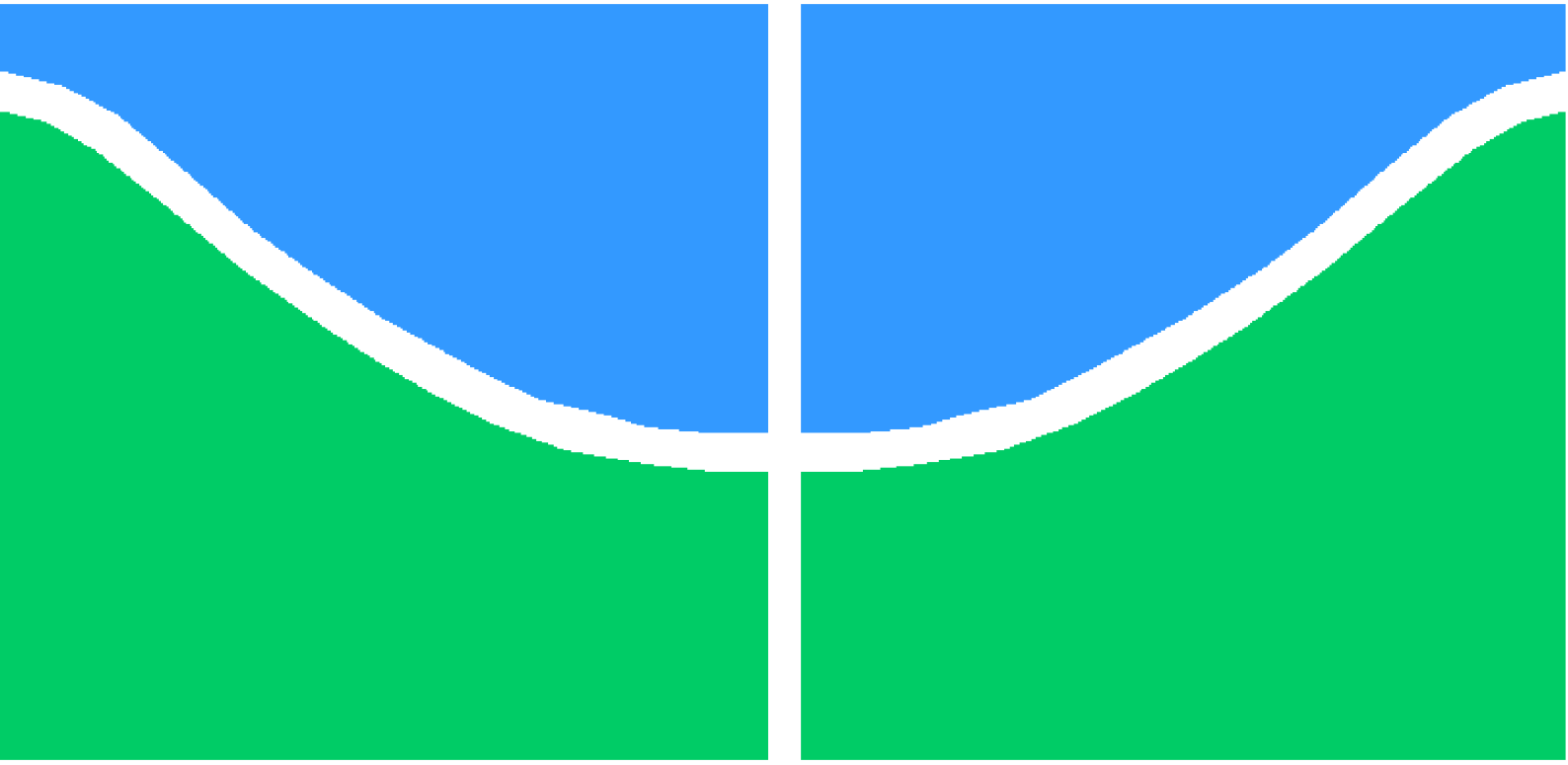


Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Predição de custo de execução de código em funções Lambda

Autores: Lucas Ganda Carvalho e Wictor Bastos Girardi

Orientador: Prof. Dr. Daniel Sundfeld Lima

Brasília, DF

2023

Lucas Ganda Carvalho e Wictor Bastos Girardi

Predição de custo de execução de código em funções Lambda

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Daniel Sundfeld Lima

Brasília, DF

2023

Lucas Ganda Carvalho e Wictor Bastos Girardi
Predição de custo de execução de código em funções Lambda/ Lucas Ganda Carvalho e Wictor Bastos Girardi. – Brasília, DF, 2023-
72 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Daniel Sundfeld Lima

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2023.

1. AWS Lambda. 2. Computação em Nuvem. 3. Previsão de execução 4. Universidade de Brasília. 5. Faculdade UnB Gama. 6. Predição de custo de execução de código em funções Lambda

CDU 02:141:005.6

Lucas Ganda Carvalho e Wictor Bastos Girardi

Predição de custo de execução de código em funções Lambda

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Prof. Dr. Daniel Sundfeld Lima
Orientador

Prof. Dr. Bruno César Ribas
Convidado 1

Prof. Dr. John Lenon Cardoso Gardenghi
Convidado 2

Brasília, DF
2023

Resumo

A computação em nuvem é uma arquitetura distribuída que permite a virtualização de recursos computacionais, configuráveis e escaláveis, sob demanda e em larga escala. Recentemente, os provedores passaram a oferecer uma grande quantidade de serviços de computação sem servidor. Neles, é possível criar e executar aplicações ou códigos sem a necessidade de fazer o gerenciamento de servidores, como é o exemplo da AWS Lambda, que permite a execução de funções e abstrai do usuário a gerência das demais tarefas, permitindo-o focar no desenvolvimento, além de cobrar somente pelos recursos utilizados, tomando como base para o cálculo do custo o número de vezes e o tempo de duração da execução do código. Neste trabalho, propomos um método que possibilita a previsão de custos de funções, no ambiente da AWS Lambda, baseado em diferentes parâmetros. Antever este custo necessita da previsão do tempo de execução. Para isso, utilizamos e avaliamos o desempenho de 4 técnicas: Regressão Linear, Regressão Polinomial, Floresta Aleatória e Árvore de Decisão. Utilizando essa predição de tempo, desenvolvemos uma ferramenta web com uma interface intuitiva e simples que permite ao usuário entrar com os dados de determinada função e receber o valor aproximado do custo de execução da função. Os resultados mostram uma previsão com até 99.76% de precisão no tempo dos casos, dessa forma o custo foi antevisto com uma baixa taxa de erro.

Palavras-chaves: computação em nuvem, previsão de execução de código, computação sem servidor, aws lambda, redes neurais.

Abstract

Cloud computing is a distributed architecture that allows the virtualization of computing resources, configurable and scalable, on demand and on a large scale. Recently, providers have started to offer a large amount of serverless computing services. In them, it is possible to create and run applications or codes without the need to manage servers, as is the example of AWS Lambda, which allows the execution of functions and abstracts the user from managing other tasks, allowing him to focus on development, in addition to charging only for the resources used, based on the number of times and duration of code execution as a basis for calculating the cost. In this work, we propose a method that allows the prediction of function costs, in the AWS Lambda environment, based on different parameters. Anticipating this cost requires forecasting the execution time. For this, we used and evaluated the performance of 4 techniques: Linear Regression, Polynomial Regression, Random Forest and Decision Tree. Using this time prediction, we developed a web tool with an intuitive and simple interface that allows the user to enter data for a given function and receive the approximate value of the cost of executing the function. The results show a prediction with up to 99.76% accuracy in the time of the cases, in this way the cost was foreseen with a low error rate.

Key-words: cloud computing. serverless computing. code execution prediction. aws lambda.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

A computação em nuvem é um modelo que permite o acesso sob demanda, por meio da rede, a recursos computacionais configuráveis(MELL; GRANCE et al., 2011), possibilitando o rápido provisionamento e liberação desses recursos com o mínimo de esforço de gerência. Esses serviços possuem diversas características, tais como elasticidade e escalabilidade ágeis, possibilitando o aumento ou diminuição dos recursos para lidar com uma carga de trabalho. Oferece também outras vantagens, tais como o provisionamento automático sem a necessidade de interação humana com o provedor na nuvem.

A computação sem servidor (AWS, 2022b) abrange um conjunto de recursos disponíveis nos provedores de computação em nuvem, nessa modalidade, o provedor assume a responsabilidade pelo gerenciamento dos servidores, permitindo que o cliente se concentre nas funcionalidades disponibilizadas. Um dos serviços que ilustram esse modelo é o AWS Lambda (AWS, 2022a) que oferece processamento sem servidor. Nele o cliente envia uma função e determina os detalhes de execução, intervalos de tempo fixos ou eventos específicos, como o envio de arquivos por parte dos usuários.

Existem ainda outras soluções sem servidor que possibilitam o armazenamento de dados, por exemplo, o Amazon Simple Storage Service (Amazon S3) que dispensa a necessidade de provisionar discos em máquinas virtuais, enviando e armazenando os arquivos em *buckets*, contêineres de objetos criados pelos próprios clientes.

Ao utilizar a computação em nuvem, são aplicadas cobranças com base no tipo de serviço, tempo de uso, tamanho e memória. Alguns serviços possuem custos fixos e previsíveis, por exemplo, ao utilizar um serviço de armazenamento de arquivos, o custo é determinado pelo tamanho dos dados. Já outros serviços possuem diferentes tipos de valores variáveis associados, como o preço das instâncias EC2 *spot* que depende da oferta e demanda nos *data centers* do provedor.

Diversos estudos foram propostos para prever os custos na computação em nuvem, uma vez que a previsão desses valores não é trivial. Esses trabalhos utilizam métodos estatísticos (MISHRA; KESARWANI; YADAV, 2019) como a regressão linear, que objetiva estimar o valor de uma variável y dado o valor de outra variável x, aprendizado de máquina com redes neurais (AGARWAL; MISHRA; YADAV, 2017), deep-learning (ALTHEIABAT et al., 2018) e floresta aleatória regressiva (KHANDELWAL; CHATURVEDI; GUPTA, 2017), sendo que nestes citados a previsão é feita para custos de instâncias *spot*.

1.1 Objetivos

A previsão de custo em computação em nuvem tem se tornado popular, utilizando métodos de aprendizagem de máquina, estatísticos e metaheurísticos. No entanto, a maior parte das pesquisas sobre previsão de custo está focada no estudo das instâncias de máquinas virtuais.

O presente Trabalho de Conclusão de Curso objetiva propor e avaliar métodos para a previsão de custos de funções Lambda na AWS. O custo das funções Lambda possui vários componentes, sendo o tempo de execução o mais imprevisível. Para isso, serão investigadas diferentes abordagens de previsão de custo de execução e serão aplicadas a funções lambda.

1.1.1 Objetivos Específicos

- Desenvolver um método de previsão de tempo de execução utilizando regressão linear;
- Desenvolver um método de previsão de códigos utilizando aprendizado de máquina supervisionado Floresta Aleatória;
- Prever o tempo de execução da função na nuvem da AWS;
- Prever o custo de execução, baseado no tempo, memória, região e outros fatores.

1.2 Organização

O presente trabalho está dividido em 7 capítulos:

- Capítulo 1 contém uma introdução e apresentação sucinta do trabalho, além dos objetivos;
- Capítulo 2 apresenta a computação em nuvem e seus principais serviços;
- Capítulo 3 apresenta inteligência artificial e métodos para a previsão de tempo de execução de um código;
- Capítulo 4 apresenta a revisão do estado da arte de previsão de custos e tempo de execução na computação local e em nuvem;
- Capítulo 5 apresenta a proposta de trabalho, incluindo tecnologias e ferramentas utilizadas.
- Capítulo 6 apresenta os resultados obtidos e o que foi desenvolvido no trabalho;

- Capítulo 7 é a conclusão, apresentando a síntese do trabalho e as considerações finais.

2 Computação em nuvem

A computação em nuvem é definida como um paradigma de computação distribuída para recursos computacionais em larga escala. Esse padrão possui características que o diferenciam da arquitetura tradicional, como a capacidade de provisionar recursos de forma dinâmica sob demanda, o uso de tecnologia de virtualização, e a alta elasticidade e escalabilidade (MELL; GRANCE et al., 2011)..

2.1 História e Evolução da Computação em Nuvem

Em seu surgimento, a Internet era restrita ao ambiente militar através da ARPAnet, no entanto começou a se expandir através de universidades e repartições públicas na década de 80, a partir da adoção do modelo de referência TCP/IP, protocolo de comunicação entre computadores em rede (TANENBAUM, 2003). Combinando esta tecnologia com a tecnologia Ethernet, protocolo de conexão de rede local, iniciou-se a popularização dos computadores pessoais e consequentemente do modelo cliente-servidor, dividido em hospedeiro, denominado servidor, que atende as requisições de diversos outros hospedeiros, denominados clientes (KUROSE; ROSS, 2014).

No final da década de 90, dois conceitos começam a se popularizar na área da tecnologia, ambos modelos distribuídos baseados na necessidade de compartilhar e dividir recursos computacionais, sendo eles o de computação em grade e o P2P. O primeiro objetiva aproveitar a ociosidade dos ciclos de processamentos em ambientes heterogêneos, tendo como foco comunidade científica e de engenheiros (FOSTER; KESSELMAN, 2003). O segundo é uma arquitetura na qual cada ponto da rede funciona como cliente e servidor de forma auto-organizada e sem um controle central (SCHOLLMEIER, 2001).

A computação em nuvem se tornou um conceito popular a partir dos anos 2000 quando empresas começaram a trocar servidores físicos por serviços em nuvem, levando em consideração benefícios como redução de custos, aumento da eficiência de operação e simplificação de processos(PORTELLA, 2021).

2.2 Conceitos Principais

Inicialmente, a computação em nuvem tinha como objetivo processar aplicações e armazenar dados fora do ambiente corporativo, em estruturas como *datacenters*, para otimizar recursos (VERAS, 2015).

Uma definição popular, apresenta a ideia de que a computação em nuvem é um

conjunto de recursos virtualizados de fácil uso e acesso, como hardwares, plataformas de desenvolvimento ou serviços, que podem ser dinamicamente reconfigurados de forma a permitir a utilização otimizada. Ademais, esse conjunto de recursos é comumente utilizado em um modelo *pay-per-use* com garantias oferecidas pelo provedor do serviço (VAQUERO et al., 2008).

A computação em nuvem possui características que as definem e diferenciam da computação tradicional, entre elas (MELL; GRANCE et al., 2011):

- **Serviço autossuficiente sob demanda**: O usuário pode requerer enquanto for necessário o fornecimento de recursos computacionais como tempo de processamento e armazenamento de rede, sem precisar interagir com cada provedor de serviço;
- **Acesso abrangente à rede**: Os recursos estão passíveis de serem utilizados através da Internet e por mecanismos padronizados que promovem acesso de dispositivos heterogêneos, como celulares ou workstations;
- **Centralização de recursos**: Os recursos computacionais fornecidos pelo provedor são agrupados para servir múltiplos usuários em um modelo multilocação, alocando e re-alocando dinamicamente os diferentes recursos de acordo com a demanda do cliente;
- **Rápida Elasticidade**: Os recursos podem ser liberados e alocados de maneira elástica, geralmente de forma automática, para escalar rapidamente de acordo com a demanda;
- **Serviço calculado**: Otimização de serviço para que haja o máximo de aproveitamento dos recursos disponíveis, a fim de gerar o mínimo de ociosidade.

Além das características que definem a computação em nuvem, ela pode ser dividida em modelos diferentes, entre eles são (MELL; GRANCE et al., 2011):

2.2.1 Modelos de Serviço Tradicionais

Os serviços de computação em nuvem oferecidos pelos provedores podem ser classificados em três tipos de acordo com a responsabilidade de gerenciamento dos servidores pelo cliente (VERAS, 2015):

Software como Serviço (SaaS)

Baseia-se no fornecimento ao consumidor da capacidade de rodar aplicações na nuvem, utilizadas pelos usuários através de diferentes dispositivos, sem a necessidade de fazer a gerência ou controle dos recursos como memória ou armazenamento. São exemplos desses softwares: Gmail, Mailchimp e Dropbox.

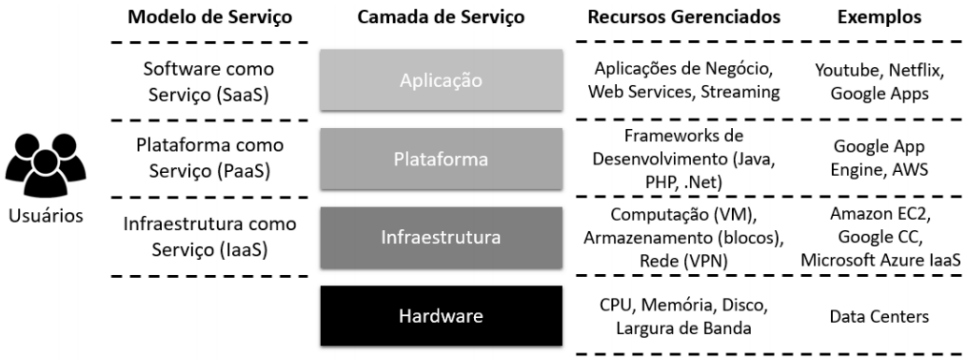


Figura 1 – Camadas da Computação em Nuvem, retirado de (PORTELLA, 2021).

Plataforma como Serviço (PaaS)

Fornece ao consumidor a capacidade de implementar aplicações utilizando infraestrutura da nuvem. Não sendo necessário a gestão de recursos como servidores e armazenamento, embora ainda tenha controle da aplicação implantada e das configurações do ambiente de hospedagem. É possível observar esta tecnologia em serviços Heroku, AWS Beanstalk e Google App Engine.

Infraestrutura como Serviço (IaaS)

Possibilita ao consumidor processamento, armazenamento, redes e outros recursos nos quais é possível implementar e rodar softwares. Neste caso o consumidor tem controle de sistemas operacionais, armazenamento, aplicações implantadas e possivelmente componentes de redes. A infraestrutura em nuvem pode ser observada na AWS, Microsoft Azure e Digital Ocean (VERAS, 2015).

Tudo como Serviço (EaaS)

Esse é um modelo de serviço mais recente que os citados acima, entretanto, com a rápida e ampla difusão da arquitetura em nuvem tem se tornado popular. Essa modalidade fornece uma gama de soluções em nuvem como comunicação, redes, entre outras. Pode-se verificar esta ideia sendo utilizada em conceitos ao exemplo de Rede como Serviço, oferecimento de tecnologias em arquitetura de rede, Monitoramento como Serviço, uma forma de facilitar a implantação de soluções de monitoramento para diversos recursos, Testes como Serviço, uma forma de facilitar as operações de testes através da sua terceirização.

2.2.2 Modelos de Implantação

As nuvens não são iguais, na verdade possuem uma quantidade de serviços e poder computacionais muito diferentes. De acordo com a forma que são implementadas, podem

ser classificadas em (VERAS, 2015):

Nuvem Privada

Estrutura é oferecida apenas para membros de uma determinada organização ou empresa, o serviço pode ser propriedade, gerenciado e operado pela própria entidade, terceiros ou combinação de ambos (MELL; GRANCE et al., 2011).

Nuvem pública

Os recursos são oferecidos para uso do público geral de uma determinada organização ou empresa, podendo ser propriedade, gerenciada e operada por uma entidade empresarial, acadêmica, governamental ou combinação (MELL; GRANCE et al., 2011). Comumente é disponibilizado no modelo *pay-per-use* por organizações com grande capacidade de processamento e armazenamento (VERAS, 2015).

Nuvem comunitária

Os serviços são fornecidos para uso de uma comunidade específica de organizações que compartilham interesses em comum, podendo ser propriedade, administrada e operada por uma ou mais entidade da comunidade, além de terceiros ou uma combinação de ambos (MELL; GRANCE et al., 2011).

Nuvem híbrida

A infraestrutura é composta de duas ou mais nuvens que podem ser privadas, públicas ou comunitárias permanecendo como entidades únicas, conectadas por tecnologias, padronizadas ou proprietárias, que possibilitam a portabilidade de dados e aplicações (MELL; GRANCE et al., 2011).

2.3 Arquitetura Sem Servidor

Originalmente, a arquitetura em nuvem era utilizada para recriar em máquinas virtuais de baixo nível os ambientes e sistemas utilizados localmente com o intuito de fazer a portabilidade da carga de serviço e reduzir a utilização dos servidores físicos das empresas. Apesar de tirar a necessidade de gerenciamento da infraestrutura física, criou-se a urgência de gerenciar recursos virtualizados. Essa nova realidade gerou um excesso de serviços criando problemas e a necessidade destes gerenciamento, como exemplo (JONAS et al., 2019):

- Redundância de disponibilidade, para que falhas de máquina não derrubem o serviço;

- Distribuição geográfica de cópias redundantes para preservar o serviço em caso de desastre;
- Balanceamento de carga e roteamento de solicitações para utilização eficiente de recursos;
- Escalonamento automático em resposta a mudanças de carga para aumentar ou diminuir o sistema;
- Monitoramento para garantir o funcionamento do serviço;
- Registro para gravar mensagens com a finalidade de encontrar erros ou aumentar a performance;
- Atualizações de sistemas, incluindo correções ou melhorias de segurança;
- Migração para novas instâncias à medida que se tornem disponíveis.

Essas necessidades levaram a Amazon a criar um novo serviço chamado *AWS Lambda*, neste serviço o usuário não possui conhecimento do servidor que está executando seu código e por isso é chamado de computação sem servidor. O oferecimento de funções em nuvem por esse sistema foi responsável pela difusão e popularização da computação sem servidor (JONAS et al., 2019).

A arquitetura sem servidor é um modelo de desenvolvimento característico da computação em nuvem que tem como objetivo a criação e o desenvolvimento de aplicações, sem a necessidade de gerenciar os servidores. Apesar do nome, os servidores ainda são essenciais neste modelo, entretanto são abstraídos dos usuários e do desenvolvimento das funcionalidades (REDHAT, 2022).

Existem três características essenciais que diferenciam esse modelo da arquitetura *Serverful*, as mais utilizadas na nuvem são (JONAS et al., 2019):

- Desacoplamento entre computação e armazenamento: Recursos computacionais escalam e são precificados de forma diferente. Sendo comum o fornecimento de armazenamento por um provedor de nuvem diferente, enquanto a computação é *Stateless*.
- Execução de código sem gerência de recursos: Em vez de solicitar recursos para a execução da aplicação, o usuário fornece determinado código, como uma função, sendo os recursos provisionados automaticamente pelo provedor.
- Pagamento proporcional: Enquanto na arquitetura tradicional há a dimensão usando parâmetros como quantidade e tamanho de máquinas virtuais alocadas, na arquitetura sem servidor o dimensionamento é feito a partir do tempo de execução do código.

Além destas características essenciais, essa arquitetura possui outras características diferentes do modelo tradicional, entre eles estão (JONAS et al., 2019):

- A aplicação não roda continuamente, apenas quando eventos escolhidos pelos usuários são acionados.
- As aplicações não possuem persistência de dados, são *Stateless*, enquanto tradicionalmente podem ou não persistir dados, sendo *Stateful* ou *Stateless*.
- A capacidade em nuvem de armazenamento é bem menor, pois essa arquitetura é utilizada principalmente para recursos que não exigem tanto memória.
- As aplicações possuem um limite de tempo de execução, sendo o máximo de 15 minutos.

2.3.1 Modelos de Serviço com Servidor

Os serviços de computação sem servidor delegam a responsabilidade de implantar os servidores virtuais e discos virtuais, atualizações de segurança e escalonamento para o provedor na nuvem. Geralmente, os serviços utilizados na computação em nuvem sem servidor estão divididos em dois grupos (REDHAT, 2022):

Função como Serviço (FaaS)

Nesta modalidade, o foco é no desenvolvimento de partes modulares do código, funções, sendo uma maneira econômica de implementar microsserviços. É um modelo orientado a eventos, executado em contêineres sem estado. Essas funções gerenciam a lógica e o estado do lado do servidor (VILLAMIZAR et al., 2017). Serviços que são exemplos deste modelo são a Amazon Lambda, Google Cloud Functions, Microsoft Azure Functions e IBM Cloud Functions.

Por serem orientadas a eventos, essas funções são acionadas por gatilhos definidos pelo usuário, como uma requisição HTTP. A escalabilidade desse modelo é facilitada por não ser orientado a recurso (SHAHRAD; BALKIND; WENTZLAFF, 2019).

Alguns usos deste modelo são :

- Processamento de Eventos - O processamento de dados é um dos usos mais conhecidos do modelo orientado a eventos, como é o exemplo da *Netflix*. A empresa envia vídeos para o serviço Amazon S3, que ao recebê-los emite um evento que aciona um gatilho responsável por ativar na Amazon Lambda uma função que realiza o processo de dividir e transcodificar o vídeo em diferentes formatos.

- IoT : Uma tecnologia que tem se popularizado é a da tecnologia da Internet das Coisas, que caracteriza uma rede de dispositivos conectados à internet e agregados de sensores. Estes aparelhos utilizam esse modelo de função como serviço para enviar e receber informações quando acionados, como é o caso da assistente pessoal da Amazon, Alexa, que é ativada quando acionada por comandos de voz dado pelo usuário.

Back-end como serviço (BaaS)

Neste modelo, são externalizadas as funções típicas de um back-end para serviços na nuvem baseados em API de terceiros, dessa forma, desenvolvedores não precisam se preocupar com funcionalidades que podem ser terceirizadas, permitindo que foquem em outros aspectos como o *front-end* ou apenas no desenvolvimento de partes modulares do código, funções, sendo uma maneira econômica de implementar microsserviços(SHAHRAD; BALKIND; WENTZLAFF, 2019).

Funcionalidades que podem ser utilizadas dessa forma:

- Gerenciamento de dados e base de dados;
- Autorização e autenticação de acesso;
- APIs (REST ou GraphQL);
- Notificação de e-mails;
- Notificação de SMS;
- *Push Notification*;
- Integração com redes sociais.
- Serviços de geolocalização.

2.3.2 AWS Lambda

A AWS Lambda é uma plataforma de computação sem servidor orientada a eventos fornecida pela Amazon como parte do Amazon Web Services, permite a execução de código base sem provisionar e/ou gerenciar servidores. A plataforma realiza esta execução em uma infraestrutura de computação de alta disponibilidade e executa automaticamente a administração de recursos de computação, incluindo manutenção de servidor e sistema operacional, provisionamento de capacidade, dimensionamento automático e registro em *log*. Com a Lambda, é possível executar aplicações para diversos tipos de serviço de *back-end*. Sendo necessário apenas o fornecimento do código base em um dos idiomas suportados pelo serviço (AMAZON, 2022).

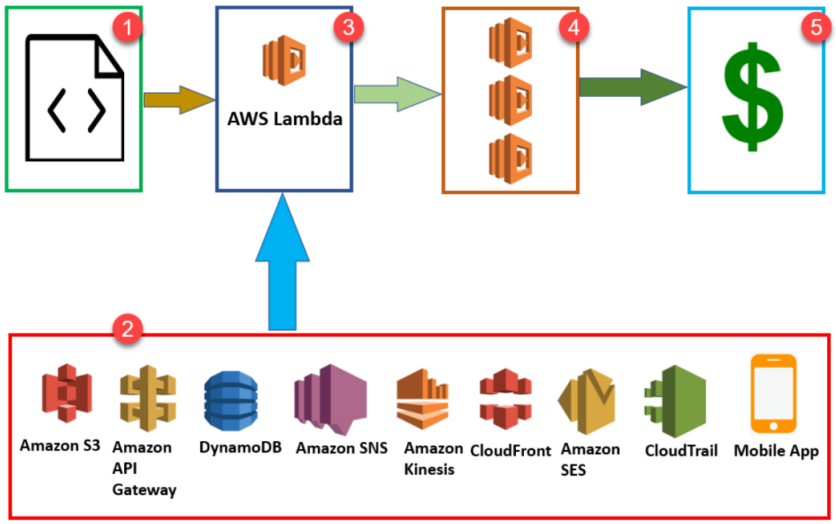


Figura 2 – Diagrama de blocos do AWS Lambda, adaptado de (GURU99, 2022).

A adoção dessa tecnologia elimina a necessidade de infraestrutura e serviços de computação tradicionais, reduzindo o custo e a complexidade de suas operações de tecnologia da informação, o que torna o desenvolvimento mais rápido e o dimensionamento mais fácil.

Funcionalidades Lambda

A AWS Lambda possui funcionalidades que auxiliam os ciclos de desenvolvimento de aplicações escaláveis, seguras e extensíveis (AMAZON, 2022):

- **Controles de simultaneidade e dimensionamento**: Oferecem controle refinado sobre o dimensionamento e a capacidade de resposta de seus aplicativos de produção, sendo estes limites de simultaneidade provisionados ou não;
- **Funções definidas como imagens de contêiner**: Cria a possibilidade do uso de ferramentas, fluxos de trabalho e dependências de imagem de contêineres permitindo criar, testar e implantar funções Lambda;
- **Assinatura de código**: Fornece controles de confiança e integridade que permitem verificar se o código publicado por desenvolvedores está implantado em suas funções da Lambda e inalterado;
- **Extensões Lambda**: Possibilitar o aumento de suas funções, por exemplo, a integração do serviço com ferramentas para monitoramento, observabilidade, segurança e governança;

- **Planos de função**: Um *blueprint* de função fornece um código de exemplo que mostra como usar a Lambda com outros serviços da AWS ou aplicativos de terceiros, incluindo código de amostra e predefinições de configuração de função para ambientes de execução Node.js e Python;
- **Acesso ao banco de dados**: Possibilita o gerenciamento de um conjunto de conexões de banco de dados e retransmição de consultas de uma função por um *proxy*, viabilizando o alcance de altos níveis de simultaneidade sem esgotar as conexões;
- **Acesso a sistemas de arquivos**: Possibilita configurar uma função para montar um sistema de arquivos do Amazon EFS em um diretório local. Com o Amazon EFS, o código de uma função pode acessar e modificar recursos compartilhados com segurança e simultaneidade.

Benefícios do AWS Lambda

AWS Lambda pode ser utilizada para executar Python, Node.js, Java, Go, C# e PowerShell.

Apesar de comum a utilização de gerência de estado em suas aplicações, a Lambda é *Stateless*, ou seja, executa sua função e não persiste nenhum dado, permitindo a criação de uma quantidade de cópias de uma função baseada na taxa de eventos recebidos, o que impulsiona a escalabilidade e o desempenho buscando minimizar os custos.

A falta de estado pode ser tratada com a conservação do estado da aplicação em um banco de dados. Neste caso, é necessário a conexão da função Lambda com outros serviços de armazenamento, podendo ser sem servidor (como o Amazon S3) ou não, como um banco de dados em uma instância EC2.

O Modelo *Pay-as-you-go* cobra o usuário pelo tempo de execução, tendo como base para cobrança intervalos de 100 ms. Esse pagamento por tempo de execução gera incentivo para a utilização de um um código mais eficiente para assim minimizar custos, além disso, é possível ajustar o uso máximo de recursos de cada Lambda, afetando o preço base das execuções de suas funções (SENTINELONE, 2020).

É possível considerar uma função que registra uma autorização do Amazon Cognito, com uma configuração de memória mínima de 128 MB, esta função Node.js realiza a seguinte execução:

- Registro em um *bucket* do AWS S3, no qual a função é executada em menos de 100 ms. Estima-se que a cobrança para aproximadamente 10 milhões de execuções dessa função seja um valor em torno de 2 dólares.
- O AWS Lambda é dimensionado automaticamente para oferecer suporte à taxa de solicitações recebidas sem exigir novas configurações, o provedor na nuvem cria a

ilusão de que não há limites e o desempenho permanece o mesmo quando o número de chamadas aumenta. Entretanto, é necessário que a aplicação seja sem estado para ser dimensionada.

- Os serviços da AWS Lambda são escalonados sob demanda, aumentando recursos de computação, como CPU, RAM e memória quando a demanda é ampliada, ou reduzindo, quando há pouca atividade, este recurso pode escalonar de zero a valores próximos ao infinito voltando ao seu estado inicial quando detectado ociosidade no sistema.

Dentre os benefícios da utilização de funções Lambda destacam-se alta flexibilidade, disponibilidade e economia de custos para a criação de aplicações sem servidor. Vale ressaltar que o modelo *pay-as-you-go* e os recursos de escalonamento automático garantem o pagamento apenas quando os recursos são utilizados.

Cálculo de custos e estimativas

A cobrança da AWS é baseada na utilização do serviço pelo usuário (AWS, 2022a), sendo ele cobrado com base no número de solicitações das funções e no tempo necessário para a execução do código. A Lambda registra uma solicitação sempre que é executada em resposta a uma notificação de evento ou chamada do próprio recurso, incluindo também os testes do console.

A duração de execução das funções é calculada a partir do momento em que o código começa a ser executado até retornar ou terminar de outra maneira, sempre arredondado para o 1ms mais próximo. O preço também depende da quantidade de memória que é alocada para a função (AWS, 2022a).

$$GB/s = NumeroExecuções * (TamanhoMemória/1024) * (DuraçãoExecução/1024) \tag{2.1}$$

Em cada função, é possível definir o tamanho máximo da memória e o tempo limite de execução, em geral, são executadas apenas quando acionadas, assim o cálculo do custo é baseado em (AWS, 2022a):

- Número de execuções e ou invocações;
- Tamanho da memória (entre 128 MB e 10 GB);
- Duração da execução em milissegundos;
- Uso da simultaneidade provisionada do AWS Lambda.

Opções de cobrança

- Nível gratuito: Este nível do AWS Lambda inclui, por mês, um milhão de solicitações gratuitas e 400.000 GB/s de computação, utilizáveis para funções com processadores x86 e Graviton2, em conjunto. Faz-se necessário ressaltar que com o uso do Lambda sob demanda deverá haver a subtração dessas requisições gratuitas da quantidade total utilizada. (AWS, 2022a)

- Lambda sob demanda: Os custos são determinados pelos preços pagos conforme o uso do AWS Lambda.

- Planos de economia do AWS Lambda: São modelos de precificação flexíveis que oferecem preços baixos em troca de um compromisso com uma quantidade consistente de uso por um período de um ou três anos, possibilitando gerar uma economia de até 17%.

3 Inteligência Artificial

O primeiro trabalho acadêmico relevante na área de Inteligência Artificial foi desenvolvido na década de 50 pelo matemático e cientista da computação Alan Turing, que propôs um teste para aferir a capacidade de computadores de exibir comportamento indistinguível ao de seres humanos, portanto inteligente (TURING, 1950). Para obter a aprovação no teste de turing, a máquina deve possuir algumas características :

- Capacidade de se comunicar com sucesso, por exemplo em inglês.
- Capacidade de armazenar informação.
- Utilizar as informações armazenadas para responder perguntas e fazer inferência
- Adaptação a novas circunstâncias, além de detectar e extrapolar padrões.

A definição varia entre os diferentes autores, no entanto as definições mais utilizadas são :

- É a ciência e engenharia de fazer máquinas inteligentes, especialmente programas de computador inteligentes (MCCARTHY et al., 2007).
- É o estudo de agentes inteligentes, ou seja, entidades que percebem seu ambiente e tomam ações que maximizam suas chances de sucesso (RUSSELL, 2010).

O primeiro programa de computador que utiliza Inteligência Artificial foi escrito pelo cientista da computação Christopher Strachey, em 1951, o programa jogava o jogo de damas em um tempo considerado razoável. Na década de 50 as aplicações desenvolvidas envolviam desde jogos de xadrez até problemas matemáticos complexos, como é o exemplo do General Problem Solver, programa desenvolvido em 1957 por Herbert Simon, J. C. Shaw e Allen Newell e que tinha como objetivo solucionar questões complexas de lógica e matemática.

Na década de 60, foram desenvolvidos diversos softwares nessa área, especialmente os que utilizavam processamento de linguagem natural, como o ELIZA desenvolvido entre os anos de 1964 e 1966 pelo MIT e o Systran desenvolvido pelo governo dos Estados Unidos. Com a evolução tecnológica que possibilitou um aumento da capacidade de processamento e de dados disponíveis, programas que utilizam Inteligência Artificial se tornaram mais populares, além de capazes de resolver problemas muito mais complexos.

A Área de Inteligência Artificial é bastante extensa e pode ser utilizada em diversos tipos de aplicações, algumas delas são:

- Jogos;
- Reconhecimento de Fala;
- Linguagem Natural;
- Visão computacional;
- Sistemas Especializados.

3.1 Previsão de Execução

A análise preditiva é um termo genérico para a ação de aplicar técnicas analíticas avançadas de dados para responder perguntas ou resolver problemas, não sendo uma tecnologia em si, mas, um conjunto de ferramentas. Geralmente descrita como um ramo da análise que utiliza modelos estatísticos, mineração de dados e técnicas de aprendizado de máquina (BOSE, 2009).

Essa área possui diversos usos, como encontrar padrões, prever eventos, detectar riscos e levantar oportunidades, tem sido muito utilizada atualmente por empresas para otimizar processos, aumentar lucros e diminuir custos.

Os modelos preditivos são divididos principalmente em 3 técnicas diferentes que utilizam abordagens e métodos distintos.

3.2 Aprendizado de Máquina

O Aprendizado de máquina é um campo multidisciplinar cujo foco é imitar em máquinas, o processo humano de aprendizagem, ou seja, a partir de dados ou algoritmos, reconhecer padrões, gerar informações, aprender a partir destas informações e aumentar a acurácia desse processo. Esta área possui como base inúmeros outros ramos do conhecimento, como inteligência artificial, ciência da computação, teoria da informação, filosofia e outras (ALZUBI; NAYYAR; KUMAR, 2018).

A aprendizagem de máquina aborda diferentes problemas, dentre eles a classificação e regressão se destacam. O primeiro é o processo de trabalhar com modelos ou funções que consigam separar os dados em categorias, ou seja, valores discretos. Na classificação, usando como base alguns parâmetros fornecidos na entrada, os dados são divididos em diferentes grupos. A regressão, no entanto, tem como finalidade encontrar modelos ou funções que sejam capazes de dividir os dados em valores reais, não discretos, ela também é utilizada para identificar o movimento de distribuição de acordo com dados históricos.

Os algoritmos utilizados nesse método são divididos em três categorias principais, aprendizado supervisionado, não-supervisionado e por reforço (BONACCORSO, 2017).

O aprendizado supervisionado, também conhecido como classificação e regressão, utiliza exemplos de entrada da mesma categoria da informação pretendida na saída durante a fase de treinamento, o aprendizado não supervisionado recebe dados que não são da mesma classe dos resultados esperados na saída, dessa forma tendo que detectar padrões não explícitos (SATHYA; ABRAHAM et al., 2013).

A aprendizagem por reforço se baseia na ideia de que o agente executa ações em um ambiente, recebe uma recompensa ou penalidade com base nessas ações e utiliza essas informações para desenvolver estratégias de tomada de decisões com o objetivo de otimizar as recompensas. A Figura 3 apresenta as principais categorias dos algoritmos de aprendizado de máquina e exemplos de problemas nos quais cada categoria é aplicada.

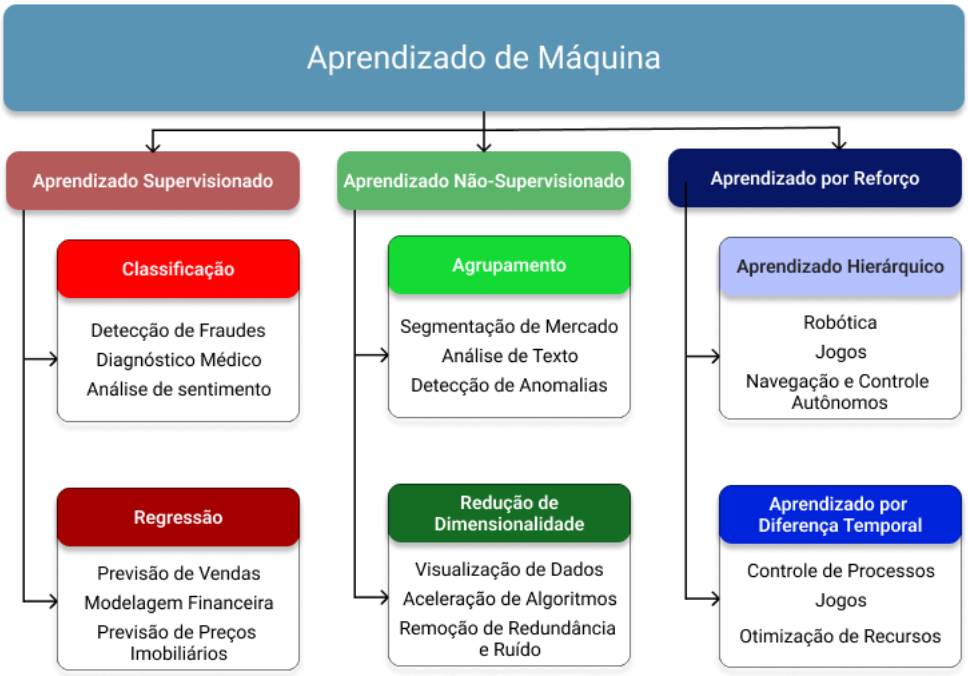


Figura 3 – Categorias dos algoritmos de aprendizado de máquina.

Um exemplo de algoritmo que se enquadra nessa classificação é o de Árvore de Decisões representado pela Figura 4, sendo este uma representação em forma de árvore de escolhas e seus resultados, onde cada árvore é formada por nós e galhos, onde um nó representa atributos do grupo a ser classificado e os galhos representam valores que o nó pode vir a tomar (MAHESH, 2020).

O algoritmo Floresta Aleatória, que possui como finalidade a classificação e regressão, é uma evolução da árvore de decisões pois esta utiliza um conjunto de árvores. Neste método, para a tarefa de classificação, a saída é a classe selecionada pela maior quantidade de árvores, e para a regressão, o retorno é a previsão média das árvores individuais. Este

algoritmo corrige o problema de sobreajuste em relação a árvore de decisões, além de ter uma acurácia maior. No entanto, é um algoritmo altamente complexo, consumindo uma quantidade de tempo maior e dependendo de mais recursos.

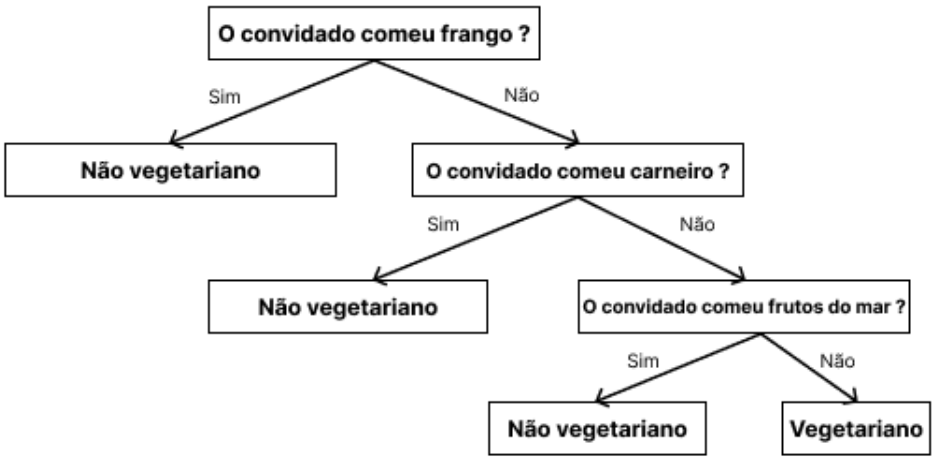


Figura 4 – Algoritmo de Árvore de Decisão, adaptado de (MAHESH, 2020).

Redes Neurais é um conjunto de algoritmos, que tem ganhado popularidade, com o objetivo de reconhecer relações fundamentais em um conjunto de dados por meio de um processo que tenta imitar a forma de funcionamento do cérebro humano. As redes neurais podem se adaptar a mudanças, como as de entrada, para que haja o melhor resultado possível sem a necessidade de redesenhar a saída (MAHESH, 2020).

Máquinas de vetor de suporte é um conjunto de métodos de aprendizagem supervisionados para analisar dados e reconhecer padrões, utilizados para classificação e análise de regressão. O padrão desse modelo é a partir de uma entrada, predizer em qual das duas possíveis classes a entrada pertence, dessa forma este algoritmo é um classificador linear binário não probabilístico. Ademais, é possível performar classificação não linear através do método Kernel, por meio da adição de uma margem entre as classes (MAHESH, 2020).

Esse método possui diferentes vantagens, entre elas:

- Identificação facilmente padrões e tendências;
- Relativa autonomia;
- Melhoramento contínuo;
- Capacidade de lidar com grande quantidade de dados;
- Tratamento de dados variados e multi-dimensionais.

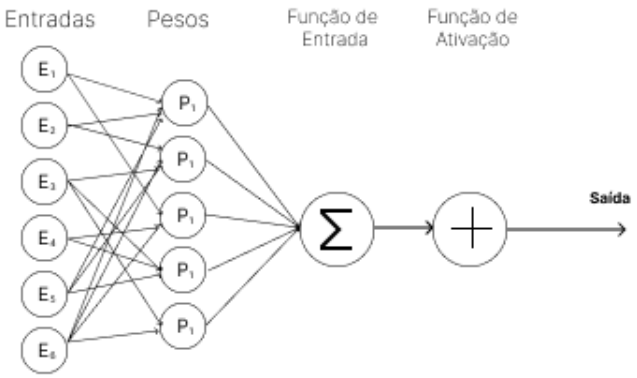


Figura 5 – Redes Neurais, adaptado de (MAHESH, 2020).

3.3 Métodos Estatísticos

A estatística é o campo da matemática responsável por coletar, classificar, organizar, analisar e interpretar dados numéricos. Os métodos estatísticos são constantemente utilizados para tomar decisões e fazer conclusões baseadas em dados.

No campo da modelagem estatística, a análise de regressão é um conjunto de processos, ferramentas e métodos que permite estimar a relação entre duas ou mais variáveis. Por padrão, a investigação se baseia na busca de um efeito causal de uma variável na outra, como o efeito da oferta na demanda ou de uma vacina no sistema imunológico (SYKES, 1993).

Os métodos da modelagem estatística são divididos em três grupos. Os paramétricos assumem que uma população pode ser adequadamente modelada por uma distribuição e que estas possuem um número fixo e finito de parâmetros, os não-paramétricos não assumem forma pré-determinada, ou seja, o modelo é construído de acordo com inferência dos dados, além disso assumem o modelo e a natureza dos parâmetros como flexíveis, os semi-parametrizados combinam ambos os métodos quanto a natureza dos parâmetros e são utilizados quando ambos os modelos não se encaixam completamente.

A análise de regressão é mais utilizada no contexto de predição, sendo também empregada em conjunto com o aprendizado de máquina.

A Regressão Linear é um dos métodos mais utilizados e consiste em estimar os dados de uma variável, dado uma outra (MONTGOMERY; PECK; VINING, 2021), um exemplo deste método está ilustrado na Figura 6.

A Regressão Linear Múltipla é um extensão da Regressão Linear, pois com essa técnica é possível correlacionar uma variável Y com um conjunto de variáveis. Em ambos os casos, o uso do termo linear se dá pois, assume-se que o valor a ser estimado está relacionada a uma combinação de termo ou termos (TRANMER; ELLIOT, 2008).

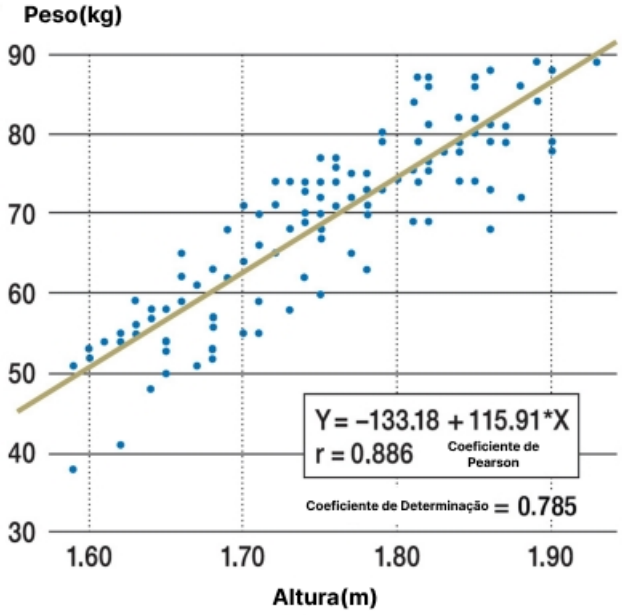


Figura 6 – Regressão linear, adaptado de (SCHNEIDER; HOMMEL; BLETTNER, 2010).

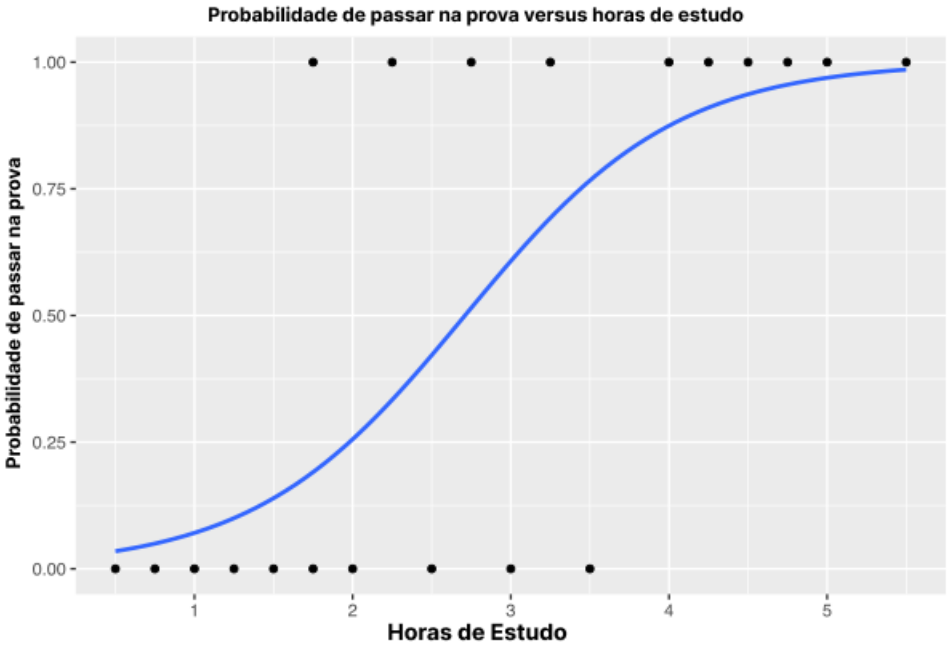


Figura 7 – Regressão logística, adaptado de (LOGISTIC. . . , 2022).

Outra técnica estatística é a Regressão Logística representada na Figura 7, que tem como objetivo estimar a probabilidade do evento de interesse, a partir de um conjunto de combinação de variáveis independentes. Nesse caso, a probabilidade de saída de ocorrência do evento se dá como um valor no intervalo entre 0 e 1 (SPERANDEI, 2014).

Regressão polinomial é uma forma especial de múltipla regressão linear no qual é descrito a relação de uma variável independente x e uma variável dependente y com base

em um polinômio de n-ésimo grau.

Originalmente, os métodos estatísticos tinham como objetivo dividir em grupos e estabelecer inferências, entretanto, começaram a ser utilizados para predição através da regressão e pela capacidade de correlacionar termos, por meio da determinação de graus de associação entre termos. É frequente a utilização em conjunto de técnicas de aprendizado de máquina para predição e previsão.

3.4 Metaheurísticas

Metaheurística é um paradigma computacional de maior abstração usado para resolver de forma sofisticada problemas de otimização através da combinação de diferentes heurísticas. Essa abordagem geralmente é utilizada em casos onde não há certeza do algoritmo mais eficiente ou quando o poder de processamento é limitado (NESMACHNOW, 2014).

Esse método pode apresentar problemas com a variação de tempo, quando envolve funções matemáticas ou um grande volume de dados. Apesar disso e de não garantir a solução ideal, essa abordagem geralmente encontra soluções de boa qualidade em um tempo de execução razoável para uma grande variação de problemas de otimização, tornando-a valiosa (NESMACHNOW, 2014).

Os algoritmos de computação flexíveis foram desenvolvidos para tolerar e até mesmo tirar proveito da existência da falta de soluções exatas para problemas, aumentando a qualidade das soluções existentes, além disso, são capazes de lidar com imprecisões, incertezas, aproximações de erros e verdades parciais.

As técnicas de metaheurísticas utilizam diversas estratégias para encontrar soluções, dentre elas busca local e busca aleatória. A busca local utiliza um único candidato a solução, que é substituído na próxima etapa, geralmente por um mais eficiente e na vizinhança, esta técnica é mais otimizada que a busca aleatória por trabalhar com apenas uma solução por vez. A busca aleatória utiliza um conjunto de múltiplos candidatos a solução por vez, essas soluções são combinadas ou alteradas para gerar um melhor resultado (GUNANTARA; PUTRA, 2019).

3.5 Medidas de erro e precisão

Nos modelos que utilizam regressão, é essencial empregar medidas de erro para avaliar como o modelo se ajusta aos dados observados, permitindo assim quantificar a diferença entre os valores observados e os preditos pelo modelo, possibilitando analisar se o modelo é capaz de explicar adequadamente a variação dos dados. Além disso, essas

medidas auxiliam na comparação entre diferentes modelos para determinar qual apresenta o melhor desempenho e avaliar a eficácia do modelo.

Coeficiente de determinação

O coeficiente de determinação, também conhecido como R^2 Score, é uma medida estatística utilizada para avaliar modelos de regressão. Essa medida indica a proporção da variação dos dados que pode ser explicada pela regressão, variando de 0 a 1. Valores próximos a 0 indicam que o modelo não explica satisfatoriamente a variação dos dados, enquanto valores próximos a 1 indicam que a variação dos dados é explicada de forma precisa.

O cálculo do coeficiente envolve a soma dos quadrados dos resíduos, representado por SQres, e a soma total dos quadrados representa por STqua. O primeiro tem como objetivo avaliar a discrepância dos valores preditos para os estimados, o segundo mede a variabilidade total da variável explicada.

$$R\check{s} = 1 - \frac{SQres}{STqua} \tag{3.1}$$

Erro médio absoluto

O erro médio absoluto é uma medida estatística utilizada em modelos de regressão para avaliar a discrepância entre os valores preditos e os valores observados. Essa medida compara dois valores diferentes do mesmo tipo de dado a partir da média dos erros absolutos entre o valor predito e o valor observado.

O cálculo do erro médio absoluto é descrito pela equação 3.4, em que X_i representa o valor observado e X o valor predito, utiliza-se o módulo de cada erro para que não ocorra a subestimação.

$$\frac{1}{n}\sum_{i=1}^{N}|X_i - X| \tag{3.2}$$

Erro quadrático médio

Esta é uma medida estatística utilizada em modelos de regressão para avaliar a discrepância entre os valores preditos e os valores observados. Diferentemente do erro médio absoluto, este leva em consideração a magnitude dos erros ao elevar as diferenças ao quadrado, destacando e penalizando os erros maiores.

O cálculo é descrito pela equação, na qual Xi representa o valor observado e X o valor predito.

$$\frac{1}{n}\sum_{i=1}^{N}(Xi - X)^2 \tag{3.3}$$

Raiz quadrada do erro médio

Esta é uma medida utilizada na estatística para mensurar a diferença entre amostras, sendo utilizada comumente para avaliar regressões. É uma técnica derivada do erro quadrático médio, porém é preferida em muitos casos por fornecer uma medida na mesma unidade da variável dependente. Essa medida é especialmente útil quando se deseja interpretar o erro médio de forma mais direta.

A raiz quadrada do erro médio é descrita pela fórmula, na qual Xi representa o valor observado e X o valor predito.

$$\sqrt{\frac{1}{n}\sum_{i=1}^{N}(Xi - X)^2} \tag{3.4}$$

O uso isolado da métrica R^2 não comprova a relação estatisticamente significativa entre os atributos de entrada e a saída estimada. Para complementar a avaliação, foram utilizadas outras métricas, como o erro médio absoluto, o erro quadrático médio e a raiz quadrada do erro médio. Essas métricas proporcionam uma análise mais abrangente do desempenho do modelo.

4 Previsão de Custos

A predição de recursos computacionais, especificamente em arquiteturas na nuvem é uma área de pesquisa com grande potencial e envolve diversos aspectos a serem estudados e detalhados. Esse capítulo apresentará um levantamento de pesquisas e estudos relacionados a esta temática, com o objetivo de identificar técnicas e métodos que auxiliem no desenvolvimento de um algoritmo capaz de prever tempo de execução de código em serviços na nuvem. Nos trabalhos levantados serão analisadas partes do problema, técnicas utilizadas e resultados obtidos.

4.1 Previsão de Custo de Computação em Nuvem

A área da computação em nuvem possui diversos trabalhos relacionados à previsão de custo, principalmente de instâncias **On Demand**:

4.1.1 Lee e coautores

Neste trabalho, Lee e coautores enfocam a análise de dados em ambientes de computação em nuvem, com o objetivo de melhorar o desempenho e a relação custo-efetividade de um *cluster*, levando em consideração a heterogeneidade do ambiente e das cargas de trabalho, bem como a equidade na divisão do trabalho quando múltiplos trabalhos compartilham o *cluster*.

A partir desses estudos, é proposta uma abordagem chamada *ProgressShare*, que se baseia no cálculo do progresso de cada tarefa em relação a uma quantidade específica de recursos. Com a aplicação desse método, os processos podem ser divididos de forma a otimizar o desempenho geral (LEE; KATZ, 2011).

4.1.2 Zhang e coautores

Baseando-se na problemática de comparação e seleção de serviços de infraestrutura, ocasionada pela popularização da computação em nuvem, Zhang e co-autores focam em formas de lidar com esse desafio, facilitando o processo de escolher instâncias com o menor custo que respeitem os requisitos da aplicação.

Este trabalho desenvolveu um sistema declarativo chamado *CloudRecomender*, que tem como objetivo sugerir instâncias *on-demand* que atendam a múltiplos critérios, como capacidade de processamento, largura de banda de rede, capacidade de armazenamento e preço. Essas sugestões são geradas com base nas necessidades específicas da empresa,

simplificando assim a comparação de ofertas de serviços de infraestrutura (ZHANG et al., 2012).

4.1.3 Ben-Yehuda e coautores

A partir da análise dos históricos de precificação de oito tipos diferentes de instâncias EC2 em quatro regiões distintas, Ben-Yehuda e coautores realizaram uma engenharia reversa dos preços praticados. Isso se deve ao fato de que, embora a Amazon oferecesse um preço dinâmico para as instâncias, não havia divulgação de como esse valor era determinado. Os pesquisadores constataram que, até 2011, os preços eram baseados em um mecanismo diferente do que era explicitado pelo provedor.

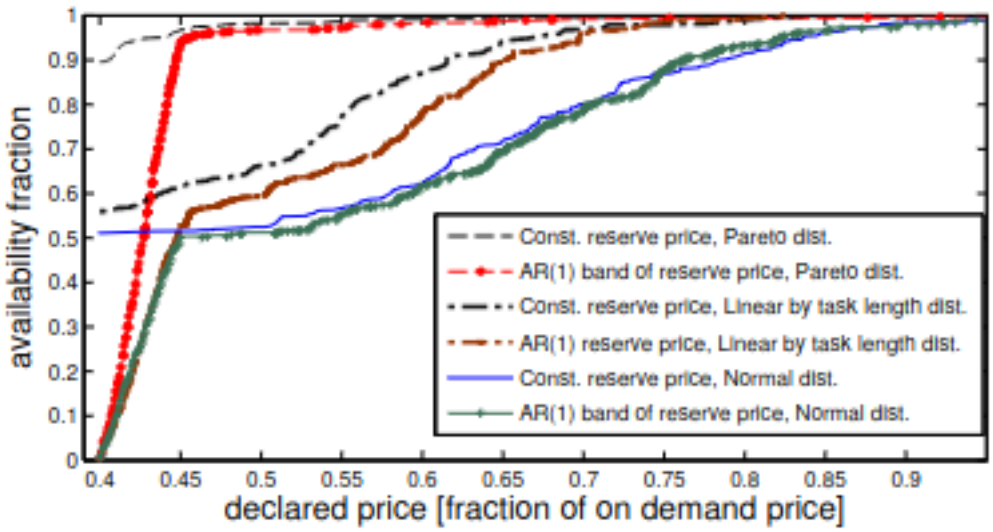


Figura 8 – Probabilidade da variação de preço de acordo com avanço do tempo, retirado de (BEN-YEHUDA et al., 2013).

Contrariamente ao que era divulgado, os autores concluíram que os preços eram gerados aleatoriamente, com base em uma pequena variação de preço, visando evitar a venda das instâncias por um valor muito baixo. No entanto, foi observado que essa abordagem possuía uma brecha que permitia que os clientes obtivessem computação gratuita em determinados períodos, como pode ser observado na Figura 8. Esse método de cálculo foi posteriormente alterado em 2012 para um mecanismo de mercado baseado em leilão (BEN-YEHUDA et al., 2013).

4.1.4 Singh e coautores

Neste trabalho, é analisado que embora o crescimento da computação em nuvem auxilie na redução significativa dos custos envolvidos com infraestrutura, a predição do preço de instância é um desafio para os usuários deste serviço. Dessa forma os autores propõem uma solução de previsão de preço de instâncias *spot* para lances de curto e médio prazo.

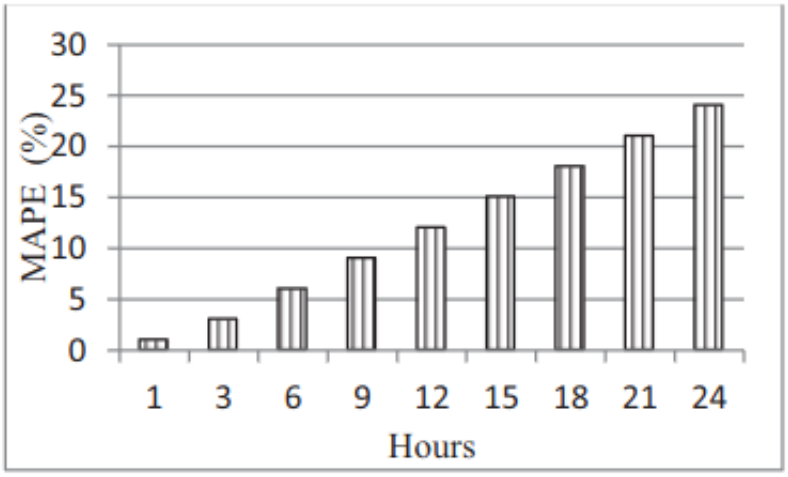


Figura 9 – Erro percentual absoluto médio para previsão de um dia, retirado de (SINGH; DUTTA, 2015).

A solução utiliza um método de otimização que emprega o algoritmo gradiente descendente tradicional. Inicialmente, os dados históricos são normalizados e é calculada a média de preço por hora. Em seguida, o algoritmo calcula os pesos dos parâmetros. Os resultados do estudo foram avaliados utilizando a métrica de erro médio absoluto percentual, obtendo-se um resultado de **9,4%** para o curto prazo e **20%** para o médio prazo. A Figura 9 apresenta os valores do erro médio absoluto percentual para o curto prazo, considerando previsões de até um dia antes. Observa-se que a previsão é mais precisa para o período de 1 hora à frente(SINGH; DUTTA, 2015).

4.1.5 Lucas-Simarro e coautores

Lucas-Simarro e coautores abordam a utilização de uma infraestrutura de múltiplas nuvens,*multi-cloud*, que oferece benefícios como escalabilidade de serviços, maior tolerância a falhas e redução de custos. No entanto, os usuários podem enfrentar dificuldades ao decidir onde implementar seus recursos devido à grande quantidade de informações envolvidas nesse processo.

Para lidar com essa questão, os autores propuseram um algoritmo de escalonamento que visa tomar decisões com base em diferentes mecanismos de precificação. O

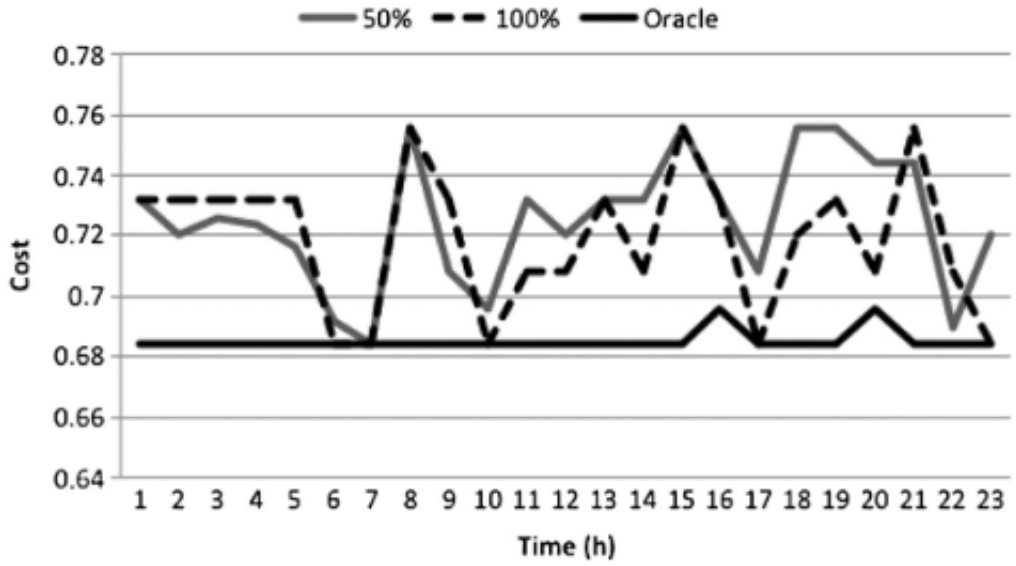


Figura 10 – Comparação de custo percentual absoluto médio para previsão de um dia, retirado de (LUCAS-SIMARRO et al., 2015).

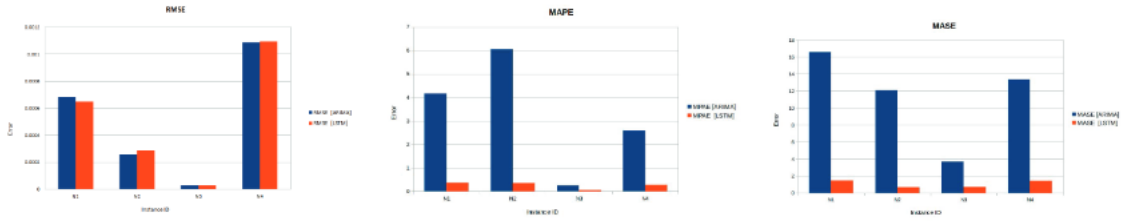
trabalho utiliza dados históricos e emprega técnicas para aproveitar o esquema de preços oferecido pelos provedores de nuvem, permitindo a migração parcial ou total do *cluster* para um provedor mais econômico. Além disso, o trabalho leva em consideração características como desempenho, tamanho mínimo de memória e número de CPUs (LUCAS-SIMARRO et al., 2015). Os autores concluem que os custos estimados com 100% de realocação são um pouco menores em comparação com 50% de realocação, porém significativamente mais altos do que ao utilizar um único provedor, como ilustrado na Figura 10.

4.1.6 Al-Theiabat e coautores

Neste trabalho, os autores destacam a má utilização das instâncias *Spot* devido a problemas relacionados a falhas e complexidades nas licitações. Portanto, é de grande importância ter a capacidade de prever os valores dessas instâncias, permitindo que os clientes se organizem adequadamente.

Para abordar a previsão de preços *Spot*, que é um problema de análise de séries temporais, foram comparados o método estatístico ARIMA e a técnica de redes neurais LSTM de 3 camadas. Os resultados deste estudo indicam que a abordagem baseada em redes neurais supera o modelo estatístico, apresentando o menor erro. Foram utilizadas as métricas de raiz quadrada do erro médio, erro percentual absoluto médio e erro escalado absoluto médio para avaliar o desempenho (AL-THEIABAT et al., 2018). Na Figura 11, na qual as barras azuis correspondem ao ARIMA e as barras laranjas ao LSTM, pode-se

observar que, em relação à raiz quadrada do erro médio, os valores de ambas as técnicas são semelhantes. No entanto, para o erro percentual absoluto médio e o erro escalar absoluto médio, o LSTM demonstra uma maior precisão.



(a) RMSE (b) MAPE (c) MASE

Figura 11 – Comparação de erro entre os métodos ARIMA e LSTM, retirado de (AL-THEIABAT et al., 2018).

4.1.7 Mishra e coautores

Mishra e coautores abordam as instâncias *spot* e estabelecem uma relação entre o baixo custo e a reduzida confiabilidade dessa infraestrutura. A partir dessa perspectiva, é explorada uma possibilidade de estudo para o uso eficiente desses recursos.

PERCENTAGE ERROR IN PREDICTION OF PRICE FOR PARTICULAR INSTANCE TYPE, AVAILABILITY ZONE AND PRODUCT TYPE.

Instance Id	Instance Type	Region	% Error of Particular Product Type			Avg. % Error
			Linux Unix	SUSE Linux	Windows	
I1	c1.medium	us-west-1	2.617	0.372	0.812	1.267
I2	c3.2xlarge	us-west-1	0.337	0.161	0.065	0.187
I3	c3.4xlarge	ap-southeast-1	13.541	8.032	1.575	7.716
I4	m3.medium	eu-west-1	0.498	0.069	0.278	0.281
I5	m3.large	sa-east-1	1.206	0.223	0.152	0.527
I6	m4.10xlarge	us-west-2	7.018	6.067	1.742	4.942

Figura 12 – Tabela de erro do trabalho, retirado de (MISHRA; KESARWANI; YADAV, 2019).

O foco do trabalho está na previsão de curto prazo, com intervalos de segundos ou minutos, utilizando análise probabilística e séries temporais, além do cálculo do erro percentual absoluto médio (MISHRA; KESARWANI; YADAV, 2019). Com base em dados

históricos de diferentes tipos de instâncias em diversas regiões, o objetivo do projeto era alcançar a maior precisão possível. Os resultados obtidos mostraram uma taxa de erro máxima de **7.716%** para a instância 3 e uma taxa de erro mínima de **0.187%** para a instância 2, com uma média geral de erro de **2.486%**, conforme ilustrado na Figura 12.

4.1.8 Li e coautores

Este trabalho objetivou realizar uma pesquisa abrangente sobre a precificação das instâncias *Spot*, analisando dezenas de artigos e explorando os limites e benefícios da abordagem de precificação da AWS. Os autores constataram a ausência de leis ou padrões, destacando a incerteza inerente ao processo de precificação e reforçando a visão de um modelo orientado pelo mercado. Os artigos selecionados foram classificados com base em teorias descritivas, preditivas, explicativas ou prescritivas, cada uma delas apresentando diferentes métodos e ferramentas de precificação.

Esta pesquisa enfatiza as vantagens do modelo de precificação dinâmica *Spot* em comparação com a precificação estática em termos de eficiência e custos. O modelo dinâmico aumenta a lucratividade para o provedor, ao permitir um uso mais eficiente dos recursos computacionais e reduzir a ociosidade. Além disso, essa abordagem também oferece serviços mais acessíveis aos usuários (LI et al., 2016).

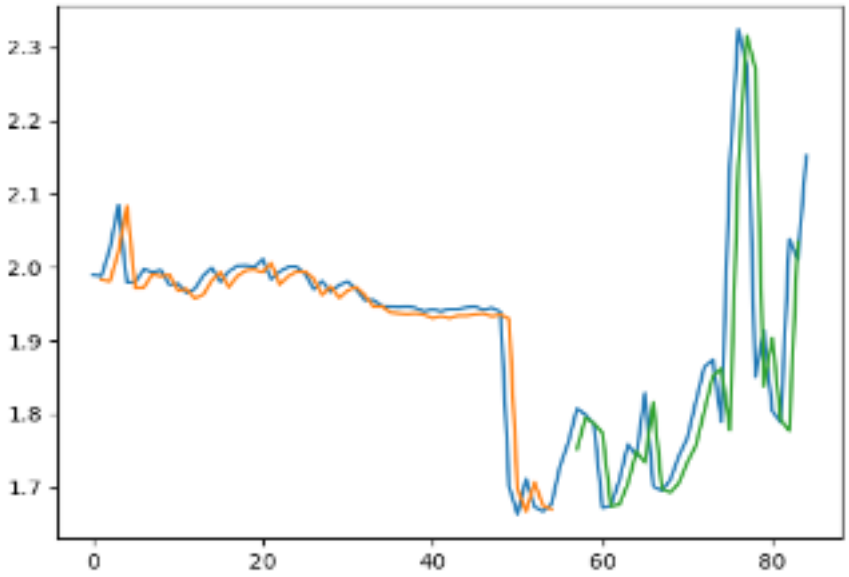


Figura 13 – Resultado da predição para uma das instâncias, retirado de (AGARWAL; MISHRA; YADAV, 2017).

4.1.9 Agarwal e coautores

Agarwal e co autores abordam a questão da previsão de preços *spot* considerando que, se o término de uma instância ocorrer antes da conclusão de uma hora, o usuário é cobrado pelo valor da hora completa, enquanto se o término for feito pela Amazon, o usuário não paga. Isso ressalta a importância de os usuários realizarem previsões de preços antes de fazerem seus lances.

Para abordar esse desafio, os autores utilizam redes neurais para prever os preços dinâmicos com base em dados históricos fornecidos pela Amazon, abrangendo um período de noventa dias. Os pesquisadores obtiveram resultados que demonstram uma taxa de erro percentual variando de **1%** a **8,6%** para a maioria das instâncias analisadas (AGARWAL; MISHRA; YADAV, 2017). Na Figura 13, é possível visualizar o resultado da predição para uma das instâncias, em que a linha azul representa o preço original, a linha laranja representa os dados de treinamento e a linha verde representa os valores preditos.

4.1.10 Khandelwal e coautores

Khandelwal e co autores apresentam um modelo de previsão de custos de instâncias *spot* com o objetivo de auxiliar os usuários na definição de lances. Neste estudo, são comparados diferentes métodos de aprendizado de máquina, incluindo regressão por vetores de suporte, redes neurais recorrentes e floresta aleatória regressiva.

ONE-WEEK-AHEAD MAPE IN DIFFERENT INSTANCE TYPES LINUX/UNIX

Instance Types	MAPE 10%	MAPE 15%
I1	93.79	98.01
I2	81.07	90.91
I3	68.49	88.75
I4	58.92	85.78
I5	77.46	92.32
I6	80.51	91.02
I7	78.93	91.35
I8	83.27	91.56
I9	47.67	70.45
I10	71.52	78.82

Figura 14 – Tabela de erro do trabalho, retirado de (KHANDELWAL; CHATURVEDI; GUPTA, 2017).

O estudo utiliza 10 instâncias de diferentes regiões geográficas com dados históricos de 1 semana. Os resultados indicam que o algoritmo de floresta aleatória apresentou a melhor precisão e menor tempo de treinamento. Obteve-se uma taxa de erro percentual

absoluto médio de **10%** para previsões de um dia e **15%** para previsões de uma semana (KHANDELWAL; CHATURVEDI; GUPTA, 2017), conforme ilustrado na Figura 14.

4.2 Previsão de tempo de execução de código

4.2.1 Yun Park

Um dos pioneiros no estudo de previsão de tempo de execução de código, Yun Park em 1993 utiliza técnicas de análises de caminhos para realizar a previsão do tempo de execução. O estudo consistia em analisar todos os possíveis e prováveis caminhos que um programa poderia percorrer, utilizando uma IDL, e realizar uma análise estática simples baseada em cronometragem de tempo.

Os resultados desse trabalho foram considerados válidos e promissores. A análise dos caminhos foi eficaz, especialmente para programas sequenciais. No caso de programas paralelos, o estudo forneceu algumas previsões confiáveis, embora tenha enfrentado desafios em relação a algumas previsões imprecisas. Esse trabalho demonstrou a viabilidade de previsões seguras e factíveis de tempo de execução para diferentes aplicações.(PARK, 1993).

4.2.2 Miu e coautor

Considerando que muitos algoritmos usados na ciência são computacionalmente caros, Miu e coautor propuseram uma abordagem para estimar o tempo de execução de programas, com o objetivo de fornecer aos usuários da computação científica uma ideia mais precisa dos custos esperados de suas pesquisas.

Neste trabalho, são empregadas técnicas de aprendizado de máquina para aprender modelos de regressão com o objetivo de obter a técnica de maior precisão na previsão do tempo de execução do programa, levando em consideração a quantidade de entradas fornecidas. Os resultados demonstram uma previsão precisa do tempo de execução para algoritmos específicos, como pode ser observado na Figura 15 (MIU; MISSIER, 2012).

4.2.3 Doan e coautores

Com base na premissa de que o tempo de execução pode ser usado para avaliar a eficiência de um algoritmo de interesse e considerando que testar diferentes técnicas pode aumentar o custo dos projetos de mineração de dados, Doan e coautores utilizam a ideia de meta-aprendizado para prever e estimar o tempo de execução de algoritmos de classificação.

Scenario	No. Attributes	Model Builder/features	RAE
Poker Hand	11	M5P_indicators	13.09%
		kNN_indicators	76.23%
		MLP_indicators	15.80%
		LR_static	29.35%
		M5P_counts	10.29%
		kNN_counts	6.21%
		MLP_counts	16.50%
Adult	15	M5P_indicators	16.73%
		kNN_indicators	52.10%
		MLP_indicators	13.92%
		LR_static	25.97%
		M5P_counts	24.55%
		kNN_counts	24.40%
		MLP_counts	27.98%
KDD Cup	40	M5P_indicators	27.60%
		kNN_indicators	34.65%
		MLP_indicators	14.96%
		LR_static	15.43%
		M5P_counts	15.55%
		kNN_counts	16.54%
		MLP_counts	16.71%

Figura 15 – Tabela de resultados do trabalho, retirado de (MIU; MISSIER, 2012).

Neste estudo, o algoritmo que obteve o melhor tempo de execução foi o Splines de Regressão Adaptativa Multivariada, seguido pela Máquina de Vetores de Suporte, ambos com um erro médio absoluto de **1.265%**. Vale ressaltar que este primeiro apresentou um desvio médio absoluto menor. No entanto, a previsão do tempo de execução foi mais desafiadora para outros algoritmos, como a Floresta Aleatória, pois este é projetado para suportar operações paralelas. (DOAN; KALITA, 2017).

4.2.4 Marcos Amaris e coautores

Nesta pesquisa, Marcos Amaris e coautores apresentam um modelo baseado em BSP para prever os tempos de execução de aplicações CUDA em GPUs. O modelo utiliza o número de operações computacionais e acessos à memória da GPU, juntamente com informações adicionais sobre o uso do cache obtidas por meio de criação de perfil. O BSP é um modelo de referência para computação paralela que permite a análise algorítmica de programas em computadores paralelos por meio da modelagem de desempenho. Esse modelo trata a comunicação e a computação como abstrações de um sistema paralelo, levando em consideração as principais propriedades físicas e de otimização das arquiteturas de GPU. A previsão de desempenho é baseada nos custos de comunicação e computação, que são calculados de forma independente.

O modelo foi aplicado para prever o desempenho da multiplicação de matrizes em quatro diferentes níveis de otimização, assim como uma solução de granularidade grosseira para o problema da subsequência máxima. As aplicações foram desenvolvidas em CUDA e executadas em diferentes placas de GPU. Em sua maioria, os resultados obtidos foram aproximadamente 0.8 a 1.2 vezes o tempo de execução previsto pelo modelo (AMARIS; CORDEIRO; CAMARGO, 2015).

4.2.5 Pariwat e coautores

Neste artigo, utilizamos técnicas de aprendizado de máquina para prever o desempenho do notebook Jupyter no JupyterHub. Essa ferramenta atua como um servidor que permite o acesso ao Jupyter sem a necessidade de instalação no computador local. Os serviços são gerenciados em nuvens públicas ou locais, e é fundamental que os administradores dessas nuvens possam estimar com precisão a carga total gerada pelos usuários, a fim de fornecer serviços eficientes e confiáveis.

Performance index	**Description**	**Regression model**		
		Random Forest	**KNN**	**Baseline**
MAPE (%)	**Train dataset**	12.802	11.231	N/A
	Test dataset	9.849	95.057	11.935
MAE (second)	**Test dataset**	13.768	128.907	16.145

Figura 16 – Resultados do trabalho com tabela de erros, retirado de (PRATHANRAT, 2018).

Os modelos utilizados neste estudo foram a floresta aleatória e o KNN. A floresta aleatória obteve um MAPE de **9.849%** e um MAE de 13,768 segundos. Por outro lado, o desempenho do KNN foi comprometido devido à sua dificuldade em distinguir entre diferentes tipos de notebooks, o que afetou a precisão na determinação do tempo de resposta no conjunto de dados de teste (PRATHANRAT, 2018). Essas informações podem ser visualizadas na Figura 16.

4.2.6 Harmon e coautores

Harmon e coautores realizaram um estudo sobre a previsão de tempos de execução de sequências de código em linha reta, que não contêm recursão nem loops. Para realizar essa previsão, eles utilizaram uma técnica chamada microanálise, que consiste em prever

tempos de execução ponto a ponto em segmentos de código. Essa abordagem utiliza regras de descrição de máquina semelhantes às usadas para geração de código e otimização de *peephole*, traduzindo o código de objeto compilado em uma sequência de instruções de nível muito baixo, chamadas micro.

No trabalho, os autores descrevem uma ferramenta para prever o tempo de execução de segmentos de código nos melhores e piores casos, independentemente da linguagem ou compilador utilizados. Essa ferramenta foi integrada a um compilador C e os resultados preliminares dos testes foram bastante promissores (HARMON, 1994).

4.2.7 Ferdinand e coautores

Em sistemas embarcados, muitas tarefas de segurança crítica apresentam desafios relacionados a características de tempo real. Uma falha em cumprir um determinado prazo pode resultar em danos significativos ou até mesmo perda de vidas. Diante disso, Ferdinand e coautores abordam a necessidade de previsão do tempo de execução no pior caso para garantir que os requisitos de tempo sejam atendidos.

Por meio de uma ferramenta de análise estática, foi desenvolvido um algoritmo que auxilia no desenvolvimento de sistemas complexos de alta complexidade, utilizando hardware de última geração. Esse programa foi utilizado em trechos de código de programas de referência do mundo real (FERDINAND; HECKMANN, 2020).

4.2.8 Ling Huang e coautores

Ling Huang e coautores desenvolveram um sistema que tem como objetivo extrair automaticamente um grande número de recursos da execução de um programa em entradas de amostra, permitindo a construção de modelos de previsão sem a necessidade de conhecimento especializado. O trabalho propõe a utilização da metodologia SPORE para construir modelos preditivos precisos do desempenho do programa, utilizando dados de recursos coletados durante sua execução em entradas de amostra.

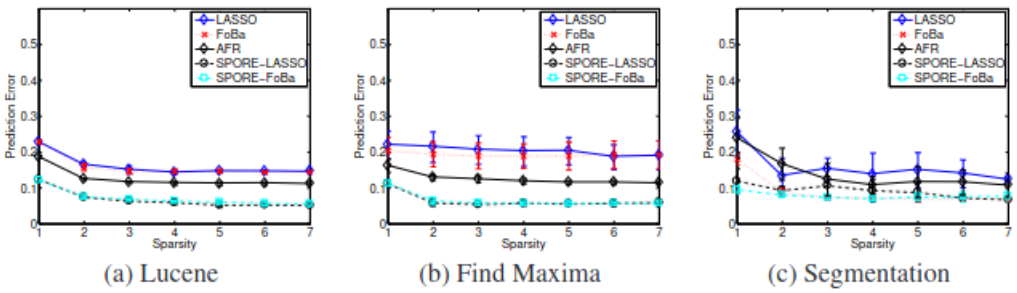


Figura 17 – Resultado do trabalho para diferentes conjunto de dados, retirado de (HUANG; YU BYUNG-GON CHUN; NAIK, 2010).

Ao usar os dois algoritmos para aprender um modelo SPORE, foi constatado que ambos podem prever o tempo de execução com uma precisão superior a **93%** para os aplicativos testados, conforme pode ser observado na Figura 17. Esses resultados representam uma melhoria significativa, com uma redução de **40%** ou mais no erro de previsão em comparação com outras técnicas de modelagem esparsa descritas na literatura quando aplicadas a esse problema (HUANG; YU BYUNG-GON CHUN; NAIK, 2010).

4.2.9 Tulio Braga e coautores

Tulio Braga e coautores exploraram dois modelos analíticos para a previsão do tempo de resposta de aplicações paralelas na plataforma Spark, amplamente utilizada para processamento de grandes volumes de dados. O primeiro modelo é baseado no conceito de fork/join, no qual uma aplicação é dividida em N tarefas que são processadas paralelamente em múltiplos servidores. O segundo modelo é baseado na teoria de filas e leva em consideração a precedência entre as tarefas para estimar os atrasos de sincronização. Diversos cenários experimentais foram considerados, incluindo atividades recorrentes como contagem de palavras, algoritmos frequentemente utilizados em aprendizado de máquina, como SVM, Regressão Logística e K-Means, e consultas ad-hoc comumente utilizadas em análise de dados.

Os modelos desenvolvidos foram aplicados a nove cenários distintos. O modelo de precedência de estágios demonstrou ser satisfatório, com um erro médio de extrapolação de apenas **3.10%**. No entanto, em alguns cenários específicos, os resultados não foram tão eficientes. Por exemplo, o cenário de contador de palavras múltiplo apresentou erros médios de **31.55%**, **27.72%** e **19.21%** em diferentes configurações (PINTO, 2019).

Autores	Algoritmos Utilizados e Técnicas Desenvolvidas
-	-
Lee e coautores	ProgressShare
Zhang e coautores	CloudRecomender
Ben-Yehuda e coautores	-
Singh e coautores	Algoritmo gradiente descendente tradicional
Lucas-Simarro e coautores	Algoritmo de escalonamento
Al-Theiabat e coautores	LSTM e ARIMA
Mishra e coautores	Análise probabilística e Séries temporais
Li e coautores	-
Khandelwal e coautores	Floresta Aleatória
Agarwal e coautores	Redes Neurais
Yun Park	IDL
Miu e coautor	kNN, M5P, MLP e LR
Doan e coautores	Splines de Regressão Adaptativa Multivariada e MVS
Marco Amaris e coautores	BSP
Pariwat e coautores	Floresta Aleatória e kNN
Harmon e coautores	Microanálise
Ferdinand e coautores	Análise estática
Ling Huang e coautores	SPORE
Tulio Braga e coautores	SVM, Regressão Logística e K-Means

Tabela 1 – Tabela resumo de autores

5 Ferramentas e Tecnologias

Neste trabalho, propomos e avaliamos uma ferramenta para a previsão de custo de funções na AWS Lambda, que é um serviço de computação em nuvem baseado em arquitetura sem servidor. Para isso, é necessário prever o tempo de execução de funções nesse ambiente. A abordagem empregada engloba diversas tecnologias e oferece quatro métodos distintos para realizar a predição, além de aplicar técnicas diferentes para calcular o erro e a acurácia de cada um desses métodos.

5.1 Arquitetura

A Figura 18 apresenta um diagrama que representa a arquitetura da aplicação web, a qual é dividida em dois microsserviços, o de front-end e o de back-end. O back-end, implementado em Flask, possui as funções de previsão de tempo, que foram implementadas através da biblioteca Scikit-learn, e a leitura da base de dados, que utiliza a biblioteca Pandas para interpretar o arquivo CSV. Após isso, o front-end, desenvolvido em React, recebe através de requisições HTTP as informações do back-end.

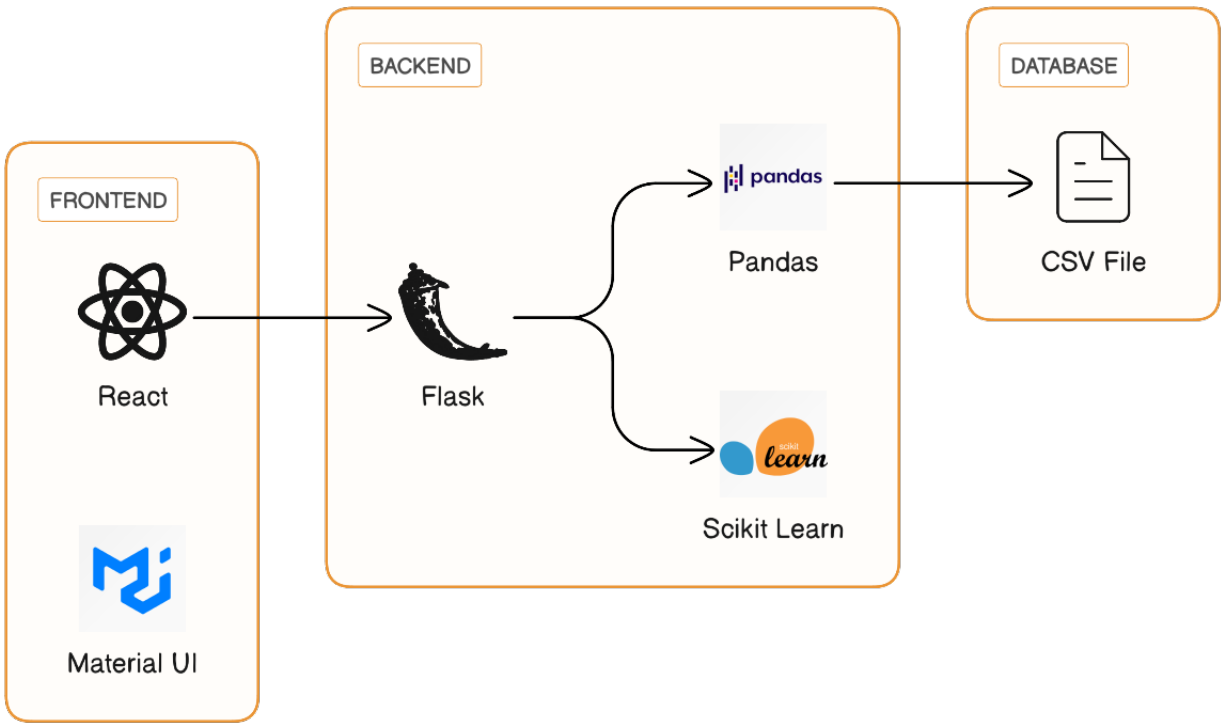


Figura 18 – Diagrama da aplicação web.

5.2 Tecnologias

No desenvolvimento deste projeto, foram empregadas diversas tecnologias, com destaque para linguagens de programação, bibliotecas e ferramentas que oferecem suporte à inteligência artificial.

Python

Python é uma linguagem de programação de alto nível, interpretada e de fácil aprendizado, em decorrência dessas e outras caraterísticas é aplicada em diversos contextos, como sistemas web, mobile e desktop.

A escolha desta linguagem se dá pelo fato de ser popularmente utilizada para aplicações que envolvem análise de dados, inteligência artificial e aprendizado de máquina. Isso se deve, em parte, à disponibilidade de bibliotecas como numpy, pandas, matplotlib, entre outras, que fornecem suporte para essas tarefas, portanto, há uma ampla gama de ferramentas e recursos disponíveis para as necessidades do projeto (RASCHKA; PATTERSON; NOLET, 2020).

Scikit-learn

A scikit-learn é uma biblioteca de código aberto de Python desenvolvida para aplicações práticas que envolvem aprendizado de máquina ou análise de dados. Possui diversos módulos, sendo cada um desenvolvido para uma finalidade específica, com algoritmos para pré-processamento, classificação, regressão, clusterização, redução de dimensionalidade e ajuste de parâmetros (PEDREGOSA et al., 2011).

A escolha da utilização desta biblioteca foi simples, haja vista que possui implementações de todas as técnicas necessárias para o desenvolvimento do projeto. Na aplicação web, o algoritmo implementado utilizando a scikit-learn foi integrado ao backend de forma eficiente, por ambos utilizarem Python como linguagem, conforme pode ser visto na Figura 18.

Matplotlib

Matplotlib é uma biblioteca de Python utilizada para visualização de dados, que oferece uma ampla variedade de recursos permitindo construção de gráficos seja em 2D ou em 3D (HUNTER, 2007).

Essa biblioteca foi utilizada com o objetivo de representar os valores de forma gráfica, facilitando assim a compreensão, comparação e análise dos resultados. Embora o projeto envolva a medição de erros em forma de valores, a apresentação visual por meio de gráficos facilita a compreensão.

Pandas

Pandas é uma biblioteca de Python para análise e manipulação de dados, que oferece suporte a estruturas de dados eficientes para trabalhar com tabelas, planilhas e bancos de dados. Essa tecnologia permite importar dados de várias fontes e formatos, além de realizar operações como filtragem e transformação(MCKINNEY et al., 2011).

A utilização desta biblioteca no projeto se deve ao fato de que a base de dados está no formato .csv, portanto, é necessário ler e transformar esses dados em um formato compatível com a scikit-learn, possibilitando o seu uso no contexto do projeto. Dessa forma, a biblioteca Pandas é utilizada para a leitura e tratamento do arquivo .csv, conforme pode ser visto na Figura 18.

React

React é uma biblioteca JavaScript *open-source* criada pela Meta para resolver desafios tecnológicos internos. É amplamente utilizada para criar interfaces de usuário e aplicações web, e ganhou popularidade devido a suas características, como componentização, desempenho e facilidade de aprendizado e uso, haja vista que essas possibilitam uma alta reutilização de código e o desenvolvimento de aplicações web dinâmicas e responsivas (GACKENHEIMER, 2015).

Utilizado no desenvolvimento do *front-end* da aplicação web, o React foi escolhido como tecnologia principal devido à sua capacidade de reutilização de código por meio da componentização e a ampla participação da comunidade. Essa escolha facilita o suporte técnico e a busca por soluções para problemas comuns enfrentados pelos desenvolvedores.

Material UI

Material UI é uma biblioteca de componentes de interface de usuário *open-source* projetada para o React e baseada no conceito de Material Design desenvolvido pela Google. Possui uma coleção de componentes pré-construídos como botões, barras e ícones prontos para utilização direta.

Foi escolhida para o desenvolvimento *front-end* devido à sua integração simples com o React, o que torna a utilização simples e direta. Ademais, o Material UI oferece componentes prontos para uso e possui flexibilidade para personalizações, dessa forma, é possível criar uma interface bonita de forma simples, permitindo concentrar os esforços na integração do algoritmo com o back-end.

Flask

O Flask é um *framework* web escrito em Python que permite a criação de aplicações web seguindo a arquitetura MVC. É amplamente utilizado devido à sua leveza e flexibilidade, permitindo a utilização de diferentes extensões e bibliotecas para tarefas como abstração da camada de dados, proporcionando ao desenvolvedor a liberdade de escolha.

No desenvolvimento do *back-end* da aplicação web, foi escolhido o Flask como tecnologia principal, pelo fato de ser baseado em Python, pois facilita a integração com o algoritmo de predição, uma vez que ambos na mesma linguagem, o que torna processo de desenvolvimento mais rápido e simples.

A Figura 18 ilustra a integração do back-end com as demais partes da aplicação, destacando a interação com as bibliotecas responsáveis pela leitura da base de dados e pela realização da predição.

Trello

Trello é uma ferramenta de gestão de projetos online, famosa por sua flexibilidade e interface visual amigável. Ela se baseia na metodologia Kanban, um sistema de gerenciamento de produção que enfatiza a eficiência e agilidade por meio de uma representação visual do fluxo de trabalho.

No Kanban, as tarefas são organizadas em colunas representando diferentes estágios do processo, e os cartões individuais são movidos ao longo dessas colunas conforme o progresso é feito. No contexto do projeto trabalhado, o Trello foi utilizado como uma ferramenta vital para organizar o fluxo de trabalho.

As tarefas foram divididas em diferentes fases, como "Para fazer", "Em progresso"e "Concluído", e cada membro da equipe tinha a capacidade de atualizar o status das tarefas em tempo real. Isso não apenas facilitou a comunicação e a colaboração entre os membros da equipe, mas também permitiu um acompanhamento contínuo do progresso, ajudando a identificar gargalos e alocar recursos de forma mais eficaz, garantindo que o projeto permanecesse no caminho certo para atingir seus objetivos.

5.3 Base de dados

Foi criada uma base de dados com o objetivo de treinar diversos algoritmos. Essa base consiste em uma variedade multifacetada de funções, abrangendo desde algoritmos de ordenação comuns, como Quick Sort e Merge Sort, até operações matemáticas complexas, como resolução de equações diferenciais e manipulação de texto, como a análise de sentimentos.

As funções foram executadas com diferentes tamanhos de entrada, desde pequenos vetores até grandes conjuntos de dados, resultando em uma ampla variação nos tempos de execução observados. Os valores coletados abrangem uma faixa de tempo que varia de 3 milissegundos a 15 minutos.

Com base nessas informações, é gerado um arquivo .csv que armazena os dados relevantes para cada execução de função. Esse arquivo não contém apenas o tempo de execução, mas também informações cruciais como complexidade assintótica, que descreve o comportamento de tempo de execução em relação ao tamanho da entrada, a quantidade de entradas, detalhando o número de elementos processados, e outros metadados que podem ser relevantes para a análise do desempenho.

Essas funções foram escolhidas especificamente por possuírem baixa quantidade de operações de entrada e saída, ou rede, garantindo que os tempos de execução coletados sejam menos influenciados por fatores externos e mais representativos do algoritmo em si. Essa escolha permite uma avaliação mais precisa e um treinamento de modelo mais robusto.

Em resumo, essa base de dados representa um esforço significativo para capturar uma imagem diversificada e detalhada do comportamento do tempo de execução em uma ampla gama de funções. Facilitando a modelagem e a análise do desempenho dos algoritmos.

A base de dados conta com inúmeras linhas e armazena informações de diferentes tipos de funções:

Função	Menor Entrada	Maior Entrada
Bubble Sort	1	1000
Selection Sort	1	11000
Insertion Sort	11000	35700
Quick Sort	500	1600000
Merge Sort	1	475
Shell Sort	1	1000
Contagem de substring	500	2500
Combinação	1000	50000
Permutação	1000	5000
Fast Fourier Transform (FFT)	1000	50000
Tim Sort	1000	50000
Finding Kth Largest Element	1000	50000
Balanced Parentheses Check	1000	50000
Permutação	1000	50000
Menor caminho	1000	12000
Conversão de polegadas para centímetros	10000	20000
Tradução para maiúsculo	100	500
Compressão de arquivos	100	10000
Redimensionar imagens	100	10000
Find all pairs	200	10000
Matrix Multiplication	10000	20000
Finding Duplicates	10000	30000
Operações Matemáticas	100	1000000

Tabela 2 – Algoritmos utilizados

6 Resultados

6.1 Algoritmos

Foram analisados estudos relacionados à predição e computação em nuvem, com objetivo de estabelecer uma base do conhecimento existente. A análise permite avaliar diferentes algoritmos que podem ser aplicados. A complexidade do problema requer a utilização de mais de um algoritmo. Portanto, são utilizados múltiplos algoritmos para a realização da predição:

- Regressão Linear;
- Regressão Polinomial;
- Floresta Aleatória;
- Árvore de Decisão.

6.2 Treinamento das funções de predição

O programa começa lendo um arquivo de dados que contém várias características, incluindo "tempo", "complexidade", "quantidade de inputs", etc. Esses dados serão usados para treinar e testar modelos de predição.

Os dados são divididos em diferentes conjuntos, dependendo do tipo de modelo de regressão a ser treinado. Em alguns casos, são usadas todas as colunas como características, enquanto em outros, apenas algumas colunas específicas são usadas.

6.2.1 Regressão Linear com Regularização (Ridge)

- **Características e Divisão de Dados**: Para este modelo, são criados dois conjuntos de treinamento e teste diferentes usando as colunas "complexidade"e "qtd inputs". Cada conjunto é dividido em 60% para teste e 40% para treinamento;
- **Modelagem**: São treinados dois modelos Ridge diferentes, um para cada uma das características escolhidas. O Ridge é uma versão da regressão linear que inclui regularização L2 para evitar o overfitting.
- **Predição**: Os modelos treinados são usados para prever os valores da variável alvo nos conjuntos de teste;

6.2.2 Árvore de Decisão

- **Características e Divisão de Dados**: A coluna "tempo"é definida como a variável dependente, e o resto das colunas são usadas como variáveis independentes. Os dados são divididos em 60% para teste e 40% para treinamento.
- **Modelagem**: Uma Árvore de Decisão é treinada usando essas características. Uma árvore de decisão divide recursivamente os dados em subconjuntos, buscando os melhores pontos de divisão para minimizar um critério de impureza.
- **Predição**: O modelo treinado é utilizado para prever os valores da variável alvo no conjunto de teste.

6.2.3 Floresta Aleatória

- **Características e Divisão de Dados**: A mesma seleção de características e divisão de dados da Árvore de Decisão é usada aqui.
- **Modelagem**: É treinado um modelo de Floresta Aleatória, que é um ensemble de Árvores de Decisão. Isso aumenta a robustez e precisão do modelo. Vários parâmetros são definidos, como o número de árvores, a profundidade máxima e o número mínimo de amostras para dividir um nó.
- **Predição**: O modelo treinado é utilizado para prever os valores da variável alvo no conjunto de teste.

6.2.4 Regressão Polinomial

- **Características e Divisão de Dados**: Os mesmos dados usados para a Árvore de Decisão e Floresta Aleatória são usados, com a mesma divisão de 60% para teste e 40% para treinamento.
- **Modelagem**: As características são transformadas em um polinômio de grau 2 usando o PolynomialFeatures. A regressão linear é então aplicada a essas características polinomiais.
- **Predição**: O modelo treinado é usado para prever os valores da variável alvo no conjunto de teste. Os valores negativos preditos são ajustados para zero, pois não fazem sentido no contexto.

Em resumo, cada modelo usa diferentes técnicas de regressão e abordagens para modelar os dados. As divisões de treinamento e teste e a variável alvo permanecem consistentes entre os modelos, permitindo uma comparação justa de seu desempenho. A escolha

das características e a transformação de dados são personalizadas para cada modelo com base na natureza da técnica de regressão utilizada.

6.3 Resultados da Predição de Tempo

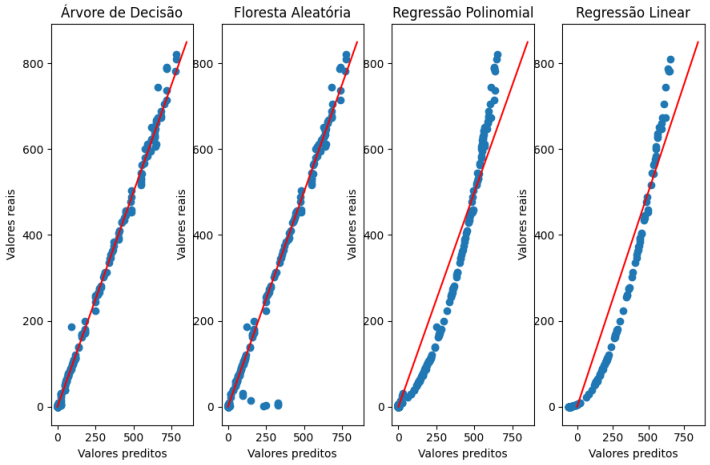


Figura 19 – Gráfico de comparação entre os valores para cada algoritmo

A forma de visualização dos valores preditos para execução do tempo é por meio de gráficos plotados utilizando a biblioteca Matplotlib, como mostrado na Figura 19. Nestes, cada algoritmo é representado em um plano cartesiano, em que o eixo Y representa os valores reais e o eixo X representa os valores preditos. Através dessa representação, é traçado um segmento de reta definido pela equação X = Y, em que cada ponto no gráfico corresponde a um dado coletado. Assim, quanto mais próximos os pontos estiverem do segmento de reta, mais eficiente foi o algoritmo para aquele tempo coletado.

A utilização de gráficos objetiva proporcionar uma visualização mais clara dos dados, facilitando a comparação entre valores preditos e coletados.

6.4 Análise

Para interpretação dos dados e comparação dos valores registrados com os preditos, são utilizadas diferentes métricas de erro e acurácia, sendo elas:

- Erro quadrático médio;
- Raiz do erro quadrático médio;
- Erro médio absoluto;
- R^2 Score.

Na tabela 3 é possível comparar as quatro técnicas de predição aplicadas ao conjunto de dados. As linhas representam as técnicas de predição, enquanto as colunas mostram as medidas de desempenho utilizadas para avaliar sua eficácia.

	MSE	RMSE	MAE	R^2
Regressão Linear	68.02	4628.00	60.01	91.76%
Regressão Polinomial	53.42	2854.67	34.29	94.64%
Árvore de Decisão	11.36	129.21	4.63	99.76%
Floresta Aleatória	18.69	349.40	6.07	99.34%

Tabela 3 – Comparação de erros

6.4.1 Regressão Linear

O modelo de regressão linear apresentou um MSE de 68.02, um RMSE de 4628.00, um MAE de 60.01 e um R^2 de 91.76%. Isso significa que o modelo tem um desempenho razoável, no entanto sua precisão é inferior em comparação aos outros modelos avaliados.

O modelo de regressão linear mostrou-se inapropriado para o conjunto de dados em questão. A regressão linear estabelece uma relação entre duas variáveis, sendo uma dependente e a outra independente. Neste estudo, existem múltiplas variáveis independentes, o que demanda a aplicação da regressão linear para cada uma delas, o que amplifica a divergência entre os valores inicialmente coletados e os previstos.

Na Figura 20, a ineficiência do modelo de regressão linear se torna evidente quando empregada na base de dados utilizada, que apresenta execuções de tempo correlacionadas com diversos graus de complexidade assintótica. Ao se aplicar o modelo de regressão linear a um conjunto de dados diversificados que engloba uma vasta gama de variáveis para analisar, o método resultou em uma média de erro significativamente alta. Esta situação sugere que o modelo de regressão linear não é a escolha mais eficiente para lidar com conjuntos de dados multivariados e complexos, visto que acarreta em limitações no desempenho e na eficácia do modelo.

Esta falha torna-se especialmente notável quando compara-se o desempenho com modelos mais sofisticados e eficazes, como floresta aleatória e árvore de decisões. Diferentemente da regressão linear, esses modelos são capazes de manipular e interpretar um maior número de variáveis, contribuindo para uma previsão mais precisa.

6.4.2 Regressão Polinomial

A regressão polinomial obteve um MSE de 53.42, um RMSE de 2854.67, um MAE de 34.29 e um R^2 de 94.64%. Os resultados obtidos demonstraram uma eficácia considerável, embora com a base de dados maior os algoritmos de árvores de decisão e floresta aleatória se mostraram mais eficientes.

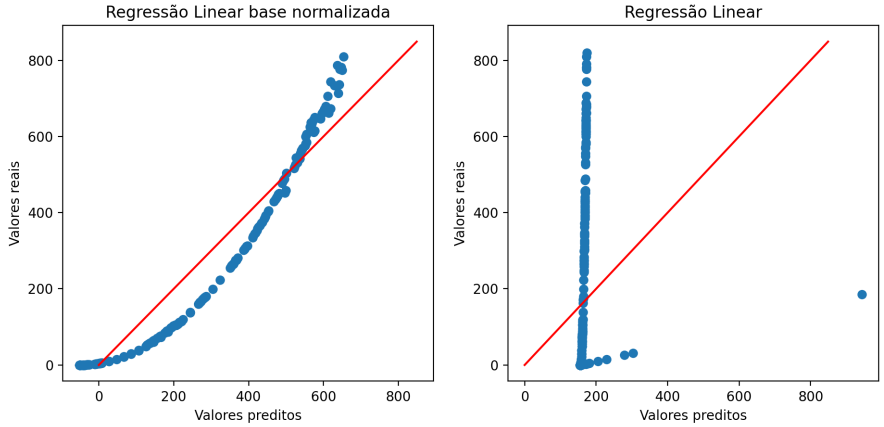


Figura 20 – Comparação entre a regressão linear de uma base de complexidade normalizada e uma base com dados diversificados.

Como uma generalização da regressão linear, a polinomial foi uma escolha natural de técnica a ser utilizada, pois seu funcionamento é semelhante ao da linear.

Esse método demonstrou ser mais eficiente do que a regressão linear, pois a função é modelada a partir da combinação de uma ou mais variáveis independentes, tornando-o mais adequado para o problema, uma vez que o tempo de execução de uma função depende de múltiplos parâmetros.

Com uma baixa quantidade de dados na base, essa técnica apresenta uma eficiência limitada, no entanto, esse algoritmo mostrou-se bastante eficiente chegando a ter o maior R^2 em determinado momento. Embora não seja a melhor, é a técnica mais eficiente para problemas $O(n \log n)$.

6.4.3 Árvore de Decisão

A Árvore de decisão apresentou um MSE de 11.36, um RMSE de 129.21, um MAE de 4.63 e um R^2 de 99.76%. Isso indica que dentre os algoritmos, esta é a técnica mais precisa e eficiente.

A popularidade deste método em problemas de regressão justifica sua utilização. Entretanto as florestas aleatórias são mais frequentemente empregadas, pois é uma técnica avançada baseada no algoritmo de árvore de decisões.

Com uma base pouco populada os métodos de regressão polinomial e floresta aleatória demonstram maior eficiência. No entanto, com uma base de dados mais populada a árvore de decisão se mostrou a técnica com o maior R^2.

6.4.4 Floresta Aleatória

O algoritmo do Floresta aleatória apresentou um MSE de 18.69, um RMSE de 349.40, um MAE de 6.07 e um R^2 de 99.34%. Portanto, é o segundo modelo com maior precisão e demonstra uma alta capacidade de generalização.

Ao revisar os estudos relacionados, observou-se que o algoritmo de inteligência artificial mais amplamente utilizado em regressões era o Florestas Aleatórias, devido à sua simplicidade e alta precisão.

Esse algoritmo mostra-se mais eficiente na resolução de problemas em comparação com métodos anteriores, principalmente com uma base menos populada, quando ambas as regressões apresentam baixa eficácia.

Com o preenchimento da base de dados, a precisão atingiu níveis satisfatórios e eficientes, se mostrando o melhor para resolver problemas de complexidade quadrática.

6.5 Predição de Custo

A obtenção de resultados satisfatórios na predição do tempo permite estimar com precisão o custo de execução das funções Lambda, utilizando a fórmula básica fornecida pela Amazon, em conjunto com as tabelas de preços mais recentes disponíveis na plataforma:

$$Custo = (TE * MA) + (NR * CR), \textbf{ onde:} \tag{6.1}$$

- TE: tempo total de execução da função Lambda, medido em segundos.
- MA: quantidade de memória alocada para a função Lambda, expressa em gigabytes (GB).
- NR: quantidade total de requisições feitas à função Lambda.
- CR: valor por requisição feita à função Lambda.

A comparação dos valores obtidos com os da ferramenta oficial de cálculo de preço da AWS Lambda, ilustrados respectivamente na Figura 22 e 21, uma referência confiável, apresenta os resultados das previsões, mostrando-se próximos. Essa proximidade de valores valida a robustez e confiabilidade do cálculo utilizado.

Nesta comparação, o algoritmo de predição demonstra um nível considerável de exatidão na previsão de custos. Utilizando a mesma quantidade de memória alocada e o

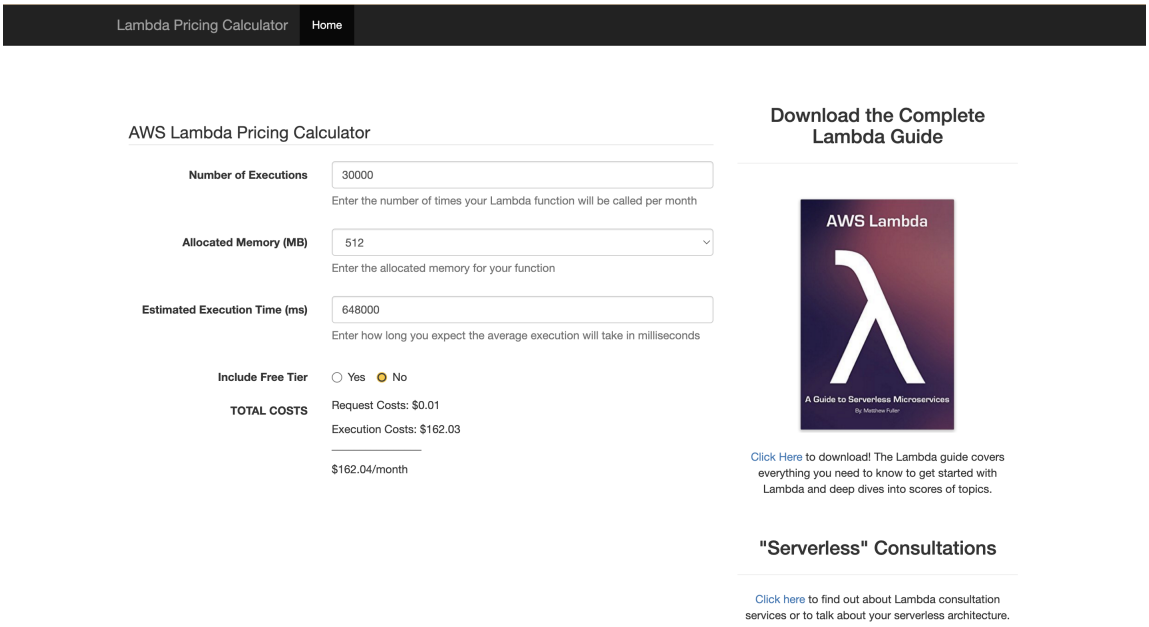


Figura 21 – Calculadora de custo AWS Lambda

número mensal de solicitações, o programa alcançou o mesmo valor estimado que a ferramenta da AWS. Este resultado não apenas reforça a validade do programa desenvolvido, mas também ressalta sua aplicabilidade e utilidade prática.

A correspondência dessas estimativas também confirma a eficácia do programa em calcular custos com base no tempo de execução previsto. Esse aspecto, em particular, é fundamental, pois a previsão de custos pode desempenhar um papel crucial na otimização dos recursos e no gerenciamento eficiente das operações. Portanto, os resultados reforçam que as projeções de custos atreladas ao tempo de execução previsto podem fornecer uma base sólida e mais informativa para decisões estratégicas e operacionais.

É importante ressaltar que o custo por requisição da função Lambda pode variar de acordo com a região em que ela é executada e o tipo de invocação, seja síncrona ou assíncrona. A ferramenta permite ao usuário controlar os parâmetros de memória alocada e escolher em qual servidor da AWS Lambda se baseia, a fim de obter um cálculo de preço mais preciso por execução. Além disso, é recomendável consultar a página de preços da AWS para obter as informações mais atualizadas sobre os valores de custo por requisição.

Além disso, destaca-se que o tempo de execução da uma função pode variar, mesmo quando executada na mesma região. Durante a construção da base de dados, foram identificadas variações de até 6% no tempo de execução das funções, essa variabilidade também pode ter impacto no cálculo do preço final, pois afeta diretamente o custo associado por requisição.

Existem outros fatores que podem adicionar custos à função Lambda, como transferência de dados, armazenamento temporário e recursos de rede, que devem ser avaliados

Figura 22 – Resultado da predição

separadamente, conforme necessário.

6.5.1 Aplicação Web

Com a conclusão da técnica de previsão, desenvolvemos uma aplicação web para facilitar a utilização pelos usuários, sendo criada utilizando Javascript, framework React, e Python para o backend, com framework Flask, permitindo uma integração eficiente com o algoritmo.

Esta permite que usuários insiram informações essenciais de suas funções para a predição, como a quantidade de entradas, execuções que aquela função será executada por mês na AWS Lambda, memória disponível e a análise assintótica conforme ilustrado na Figura 23. Essas informações são inseridas por meio de uma interface simples e amigável, em seguida, são enviadas para o servidor, onde são processadas e transformadas em dados para a execução do algoritmo.

As informações fornecidas pelo usuário são recebidas pelo back-end e transformadas em entradas para o algoritmo, que realiza a análise e a previsão desejada. O resultado da previsão é enviado de volta para o front-end e apresentado de forma clara e organizada para o usuário, conforme mostra a Figura 22.

Dessa forma, a aplicação web desenvolvida oferece uma solução visual completa para a utilização do algoritmo de previsão. Por meio dela, os usuários podem utilizar uma interface intuitiva para inserir os dados necessários e obter os resultados de forma rápida e eficiente. A integração bem-sucedida entre a aplicação web e o algoritmo proporciona uma experiência mais fácil aos usuários, tornando a previsão mais acessível e simplificada.

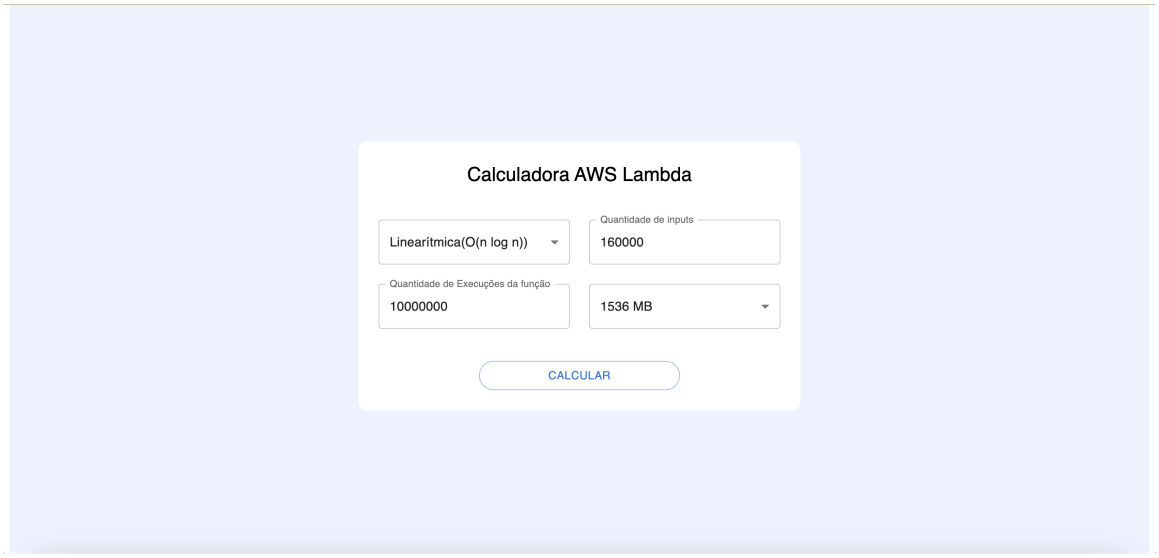


Figura 23 – Interface de entrada dos dados.

7 Conclusão e Trabalhos Futuros

Os estudos e aplicações de modelo para prever o tempo de execução de códigos na AWS permitiram a antecipação dos custos de utilização dessa ferramenta. Os métodos utilizam técnicas de análise estatística e aprendizado de máquina com o objetivo de auxiliar usuários a tomarem decisões de negócio que envolvam o cálculo do custo-benefício de serviços sem servidor. A realização desse estudo baseia-se no fato de que, em grande parte, os projetos que visam a predição na arquitetura em nuvem se concentram na previsão de carga de trabalho ou tempo de execução, enquanto a aplicação para custos não é tão explorada.

A organização da base e tema do trabalho torna possível o estudo dos tópicos e temas necessários para o entendimento do problema e das possíveis soluções, como computação em nuvem e inteligência artificial. As decisões foram tomadas com base em critérios como, eficiência e quantidade de uso em trabalhos similares, às técnicas a serem utilizadas no desenvolvimento do projeto, sendo elas regressão linear e polinomial, árvore de decisão e floresta aleatórias.

O desenvolvimento da parte do projeto responsável pela previsão do tempo de execução iniciou-se com a construção da base de dados de funções e logo após, a implementação das técnicas escolhidas. Os resultados apresentados no Capítulo 6 e descritos na Tabela 3, mostram que foram obtidas acurácias de 99.76% para o algoritmo mais eficiente e 91.76% para o menos, árvore de decisão e regressão linear, respectivamente, além de 99.34% para floresta aleatória e 94.64% para regressão polinomial.

A partir da previsão de tempo precisa, tornou-se possível o início o processo de estimativa do preço, utilizando como base uma fórmula fornecida pela Amazon, tomando como referência parâmetros fornecidos pelo usuário, a ferramenta prediz o tempo de execução e insere na fórmula para retornar um provável valor.

Estimar o custo de execução das funções AWS Lambda é uma prática essencial para aqueles que desejam otimizar a eficiência e controlar gastos na plataforma. As informações de custo obtidas proporcionam uma visão clara e detalhada do investimento necessário, permitindo um planejamento mais preciso, além disso, essas informações auxiliam na tomada de decisões, como seleção entre diferentes configurações de execução ou determinação do momento e da forma adequada para escalar as funções e atender à demanda.

Finalmente, além das contribuições nesse trabalho existem formas promissoras de evoluir e melhorar este projeto. Propomos como trabalhos futuros aumentar a base de dados construída, pois quanto mais informações utilizadas para o treinamento dos

algoritmos mais eficiente será a predição. Outra possível linha é a ampliação das linguagens utilizadas, haja vista que o algoritmo é treinado apenas para funções em Python, ocorre que por analisar apenas complexidade assintótica e quantidade de entrada, poderia ser expandido para outras linguagens de programação. A abordagem dessas melhorias poderá contribuir significativamente para o aprimoramento e a aplicação prática das técnicas de previsão de custos em ambientes de computação em nuvem.

Apêndices

APÊNDICE A – Aplicação Web

Figura 24 – Interface inicial.

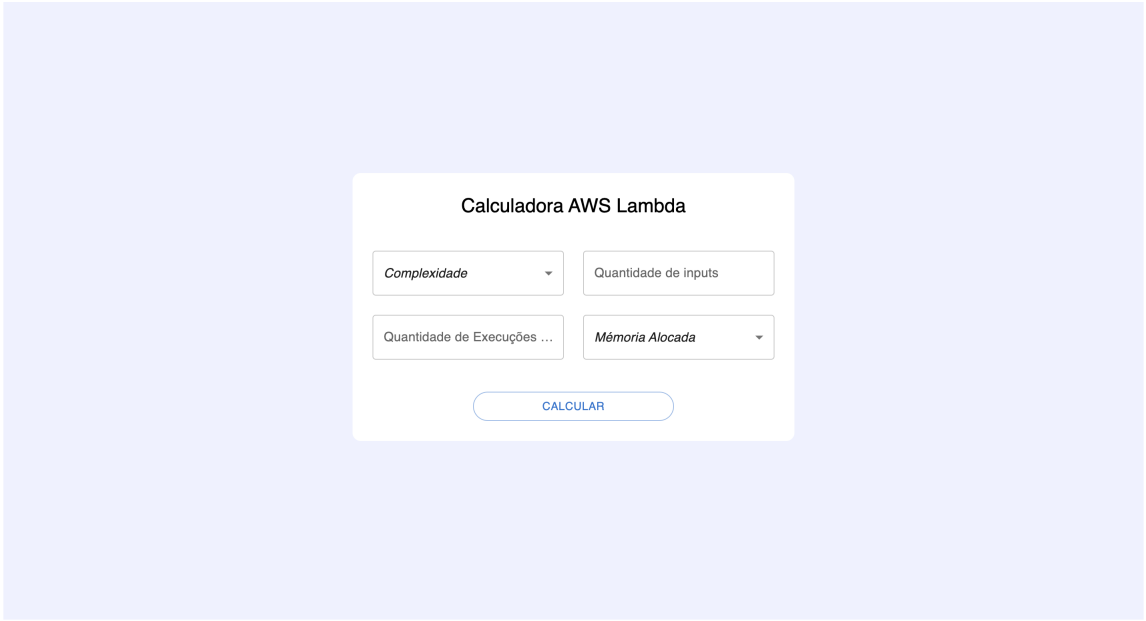


Figura 25 – Calculadora de custo.

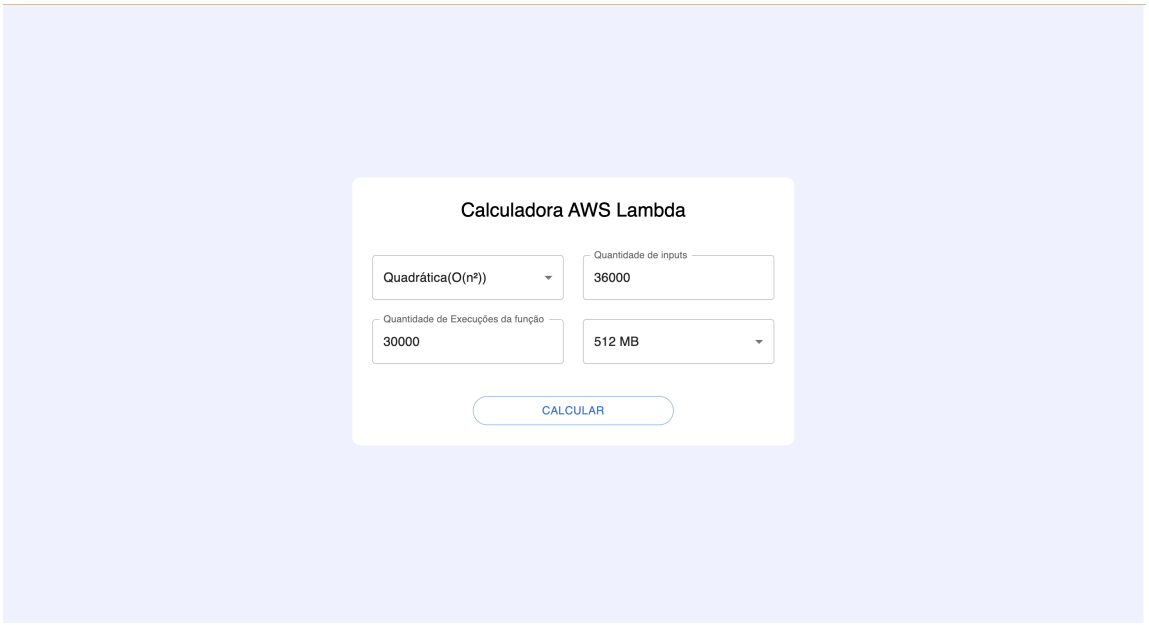


Figura 26 – Interface de predição

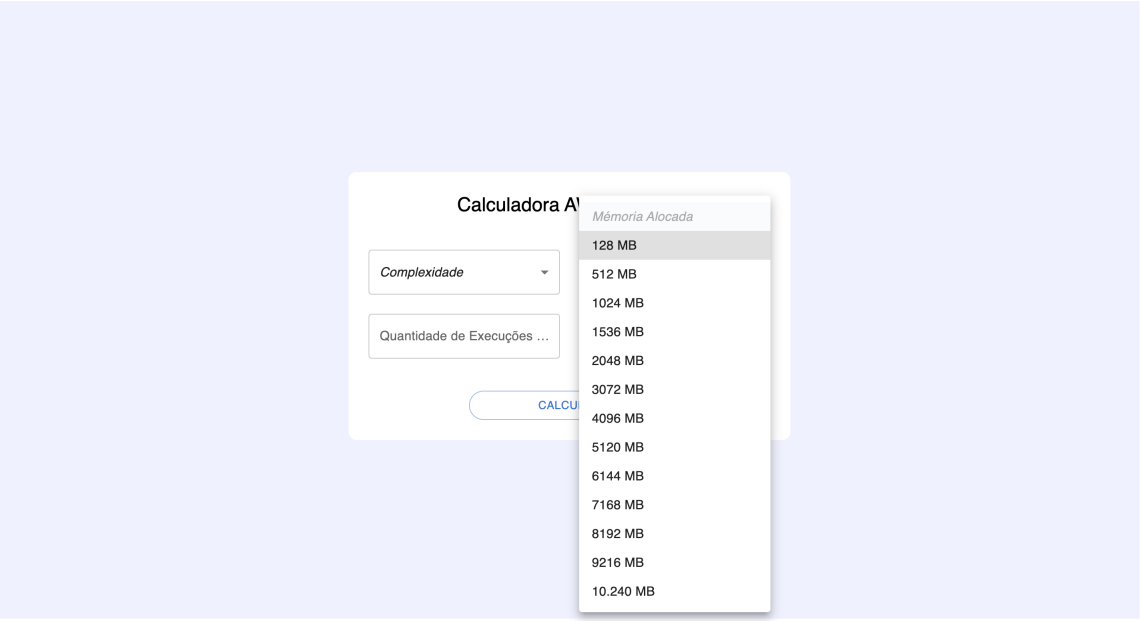


Figura 27 – Seleção de memória da calculadora.

APÊNDICE B – Tabelas de custo da AWS Lambda

Mémoria(MB)	Preço por 1 milissegundo
128	0,0000000021 USD
512	0,0000000083 USD
1024	0,0000000167 USD
1536	0,0000000250 USD
2048	0,0000000333 USD
3072	0,0000000500 USD
4096	0,0000000667 USD
5120	0,0000000833 USD
6144	0,0000001000 USD
7168	0,0000001167 USD
8192	0,0000001333 USD
9216	0,0000001500 USD
10.240	0,0000001667 USD

Tabela 4 – Preço por quantidade de Memória Alocada por Milissegundo(Jun/2023).

Duração	Solicitações
0,0000166667 USD por cada GB-segundo	0,20 USD por 1 milhão de solicitações
0,000015 USD para cada GB-segundo	0,20 USD por 1 milhão de solicitações
0,0000133334 USD para cada GB-segundo	0,20 USD por 1 milhão de solicitações

Tabela 5 – Custo de duração e quantidades de requisições na Lambda(Jun/2023).

Referências

AGARWAL, S.; MISHRA, A. K.; YADAV, D. K. Forecasting price of amazon spot instances using neural networks. *Int. J. Appl. Eng. Res*, v. 12, n. 20, p. 10276–10283, 2017. Citado 4 vezes nas páginas 6, 11, 39 e 40.

AL-THEIABAT, H. et al. A deep learning approach for amazon ec2 spot price prediction. In: IEEE. *2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA)*. [S.l.], 2018. p. 1–5. Citado 4 vezes nas páginas 6, 11, 37 e 38.

ALZUBI, J.; NAYYAR, A.; KUMAR, A. Machine learning from theory to algorithms: an overview. In: IOP PUBLISHING. *Journal of physics: conference series*. [S.l.], 2018. v. 1142, n. 1, p. 012012. Citado na página 26.

AMARIS, M.; CORDEIRO, A. G. D.; CAMARGO, R. Y. D. A simple bsp-based model to predict execution time in gpu applications. In: IEEE. *2015 IEEE 22nd International Conference on High Performance Computing (HiPC)*. [S.l.], 2015. Citado na página 43.

AMAZON. *What is AWS Lambda?* 2022. Disponível em: <https://docs.aws.amazon.com/lambda/latest/dg/welcome.html>. Citado 2 vezes nas páginas 20 e 21.

AWS. *AWS Lambda*. 2022. Disponível em <https://aws.amazon.com/lambda/> (acessado 17 de Setembro de 2022). Citado 3 vezes nas páginas 11, 23 e 24.

AWS. *Serverless on AWS*. 2022. Disponível em <https://aws.amazon.com/serverless/> (acessado 17 de Setembro de 2022). Citado na página 11.

BEN-YEHUDA, O. A. et al. Deconstructing amazon ec2 spot instance pricing. *ACM Transactions on Economics and Computation (TEAC)*, ACM New York, NY, USA, v. 1, n. 3, p. 1–20, 2013. Citado 2 vezes nas páginas 6 e 35.

BONACCORSO, G. *Machine learning algorithms*. [S.l.]: Packt Publishing Ltd, 2017. Citado na página 26.

BOSE, R. Advanced analytics: opportunities and challenges. *Industrial Management & Data Systems*, Emerald Group Publishing Limited, 2009. Citado na página 26.

DOAN, T.; KALITA, J. Predicting run time of classification algorithms using meta-learning. *International Journal of Machine Learning and Cybernetics*, Springer, v. 8, n. 6, p. 1929–1943, 2017. Citado na página 42.

FERDINAND, C.; HECKMANN, R. ait:worst-case execution time prediction by static program analysis. In: IEEE. *AbsInt Angewandte Informatik GmbH*. [S.l.], 2020. Citado na página 44.

FOSTER, I.; KESSELMAN, C. *The Grid 2: Blueprint for a new computing infrastructure*. [S.l.]: Elsevier, 2003. Citado na página 14.

GACKENHEIMER, C. *Introduction to React*. [S.l.]: Apress, 2015. Citado na página 49.

GUNANTARA, N.; PUTRA, I. N. The characteristics of metaheuristic method in selection of path pairs on multicriteria ad hoc networks. *Journal of Computer Networks and Communications*, Hindawi, v. 2019, 2019. Citado na página 31.

GURU99. *What is AWS Lambda? Lambda Function with Examples*. 2022. Disponível em: <https://www.guru99.com/aws-lambda-function.html>. Citado 2 vezes nas páginas 6 e 21.

HARMON, T. P. B. . D. B. W. M. G. A retargetable technique for predicting execution time of code segments. In: IEEE. *SpringlerLink*. [S.l.], 1994. Citado na página 44.

HUANG, J. J. L.; YU BYUNG-GON CHUN, P. M. B.; NAIK, M. Predicting execution time of computer programs using sparse polynomial regression. In: IEEE. *Advances in Neural Information Processing Systems 23 (NIPS 2010)*. [S.l.], 2010. Citado 3 vezes nas páginas 6, 44 e 45.

HUNTER, J. D. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, IEEE Computer Society, v. 9, n. 03, p. 90–95, 2007. Citado na página 48.

JONAS, E. et al. Cloud programming simplified: A berkeley view on serverless computing. *arXiv preprint arXiv:1902.03383*, 2019. Citado 3 vezes nas páginas 17, 18 e 19.

KHANDELWAL, V.; CHATURVEDI, A. K.; GUPTA, C. P. Amazon ec2 spot price prediction using regression random forests. *IEEE Transactions on Cloud Computing*, IEEE, v. 8, n. 1, p. 59–72, 2017. Citado 4 vezes nas páginas 6, 11, 40 e 41.

KUROSE, J.; ROSS, K. *Computer networks: A top down approach featuring the internet*. [S.l.]: Pearson Addison Wesley, 2014. Citado na página 14.

LEE, G.; KATZ, R. H. {Heterogeneity-Aware} resource allocation and scheduling in the cloud. In: *3rd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 11)*. [S.l.: s.n.], 2011. Citado na página 34.

LI, Z. et al. Spot pricing in the cloud ecosystem: A comparative investigation. *Journal of Systems and Software*, Elsevier, v. 114, p. 1–19, 2016. Citado na página 39.

LOGISTIC regression. Wikimedia Foundation, 2022. Disponível em: <https://en.wikipedia.org/wiki/Logistic_regression>. Citado 2 vezes nas páginas 6 e 30.

LUCAS-SIMARRO, J. L. et al. Cost optimization of virtual infrastructures in dynamic multi-cloud scenarios. *Concurrency and Computation: Practice and Experience*, Wiley Online Library, v. 27, n. 9, p. 2260–2277, 2015. Citado 2 vezes nas páginas 6 e 37.

MAHESH, B. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, v. 9, p. 381–386, 2020. Citado 4 vezes nas páginas 6, 27, 28 e 29.

MCCARTHY, J. et al. What is artificial intelligence. Stanford University, 2007. Citado na página 25.

MCKINNEY, W. et al. pandas: a foundational python library for data analysis and statistics. *Python for high performance and scientific computing*, Seattle, v. 14, n. 9, p. 1–9, 2011. Citado na página 49.

MELL, P.; GRANCE, T. et al. The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National . . . , 2011. Citado 4 vezes nas páginas 11, 14, 15 e 17.

MISHRA, A. K.; KESARWANI, A.; YADAV, D. K. Short term price prediction for preemptible vm instances in cloud computing. In: IEEE. *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*. [S.l.], 2019. p. 1–9. Citado 3 vezes nas páginas 6, 11 e 38.

MIU, T.; MISSIER, P. Predicting the execution time of workflow activities based on their input features. In: IEEE. *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*. [S.l.], 2012. p. 64–72. Citado 3 vezes nas páginas 6, 41 e 42.

MONTGOMERY, D. C.; PECK, E. A.; VINING, G. G. *Introduction to linear regression analysis*. [S.l.]: John Wiley & Sons, 2021. Citado na página 29.

NESMACHNOW, S. An overview of metaheuristics: accurate and efficient methods for optimisation. *International Journal of Metaheuristics*, Inderscience Publishers (IEL), v. 3, n. 4, p. 320–347, 2014. Citado na página 31.

PARK, C. Y. Predicting program execution times by analyzing static and dynamic program paths. *Real-Time Systems*, Springer, v. 5, n. 1, p. 31–62, 1993. Citado na página 41.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, JMLR. org, v. 12, p. 2825–2830, 2011. Citado na página 48.

PINTO, T. B. M. Previsão do tempo de resposta de aplicações paralelas de processamento de dados massivos em ambientes de nuvem. In: IEEE. *Universidade Federal de Minas Gerais*. [S.l.], 2019. Citado na página 45.

PORTELLA, G. J. Precificação em computação em nuvem para instâncias permanentes e transientes: modelagem e previsão. 2021. Citado 3 vezes nas páginas 6, 14 e 16.

PRATHANRAT, C. P. P. Performance prediction of jupyter notebook in jupyterhub using machine learning. In: IEEE. *2018 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*. [S.l.], 2018. Citado 2 vezes nas páginas 6 e 43.

RASCHKA, S.; PATTERSON, J.; NOLET, C. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *Information*, Multidisciplinary Digital Publishing Institute, v. 11, n. 4, p. 193, 2020. Citado na página 48.

REDHAT. *What is serverless?* 2022. Disponível em: <https://www.redhat.com/en/topics/cloud-native-apps/what-is-serverless>. Citado 2 vezes nas páginas 18 e 19.

RUSSELL, S. J. *Artificial intelligence a modern approach*. [S.l.]: Pearson Education, Inc., 2010. Citado na página 25.

SATHYA, R.; ABRAHAM, A. et al. Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence*, Citeseer, v. 2, n. 2, p. 34–38, 2013. Citado na página 27.

SCHNEIDER, A.; HOMMEL, G.; BLETTNER, M. Linear regression analysis: part 14 of a series on evaluation of scientific publications. *Deutsches Ärzteblatt International*, Deutscher Arzte-Verlag GmbH, v. 107, n. 44, p. 776, 2010. Citado 2 vezes nas páginas 6 e 30.

SCHOLLMEIER, R. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In: IEEE. *Proceedings First International Conference on Peer-to-Peer Computing*. [S.l.], 2001. p. 101–102. Citado na página 14.

SENTINELONE. *AWS Lambda Use Cases: 11 Reasons Devs Should Use Lambdas*. 2020. Disponível em: <https://www.sentinelone.com/blog/aws-lambda-use-cases/>. Citado na página 22.

SHAHRAD, M.; BALKIND, J.; WENTZLAFF, D. Architectural implications of function-as-a-service computing. In: *Proceedings of the 52nd annual IEEE/ACM international symposium on microarchitecture*. [S.l.: s.n.], 2019. p. 1063–1075. Citado 2 vezes nas páginas 19 e 20.

SINGH, V. K.; DUTTA, K. Dynamic price prediction for amazon spot instances. In: IEEE. *2015 48th Hawaii International Conference on System Sciences*. [S.l.], 2015. p. 1513–1520. Citado 2 vezes nas páginas 6 e 36.

SPERANDEI, S. Understanding logistic regression analysis. *Biochemia medica*, Medicinska naklada, v. 24, n. 1, p. 12–18, 2014. Citado na página 30.

SYKES, A. O. An introduction to regression analysis. 1993. Citado na página 29.

TANENBAUM, A. S. *Computer networks*. [S.l.]: Pearson Education India, 2003. Citado na página 14.

TRANMER, M.; ELLIOT, M. Multiple linear regression. *The Cathie Marsh Centre for Census and Survey Research (CCSR)*, v. 5, n. 5, p. 1–5, 2008. Citado na página 29.

TURING, A. M. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX, n. 236, p. 433–460, 10 1950. ISSN 0026-4423. Disponível em: <https://doi.org/10.1093/mind/LIX.236.433>. Citado na página 25.

VAQUERO, L. M. et al. *A break in the clouds: towards a cloud definition*. [S.l.]: ACM New York, NY, USA, 2008. 50–55 p. Citado na página 15.

VERAS, M. *Computação em nuvem*. [S.l.]: Brasport Livros e Multimídia Ltda., 2015. Citado 4 vezes nas páginas 14, 15, 16 e 17.

VILLAMIZAR, M. et al. Cost comparison of running web applications in the cloud using monolithic, microservice, and aws lambda architectures. *Service Oriented Computing and Applications*, Springer, v. 11, n. 2, p. 233–247, 2017. Citado na página 19.

ZHANG, M. et al. A declarative recommender system for cloud infrastructure services selection. In: SPRINGER. *International Conference on Grid Economics and Business Models*. [S.l.], 2012. p. 102–113. Citado na página 35.