Gabriel Costa de Lucca

Construção de uma Base de Dados Multi-Cloud e Aplicação de Aprendizado de Máquina para Análise Comparativa de Recursos de Computação em Nuvem

São Carlos – SP
2026

Gabriel Costa de Lucca

Construção de uma Base de Dados Multi-Cloud e Aplicação de Aprendizado de Máquina para Análise Comparativa de Recursos de Computação em Nuvem

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação da Universidade Federal de São Carlos como requisito parcial para obtenção do título de Engenheiro de Computação, sob orientação do Prof. Dr. Hélio Crestana Guardia.

Universidade Federal de São Carlos – UFSCar
Centro de Ciências Exatas e de Tecnologia
Departamento de Computação
Curso de Engenharia de Computação

Orientador: Prof. Dr. Hélio Crestana Guardia

São Carlos – SP
2026

Dedico este trabalho aos meus pais, por todo o amor, apoio e incentivo ao longo da minha vida. Por cada esforço silencioso, por cada renúncia feita para que eu pudesse ter oportunidades que eles muitas vezes não tiveram e por acreditarem em mim mesmo nos momentos em que eu duvidei. Esta conquista também pertence a vocês.

À minha avó Tereza, que não teve a oportunidade de estudar e trabalhou ao longo da vida enfrentando uma realidade marcada pela escassez de oportunidades, mas que sempre foi um exemplo de força e resiliência.

À minha avó Neide (in memoriam), por todo o carinho, amor e pelas lembranças afetivas que permanecem comigo.

À minha namorada, pelo apoio constante nos momentos difíceis, pela compreensão, carinho e por ser também meu porto seguro ao longo desta jornada.

Agradecimentos

Agradeço profundamente aos meus pais por todo o amor, apoio incondicional e incentivo ao longo de toda a minha vida. Em especial, por todo o suporte financeiro dedicado aos meus estudos, muitas vezes abrindo mão de seus próprios sonhos e necessidades para que eu pudesse seguir o meu caminho. Tudo o que conquistei até aqui carrega o esforço, a dedicação e o amor de ambos, que sempre foram meu maior exemplo e minha base em todos os momentos.

Agradeço à minha namorada por todo o carinho, paciência e apoio nos momentos difíceis, por sempre estar ao meu lado e por tornar essa jornada mais leve e significativa ao longo destes últimos seis anos. Sua presença foi essencial em cada etapa desta caminhada.

Agradeço aos meus amigos por todos os momentos de alegria, descontração, estudo e companheirismo ao longo da graduação. Cada risada, cada desafio e cada conquista compartilhada tornaram essa trajetória mais leve, especial e inesquecível.

Agradeço aos professores, técnicos e servidores da Universidade Federal de São Carlos (UFSCar) por toda a dedicação à minha formação acadêmica e profissional, por compartilharem conhecimento com generosidade e por contribuírem de forma significativa para minha trajetória.

Em especial, agradeço ao professor Hélio pela orientação, confiança e parceria ao longo do desenvolvimento deste trabalho, sempre com atenção e apoio fundamentais para a conclusão desta etapa.

"A simplicidade é o último grau de sofisticação."
— Leonardo da Vinci

Resumo

A comparação de custos entre máquinas virtuais em ambientes multi-cloud é uma tarefa complexa devido à heterogeneidade dos catálogos, das nomenclaturas, dos modelos de precificação e das APIs disponibilizadas pelos provedores. Essa fragmentação dificulta a identificação de instâncias tecnicamente equivalentes e a avaliação objetiva do custo-benefício entre ofertas de Amazon Web Service, Microsoft Azure e Google Cloud Platform, por exemplo. Este trabalho apresenta uma arquitetura para coleta, normalização e análise de preços de instâncias de máquinas virtuais, combinando engenharia de dados e aprendizado de máquina para apoiar decisões de comparação e seleção de recursos em nuvem.

A solução desenvolvida contempla um pipeline ETL multi-cloud capaz de extrair dados públicos de precificação e especificações técnicas, transformar estruturas heterogêneas em um esquema comum e consolidá-las em uma base relacional unificada. A base final reuniu 34 mil registros de instâncias *On-Demand* e *Spot*, sendo aproximadamente metade dessas instâncias utilizadas nos experimentos de modelagem. Sobre esse subconjunto foram avaliadas duas abordagens complementares. A primeira utiliza técnicas de clusterização para identificar perfis técnicos semelhantes entre instâncias. A segunda emprega modelos de regressão para estimar o preço por hora das máquinas virtuais. Na etapa de clusterização foram comparados os algoritmos K-Means e Gaussian Mixture Model, sendo o K-Means selecionado por apresentar melhor qualidade de agrupamento. Na etapa de regressão foram avaliados os modelos de Regressão Linear, Random Forest e XGBoost, com melhor desempenho marginal para o XGBoost.

A partir das estimativas produzidas pelo modelo de regressão selecionado foi definido o indicador de custo-benefício, que relaciona o preço observado de uma instância ao preço previsto para sua configuração. Esse indicador permite avaliar se uma instância se encontra potencialmente abaixo, próxima ou acima do padrão de preço aprendido a partir do catálogo, complementando análises baseadas apenas no preço absoluto. A integração entre clusterização, regressão e o custo-benefício foi demonstrada em cenários conceituais voltados à análise de aderência de preços e à busca de alternativas equivalentes entre provedores.

A padronização dos catálogos, associada à clusterização de perfis técnicos e à modelagem preditiva de preços, mostrou-se capaz de apoiar análises comparativas em ambientes multi-cloud.

Palavras-chave: computação em nuvem; multi-cloud; ETL; máquinas virtuais; aprendizado de máquina; clusterização; regressão; FinOps.

Abstract

The comparison of virtual machine costs across multi-cloud environments is challenging due to differences in service catalogs, naming conventions, pricing models, and provider-specific APIs. Such heterogeneity makes it difficult to identify technically equivalent instances and to objectively evaluate cost-effectiveness among offerings from Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). This work proposes an architecture for collecting, normalizing, and analyzing virtual machine pricing data, combining data engineering and machine learning techniques to support cloud resource comparison and selection.

A multi-cloud ETL pipeline was developed to extract public pricing and technical specification data, transform heterogeneous catalog structures into a common schema, and consolidate the information into a unified relational database. The resulting dataset contains 34 thousand On-Demand and Spot virtual machine records collected from the three providers. Based on this dataset, clustering and regression models were evaluated to support comparative analyses. Clustering techniques were employed to identify groups of instances with similar technical characteristics, while regression models were used to estimate hourly prices from instance specifications. Among the evaluated approaches, K-Means achieved the best clustering quality and XGBoost obtained the best predictive performance for price estimation.

The selected regression model was further used to derive a cost-effectiveness indicator based on the relationship between observed and predicted prices. This indicator enables the identification of instances whose pricing deviates from the expected market pattern learned from the consolidated catalog, complementing analyses based solely on absolute cost values. The integration of data normalization, clustering, regression, and cost-effectiveness assessment was demonstrated through comparative multi-cloud scenarios involving price evaluation and identification of equivalent alternatives across providers.

The results indicate that the proposed approach can support comparative analyses in multi-cloud environments by providing a standardized view of virtual machine catalogs and enabling data-driven evaluation of cloud resource pricing.

Keywords: cloud computing; multi-cloud; ETL; virtual machines; machine learning; clustering; regression; FinOps.

Sumário

1 INTRODUÇÃO 12
1.1 Contexto e motivação 12
1.2 Objetivos 13
1.2.1 Objetivos específicos 14
1.3 Organização do trabalho 14

2 FUNDAMENTAÇÃO TEÓRICA 15
2.1 Computação em nuvem 15
2.2 Modelos de serviço e precificação em nuvem 15
2.3 FinOps e padrão FOCUS 17
2.4 ETL (Extração, Transformação e Carregamento dos dados) 18
2.5 Aprendizado de máquina 18
2.5.1 K-Means 19
2.5.2 Gaussian Mixture Model 19
2.5.3 Regressão linear com regularização Ridge 20
2.5.4 Random Forest 20
2.5.5 Gradient Boosting e XGBoost 21
2.6 Trabalhos Relacionados 21

3 METODOLOGIA 23
3.1 Visão Geral 23
3.2 Construção da base de dados multi-provedores 24
3.2.1 Normalização dos dados e mapeamento para o padrão FOCUS 25
3.2.2 Extrator AWS [Bulk Pricing API] 26
3.2.3 Extrator Azure [Retail Prices API] 27
3.2.4 Extrator GCP [Cloud Billing Catalog API] 29
3.2.5 Tratamento e limpeza dos dados 31
3.2.6 Seleção das instâncias para análise 32
3.3 Clusterização de instâncias utilizando K-Means e GMM 32
3.3.1 Construção da matriz de atributos 33
3.3.2 Pré-processamento 33
3.3.3 Treinamento dos modelos 34
3.3.4 Avaliação e seleção de K 34
3.4 Regressão de preços utilizando Regressão Linear e Modelos baseados em Árvores 35
3.4.1 Base de dados e variáveis 35

3.4.2 Treinamento dos modelos 36
3.4.3 Avaliação e seleção do regressor 37
3.4.4 Indicador de custo-benefício 37
3.5 Cenários de aplicação 38
3.5.1 Cenário 1: análise de aderência de preço 38
3.5.2 Cenário 2: busca exclusiva por alternativas equivalentes 38

4 EXPERIMENTOS E RESULTADOS 40
4.1 Estrutura do conjunto de dados obtido 40
4.2 Análise exploratória do dataset obtido 41
4.2.1 Distribuição por provedor e modalidade de preço 42
4.2.2 Distribuição de preços 42
4.2.3 Recursos computacionais *On-Demand* e heterogeneidade entre provedores . 44
4.2.4 Matriz de correlação 45
4.3 Experimentos com Modelos de Agrupamento 46
4.3.1 Modelo K-Means 46
4.3.2 Modelo de Mistura de Gaussianas (GMM) 47
4.3.3 Análise do Agrupamento do Modelo Escolhido 48
4.4 Experimentos com Modelos de Regressão 51
4.4.1 Comparação entre Regressão Linear, Random Forest e XGBoost 51
4.4.2 Análise do erro por faixa de preço 52
4.4.3 Seleção do modelo final 52
4.4.4 Análise do modelo selecionado 53
4.5 Aplicação da solução em cenários representativos 55
4.5.1 Cenário 1: auditoria de preço no mesmo perfil técnico 55
4.5.2 Cenário 2: busca por alternativas equivalentes 56

5 CONCLUSÕES E TRABALHOS FUTUROS 58

REFERÊNCIAS 59

Lista de ilustrações

Figura 1 – Pipeline ETL, banco de dados unificado e aplicação dos algoritmos. . . 23
Figura 2 – Arquitetura do extrator AWS. 26
Figura 3 – Fluxo de extração e filtros aplicados no extrator AWS 26
Figura 4 – Arquitetura do extrator Azure. 27
Figura 5 – Fluxo de extração e filtros aplicados no extrator Azure 28
Figura 6 – Arquitetura do extrator GCP. 29
Figura 7 – Fluxo de coleta e processamento dos SKUs do GCP. 30
Figura 8 – Fluxo conceitual do Cenário 1: análise de aderência de preço 38
Figura 9 – Fluxo conceitual do Cenário 2: busca por alternativas equivalentes . . . 39
Figura 10 – Histograma de VMs On-Demand e seus respectivos preços por hora . . 43
Figura 11 – Histograma de VMs Spot e seus respectivos preços por hora 43
Figura 12 – Distribuição de preço por hora, vCPUs e memória RAM por provedor. 44
Figura 13 – Matriz de correlação de Pearson entre as variáveis numéricas do catálogo consolidado. 45
Figura 14 – Inércia e Método do cotovelo 47
Figura 15 – Distribuição de instâncias por cluster 49
Figura 16 – XGBoost para preços reais versus previstos no conjunto de teste (escala logarítmica) 54
Figura 17 – Dez variáveis de maior importância na predição de preço para XGBoost 55

Lista de tabelas

Tabela 1 – Comparação dos modelos de precificação de VMs por provedor 16
Tabela 2 – Correspondência conceitual entre dimensões do FOCUS e o esquema construído 25
Tabela 3 – Principais campos retornados pela Bulk Pricing API da AWS 27
Tabela 4 – Razão RAM/vCPU estimada por família de instâncias Azure 28
Tabela 5 – Principais campos retornados pela Retail Prices API do Azure 29
Tabela 6 – Principais campos utilizados da Cloud Billing Catalog API 31
Tabela 7 – Pipeline de normalização: registros removidos por etapa 31
Tabela 8 – Variáveis utilizadas na clusterização On-Demand 33
Tabela 9 – Variáveis utilizadas na regressão de preços On-Demand 36
Tabela 10 – Esquema completo da tabela `cloud_instances`. 40
Tabela 11 – Distribuição do dataset por provedor e tipo de preço 42
Tabela 12 – Métricas internas do K-Means para diferentes valores de K 46
Tabela 13 – Métricas internas do GMM para diferentes valores de K 48
Tabela 14 – Perfil técnico dos dez grupos 50
Tabela 15 – Comparação de regressores no conjunto de teste 51
Tabela 16 – Distribuição do erro absoluto do XGBoost por faixa de preço 52
Tabela 17 – Desempenho do XGBoost por conjunto (USD/h) 53
Tabela 18 – Cenário 1: auditoria de preço e alternativas no *cluster* 2 56
Tabela 19 – Cenário 2: alternativas 8 vCPU / 64 GB / Linux no *cluster* 6 56

Lista de abreviaturas e siglas

API	*Application Programming Interface*
AWS	*Amazon Web Services*
ETL	*Extract, Transform, Load*
FinOps	*Cloud Financial Operations*
FOCUS	*FinOps Open Cost and Usage Specification*
GMM	*Gaussian Mixture Model*
GPU	*Graphics Processing Unit*
IaaS	*Infrastructure as a Service*
JSON	*JavaScript Object Notation*
MAE	*Mean Absolute Error*
ML	*Machine Learning*
OData	*Open Data Protocol*
PaaS	*Platform as a Service*
RAM	Random Access Memory
RF	*Random Forest*
RMSE	*Root Mean Squared Error*
SaaS	*Software as a Service*
SKU	*Stock Keeping Unit*
SO	Sistema Operacional
vCPU	*Virtual Central Processing Unit*
VM	Virtual Machine
VR	*Value Ratio*
XGBoost	*eXtreme Gradient Boosting*

1 Introdução

1.1 Contexto e motivação

A Computação em Nuvem (*Cloud Computing*) se consolidou como um paradigma predominante nas infraestruturas de Tecnologia da Informação nos últimos anos. Esse paradigma permite alugar cargas computacionais sob demanda ao invés de investir em servidores físicos, transformando significativamente a forma como empresas e organizações planejam, implantam e operam suas infraestruturas tecnológicas. De acordo com o relatório da Fortunate Business Insights (Fortune Business Insights, 2024), os gastos globais com serviços de nuvem atingiram 780 bilhões de dólares em 2025 e estão projetados para atingir 900 bilhões de dólares em 2026, sendo esse crescimento derivado de aplicações que envolvem transformações digitais, modelos de inteligência artificial e investimentos contínuos de capital em hiperescalabilidade.

No universo de Computação em Nuvem três provedores concentram aproximadamente 70% do mercado de infraestrutura. Amazon Web Services (AWS) com 30%, Microsoft Azure com 26% e Google Cloud Platform (GCP) com 14% do mercado (CRN News, 2026).Embora esses três provedores dominem o mercado, existem diferenças significativas entre seus serviços. Essas diferenças abrangem a quantidade e tipos de máquinas disponibilizadas em seus catálogos, modelos de precificação, nomenclatura dos produtos e os mecanismos disponibilizados para consulta de especificações e preços por meio de APIs.

Essa diversidade de ofertas torna o processo de seleção de recursos computacionais uma tarefa complexa. Cada provedor disponibiliza centenas ou até milhares de tipos de instâncias de máquinas virtuais, caracterizadas por diferentes combinações de capacidade de processamento, memória RAM, sistemas operacionais suportados, regiões geográficas de implantação e modelos de precificação. Como consequência, empresas, organizações e usuários individuais que desejam comparar especificações técnicas e custos entre múltiplos provedores enfrentam um espaço de busca amplo e heterogêneo. A análise manual dessas alternativas torna-se trabalhosa e suscetível a erros, dificultando a identificação das instâncias que melhor atendem simultaneamente aos requisitos de desempenho, disponibilidade e orçamento.

A literatura reconhece a heterogeneidade entre provedores como um dos principais desafios em ambientes multi-cloud. Segundo Elhabbash (ELHABBASH et al., 2018), diferenças em modelos de serviço, mecanismos de precificação, APIs e formatos de representação dificultam a comparação sistemática entre recursos de nuvem e motivam o desenvolvimento de soluções para apoiar em tomadas de decisões. De forma complementar, Di Martino (MARTINO, 2014) destaca que a ausência de padrões comuns entre provedores compromete

a interoperabilidade e a portabilidade de aplicações, uma vez que cada plataforma adota nomenclaturas, estruturas de dados e mecanismos próprios de acesso aos seus serviços.

Diante desse contexto, torna-se necessário desenvolver mecanismos capazes de consolidar informações provenientes de múltiplos provedores e apoiar a comparação sistemática entre instâncias com características semelhantes e relações de custo-benefício comparáveis. A utilização de técnicas de integração de dados e aprendizado de máquina permite automatizar parte desse processo, possibilitando a identificação de padrões e relações que dificilmente seriam observados por meio de análises manuais.

Com o objetivo de contribuir para esse cenário, este projeto busca desenvolver uma solução inicial composta por 3 camadas. A primeira camada consiste em um *pipeline* de extração e tratamento de dados, responsável por capturar dados das APIs públicas de catálogo de máquinas e seus preços da AWS, Azure e GCP, transformar as diferentes estruturas de catálogo para um formato comum e armazenar os dados em um banco de dados relacional unificado e padronizado.

A segunda camada utiliza técnicas de aprendizado de máquina para analisar as instâncias coletadas sob duas perspectivas complementares. A primeira busca identificar grupos de instâncias com características técnicas semelhantes, independentemente do provedor ao qual pertencem. Para essa tarefa, foram avaliados dois algoritmos de agrupamento não supervisionado, sendo eles K-Means e Gaussian Mixture Model. A segunda perspectiva concentra-se na modelagem dos preços das instâncias com o objetivo de identificar padrões de precificação. Nesse contexto, foram comparados diferentes algoritmos de regressão, incluindo Regressão Linear, Random Forest e XGBoost.

Por fim, a terceira camada combina os resultados produzidos pelos modelos de agrupamento e regressão para apoiar análises comparativas entre provedores. Os agrupamentos permitem identificar instâncias com características técnicas semelhantes e potencialmente adequadas para cenários de uso equivalentes. Em paralelo, os modelos de regressão possibilitam estimar preços esperados a partir das características técnicas das instâncias, permitindo avaliar se os valores praticados estão alinhados aos padrões observados no conjunto de dados. Em conjunto, essas informações fornecem subsídios para a comparação de alternativas entre provedores e para a análise de custo-benefício em diferentes cenários de utilização.

1.2 Objetivos

Objetivo geral: o objetivo deste trabalho é extrair, tratar e compilar dados de precificação e especificações de VMs dos três principais provedores de computação em nuvem, visando à criação de uma base de dados unificada, acessível e padronizada. A partir dessa base, busca-se investigar a aplicação de técnicas de aprendizado de máquina para identificar instâncias tecnicamente semelhantes e modelar padrões de precificação em ambientes

multi-cloud.

1.2.1 Objetivos específicos

1. Projetar e implementar um pipeline de ETL para extrair dados de precificação das APIs públicas da AWS, Azure e GCP, normalizar as estruturas heterogêneas de cada catálogo e consolidá-los em um banco de dados relacional padronizado e unificado.

2. Avaliar e selecionar um modelo de clusterização capaz de agrupar instâncias com especificações técnicas semelhantes, independentemente do provedor, viabilizando a identificação de alternativas equivalentes entre plataformas.

3. Avaliar e selecionar um modelo de regressão para estimar o preço esperado de instâncias a partir de seus atributos técnicos e contextuais, e calcular uma razão de valor entre o preço real e o preço estimado, como indicador de custo-benefício relativo.

4. Avaliar os modelos produzidos sobre o catálogo coletado e analisar os resultados em cenários de uso representativos da aplicação proposta.

1.3 Organização do trabalho

O restante deste documento está organizado da seguinte forma: o Capítulo 2 apresenta a fundamentação teórica sobre computação em nuvem, processos ETL, os algoritmos de aprendizado de máquina utilizados e métricas de avaliação. O Capítulo 3 detalha a Metodologia, descrevendo a arquitetura do sistema, cada um dos três extratores de dados, o *pipeline* de normalização, o esquema do banco de dados, os modelos de ML e seus desdobramentos. O Capítulo 4 apresenta os experimentos realizados e discute os resultados obtidos. O Capítulo 5 encerra com as conclusões, limitações e propostas de trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresenta os conceitos teóricos que fundamentam o sistema desenvolvido. A Seção 2.1 introduz computação em nuvem e seus modelos de precificação. A Seção 2.4 discute processos ETL. A Seção 2.5 descreve os algoritmos de aprendizado de máquina utilizados. A Seção 2.6 posiciona o trabalho em relação à literatura.

2.1 Computação em nuvem

A computação em nuvem é definida pelo NIST (*National Institute of Standarts and Technology*) como um tipo de modelo úbiquo, conveniente e sob demanda a um conjunto de recursos computacionais compartilhados que podem ser configuráveis e rapidamente provisionados e liberados com mínimo esforço de gerenciamento ou interação com o provedor de serviço (MELL; GRANCE, 2011).

Segundo o modelo NIST, existem algumas premissas essenciais para a definição de computação em nuvem que devem ser claras tanto para o usuário, quanto para o provedor:

1. Acesso à rede de internet: os recursos ficam disponíveis por meio da rede e são acessados de maneira padronizada em diferentes tipos de dispositivos eletrônicos.

2. Ter autoatendimento sob demanda: o usuário final (cliente independente, empresa ou organização) pode provisionar os recursos que deseja (VMs) a qualquer momento e sem a necessidade de interação humana.

3. Serviço com métricas de monitoramento: o uso dos recursos é monitorado e controlado pelo provedor de maneira transparente, sendo o modelo de cobrança baseado no consumo efetivamente realizado.

4. Agrupamento de recursos: os recursos computacionais são organizados em conjunto, compartilhados fisicamente e virtualmente e atribuídos de maneira dinâmica a depender do consumo.

Essas características possibilitam que provedores de nuvem disponibilizem recursos computacionais de forma escalável e sob demanda, estabelecendo a base para os modelos atuais de comercialização de infraestrutura.

2.2 Modelos de serviço e precificação em nuvem

Os recursos computacionais em nuvem podem ser disponibilizados por meio de diferentes modelos de serviço (MELL; GRANCE, 2011). Entre eles, destaca-se o modelo IaaS (*Infras-*

tructure as a Service), no qual o provedor oferece recursos fundamentais de computação, como máquinas virtuais, redes e armazenamento. Nesse cenário, cabe ao usuário administrar o sistema operacional, configurar o ambiente e gerenciar as aplicações que deseja executar sobre a infraestrutura provisionada. No modelo PaaS (*Platform as a Service*), o provedor disponibiliza uma plataforma completa para desenvolvimento, implantação e execução de aplicações. Dessa forma, o usuário concentra seus esforços na construção e manutenção do software, sem a necessidade de gerenciar diretamente os recursos de infraestrutura subjacentes. Por fim, o modelo SaaS (*Software as a Service*), as aplicações são fornecidas prontas para utilização por meio da internet. Nesse caso, as atividades relacionadas à infraestrutura, à plataforma e à manutenção do sistema permanecem sob responsabilidade do dono da aplicação, cabendo ao usuário apenas utilizar o serviço disponibilizado.

Este trabalho tem como foco o modelo IaaS, visto que os dados analisados correspondem a recursos de infraestrutura virtualizada, cujas características computacionais e preços constituem a base para as comparações e análises realizadas.

No modelo IaaS, a precificação de máquinas virtuais varia drasticamente conforme o compromisso financeiro e a flexibilidade do usuário. A Tabela 1 sintetiza como os três principais provedores organizam os modelos de cobrança.

Tabela 1 – Comparação dos modelos de precificação de VMs por provedor

Modelo	**AWS**	**Azure**	**GCP**
Sob demanda	Por segundo, mínimo de 60s	Por minuto	Por segundo, mínimo de 60s
Reserva	1 ou 3 anos; pagamento adiantado, parcial ou mensal	1 ou 3 anos; desconto aplicado automaticamente	1 ou 3 anos; desconto aplicado automaticamente com base em compromisso de uso
Spot	Capacidade ociosa; revogação com aviso de 2 minutos	Capacidade ociosa; revogação com aviso de 30 segundos	Capacidade ociosa; revogação com aviso de 30 segundos

Fonte: (Amazon Web Services, 2024), (Microsoft Azure, 2024) e (Google Cloud, 2024).

Instâncias *Spot* podem apresentar preços significativamente inferiores aos das respectivas instâncias *On-Demand*, com descontos que em determinados cenários ultrapassam 70%. Entretanto, a comparação direta entre esses dois modelos de contratação pode introduzir distorções nas análises de preço. Como o menor preço das instâncias *Spot* está associado à possibilidade de interrupção pelo provedor a qualquer momento, sua utilização é adequada apenas para cargas de trabalho tolerantes a falhas. Dessa forma, um sistema de recomendação baseado exclusivamente em preço tenderia a privilegiar esse tipo de instância, desconsiderando restrições de disponibilidade e confiabilidade que as tornam inadequadas para aplicações críticas e outros serviços que exigem execução contínua.

2.3 FinOps e padrão FOCUS

O modelo IaaS transformou a forma como recursos computacionais são consumidos, eliminando a necessidade de aquisição e manutenção de infraestrutura física própria. Em contrapartida, a cobrança baseada no consumo introduziu novos desafios relacionados ao controle de custos, à previsibilidade de orçamento e ao dimensionamento adequado dos recursos. Na computação em nuvem, decisões inadequadas de provisionamento podem resultar tanto em desperdício financeiro quanto em degradação do desempenho das aplicações. Visando otimizar esse cenário, surgiu o conceito de FinOps, resultado da combinação dos termos *Finance* e *Operations*, com o objetivo de promover uma gestão mais eficiente dos gastos em nuvem.

De acordo com a FinOps Foundation (FinOps Foundation, 2024), o termo não se limita a um conjunto de ferramentas voltadas apenas à redução de custos. Trata-se de uma prática organizacional que busca alinhar equipes técnicas, financeiras e de negócios em torno de uma gestão orientada por dados e baseada na maximização do valor obtido a partir dos recursos de nuvem. Entre seus princípios fundamentais, destacam-se:

1. Promover visibilidade contínua sobre os custos e o consumo de recursos.
2. Ajustar o provisionamento para evitar desperdícios decorrentes de superprovisionamento ou subprovisionamento.
3. Automatizar processos e integrar métricas financeiras ao ciclo de desenvolvimento e operação dos sistemas.

Um dos principais obstáculos para a adoção dessas práticas em ambientes multi-cloud está na falta de padronização de dados entre os provedores. Cada plataforma adota convenções próprias para nomenclatura de serviços, métricas de uso, estruturas de faturamento e modelos de precificação. Como consequência, consolidar informações provenientes de múltiplos provedores em um único banco de dados exige um processo manual de interpretação e transformação dos dados.

Com o objetivo de mitigar esse problema, a FinOps Foundation propôs o padrão FOCUS (*FinOps Open Cost and Usage Specification*). Trata-se de uma iniciativa *open-source* que define um vocabulário comum e um modelo padronizado para representação de informações de custo, uso e faturamento em ambientes de nuvem. Embora cada provedor mantenha seus formatos internos de dados, FOCUS tem por objetivo estabelecer uma camada de abstração que permite consolidar essas informações em uma estrutura uniforme, facilitando análises financeiras e operacionais em cenários multi-cloud.

Por meio da definição de esquemas padronizados e terminologias compartilhadas, a iniciativa visa reduzir inconsistências entre provedores e facilitar análises comparativas, geração de relatórios e processos de governança financeira. Dessa forma, o FOCUS constitui

uma importante referência para trabalhos que envolvem integração e normalização de dados provenientes de diferentes plataformas de computação em nuvem.

2.4 ETL (Extração, Transformação e Carregamento dos dados)

O processo de ETL é um paradigma fundamental em engenharia de dados. O acrônimo descreve três fases distintas:

1. **Extração (*Extract*)**: coleta de dados a partir de fontes heterogêneas que podem incluir bancos de dados, APIs, arquivos, streams de eventos ou páginas web. O desafio principal é lidar com a diversidade de formatos, protocolos de acesso e modelos de dados das fontes.

2. **Transformação (*Transform*)**: processo de limpeza, validação, normalização, enriquecimento e reestruturação dos dados extraídos para que se conformem a um esquema comum. Inclui atividades como remoção de duplicatas, tratamento de valores ausentes, conversão de unidades, inferência de campos faltantes e cálculo de métricas derivadas.

3. **Carga (*Load*)**: inserção dos dados transformados em um sistema de armazenamento de destino, como um banco de dados relacional, onde os dados ficam disponíveis para consulta.

Os processos de ETL são amplamente utilizados em cenários que envolvem a integração de dados provenientes de múltiplas fontes heterogêneas. Nesses ambientes, as diferenças de formato, nomenclatura, granularidade e qualidade dos dados dificultam sua utilização direta, sem tratamentos. A etapa de transformação desempenha um papel central nesse contexto, pois permite converter informações originalmente incompatíveis em uma estrutura comum e consistente.

2.5 Aprendizado de máquina

Esta seção apresenta os fundamentos dos algoritmos de aprendizado de máquina utilizados neste trabalho. O estudo abrange duas tarefas. A primeira envolve o agrupamento de instâncias de máquinas virtuais com características técnicas semelhantes, e a segunda, a modelagem do preço horário dessas instâncias. Para cada tarefa, foram avaliados diferentes algoritmos, cuja seleção final e respectivos resultados são discutidos nos capítulos de Metodologia e Experimentos e Resultados.

2.5.1 K-Means

O K-Means (MACQUEEN, 1967) é um algoritmo de clusterização que particiona n observações em K grupos de modo que cada ponto pertença ao agrupamento cujo centroide é o mais próximo. O objetivo é minimizar a inércia, definida como a soma das distâncias quadráticas intra-cluster:

$$J = \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2, \tag{2.1}$$

em que S_i é o conjunto de pontos do cluster i e μ_i é o centroide correspondente.

O algoritmo alterna dois passos até convergir. O primeiro passo consiste na atribuição onde cada ponto é associado ao centroide mais próximo utilizando a distância euclidiana. O segundo consiste na atualização dos centróides onde cada um é recalculado como a média aritmética dos pontos atribuídos ao cluster. Cada iteração reduz ou mantém J e o processo termina quando os centroides deixam de se deslocar ou quando se atinge um limite de iterações. A partição é rígida e cada instância pertence a exatamente um grupo.

O número de agrupamentos K pode ser calculado usando diferentes métricas, entre elas:

- **Silhouette** (ROUSSEEUW, 1987) ($s \in [-1, 1]$): mede coesão intra-cluster e separação inter-cluster, sendo que valores maiores indicam grupos mais definidos.
- **Davies-Bouldin** (DAVIES; BOULDIN, 1979): relaciona dispersão intra-cluster à distância entre centroides, sendo que valores menores indicam melhor separação entre grupos.

2.5.2 Gaussian Mixture Model

O Gaussian Mixture Model (GMM) (HASTIE; TIBSHIRANI; FRIEDMAN, 2009) modela os dados como uma combinação de K distribuições gaussianas multivariadas. A densidade de uma observação x é:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k), \tag{2.2}$$

em que π_k é o peso da componente k ($\sum_k \pi_k = 1$), μ_k é o vetor de médias e Σ_k é a matriz de covariância. Cada componente representa um *cluster*. A probabilidade de pertencimento de x à componente k é proporcional a $\pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k)$. Esse mecanismo produz atribuição parcial, em contraste com a partição fixa do K-Means.

Os parâmetros π_k, μ_k, Σ_k são estimados pelo algoritmo de Expectação-Maximização (EM). Na etapa de expectativa (*Expectation*), calculam-se as probabilidades posteriores de cada observação pertencer a cada componente da mistura. Em seguida, na etapa de maximização (*Maximization*), atualizam-se os parâmetros das distribuições gaussianas

de forma a maximizar a verossimilhança dos dados observados. Esse processo é repetido iterativamente até que seja atingido um critério de convergência.

Diferentemente de métodos de agrupamento rígido, como o K-Means, o GMM realiza uma atribuição probabilística das observações aos grupos, permitindo que um mesmo ponto apresente diferentes graus de pertencimento a múltiplos clusters. Essa característica torna o modelo particularmente adequado para cenários em que existem sobreposições entre grupos ou fronteiras pouco definidas entre as observações.

2.5.3 Regressão linear com regularização Ridge

A regressão linear modela a variável resposta y (preço por hora) como combinação linear das *features* $x_1, \ldots, x_p$. Os coeficientes β são estimados minimizando o erro quadrático entre valores observados e preditos. Na versão com regularização Ridge (HASTIE; TIBSHIRANI; FRIEDMAN, 2009), adiciona-se penalização L_2 aos coeficientes:

$$\min_{\beta} \ \|y - X\beta\|_2^2 + \alpha\|\beta\|_2^2, \tag{2.3}$$

em que X é a matriz de atributos, β é o vetor de coeficientes e $\alpha > 0$ controla a intensidade da regularização. Coeficientes de atributos correlacionados são encolhidos em direção a zero, o que reduz variância do estimador e mitiga instabilidade quando há multicolinearidade, situação comum em catálogos com variáveis categóricas codificadas.

Ridge funciona como linha inicial de comparação entre modelos de regressão pois captura relações aproximadamente lineares entre atributos, mas tem capacidade limitada para interações não lineares.

2.5.4 Random Forest

O Random Forest (BREIMAN, 2001) é um método de *ensemble* que combina múltiplas árvores de decisão. Para cada árvore $b = 1, \ldots, B$:

1. amostra-se com reposição um subconjunto *bootstrap* do conjunto de treino;
2. em cada divisão de nó, considera-se apenas um subconjunto aleatório de atributos para escolher a melhor divisão.

A predição em regressão é a média das saídas das árvores:

$$\hat{y}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x), \tag{2.4}$$

em que $T_b(x)$ é a predição da árvore b. A agregação reduz a variância em relação a uma única árvore profunda e permite capturar interações entre variáveis sem transformações manuais dos atributos.

2.5.5 Gradient Boosting e XGBoost

O Gradient Boosting (HASTIE; TIBSHIRANI; FRIEDMAN, 2009) constrói o modelo preditivo de forma iterativa. Inicializa-se com uma predição constante $F_0(x)$ e a cada etapa $m = 1, \ldots, M$, ajusta-se uma árvore fraca $h_m(x)$ aos resíduos do modelo acumulado. O modelo evolui como:

$$F_m(x) = F_{m-1}(x) + \eta \cdot h_m(x), \tag{2.5}$$

em que $\eta \in (0, 1]$ é a taxa de aprendizado. Cada árvore corrige erros remanescentes do *ensemble* anterior e o método aproxima a minimização da função de perda seguindo o gradiente negativo em relação à predição corrente.

O XGBoost (*eXtreme Gradient Boosting*) (CHEN; GUESTRIN, 2016) é uma implementação otimizada desse princípio. Além do ajuste sequencial de árvores, incorpora regularização nos termos folha e penalização de complexidade estrutural, o que restringe o crescimento excessivo de árvores individuais.

A qualidade dos regressores é avaliada por métricas complementares:

- R^2 (coeficiente de determinação): proporção da variância de y explicada pelo modelo;
- MAE (*Mean Absolute Error*): média das diferenças absolutas entre valores reais e preditos;
- RMSE (*Root Mean Squared Error*): raiz da média dos erros quadráticos, penalizando mais desvios grandes.

2.6 Trabalhos Relacionados

Diversos trabalhos acadêmicos e ferramentas comerciais abordam a comparação de serviços de nuvem, cada um com escopo e limitações distintos.

O CloudCmp (LI et al., 2010) foi um dos primeiros trabalhos acadêmicos a propor uma metodologia sistemática para comparação de provedores de nuvem. Os autores avaliaram métricas de desempenho, como latência, *throughput* e disponibilidade, por meio de experimentos distribuídos realizados em diferentes regiões e provedores. De forma semelhante, Ostermann (OSTERMANN et al., 2010) analisaram o desempenho de instâncias da Amazon EC2 em diferentes cenários de execução. Embora esses estudos forneçam informações relevantes sobre o comportamento dos provedores, ambos concentram-se em métricas de desempenho observadas experimentalmente, sem abordar a comparação sistemática de preços e características dos catálogos de instâncias.

Com a crescente adoção de estratégias multi-cloud, diversos trabalhos passaram a investigar mecanismos para seleção e comparação de serviços entre provedores. Elhabbash (ELHABBASH et al., 2018) apresentam uma revisão sobre sistemas de *cloud brokerage*,

destacando que a heterogeneidade de APIs, modelos de precificação e descrições de serviços dificulta a comparação direta entre recursos computacionais. De forma complementar, Di Martino (MARTINO, 2014) discute desafios de interoperabilidade e portabilidade em ambientes de nuvem, mostrando que a ausência de padrões comuns entre provedores dificulta tanto a migração de aplicações quanto a comparação de serviços equivalentes.

Sob a perspectiva econômica, LI (LI et al., 2016) apresentam uma taxonomia dos modelos de precificação utilizados por provedores de nuvem, evidenciando diferenças significativas entre estratégias de cobrança e contratos de utilização. Os autores identificam a comparação de custos entre provedores como um problema complexo devido à heterogeneidade dos modelos de precificação e à dificuldade de estabelecer equivalências entre recursos. Na mesma direção, Goodarzy (GOODARZY et al., 2020) apresentam uma revisão sobre aplicações de aprendizado de máquina para gerenciamento de recursos em nuvem. Embora o estudo demonstre o potencial dessas técnicas para otimização operacional e redução de custos, os autores observam que a maior parte dos trabalhos concentra-se na gestão interna dos recursos de um único provedor, sem abordar a comparação sistemática entre catálogos multi-cloud.

Entre as soluções disponíveis comercialmente, o Infracost (Infracost, 2026) permite estimar custos de infraestrutura a partir de configurações descritas em Terraform, auxiliando equipes DevOps durante o planejamento de implantações. Entretanto, a ferramenta limita-se à estimativa de custos e não realiza recomendações ou análises de equivalência entre instâncias. Outra solução amplamente utilizada é o Cloudorado (Cloudorado, 2026), que disponibiliza mecanismos de comparação de instâncias entre provedores por meio de filtros manuais de especificação. Apesar de facilitar consultas pontuais, a ferramenta não emprega modelos de aprendizado de máquina nem mecanismos automatizados para identificação de alternativas equivalentes ou avaliação de custo-benefício.

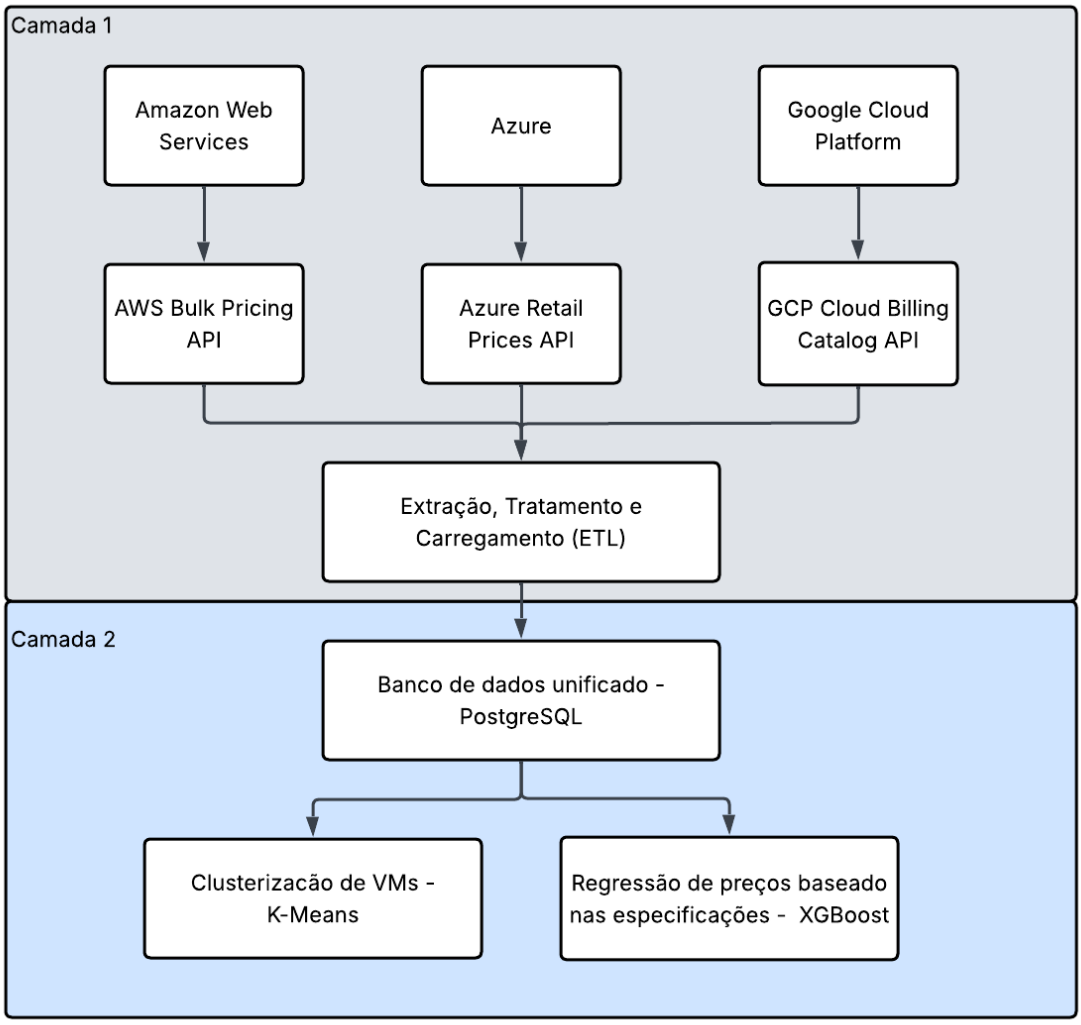
Embora os trabalhos analisados abordem aspectos relevantes da computação em nuvem, como desempenho, interoperabilidade e precificação, ainda existe uma lacuna em soluções que integrem dados de múltiplos provedores e utilizem aprendizado de máquina para apoiar análises comparativas. Este trabalho busca contribuir nesse contexto por meio da construção de uma base unificada de instâncias de nuvem e da aplicação de técnicas de agrupamento e regressão para comparação entre provedores.

3 Metodologia

3.1 Visão Geral

O trabalho segue uma arquitetura com duas camadas de desenvolvimento principais conforme exemplificado na Figura 1. A primeira camada envolveu todo o processo de extração dos dados de cada um dos provedores, tratamentos necessários para a normalização e padronização até o carregamento para o banco de dados unificado. A segunda camada envolveu a utilização da base de dados construída, a partir do pipeline ETL, para o treinamento dos modelos de aprendizado de máquina desenvolvidos.

Figura 1 – Pipeline ETL, banco de dados unificado e aplicação dos algoritmos.



Fonte: Elaborado pelo autor.

A partir desse desenvolvimento, foram definidos e explorados dois cenários de uso para

demonstrar como os tópicos desenvolvidos podem ser aplicados em uma análise comparativa de instâncias em ambientes multi-cloud. Esses cenários permitiram avaliar, de forma conceitual, como os resultados da clusterização e da regressão podem ser combinados para apoiar a interpretação de preços e a identificação de instâncias tecnicamente semelhantes entre provedores.

3.2 Construção da base de dados multi-provedores

A base de dados foi construída por meio de um *pipeline* ETL implementado em Python para coletar, normalizar e consolidar catálogos de máquinas virtuais da AWS, Azure e GCP em um esquema relacional único. O pipeline foi executado separadamente para cada provedor, mantendo a mesma estrutura geral de processamento, mas com regras específicas para interpretar os diferentes formatos de catálogo.

Na etapa de extração, cada módulo consultou a API pública de precificação do respectivo provedor e coletou registros brutos de instâncias, regiões, sistemas operacionais, modalidades de preço e valores publicados. Na AWS, as especificações técnicas foram obtidas diretamente dos atributos do catálogo. Na Azure, parte das informações de vCPUs e memória foi inferida a partir da nomenclatura dos SKUs. No GCP, o preço final da instância foi reconstruído a partir dos componentes de cobrança de vCPU e RAM.

Na etapa de transformação, os registros heterogêneos foram convertidos para um conjunto comum de campos com a estrutura de provedor, tipo de instância, região, sistema operacional, modalidade de preço, vCPUs, memória RAM, quantidade de GPUs e preço por hora em USD. Também foram calculados atributos derivados, como `ram_per_vcpu`, preço mensal estimado, custo por vCPU e custo por GB de RAM. Em seguida, foram removidos registros incompletos, inconsistentes ou duplicados, utilizando uma chave composta por provedor, tipo de instância, região, sistema operacional, especificações técnicas e modalidade de preço.

Na etapa de carregamento, os dados normalizados de cada provedor foram gravados no banco. Cada execução registrou metadados como provedor processado, data de início e término, quantidade de registros coletados, permitindo rastrear atualizações do catálogo e repetir o processo quando os preços públicos são republicados.

A implementação foi organizada em módulos com responsabilidades separadas por extratores específicos por provedor, rotinas comuns de transformação e validação, e uma camada de persistência responsável pela gravação no banco. O código-fonte do pipeline ETL e dos experimentos de aprendizado de máquina foi disponibilizado na plataforma do GitHub em (LUCCA, 2026) .

Nas subseções seguintes, são apresentados os detalhes de cada provedor e as regras utilizadas para conversão dos dados ao esquema proposto.

3.2.1 Normalização dos dados e mapeamento para o padrão FOCUS

A etapa de transformação dos dados exigiu a definição prévia do esquema de dados capaz de representar, de maneira uniforme, as informações provenientes dos catálogos dos provedores citados. Embora os três disponibilizem atributos relacionados à configuração e à precificação de máquinas virtuais, a nomenclatura e o nível de detalhamento variam significativamente entre as plataformas.

Esse modelo contempla características técnicas das instâncias, como quantidade de vCPUs, memória RAM e GPUs, além de informações contextuais, incluindo provedor, região, sistema operacional e modalidade de contratação. Também foram incorporados atributos de custo e indicadores derivados utilizados nas análises posteriores, como citado na seção anterior.

A definição desse esquema foi guiada pelos princípios de padronização propostos pelo FOCUS, que estabelece uma estrutura comum para representação de custos e recursos em ambientes multi-cloud. Embora o trabalho não implemente integralmente o padrão, diversas dimensões do modelo adotado possuem correspondência conceitual com elementos definidos pela especificação.

A Tabela 2 apresenta essa correspondência entre as principais dimensões do FOCUS e os campos utilizados na base unificada. Nas subseções seguintes, são descritas as regras de transformação aplicadas aos dados extraídos de cada provedor para preenchimento desses atributos. A descrição completa do esquema persistido resultante no banco de dados é apresentada no Capítulo 4.

Tabela 2 – Correspondência conceitual entre dimensões do FOCUS e o esquema construído

Dimensão FOCUS	**Campo unificado**	**Observação**
ProviderName	`provider`	Valor fixo por extrator (AWS, Azure, GCP)
RegionId	`region`	Código regional nativo de cada API
ResourceId / SkuId	`instance_type`	Identificador da configuração de VM
PricingQuantity	`vcpus`, `ram_gb`	Capacidade técnica da instância em vez de consumo medido
BilledCost / ListCost	`price_per_hour_usd`	Preço de referência por hora em USD
ChargeCategory	`price_type`	Modalidade de contratação (On-Demand, Spot ou Reserved)

Fonte: Elaborado pelo autor.

3.2.2 Extrator AWS [Bulk Pricing API]

Figura 2 – Arquitetura do extrator AWS.

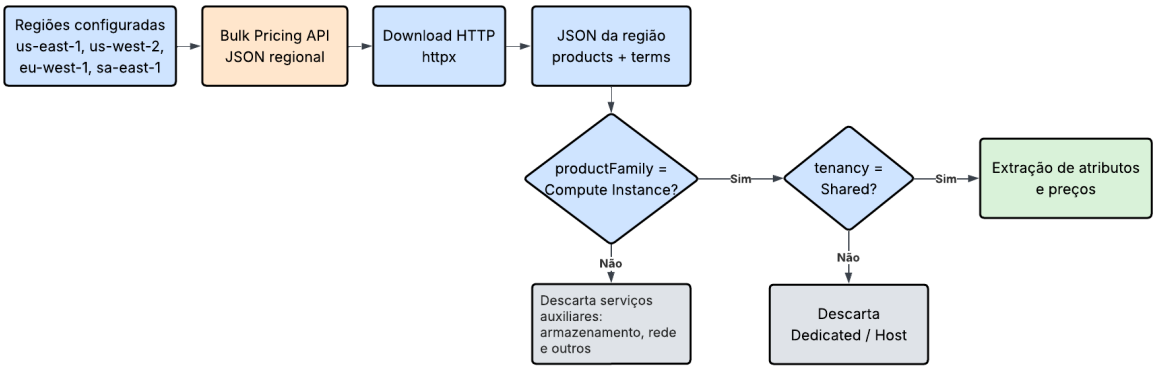
Atributo	Detalhes
URL Base	https://pricing.us-east-1.amazonaws.com/offers/v1.0/aws/AmazonEC2/current/\{region\}/index.json}
Autenticação	Nenhuma. API pública
Formato de resposta	Formato de resposta JSON monolítico por região
Estrutura	products (especificações) e terms (preços por SKU)
Regiões coletadas	us-east-1, us-west-2, eu-west-1, sa-east-1
Documentação oficial	https://aws.amazon.com/ec2/pricing

Fonte: Elaborado pelo autor.

A AWS expõe seus preços por meio da Bulk Pricing API (Amazon Web Services, 2024), que disponibiliza um arquivo JSON completo por região contendo produtos e termos de precificação. Entre os três extratores, este é o mais direto, pois a API já fornece especificações técnicas estruturadas nos atributos de cada produto. Demais atributos presentes no JSON, como `physicalProcessor` e `networkPerformance`, foram utilizados quando disponíveis, mas não aparecem de forma equivalente nas demais APIs.

Para cada região configurada, o extrator realiza o download do JSON regional via HTTP com `httpx`, filtra produtos de computação em *tenancy* compartilhada, converte a memória de `GiB` para valor numérico e extrai o preço sob demanda por hora.

Figura 3 – Fluxo de extração e filtros aplicados no extrator AWS



Fonte: Elaborado pelo autor.

A Tabela 3, representa os principais campos resultantes da extração dos dados desse provedor que posteriormente foram atribuídos ao esquema de dados final.

Tabela 3 – Principais campos retornados pela Bulk Pricing API da AWS

Campo na API	**Conteúdo**	**Uso no pipeline**
`productFamily`	Família do produto	Filtro (`Compute Instance`)
`instanceType`	Nome da instância (ex. `m5.xlarge`)	`instance_type`
`vcpu`	Número de vCPUs	`vcpus`
`memory`	Memória em string (`64 GiB`)	`ram_gb` após *parsing*
`operatingSystem`	Sistema operacional	`os`
`regionCode`	Região	`region`
`instanceFamily`	Família de hardware	`instance_family`
`tenancy`, `capacitystatus`	Tenancy e status de capacidade	Filtros de exclusão
`terms.OnDemand`	Preço por hora em USD	`price_per_hour_usd`
`terms.Reserved`	Termos de reserva	Registros `reserved`

Fonte: Elaborado pelo autor.

3.2.3 Extrator Azure [Retail Prices API]

Figura 4 – Arquitetura do extrator Azure.

Atributo	Detalhes
URL Base	https://prices.azure.com/api/retail/prices
Autenticação	Nenhuma. API pública
Formato de resposta	JSON paginado
Filtros utilizados	serviceFamily = 'Compute' e serviceName = 'Virtual Machines' e priceType = 'Consumption'
Regiões coletadas	eastus, westus2, westeurope, brazilsouth
Documentação oficial	https://azure.microsoft.com/en-us/pricing/details/virtual-machine

Fonte: Elaborado pelo autor.

A Azure oferece a Retail Prices API (Microsoft Azure, 2024), com filtros que permitem selecionar campos específicos a serem extraídos (filtros OData) e paginação automática para extrair dados das máquinas no catálogo e seus preços. O principal desafio de engenharia neste extrator residiu no fato de que a API de preços não informa diretamente vCPUs e RAM. Essas informações precisaram ser inferidas a partir do nome do SKU da máquina virtual analisada.

O campo `armSkuName` identifica a máquina virtual retornada pela API e segue a convenção de nomenclatura da Microsoft (Microsoft, 2026). Como a Retail Prices API não

expõe vCPUs e memória como atributos estruturados, esse campo constitui a principal fonte para a inferência das especificações técnicas no extrator Azure.

De forma geral, o nome do SKU pode ser decomposto nos segmentos abaixo:

```
[Family][vCPUs][-Constrained][Features]_v[Version]
Exemplo: Standard_D4s_v3   -> família D, 4 vCPUs
Exemplo: Standard_E8-2s_v4 -> família E, 8 vCPUs
Exemplo: Standard_M128ms_v2 -> família M, 128 vCPUs
```

A partir dessa convenção, o extrator empregou expressões regulares para extrair a família da instância e o número de vCPUs associado ao SKU. Em seguida, a quantidade de RAM foi estimada multiplicando o número de vCPUs por uma razão típica de memória por processador, definida conforme a família da instância (Tabela 4). Quando a família não possui valor específico cadastrado, adota-se a razão padrão de 4 GB por vCPU. Essa abordagem permite completar o esquema unificado de dados, ainda que a memória resultante represente uma aproximação em relação ao catálogo oficial da Microsoft.

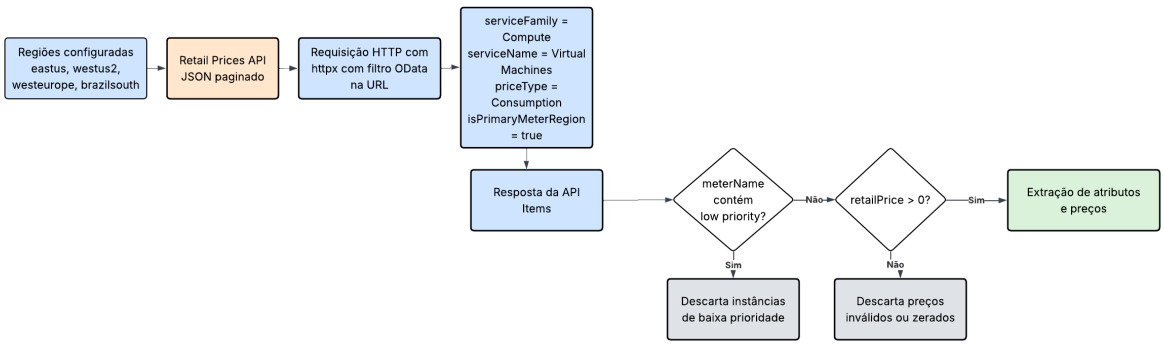
Tabela 4 – Razão RAM/vCPU estimada por família de instâncias Azure

Família	**RAM/vCPU (GB)**
A, F, H, HB, HC	2,0
B, D, DC, FX, NV, NP	4,0
E, EC, G, GS, L	8,0
M	16,0
NC, ND	8,0

Fonte: Elaborado pelo autor.

O diagrama abaixo exemplifica como o processo de extração dos dados acontece e os devidos filtros aplicados:

Figura 5 – Fluxo de extração e filtros aplicados no extrator Azure



Fonte: Elaborado pelo autor.

A Tabela 5 resume os campos retornados em cada item da lista `Items` e como eles alimentam o esquema unificado.

Tabela 5 – Principais campos retornados pela Retail Prices API do Azure

Campo na API	**Conteúdo**	**Uso no pipeline**
`armSkuName`	Nome do SKU (ex. `Standard_D4s_v3`)	`instance_type`, inferência de vCPUs e RAM
`retailPrice`	Preço de varejo por hora	`price_per_hour_usd`
`armRegionName`	Região	`region`
`meterName`	Nome do medidor de cobrança	Detecção de Spot e SO
`productName`	Nome comercial do produto	`instance_family`, detecção de SO
`priceType`	Tipo de preço	Restringe a `Consumption`
`NextPageLink`	URL da página seguinte	Paginação automática

Fonte: Elaborado pelo autor.

3.2.4 Extrator GCP [Cloud Billing Catalog API]

Figura 6 – Arquitetura do extrator GCP.

Atributo	Detalhes
URL Base	https://cloudbilling.googleapis.com/v1/services/{serviceId}/sku
Autenticação	Necessita de API Key
Formato de resposta	JSON paginado
Regiões coletadas	us-east1, us-central1, europe-west1, southamerica-east1
Documentação oficial	https://cloud.google.com/billing/v1/how-tos/catalog-api

Fonte: Elaborado pelo autor.

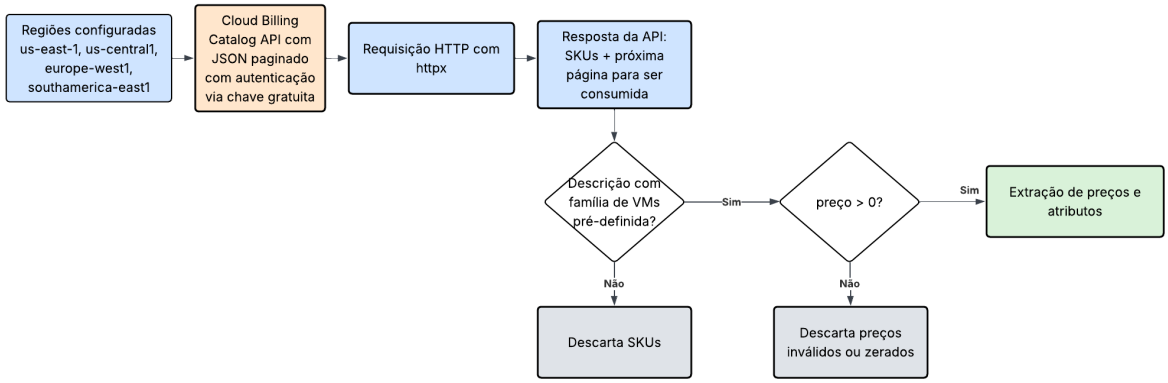
Diferentemente dos provedores AWS e Azure, o Google Cloud Platform não disponibiliza diretamente o preço consolidado de cada tipo de máquina virtual por meio da Cloud Billing Catalog API (Google Cloud, 2024). A precificação é exposta em nível de componentes de infraestrutura, com tarifas independentes para vCPUs e memória RAM, variando conforme a família de hardware e a região de implantação. Dessa forma, a obtenção do custo de uma instância exige a reconstrução do preço a partir da combinação desses componentes.

Para contornar essa limitação, o extrator combinou duas fontes de informação. A primeira envolve os SKUs retornados pela API, responsáveis por fornecer as tarifas

unitárias de processamento e memória e a segunda envolvendo um catálogo interno de máquinas pré-definidas, contendo as especificações de cada instância, como quantidade de vCPUs, memória RAM e GPUs associadas.

A coleta é realizada por meio de requisições HTTP autenticadas com chave de API. Para cada região configurada, os SKUs são extraídos de forma paginada até o esgotamento dos resultados disponíveis. Em seguida, são aplicados filtros para manter apenas registros relacionados a instâncias pré-definidas, descartando componentes de computação dedicada, serviços auxiliares e preços inválidos ou nulos. O fluxo completo é apresentado na Figura 7.

Figura 7 – Fluxo de coleta e processamento dos SKUs do GCP.



Fonte: Elaborado pelo autor.

Após a etapa de filtragem, cada SKU é classificado de acordo com sua família de hardware e com o recurso faturado. Os valores extraídos são então organizados em uma estrutura indexada por região, família e tipo de componente, formando um catálogo de preços unitários que serve como base para o cálculo do custo das instâncias.

A reconstrução do preço é realizada associando as tarifas unitárias às especificações de cada tipo de máquina presente no catálogo interno. O preço por hora de uma instância é calculado pela soma do custo proporcional de processamento e memória, conforme definido pela Equação 3.1.

$$P_{\text{total}}^{\text{GCP}} = P_{\text{core}} \cdot N_{\text{vcpus}} + P_{\text{ram}} \cdot N_{\text{ram\_gb}} \tag{3.1}$$

em que P_{core} e P_{ram} representam as tarifas cobradas por hora de uso retornadas pela API para vCPU e memória, enquanto N_{vcpus} e $N_{\text{ram\_gb}}$ correspondem às especificações da instância analisada.

Quando aplicável, o atributo `gpu_count` é preenchido com base nas especificações oficiais do tipo de máquina. Entretanto, o preço reconstruído considera apenas os componentes de processamento e memória, uma vez que as GPUs são cobradas separadamente por SKUs específicos no modelo de faturamento do GCP.

Adicionalmente, para cada instância *On-Demand* gerada, o pipeline cria um registro *Spot* correspondente utilizando uma aproximação baseada em percentual do valor inicial, já que a API utilizada não fornece preços preemptíveis diretamente associados aos tipos de máquina.

A Tabela 6 apresenta os principais atributos consumidos da Cloud Billing Catalog API e sua utilização no processo de transformação para o esquema unificado.

Tabela 6 – Principais campos utilizados da Cloud Billing Catalog API

Campo na API	**Conteúdo**	**Uso no pipeline**
`description`	Texto do SKU (ex.: N2 Instance Core)	Família e componente (*core* ou *ram*)
`usageType`	Modalidade (ex.: `OnDemand`)	Filtragem de modalidade
`serviceRegions`	Regiões (ex.: `us-east1`)	`region`
`unitPrice`	Tarifa/hora em `pricingInfo` (ex.: 0,03314 USD/h)	`price_per_hour_usd`
Catálogo	Tipo e specs (ex.: `n2-standard-4`, 4 vCPUs, 16 GB)	`instance_type`, `vcpus`, `ram_gb`

Fonte: Elaborado pelo autor.

3.2.5 Tratamento e limpeza dos dados

Após a extração dos três provedores, os registros brutos passaram por um módulo de transformação que aplica, de forma sequencial, etapas de filtragem e limpeza de dados. O volume bruto inicial extraído foi de 158.898 registros. Ao final do pipeline de limpeza, o banco unificado contém 34.026 instâncias válidas (21,4% de retenção). A Tabela 7 resume a contagem de linhas após cada etapa.

Tabela 7 – Pipeline de normalização: registros removidos por etapa

Etapa	**Operação**	**Removidos**	**Restantes**
—	Dados brutos (AWS, Azure e GCP)	—	158.898
1	Validação de especificações técnicas	4.309	154.589
2	Remoção de taxa adicional de licença	61	154.528
3	Deduplicação por chave composta	120.502	34.026
	Total retido		**34.026 (21,4%)**

Fonte: Elaborado pelo autor com base na execução do pipeline ETL.

A etapa de transformação removeu registros com especificações ou preços inválidos, excluiu cobranças adicionais de licenciamento e eliminou duplicidades por meio de uma chave composta formada por `provider`, `instance_type`, `vcpus`, `ram_gb`, `region`, `os` e `price_type`.

Após esses filtros, 21,4% dos registros originais foram preservados na base consolidada. A maior redução ocorre nas extrações da AWS e Azure devido à elevada quantidade de registros redundantes, enquanto os dados reconstruídos do GCP apresentaram perdas desprezíveis ao longo do pipeline.

3.2.6 Seleção das instâncias para análise

A base consolidada contém registros de instâncias nas modalidades *On-Demand* e *Spot*. Entretanto, apenas as instâncias On-Demand foram consideradas nas etapas de clusterização e modelagem preditiva desenvolvidas neste trabalho.

Embora as instâncias *Spot* representem uma alternativa de menor custo, sua precificação é influenciada por mecanismos de mercado específicos de cada provedor, como a disponibilidade momentânea de capacidade computacional e políticas internas de desconto. Além disso, essas instâncias podem ser interrompidas a qualquer momento pelo provedor, caracterizando uma modalidade de contratação distinta daquela utilizada em cenários de uso contínuo.

Por outro lado, as instâncias *On-Demand* apresentam um modelo de cobrança mais uniforme entre AWS, Azure e GCP, no qual o preço está diretamente associado às características computacionais provisionadas. Dessa forma, restringir a análise a essa modalidade reduz fontes externas de variabilidade e permite que os modelos capturem predominantemente a relação entre recursos de hardware e custo.

A utilização exclusiva de instâncias *On-Demand* também favorece a comparabilidade entre provedores, uma vez que os agrupamentos passam a refletir similaridades de configuração computacional, enquanto os modelos preditivos podem ser treinados sobre uma estrutura de preços mais consistente. Assim, todas as etapas de clusterização e regressão descritas nas seções seguintes foram realizadas exclusivamente sobre o subconjunto de instâncias *On-Demand* da base consolidada.

3.3 Clusterização de instâncias utilizando K-Means e GMM

A clusterização constitui a primeira etapa de modelagem aplicada ao catálogo unificado de instâncias. Seu objetivo é identificar grupos de máquinas virtuais com características computacionais semelhantes, representando perfis de capacidade compartilhados entre diferentes configurações.

Para essa tarefa foram avaliados dois algoritmos de aprendizado não supervisionado amplamente utilizados na literatura, K-Means e Gaussian Mixture Models. O K-Means foi escolhido por sua simplicidade e interpretabilidade, enquanto o GMM foi incluído por permitir uma representação probabilística dos agrupamentos e maior flexibilidade na modelagem dos dados.

A comparação entre ambos os métodos permite avaliar qual abordagem melhor representa a estrutura do catálogo. Como resultado, cada instância *On-Demand* recebe um identificador inteiro de agrupamento (`cluster_id`), posteriormente utilizado nas análises comparativas e nos cenários de uso desenvolvidos neste trabalho.

3.3.1 Construção da matriz de atributos

A clusterização foi conduzida utilizando apenas atributos relacionados às características computacionais das instâncias. O objetivo é que os agrupamentos reflitam similaridades de capacidade de processamento e memória, e não diferenças comerciais associadas ao provedor, à região ou ao modelo de precificação.

Por esse motivo, variáveis como `provider`, `region` e `price_per_hour_usd` não foram incluídas na matriz de entrada. A inclusão desses atributos tenderia a segmentar as instâncias por nuvem, localização geográfica ou faixa de preço, desviando o foco da análise de perfis computacionais equivalentes.

As variáveis efetivamente utilizadas na construção da matriz de atributos são apresentadas na Tabela 8.

Tabela 8 – Variáveis utilizadas na clusterização On-Demand

Tipo	**Campo**	**Papel no modelo**
Numérico	`vcpus`	Capacidade de processamento
Numérico	`ram_gb`	Memória principal
Numérico	`gpu_count`	Presença e quantidade de GPUs
Numérico	`ram_per_vcpu`	Relação memória por vCPU
Categórico	`os`	Sistema operacional
Excluído	`price_per_hour_usd`	Não utilizado no agrupamento

Fonte: Elaborado pelo autor.

Dessa forma, os agrupamentos são definidos exclusivamente por características computacionais das instâncias, evitando que fatores comerciais ou geográficos influenciem a formação dos clusters.

3.3.2 Pré-processamento

O ajuste restringiu-se ao subconjunto *On-Demand* do catálogo consolidado pelo ETL (Seção 3.2), totalizando 16.310 instâncias. As instâncias *Spot* permaneceram no catálogo para consulta e análise exploratória, mas ficaram fora desta etapa por refletirem modalidade contratual e dinâmica de preço distintas.

A variável derivada `ram_per_vcpu` foi calculada como `ram_gb`/`vcpus` e os valores ausentes em `gpu_count` foram tratados como zero. As variáveis numéricas foram padronizadas utilizando o método *StandardScaler*, de modo que todos os atributos contribuíssem de

forma equilibrada para o cálculo das distâncias utilizadas pelos algoritmos de agrupamento. A variável categórica `os` foi transformada por meio de *one-hot encoding*, permitindo sua incorporação ao vetor de características sem introduzir relações ordinais artificiais entre categorias.

Após o pré-processamento, cada instância passou a ser representada por um vetor numérico pertencente à matriz de entrada X, utilizada pelos algoritmos de clusterização.

3.3.3 Treinamento dos modelos

Após a construção da matriz de características X, foram treinadas os dois modelos de agrupamento amplamente utilizadas para identificação de instâncias tecnicamente similares.

A utilização de ambos os métodos teve como objetivo investigar qual estratégia representa de forma mais adequada a estrutura do catálogo de máquinas virtuais. Enquanto o K-Means assume agrupamentos compactos organizados em torno de protótipos centrais, o GMM permite que uma mesma instância apresente diferentes graus de pertencimento aos agrupamentos, capturando possíveis transições entre perfis de configuração semelhantes.

Para garantir uma comparação justa entre os algoritmos, foram mantidos constantes o conjunto de dados, o processo de pré-processamento e os critérios de avaliação. Os modelos foram treinados para diferentes números de agrupamentos, considerando $K \in 2, \ldots, 20$.

No K-Means, foram utilizados os hiperparâmetros `n_init=10` e `max_iter=300`, permitindo múltiplas inicializações dos centróides e reduzindo a sensibilidade a soluções locais. Para o GMM, foram adotados `n_init=5` e `max_iter=300`, mantendo o mesmo limite de iterações e múltiplas inicializações do processo de estimação.

Como o objetivo do trabalho consiste em identificar grupos de instâncias com características técnicas semelhantes, os rótulos finais do GMM foram obtidos por meio da atribuição de cada observação ao componente com maior probabilidade posterior. Dessa forma, ambos os algoritmos produziram partições diretamente comparáveis, permitindo a análise quantitativa da qualidade dos agrupamentos e a seleção da configuração mais adequada para o catálogo consolidado.

3.3.4 Avaliação e seleção de K

A qualidade dos agrupamentos foi avaliada por meio de métricas de clusterização, calculadas para cada combinação de algoritmo e valor de K. Foram utilizados três indicadores complementares, conforme a literatura de validação de agupamentos:

- **Silhueta** ($s \in [-1, 1]$): quanto maior, melhor a coesão intra-*cluster* e a separação entre grupos (ROUSSEEUW, 1987; KAUFMAN; ROUSSEEUW, 1990);
- **Davies-Bouldin**: quanto menor, melhor a relação entre compacidade interna e separação entre centroides (DAVIES; BOULDIN, 1979);

- **Participação do maior *cluster***: percentual de instâncias no grupo mais populoso, empregado para identificar partições excessivamente concentradas em um único segmento.

A seleção do número de *clusters* foi realizada por busca exaustiva na faixa $K \in \{2, \ldots, 20\}$. Para cada configuração, as três medidas foram registradas e comparadas relativamente entre os valores de K, em linha com a recomendação de empregar múltiplos índices de forma conjunta, e não depender de um único indicador isolado (HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2001; ARBELAITZ et al., 2013).

A decisão final buscou um equilíbrio entre qualidade geométrica (Silhueta e Davies-Bouldin) e distribuição amostral (participação do maior *cluster*). Valores muito baixos de K tendem a produzir Silhueta artificialmente elevada, mas concentram quase todo o catálogo em um único grupo, por isso, configurações com concentração extrema foram descartadas independentemente da Silhueta. Entre os valores restantes, privilegiou-se o menor K em que as duas métricas geométricas apresentam melhora conjunta e a distribuição deixa de ser dominada por um único *cluster*, favorecendo parcimônia e interpretabilidade dos perfis.

O GMM foi aplicado sob o mesmo protocolo, apenas como referência comparativa. Caso o método probabilístico não apresentasse desempenho equivalente ou superior ao K-Means na mesma faixa de K, permaneceria fora da geração dos agrupamentos finais.

3.4 Regressão de preços utilizando Regressão Linear e Modelos baseados em Árvores

A regressão de preços constitui o segundo estágio de modelagem sobre o conjunto de dados. Trata-se de aprendizado supervisionado onde o modelo recebe atributos técnicos e contextuais de cada instância e aprende a estimar o preço de referência por hora em dólares. O resultado é o valor previsto $\hat{P}$ (USD/h) e, derivado dele, o indicador *Value Ratio*, utilizados na análise comparativa multi-provedor e na calculadora conceitual.

Diferentemente da clusterização, a regressão incorpora `provider`, `region` e `os`, pois o preço de tabela depende do contexto de comercialização além da capacidade bruta de *hardware*. O `cluster_id` produzido na etapa anterior permanece no catálogo para consulta e segmentação descritiva, mas não entra como variável preditora, evitando que o regressor reproduza o agrupamento em vez de aprender o mapeamento dos preços e atributos diretamente.

3.4.1 Base de dados e variáveis

O ajuste restringiu-se novamente ao mesmo subconjunto *On-Demand* empregado na clusterização.

Por existir variável-alvo, adotou-se avaliação de generalização em amostra reservada. O conjunto foi particionado aleatoriamente em 80% para treino e 20% para teste. Em paralelo, aplicou-se validação cruzada com cinco *folds* embaralhados (KFold) sobre o bloco de treino, para estimar a estabilidade do R^2 entre partições.

A variável resposta é `price_per_hour_usd`. O treinamento foi conduzido na escala original (USD/h), privilegiando a interpretabilidade direta das métricas de erro e das predições.

As variáveis preditoras e exclusões constam na Tabela 9. Foram removidas colunas que reproduzem o alvo ou são funções determinísticas dele, pois induzem a um vazamento do atributo alvo de maneira indireta: `cost_per_vcpu_hour`, `cost_per_gb_ram_hour`, `price_per_vcpu`, `price_per_ram_gb` e `price_per_month_usd`.

Tabela 9 – Variáveis utilizadas na regressão de preços On-Demand

Tipo	**Campo**	**Papel no modelo**
Numérico	`vcpus`	Capacidade de processamento
Numérico	`ram_gb`	Memória principal
Numérico	`ram_per_vcpu`	Relação memória por vCPU
Numérico	`gpu_count`	Presença e quantidade de GPUs
Categórico	`provider`	Provedor (AWS, Azure, GCP)
Categórico	`region`	Região de listagem
Categórico	`os`	Sistema operacional
Alvo	`price_per_hour_usd`	Preço On-Demand por hora (USD)
Excluído	Derivadas de preço/unidade	Evitam *leakage*

Fonte: Elaborado pelo autor.

Os atributos numéricos foram padronizados utilizando normalização (`StandardScaler`) e os categóricos foram codificados por indicadores binários (`OneHotEncoder`). Os valores ausentes nas variáveis numéricas foram tratados com zero antes da padronização e categorias ausentes foram mapeadas para a categoria desconhecida.

3.4.2 Treinamento dos modelos

Três algoritmos foram comparados sob o mesmo pré-processamento, a mesma partição treino/teste e a mesma validação cruzada:

- **Regressão Linear com regularização Ridge** (HASTIE; TIBSHIRANI; FRIEDMAN, 2009) (`alpha=1,0`), como cenário base de parâmetro;
- **Random Forest** (BREIMAN, 2001) (`n_estimators=300`, `max_depth=6`, `min_samples_split=5`
- **XGBoost** (CHEN; GUESTRIN, 2016) (`n_estimators=300`, `max_depth=6`, `learning_rate=0,0` `subsample=0,8`, `colsample_bytree=0,8`).

A comparação isola o efeito do algoritmo mantendo constantes as *features*, a transformação do alvo e as partições. O XGBoost foi incluído por combinar capacidade de capturar relações não lineares entre atributos e preço com mecanismos de regularização (*subsample*, *colsample_bytree*) que limitam sobreajuste em presença de categorias de alta cardinalidade, como `region`.

3.4.3 Avaliação e seleção do regressor

A qualidade de cada algoritmo foi medida por métricas complementares no conjunto de teste reservado e, de forma auxiliar, por validação cruzada no treino:

- **MAE** (erro absoluto médio, USD/h): interpretação direta do desvio médio em moeda;
- **RMSE** (raiz do erro quadrático médio, USD/h): penaliza erros grandes na cauda de preços;
- R^2 (coeficiente de determinação): fração da variância do preço explicada pelo modelo.

A seleção do regressor principal não maximizou isoladamente o R^2 no teste, pois modelos mais flexíveis podem ajustar ruído amostral sem reduzir o erro médio. Adotou-se o algoritmo com menor MAE no conjunto de teste, utilizando R^2 e a média (desvio padrão) do R^2 na validação cruzada como critérios de suporte. Os resultados obtidos da comparação entre Ridge, Random Forest e XGBoost estão no Capítulo 4.

3.4.4 Indicador de custo-benefício

Após a seleção do melhor regressor, cada instância *On-Demand* recebe o *Value Ratio* (VR), definido como:

$$\textit{Value Ratio} = \frac{P_{\text{real}}}{\hat{P}_{\text{predito}}} \tag{3.2}$$

em que P_{real} é o preço On-Demand registrado no catálogo e $\hat{P}_{\text{predito}}$ é a estimativa em USD/h produzida pelo modelo selecionado. Valores menores que 1 indicam preço real inferior ao previsto para aquela configuração (melhor custo-benefício relativo ao padrão aprendido). Valores maiores que 1 indicam preço real superior ao previsto. O VR complementa a predição pontual com uma medida relativa de posicionamento de preço no catálogo, derivada exclusivamente da comparação entre preço observado e preço estimado.

3.5 Cenários de aplicação

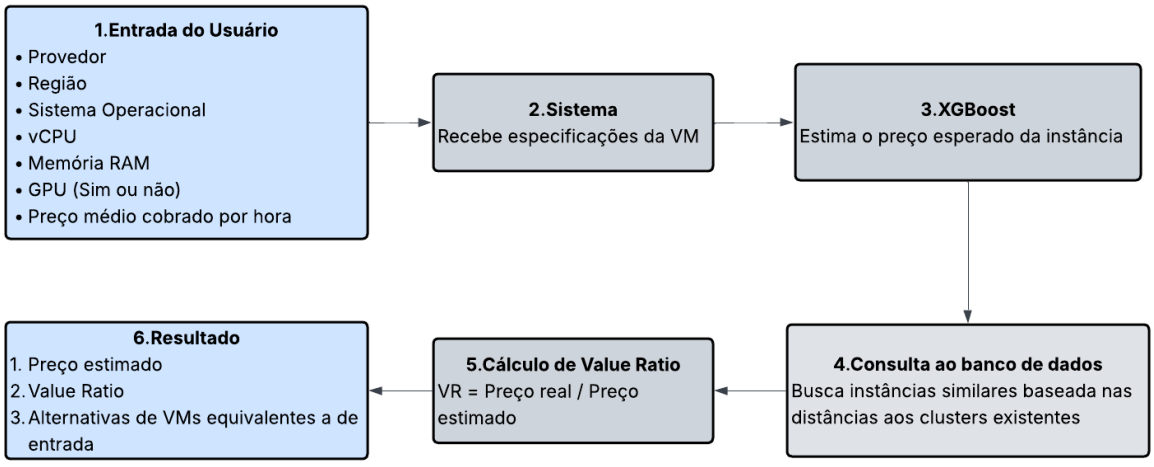
3.5.1 Cenário 1: análise de aderência de preço

O primeiro cenário considera uma instância de referência cuja configuração técnica e preço são conhecidos pelo usuário. O objetivo é verificar se o valor cobrado está alinhado ao padrão observado no catálogo para instâncias de características semelhantes.

Inicialmente, os atributos da instância são fornecidos ao modelo XGBoost, responsável por estimar o preço esperado para aquela configuração. Em seguida, calcula-se o indicador *Value Ratio*, obtido pela razão entre o preço observado e o preço estimado pelo modelo. Por fim, o identificador de agrupamento (`cluster_id`) é utilizado para localizar instâncias pertencentes ao mesmo perfil técnico, permitindo contextualizar o resultado obtido em relação ao conjunto de alternativas equivalentes disponíveis no catálogo.

Esse processo produz uma interpretação relativa do preço analisado, indicando se a instância se encontra alinhada, potencialmente acima ou abaixo do comportamento esperado para configurações semelhantes. A Figura 8 apresenta o fluxo conceitual adotado.

Figura 8 – Fluxo conceitual do Cenário 1: análise de aderência de preço



Fonte: Elaborado pelo autor.

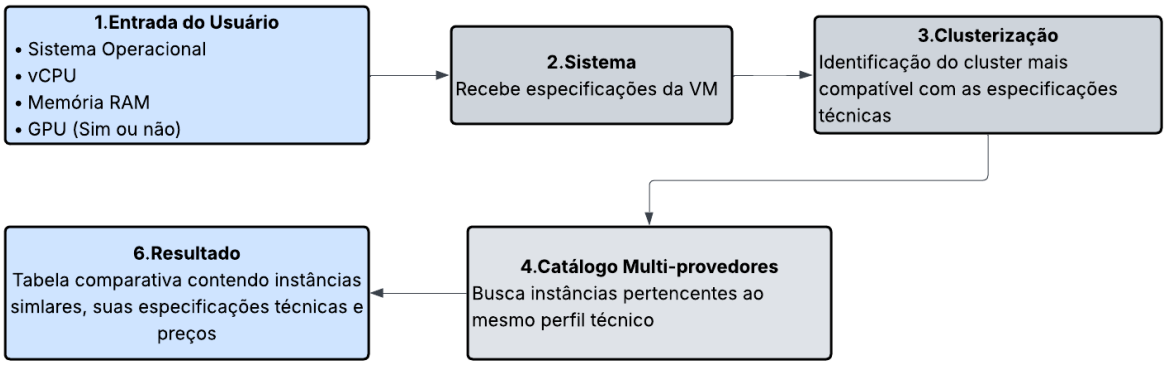
3.5.2 Cenário 2: busca exclusiva por alternativas equivalentes

O segundo cenário considera uma configuração de referência definida pelo usuário e tem como objetivo identificar instâncias comparáveis ofertadas por diferentes provedores de nuvem.

A partir das características técnicas informadas, o modelo de clusterização determina o agrupamento mais compatível com a configuração analisada. Em seguida, são recuperadas do catálogo as instâncias pertencentes ao mesmo perfil de capacidade computacional. Para

cada alternativa encontrada, são disponibilizados as informações técnicas e o preço de cobrado. A Figura 9 apresenta o fluxo conceitual correspondente a esse caso de uso.

Figura 9 – Fluxo conceitual do Cenário 2: busca por alternativas equivalentes



Fonte: Elaborado pelo autor.

4 Experimentos e Resultados

Este capítulo descreve o conjunto de dados produzido pelo *pipeline* ETL e apresenta os resultados dos experimentos de aprendizado de máquina conduzidos sobre o subconjunto *On-Demand*. A exposição segue a ordem metodológica. Inicialmente, caracteriza-se o conjunto de dados por meio de estatística descritiva. Em seguida, comparam-se os algoritmos de clusterização, justificando as decisões de modelagem, as métricas de avaliação adotadas e os perfis resultantes dos agrupamentos. Posteriormente, são comparados os modelos de regressão, analisando-se o desempenho do modelo selecionado para a predição de preços com base nos padrões observados no mercado.

Por fim, os modelos escolhidos são empregados em dois cenários conceituais de uso, demonstrando na prática como a arquitetura proposta pode ser aplicada à análise de preços e à identificação de alternativas equivalentes entre provedores de nuvem.

4.1 Estrutura do conjunto de dados obtido

O pipeline de extração e transformação produziu um banco de dados unificado com 34.026 linhas e 21 colunas no esquema relacional persistido. Dessas colunas, 16 provêm diretamente das APIs dos provedores depois de passarem pela normalização, 3 são atributos derivados calculados na camada de transformação e 2 registram metadados quando o código é executado. Cada linha representa uma combinação única de provedor, tipo de instância, região, sistema operacional, GPU e modalidade de preço.

A Tabela 10 resume o esquema relacional persistido na tabela resultante, denominada `cloud_instances`. Os campos técnicos (`vcpus`, `ram_gb`, `gpu_count`) alimentam tanto a clusterização quanto a regressão. Os campos de contexto (`provider`, `region`, `os`, `category`) permitem capturar diferenças de mercado entre provedores e regiões. O preço por hora de uso das VMs, `price_per_hour_usd`, é a variável-alvo da regressão, mas não entra no agrupamento K-Means, conforme definido na Seção 3.3.

Tabela 10 – Esquema completo da tabela `cloud_instances`.

Coluna	**Tipo SQL**	**Descrição**
Campos primários (extraídos e normalizados do ETL)		
`id`	INTEGER PK	Identificador auto-incrementado
`provider`	VARCHAR(20)	Provedor: `AWS`, `Azure` ou `GCP`. Indexado
`instance_type`	VARCHAR(100)	Nome do tipo (e.g., `m5.xlarge`, `Standard_D4s_v3`)

(continuação da Tabela 10)

Coluna	Tipo SQL	Descrição
`vcpus`	INTEGER	Número de processadores virtuais
`ram_gb`	FLOAT	Memória RAM em GB
`region`	VARCHAR(50)	Código da região do datacenter. Indexado
`os`	VARCHAR(20)	Sistema operacional: `Linux` ou `Windows`
`price_per_hour_usd`	FLOAT	Preço *on-demand* por hora em USD
`price_per_month_usd`	FLOAT	Preço mensal estimado
`price_type`	VARCHAR(20)	Modalidade: `on_demand` ou `spot`
`gpu_count`	INTEGER	Número de GPUs
`gpu_model`	VARCHAR(100)	Modelo da GPU (se aplicável)
`instance_family`	VARCHAR(100)	Família/série
`architecture`	VARCHAR(100)	Processador (e.g., `Intel Xeon Platinum`)
`network_performance`	VARCHAR(100)	Performance de rede (e.g., `Up to 10 Gbps`)
Features derivadas (calculadas na normalização)		
`cost_per_vcpu_hour`	FLOAT	P_{hora}/N_{vcpus}
`cost_per_gb_ram_hour`	FLOAT	P_{hora}/N_{ram_gb}
`ram_per_vcpu`	FLOAT	N_{ram_gb}/N_{vcpus}
Metadados		
`extraction_run_id`	INTEGER	FK para tabela `extraction_runs`. Indexado
`created_at`	TIMESTAMP	Data de inserção (com timezone, default `now()`)

Fonte: Elaborado pelo autor.

4.2 Análise exploratória do dataset obtido

Esta seção apresenta uma visão geral do catálogo consolidado após as etapas de extração, normalização e deduplicação. O objetivo é caracterizar a composição dos dados que foi utilizada nos experimentos.

4.2.1 Distribuição por provedor e modalidade de preço

A Tabela 11 apresenta a distribuição do catálogo consolidado por provedor e modalidade de contratação. Ao todo, foram obtidos 34.026 registros, dos quais 16.310 correspondem a instâncias *On-Demand* e 17.716 a instâncias *Spot*.

Observa-se uma predominância de registros provenientes da Azure, responsável por 57,2% do catálogo final, seguida pela AWS com 38,6%. O GCP representa 4,2% da base, participação inferior aos demais provedores em razão das limitações impostas pela estratégia de reconstrução de preços adotada para a Cloud Billing Catalog API, que restringe a cobertura a famílias específicas de máquinas virtuais.

Em relação às modalidades de contratação, verifica-se uma distribuição relativamente equilibrada entre instâncias *On-Demand* e *Spot*. Essa característica amplia o potencial de análises comparativas entre modelos de precificação. Entretanto, conforme discutido na Metodologia, apenas as instâncias com modelo de precificação foram consideradas paras as experimentos posteriores.

Tabela 11 – Distribuição do dataset por provedor e tipo de preço

Provedor	**On-Demand**	**Spot**	**Total**	**Proporção**
Amazon Web Services	6.559	6.559	13.118	38,6%
Microsoft Azure	9.027	10.433	19.460	57,2%
Google Cloud Platform	724	724	1.448	4,3%
Total	**16.310**	**17.716**	**34.026**	**100%**

Fonte: Elaborado pelo autor com base no catálogo consolidado.

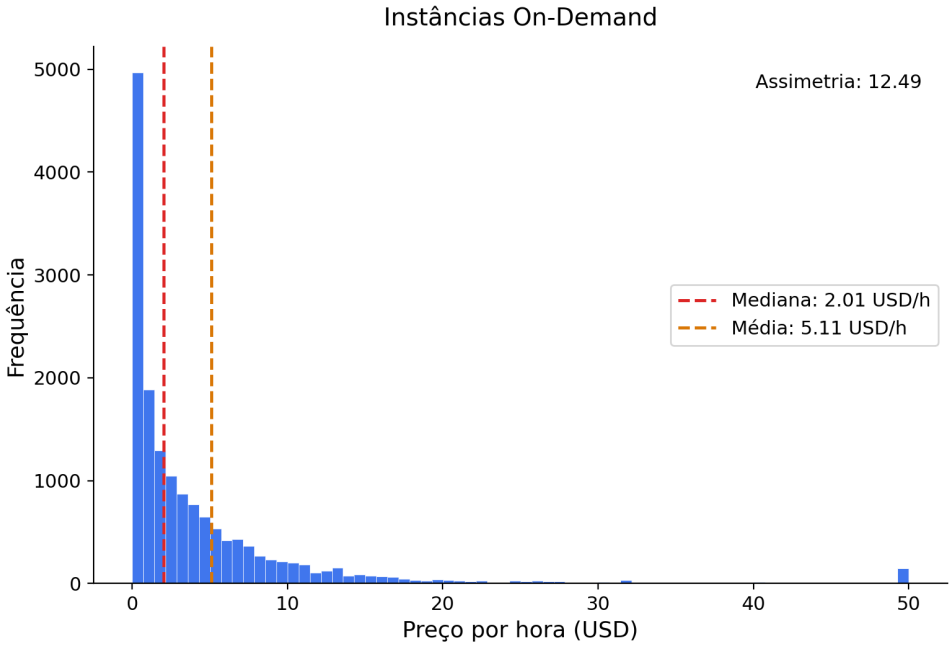
4.2.2 Distribuição de preços

Para entender a distribuição dos preços a partir do dataset construído, segmentou-se a análise a partir dos perfis de precificação.

Analisando o subconjunto de instâncias *On-Demand*, o preço por hora de provisionamento varia de USD 0,004/h a USD 360,99/h. A média amostral é USD 5,10/h e a mediana USD 2,01/h. A mediana inferior à média indica concentração de instâncias de entrada e cauda longa de configurações especializadas. Essa concentração indica que o consumo computacional dos provedores é destinado principalmente a máquinas com capacidade computacional inferior, focadas em execução de propósito geral.

A Figura 10 reúne a distribuição em escala linear das instâncias do conjunto e a distribuição do preço por hora. Todas as VMs com valores iguais ou superiores a 50 USD por hora foram inseridas no último *bin* do gráfico.

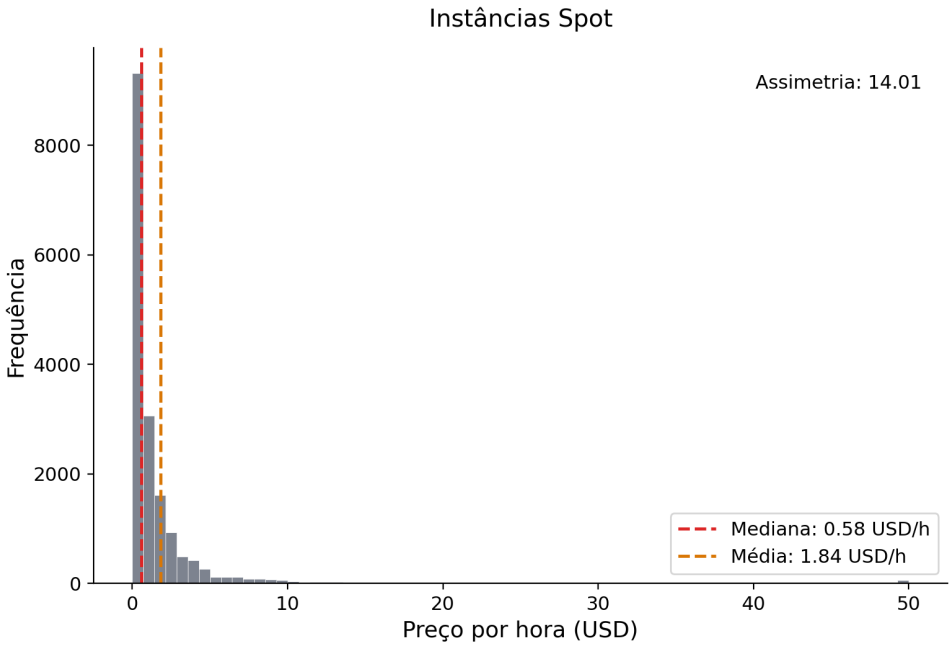
Figura 10 – Histograma de VMs On-Demand e seus respectivos preços por hora



Fonte: Elaborado pelo autor.

No subconjunto de VMs *Spot*, o preço horário varia de USD 0,0013/h a USD 184,60/h. A média amostral é USD 1,83/h e a mediana USD 0,59/h. Os valores são substancialmente inferiores aos observados em On-Demand. A mediana inferior à média indica, também aqui, concentração de instâncias de entrada e cauda longa de configurações especializadas.

Figura 11 – Histograma de VMs Spot e seus respectivos preços por hora



Fonte: Elaborado pelo autor.

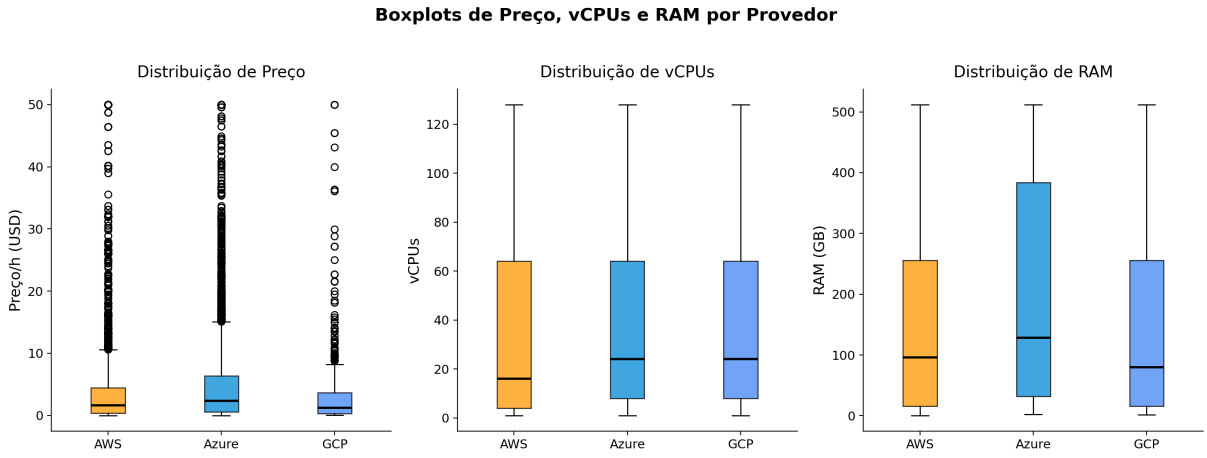
Instâncias *Spot* tendem a ser mais baratas porque o provedor comercializa capacidade

ociosa com desconto, sem garantia de disponibilidade contínua. Nesse contexto, a máquina pode ser interrompida (preempção) quando a demanda por capacidade sob demanda ou reservada aumenta. Esse desconto compensa o risco operacional assumido pelo consumidor e está de acordo com o comportamento esperado.

4.2.3 Recursos computacionais *On-Demand* e heterogeneidade entre provedores

A Figura 12 sintetiza a distribuição de preço por hora, vCPUs e memória RAM das instâncias On-Demand por provedor.

Figura 12 – Distribuição de preço por hora, vCPUs e memória RAM por provedor.



Fonte: Elaborado pelo autor.

AWS e Azure apresentam maior variabilidade em capacidade computacional, refletida pelas amplas faixas interquartis observadas para vCPUs e RAM. As medianas de vCPUs são de 16 e 24 para AWS e Azure, respectivamente, enquanto as medianas de memória RAM atingem 96 GB e 128 GB. No GCP, a mediana situa-se em 24 vCPUs e 80 GB de RAM, com participação amostral inferior à dos demais provedores (724 instâncias On-Demand, contra 6.559 da AWS e 9.027 da Azure).

No painel de preços, Azure concentra valores medianos ligeiramente superiores (2,32 USD/h) em relação à AWS (1,64 USD/h) e ao GCP (1,21 USD/h), com caudas longas até o limite de visualização de 50 USD/h adotado no gráfico.

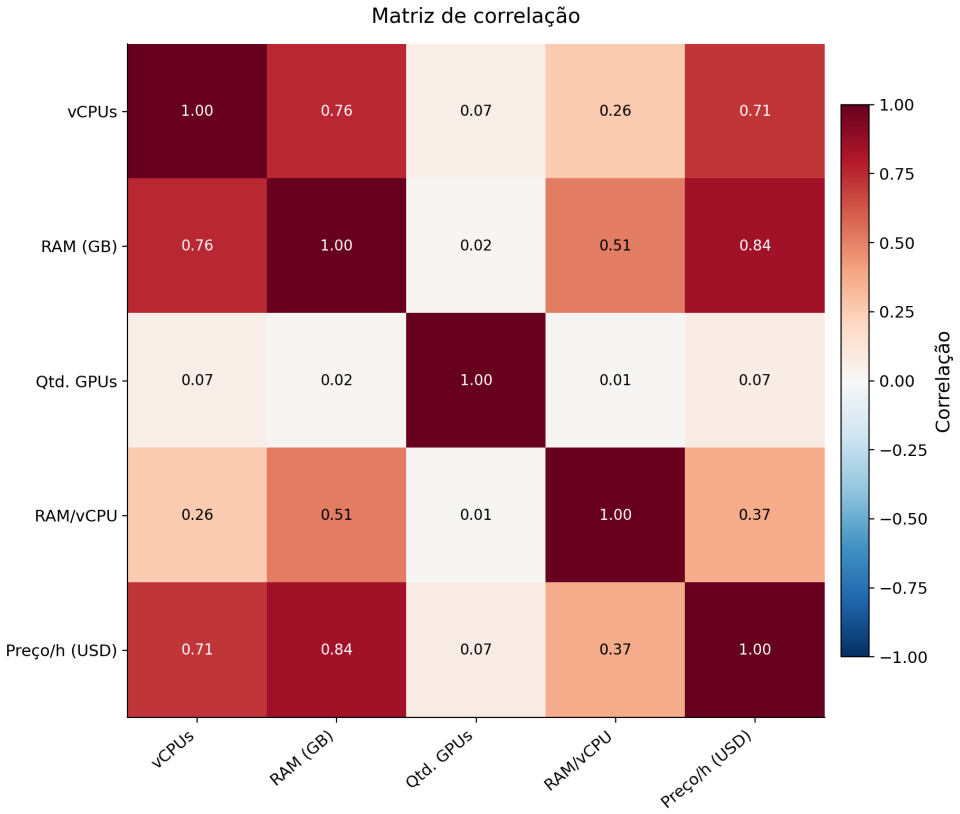
Para preservar a legibilidade da região central das distribuições de vCPUs e RAM, esses painéis foram limitados a 128 vCPUs e 512 GB de RAM. O recorte afeta apenas a visualização gráfica, não os dados utilizados nas análises. No catálogo completo existem instâncias substancialmente maiores, alcançando até 896 vCPUs e 32 TB de RAM na AWS (`u7in-32tb.224xlarge`) e até 416 vCPUs e 6,6 TB de RAM na Azure (`Standard_M416s_v2`). Os limites afetam 6,2% dos registros da AWS e 3,9% da Azure em vCPUs, além de 12,2% e 13,3% desses provedores em memória RAM, respectivamente.

No GCP, 5,5% dos registros ultrapassam 128 vCPUs (famílias M3 e equivalentes, até 256 vCPUs) e 12,7% ultrapassam 512 GB de RAM.

4.2.4 Matriz de correlação

A Figura 13 apresenta os coeficientes de correlação de Pearson entre as variáveis numéricas do catálogo consolidado. As variáveis categóricas, como provedor (`provider`) e sistema operacional (`os`), não foram incluídas por não possuírem interpretação direta em termos de correlação linear.

Figura 13 – Matriz de correlação de Pearson entre as variáveis numéricas do catálogo consolidado.



Fonte: Elaborado pelo autor.

Observa-se forte correlação positiva entre memória RAM e preço por hora ($r = 0{,}84$), indicando que instâncias com maior capacidade de memória tendem a apresentar valores mais elevados. Relações também expressivas são observadas entre número de vCPUs e preço ($r = 0{,}71$) e entre vCPUs e RAM ($r = 0{,}76$), refletindo a tendência dos provedores de dimensionarem conjuntamente recursos de processamento e memória.

A variável `ram_per_vcpu` apresenta correlação moderada com o preço ($r = 0{,}38$), sugerindo que a densidade de memória influencia o valor das instâncias, embora em menor magnitude que os recursos absolutos de processamento e memória. Em contraste, a quantidade de GPUs exibe correlação linear reduzida com o preço ($r = 0{,}07$). Esse

comportamento pode ser atribuído à menor representatividade de instâncias aceleradas no conjunto de dados e à elevada variabilidade de preços observada nesse segmento.

De modo geral, os resultados indicam que memória RAM e número de vCPUs concentram a maior parcela da variação de preços observada no catálogo. Essa evidência é consistente com a utilização dessas variáveis como atributos centrais nos modelos de clusterização e regressão desenvolvidos neste trabalho.

4.3 Experimentos com Modelos de Agrupamento

Esta seção compara os resultados obtidos da execução do K-Means e GMM para tarefa de agrupamento de instâncias *On-demand*. Para ambos os métodos, avaliaram-se configurações com $K \in \{2, \ldots, 20\}$, registrando Silhueta, Davies-Bouldin e a participação do maior *cluster*, conforme o protocolo da Metodologia. O K-Means apresentou desempenho consistentemente superior ao GMM na faixa analisada. A escolha de $K = 10$ resultou de análise comparativa entre os três indicadores, buscando equilíbrio entre qualidade geométrica dos agrupamentos e distribuição amostral, sem fixar limiares numéricos absolutos.

4.3.1 Modelo K-Means

Para cada valor de K, o modelo foi treinado e avaliado pelos índices da Seção 3.3.4. A Tabela 12 resume os resultados da varredura.

Tabela 12 – Métricas internas do K-Means para diferentes valores de K

K	**Silhueta**	**Davies-Bouldin**	**Maior cl. (%)**
2	0,852	0,597	99,3
3	0,849	0,525	99,1
4	0,486	0,963	85,6
5	0,402	0,897	52,1
6	0,433	0,918	46,3
7	0,422	0,808	47,1
8	0,446	0,766	39,1
9	0,459	0,775	35,7
10	**0,503**	**0,736**	**27,1**
11	0,510	0,737	27,1
12	0,522	0,808	25,6
13	0,523	0,732	25,6
14	0,547	0,716	25,2
15	0,543	0,686	25,6
16	0,550	0,700	19,4
17	0,553	0,672	22,2
18	0,570	0,684	19,4
19	0,568	0,644	19,4
20	0,582	0,643	19,4

Fonte: Elaborado pelo autor.

Em $K = 2$ e $K = 3$, a Silhueta é elevada (0,852 e 0,849), porém mais de 99% das instâncias permanecem no mesmo *cluster*. Essas configurações foram descartadas por segmentação degenerada, independentemente da Silhueta. Entre $K = 4$ e $K = 9$, as métricas evoluem gradualmente, mas a concentração no maior grupo permanece elevada ou a combinação Silhueta/Davies-Bouldin ainda é inferior à observada em valores maiores de K. A partir de $K = 10$ observa-se melhora conjunta nos três indicadores, com Silhueta de 0,503, Davies-Bouldin de 0,736 e maior *cluster* com 27,1% do catálogo. Configurações superiores ($K = 11$ a $K = 20$) incrementam marginalmente Silhueta e Davies-Bouldin, por exemplo, $K = 20$ atinge Silhueta de 0,582 e Davies-Bouldin de 0,643, porém aumentam o número de grupos e a fragmentação interpretativa dos perfis. Adotou-se $K = 10$ por representar o menor valor em que qualidade geométrica e balanceamento amostral convergem de forma satisfatória.

Além disso, a Figura 14 apresenta o gráfico da inércia, permitindo identificar o número de clusters pelo método do cotovelo. Observa-se que a inflexão da curva ocorre entre K=9 e K=10, reforçando a adoção de K=10 como o número de clusters utilizado neste trabalho.

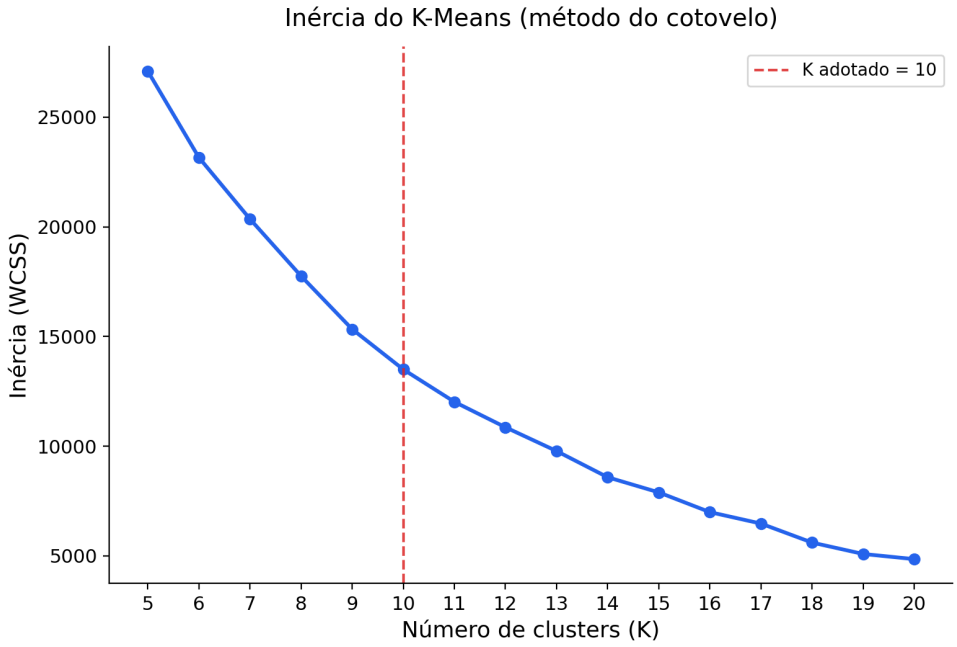


Figura 14 – Inércia e Método do cotovelo

4.3.2 Modelo de Mistura de Gaussianas (GMM)

O GMM foi submetido ao mesmo protocolo do K-Means. A Tabela 13 apresenta os resultados para cada K.

Tabela 13 – Métricas internas do GMM para diferentes valores de K

K	**Silhueta**	**Davies-Bouldin**	**Maior cl. (%)**
2	0,653	1,716	92,2
3	0,377	1,596	51,8
4	0,388	1,220	51,0
5	0,375	1,255	41,3
6	0,384	1,182	41,3
7	0,404	1,482	39,7
8	0,428	1,174	28,5
9	0,431	0,933	28,4
10	0,283	2,126	21,5
11	0,295	1,872	21,6
12	0,292	3,082	21,5
13	0,283	2,525	21,5
14	0,302	1,314	21,5
15	0,135	1,354	21,5
16	0,297	1,482	21,5
17	0,243	1,836	21,5
18	0,316	1,501	21,5
19	−0,044	1,573	21,5
20	0,276	1,823	21,5

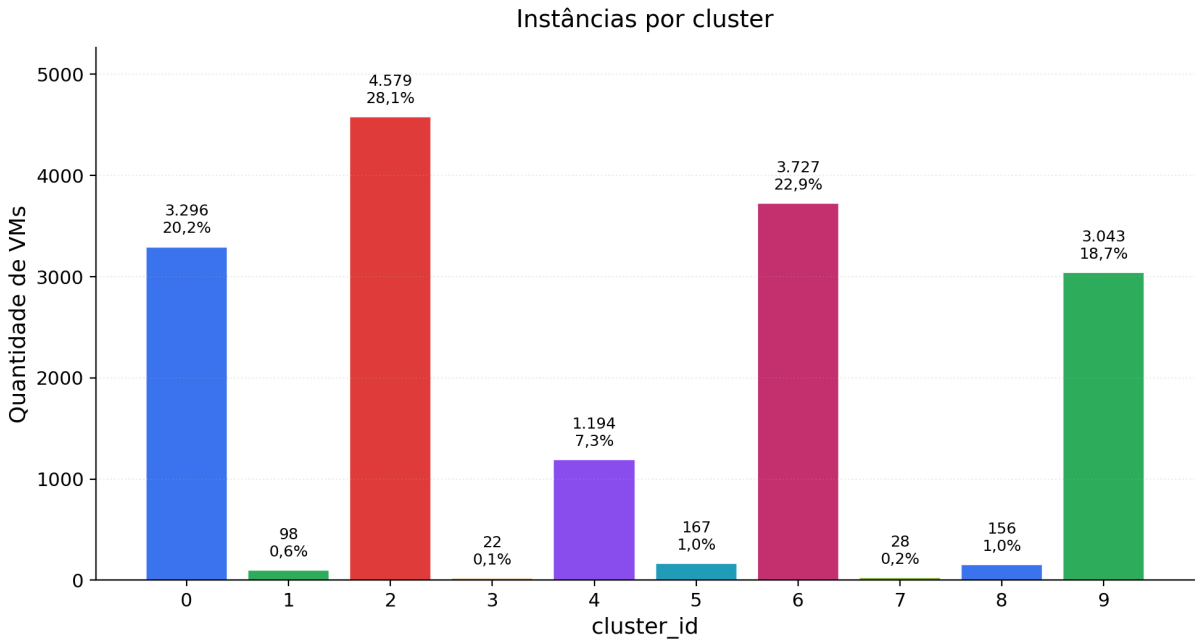
Fonte: Elaborado pelo autor.

O GMM não reproduziu a qualidade geométrica observada no K-Means com o mesmo vetor de entrada. Em $K = 2$, a Silhueta alcança 0,653, porém 92,2% das instâncias concentram-se no maior componente e o Davies-Bouldin é 1,716, padrão semelhante ao K-Means para K muito baixo. Entre $K = 3$ e $K = 9$, embora a participação do maior componente reduza, Silhueta e Davies-Bouldin permanecem inferiores aos do K-Means nos mesmos valores de K. Para $K = 10$ adotado no K-Means, o GMM registra Silhueta de 0,283 e Davies-Bouldin de 2,126. Valores maiores de K não revertem o quadro. O GMM atribui probabilidade de pertencimento a mais de um componente, mas essa flexibilidade não se traduziu em separação superior neste catálogo padronizado. Diante do desempenho comparativo, o GMM permaneceu apenas como referência e a segmentação final adotou K-Means com $K = 10$.

4.3.3 Análise do Agrupamento do Modelo Escolhido

Tendo como base os resultados obtidos, após a escolha do K-Means como modelo base, os resultados da tarefa de agrupamento foram analisados. A Figura 15 apresenta a distribuição das instâncias analisadas entre os dez *clusters*.

Figura 15 – Distribuição de instâncias por cluster



Fonte: Elaborado pelo autor.

Observa-se forte concentração em quatro grupos. Os *clusters* 0, 2, 6 e 9 reúnem 14.645 instâncias, equivalentes a aproximadamente 89% do catálogo utilizado na clusterização. Os demais grupos representam faixas de maior porte ou nichos especializados, com participação baixa participação individual, evidenciando a predominância de configurações de determinados tipos de instâncias no catálogo.

A Tabela 14 resume os dez grupos formados e suas especificações técnicas. Para evitar que valores extremos dominem a interpretação, as colunas de vCPUs e RAM mostram a faixa interquartil (P25 até P75), com a mediana indicada entre parênteses. O campo GPU apresenta a mediana e o maior valor observado no grupo.

Tabela 14 – Perfil técnico dos dez grupos

Cl.	%	SO	Perfil técnico	vCPUs P25–P75 (med.)	RAM GB P25–P75 (med.)	GPU med.–máx.	$/h med.
2	28,1	Linux	Núcleo de entrada, RAM baixa	4–48 (16)	16–128 (48)	0–2	0,89
0	20,2	Windows	Núcleo de entrada, RAM média-baixa	4–48 (16)	16–128 (64)	0–2	1,33
6	22,9	Linux	Núcleo médio, mais RAM	8–48 (16)	64–384 (160)	0–2	1,97
9	18,7	Windows	Núcleo médio, mais RAM	8–48 (20)	64–512 (160)	0–2	2,85
4	7,3	Misto	Grande porte, muitos vCPUs	128–192 (160)	640–1536 (1024)	0–0	14,76
8	1,0	Misto	Alto porte com RAM elevada	8–64 (32)	256–2048 (976)	0–0	6,67
5	1,0	Misto	GPU moderada	48–148 (96)	250–852 (488)	4–8	9,46
3	0,1	Misto	GPU alta densidade	72–128 (96)	512–768 (732)	16–16	14,40
1	0,6	Misto	Ultra escala	416–448 (416)	6144–8192 (6656)	0–0	81,90
7	0,2	Misto	Ultra escala máxima	448–896 (672)	18432–24576 (21504)	0–0	218,40

Fonte: Elaborado pelo autor.

Esses grupos separam as instâncias principalmente por sistema operacional, vCPU por quantidade de memória. Quatro agrupamentos concentram quase todo o catálogo juntos, os *clusters* 2, 0, 6 e 9 reúnem cerca de 89% das instâncias On-Demand.

O *cluster* 2 reúne máquinas Linux de entrada, com mediana de 48 GB de RAM. O *cluster* 0 concentra máquinas Windows de entrada, com mediana de 64 GB. Nos *clusters* 6 e 9, o porte sobe para o segmento médio, onde ambos apresentam mediana de 160 GB de RAM, diferenciando-se pelo sistema operacional. Nesses quatro grupos, a mediana de vCPUs fica entre 16 e 20, o que indica que a separação principal não ocorre pela escala de processamento, mas pelo sistema operacional e pela densidade de memória.

Fora desse núcleo, aparecem agrupamentos de maior porte e uso mais especializado. O *cluster* 4 agrupa instâncias com muitos vCPUs e memória mediana em torno de 1 TB. O *cluster* 8 concentra máquinas com RAM elevada, em torno de 976 GB, porém com menos vCPUs. Juntos, esses dois grupos representam cerca de 8% do catálogo e funcionam como faixa de transição entre o núcleo comum e os segmentos mais específicos.

As configurações com GPU ficam nos *clusters* 5 e 3. O primeiro corresponde a GPU moderada, com mediana de quatro placas e até oito no grupo; o segundo, embora pequeno, concentra instâncias de alta densidade com 16 GPUs.

Por fim, os *clusters* 1 e 7 formam a cauda de ultra escala. O *cluster* 1 reúne instâncias com centenas de vCPUs e cerca de 6,6 TB de RAM; o *cluster* 7 concentra as maiores configurações do catálogo, com mediana de 672 vCPUs e cerca de 21 TB de RAM.

4.4 Experimentos com Modelos de Regressão

Esta seção apresenta os resultados obtidos na modelagem preditiva de preços das instâncias *On-Demand*. Inicialmente, são comparados os modelos de Regressão Linear com Ridge, Random Forest e XGBoost por meio das métricas de avaliação definidas na metodologia. Em seguida, o modelo selecionado é analisado em maior detalhe, considerando seu desempenho nos conjuntos de treino e teste, a aderência entre preços reais e previstos e a importância das variáveis utilizadas na predição.

4.4.1 Comparação entre Regressão Linear, Random Forest e XGBoost

A Tabela 15 apresenta os resultados obtidos pelos três modelos avaliados para predição do preço horário de instâncias *On-Demand*.

Tabela 15 – Comparação de regressores no conjunto de teste

Modelo	**MAE**	**RMSE**	R^2 **teste**	R^2 **CV**	**DP CV**
Regressão Linear (Ridge)	1,571	3,768	0,932	0,923	0,008
Random Forest	1,280	3,212	0,951	0,934	0,011
XGBoost	**0,747**	**2,717**	**0,965**	**0,936**	**0,024**

Fonte: Elaborado pelo autor.

Os resultados demonstram que todos os modelos foram capazes de capturar a maior parte da variabilidade dos preços observados, apresentando coeficientes de determinação superiores a 0,93. A Regressão Linear com regularização Ridge obteve $R^2 = 0,932$ e MAE de 1,571 USD/h, indicando que uma parcela significativa da formação de preços pode ser explicada por relações aproximadamente lineares entre atributos de capacidade computacional, como quantidade de vCPUs, memória RAM e presença de GPU.

A utilização do Random Forest resultou em melhoria consistente do desempenho preditivo. O modelo reduziu o MAE para 1,280 USD/h e elevou o coeficiente de determinação para 0,951, evidenciando a presença de não linearidades e interações entre variáveis que não são adequadamente capturadas por modelos lineares.

O melhor desempenho foi obtido pelo XGBoost, que alcançou MAE de 0,747 USD/h, RMSE de 2,717 USD/h e $R^2 = 0,965$. Em relação ao Random Forest, observou-se redução de aproximadamente 41,7% no erro absoluto médio, demonstrando maior capacidade de modelar a estrutura complexa de precificação presente no catálogo consolidado.

A validação cruzada reforça a robustez dos resultados. Os valores médios de R^2 permaneceram próximos aos observados no conjunto de teste para os três modelos, sugerindo boa capacidade de generalização. Embora o XGBoost apresente maior variabilidade entre as partições ($DP = 0,024$), seu desempenho superior em todas as métricas de erro indica que o ganho de precisão compensa essa diferença.

4.4.2 Análise do erro por faixa de preço

Apesar do MAE global de 0,747 USD/h obtido pelo XGBoost, essa métrica representa uma média agregada sobre instâncias com características e preços bastante distintos. Como a distribuição dos preços apresenta forte assimetria à direita, torna-se importante avaliar como o erro se distribui ao longo das diferentes faixas de preço.

A Tabela 16 apresenta a distribuição do erro absoluto do modelo por intervalo de preço. Observa-se que as instâncias com preços inferiores a 2,00 USD/h apresentam erros médios reduzidos, variando entre 0,10 e 0,25 USD/h. Nessas faixas, que representam aproximadamente metade das observações do conjunto de teste, o erro permanece significativamente abaixo do valor médio global.

Tabela 16 – Distribuição do erro absoluto do XGBoost por faixa de preço

Faixa de preço (USD/h)	**% instâncias**	**MAE (USD/h)**	**% erro abs. total**
$< 0,50$	25,3	0,10	3,4
0,50 – 2,00	25,6	0,25	8,4
2,00 – 10,00	38,0	0,67	33,9
$> 10,00$	11,2	3,64	54,3
Global	100,0	**0,747**	100,0

Fonte: Elaborado pelo autor.

Por outro lado, as instâncias mais caras concentram a maior parcela do erro absoluto acumulado. As configurações com preço superior a 10,00 USD/h representam apenas 11,2% das observações, mas respondem por 54,3% do erro absoluto total. Quando combinadas com a faixa entre 2,00 e 10,00 USD/h, essas instâncias passam a concentrar aproximadamente 88% de todo o erro observado no conjunto de teste.

Esse comportamento é compatível com a distribuição de preços do catálogo, caracterizada por uma cauda longa composta por instâncias de alta capacidade computacional. Nesses casos, pequenas diferenças relativas entre valor real e valor previsto podem gerar erros absolutos significativamente maiores, mesmo quando a previsão permanece próxima do preço efetivamente praticado.

Os resultados indicam que o modelo apresenta maior precisão justamente nas faixas de preço mais representativas do catálogo, mantendo desempenho consistente para instâncias de pequeno e médio porte, que correspondem à maior parte das opções disponíveis aos usuários.

4.4.3 Seleção do modelo final

A comparação entre os modelos evidencia um ganho progressivo de desempenho à medida que a capacidade de representar relações não lineares aumenta. Enquanto a Regressão Linear fornece uma aproximação inicial satisfatória, os métodos baseados em árvores

conseguem capturar padrões mais complexos presentes na estrutura de precificação dos provedores de nuvem.

Considerando conjuntamente as métricas de erro, os coeficientes de determinação e os resultados da validação cruzada, o XGBoost apresentou o melhor desempenho global entre os modelos avaliados. O algoritmo alcançou a menor taxa de erro, preservando elevada capacidade de generalização e apresentando comportamento consistente em diferentes faixas de preço.

Dessa forma, o XGBoost foi selecionado como modelo final para estimativa do preço esperado das instâncias e para as análises de auditoria de preços apresentadas nas próximas seções.

4.4.4 Análise do modelo selecionado

Por conta do melhor desempenho em capturar a relação entre as variáveis de entrada, o XGBoost foi analisado em maiores detalhes. A Tabela 17 apresenta as métricas obtidas nos conjuntos de treino e teste.

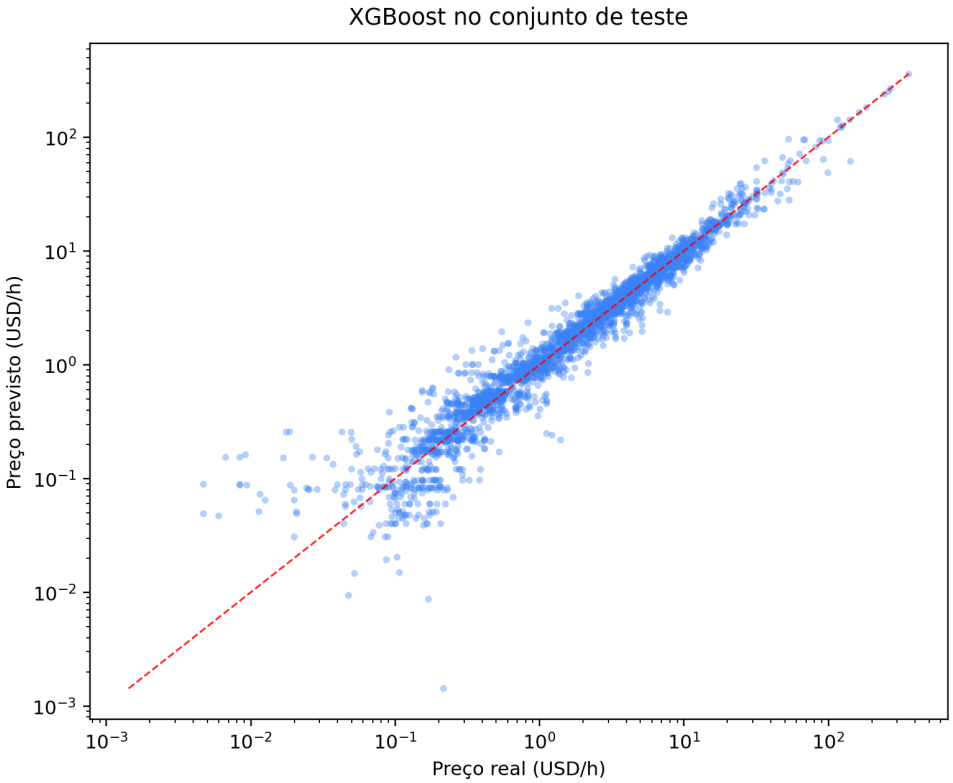
Tabela 17 – Desempenho do XGBoost por conjunto (USD/h)

Conjunto	**MAE (USD/h)**	**RMSE (USD/h)**	R^2
Treino 80%	0,641	2,015	0,979
Teste 20%	0,747	2,717	0,965

Fonte: Elaborado pelo autor.

O desempenho no teste permaneceu próximo ao observado no treino. O R^2 passou de 0,979 para 0,965, enquanto o MAE aumentou de 0,641 para 0,747 USD/h. Essa diferença indica perda moderada entre treino e teste, sem evidência de sobreajuste severo. O aumento do RMSE, de 2,015 para 2,717 USD/h, mostra que os maiores desvios estão associados a erros pontuais em instâncias de preço mais alto, o que é coerente com a distribuição assimétrica do catálogo. A Figura 16 compara os preços reais e previstos no conjunto de teste em escala logarítmica para visualizar simultaneamente instâncias de baixo custo e configurações de maior porte. A maior parte dos pontos acompanha a reta de predição perfeita, indicando boa aderência geral do modelo ao padrão de preços observado.

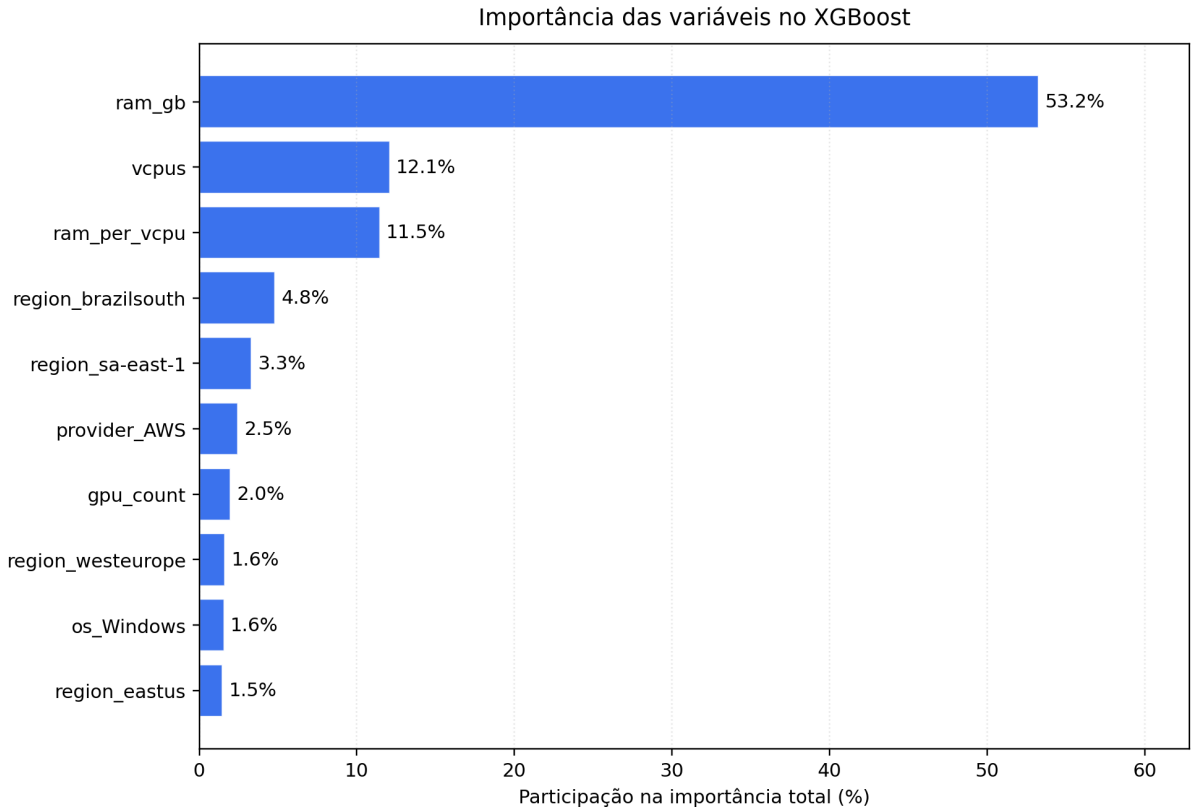
Figura 16 – XGBoost para preços reais versus previstos no conjunto de teste (escala logarítmica)



Fonte: Elaborado pelo autor.

A Figura 17 apresenta as dez variáveis de maior importância no XGBoost. A variável `ram_gb` respondeu por 53,2% da importância total, seguida de `vcpus`, com 12,1% e a proporção entre essas duas variáveis `ram_per_vcpu` com 11,5%. Em seguida aparecem atributos regionais e contextuais, como `region_brazilsouth`, `ram_per_vcpu`, `region_sa-east-1`, `provider_AWS`, `gpu_count`, sistema operacional.

Figura 17 – Dez variáveis de maior importância na predição de preço para XGBoost



Fonte: Elaborado pelo autor.

4.5 Aplicação da solução em cenários representativos

Esta seção exemplifica, com instâncias reais do conjunto de teste, como a combinação K-Means, XGBoost e *Value Ratio* apoiaria decisões em uma interface de comparação. Cada cenário segue a mesma estrutura de fluxo da aplicação, tabela de resultados e interpretação dos indicadores.

4.5.1 Cenário 1: auditoria de preço no mesmo perfil técnico

Considere a instância `Standard_D4ads_v6` da Azure com região provisionada em "brazilsouth", configurada com Linux, 4 vCPUs e 16 GB de memória RAM. O objetivo é avaliar se o preço praticado está consistente com o padrão aprendido pelo modelo e identificar alternativas funcionalmente equivalentes dentro do mesmo *cluster*.

O K-Means identifica e classifica a instância no *cluster* 2 (núcleo Linux de entrada). O XGBoost estima $\hat{P} = 0{,}325$ USD/h, enquanto o preço observado é $P_{\text{real}} = 0{,}366$ USD/h, resultando em VR $= 1{,}13$.

Tabela 18 – Cenário 1: auditoria de preço e alternativas no *cluster* 2

Papel	**Instância**	**Região**	**vCPU**	**RAM**	P_{real}	$\hat{P}$	**VR**
Analisada	`Standard_D4ads_v6`	`brazilsouth`	4	16 GB	0,366	0,325	1,13
Alternativa	`Standard_D4s_v6`	`brazilsouth`	4	16 GB	0,321	0,325	0,99
Alternativa	`Standard_NV4as_v4`	`eastus`	4	16 GB	0,233	0,232	1,00

Fonte: Elaborado pelo autor com instâncias do conjunto de teste.

A instância analisada apresenta preço real de 0,366 USD/h e estimativa de 0,325 USD/h, resultando em VR = 1,13. Isso indica que a VM está, potencialmente, cerca de 13% acima do preço de referência para configurações equivalentes, sinalizando margem para revisão de contratação ou troca de SKU, sem configurar um desvio extremo. A comparação com alternativas do *cluster* 2 evidencia o valor prático do pipeline. Na mesma região (`brazilsouth`), a `Standard_D4s_v6` mantém 4 vCPUs e 16 GB de RAM, com preço de 0,321 USD/h e VR = 0,99, ou seja, alinhada ao previsto pelo modelo. A diferença entre as duas SKUs Azure na mesma localidade mostra que o alerta de preço decorre da instância escolhida, e não apenas do provedor ou da região. Por fim, a `Standard_NV4as_v4` em `eastus` confirma a equivalência técnica do agrupamento hava visto a mesma capacidade (4/16), VR de 1,00 e menor custo horário (0,233 USD/h). O indicador próximo de 1 sugere comportamento consistente com o *cluster* e a variação de preço absoluto reflete, principalmente, efeito regional. Em uma interface, esse resultado poderia ser exibido como alerta moderado na VM analisada, acompanhado de alternativas equivalentes ordenadas por preço e VR, permitindo decidir entre manter a configuração, trocar SKU ou avaliar outra região com base em evidência quantitativa.

4.5.2 Cenário 2: busca por alternativas equivalentes

O segundo cenário simula a escolha de uma VM Linux com 8 vCPUs e 64 GB de RAM. O usuário informa os requisitos de capacidade, o sistema identifica o *cluster* compatível e recupera do catálogo instâncias tecnicamente afins ofertadas por diferentes provedores, exibindo o preço de cada candidato. A Tabela 19 apresenta, para cada provedor, a opção de menor preço *On-Demand* no conjunto de teste dentro desse agrupamento.

Tabela 19 – Cenário 2: alternativas 8 vCPU / 64 GB / Linux no *cluster* 6

Provedor	**Instância**	**Região**	**vCPU**	**RAM**	P_{real} **(USD/h)**
GCP	`e2-highmem-8`	`us-east-1`	8	64 GB	0,380
Azure	`Standard_E8_v5`	`eastus`	8	64 GB	0,504
AWS	`r7g.2xlarge`	`us-west-2`	8	64 GB	0,428

Fonte: Elaborado pelo autor com instâncias do conjunto de teste.

As três instâncias compartilham características técnicas equivalentes, incluindo 8 vCPUs, 64 GB de RAM, sistema operacional Linux e ausência de GPU, além de estarem agrupadas no *cluster* 6. Essa padronização assegura comparabilidade direta entre os recursos analisados, uma vez que o processo de clusterização reduz o espaço de busca a um conjunto de instâncias com capacidade computacional homogênea. Dessa forma, a análise de preços não depende de correspondência exata de nomenclatura ou família entre provedores, mas sim da equivalência funcional inferida pelo modelo.

A comparação evidencia diferenças relevantes de custo entre provedores para configurações tecnicamente equivalentes. A instância da GCP (`e2-highmem-8`) apresentou o menor preço (0,380 USD/h), seguida pela AWS (`r7g.2xlarge`) com 0,428 USD/h e pela Azure (`Standard_E8_v5`) com 0,504 USD/h. A diferença entre a opção mais barata e a mais cara foi de aproximadamente 32%.

Esse cenário complementa o Cenário 1 ao demonstrar o uso da clusterização para descoberta de alternativas equivalentes entre provedores. Ao restringir a comparação a instâncias com características técnicas semelhantes, o sistema permite identificar potenciais oportunidades de redução de custos sem comprometer os requisitos computacionais da aplicação.

5 Conclusões e trabalhos futuros

Conclui-se que a abordagem proposta atingiu o objetivo de construir uma visão unificada de instâncias de máquinas virtuais em ambientes multi-cloud e demonstrar como técnicas de aprendizado de máquina podem apoiar sua análise. A integração entre o *pipeline* de consolidação de dados, a clusterização de perfis técnicos e a modelagem de preços permitiu criar mecanismos capazes de comparar instâncias de diferentes provedores sob uma mesma perspectiva. Os resultados obtidos indicam que essa estratégia é viável para apoiar processos de seleção, análise e recomendação de máquinas virtuais, constituindo uma base sólida para futuras aplicações de apoio à decisão em ambientes multi-cloud.

Como desdobramentos deste trabalho, destaca-se inicialmente a evolução do esquema conceitual proposto para uma aplicação operacional. Uma interface interativa permitiria que usuários informassem requisitos de capacidade, sistema operacional, região e orçamento, recebendo recomendações fundamentadas nos modelos desenvolvidos de maneira prática. Além da comparação de preços absolutos, a ferramenta poderia explorar a visão de custo benefício e os perfis de clusterização para fornecer uma visão mais abrangente do posicionamento relativo das instâncias disponíveis.

Outra oportunidade de evolução consiste na ampliação do catálogo consolidado, com a inclusão de novas regiões, famílias de instâncias e modalidades de contratação. Em particular, a incorporação de instâncias Spot permitiria estender as análises para cenários que envolvem não apenas custo, mas também volatilidade de preços e risco de interrupção. Além disso, a utilização de métricas observadas de desempenho, como *benchmarks* de processamento, armazenamento e rede, poderia complementar as especificações disponibilizadas pelos provedores, tornando as comparações entre instâncias mais representativas do desempenho efetivamente percebido pelas aplicações.

Apesar dos resultados alcançados, algumas limitações devem ser consideradas. A base de dados foi construída a partir das APIs e catálogos públicos disponíveis durante o período de coleta, estando sujeita a diferenças de cobertura, atualizações de nomenclatura e alterações futuras nas ofertas dos provedores. O catálogo consolidado também apresenta menor representatividade do GCP em comparação com AWS e Azure, o que limita algumas análises comparativas.

Além disso, os modelos foram treinados exclusivamente sobre instâncias *On-Demand*, não contemplando a dinâmica de preços das instâncias *Spot*. Da mesma forma, a noção de similaridade utilizada na clusterização baseia-se em atributos de catálogo, como vCPUs, memória, GPU, sistema operacional e região, não incorporando medições reais de desempenho. Consequentemente, a equivalência identificada entre instâncias deve ser interpretada como uma aproximação baseada em especificações técnicas, e não como garantia de comportamento idêntico para cargas de trabalho específicas.

Referências

Amazon Web Services. **AWS Pricing — Amazon EC2 Pricing**. 2024.

ARBELAITZ, O. et al. An extensive comparative study of cluster validity indices. **Pattern Recognition**, v. 46, n. 1, p. 243–256, 2013.

BREIMAN, L. Random forests. **Machine Learning**, 2001.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: **ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. [S.l.: s.n.], 2016.

Cloudorado. **Cloudorado: Cloud Server Comparison**. 2026.

CRN News. **Cloud Market Share Q1 2026: AWS, Microsoft, Google Battling In AI Era**. 2026. Disponível em: <https://www.crn.com/news/cloud/2026/cloud-market-share-q1-2026-aws-microsoft-google-battling-in-ai-era>. Acesso em: 10 mar. 2026.

DAVIES, D. L.; BOULDIN, D. W. A cluster separation measure. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 1979.

ELHABBASH, A. et al. Cloud brokerage: A systematic survey. **arXiv preprint arXiv:1805.09018**, 2018. Disponível em: <https://arxiv.org/abs/1805.09018>. Acesso em: 6 jun. 2026.

FinOps Foundation. **What is FinOps? The Operating Model for the Cloud**. 2024.

Fortune Business Insights. **Cloud Computing Market Size, Share & Industry Analysis**. 2024. Disponível em: <https://www.fortunebusinessinsights.com/cloud-computing-market-102697>. Acesso em: 10 mar. 2026.

GOODARZY, S. et al. Resource management in cloud computing using machine learning: A survey. **Journal of Network and Computer Applications**, v. 151, p. 102490, 2020.

Google Cloud. **Google Cloud Compute Engine Pricing**. 2024.

HALKIDI, M.; BATISTAKIS, Y.; VAZIRGIANNIS, M. On clustering validation techniques. **Journal of Intelligent Information Systems**, v. 17, n. 2–3, p. 107–145, 2001.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning**. [S.l.: s.n.], 2009.

Infracost. **Infracost: Cloud Cost Estimates for Terraform**. 2026.

KAUFMAN, L.; ROUSSEEUW, P. J. **Finding Groups in Data: An Introduction to Cluster Analysis**. Hoboken: Wiley, 1990.

LI, A. et al. Cloudcmp: Comparing public cloud providers. In: **ACM Internet Measurement Conference**. [S.l.: s.n.], 2010.

LI, Z. et al. Cloud pricing models: A survey and taxonomy. In: IEEE. **IEEE International Conference on Services Computing**. [S.l.], 2016. p. 50–57.

LUCCA, G. C. de. **Construção de uma Base de Dados Multi-Cloud e Aplicação de Aprendizado de Máquina para Análise Comparativa de Recursos de Computação em Nuvem**. GitHub, 2026. Disponível em: <https://github.com/GabrieldeLucca1/TCC2-Gabriel>. Acesso em: 09 jun. 2026.

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In: **Berkeley Symposium on Mathematical Statistics and Probability**. [S.l.: s.n.], 1967.

MARTINO, B. D. Applications portability and services interoperability among multiple clouds. **IEEE Cloud Computing**, v. 1, n. 1, p. 74–81, 2014.

MELL, P.; GRANCE, T. **The NIST Definition of Cloud Computing**. [S.l.], 2011.

Microsoft. **Azure Retail Prices API**. 2026. Disponível em: <https://learn.microsoft.com/en-us/rest/api/cost-management/retail-prices/azure-retail-prices>. Acesso em: 6 jun. 2026.

Microsoft Azure. **Azure Virtual Machines Pricing**. 2024.

OSTERMANN, S. et al. A performance analysis of ec2 cloud computing services for scientific computing. **Grid Computing**, 2010.

ROUSSEEUW, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. **Journal of Computational and Applied Mathematics**, 1987.