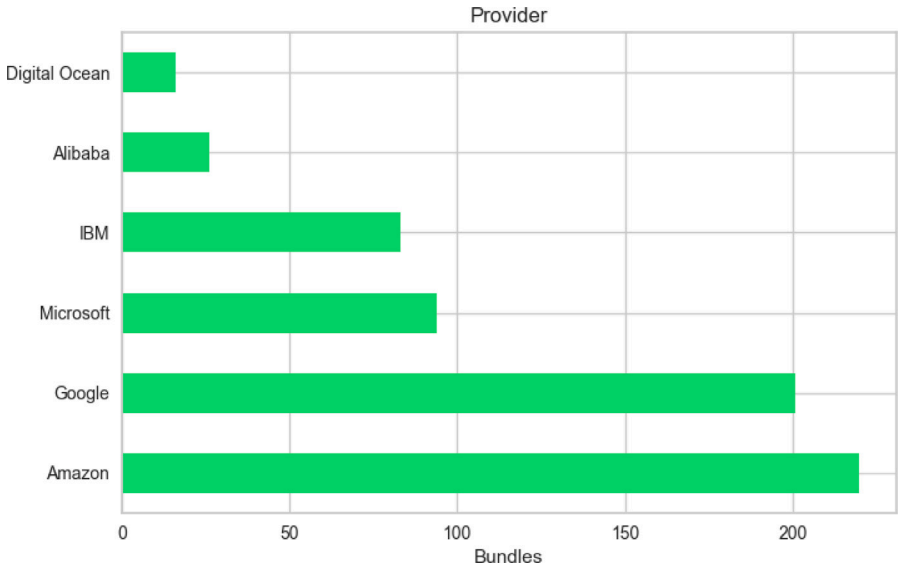

Fig. 4. Number of bundles per provider in our data set.

5.2. Exploratory data analysis

Exploratory data analysis is necessary in order to investigate the quality of the data and reveal any hidden correlations between the variables [53]. After searching for zero values in the data set, the display of the data distribution for each variable will reveal the existence of extreme values and determine whether data transformation is required to smooth out the distribution, such as a logarithmic conversion, a variable encoding, etc. Data inspection by visualization provides a first indication of the importance of each input variable, if it exists. An example of such linear correlation between dependent (Price) and independent (CPU) variable is shown in Fig. 5 where we have plotted how price is varied when the number of cluster cores increases. It is evident that the number

Table 2
Functional and non-functional features of cloud container orchestration clusters.

Feature	Type	No. values	Values
CPU (Cores)	N	7	2, 4, 8, 12, 16, 20, 32
RAM (GB)	N	8	4, 8, 16, 32, 48, 64, 128, 160
Storage (GB)	N	8	100, 200, 300, 400, 500, 800, 1 TB, 2 TB
Internal Egress Traffic (GB)	N	10	30, 100, 400, 500, 600, 1 TB, 2 TB, 5 TB, 9 TB, 10 TB
External Egress Traffic (GB)	N	12	300, 400, 600, 1 TB, 3 TB, 4 TB, 10 TB, 12 TB, 20 TB, 30 TB, 50 TB
Disk type	C	2	Standard, SSD
Operating system	C	3	Free OS, Linux licensed, Windows licensed
Instance type	C	3	On-Demand, Spot, Dedicated
Cluster Management Fee	C	2	Yes/No
Regional redundancy	C	2	Yes/No
Region	C	4	Europe, US, South America, Asia
Auto-scaling	C	2	Horizontal, horizontal–vertical
Hybrid & multi-cloud support	C	2	Yes/No
Vendor agnostic	C	2	Yes/No
Hybrid & multi-cloud support	C	2	Yes/No
Pay per container usage	C	2	Yes/No
Term Length Commitment	C	3	No, 1 year, 3 years
Payment Option Upfront	C	3	No, Partially, All

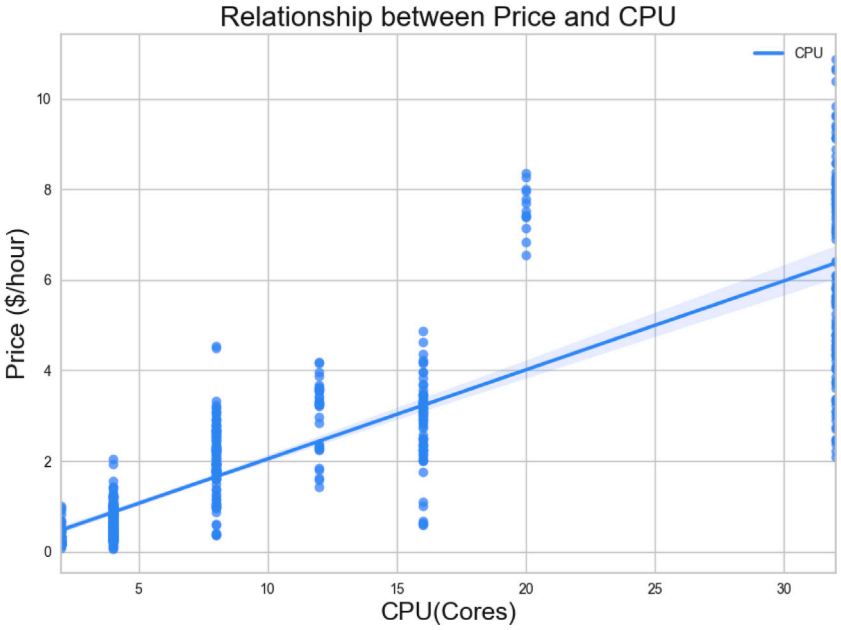

Fig. 5. Linear relationship between Price and CPU.

of cores significantly affect the price and hence we expect that this will be reflected in the corresponding regression coefficient b_k in (2).

5.3. Correlation analysis

Correlation analysis seeks to estimate the correlation between possible pairs of variables, as already mentioned in Section 4.2. The elements of the correlation matrix $\mathbf{C} = [C_{pq}]$ is calculated based on (3). The correlation matrix can be visualized using a heatmap where the values of C_{pq} are mapped into appropriate colors. Since $C_{pq} = C_{qp}$, the correlation matrix is Hermitian and therefore we only need to visualize the elements in the lower triangular part ($q > p$). Besides the input variables X_p it is useful to extend the correlation matrix to include the correlations between the dependent variable, i.e. the price P and X_p. This will provide an additional indication regarding the correlation of the price and the architectural attributes. Depending on the values of the elements C_{pq}, a

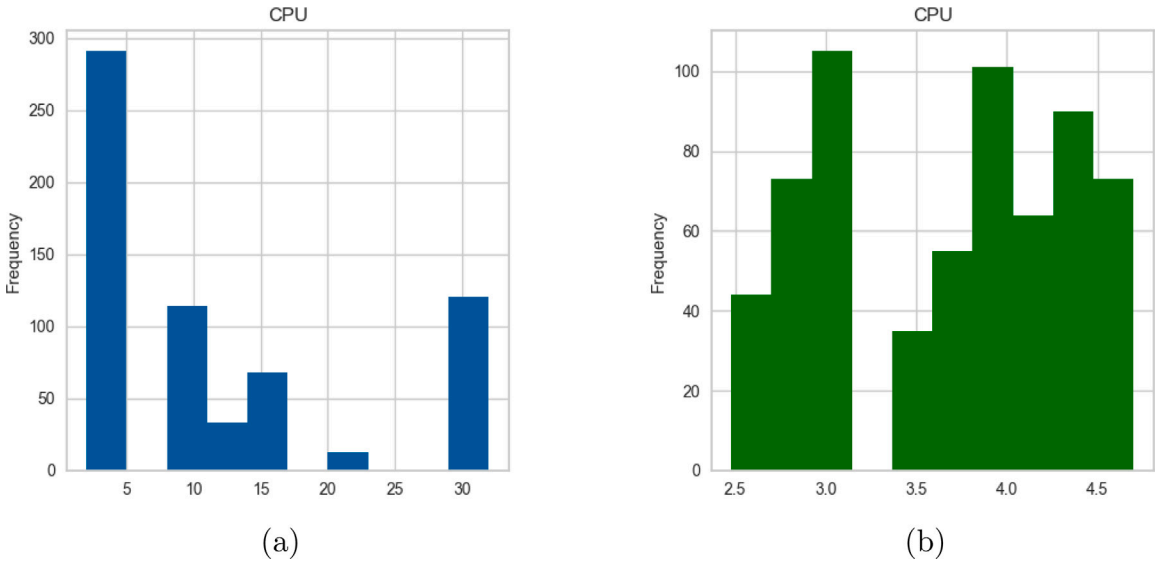

Fig. 6. Distribution of collected data regarding the number of cores assuming (a) no transformation and (b) logarithmic transformation.

Region
US
US
Asia
Europe
Europe

(a)

Region_Asia	Region_Europe	Region_South America	Region_US
0	0	0	1
0	0	0	1
1	0	0	0
0	1	0	0
0	1	0	0

(b)

Fig. 7. (a) Representation of variable *Region* with 4 options and (b) after the apply of one hot coding method.

high degree of correlation between input variables is calculated, therefore the estimation of the value of VIF is needed. Excluding variables with high VIF, significantly improves the accuracy regression model.

5.4. Feature conversion

For categorical variables it is necessary to adopt some sort of arithmetic representation, allowing the estimation of C_{pq} in this case as well. Table 2 summarizes the type of each feature and the number of available options. For numerical values, which vary several orders of magnitude, it is advantageous to use a logarithmic transformation [54]. The numerical variables displayed in Table 2 were therefore converted to a logarithmic scale before incorporated into the regression model. Fig. 6 show the distribution of the values for the number of cores with and without logarithmic conversion, where the first plot (a) represents the distribution of the collected data and the second (b) the converted one. The reformed data is closer to a normal distribution that will consequently lead to the smoothness of the results.

For categorical variable having exactly 2 options the process uses a single binary representation method. In the case of more options, the method of one hot coding is used which compares each level of the categorical variable to a fixed reference level. It transforms a single variable with n options and d distinct values, to d binary variables with n options each. Each observation indicating the presence (1) or absence (0) of the dichotomous binary [55]. As an example the variable of *Region* has 4 different values (US, Europe, S.America, Asia). This categorical variable replaced by 4 other numerical variables (Region_US, Region_Europe, Region_S.America, Region_Asia), where their values are of (0) and (1), as shown in the Fig. 7. At the end of this process the initial column (7a) replaced by the new 4 columns (7b) and attached to the initial data frame.

Table 3
Variance inflation factor calculation.

Feature	VIF
Pay per container	1.48317
Storage	1.51963
Disk type	1.55140
Cluster mgmt fee	1.81215
Vendor agnostic	2.06828
Multicloud support	2.16828
RAM	5.44444
External traffic	6.29810
CPU	9.68523
all other	> 10

Table 4
Statistical metrics results.

Statistical metric	Result
Coefficient of determination R^2	0.901
F-statistic	642.8
Mean Square Error (MSE)	0.008

5.5. Regression analysis

The second phase of the methodology shown in Fig. 3, involves regression analysis. The data set obtained from the correlation analysis is applied to the regression algorithm, leading to the results described in the following Section 6. Our `Python` implementation is publicly available under an open-source license[2] and it is using linear approach for modeling, as the various data involved in the algorithm are represented ordered in categories.

Our implementation uses publicly available open-source `Python` modules such as `numpy`, `sklearn`, `matplotlib`, `statsmodels`, `pandas` and others.

6. Results

6.1. Correlation matrix and VIF

The elements of the correlation matrix discussed in Section 5.3 is shown in Fig. A.10. The heatmap presented in this Figure is useful for identifying dependencies between pairs of input variables. In case of strong dependency, the contribution to the model of the specific variables occurs through one of the two independent variables. Typical example include dependencies between *Operating System Linux* and *Multi-cloud Support* where the correlation is equal to 0.59. These variables are expected to be highly correlated from a technical point of view because the CaaS solutions contain offerings which support hybrid/multi-cloud environments in commercial Linux orchestration platforms.

The results, of the correlation calculations between the factors presented in Fig. A.10, confirm the necessity of calculating the VIF for the various characteristics. This leads our methodology to the next step, calculating the variance inflation factor of each factor, shown in Table 3. The table includes attributes with a value of VIF smaller than 10 in accordance with Table 1. As seen, 9 out of the 17 features have values of VIF < 10 and are therefore retained in the regression process, reducing the dimension of the input variable space.

6.2. Regression coefficients—model performance

Table 5 and Fig. 9 summarize the values of b_i obtained by the regression algorithm. The analysis of the variation of the regression error is depicted in the scatterplot of Fig. 8. The fact that the residues are scattered near the horizontal axis is first evidence that the model of (5) is suitable to describe the data. This is also supported by the histogram of values, which reveals that the residues are nearly normally distributed around zero, indicating a well-adjusted model.

Table 4 shows the statistical metrics of the regression, highlighting the rigor of the proposed approach.

It is useful to comment on the values of the regression coefficients b_i shown in Table 5 and Fig. 9.

CPU is the most important parameter, contributing most to the price strategy, followed by the external traffic and cluster fee. The cluster fee factor clearly shows that customers could have cost savings, if the architecture incorporates one large instead of

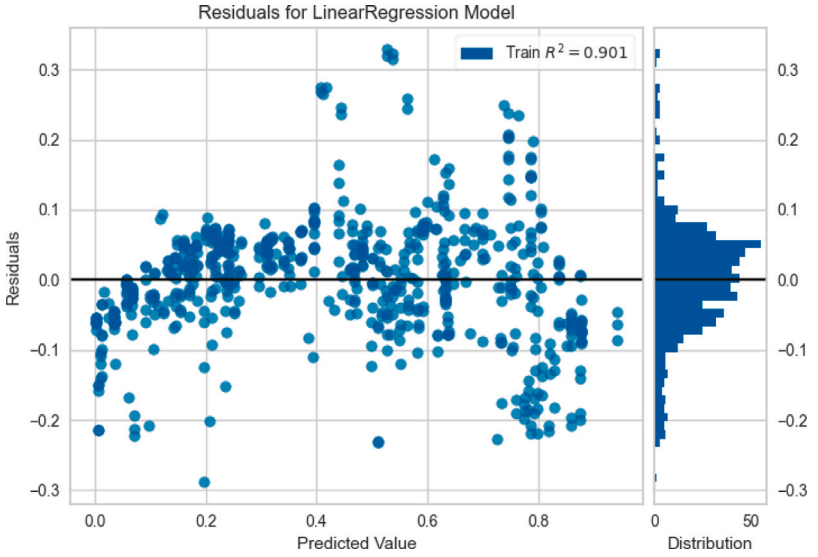

Fig. 8. Residuals for linear regression model.

Table 5
Regression results.

i		b_i	std err	p-value
0	constant	−0.8198	0.035	0.000
1	External traffic	0.1792	0.012	0.000
2	CPU	0.3582	0.027	0.000
3	RAM	0.0670	0.020	0.001
4	Storage	0.0251	0.010	0.015
5	Cluster mgmt fee	0.0854	0.010	0.000
6	Disk type	0.0188	0.008	0.018
7	Multicloud support	0.0215	0.008	0.005
8	Pay per container	0.0350	0.012	0.005
9	Vendor agnostic	0.0449	0.018	0.011

many small clusters. It is also clear that all functional attributes (CPU, RAM, external egress traffic and storage) of the model highly affect the strategy of CaaS pricing.

The pricing model derived in this work is therefore based on (2) along with the coefficients of this table, we end up capturing our final model

$$\begin{aligned} P = &-0.82 + 0.18 * ExternalTraffic + 0.32 * CPU + 0.067 * RAM + 0.025 \\ &* Storage + 0.085 * ClusterFee + 0.02 * Disktype + 0.02 * Support \\ &+ 0.035 * PayPerContainer + 0.045 * Vendor\ Agnostic + 0.008 \end{aligned} \tag{5}$$

Within containerized ecosystems, users have the option to specify requests and limits for the CPU and the RAM, which constitute the minimum amount of resources a container needs to run. Engineers tend to overestimate the required resources, in order to ensure the application performance and the high level user experience. This overestimation naturally leads to underutilized containers and clusters. According to a Data Dog report [56], half of the containers in production systems use less than 30% of the requested CPU and memory capacity. Since clusters are billed based on provisioning and not on actual use, this gap between reserved computing resources and actual utilization potentially leads for customers to pay more than they actually need. Moreover and according to the results, since CPU and RAM contribute substantially to the shaping of the final price, it becomes evident that utilization constitutes the most important financial aspect of a container cluster.

Analysis of the results also shows that the external egress traffic is the factor that has the largest impact on price, after CPU. Egress is the traffic going from the cluster nodes out to the internet. Its effect on price can be expected, since the provider's cost for network infrastructure components or bandwidth usage from transit service providers to carry traffic to the customer's end is an

[2] https://github.com/gfragi/py_regression.git.

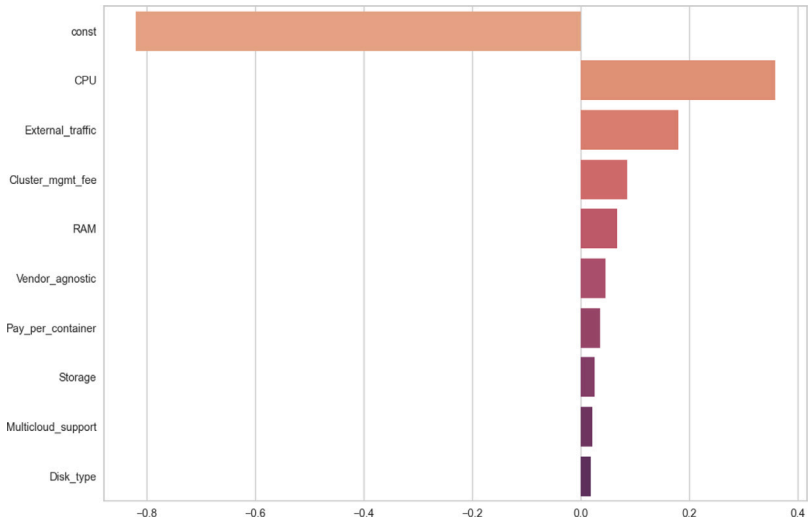

Fig. 9. Coefficients in descending order.

expensive task. From the customer's point of view, the egress fees are one of the hidden costs, as it is hard to predict and control them.

Since the "pay-per-container" attribute is also one of the factors contributing most to the final price, it is interesting to further analyze it. As explained in Section 3, "pay-per container" indicates if the customer is charged based on container usage capacity or on the underlying virtual machine capacity. It is a parameter that seems to sit in between containers and serverless services, indicating that providers strive to provide a serverless experience, thus allowing customers to pay only for the containers they use. From a customer's point of view, this pricing model could be more cost efficient than paying for node capacity. But this is another case where the customer pays for what is allocated and not for what it is used. Thus, cost optimization is not relevant without understanding the technical implications.

Furthermore, vendor-agnostic and hybrid support characteristics indicate the fast-growing multi-cloud trend of the market. Customers seek to use multiple cloud platforms, obtaining the optimal solution in terms of performance and cost, while avoiding vendor lock-ins. Flexera's cloud report showed that 92% of enterprises have a multi-cloud or hybrid cloud strategy [57]. Cloud providers on the other hand, evolve to meet customers demands for hybrid and multi-cloud support. They also need to respond to the rise of container-based edge infrastructure solutions, like VMware Tanzu solution [58] and release hybrid container platform solutions managed from the cloud. Moreover, the storage and disk type parameters seem to affect the pricing less. Although the cloud storage pricing is falling over the years [59], the big amount of stored data usually put together the need for data aggregation. Cloud vendors can benefit for this demand providing as-a-service platforms for data analytic.

7. Conclusions

CaaS is a business model that facilitates software developers in organizing, running, managing and deploying containers, by using container-based virtualization. CaaS sits in between the IaaS and PaaS cloud computing models, taking advantage of container technology. All the leading IaaS providers e.g. Google, AWS and Microsoft Azure, offer CaaS solutions built on top of their IaaS platforms with the aim to increase their profitability and market share. Provider pricing policies is therefore of paramount importance. The work developed in this paper formulates a hedonic price index to model CaaS pricing, based on data collected from popular providers.

The CaaS bundles used for the evaluation of the proposed methodology were collected out of six providers and they are described by 17 attributes, functional and non-functional. The application of a hedonic method led to the construction of hedonic price index based on 9 attributes, after excluding the highly correlated ones. Based on the final price index, CPU is the parameter contributing more to the shaping of the price, indicating the importance of efficient utilization. It is followed by external egress traffic, revealing the network infrastructure cost paid by the providers. The cluster management fee has also a significant impact on price showing the cost advantage when the architecture incorporates a large cluster instead of many small clusters. In addition, RAM has a significant influence on price and it is evident that the functional attributes (CPU, RAM, external egress traffic) highly affect the shaping of the of CaaS pricing scheme. The "pay-per-container" aspect has also a noteworthy impact, highlighting the fact that providers promote server-less architecture, enabling customers to pay only for the containers they utilize. Moreover, the vendor-agnostic and the hybrid cloud support features which are next in terms of importance, indicate the fast-growing multi-cloud trend of the market. Storage and disk type seem to have a secondary contribution to the CaaS price.

The results of the proposed method can be a powerful tool for providers and users. Providers can examine and assess the pricing policy of the competitors and gain a competitive advantage in the cloud market, whereas users can comprehend the complicated pricing policies of the providers. Further extensions and future research directions include the adoption a nonlinear or even non-functional regression analysis, which would probably highlight different aspects of the pricing policy. In addition, it would be interesting study the update of the model based on yearly data sets to analyze the gradual evolution of pricing strategy and the involved parameters. Such an approach could actually allow to predict future CaaS prices. It would also be interesting, to construct hedonic price index for each major cloud provider and compare their pricing policies. Last but not least, it would be interesting to study the pricing for hybrid CaaS solutions between public cloud vendors and enterprise vendors.

Appendix. Correlation matrix — heatmap

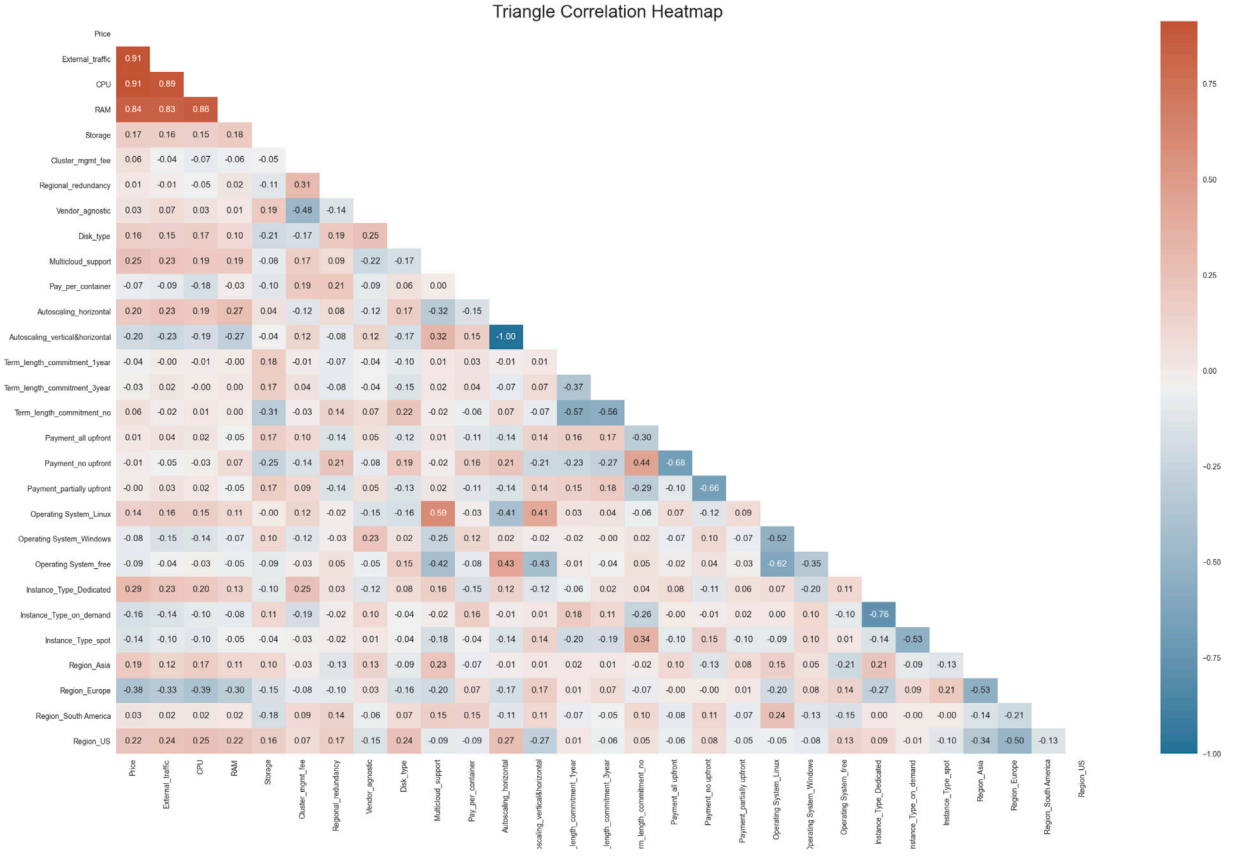

Fig. A.10. Triangle heatmap — correlation matrix.

References

[1] C. Pahl, A. Brogi, J. Soldani, P. Jamshidi, Cloud container technologies: a state-of-the-art review, IEEE Trans. Cloud Comput. 7 (3) (2017) 677–692.
[2] X. Xie, T. Yuan, X. Zhou, X. Cheng, Research on trust model in container-based cloud service, Comput. Mater. Contin. 56 (2) (2018) 273–283.
[3] I. Mavridis, H. Karatza, Combining containers and virtual machines to enhance isolation and extend functionality on cloud computing, Future Gener. Comput. Syst. 94 (2019) 674–696.
[4] M.K. Hussein, M.H. Mousa, M.A. Alqarni, A placement architecture for a container as a service (CaaS) in a cloud environment, J. Cloud Comput. 8 (1) (2019) 1–15.
[5] D. Bernstein, Containers and cloud: From lxc to docker to kubernetes, IEEE Cloud Comput. 1 (3) (2014) 81–84.
[6] M. Warrilow, 2021, URL https://www.gartner.com/en/newsroom/press-releases/2020-06-25-gartner-forecasts-strong-revenue-growth-for-global-co. (Accessed 28 August 2021).
[7] J. Triplett, Handbook on Hedonic Indexes and Quality Adjustments in Price Indexes: Special Application to Information Technology Products, OECD, 2004.
[8] H. Xu, B. Li, Dynamic cloud pricing for revenue maximization, IEEE Trans. Cloud Comput. 1 (2) (2013) 158–171.
[9] C. Wu, R. Buyya, K. Ramamohanarao, Value-based cloud price modeling for segmented business to business market, Future Gener. Comput. Syst. 101 (2019) 502–523.
[10] Y. Lu, X. Zheng, L. Li, L.D. Xu, Pricing the cloud: a QoS-based auction approach, Enterp. Inf. Syst. 14 (3) (2020) 334–351.
[11] H. Jahandideh, J.W. Drew, F. Balestrieri, K. McCardle, Individualized pricing for a cloud provider hosting interactive applications, Serv. Sci. 12 (4) (2020) 130–147.

[12] E. Filiopoulou, G. Dede, C. Michalakelis, T. Kamalakis, End-user and provider requirements for pricing policies of infrastructure as a service, IEEE Internet Comput. 23 (5) (2019) 36–48.
[13] A.C. Goodman, Hedonic prices, price indices and housing markets, J. Urban Econ. 5 (4) (1978) 471–484.
[14] R. Michaels, Hedonic prices and the structure of the digital computer industry, J. Ind. Econ. (1979) 263–275.
[15] R. Cole, Y. Chen, J.A. Barquin-Stolleman, E. Dulberger, N. Helvacian, J.H. Hodge, Quality-adjusted price indexes for computer processors and selected peripheral equipment, Surv. Cur. Bus. 66 (1) (1986) 41–50.
[16] R.A. Nelson, T.L. Tanguay, C.D. Patterson, A quality-adjusted price index for personal computers, J. Bus. Econom. Statist. 12 (1) (1994) 23–31.
[17] A. Pakes, A reconsideration of hedonic price indexes with an application to PC's, Amer. Econ. Rev. 93 (5) (2003) 1578–1596.
[18] E.R. Berndt, N.J. Rappaport, Hedonics for personal computers: A reexamination of selected econometric issues, in: R&D, Education and Productivity", an International Conference in Memory of Zvi Griliches (1930-1999), August, Citeseer, 2003, pp. 25–27.
[19] A. Aizcorbe, D.M. Byrne, D.E. Sichel, Getting smart about phones: new price indexes and the allocation of spending between devices and services plans in personal consumption expenditures, in: Measuring Economic Growth and Productivity, Elsevier, 2020, pp. 387–411.
[20] N. Watanabe, R. Nakajima, T. Ida, Quality-adjusted prices of Japanese mobile phone handsets and carriers' strategies, Rev. Ind. Organ. 36 (4) (2010) 391–412.
[21] D.M. Byrne, C.A. Corrado, Prices for communications equipment: Rewriting the record, FEDS, 2015. Working Paper.
[22] D. Varoutas, K. Deligiorgi, C. Michalakelis, T. Sphicopoulos, A hedonic approach to estimate price evolution of telecommunication services: evidence from Greece, Appl. Econ. Lett. 15 (14) (2008) 1131–1134.
[23] S. El Kihal, C. Schlereth, B. Skiera, Price comparison for infrastructure-as-a-service, in: ECIS, 2012, pp. 161.
[24] C. Wu, A.N. Toosi, R. Buyya, K. Ramamohanarao, Hedonic pricing of cloud computing services, IEEE Trans. Cloud Comput. (2018).
[25] D. Byrne, C. Corrado, D.E. Sichel, The rise of cloud computing: minding your P's, Q's and K's, Tech. rep., National Bureau of Economic Research, 2018.
[26] G. Portella, G.N. Rodrigues, E. Nakano, A.C. Melo, Statistical analysis of Amazon EC2 cloud pricing models, Concurr. Comput.: Pract. Exper. 31 (18) (2019) e4451.
[27] C.L.M. Belusso, S. Sawicki, V. Basto-Fernandes, R.Z. Frantz, F. Roos-Frantz, A proposal of infrastructure-as-a-service providers pricing model using linear regression, Rev. Bras. Comput. Apl. 10 (2) (2018) 44–53.
[28] C.L. Belusso, S. Sawicki, V. Basto-Fernandes, R.Z. Frantz, F. Roos-Frantz, Price modeling of iaas providers-an approach focused on enterprise application integration, in: International Conference on Enterp. Inf. Syst., Vol. 2, SCITEPRESS, 2017, pp. 371–376.
[29] P. Mitropoulou, E. Filiopoulou, C. Michalakelis, M. Nikolaidou, Pricing cloud iaaS services based on a hedonic price index, Computing 98 (11) (2016) 1075–1089.
[30] P. Mitropoulou, E. Filiopoulou, M. Nikolaidou, C. Michalakelis, Pricing IaaS: A hedonic price index approach, in: International Conference on the Economics of Grids, Clouds, Systems, and Services, Springer, 2017, pp. 18–28.
[31] J. Watada, A. Roy, R. Kadikar, H. Pham, B. Xu, Emerging trends, techniques and open issues of containerization: a review, IEEE Access 7 (2019) 152443–152472.
[32] Amazon, 2021, URL https://kubernetes.io/. (Accessed 14 July 2021).
[33] Amazon, 2021, URL https://www.redhat.com/en/resources/. (Accessed 14 July 2021).
[34] A. Poniszewska-Marańda, E. Czechowska, Kubernetes cluster for automating software production environment, Sensors 21 (5) (2021) 1910.
[35] Amazon, URL https://aws.amazon.com/ecs/.
[36] amazon, URL https://aws.amazon.com/savingsplans/.
[37] Google, URL https://cloud.google.com/kubernetes-engine/pricings.
[38] G. Sayfan, Mastering Kubernetes, Packt Publishing Ltd, 2017.
[39] D. Midigudla, Performance analysis of the impact of vertical scaling on application containerized with Docker: Kubernetes on Amazon web services-EC2, 2019.
[40] S. Herath, G. Maier, The hedonic price method in real estate and housing market research: a review of the literature, 2010.
[41] Z. Griliches, Hedonic price indexes for automobiles: An econometric of quality change, in: The Price Statistics of the Federal Goverment, NBER, 1961, pp. 173–196.
[42] S. Rosen, Hedonic prices and implicit markets: product differentiation in pure competition, J. Polit. Econ. 82 (1) (1974) 34–55.
[43] M.H. Licht, Multiple regression and correlation, American Psychological Association, 1995.
[44] R. Taylor, Interpretation of the correlation coefficient: A basic review, J. Diagn. Med. Sonogr. 6 (1) (1990) 35–39, http://dx.doi.org/10.1177/875647939000600106.
[45] N. Shrestha, Detecting multicollinearity in regression analysis, Am. J. Appl. Math. Stat. 8 (2) (2020) 39–42.
[46] F.-J. Lin, Solving multicollinearity in the process of fitting regression model using the nested estimate procedure, Qual. Quant. 42 (3) (2008) 417–426.
[47] Google, URL https://cloud.google.com/kubernetes-engine.
[48] Microsoft, URL https://azure.microsoft.com/de-de/services/.
[49] IBM, URL https://www.ibm.com/cloud/kubernetes-service.
[50] Alibaba, URL https://www.alibabacloud.com/product/.
[51] DigitalOcean, URL https://www.digitalocean.com/products/kubernetes/.
[52] S.B. Analytics, URL https://searchbusinessanalytics.techtarget.com/definition/data-preparation.
[53] OmniSci, 2021, URL https://www.omnisci.com/technical-glossary/feature-engineering. (Accessed 14 July 2021).
[54] R.H. McCuen, R.B. Leahy, P.A. Johnson, Problems with logarithmic transformations in regression, J. Hydraul. Eng. 116 (3) (1990) 414–428.
[55] K. Potdar, T.S. Pardawala, C.D. Pai, A comparative study of categorical variable encoding techniques for neural network classifiers, Int. J. Comput. Appl. 175 (2017) 7–9.
[56] Datadoghq, URL https://www.datadoghq.com/container-report/.
[57] Flexera, URL https://www.flexera.com/about-us/press-center/flexera-releases-2021-state-of-the-cloud-report.html.
[58] VMware, URL https://tanzu.vmware.com/tanzu.
[59] Amazon, URL https://aws.amazon.com/blogs/aws/aws-update-new-m3-features-reduced-ebs-prices-reduced-s3-prices/.