Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Predição de Recursos para Workflows Científicos de Bioinformática em Nuvens Federadas com Aprendizado de Máquina

Matheus de Carvalho Sobrinho

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Orientadora
Prof.a Dr.a Aletéia Patrícia Favacho de Araújo Von Paumgartten

Brasília
2021

Ficha Catalográfica de Teses e Dissertações

Está página existe apenas para indicar onde a ficha catalográfica gerada para dissertações de mestrado e teses de doutorado defendidas na UnB. A Biblioteca Central é responsável pela ficha, mais informações nos sítios:

http://www.bce.unb.br
http://www.bce.unb.br/elaboracao-de-fichas-catalograficas-de-teses-e-dissertacoes

Esta página não deve ser inclusa na versão final do texto.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Predição de Recursos para Workflows Científicos de Bioinformática em Nuvens Federadas com Aprendizado de Máquina

Matheus de Carvalho Sobrinho

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Prof.a Dr.a Aletéia Patrícia Favacho de Araújo Von Paumgartten (Orientadora)
CIC/UnB

Prof.a Dr.a Célia Ghedini Ralha
Universidade de Brasília - CIC/UnB

Prof. Dr. José Viterbo Filho
Universidade Federal Fluminense - IC/UFF

Prof. Dr. Ricardo Pezzuol Jacobi
Coordenador do Programa de Pós-graduação em Informática

Brasília, 12 de Novembro de 2021

Dedicatória

A todos aqueles que, direta e/ou indiretamente, contribuíram para que este trabalho pudesse ser realizado.

Agradecimentos

As pessoas que sempre estiveram por perto para trazer o conforto necessário frente ao trabalho árduo que esta pesquisa demandou em meio a um momento tão nebuloso. Diante disso, agradeço:

À Prof[a]. Dr[a]. Aletéia Patrícia Favacho de Araújo Von Paumgartten, orientadora e mentora, pelo olhar atencioso a este trabalho, pela confiança, pela força, pela direção e por sempre acreditar no meu crescimento pessoal e intelectual durante o processo de construção desta dissertação. Sem seu apoio, este trabalho não seria de todo completo.

À minha mãe, exemplo de mulher de luta, pelo reconhecimento e todo amor oferecido sem esperar nada em troca. A senhora é um alicerce em minha vida e motivo de muito orgulho. Te amo!

Ao meu pai, que sempre se manteve por perto, e de maneira muito particular sabe manifestar a sua admiração pelos filhos. Te amo!

Ao meu irmão Prof. Dr. Hugo de Carvalho Sobrinho, por todo o poio até aqui, pelo excelente pesquisador, sonhador e enorme coração. Muito deste trabalho se deu pelo seu incentivo. Te amo!

À minha irmã Gislaine Maria de Carvalho Sobrinho, que com sua sabedoria sabe dar força, direção, e demonstração de afeto nos momentos mais necessitados. Te amo!

À minha irmã Vitória Gabrielly de Carvalho Sobrinho, a qual sempre cuidei e desejei o melhor, que acabou por me cuidar e me apoiar em qualquer momento, da maneira certa e sem hesitar. Você me dá muito orgulho, Te amo!

Aos meus queridos irmãos Paulo Henrique de Carvalho Sobrinho, Alexandre de Carvalho Sobrinho(in memoriam) pela participação neste trabalho, mesmo que indiretamente, pela torcida e apoio das diferentes formas, para que ele fosse concebido. Amo vocês!

Aos meus sobrinhos, Manuela Carvalho de Alcântara e Théo Carvalho de Alcântara, por chegarem em nossas vidas em um momento tão assertivo e necessário. Fomos presenteados por vocês dois!

À minha companheira para a vida Daniela Mamede, que soube construir e gerenciar uma forte parceria, com sustentação e compreensão manifestadas nas mais diversas situações. Te amo!

Ao Prof. Dr. Waldeyr Mendes Cordeiro da Silva, pela direção, conhecimentos de base, atenção, acompanhamento e amizade. Eternamente grato!

Ao Prof. Dr. Daniel Alves da Silva, pela oportunidade de participação em sua equipe, grande contribuição no meu aperfeiçoamento técnico e acadêmico, amizade, confiança e suporte nas mais diversas situações. Obrigado pela parceria.

Aos meus grandes amigos Lucas Martins Arruda e Geraldo Alves Pereira Júnior, que estiveram sempre presentes e que nunca mediram esforços para ajudar nas adversidades encontradas nos últimos oito anos da minha vida. Vocês também são meus irmãos!

Aos meus cunhados e amigos Geisson Esteves de Alcântara, Gleiser Mateus Ferreira Valério, e Eva Taynara Santana da Silva, pela torcida, suporte, e por estarem na família contribuindo para a constante evolução. Minha forte admiração e amor por vocês.

Agradeço o apoio técnico do Laboratório de Tecnologias da Tomada de Decisão - LATITUDE, da Universidade de Brasília, que conta com apoio do CNPq - Conselho Nacional de Pesquisa (Outorgas 312180/2019-5 PQ-2, BRICS2017-591 LargEWiN e 465741/2014-2 INCT em Cibersegurança), da CAPES - Coordenação de Aperfeiçoamento do Pessoal de Nível Superior (Outorgas 23038.007604/2014-69 FORTE e 88887.144009/2017-00 PROBRAL), da FAP-DF - Fundação de Amparo à Pesquisa do Distrito Federal (Outorgas 0193.001366/2016 UIoT e 0193.001365/2016 SSDDC), do Ministério da Economia (Outorgas 005/2016 DIPLA e 083/2016 ENAP), da Secretaria de Segurança Institucional da Presidência da República do Brasil (Outorga ABIN 002/2017), do Conselho Administrativo de Defesa Econômica (Outorga CADE 08700.000047/2019-14), da Advocacia Geral da União (Outorga AGU 697.935/2019), do Ministério das Cidades, (Outorga MC 01/2019), do Ministério da Justiça e Segurança Pública, (Outorga MJSP 01/2019) e dos Decanatos de Pesquisa e Inovação e de Pós-Graduação da Universidade de Brasília (DPI/DPG/UnB).

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

A federação em nuvem surgiu para estender os recursos disponíveis entre diferentes provedores de nuvem, interconectados para aumentar a disponibilidade de maneira transparente e ilimitada para o usuário final. As plataformas de orquestração em nuvem se tornaram uma forma de gerenciar as demandas por alto poder computacional em diferentes provedores, que executam aplicativos que demandam alto consumo de memória e/ou processamento, tais como os *workflows* de Bioinformática. A grande quantidade de recursos disponíveis entre vários provedores em uma federação torna difícil escolher qual é o mais adequado para determinados *workflows*. Este trabalho propõe um Serviço de Predição de Recursos por Aprendizado de Máquina, denominado sPCRAM. O sPCRAM utiliza um modelo de aprendizado de máquina combinado com uma meta-heurística GRASP para dimensionar os recursos de forma transparente e adequada, determinando o custo monetário e o tempo de execução antes da execução do *workflows*. O sPCRAM permite que o usuário defina de forma interativa o tipo de execução, calibre o tempo e o custo. Os resultados demonstram que o sPCRAM pode estimar adequadamente o tempo de execução e o custo dos recursos de federação em nuvem, em média, 97,70% mais rápido do que a técnica de força bruta para seleção de recursos.

Palavras-chave: Federação da nuvem, Previsão de Recursos, Aprendizado de Máquina, Meta-heurística GRASP, Workflows de Bioinformática.

Abstract

Cloud federation emerged to extend the resources available across different cloud providers, interconnected to increase availability in a transparent and unlimited way for the end-user. As cloud orchestration platforms if needed a way to manage how demands for high computational power in different providers that run applications that demand high consumption of memory and/or processing, such as the workflows of Bioinformatics. A large amount of resources available across multiple providers in a federation makes it difficult to choose which one is best suited for certain workflows. This work offers a Machine Learning Resource Prediction Service, called sPCRAM. sPCRAM uses a machine learning model combined with a GRASP metaheuristic to transparently and appropriately size resources, determining the monetary cost and execution time before executing the workflows. The sPCRAM allows the user to interactively adjust run type, gauge or time, and cost. The results demonstrate that sPCRAM can estimate the runtime and cost of cloud federation resources on average 97.70% faster than the brute force technique for resource selection.

Keywords: Cloud Federation, Resource Prediction, Machine Learning, GRASP Metaheuristic, Bioinformatics Workflows.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AD Árvore de Decisão.

AM Aprendizado de Máquina.

API *Application Programming Interface*.

AWS *Amazon Web Services*.

CP Conselho de Preditores.

CPU Unidade Central de Processamento.

CSV Valores Separados por Vírgulas.

DNA Ácido Desoxirribonucleico.

DSTAT *Versatile Resource Statistics Tool*.

DT *Decision Tree*.

EAM Erro Absoluto Médio.

EC2 *Elastic Compute Cloud*.

EM Erro Médio.

EPMA Erro Percentual Médio Absoluto.

FA Floresta Aleatória.

FIV Fator de Inflação de Variância.

GCE Google Compute Engine.

GCP *Google Cloud Platform*.

GRASP Procedimento de Pesquisa Adaptativa Aleatória Gulosa (do inglês, *Greedy Randomized Adaptive Search Procedure)*.

HTTP *Hyper Text Transfer Protocol*.

HTTPS *Hyper Text Transfer Protocol Secure*.

IA Inteligência Artificial.

IaaS *Infraestructure-as-a-Service*.

IoT Internet-das-Coisas.

JSON *JavaScript Object Notation*.

JWT *JSON Web Token*.

MAE *Mean Absolute Error*.

MAPE *Mean Absolute Percentage Error*.

ME *Mean Error*.

MLP *Perceptron* Multicamadas (do inglês, *Multi-layers Perceptron*).

MVS Máquina de Vetores de Suporte.

NIST Instituto Nacional de Padrões e Tecnologia dos Estados Unidos.

P2P *Peer-to-peer*.

PaaS *Platform-as-a-Service*.

QoS Qualidade do Serviço.

RAM Memória de Acesso Aleatório.

REMQ Raiz do Erro Médio Quadrático.

REST *Representational State Transfer*.

RF *Random Forest*.

RML Regressão Multilinear.

RMSE *Root Mean Squared Error*.

RNA Ácido Ribonucleico.

RNAs Redes Neurais Artificiais.

RPC Chamadas de Procedimento Remoto.

RVS Regressão de Vetor de Suporte.

SaaS *Software-as-a-Service*.

SLA Acordo de Nível de Serviço.

sPCRAM Serviço de Predição de Custos e Recursos Computacionais com Aprendizado de Máquina.

SQL Linguagem de Consulta Estruturada.

SVM *Support Vector Machine*.

SVR *Support Vector Regression*.

TC *Task Coordinator*.

TE *Task Executor*.

VM Máquina Virtual.

WWSVM Máquina de Vetor de Suporte de *Wavelet* Ponderada.

XaaS *Everything-as-a-Service*.

Capítulo 1

Introdução

O paradigma de entrega de serviços sob demanda influenciou no direcionamento tomado pela computação no que tange o acesso aos seus serviços. A principal motivação para este modelo está na entrega do serviço baseada em seu consumo. Desta forma, a Computação em Nuvem é um paradigma computacional que provê este tipo de modelo de serviço para o usuário final, de modo a atender sua necessidade atual, sem qualquer limitação geográfica, com baixo custo, transparente e flexível [15].

A necessidade de aumento da variabilidade e da disponibilidade dos serviços de nuvem concebeu o conceito de federação de nuvens. As nuvens federadas são conjuntos de nuvens que possuem todos os seus recursos gerenciados por meio de uma interface comum, o que torna possível gerenciar as diversas demandas de forma compartilhada com todas as nuvens da federação, atendendo a demanda computacional [97].

A demanda por recursos computacionais de alto desempenho não se restringe somente a grandes corporações. Pesquisas que demandam experimentos complexos como os *workflows* da área de Bioinformática necessitam de alto poder de processamento. Esses *workflows* consomem e produzem grande quantidade de dados, além de quantidades significativas de recursos computacionais [100]. Além disso, são descritos como um processo que consiste em uma sequência encadeada de atividades, nas quais as saídas de uma atividade constituem-se em entradas para a próxima, até que o fluxo do processo termine [28].

A necessidade de serviços de predição que auxiliem os usuários na escolha correta dos recursos a serem demandados em um *workflow* surge com a utilização da computação em nuvens federadas. A escolha eficiente de recursos traz melhorias nos custos e na redução de falhas durante o processamento de um *workflow* de Bioinformática, por exemplo.

Um modelo de predição que utiliza métodos estatísticos foi desenvolvido e integrado à plataforma BioNimbuZ [98]. O BioNimbuZ é uma plataforma de federação que garante a integração entre diferentes nuvens de maneira simples, dinâmica e transparente. O

modelo proposto em [100] tem o objetivo de auxiliar o usuário na predição de tempo e no dimensionamento dos recursos que são utilizados durante a execução do *workflow* na plataforma BioNimbuZ [75], por meio de um modelo estatístico.

1.1 Motivação

O serviço de predição de recursos, utilizado na primeira versão da plataforma BioNimbuZ, faz uso de um modelo estatístico e obteve resultados satisfatórios, como pode ser observado em [100]. Diante disso, é interessante uma implementação que utiliza modelos de predição por meio de aprendizado de máquina, obtendo o aumento da eficiência na escolha a partir da necessidade de cada usuário, uma vez que o BioNimbuZ 2 carece de um serviço de predição.

Assim, o serviço de predição proposto neste trabalho pretende ser um auxílio mais preciso ao usuário, por meio de técnicas de aprendizado de máquina, na escolha mais adequada a realidade de cada usuário, ou seja, será possível variar entre custo e performance, escolhendo entre um baixo custo e maior tempo de execução ou alto custo com menor tempo de execução de um *workflow* de Bioinformática.

1.2 Definição do Problema

A ausência de estimativas de demanda de recursos computacionais, ou até mesmo estimativas imprecisas, oferece um risco e pode trazer resultados insatisfatórios na execução de um *workflow* de Bioinformática em um ambiente de computação em nuvem, sendo agravado em nuvens federadas. Desta forma, ferramentas para estimar com melhor precisão as necessidades de recursos computacionais tornam-se necessárias para a execução de *workflows* para que não causem prejuízo financeiro e de produtividade a usuários que, por inviabilidade financeira ou de projeto, não selecionam os recursos computacionais adequados para a execução de seus *workflows*, uma vez que o executor de um *workflow* científico de Bioinformática pode não possuir conhecimentos técnicos para mensurar suas necessidades.

De acordo com [100], a escolha ideal desses recursos não é uma tarefa trivial para os usuários que operam tais *workflows*, pois existem inúmeras possibilidades de alocação de recursos, o que resulta em diferentes custos financeiros e tempo diferente para cada execução. Assim, as execuções se tornam imprevisíveis, o que pode levar a resultados inesperados, imprecisos e com um alto custo para o usuário [20].

Neste sentido, construir *workflows* científicos eficientes é uma tarefa complexa, na qual a disponibilidade de recursos pode ser decisiva para o sucesso de um projeto de

Bioinformática complexo e caro computacionalmente. Portanto, um serviço de predição é fundamental para viabilizar o projeto do *workflow* de Bioinformática aos usuários e na escolha dos recursos de forma eficiente, evitando prejuízos no projeto [100].

Logo, este trabalho propõe um serviço, que esteja acoplado aos microsserviços do ambiente de nuvens federadas do BioNimbuZ 2 (ver Capítulo 4), que selecione o recurso, a partir da diversidade de máquinas que um ambiente de nuvem federada disponibiliza, mais indicado para atender a necessidade do usuário, evitando o sub ou super dimensionamento de recursos.

1.3 Questão de Pesquisa

Executar *workflows* científicos sem qualquer conhecimento sobre a quantidade de recursos e/ou tempo necessário para sua conclusão pode acarretar em problemas de dimensionamento para projetos de Bioinformática. Tais problemas podem ser, por exemplo, a extrapolação no tempo de execução e/ou recursos disponíveis no projeto de Bioinformática; a subutilização dos recursos selecionados; a falta de previsão sobre o projeto de Bioinformática; entre outros problemas relacionados ao dimensionamento do tempo de execução e/ou recurso, o que acarreta na má gestão dos recursos.

Assim sendo, tem-se como questão de pesquisa: como definir um serviço de predição de recursos computacionais que utilize modelos de aprendizado de máquina para predizer o tempo, em conjunto da meta-heurística GRASP, para dimensionar o recurso adequado, de execuções de *workflows* científicos de Bioinformática, para uma arquitetura de federação de nuvens?

1.4 Objetivos

Além de modelos estatísticos, é possível aplicar o aprendizado de máquina para prover um processo de predição de tempo e dimensionamento de recursos mais eficiente, que são utilizados durante a execução do *workflow* de Bioinformática [104]. O aprendizado de máquina é um subcampo da Inteligência Artificial (IA), e fornece o estudo de reconhecimento de padrões e teoria sobre o aprendizado computacional [83].

Neste sentido, o principal objetivo deste trabalho é propor um serviço, que utilize modelos de aprendizado de máquina supervisionado e a meta-heurística GRASP, que auxilie os usuários na escolha adequada de recursos a partir de uma plataforma de nuvem federada gerenciada por meio do BioNimbuZ 2.

Assim, com base no objetivo principal, este trabalho possui os seguintes objetivos específicos:

- Analisar os dados históricos de execuções anteriores para que seja possível a geração de uma base de aprendizado em ambiente de nuvem federada;
- Propor e desenvolver um modelo de conselho de preditores por meio de técnicas de aprendizado de máquina supervisionado;
- Utilizar a meta-heurística GRASP para efetuar a seleção do recurso em conjunto do modelo de predição de tempo.
- Realizar um estudo de caso, no qual este modelo será desenvolvido e integrado aos microsserviços do ambiente de nuvem federada BioNimbuZ versão 2;

1.5 Estrutura do Trabalho

Este trabalho contém, além deste capítulo introdutório, mais seis capítulo. No Capítulo 2 são tratadas as estruturas de projetos com *workflows* científicos de Bioinformática, suas características e alguns softwares utilizados.

O Capítulo 3 apresenta a computação em nuvem, suas características, algumas arquiteturas de referência, e seus modelos de serviço. Além disso, também é descrito o modelo de federação de nuvens, seus requisitos, benefícios e limitações, e descreve brevemente alguns orquestradores de nuvens.

Na sequência, o Capítulo 4 descreve a plataforma de nuvem federada BioNimbuZ versão 2, mostrando sua estrutura, organização, objetivos e as modificações sofridas desde a sua proposta original.

No Capítulo 5 são apresentados os modelos de aprendizado de máquina, suas categorias, além de modelos que são específicos para problemas de regressão.

Em seguida, o Capítulo 6 descreve o serviço de predição proposto neste trabalho. Para isso, são apresentados a estrutura do serviço proposto, o *workflow* utilizado como experimento, os modelos de predição implementados neste trabalho, as métricas de avaliação selecionadas, e a descrição da meta-heurística GRASP.

Por último, o Capítulo 7 apresenta as considerações finais e os trabalhos futuros.

Capítulo 2

Workflow Científico

Neste capítulo são apresentadas as estruturas básicas de um *workflow*, assim como a aplicação de tais estruturas em *workflows* científicos de Bioinformática. Para isso, a Seção 2.1 apresenta as considerações iniciais sobre os *workflows*. Em seguida, a Seção 2.2 apresenta os *workflows* científicos e suas estruturas básicas. A Seção 2.3 apresenta os *workflows* científicos de Bioinformática. A Seção 2.4 são descritas as partes presentes no *workflow* executado neste trabalho para a geração da base de dados de treinamento para o modelo de predição de tempo. Por fim, a Seção 2.5 apresenta as considerações finais deste capítulo.

2.1 Considerações Iniciais

Originado por volta da década de 70, o termo *workflow* era associado aos processos de automação presentes em escritórios, os quais objetivaram oferecer soluções para diminuir a geração e a distribuição de documentos em papel em uma organização.

Assim, o termo surgiu para representar a automação total ou parcial de um processo, nos quais informações ou tarefas são passadas de uma entidade para outra de acordo com um conjunto de regras, a partir de entradas e saídas bem definidas. Nesse caso, as saídas de uma parte do processo podem ser entradas para uma ou mais partes posteriores ao seguimento [29].

Atualmente, os *workflows* estão cada vez mais inseridos no contexto das pesquisas científicas, os quais passaram a ser chamados de *workflows* científicos. Esses *workflows* são formados por uma série de atividades estruturadas e cálculos que surgem na resolução de problemas científicos [1, 81]. Deste modo, os *workflows* podem ser utilizados na automação de experimentos computacionais que necessitam de grande capacidade de computação, e manipulam grande quantidade de dados, na maioria das vezes de forma distribuída. Assim, eles podem ser vistos como um processo automatizado que combina

dados e processos em um conjunto estruturado de passos, para implementar soluções computacionais para um problema científico [81].

Execuções de *workflows* científicos, geralmente, fazem uso de *softwares* que necessitam de grande quantidade de recursos computacionais, de processamento e de armazenamento, e por isso são adequadas para o ambiente de nuvem computacional e/ou federação de nuvens [75].

2.2 Estrutura de *Workflow* Científico

As estruturas dos *workflows* científicos são complexas e demandam grande quantidade de recursos computacionais de maneira confiável [75]. Ou seja, qualquer exceção que possa ocorrer na execução da estrutura por falta de previsão da dimensão do processamento, pode encadear em perdas de tempo e/ou recursos financeiros, o que pode inviabilizar um projeto de Bioinformática em diversos casos.

A computação em nuvem permite que os recursos computacionais sejam consumidos como serviços, o que facilita o processo de execução de um *workflow* científico. Esses serviços são concebidos, principalmente, a partir de tecnologias de virtualização, sendo um modelo no qual o poder de processamento, de armazenamento ou de rede são disponibilizados através da Internet e sob demanda. Isso permite que o custo do serviço seja taxado de acordo com a sua utilização.

Um *workflow* possui grande capacidade de definição de diversas combinações entre as tarefas, assim como seus dados de entrada e saída. Estes dados de entrada podem ser tanto dados parametrizados, ou resultados de outras tarefas presentes na estrutura do *workflow*. Neste sentido, mesmo possuindo estruturação e componentes, muitas vezes complexos, um *workflow* emerge de algumas estruturas básicas. Na Figura 2.1 é possível observar algumas estruturas simples utilizadas em *workflows* científicos.

A primeira estrutura é definida como um Processo, no qual uma entrada é geradora de uma saída. A segunda estrutura demonstra um encadeamento de processos chamado *Pipeline*, no qual um entrada encadeia em saídas que se tornam entradas para as tarefas subsequentes. Nas próximas estruturas o encadeamento pode ser efetuado de forma paralela, o que contribui para a agilidade na execução, conforme o objetivo do *workflow*.

Na estrutura de Distribuição de dados, uma entrada é fornecida, e após o processamento, várias saídas são efetuadas e distribuídas para elementos ou localidades diferentes. Este tipo de *workflow* é bastante utilizado quando existe a necessidade de quebra de elementos em partes menores, que podem servir como entrada para tarefas posteriores. Em contrapartida, na Agregação de Dados, muitas entradas são fornecidas, gerando ao final apenas uma saída. Já o modelo de Redistribuição de dados é mostrado como as entradas

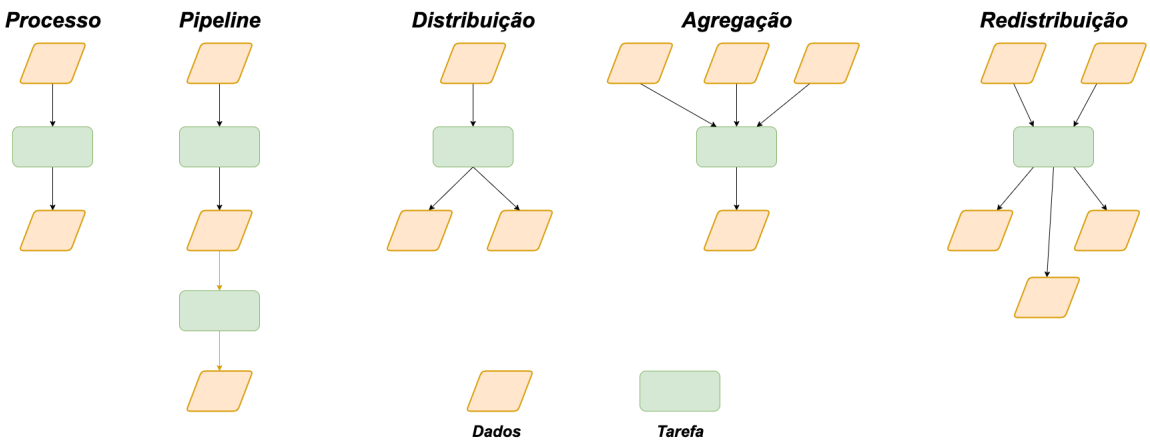

Figura 2.1: Estruturas Básicas de um *Workflow* [12].

são concatenadas em um momento em uma ou mais tarefas, e desencadeia em um ou mais dados de saídas, e pode ser definida como a união das estruturas anteriores.

Neste sentido, a utilização de todas as estruturas supracitadas é capaz de fornecer uma estrutura de controle que contribui na automatização de soluções para os mais diversos problemas que são estruturados a partir de um fluxo básico de processo, nas quais entradas são trabalhadas de forma a gerar saídas desejadas de maneira automatizada. Esta estrutura automatizada gera benefícios para um projeto de Bioinformática, quando este necessita de automatização de tarefas e/ou análise de dados, o que facilita a análise das diversas variáveis atreladas ao projeto, e contribui para o cumprimento dos seus objetivos.

2.3 *Workflows* Científicos de Bioinformática

Os *workflows* podem ser utilizados em diversas áreas de estudo e trazem benefícios no que tange a automatização de processos. Na Informática, como uma área de estudo, os *workflows* dão suporte para o desenvolvimento de soluções e modelos de dados para a resolução dos mais diversos problemas.

A utilização de ferramentas computacionais para resolução de problemas biológicos deu origem ao termo Bioinformática. Neste contexto, a análise dos dados, obtidos a partir de ferramentas de sequenciamento automático, é realizada em diferentes fases ou passos. Em cada fase estão presentes ferramentas a serem utilizadas e, para cada pesquisa, existe uma gama de opções de combinação de diferentes ferramentas, cada uma com um objetivo específico que contribui para o projeto. Assim sendo, este fluxo de passos em conjunto com estas ferramentas é chamado de *workflow* científico de Bioinformática [103].

Projetos de Bioinformática estão atrelados ao processamento de dados biológicos e de sequenciamento de Ácido Desoxirribonucleico (DNA) ou Ácido Ribonucleico (RNA) [107],

que são denominadas *reads*. Estes tipos de sequenciamento consistem no processo de descoberta, para um determinado organismo, de qual é a sequência de bases nitrogenadas que forma cada fragmento de DNA e RNA que está sendo analisado [86]. Neste sentido, projetos genoma consistem em pesquisar cromossomos (DNA genômico), enquanto que projetos transcritoma consistem no estudo dos transcritos (RNA). A partir desses fragmentos, tanto de DNA quanto de RNA, diversos tipos de processos computacionais podem ser executados com foco no objetivo do projeto. Assim, ferramentas sequenciadoras de alto desempenho são selecionadas e passam a dar suporte computacional aos projetos de Bioinformática, as quais são inseridas de maneira estratégica no decorrer do processamento de um *workflow* científico de Bioinformática [86].

Portanto, os *workflows* possuem estruturas que permitem que sejam inseridos em projetos de Bioinformática, assim como em projetos com objetivos diversos. Com isso, a computação em nuvem, com sua capacidade de disponibilizar diversos recursos de forma heterogênea, contribui para que um projeto de Bioinformática seja eficiente, pois possibilita que diferentes máquinas virtuais geograficamente dispersas possam ser utilizadas pelo projeto de forma otimizada.

Desta maneira, predizer os recursos necessários para a execução de um *workflow*, sendo ele de alto desempenho ou não, torna-se importante para a economia de tempo e/ou recursos financeiros em um projeto de Bioinformática, assim como para se ter noção da viabilidade e das variáveis inerentes ao projeto. No entanto, estimar a quantidade necessária de recursos está longe de ser trivial. Assim, a má escolha na quantidade ou até mesmo na capacidade do recurso poderá produzir impactos negativos na execução do *workflow* e afetar o seu custo financeiro [20]. Desta forma, é necessário que estruturas de predição sejam desenvolvidas com o intuito de auxiliar o usuário final na escolha dos recursos de maneira eficiente.

Diante do exposto, este trabalho propõe um serviço capaz de predizer o tempo e o dimensionamento do recurso integrado à plataforma BioNimbuZ 2 (ver Capítulo 4), a partir de um ambiente de nuvens federadas (ver Capítulo 3) que forneça suporte ao usuário final na escolha de recursos para utilização em um projeto de Bioinformática.

2.4 *Workflow* Executado

A Figura 2.2 descreve a estrutura do *workflow* executado. Este *workflow* executa um experimento de RNA-Seq do fungo *Aspergillus Fumigatus* [61]. Foi selecionado por meio do trabalho de [99] e por se tratar de um *workflow* que possui uma variabilidade de programas sendo utilizados para execução. Este *workflow* possui as fases de Filtragem, Mapeamento, Montagem e Anotação. As ferramentas utilizadas para a execução foram

Prinseq [105] para Filtragem; Bowtie [60] e TopHat [113] para Mapeamento; o Trinity [40] para Montagem; e o Blast [34] para Anotação. Na Figura 2.2 são ilustrados os arquivos de entrada do *workflow*, representados com a cor verde no fluxo de execução, enquanto os arquivos de saída de um programa e/ou entrada para o programa subsequente estão ilustrados em azul escuro.

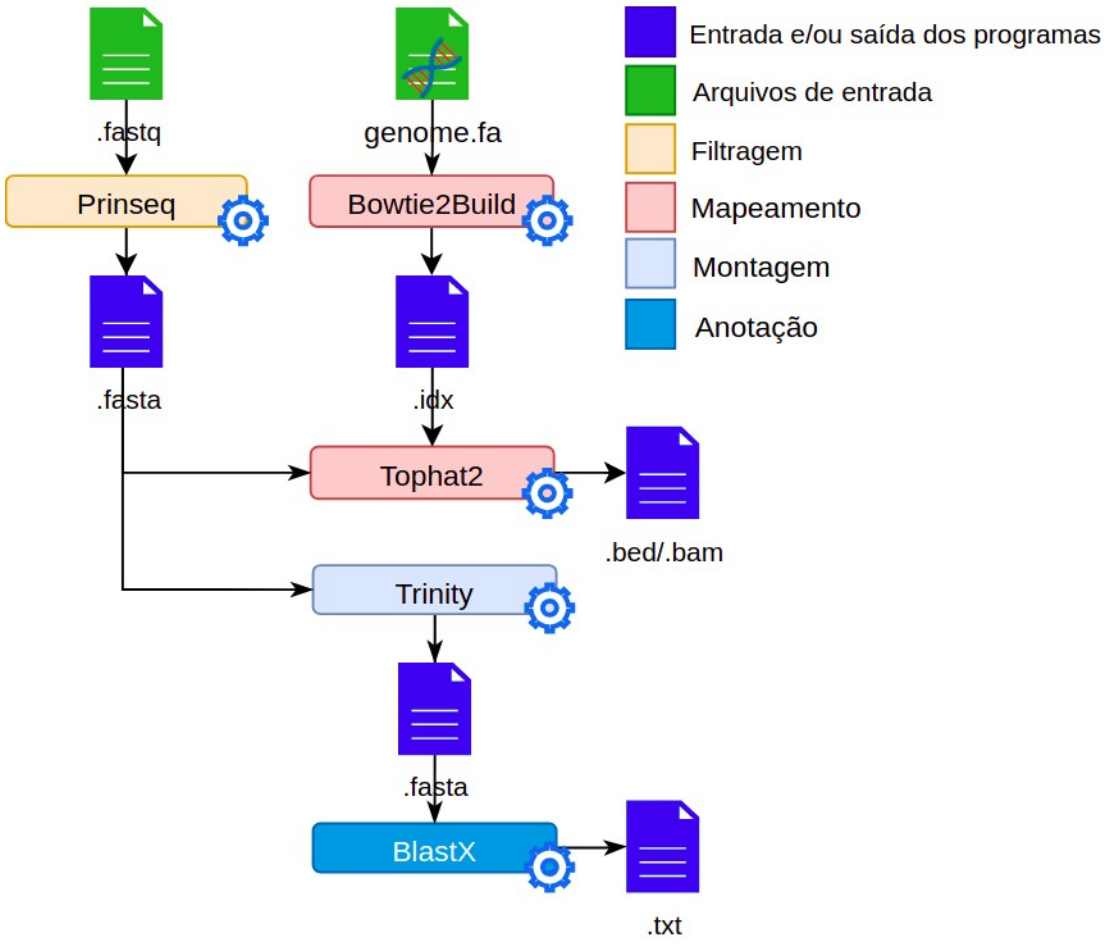

Figura 2.2: *Workflow* do Experimento de RNA-Seq do Fungo *Aspergillus Fumigatus* [61].

Foram executadas várias instâncias de um mesmo *workflow* de Bioinformática em diferentes provedores reais de nuvem, com o objetivo de obter dados de suas execuções para a construção da base de dados de treinamento (ver Capítulo 6) e posterior utilização nos modelos de aprendizado de máquina desenvolvidos neste trabalho.

Neste sentido, as saídas dos modelos serviram como base para o treinamento do conselho de preditores a ser detalhado na Seção 6.5. Este conjunto de modelos de aprendizado de máquina define as estimativas de tempo do *workflow* de Bioinformática proposto por este trabalho.

2.5 Considerações Finais

Gerenciar *workflows* científicos de maneira eficiente, garantir a disponibilidade de recursos e a realização dos objetivos inerentes à execução do *workflow*, é uma tarefa complexa, na qual muitas vezes a disponibilidade de recursos pode decidir entre uma execução bem ou mal sucedida de um *workflow* científico [32].

O poder computacional entregue pela computação em nuvem tornou-se importante para que os *workflows* científicos alcancem a automação durante seu processamento de forma flexível. A execução de *workflows* científicos de Bioinformática demanda alta disponibilidade de recursos computacionais de maneira confiável [100].

Na busca por melhor atender os consumidores dos serviços de nuvem, emergiu a necessidade de integrá-las, aumentando os recursos disponíveis dos serviços computacionais. Dessa integração de nuvens, o termo federação de nuvens surgiu, e pode ser definido como conjuntos de nuvens que possuem todos os seus recursos gerenciados por intermédio de uma interface conectada a todas elas, de maneira que, se uma nuvem não tiver recursos para atender determinada demanda, outras, que estejam com recursos disponíveis no momento, possam ser integradas, atendendo à demanda computacional [103].

A necessidade que os projetos de Bioinformáticas têm de demanda de recursos computacionais de maneira rápida e escalável, faz com que este tipo de estrutura obtenha benefícios de utilização em um contexto de nuvem federada, que será detalhada no Capítulo 3.

Capítulo 3

Computação em Nuvem e Federação

Neste capítulo são apresentados os conceitos relacionados a nuvem computacional, assim como a federação de nuvens. A Seção 3.1 apresenta as considerações iniciais deste capítulo. A Seção 3.2 apresenta alguns modelos de arquitetura de nuvem computacional. Em seguida, a Seção 3.3 descreve os modelos de serviço comumente disseminados na literatura, e a Seção 3.4 apresenta os modelos de implantação. A Seção 3.5 apresenta os requisitos para implantação de uma federação de nuvens, os benefícios e limitações de sua implantação, assim como modelos de orquestração de nuvens. Por fim, a Seção 3.6 apresenta as considerações finais deste capítulo.

3.1 Considerações Iniciais

O termo computação em nuvem define um paradigma de computação distribuída que surgiu como uma tendência para a provisão de recursos e serviços de forma escalável, rápida, flexível, elástica, com alta disponibilidade, e com baixo custo [35]. Diversos tipos de recursos podem ser disponibilizados, tais como memória, capacidade de processamento, de armazenamento, entre outros, todos fornecidos por meio da Internet [82]. Assim, há na literatura diversas definições de computação em nuvem. Algumas delas são descritas a seguir:

- Armbrust et al. [36] definem como a união de aplicações oferecidas como serviço pela Internet, com o hardware e o software localizados em *datacenters* de onde o serviço é provido;
- De acordo com Foster et al. [35], trata-se de um paradigma computacional altamente distribuído, direcionado por uma economia de escala, na qual poder computacional, armazenamento, serviços e plataformas abstratas, virtualizadas, gerenciadas e dina-

micamente escaláveis são oferecidos sob demanda para usuários externos por meio da Internet;

- Buyya et al. [15] descrevem como um tipo de sistema paralelo e distribuído, que consiste em uma coleção de computadores virtuais interconectados que são provisionados de forma dinâmica e vistos como um ou mais recursos computacionais unificados, baseados em acordos de nível de serviço estabelecidos entre provedor de recursos e o consumidor do mesmo;
- Mell e Grance [74] descrevem como um modelo que possibilita acesso, de modo conveniente e sob demanda, a um conjunto de recursos computacionais configuráveis (por exemplo, redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente adquiridos e liberados com mínimo esforço de configuração ou interação com o provedor de serviços;
- Marston et al. [70] definem a computação em nuvem como um modelo de serviço em Tecnologia da Informação em que serviços computacionais, tanto hardware quanto software, são entregues sob demanda para os usuários por intermédio de uma rede de comunicações de maneira autônoma, independente de dispositivo e localização. Os recursos necessários para prover os níveis de qualidade de serviço demandados pelos usuários são compartilhados, escaláveis dinamicamente, disponibilizados rapidamente, virtualizados e entregues com mínima interação do provedor de serviço. Os usuários pagam pelo serviço como um custo operacional sem impactar em qualquer despesa de capital inicial em ativos físicos.

Nesta linha, outra definição é estabelecida pelo Instituto Nacional de Padrões e Tecnologia dos Estados Unidos (NIST) [66], o qual define computação em nuvem como um modelo que permite acesso conveniente, e sob demanda a um conjunto compartilhado de recursos de computação facilmente configuráveis (aplicações, serviços, armazenamento, redes), que podem ser provisionados de maneira rápida, e liberados com gerenciamento ou interação mínimos com o provedor de serviços de forma direta.

3.2 Arquitetura de Nuvem Computacional

A visão geral da arquitetura de referência da computação em nuvem definida pelo NIST pode ser vista na Figura 3.1, que identifica o principais atores, suas atividades e funções na computação em nuvem [66]. O diagrama descreve de maneira genérica uma arquitetura de referência, e destina-se a facilitar o entendimento dos requisitos, usos, características e padrões de computação em nuvem [66].

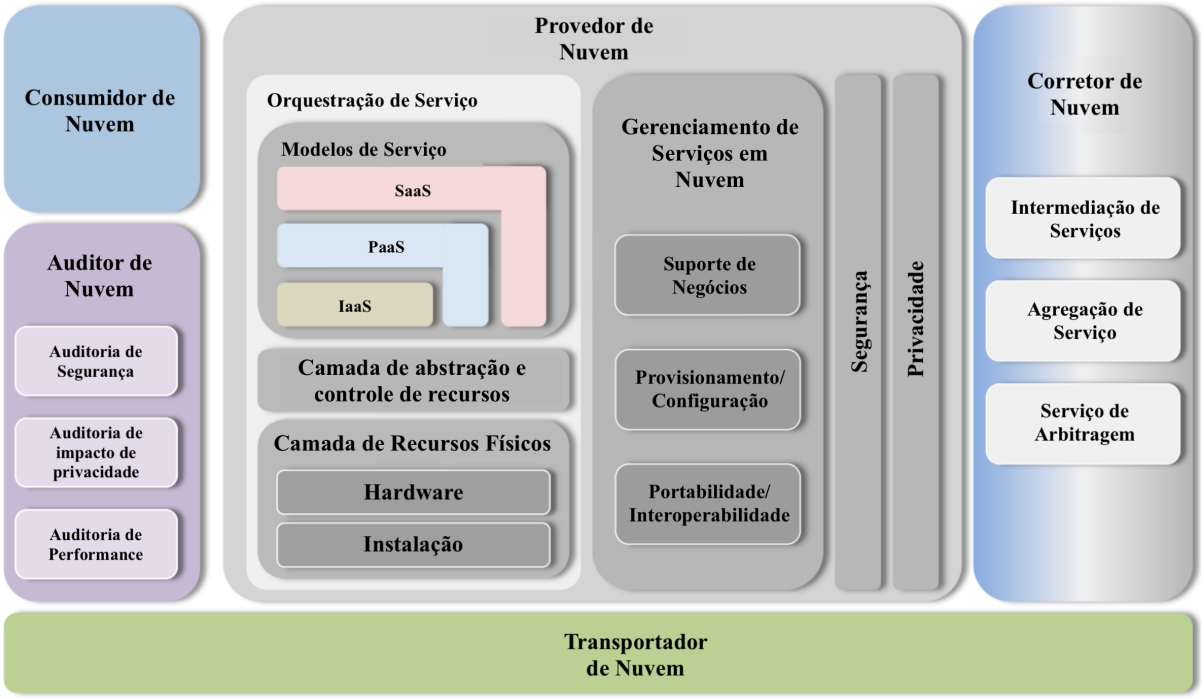

Figura 3.1: Arquitetura de Referência da Computação em Nuvem, adaptado de [66].

A arquitetura de referência da computação em nuvem, proposta pelo NIST, descreve cinco principais atores: Consumidor de Nuvem, Provedor de Nuvem, Transportador de Nuvem, Auditor de Nuvem e Corretor de Nuvem. Cada ator é uma entidade, podendo ser uma pessoa ou organização, que participa de uma transação ou processo e/ou executa tarefas no ambiente de computação em nuvem. Assim, as principais características desses atores, definidos pelo NIST [66], são:

- Consumidor de Nuvem: uma pessoa ou organização que mantém um relacionamento comercial e utiliza os serviços de provedores em nuvem;
- Provedor de Nuvem: uma pessoa, organização ou entidade responsável por disponibilizar serviços às partes interessadas;
- Auditor de Nuvem: uma parte que pode realizar uma avaliação independente dos serviços em nuvem, operações do sistema de informações, desempenho e segurança da implementação da nuvem;
- Corretor de Nuvem: trata-se de uma entidade que gerencia o uso, o desempenho e a entrega de serviços em nuvem, e negocia os relacionamentos entre os provedores de nuvem e os consumidores de nuvem;
- Transportador de Nuvem: é um intermediário que fornece conectividade e transporte de serviços em nuvem dos Provedores de Nuvem aos Consumidores de Nuvem.

Além disso, o modelo supracitado é proposto por um órgão que se destina a padronizar processos, e trata-se de uma visão de todo o processo que envolve a computação em nuvem. Contudo, é possível encontrar também outras definições na literatura, pois não há um padrão globalmente definido quanto a arquitetura de uma plataforma de nuvem.

Assim, este trabalho apresenta também a proposta publicada em [114], na qual os autores identificaram um modelo que pode ser visto na Figura 3.2. Nela são apresentados três atores: os Provedores de Serviço (*Service Providers*), os Provedores de Infraestrutura (*Infrastructure Providers*) e os Usuários de Serviço (*Service Users*). Em contraste com o modelo proposto pelo NIST, possui menos atores e abstrai algumas atividades de forma a facilitar o entendimento do modelo arquitetural.

Esta arquitetura de nuvem está dividida em três camadas distintas. A mais próxima do usuário possui os serviços disponibilizados. Ela possui uma plataforma para que os usuários finais do serviço tenham acesso às aplicações, na qual esta interface é acessada através da Internet. A camada mais baixa é a infraestrutura, na qual estão localizados os servidores, os *datacenters*, a rede, e toda a parte física que compõe a estrutura da nuvem.

A camada fornecida pelos Provedores de Infraestrutura provê facilidades para que o provedor de serviços consiga disponibilizar suas aplicações, sem ter que se preocupar com as individualidades do *hardware*. Isso faz com que cada ator trabalhe somente com o que é de sua competência e expertise, o que contribui para a elevação da qualidade do serviço [81].

Os provedores de serviços fazem a hospedagem de suas aplicações nas estruturas disponibilizadas pelos provedores de infraestrutura, sem necessidade de preocupação com as questões de estrutura física, delegando funções a outro ator presente na arquitetura, a fim de ganhar escalabilidade e flexibilidade. Os usuários de serviço acessam as aplicações que foram colocadas a disposição por meio da Internet pelos provedores de serviço.

3.3 Modelos de Serviço

A divisão dos serviços oferecidos no paradigma de computação em nuvem pode ser observada, popularmente, em três categorias, sendo *Software-as-a-Service* (SaaS), *Platform-as-a-Service* (PaaS), e *Infraestructure-as-a-Service* (IaaS). A distribuição desses modelos pode ser vista na Figura 3.3, e são descritos a seguir [66]:

- **IaaS**: nesta categoria são oferecidos serviços de infraestrutura, podendo ser desde capacidade de processamento, de armazenamento ou até de máquinas virtuais completas e com sistema operacional desejado. Assim, são exemplos de serviços desse modelo o *Elastic Compute Cloud* (EC2) [4] e o Google Compute Engine (GCE) [39].

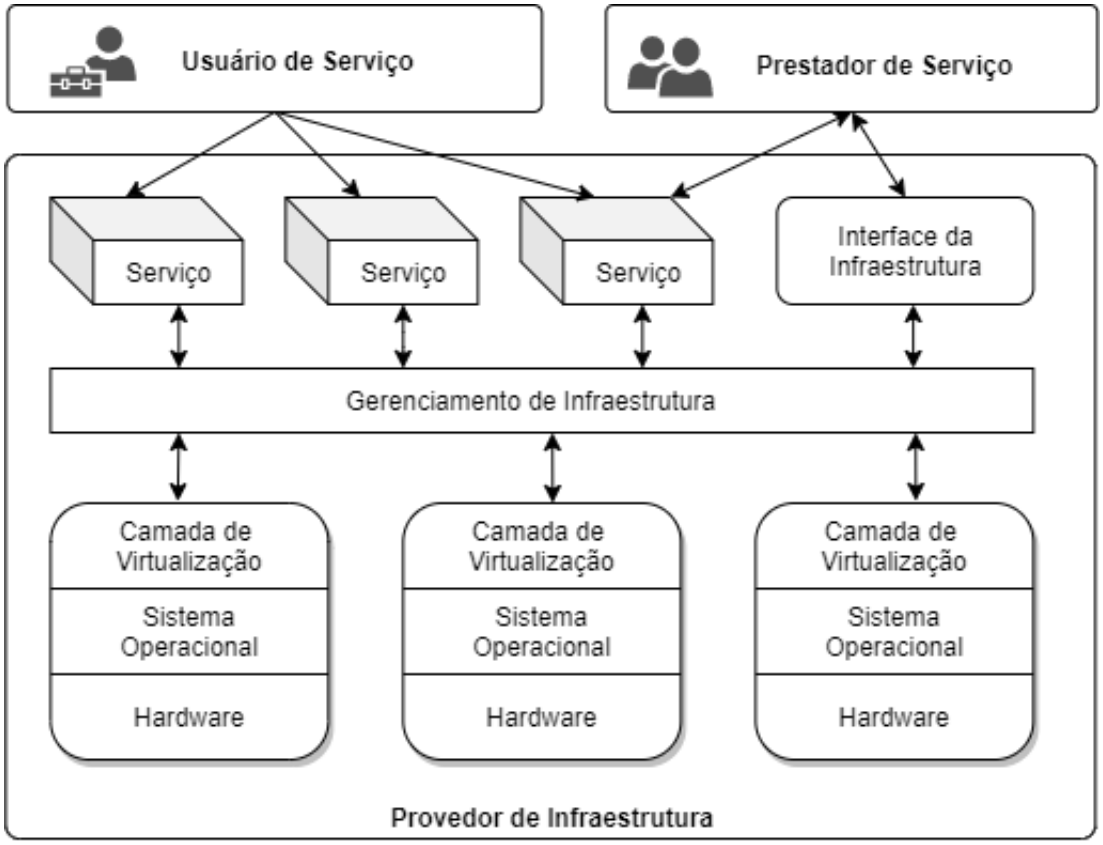

Figura 3.2: Arquitetura de Nuvem Computacional, adaptado de [114].

Ambos os serviços disponibilizam máquinas virtuais com diferentes configurações para memória RAM, CPU, armazenamento, entre outros recursos de infraestrutura.

- **PaaS**: são um conjunto de serviços de nuvem que fornecem um ambiente para desenvolvimento, gerenciamento, implementação e integração de aplicativos. Esse tipo de serviço é específico para o desenvolvimento de softwares, e permite que soluções informatizadas sejam criadas sem que haja a necessidade dos desenvolvedores configurarem um ambiente de desenvolvimento ou elementos de infraestrutura. O *CloudFoundry* [21] e o *AppEngine* [39], disponibilizam plataformas via web para desenvolvimento e implantação de aplicativos.
- **SaaS**: esta categoria é definida a partir da disponibilização, pelos provedores de serviços, de aplicações aos usuários comuns, cobrando pelo seu uso ou de forma gratuita. O *Office* 365 [77] é um exemplo de software disponibilizado como serviço em nuvem.

Estes modelos possuem relações e podem ser dispostos em camadas, nas quais as camadas superiores possuem dependência das camadas inferiores. Assim, os serviços fornecidos pelo IaaS são disponibilizados, principalmente, por meio de tecnologias de

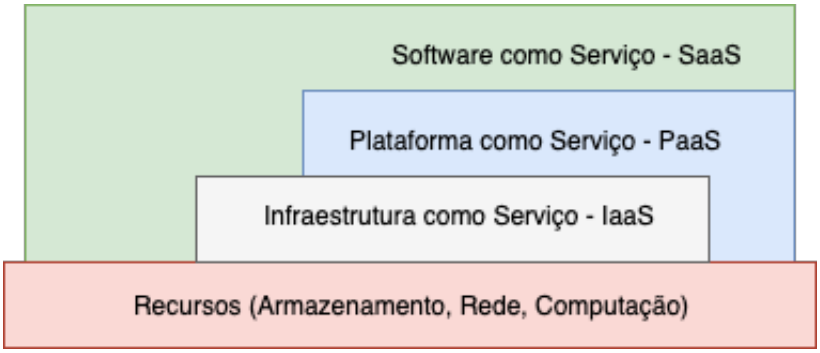

Figura 3.3: Modelos de Serviço de Nuvem Computacional, adaptado de [66, 112].

virtualização, e são implantados diretamente nos recursos físicos de *hardware*. A partir disso, os serviços fornecidos pelo PaaS podem ser implantados diretamente nos serviços IaaS ou no hardware e, por último, os serviços SaaS, que podem ser implantados nos serviços dos modelos PaaS, IaaS, ou diretamente nos recursos de hardware.

Contudo, com o constante amadurecimento das tecnologias disponibilizadas pelos provedores de nuvem e a partir da integração entre tais provedores, surgem modelos de serviços com foco em tarefas específicas. Assim, os provedores passaram a denominar seus serviços baseando-se no tipo de tarefa disponibilizada aos clientes. Desta forma, a nomenclatura passou a ser constantemente revista na literatura e nos provedores de nuvem, e, em alguns casos, não é mantido um padrão global de nomes.

O termo *Everything-as-a-Service* (XaaS) tem sido usado para definir essa classe de modelos de serviço em nuvem [30]. Assim, por não existir ainda uma definição global para os nomes dos modelos de serviços disponibilizados pelos provedores, são elencados a seguir alguns modelos de serviços com siglas diferentes e objetivos parecidos e/ou outros com siglas idênticas e objetivos totalmente diferentes [27]:

- **STaaS**: *Storage as a Service*, armazenamento como serviço [101];
- **DaaS**: *Database as a service*, banco de dados como serviço [26];
- **DaaS**: *Desktop as a service*, área de trabalho como serviço [18];
- **DaaS**: *Data as a service*, dados (informações) como serviço [117];
- **BaaS**: *Backup as a Service*, cópia de segurança como serviço [33];
- **DRaaS**: *Disaster Recovery as a Service*, recuperação de desastres como serviço [2];
- **SecLaaS**: *Secure logging-as-a-service*, registro seguro como serviço [121];
- **ELPaaS**: *Event Log Privacy as a Service*, privacidade do *log* de eventos como um serviço [9];

- **MaaS**: *Monitoring as a Service*, monitoramento como serviço [76].

3.4 Modelos de Implantação

Os diferentes tipos de nuvem, de acordo com a abrangência de acesso por usuários ou organizações, são definidos como Modelos de Implantação. Essa diferenciação entre tais modelos pode ser descrita como: Nuvens Privadas, Comunitárias, Públicas, e Híbridas. As características desses diferentes modelos são:

- Nuvem Privada: é uma infraestrutura de nuvem que é provisionada para uso exclusivo por uma única organização. Este modelo pode ser gerenciado e operado pela própria organização proprietária da infraestrutura, por terceiros ou por uma combinação deles. Além disso, este modelo pode existir dentro ou fora das instalações da organização detentora da infraestrutura [66];
- Nuvem Comunitária: trata-se de uma nuvem que possui infraestrutura que é provisionada para uso exclusivo por uma comunidade específica de consumidores de organizações que compartilham preocupações (por exemplo, missão, requisitos de segurança, política e considerações de conformidade) ou objetivos. Ela pode ser gerenciada por uma ou mais organizações da comunidade, por terceiros ou por uma combinação delas [96];
- Nuvem Pública: ainda de acordo com [66], uma nuvem pública possui a sua infraestrutura de nuvem para provisionamento pelo público em geral. Pode ser de propriedade de uma organização com fins lucrativos, gerenciada e operada por ela, ou organização acadêmica ou governamental, ou alguma combinação dessas, e pode possuir estruturas gratuitas ou pagas baseadas em seu uso;
- Nuvem Híbrida: neste modelo de implantação há uma combinação de dois ou mais dos modelos supracitados, obtido por tecnologias ou padrões que permitem a portabilidade de dados ou aplicações [96].

3.5 Federação de Nuvens

A necessidade de maior poder computacional exigido por determinadas aplicações, presentes tanto no meio científico quanto no corporativo, fez com que a Federação de Nuvens se manifestasse como um modelo computacional, o qual permite que provedores de nuvem possam estender e aumentar seus recursos. Isso é possível por meio de acordos estabelecidos, a partir de uma integração com outras nuvens. Os provedores obtêm ganhos a

partir da economia de escala, eficiência na utilização de seus recursos, além do aumentos de suas capacidades de infraestrutura [19].

Contudo, alguns autores na literatura utilizam termos diferentes para definição de federação de nuvens. Assim sendo, neste trabalho os termos usados seguem as definições especificadas nesta seção.

Inicialmente, a federação de nuvens concentra-se em definir a participação voluntária na interconexão com outras nuvens para possibilitar a alocação de recursos com o objetivo de estendê-los de forma automática, sem intervenção de mediadores [7, 75, 112].

Com isso, a relação na qual muitas nuvens são utilizadas para disponibilização de recursos, sem que estas tenham conhecimento entre si, é definida com *Multi-cloud*. Neste tipo de conexão entre as nuvens, a participação não é feita de forma voluntária, ou seja, existe um terceiro ator que é responsável por gerenciar os recursos das diferentes nuvens, seja o próprio usuário alocando recursos diretamente em uma nuvem, seja por meio de um agente intermediário entre as nuvens e o usuário final. Este agente recebe o nome de *Cloud-broker* [7, 112]. É importante ressaltar que a utilização do *Cloud-broker* é proposto por [14] em seu modelo de arquitetura para federação de nuvens.

Assim, qualquer relação existente entre nuvens, com inclusão das duas anteriores, de forma que seja possível a extensão dos recursos de determinada nuvem, é denominada de *Intercloud*. Logo, o termo *Intercloud* tem sido usado para estabelecer uma estrutura de nuvens-de-nuvens [19].

Diante do exposto, é necessário observar dois tipos de comportamentos na comunicação entre, pelo menos, duas nuvens [37]:

- *Insourcing*: o provedor que possui recursos, para evitar um alto custo de infraestrutura, fornece os seus recursos para provedores externos, e atende as necessidades de provedores com recursos escassos;
- *Outsourcing*: este comportamento auxilia na disponibilidade dos serviços da nuvem, com isso, ao perceber que os recursos da nuvem estão se esgotando, é feita uma busca por novos recursos em outros provedores. Esta estratégia evita a perda de consumidores dos serviços da nuvem.

A Figura 3.4 apresenta uma maneira de como os recursos são alocados em uma federação de nuvens, destacando o histórico e a evolução do paradigma de computação em nuvem. As nuvens estrangeiras A e B disponibilizam recursos para nuvens externas, efetuando o processo de *Insorucing*, e com isso, a nuvem local pode adquirir recursos disponibilizados pelas nuvens A ou B, efetuando o processo de *Outsourcing*.

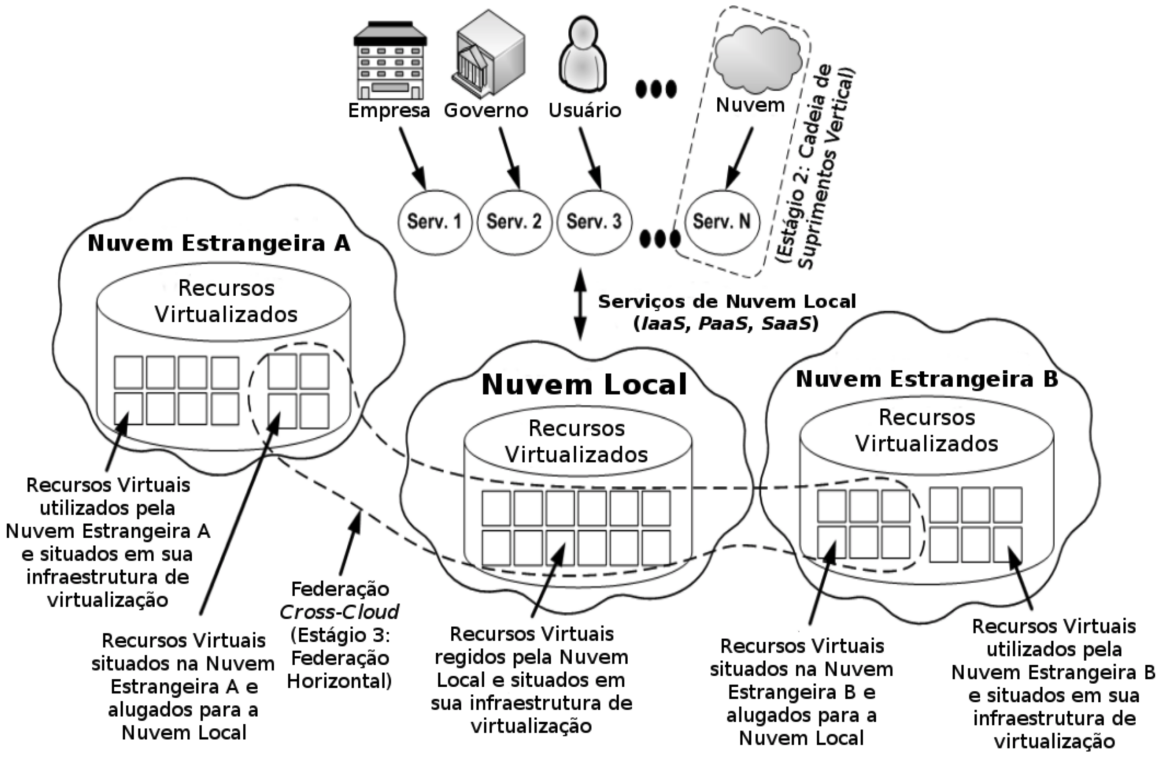

Figura 3.4: Arquitetura para Federação de Nuvens, proposta por [19].

3.5.1 Requisitos para Criação

As diferenças entre as nuvens, na qual cada uma possui suas particularidades, tanto em software quanto em hardware, faz com que a implantação de um modelo de federação de nuvens não seja uma tarefa simples. Desta maneira, alguns requisitos tornam-se necessários para que a federação de nuvens seja implantada [14, 19]:

- Automatismo e Escalabilidade: diz respeito a uma nuvem que utiliza mecanismos de descoberta para que seja possível escolher qual atende suas necessidades no momento, dentre as nuvens existentes e conhecidas, e deve ser ágil quando existirem mudanças;
- Segurança Interoperável: descreve que é necessária uma integração entre diversas tecnologias de segurança, de forma que uma nuvem não precise mudar políticas de segurança internas para que a comunicação ocorra, ou seja, a comunicação segura deve-se manter em um padrão ou em um limiar de padrões conhecidos por todas as nuvens;
- Previsão de Comportamento da Aplicação: para que boas decisões possam ser tomadas quanto à alocação de recursos, dimensionamento dinâmico, armazenamento e/ou largura de banda, e de maneira eficiente, é importante que, na criação de uma

federação de nuvens, o sistema seja capaz de prever o comportamento das aplicações. Desta maneira, um modelo deve ser construído para tentar predizer tais comportamentos. Logo, com o ajuste das variáveis e dos serviços a partir de modelos estatísticos e padrões de utilização, espera-se que o modelo seja o mais próximo possível da realidade. Alguns modelos que fazem previsão de demanda ou carga de trabalho podem ser vistos em [54, 57, 65, 122];

- Mapeamento Flexível de Recursos e Serviços: é importante que o sistema seja eficiente, uma vez que o custo deve ser levado em consideração em qualquer projeto, sempre tentando encontrar a melhor configuração de hardware e de software que atenda às demandas de qualidade de serviço que foram estabelecidas. De acordo com [100], é uma tarefa complexa devido ao comportamento imprevisível das aplicações e dos serviços que são utilizados em ambiente de nuvem computacional;
- Modelo Econômico Impulsionado por Técnicas de Otimização: ao tentar encontrar a melhor combinação entre serviços e planos disponíveis de forma eficiente, encontra-se um problema de otimização combinatória. Os modelos de otimização visam melhorar tanto os recursos centralizados (utilização, disponibilidade e incentivo) quanto os centrados nos usuários (tempo de resposta, o orçamento gasto e a justiça) [100];
- Integração e Interoperabilidade: questões de segurança, como dados sensíveis, fazem com que algumas instituições não disponibilizem de forma integral suas aplicações ou seus dados em um ambiente de nuvem. Desta forma, tais instituições optam por uma interação entre elementos em um ambiente conhecido/gerenciado e o ambiente de nuvem, de maneira a isolar deus dados do ambiente da federação. Assim, parte dos serviços podem ser alocados em nuvem, enquanto outros poderão estar restritos ao ambiente desta instituição;
- Monitoramento Escalável dos Componentes do Sistema: é importante que qualquer monitoramento seja feito de maneira distribuída e escalável, para que não existam gargalos, de desempenho e de confiabilidade causados pelas rotinas de monitoramento constante dos integrantes da federação.

3.5.2 Benefícios e Limitações

Por meio da gama de recursos disponibilizados pela federação de nuvens, benefícios como o aproveitamento e extensão dos recursos, economia de escala, alta disponibilidade e baixa latência podem ser facilmente observados. Por outro lado, a dificuldade no monitoramento dos recursos ou serviços provenientes de uma federação de nuvens cresce a medida que mais nuvens são inseridas na federação e estão situadas em diferentes regiões. Isso provoca

aumento da latência na rede de comunicação [37]. Diante disso, os principais benefícios presentes na utilização da federação de nuvens são [37, 112]:

- Eficiência no aproveitamento de recursos: recursos subutilizados que são alugados reduzem os custos atrelados a manutenção de infraestrutura de uma nuvem;
- Extensão de recursos: o esgotamento de recursos pode comprometer a disponibilidade e o Acordo de Nível de Serviço (SLA) feito entre o usuário e o provedor. Assim, estender os recursos pode aumentar sua disponibilidade a partir de solicitações feitas a outras nuvens, o que evita a negação de serviços aos usuários por motivo de falta de recursos;
- Economia de escala: a infraestrutura pode ser reduzida, visto que as empresas contratam recursos extras em momentos de extrema necessidade, e isso reduz os custos com espaço físico e até mesmo energia;
- Alta disponibilidade e baixa latência: sistemas implantados em nuvens diferentes, e podendo estar em diferentes regiões, permitem a redundância de informações e a aproximação dessas informações dos usuários, o que provê maior velocidade na resposta;
- Questões legais: dados e sistemas que dependem de questões legais específicas são beneficiados ao serem implantados em regiões diferentes.

Contudo, a manutenção dos recursos é outro fator dificultante na implementação de uma federação de nuvens, pois eles diferem-se entre os diversos provedores de nuvens, e garantir que todas as nuvens participantes estejam de acordo é uma tarefa complexa. Desta forma, é comum que grandes provedores imponham altos valores na quebra desses acordos, para que não haja grande vazão dos clientes, o que se torna um problema para pequenas empresas, pois custear tais acordos pode inviabilizar seus negócios [7].

Outro fator é a carência de padrões abertos para comunicação entre as diferentes nuvens presentes na federação. A comunicação definida por cada provedor pode não ser utilizada pelo restante (por exemplo mecanismos de segurança e autenticação, comunicação, máquinas virtuais, armazenamento, entre outros), o que dificulta o desenvolvimento de sistemas que forneçam soluções para federação de nuvens. De acordo com [75], surge o problema chamado de *vendor lock-in*, no qual as empresas ou os desenvolvedores ficam presos à determinadas tecnologias, impossibilitando a migração de negócios para outras tecnologias por haver um preço incompatível com a realidade do cliente que faz uso de determinada estrutura de nuvem computacional [72].

3.6 Considerações Finais

O modelo de computação em nuvem se encontra em constante evolução, no que tange a oferta de seus serviços, com novos modelos de serviços surgindo para suprir as diversas demandas dos usuários. Todavia, ainda que as nuvens já estejam sendo bastante utilizadas por diferentes organizações e o público em geral, elas ainda carecem de padrões globais de comunicação e arquiteturas para que as federações sejam disponibilizadas mais facilmente. Para cada federação a ser realizada, são necessárias implementações de *plugins*, que são adaptações para os diversos cenários das diferentes tecnologias encontradas nos provedores de nuvem [75]. Por outro lado, com o crescimento de tecnologias como a Internet-das-Coisas (IoT), na qual todas as "coisas" necessitam estar conectadas na rede, acarretará em uma grande demanda por padronização nas comunicações de recursos computacionais disponíveis em nuvem.

Neste sentido, ainda que não existam padrões para implementação de comunicação com os ambientes de nuvens presentes em uma federação, os orquestradores de nuvens têm como objetivo tratar as questões levantadas pelo modelo no que tange essa interoperabilidade. Com isso, a utilização de microsserviços pelos provedores de nuvem em vista dos benefícios que podem ser alcançados e, aliados a definições de padrões de comunicação para acesso de seus recursos, pode ser uma resposta às dificuldades apontadas para a criação de plataformas de federação de nuvem [75]. Neste sentido, a plataforma de orquestração de nuvens BioNimbuZ 2 é detalhada no Capítulo 4.

Capítulo 4

BioNimbuZ 2: Federação de Nuvens Sob Microsserviços

Este capítulo apresenta a plataforma de federação BioNimbuZ, em sua segunda versão. Para isso, a Seção 4.1 introduz os conceitos relacionados à plataforma. A Seção 4.2 apresenta a plataforma BioNimbuz 2, descreve a arquitetura da plataforma desenvolvida sob microsserviços, as Camadas de Aplicação, de Federação, de Coordenação, e de Execução. Por fim, a Seção 4.3 apresenta as considerações finais deste capítulo.

4.1 Considerações Iniciais

Orquestradores de nuvens podem ser definidos como plataformas nas quais são criados descritores de infraestrutura que o orquestrador segue para implementar, de forma autônoma, sem qualquer interação com o usuário, os serviços disponíveis em determinado ambiente de nuvem. Esse conceito garante que um descritor de infraestrutura, atualizado e testado, esteja disponível para os desenvolvedores iniciarem o processo de implantação, fornecendo ao orquestrador simplesmente o código descritivo da infraestrutura [56].

Existem várias implementações para o conceito de "orquestração multi-cloud", tais como: Roboconf [88], Trans-Cloud [17], Live Cloud [118], SALSA [62], IM do GRyCAP [16], BioNimbuZ 1 [103], BioNimbuZ 2 [75], CloudFormation [5], Terraform [13], e Cloud Assembly [116].

Neste capítulo é descrito o orquestrador BioNimbuZ 2 [75], que apresenta uma arquitetura sob microsserviços para orquestração de nuvens, através de *plugins* específico para cada nuvem presente na federação, para execução de *workflows* científicos de Bioinformática. Sendo assim, este trabalho propõe um serviço de predição de tempo e dimensionamento de recursos integrado ao BioNimbuZ 2.

4.2 Plataforma BioNimbuZ

O BioNimbuZ é uma plataforma para federação de nuvens híbridas, que foi proposta inicialmente por [103] a partir de uma arquitetura monolítica. O objetivo é atender a demanda por plataformas de nuvens federadas na execução de *workflows* científicos de Bioinformática, nas necessidades de processamento, de armazenamento, entre outras [100]. Neste sentido, com o passar do tempo, tornou-se necessária a implementação da plataforma sobre a arquitetura de microsserviços, proposta por [75].

Neste sentido, várias implementações com o intuito de melhorar a plataforma foram concebidas em trabalhos que proporcionaram mais eficiência e novas funcionalidades à solução [6, 22, 48, 75, 81, 92, 98, 100, 115]. Cada trabalho colaborou com a implementação da solução proposta, adicionando ou melhorando as diversas partes que compõem a versão atual da plataforma BioNimbuZ.

A arquitetura do BioNimbuZ se iniciou por meio da comunicação *Peer-to-peer* (P2P) para que fosse possível uma comunicação distribuída com a camada de núcleo do sistema. Posteriormente, mecanismos de coordenação de tarefas, manutenção e sincronização das informações geradas em um ambiente distribuído foram inseridos à plataforma, com ferramentas como o ZooKeeper[1] para serializar os dados, e a ferramenta Avro[2] para as Chamadas de Procedimento Remoto (RPC).

Todavia, alguns dos serviços mantidos na versão inicial da plataforma ainda não foram implementados na segunda versão. Assim, este trabalho tem como um de seus objetivos a implementação de um serviço de predição de tempo e dimensionamento de recursos para a segunda versão da plataforma BioNimbuZ, utilizando métodos de aprendizado de máquina. O serviço de predição proposto por [100], auxilia o usuário na escolha dos recursos necessários para uma eficiente execução de seu *workflow*, com estimativas de tempo, de custo e de recursos computacionais necessários para sua execução.

Para isso, na versão inicial da plataforma é necessário que algumas informações sobre o *workflow* sejam fornecidas para a Camada de Aplicação do sistema para que o serviço efetue as estimativas supracitadas. De acordo com [100], este serviço constitui-se em quatro fases: a coleta de informações, na qual o usuário informa os parâmetros desejados para a execução do *workflow*; avaliação da base histórica de execução de *workflows*, com o intuito de melhorar a acurácia do modelo preditivo a partir, também, de algumas transformações em suas entradas; execução da predição a partir de um modelo estatístico de regressão multilinear; e por fim, armazenar e informar ao usuário os dados das predições obtidas com a aplicação das três fases anteriores.

[1] https://zookeeper.apache.org/
[2] https://avro.apache.org/

Neste sentido, o BioNimbuZ 2 foi proposto como uma plataforma de orquestração para federação de nuvens, utilizando conceitos da arquitetura de microsserviços, arquitetura paralela e distribuída, para a execução de diferentes tarefas de maneira simplificada, eficiente e escalável a partir da inserção de novos provedores de nuvens.

Com o intuito de transparecer independência de provedores, o trabalho de [75], concebeu uma melhor utilização dos recursos computacionais e redução de custos para os usuários, fornecendo suas credenciais para que sejam administradas pela plataforma de maneira mais eficiente. Com a utilização direta das credenciais dos usuários, a solução proposta não necessita lidar com questões de tarifação indireta, e nem com mecanismos de repasse de custos, o que garante que cada usuário pague unicamente pelos recursos consumidos, independente dos provedores usados na execução da tarefa submetida.

A arquitetura propicia uma melhor distribuição de seus serviços, contribuindo na agilidade de comunicação e redução de latência de rede, além de um monitoramento mais eficiente dos *workflows* executados. Outro benefício da plataforma BioNimbuZ 2 é o de evitar o *vendor lock-in*, como dito na Seção 3.5.2 é um aprisionamento tecnológico, no qual as empresas ou os desenvolvedores ficam presos à determinadas tecnologias, o que impede a migração de negócios para outras tecnologias por conta do preço proibitivo.

4.2.1 Arquitetura

A arquitetura da plataforma BioNimbuZ 2 foi projetada de maneira hierárquica e distribuída, com o intuito de facilitar a distribuição dos componentes em diferentes provedores de nuvem por meio de seus *plugins* previamente desenvolvidos a partir de padrão de comunicação proposto.

Os componentes presentes na arquitetura podem ser visualizados na Figura 4.1. Assim, os *plugins* contribuem na estruturação sob microsserviços, construídos de forma desacoplada e com execução independente da plataforma, o que constitui uma independência entre as pequenas partes do processo, diminuindo o acoplamento e isolando os diferentes serviços na arquitetura proposta.

A comunicação entre os componentes da arquitetura é feita em dois níveis, uma em alto nível e a outra em baixo nível. A primeira, em alto nível, permite a troca de informações de forma objetiva e direta entre as nuvens, enquanto a segunda, em nível mais baixo, é feita entre os componentes da federação, e serve para trafegar informações das tarefas, como mensagens de coordenação e/ou execução. A arquitetura proposta possibilita uma atualização de informações em tempo real aos seus múltiplos usuários de maneira descentralizada.

A utilização de credenciais dos usuários nas diferentes nuvens da federação é uma característica importante no que tange a descentralização de recursos de nuvem em uni-

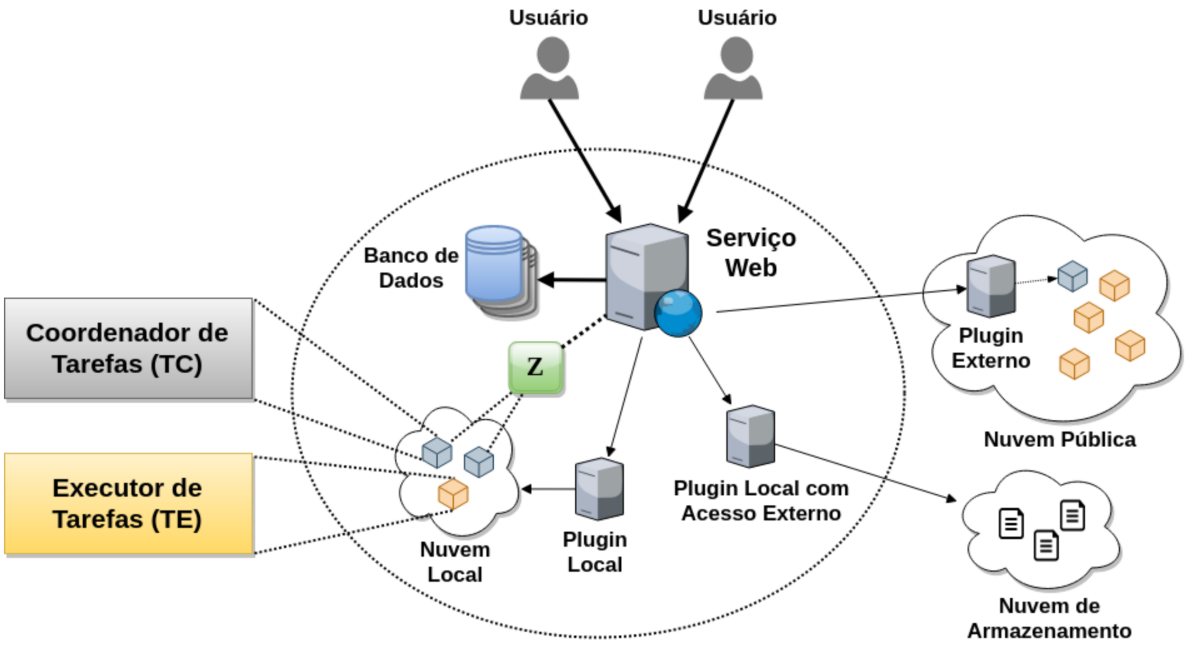

Figura 4.1: Distribuição dos Componentes da Arquitetura do BioNimbuZ 2 [75].

versidades ou instituições que possuem uma conta específica para processamento de tarefas em ambiente de nuvem. Isso é importante porque a estruturação de grupos, fornecida pela plataforma através de um Serviço web, é possível repassar tais credenciais entre diferentes usuários presentes na plataforma. Assim, a tarifação é feita diretamente pelos provedores de nuvem, sem que a plataforma de federação tenha a responsabilidade sobre o controle e o processo de cobrança de cada grupo de usuários.

Neste sentido, a arquitetura foi divida em quatro camadas, as quais podem ser vistas na Figura 4.2, detalhadas nas próximas seções e descritas resumidamente a seguir:

- **Camada de Aplicação**: fornece mecanismos para que os usuários possam interagir com a plataforma por meio do Serviço web;
- **Camada de Federação**: possui mecanismos facilmente acopláveis à plataforma e que permitem a criação da federação de nuvens;
- **Camada de Coordenação**: implementada pelo Coordenador de Tarefas (TC), e é responsável por coordenar o fluxo e o monitoramento de execução das tarefas em execução;
- **Camada de Execução**: implementada pelo Executor de Tarefas (TE), e é responsável por efetivamente processar as tarefas e monitorar seus estados e o consumo de recursos, como CPU, memória RAM, disco, entre outros.

Por se tratar de uma divisão de responsabilidades bem definidas de cada camada presente na arquitetura, é importante o detalhamento de cada uma de maneira a simplificar

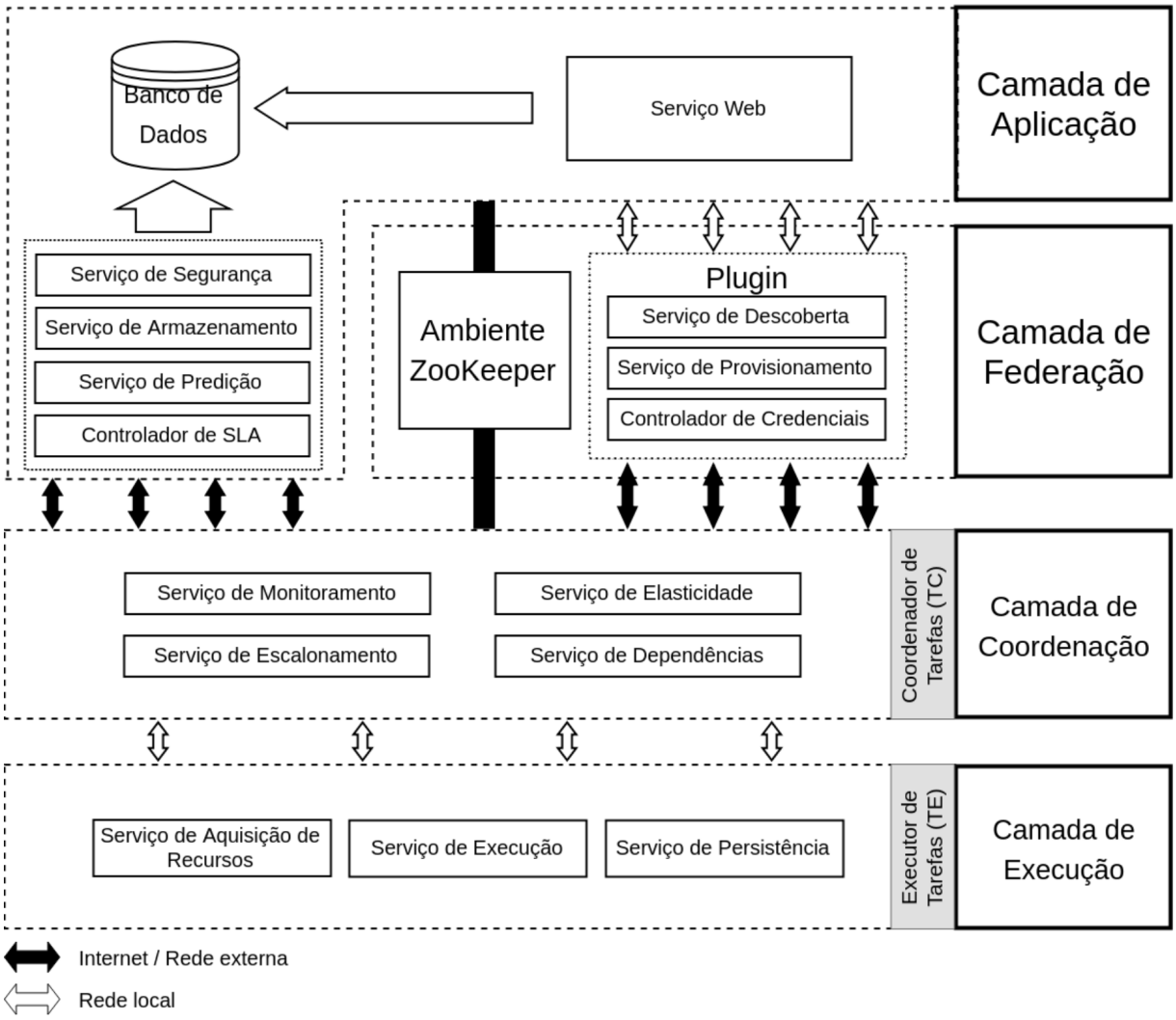

Figura 4.2: Arquitetura Orientada a Microsserviços do BioNimbuZ 2 [75].

os conceitos relacionados, suas funcionalidades e seu papel na arquitetura. As próximas seções descrevem cada uma dessas camadas em detalhes.

4.2.2 Camada de Aplicação

O objetivo desta camada é fornecer uma interface gráfica de usuário responsiva, além de serviços relativos ao armazenamento, segurança, comunicação em banco de dados, entre outros. Esta camada foi desenvolvida utilizando o *framework* Playframework em sua versão para Java [63], de código aberto e com os padrões de interoperabilidade para o contexto web. Além disso, tratando-se de arquitetura *stateless*, que não armazena estado de seus usuários, esta camada também utiliza o padrão de *tokens* JWT [50] para autorização do acesso aos usuários à plataforma.

Com o Serviço web os usuários fazem o acesso à plataforma e podem gerenciar seus *workflows*, *upload/download* de arquivos, e monitorar seus estados de execução. Nesta linha, o Serviço de Segurança é responsável por gerenciar a autenticação dos usuários na plataforma, e as credenciais dos serviços de nuvem disponíveis. Assim, o Serviço de Segurança viabiliza o acesso de múltiplos usuários à plataforma, garantindo o isolamento entre os dados, execuções, e etc. dos diversos usuários e seus grupos.

O Serviço web faz distinção entre papéis desempenhados pelo perfil Administrador e Cliente. O usuário Administrador possui tanto as funcionalidades de usuário Cliente, quanto funcionalidades responsáveis pela configuração dos *plugins*, credenciais, grupos, e tabelas de preços.

Para o Serviço de Armazenamento é delegada a responsabilidade de tomar decisões de armazenamento dos arquivos de entrada e de saída das tarefas executadas, em execução ou em preparação. As decisões de armazenamento consideram questões como o custo e a capacidade, e são tomadas com base nos dados levantados pelos *plugins* e o Serviço de Descoberta, que são vistos na Seção 4.2.3.

Nesta linha, o Serviço de SLA mantém o controle dos SLAs criados pelos usuários para a execução das tarefas. Além disso, ele analisa constantemente as informações de monitoramento para garantir os Acordos de Nível de Serviços estabelecidos. Este controlador está sempre atualizado em relação às informações de monitoramento coletadas pelos Coordenadores, presentes na Camada de Coordenação.

Diante do exposto, este trabalho propõe a implementação do Serviço de Predição, previsto nesta Camada de Aplicação, a partir de métodos de aprendizado de máquina. Assim, o seu principal objetivo é fornecer ao usuário uma forma simplificada de predizer o tempo de execução, os recursos necessários, além do custo predito dos *workflows* criados e gerenciados a partir do Serviço web supracitado.

4.2.3 Camada de Federação

A Camada de Federação é responsável por garantir a integração dos diferentes componentes do sistema situados em diferentes nuvens da federação. Assim, por se tratar de um ambiente distribuído, a plataforma BioNimbuZ 2 é dependente dos *plugins* que implementam as funcionalidades de determinado provedor de nuvem, e do Ambiente ZooKeeper [49].

Esta camada, em conjunto com os *plugins*, foi desenvolvida utilizando o *framework Spring Boot* [89], que permite criar serviços na linguagem Java auto-contidos e com um Servidor web embutido, responsável por atender as requisições HTTP ou HTTPS solicitadas por seus usuários, sem a necessidade de ser integrado em outros ambientes para sua completa execução.

O ZooKeeper [49], criado pela Fundação Apache, é um serviço de coordenação de sistemas distribuídos, que tem o objetivo de simplificar e democratizar esse tipo de estrutura computacional. Ele utiliza um modelo de dados que simula uma estrutura de diretórios, e tem como finalidade facilitar a criação e a gestão dessa estrutura, que pode ser de alta complexidade e de difícil coordenação e manutenção [100].

O ZooKeeper possui estruturação a partir de espaços de nomes chamados de *znodes*, que são organizados de forma hierárquica similar aos sistemas de arquivos. Neles podem ser armazenadas informações que facilitem o controle do sistema distribuído, tais como metadados, caminhos, dados de configuração e endereços [49, 100]. Cada *znode* tem a capacidade de armazenar até 1 Megabyte (MB) de informação, e são identificados pelo seu caminho na estrutura.

Na estrutura do Zookeeper existem dois tipos de *znodes*, os persistentes e os efêmeros. Os persistentes, como o nome descreve, são aqueles que continuam persistentes após um provedor sair do ar, sendo útil para armazenar informações sensíveis ao processo distribuído as quais se deseja manter a posse. Por outro lado, os efêmeros são criados de maneira específica para cada novo participante da federação, pois assim que este novo participante ficar indisponível, o *znode* efêmero referente a ele é eliminado. Portanto, a falta do *znode* de determinado participante sinaliza para todos os outros participantes da federação que este participante está indisponível.

Em conjunto com os *znodes* efêmeros são definidos os *watchers*, que funcionam como observadores de mudanças, e enviam alertas sobre as alterações ocorridas em algum dos *znodes* efêmeros, ativando este *watcher* para que os serviços recuperem as tarefas e os arquivos que estavam na máquina que ficou indisponível. Este conceito é útil para sistemas distribuídos, pois permitem o monitoramento constante do sistema, pois os *watchers* sinalizam quando um provedor está fora do ar, ou quando um novo recurso estiver disponível [49, 100].

No contexto da arquitetura proposta por [75], a implementação do ZooKeeper fornece um ambiente no qual são mantidos os dados, o estado de execução das tarefas, os endereços das máquinas monitoradas pelos Coordenadores, o consumo de recursos e os metadados dos arquivos de entrada/saída das tarefas. Esse armazenamento é feito de forma distribuída entre os Coordenadores e a Camada de Aplicação, para serem consultados e acompanhados em tempo real.

Os *plugins* são estruturados a partir da arquitetura de microsserviços e implementam funcionalidades com comunicações padronizadas e de maneira simples por meio de Transferência de Estado Representacional, no inglês, REST e Interface de Programação de Aplicativos, no inglês, API [71]. Desta maneira, cada *plugin* é responsável por gerenciar e repassar informações da nuvem na qual o mesmo foi implementado e acoplado

ao ambiente da plataforma BioNimbuZ 2. Assim sendo, é indicado que os *plugins* não sejam executados dentro da nuvem para a qual são especializados, para que não haja consumo de recursos das credenciais cadastradas. Desta maneira, os *plugins*, na Camada de Federação, são estruturados da seguinte maneira:

- **Serviço de Descoberta**: gerencia informações de tabelas de preços, e tipos de recursos de computação e de armazenamento de determinada plataforma de nuvem;
- **Serviço de Provisionamento**: fornece à Camada de Aplicação a possibilidade de alocar e desalocar máquinas de determinada nuvem. Também é responsável por provisionar espaços para *upload/download* de arquivos de entrada/saída das tarefas definidas na Camada de Aplicação;
- **Controlador de Credenciais**: interpreta as informações de credenciais enviadas pela Camada de Aplicação por meio dos arquivos na forma de chave-valor, e traduz para o mecanismo de segurança utilizado pela nuvem sob seu controle.

A Figura 4.3 apresenta a estrutura de comunicação entre os componentes da arquitetura para estabelecimento da federação com a Camada de Aplicação, e comunicando-se com os *plugins* por meio de pacotes JSON [47]. Os *plugins* são responsáveis por provisionar recursos dos diferentes provedores de nuvem.

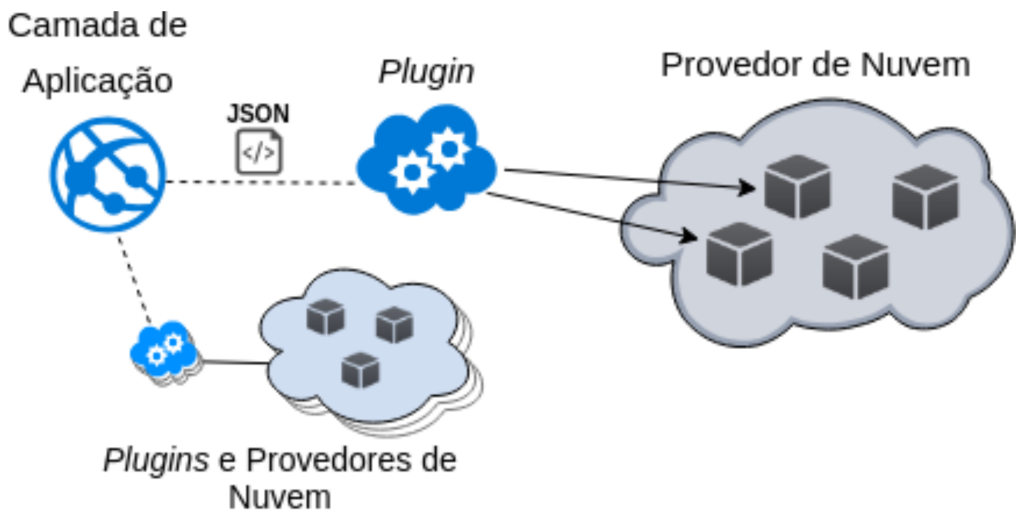

Figura 4.3: Estrutura de Comunicação com os Microsserviços [75].

Assim, a Camada de Federação é responsável por acoplar os *plugins* das nuvens participantes da federação, além de intermediar dados para a Camada de Aplicação (Seção 4.2.2), Camada de Coordenação (Seção 4.2.4), e Camada de Execução (Seção 4.2.5).

4.2.4 Camada de Coordenação

A Camada de Coordenação é implementada pelos Coordenadores de Tarefas (TC), como pode ser observado na Figura 4.4, e leva em consideração o contexto de utilização eficiente. Os TCs são responsáveis por monitorar e coordenar o fluxo das tarefas criadas pelos usuários, e também gerenciar as dependências existentes entre as tarefas relativas a nuvem em que estão executando na federação.

Um TC é criado no momento em que as tarefas são criadas e suas execuções solicitadas na nuvem de destino. O objetivo é que o respectivo TC fique mais próximo das tarefas que este deve monitorar, ocasionando em menor latência de comunicação. Neste sentido, é necessário que as credenciais dos usuários ou do grupo de usuários estejam disponíveis para a devida criação do TC que cria a tarefa. Assim, ao utilizar a plataforma, o custo total para a execução de tarefas também inclui o custo para a execução de seus Coordenadores.

Neste sentido, trata-se de um custo além das tarefas já definidas pelo usuário, e que objetiva fornecer benefícios como a execução mais eficiente de suas tarefas com serviços de provisionamento e de tolerância a falhas, providos pelo uso da plataforma de federação.

Assim, um TC solicita a execução de uma determinada tarefa a um Executor de Tarefa específico, ou TE, vistos na Figura 4.4 e detalhados na Seção 4.2.5. Desta forma, os TCs analisam constantemente as tarefas que devem ser executadas e repassadas aos TEs, para que, caso já devam ser encerrados, executem seu encerramento de forma autônoma, evitando o consumo de recursos. O TE procede com a execução da tarefa em três passos, fazendo o *download* de arquivos, executando a aplicação solicitada e, por último, efetuando o *upload* de arquivos gerados.

Dessa maneira, pode ser observado à esquerda da Figura 4.4, a execução percebida pelo usuário, e à direita o comportamento real de execução no contexto da plataforma BioNimbuZ 2. O exemplo de execução de comportamento de tarefas no contexto de nuvem presente na Figura 4.4 demonstra a visão do usuário ao criar as tarefas e interligá-las de maneira a formar um *pipeline*, e como realmente será distribuída esta execução no contexto da federação com o suporte do ambiente ZooKeeper, citado na Seção 4.2.3.

Desta forma, alguns serviços são disponibilizados nesta camada, e servem para dar suporte na coordenação das tarefas, assim como, no controle das atividades destinadas ao gerenciamento das tarefas da Camada de Execução em ambiente de nuvem, sendo eles:

- **Serviço de Monitoramento**: está em constante comunicação com TEs, a partir da coleta dos dados de execução levantados por estes. Além disso, ele é responsável por monitorar o consumo de recursos próprios e os utilizados pelo seu respectivo Coordenador. Esse monitoramento não é utilizado para a tarifação dos usuários, mas para o monitoramento e garantia dos SLAs, pois a tarifação dos usuários é dada pelas nuvens usadas na execução;

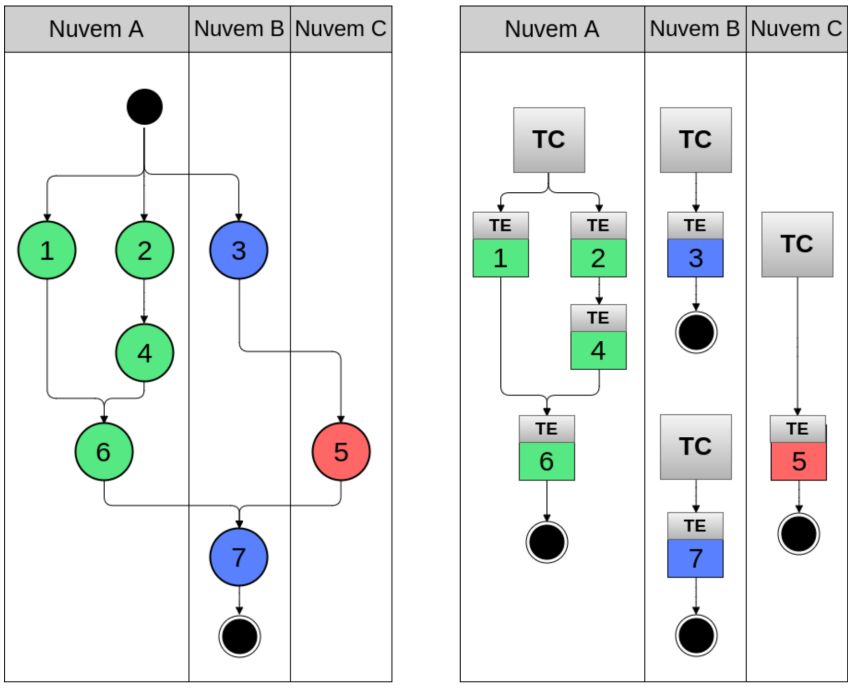

Figura 4.4: Componentes de Coordenação e Execução de Tarefas [75].

- **Serviço de Dependências**: além de verificar as demandas das tarefas, ele monitora suas dependências, ou seja, quais serão as próximas tarefas a serem executadas, e se determinado lote de tarefas foi concluído ou não. Ele também verifica se o TC necessita ser encerrado;
- **Serviço de Elasticidade**: com o auxílio do Serviço de Monitoramento, analisa as informações coletadas pelos TEs. Desta forma, é capaz de decidir quando aumentar ou reduzir recursos de nuvem como quantidade de máquinas virtuais, tamanho de memória, número de CPUs, etc.;
- **Serviço de Escalonamento**: responsável por distribuir as tarefas que devem ser executadas na nuvem em que o TC está executando, e solicitar o início da execução da tarefa ou o lote delas.

Assim, esta camada fornece comunicação para as outras camadas da arquitetura com o objetivo de gerenciar o andamento das tarefas e coordenar seu fluxo. Esta camada é dependente da Camada de Execução, pois toda tarefa é gerida por um TC, e um TC pode coordenar um ou mais TEs.

4.2.5 Camada de Execução

Esta camada implementa os serviços de menor nível de abstração dentro da arquitetura. Desta forma, são responsáveis por executar as tarefas definidas pelo usuário na Camada

de Aplicação, e possui serviços que contribuem para o alcance de seus objetivos, sendo eles [75]:

- **Serviço de Aquisição de Recursos**: é responsável por obter os arquivos de entrada a serem utilizados durante o processo de execução, dando início à execução da tarefa;
- **Serviço de Execução**: responsável por manter o ciclo de vida das tarefas, assim como seu constante monitoramento repassando informações para a Camada de Coordenação;
- **Serviço de Persistência**: após finalizada com sucesso a execução da tarefa, este serviço é iniciado para efetuar o *upload* dos arquivos gerados para a nuvem especificada na Camada de Aplicação.

Dessa maneira, esta camada é marcada por três estágios, sendo eles: o *download* dos arquivos necessários para execução a partir do Serviço de Obtenção de Recursos; a execução, efetuada pelo Serviço de Execução, que prepara a linha de comando para execução; e *upload* dos arquivos gerados a partir da execução com auxílio do Serviço de Persistência.

Após cada mudança de estágio é gerada uma notificação para o TC, o qual filtra e detecta informações redundantes com o objetivo de evitar o seu repasse. O TE mantém constante comunicação com o seu respectivo coordenador, enviando informações de monitoramento da máquina virtual, assim como a fase de execução da tarefa.

4.3 Considerações Finais

A plataforma de orquestração para federação de nuvens BioNimbuZ 2 objetiva resolver o *vendor lock-in*, em que os usuários se tornam dependentes de uma determinada plataforma de nuvem, em vista da carência de utilização de padrões abertos pelos provedores. Além de propor um novo modelo arquitetural de orquestração de nuvens.

Desta forma, com a necessidade de computação de alto desempenho para os projetos de Bioinformática, a estrutura de federação de nuvens ofertada por esta arquitetura proporciona maiores ganhos no que tange a economia de escala, e diferentes possibilidades de execução de projetos de Bioinformática nas diversas máquinas a partir dos provedores de nuvens.

Diante do exposto, a arquitetura do BioNimbuZ não possui um serviço de predição de recursos em sua estruturação inicial. Desta forma, este trabalho propõe um serviço integrado à plataforma BioNimbuZ para que seja possível disponibilizar aos seus usuários

uma forma de conhecer melhor as características de tempo e desempenho de seus projetos de Bioinformática, utilizando modelos de aprendizado de máquina supervisionado para problemas de regressão que serão melhor detalhados no Capítulo 5.

Capítulo 5

Aprendizado de Máquina

Neste capítulo são tratados os conceitos de aprendizado de máquina, suas categorias, além de alguns algoritmos que são direcionados para problemas de regressão. Para isso, a Seção 5.1 apresenta os aspectos introdutórios deste capítulo. A Seção 5.2 apresenta o conceito de IA e os primeiros estudos no campo do aprendizado de máquina. Em seguida, a Seção 5.3 descreve as suas categorias. Adiante, a Seção 5.5 apresenta o desafio na fase de treinamento e a forma de resolver tais empecilhos. Na Seção 5.4 são apresentados os modelos de regressão que servirão de base para o entendimento do modelo de conselho de preditores proposto neste trabalho. São apresentados os trabalhos relacionados na Seção 5.6. Por fim, na Seção 5.7 estão as considerações finais deste capítulo.

5.1 Considerações Iniciais

Definido como um subcampo da Inteligência Artificial (IA), o Aprendizado de Máquina (AM), do inglês, *Machine Learning* (ML), no qual um programa aprende uma tarefa se seu desempenho avaliado por uma dada métrica aumenta com a experiência, ou se toma decisões melhores baseada na solução de problemas anteriores [78].

As categorias de AM possuem denominações a partir de suas entradas, formulação do problema, método de resolução e resultado esperado. Desta forma, as categorias de aprendizado são definidas pela literatura em supervisionado, não supervisionado e por reforço. Além disso, elas apresentam diferentes classificações quanto ao problema, sendo divididos em problemas linearmente separáveis, nos quais é possível separar os dados de entrada a partir de uma linha reta em um hiperplano; e os não-linearmente separáveis, os quais não podem ser definidos somente a partir de uma reta de separação única [80].

Nesta linha, alguns autores prezam pela união conceitual entre o aprendizado supervisionado e o não supervisionado, formando assim o aprendizado semi-supervisionado. Assim sendo, a IA, por meio do AM, fornece mecanismos que são aproveitados neste

trabalho, no que tange os modelos de aprendizado supervisionado para problemas de regressão.

5.2 Inteligência Artificial

Neste sentido, a IA tornou-se nos últimos anos novamente uma pauta para pesquisadores das mais diversas áreas. O termo é originado em 1956 e é um dos campos de estudo mais recentes em ciência e engenharia [83]. Atualmente, se mostra uma área de estudo que fornece modelos de análise de dados e de aprendizado para diversas outras áreas.

A IA abrange desde o geral, como aprendizagem e percepção, até tarefas específicas, como automação de processos industriais, escrita de músicas, predição de dados, diagnóstico de doenças, carros autônomos, entre outras atividades. Desta forma, tornou-se relevante para qualquer atividade intelectual, e tem demonstrado ser um campo de estudo e de aplicação universal [83].

A grande maioria dos trabalhos relacionados à IA, atualmente, faz referência ao trabalho de Warren McCulloch e Walter Pitts [73], o qual é considerado o primeiro trabalho sobre IA publicado. Esses dois pesquisadores propuseram uma abordagem de aprendizado a partir do modelo de neurônios cerebrais, ou seja, definindo um modelo de neurônio artificial. Neste sentido, o termo "redes neurais artificiais" emerge ao arranjar vários neurônios artificiais em camadas, e cada neurônio se caracteriza por estar "ativado" ou "desativado". A ativação ocorre em resposta à estimulação por um número suficiente de neurônios vizinhos.

Assim, outros trabalhos surgiram demonstrando a eficácia do modelo proposto, no entanto, o modelo se mostrou custoso computacionalmente para os computadores da época. Neste sentido, com o advento dos computadores de alto desempenho, e até mesmo os convencionais, tornou-se possível a democratização deste modelo, e novos trabalhos estão sendo feitos na atualidade utilizando este modelo proposto no fim da década de 50.

Assim sendo, com o passar do tempo, modelos de AM tiveram uma evolução e, por meio de suas categorias, os algoritmos foram aprimorados, o que aumentou a capacidade de análise de dados e resolução de problemas para as mais diversas áreas. Nesse contexto, nas próximas seções essas categorias serão descritas em detalhes.

5.3 Categorias do Aprendizado de Máquina

O AM é definido como a capacidade de agentes adquirirem conhecimento por meio da observação e/ou de sua interação com o ambiente. Neste sentido, o aprendizado acontece com a experiência deste agente ou a partir de um conjunto de dados generalizados conhe-

cidos, e para isso é empregada a aprendizagem indutiva, na qual a partir de um conjunto de exemplos é possível gerar conclusões genéricas [83].

O processo de inferência a partir de observações de um conjunto particular de eventos é definido como indução. Ela é caracterizada como o raciocínio que se origina em um conceito específico e o generaliza, ou seja, do particular para o geral. Na indução, algo é aprendido efetuando-se inferência indutiva sobre os exemplos apresentados a partir de um conjunto de observações. Desta forma, as hipóteses geradas pela inferência indutiva podem ser assertivas ou não [79].

A inferência indutiva é um dos principais métodos utilizados para derivar conhecimento novo e predizer eventos futuros. Apesar disso, este modelo de inferência deve ser utilizado partindo-se do pressuposto de que os dados, observados previamente, representam e conseguem generalizar os resultados esperados, pois, de acordo com [79], se o número de exemplos for insuficiente, os resultados obtidos podem não ser adequados, porque a partir dos exemplos o modelo não possuirá uma capacidade de generalização.

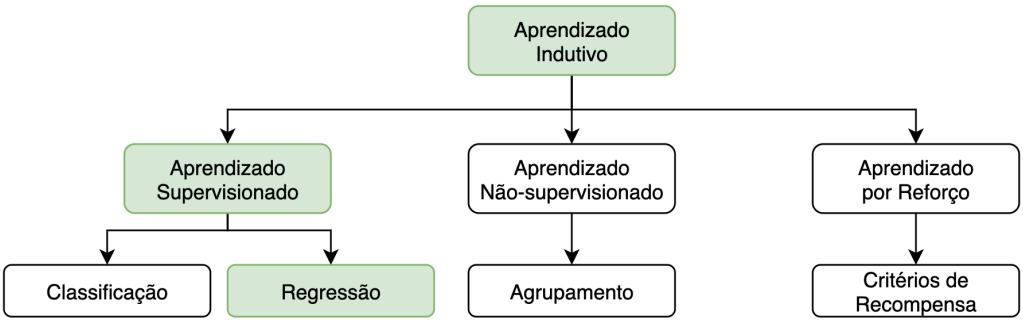

Figura 5.1: Hierarquia do Aprendizado Indutivo [79].

Na Figura 5.1 é descrita uma hierarquia de aprendizado, na qual é possível perceber o percurso que leva ao modelo de aprendizado supervisionado do tipo regressão. O tipo regressão é um dos objetos de estudo deste trabalho, no que tange a predição de tempo de execução de programas utilizados em *workflows* de Bioinformática.

Neste sentido, o aprendizado indutivo pode ser dividido em supervisionado, não supervisionado e por reforço. Resumidamente, o aprendizado supervisionado é efetuado quando um conjunto de exemplos de treinamento é fornecido ao algoritmo, ou indutor, para os quais o rótulo da classe associada é conhecido [83].

No aprendizado não-supervisionado, o indutor analisa os exemplos fornecidos e tenta determinar a forma como podem ser agrupados, formando *clusters* dos dados de entrada. E por fim, o aprendizado por reforço é caracterizado como um método de aprendizado de agentes através do oferecimento de recompensas ou punições, sem necessidade de especificar previamente qual tarefa deve ser executada [79].

Portanto, o AM trata-se de um processo computacional iterativo em que os parâmetros de um modelo matemático são ajustados a fim de minimizar uma medida de erro relacionado às previsões por ele feitas [108].

5.3.1 Supervisionado

O processo de AM supervisionado é feito a partir de entradas, as quais se deseja generalizar para predizer observações futuras; um processamento a partir de alguma técnica de generalização dessas entradas; e uma conclusão, a qual pode ser avaliada a partir de saídas rotuladas. Assim, trata-se de um processo de extração de conhecimento por meio de dados previamente rotulados com o objetivo de predizer classes ou valores contínuos [83].

Assim, a classe apresenta valores discretos e o conceito induzido pelos algoritmos é conhecido como classificador, embora também existam algoritmos que definem modelos de regressão, devido a classe apresentar valores contínuos [41]. O objetivo de um algoritmo de aprendizado supervisionado é predizer qual é o rótulo que mais se aproxima a um conjunto de dados de entrada a ele repassados.

Este trabalho possui foco nos modelos de aprendizado supervisionado para problemas de regressão. Desta forma, na Seção 5.4 serão detalhados os algoritmos mais utilizados no contexto desse tipo de aprendizado, com maior foco nos que resolvem problemas de regressão.

5.3.2 Não-supervisionado

Na aprendizagem não-supervisionada o agente aprende padrões na entrada, embora não seja fornecido nenhum *feedback* ou rótulo explícito. Dessa maneira, não é o objetivo da solução explicitar quais são os caminhos que o agente segue e, em muitos casos, descobertas de padrões são identificados sem que o projetista saiba da relação. Este tipo de solução é empregada para agrupar elementos com características semelhantes [83].

Neste sentido, a tarefa mais comum de aprendizagem não-supervisionada é o agrupamento. O agrupamento é a detecção de grupos de exemplos de entrada potencialmente úteis. Por exemplo, um agente de táxi pode desenvolver gradualmente um conceito de "dia de tráfego bom" e "dia de tráfego ruim" sem nunca terem sido rotulados exemplos de cada um deles por um professor, por exemplo [83]. Portanto, após a determinação dos agrupamentos, normalmente, é necessária uma análise para determinar o que cada agrupamento representa no contexto do problema.

5.3.3 Semi-supervisionado

De acordo com [83], em casos práticos, essas distinções sobre os tipos de aprendizado nem sempre são tão nítidas. Na aprendizagem semi-supervisionada são dados alguns poucos exemplos rotulados, e deve-se extrair características a partir dos dados rotulados para que os dados não rotulados sejam de fato conhecidos.

Mesmo os rótulos em si podem não ser as verdades oraculares que é esperado. Por exemplo, ao tentar construir um sistema para predizer a idade de uma pessoa a partir de uma foto, são reunidos alguns exemplos rotulados com fotos e as idades reais de pessoas previamente conhecidas. Assim, tem-se a aprendizagem supervisionada [110]. Nesta linha, é conhecido que algumas das pessoas mentiram sua idade. Não é só que haja ruído aleatório nos dados, mas as imprecisões são sistemáticas, e descobri-las é um problema de aprendizagem não-supervisionada, envolvendo imagens, idades autorrelatadas e idades (desconhecidas) verdadeiras. Deste modo, tanto o ruído como a falta de rótulos cria-se uma relação entre aprendizagem supervisionada e não-supervisionada [83].

5.3.4 Por Reforço

Em aprendizagem por reforço, o agente aprende a partir de uma série de reforços, ou seja, recompensas ou punições. Este conceito foi observado inicialmente em estudos da teoria comportamental [84]. Essa teoria explica como ações são tomadas a partir de reforços, sendo positivos ou negativos, de inserção ou retirada, sobre o indivíduo.

Um exemplo prático, do contexto real, é a falta de gorjeta ao final de uma corrida, que dá ao agente do táxi a indicação de que algo saiu errado. Os dois pontos de vitória no final de um jogo de xadrez informam ao agente que fez a coisa certa. Cabe ao agente decidir qual das ações anteriores ao reforço foram as maiores responsáveis pelo reforço repassado [83].

Esta teoria é aplicada aos agentes, e por exemplo, rotinas de recompensas são inseridas em um processo de ranqueamento em um algoritmo. Esse ranqueamento faz com que a mudança de contexto entre supostas pontuações acumuladas por ele, a partir de ações tomadas no ambiente, sejam percebidas. Esta estrutura é bastante utilizada na construção de jogos eletrônicos competitivos [111].

5.4 Algoritmos para Modelos de Regressão

Os algoritmos de AM resolvem os mais diversos problemas, no entanto, cada algoritmo se adéqua a um contexto específico. É possível utilizar diferentes algoritmos para soluções semelhantes e vice-versa.

Os problemas de classificação estão interessados em rotular suas saídas com a finalidade de predizer a qual classe tal conjunto de entradas pertence. Por exemplo, predizer a idade de uma pessoa relacionando um conjunto de imagens, sendo os dados de entrada, a um conjunto de idades, sendo seus respectivos rótulos. Assim, será possível classificar a idade de uma nova imagem, desconhecida do conjunto de aprendizado, e predizer sua respectiva idade.

Em problemas de regressão o objetivo é encontrar um valor contínuo que possa predizer um evento ou um valor específico que possa ser generalizado a um resultado numérico. Como exemplo, um modelo de regressão pode ser utilizado para predizer a análise de crédito de um cliente baseando-se em seu padrão de compra, podendo haver uma oferta de empréstimo a juros mais baixos ou mais altos, pois houve uma previsão dos riscos a partir desse modelo de regressão.

Este trabalho tem como um de seus objetivos propor um modelo de predição de tempo de execução de programas contidos em *workflows* científicos de Bioinformática, ou seja, esta proposta encaixa-se em um problema de regressão. Neste sentido, é importante destacar os algoritmos comumente utilizados para predizer valores contínuos. Nas próximas seções são descritos alguns dos principais algoritmos de AM supervisionado que são utilizados na solução de problemas de regressão. A apresentação destes modelos auxilia no entendimento da solução para o conselho de preditores, detalhado na Seção 6.5.

É importante ressaltar que o modelo de Regressão Linear Simples e o de Regressão Linear Múltipla são, também, considerados como modelos de predição estatísticos lineares em muitos trabalhos.

5.4.1 Regressão Linear Simples

Uma função linear simples, na qual uma linha reta é formada em um hiperplano de duas dimensões, com entrada x e saída y. Neste caso, a letra w é utilizada, na equação que forma a reta, como definição para pesos, do inglês *weights*. Assim, o valor de y é definido pelo ajuste do peso relativo de um termo para outro a partir da entrada x. Normalmente, w é definido como um vetor $[w_0, w_1]$, representado na Equação 5.1 e definem os coeficientes de valores reais a serem aprendidos pelo modelo [83].

$$y = w_0 + w_1 x + \epsilon \tag{5.1}$$

O parâmetro w_0 e w_1 da Equação 5.1 representam os coeficientes de regressão, sendo o w_0 o intercepto da reta, e w_1 a sua inclinação [80]. Assim, na Equação 5.1, o termo ϵ é o erro aleatório, reconhecido como normal e independente distribuído, com variância – σ^2 constante e desconhecida, além de possuir média zero [80].

Desta forma, é um modelo estatístico comumente utilizado como modelo preditivo simples, na qual há relação forte entre a variável preditora e a predita, ou seja, uma técnica estatística para modelar e investigar a relação entre duas variáveis [80].

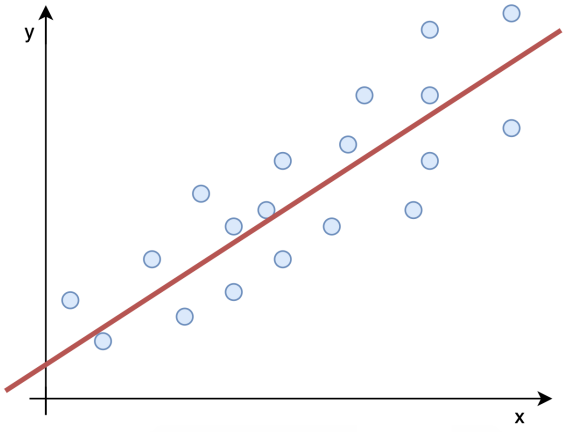

Figura 5.2: Exemplo de Regressão Linear Simples, adaptado de [69].

O exemplo mais clássico de aplicação da regressão linear simples é a predição de preços de casas baseando-se somente no valor do metro quadrado. Neste caso, é desconsiderado fatores externos, como localização, andar, estado de conservação, por exemplo. Na Figura 5.2 é possível observar a distribuição de dados de preços de casas. Trata-se de um modelo fictício, no entanto, é aplicado de maneira didática, e no eixo x estão os preços de casas, e no eixo y o valor que representa o metro quadrado, sem marcação específica na reta do eixo. Assim, a reta demarcada em vermelho, é o resultado da Equação 5.1 para este exemplo.

Neste sentido, é possível observar a relação forte e linear entre as duas variáveis. Essa observação se dá pelo aspecto linear de dispersão dos dados, ou seja, quando o valor que representa o metro quadrado aumenta, o preço da casa, linearmente, também cresce.

5.4.2 Regressão Linear Múltipla

A regressão linear múltipla é um modelo de regressão linear que possui k variáveis preditoras, ou seja, para cada variável preditora inserida na equação, um novo peso a esta é relacionado. É comum que a literatura descreva este modelo como regressão linear multivariada [83]. Assim sendo, o parâmetro w_0, w_1, ..., w_k da Equação 5.2 são coeficientes de regressão, sendo o w_0 o intercepto da reta, e w_1, ..., w_k a sua inclinação [80].

$$y = w_0 + w_1 x_1 + w_2 x_2 + w_k x_k + \epsilon \tag{5.2}$$

Um exemplo de aplicação para este modelo é a predição de tempo de dimensionamento de recursos para *workflows* científicos em nuvens federadas descrito em [100]. Nesse trabalho, o autor utiliza três variáveis de entrada para predizer o tempo de execução de programas de Bioinformática, e apresentou bons resultados na predição. No entanto, para a utilização deste modelo, faz-se necessária uma análise rigorosa de ruídos para evitar generalizações ruins do modelo.

Na Figura 5.3 é possível observar a distribuição de três variáveis em um hiperplano com três dimensões. A principal diferença entre a regressão linear simples e a regressão linear múltipla é a quantidade de variáveis preditoras, sendo que na Equação 5.2 o k é o número de variáveis preditoras, já na regressão linear simples tem-se o $k = 1$. Portanto, w é definido como um vetor $[w_0, w_1, ..., w_k]$, e define os coeficientes de valores reais a serem aprendidos por este modelo [83].

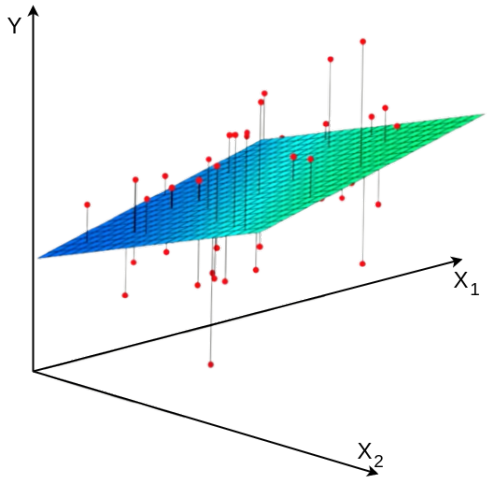

Figura 5.3: Exemplo de Regressão Linear Múltipla, adaptado de [69].

5.4.3 Máquina de Vetores de Suporte

As máquinas de vetores de suporte (no inglês, *Support Vector Machine* (SVM), basicamente, são máquinas de aprendizagem que se baseiam na Teoria da Aprendizagem Estatística, treinadas por um algoritmo supervisionado [25, 100].

Conceitualmente, vetores do espaço de entrada são mapeados não linearmente em um espaço de características de alta dimensionalidade, através de um mapeamento estabelecido a *priori*. A partir desse espaço é construída uma superfície de decisão linear, constituída de um hiperplano de separação ótima, que expressa propriedades que garantem aptidão na habilidade de generalização da máquina de aprendizagem [25].

A máquina de vetor de suporte é, atualmente, a abordagem mais popular para aprendizado supervisionado [83]. Existem algumas propriedades que tornam a SVM interessante, sendo elas [83]:

- Construção de um separador de margem máxima — um limite de decisão com a maior distância possível dos pontos de exemplo. Isso contribui para a generalização do modelo (observar os pontos em vermelho na Figura 5.4);
- Criação de uma separação linear em hiperplano tem a capacidade, porém, de incorporar os dados em um espaço de dimensão superior, usando assim o chamado truque de *kernel* [25]. Muitas vezes os dados que não são separáveis linearmente no espaço de entrada original são facilmente separáveis em um espaço de dimensão superior. Assim, o separador linear de dimensão superior é realmente não linear no espaço original. Isso significa que o espaço de hipótese é expandido em relação aos métodos que usam representações estritamente lineares;
- Método não paramétrico — requerem mais dados de treinamento para estimar a função de mapeamento e são mais lentos para treinar, pois diversas vezes precisam de muitos ajustes. Também são mais propensos a *Overfitting*.

Um esquema básico da técnica utilizada por uma SVM, que busca separar ao máximo os dados em conjuntos, através de hiperplanos, pode ser visto na Figura 5.4, ou seja, utiliza vetores de suporte para separar o máximo possível o contexto de classificação dos dados. Os vetores de suporte são demarcados pela cor vermelha na Figura 5.4.

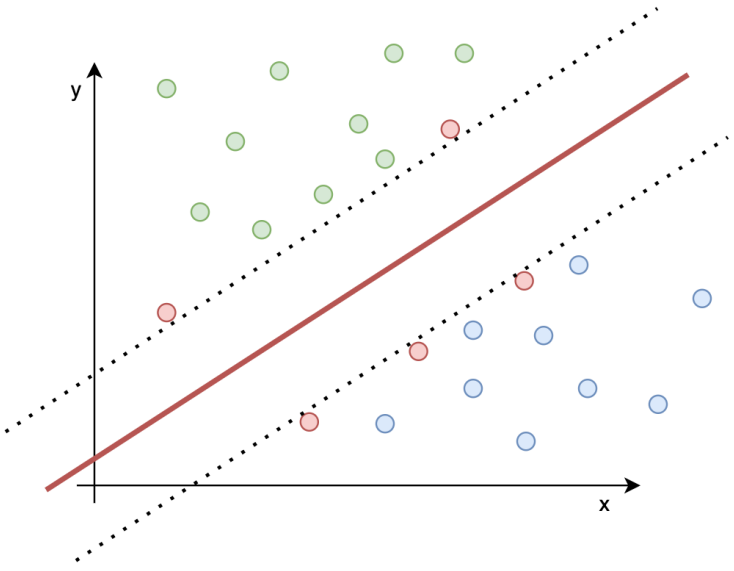

Figura 5.4: Exemplo de Classificação com uma SVM, adaptado de [83].

No contexto da predição de recursos, esta técnica pode ser utilizada para prever o tempo de execução de programas a partir de um conjunto de informações anteriores, uma vez que utiliza preditores baseados em regressões estatísticas [59, 100]. No entanto, há necessidade de um conjunto massivo de dados para que o aprendizado se mostre eficaz.

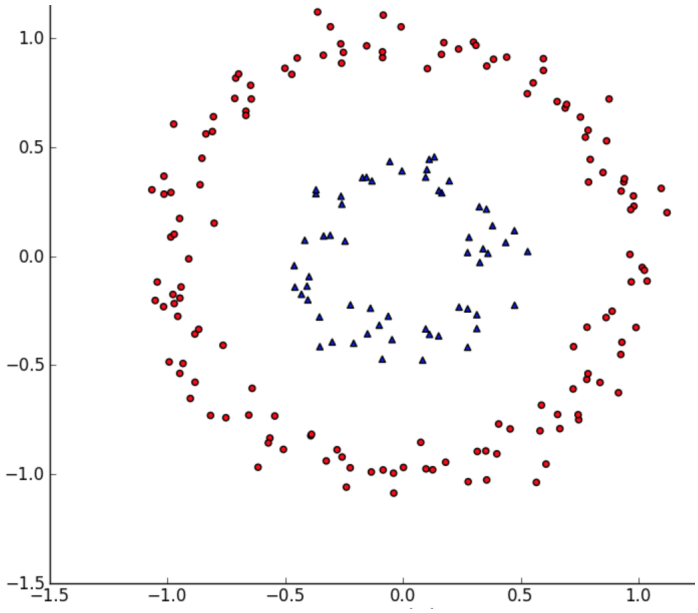

Figura 5.5: Representação de Dados Distribuídos em Duas Dimensões [55].

Para separar de maneira linear elementos não linearmente separáveis, a SVM utiliza um conjunto de funções matemáticas, conhecidas como *kernels*. Esse mapeamento é conhecido como o processo de reorganização dos objetos. Assim, é possível observar o mapeamento de elementos de duas dimensões (veja a Figura 5.5) para três dimensões (veja a Figura 5.6), a partir do truque de *kernel* aplicado [8].

Um exemplo de conjunto de dados, que não é linearmente separável, é apresentado na Figura 5.5. Todavia, esse mesmo conjunto de dados pode ser separado linearmente quando aplicada a seguinte transformação: $[x_1, x_2] = [x_1, x_2, x_1^2 + x_2^2]$ (veja a Figura 5.6). Esse mapeamento é considerado um truque de *kernel* básico e, sendo assim, outros *kernels* podem ser criados a partir dos dados de entrada. Este método de alteração das dimensões dos dados passam a ser de difícil demonstração quando o número de dimensões é maior do que três.

5.4.4 Árvore de Decisão

Uma árvore de decisão representa uma função que considera como entrada um vetor de valores de atributos e retorna uma "decisão" como resultado, ou seja, um valor de saída único dentre o espaço de possibilidades. Os valores de entrada e saída podem ser

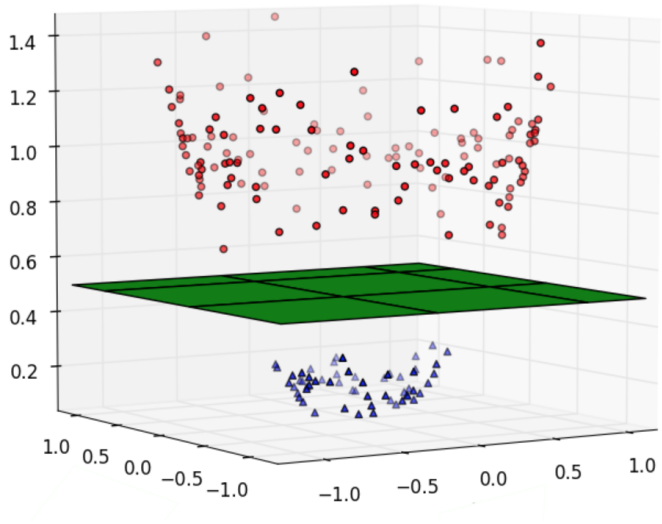

Figura 5.6: Aplicação de Truque de *Kernel* em Dados da Figura 5.5 [55].

discretos ou contínuos [83]. Desta forma, este modelo alcança sua decisão executando uma sequência de testes encadeados, em um modelo de árvore com espaço de resultados testados recursivamente para separação dos valores de saída em um hiperplano.

Cada nó interno na árvore corresponde a um teste do valor de um dos atributos de entrada, A_i, e as ramificações dos nós são classificadas com os valores possíveis do atributo, $A_i = v_{ik}$. Cada nó folha na árvore especifica o valor a ser retornado pela função. Ainda de acordo com [83], a representação de árvores de decisão é algo natural e facilmente entendido pelo seres humanos, como exemplo, os manuais são inteiramente escritos como uma única árvore de decisão que se estende por centenas de páginas.

Assim, uma árvores de decisão é montada a partir dos dados de treinamento, e probabilidades de ocorrência de cada nó é calculado baseando-se nos dados de entrada para aquele nó. Logo, a probabilidade de um evento ocorrer, somada com as probabilidades dos outros k eventos relacionados aquele nó ocorrerem, totalizam 100%. Dessa maneira, ao se percorrer uma árvore de decisão, se chega a um nó folha, então, o resultado da classificação ou regressão é encontrado.

Assim sendo, na extrema esquerda do gráfico, na linha em azul da Figura 5.7, há uma região completamente pura, e significa que qualquer corte que for feito nessa região os resultados serão considerados de região pura, desde que sejam respeitados os limites para os grupos de dados. A linha azul, traçada verticalmente na Figura 5.7, demonstra uma condição a ser tomada na estrutura da árvore, ou seja, caminhar para a direita e continuar testando, ou caminhar para a esquerda e assumir a região pura como resultado das entradas. Portanto, o particionamento recursivo para o lado esquerdo do gráfico é parado, ou seja, essa região será considerada um nó folha na árvore de decisão. Já a região da direita não é pura, assim se deve continuar o particionamento por ela até que

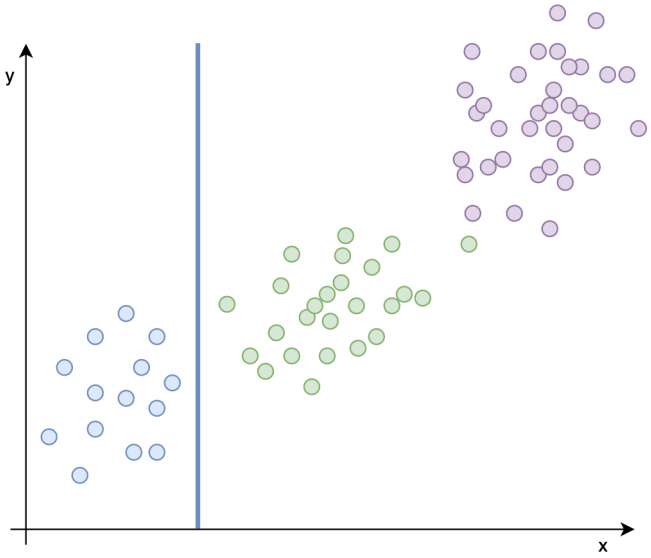

Figura 5.7: Exemplo de Particionamento Recursivo na Região Esquerda do Gráfico, adaptado de [68].

se encontre sub-regiões puras.

A Figura 5.8 apresenta a próxima partição efetuada no lado direito do gráfico recursivamente. O particionamento recursivo dos dados em retângulos do lado direito do gráfico é efetuado a partir da linha verde horizontal na Figura 5.8. Assim, na parte da direita e superior do gráfico não é considerada pura neste exemplo, sendo assim, uma chamada recursiva será feita e, portanto, uma linha vertical em coloração roxa será demarcada no gráfico e esse corte pode ser visualizado na Figura 5.9.

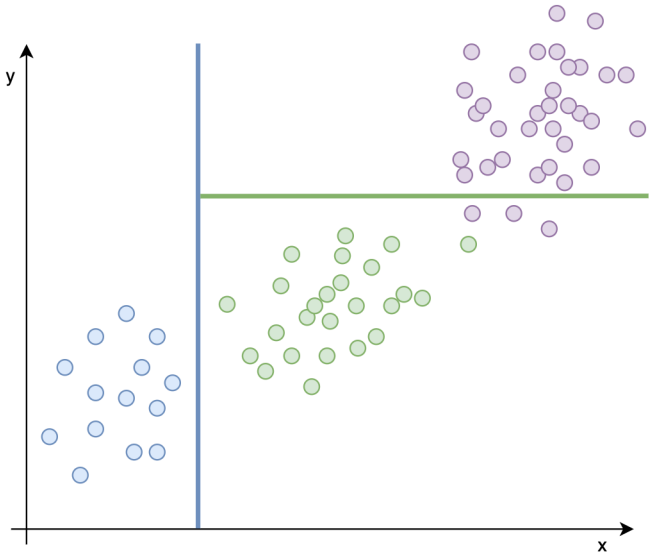

Figura 5.8: Exemplo de Particionamento Recursivo na Região da Direita do Gráfico, adaptado de [68].

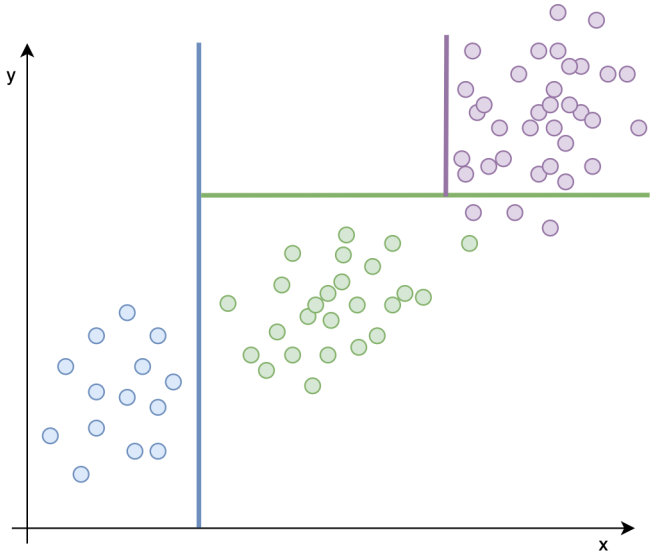

Figura 5.9: Exemplo de Particionamento Recursivo na Região da Direita e Superior do Gráfico, adaptado de [68].

Portanto, a intuição do algoritmo de construção de uma árvore de decisão é considerada bastante simples. Há vários algoritmos para aprender sua estrutura, os principais são: o CART [102], o ID3 [46] e o C4.5 [90]. Dessa maneira, todos seguem a lógica de implementação apresentado nesta seção.

5.4.5 Floresta Aleatória

A Floresta Aleatória trata-se de um modelo de aprendizado supervisionado, bastante utilizado para classificação e regressão, e, resumidamente, cria várias árvores de decisão e as combina para obter uma predição com maior acurácia e mais estável [93].

Desta forma, por se tratar de um modelo que é composto pela junção de resultados do modelo de árvore de decisão, visto na Seção 5.4.4, uma função de junção precisa ser especificada. Isso significa que os resultados agreados dos modelos, em um caso de regressão, várias funções podem ser utilizadas. Um exemplo deste modelo pode ser observado na Figura 5.10. Nesse exemplo, o resultado da floresta é definido como a média dos valores recebidos de cada árvore presente na estrutura.

Neste sentido, a função de generalização dos resultados, providos pelas diversas Árvores de Decisão, pode ser de dois tipos: média ou voto maioritário. A função de média é comumente utilizada em casos de regressão, o qual os resultados são valores contínuos e uma média aritmética é introduzida na função gerando o valor predito pelo modelo. No caso da função por voto maioritário, é considerado para problemas de classificação, ou seja, o resultado é o que obteve mais ocorrência nas saídas de cada árvore de decisão do modelo [93].

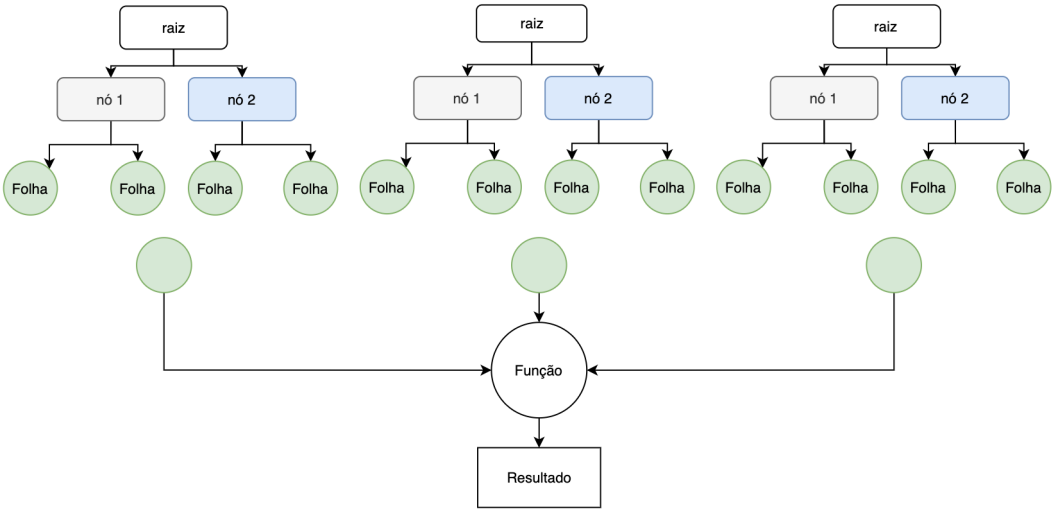

Figura 5.10: Exemplo de Floresta Aleatória e Função de Agregação de Resultados, adaptado de [44].

5.4.6 Rede Neural Artificial

Os trabalhos em Redes Neurais Artificiais (RNAs) têm sido motivados desde as primeiras pesquisas pelo reconhecimento e maneira com que o cérebro humano processa informações. Esta maneira é inteiramente diferente do computador convencional. Assim sendo, o cérebro é considerado um computador, pois é um sistema de processamento de informação, altamente complexo, não-linear e paralelo, possuindo capacidade de organizar seus constituintes estruturais, chamados neurônios, de maneira a realizar certos processamentos, como por exemplo reconhecimento de padrões, percepção, raciocínio lógico, controle motor, etc., com maior rapidez que o computador digital da década de 80 [42].

Desta maneira, o cérebro humano possui intensa capacidade de aprendizado baseado na experiência e, no momento do nascimento, o cérebro tem uma grande estrutura e a habilidade de desenvolver suas próprias regras através dessa experiência acumulada e gerada a partir do contato com o ambiente. Neste sentido, de acordo com [42], esta experiência é uma decorrência de observações do tempo, sendo que o mais dramático desenvolvimento, ou seja, por ligações estruturais e físicas, do cérebro humano acontece durante os dois primeiros anos de vida, e permanece em desenvolvimento para muito além desse estágio.

Os neurônios são as unidades elementares do sistema nervoso. Assim, cada célula neuronal é composta por um corpo celular, um axônio e dendritos (veja a Figura 5.11). Os dendritos são um conjunto de terminais de entrada, pelo corpo central. O corpo central é o local em que a informação recebida é tratada, ativada ou não, e, caso seja ativada, a mesma é trafegada pelo axônio. Assim, o axônio é um prolongamento caudal que conduz

o sinal elétrico que trafega pelo sistema nervoso, transitando a informação de um neurônio a outro como um terminal de saída.

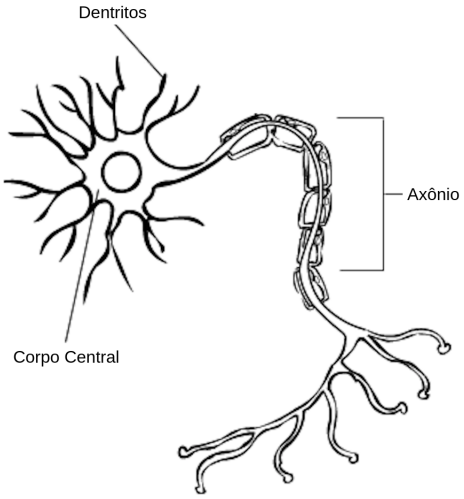

Figura 5.11: Estrutura Neuronal Humana, adaptado de [51].

Um modelo de neurônio artificial, baseado na estrutura do sistema nervoso humano, é fundamental para a operação de uma RNA. Assim, na Figura 5.12, um exemplo de neurônio artificial pode ser visto. Esta estrutura é a base para projetos de redes neurais artificiais. Portanto, de acordo com [42], é possível identificar três elementos basilares para um modelo neuronal artificial:

- Um conjunto de sinapses ou elos de conexão, cada uma com seu respectivo peso ou força própria representados, na Figura 5.12, como $x_1, x_2, ..., x_k$ e seus respectivos pesos $w_{k1}, w_{k2}, ..., w_{km}$;
- Um somador para efetuar a soma dos sinais de entrada ponderados pelas respectivas sinapses do neurônio;
- Uma função de ativação para restringir a amplitude da saída do neurônio. Esta função de ativação também pode ser referida como função restritiva, uma vez que seu objetivo é limitar o intervalo permissível de amplitude do sinal de saída a um valor finito. Assim, o neurônio artificial é "ativo" ou "inativo" para as respectivas entradas multiplicadas pelos seus respectivos pesos.

É interessante ressaltar que as RNAs obtêm sua aprendizagem a partir dos ajustes dos respectivos pesos relacionadas a cada entrada. As funções de ativação proporcionam uma análise não-linear nas redes neurais, e assim são mais eficientes na aprendizagem e conseguem identificar mais do que relações lineares simples entre as variáveis dependentes

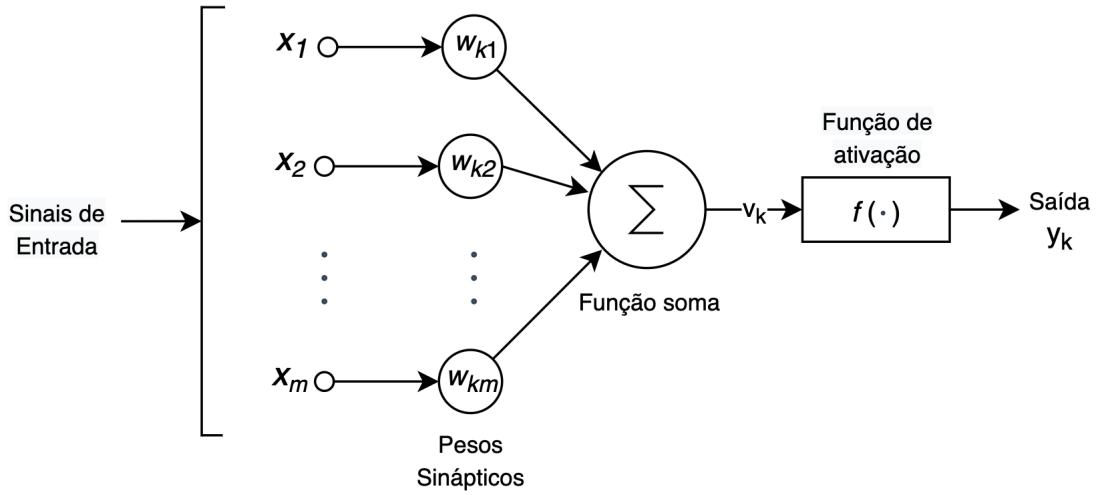

Figura 5.12: Modelo Não-linear de Neurônio Artificial [42].

e independentes. No entanto, dependem de uma quantidade massiva de dados para que seus resultados se mostrem eficazes [83].

Desta forma, um neurônio somente não é suficiente para resolver problemas complexos. Assim, a partir da junção de vários neurônios alinhados em uma ou mais camadas, o aprendizado se mantém bastante eficaz na grande maioria dos casos. Redes com várias camadas são chamadas de *Perceptron* Multicamadas (do inglês, *Multi-layers Perceptron*) (MLP) [83].

A rede neural do tipo MLP é semelhante ao *perceptron* simples de camada única proposto por [73], no entanto, tem-se mais de uma camada de neurônios, dando origem ao termo "camada oculta". As camadas ocultas são posicionadas internamente às camadas de entrada e de saída. Em casos em que não é possível separar os elementos a partir de uma única reta, o uso da MLP que gera mais de uma reta classificadora, mostra-se bastante eficiente.

5.4.7 Métricas de Avaliação

Para a correta validação de um modelo de predição para problemas de regressão, torna-se necessário a utilização de algumas métricas de avaliação [106], tais como: o Erro Médio, no inglês, *Mean Error* (ME), o Erro Absoluto Médio, no inglês, *Mean Absolute Error* (MAE), a Raiz do Erro Médio Quadrático, no inglês, *Root Mean Squared Error* (RMSE), o Erro Percentual Médio Absoluto, no inglês, *Mean Absolute Percentage Error* (MAPE) [3], o R^2, que é conhecido como coeficiente de determinação e, por fim, o R^2_{adj}, conhecido como coeficiente de determinação ajustado. Os resultados dessas métricas de avaliação podem ser vistos na Seção 6.5.1 [87].

O Erro Médio (no inglês, *Mean Error* - ME) [106], mostra o desvio médio do modelo em relação à variável. Além disso, ele oferece informações de performance do modelo em um tempo longo e o erro sistemático. Ele também apresenta resultados positivos ou negativos, sendo que quanto mais próximo do zero, melhor o resultado. O ME é calculado de acordo com a Equação 5.3, no qual n é o número de amostras ou observações, $\hat{y}$ é o valor predito pelo modelo, e y é o valor real observado durante a execução do modelo.

$$ME(y, \hat{y}) = \frac{1}{n_{amostras}} \sum_{i=1}^{n_{amostras}} (y_i - \hat{y}_i) \tag{5.3}$$

O Erro Absoluto Médio (no inglês, *Mean Absolute Error* - MAE) [106] representa o erro médio absoluto da diferença entre a observação e a previsão. Quanto mais próximo de zero, mais adequado o modelo demonstra ser. E com isso faz-se a projeção do erro total. A Equação 5.4 apresenta a formulação do MAE, sendo o n o número de amostras ou observações, $\hat{y}$ o valor predito pelo modelo e y o valor real observado durante a execução do modelo.

$$MAE(y, \hat{y}) = \frac{1}{n_{amostras}} \sum_{i=1}^{n_{amostras}} |y_i - \hat{y}_i| \tag{5.4}$$

A Raiz do Erro Médio Quadrático (no inglês, *Root Mean Squared Error* - RMSE) [106] consiste nas diferenças individuais entre a previsão do modelo e as observações. Ele mede os erros sistemáticos e randômicos, ou seja, é utilizado para medir a magnitude do erro. Nesse caso, n é o número de observações, $\hat{y}$ é o valor predito pelo modelo e y é o valor real observado durante a execução do modelo. A sua formulação é mostrada na Equação 5.5.

$$REMQ(y, \hat{y}) = \sqrt{\frac{1}{n_{amostras}} \sum_{i=1}^{n_{amostras}} (y_i - \hat{y}_i)^2} \tag{5.5}$$

O Erro Percentual Médio Absoluto (no inglês, Mean Absolute Percentage Error - MAPE) [106], também conhecido como Desvio Percentual Absoluto Médio (no inglês, *Mean Absolute Percentage Deviation* - MAPD), é uma métrica de avaliação para problemas de regressão. O objetivo dessa métrica é ser sensível a erros relativos, por exemplo, não ser alterado por uma escala global da variável de destino. Assim, esta métrica mede o erro em porcentagem. Por meio da Equação 5.6, o ϵ é um número arbitrário pequeno, mas estritamente positivo, para evitar resultados indefinidos quando y é zero, n é o número de observações, $\hat{y}$ é o valor predito pelo modelo e y é o valor real observado durante a execução do modelo.

$$MAPE(y, \hat{y}) = \frac{1}{n_{amostras}} \sum_{i=1}^{n_{amostras}} \frac{|y_i - \hat{y}_i|}{\max(\epsilon, |y_i|)} \tag{5.6}$$

Outra métrica bastante utilizada para mensurar resultados de modelo de predição é o Coeficiente de Determinação (no inglês, *Coefficient of Determination* - R^2) [106]. Este coeficiente representa a proporção da variância de y que foi explicada pelas variáveis independentes no modelo de regressão. Ele fornece uma indicação da qualidade do ajuste e, portanto, uma medida de quão bem as amostras não vistas são provavelmente previstas pelo modelo, por meio da proporção da variância explicada. Como essa variação é dependente do conjunto de dados, R^2 pode não ser significativamente comparável em conjuntos de dados diferentes e, desta forma, é utilizada no conjuntos de dados de teste. A melhor pontuação possível é 1 e pode ser negativa (porque o modelo pode ser arbitrariamente pior). Um modelo constante que sempre prevê o valor esperado de y, desconsiderando os recursos de entrada, obteria uma pontuação de R^2 de 0.0.

É possível observar a formulação dessa medida na Equação 5.7, na qual $\hat{y}$ representa o valor predito, $\bar{y}$ representa o valor médio das amostras, ou seja, $\bar{y_i} = \frac{1}{n_{amostras}} \sum_{i=1}^{n_{amostras}} y_i$, e y representa o valor real.

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n_{amostras}} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n_{amostras}} (y_i - \bar{y_i})^2} \tag{5.7}$$

Outra métrica de avaliação é o Coeficiente de Determinação Ajustado (no inglês, *Adjusted Coefficient of Determination* - $R^2_{adj}j$) [106]. O R^2_aj é uma versão modificada do R^2 que foi ajustada para o número de preditores no modelo. Assim sendo, leva-se em consideração o número de amostras em conjunto com o números de variáveis preditoras, também conhecidas como recursos ou variáveis explicativas, no modelo. Dessa maneira, a partir do R^2 é possível calcular o R^2_{adj}. A Equação 5.8 representa sua formulação, na qual n é o número de observações e p é o número de variáveis preditoras presentes no modelo.

$$R^2_{adj} = 1 - (1 - R^2) \frac{n_{amostras} - 1}{n_{amostras} - p_{preditores} - 1} \tag{5.8}$$

Desta forma, com o auxílio das métricas supracitadas é possível avaliar os resultados do modelo de conselho de preditores, presente no Capítulo 6, proposto neste trabalho. Assim sendo, procura-se adequar o modelo de regressão para que seja possível a minimização dos erros das métricas de avalização, e o aumento dos coeficientes de determinação R^2 e R^2_{adj}.

5.5 Subajuste e Sobreajuste

O principal desafio no AM supervisionado é que o modelo de aprendizado consiga predizer de maneira eficiente elementos novos e nunca observados por ele, e não apenas predizer

relações entre os mesmos dados de treinamento. Essa capacidade de performar bem em dados de entrada nunca observados é chamado de generalização [38].

Neste sentido, modelos que não são bons em generalizar são ruins em cenários reais, ou seja, a capacidade de generalização do modelo deve ser feito inteiramente a partir da base de treinamento. Assim, a capacidade apropriada para um algoritmo de AM está diretamente ligado a sua capacidade de generalização, como pode ser observado na Figura 5.13.

O processo de *underfitting*, ou subajuste, não reflete toda a capacidade de generalização do algoritmo, pois o modelo se adaptou de maneira superficial aos dados. O exemplo da Figura 5.13, no quadrante à esquerda, demonstra como o modelo se comporta ao tentar traçar uma reta linear aos dados do mesmo. Desta forma, o treinamento ou o algoritmo utilizado não possui capacidade mínima de generalização.

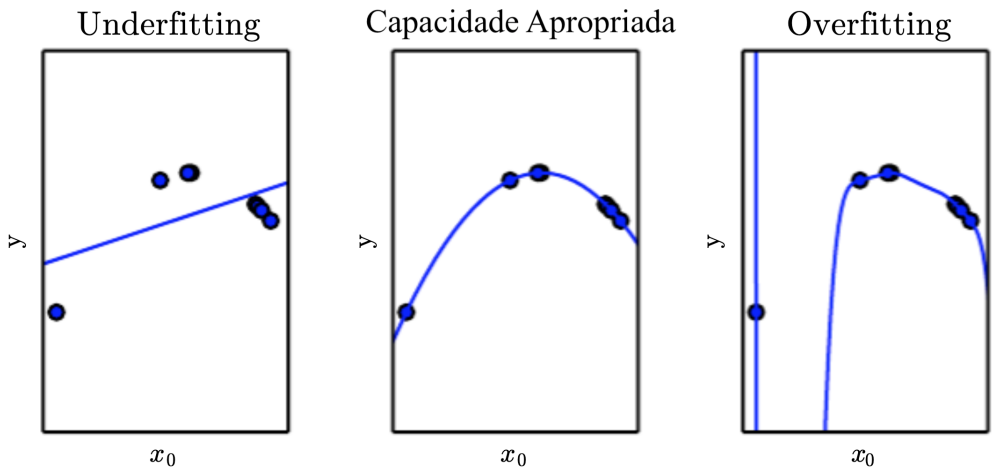

Figura 5.13: Elementos para Análise de Capacidade de Generalização [38].

Nesta mesma linha, o processo de *overfitting*, ou sobreajuste, com exemplo no quadrante à direita da Figura 5.13, é o processo de forçar a predição do modelo ao ajuste ideal dos dados, pois o modelo se adaptou demais aos dados de treinamento. Isso pode ocorrer por ajuste excessivo nos parâmetros do algoritmo, como também pela relação existente entre os dados de treinamento, causando engessamento em seus resultados.

Assim sendo, por meio de uma validação cruzada, em que é testada uma parte reservada do conjunto de dados que não foi utilizada no treino do modelo em questão, é possível ter uma ideia se o modelo possui capacidade aceitável de generalização ou não [83].

5.6 Trabalhos Relacionados

Para esta seção é importante elucidar os termos utilizados para seleção dos trabalhos relacionados. Assim sendo, os termos foram divididos conforme segue: i) “cloud resource prediction management artificial intelligence”; ii) “prediction machine learning scientific workflow”; iii) “resource and cost prediction management”; iv)“cloud resource cost and prediction management”. As buscas foram efetuadas em diferentes repositórios, tais como Portal de Periódicos da Capes, ACM, Springer, IEEE, Elsevier, e o Repositório de Teses e Dissertações da Universidade de Brasília (UnB), no período de 2017 e 2021. Desta forma, a seleção dos trabalhos foi efetuada de forma a coadunar com o tema deste trabalho, e tais trabalhos são comentados na sequência.

Os critérios para inclusão do trabalho foram: contexto do trabalho; adequação da proposta com este trabalho no resumo, introdução e conclusão; termos presentes no trabalho observado. Já os critérios de exclusão foram: desacordo com tema proposto por este trabalho; sem acesso completo ao artigo; já ter sido selecionado ou descartado. O total de trabalhos encontrados, não incluídos os descartados e incluídos os selecionados podem ser observados por meio da Figura 5.14. Diante do gráfico é possível também observar a sobreposição de alguns trabalhos, os quais foram encontrados por meio do próprio repositório ou por meio de indexação efetuada pelo Portal de Periódicos da Capes. Assim sendo, o número de trabalhos relacionados apresentados no gráfico consta maior do que os trabalhos comentados nos próximos parágrafos.

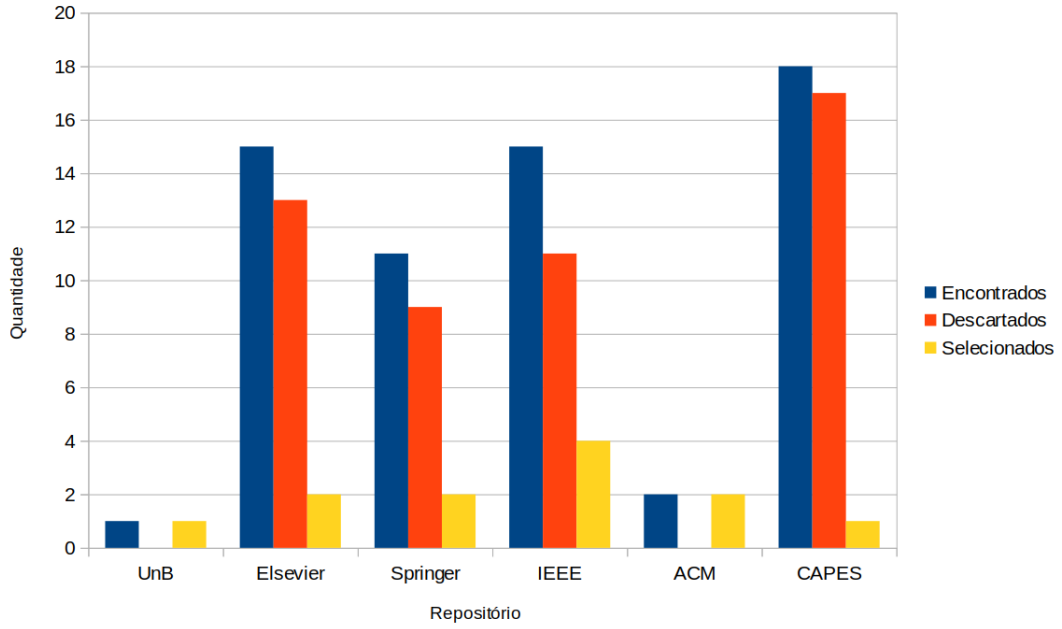

Figura 5.14: Trabalhos Encontrados, Descartados e Selecionados.

Diante disso, Baughman *et al.* [10] desenvolveram um modelo baseado em Regressão Linear para prever o tempo de execução de cada programa e tipo de instância. Para traçar um perfil de cada um desses programas, implementaram uma Regressão Linear utilizando a função de ajuste de curva do SciPy[1], que faz uso de mínimos quadrados não-lineares. O foco apresentado foi em *workflows* de genômica, pois eles são, geralmente, executados em plataformas de nuvem e os programas individuais exibem características muito diferentes. Nesse trabalho, utiliza-se dados históricos coletados de ambientes, que não são da nuvem, para criar um modelo composto que pode prever tempos de execução em um determinado tipo de instância de nuvem para um determinado tamanho de dados de entrada. Os resultados apresentados demonstram um erro médio de 17.2% em nove programas usados em *workflows* científicos de Bioinformática.

Neste mesmo seguimento, Hilman *et al.* [43] propuseram uma abordagem de aprendizado incremental *online* para prever o tempo de execução de tarefas em *workflows* científicos em nuvens. Para melhorar o desempenho das previsões eles utilizaram dados de monitoramento de recursos. Esses dados foram refinados na forma de registros de séries temporais de utilização dos núcleos de processamento da máquina, uso de memória e atividades de entrada e saída que refletem as características exclusivas da execução de um *workflow* científico.

Uma abordagem de Previsão do Conjunto Regressivo Inteligente (REAP) é proposta por Kaur *et al.* [52], o qual integra técnicas de seleção de recursos e de previsão de uso de recursos para obter alto desempenho para que seja possível predizer o uso de CPU para um *workflow* científico em ambiente de nuvem. Várias técnicas de AM são utilizados como um *pool* de preditores de regressão. Além disso, eles propuseram também parâmetros de melhoria para modelos de regressão com AM.

Ainda nesta linha, no que tange a previsão de carga de trabalho, Kumar *et al.* [57] apresentaram uma estrutura de previsão baseada em Rede Neural e Algoritmo Adaptativo de Aprendizado Evolutivo. Em vez de usar a abordagem baseada em descida do gradiente ou em retropropagação, o trabalho introduziu o algoritmo de aprendizado de evolução diferencial adaptativa (*BiPhase* - BaDE) para melhorar o processo de aprendizado da rede. O recurso de adaptação *BiPhase* permite um aprendizado adaptativo e aprimorado de padrões para melhorar a precisão da previsão, e a taxa de convergência mais rápida do modelo de previsão baseado em rede neural. O modelo objetiva predizer novas cargas de trabalho para ambientes de nuvem. No entanto, este trabalho não possui como foco auxiliar o usuário final na escolha, ou seja, seu viés está no provedor de nuvem.

Por outro lado, Kim *et al.* [54] propuseram uma estrutura de previsão de carga de trabalho em nuvem chamada *CloudInsight*, que aproveita o poder combinado de vários

[1]https://www.scipy.org/

preditores de carga de trabalho que coletivamente fornecem um "conselho de especialistas". Os pesos dos preditores neste modelo de conjunto são determinados em tempo real, com base na precisão da carga de trabalho atual usando regressão de várias classes. A estrutura do *CloudInsight* consiste em quatro componentes principais: um *pool* de preditores, um repositório de carga de trabalho, um construtor de modelos e um preditor de carga de trabalho do *CloudInsight*. A entrada dessa estrutura são as cargas de trabalho reais/atuais (por exemplo, chegadas de trabalhos), e a saída é a previsão de uma carga de trabalho no futuro próximo. O conjunto de preditores é uma coleção de preditores de carga de trabalho. Assim, o repositório de carga de trabalho armazena o histórico de tarefas da carga de trabalho e o histórico de previsões de todos os preditores locais no conjunto de preditores. O construtor de modelos é responsável por criar um modelo de previsão de conjunto, avaliando o desempenho dos preditores no conjunto de preditores.

Zhong *et al.* [122] propuseram um modelo de previsão de carga na nuvem baseado na Máquina de Vetor de Suporte de *Wavelet* Ponderada (WWSVM), para prever a sequência de carga do *host* no *datacenter* na nuvem. O modelo combina a transformada *wavelet* [67] e a máquina de vetores de suporte para integrar as suas vantagens, e atribui peso à amostra, o que reflete a importância de diferentes pontos da amostra e melhora a precisão da previsão de carga.

O problema de previsão futura de recursos é resolvido a partir de um modelo de redes neurais em dois estágios, com classificação e regressão para prever a acurácia da predição dos recursos, proposto por Kumar *et al.* [58]. No primeiro estágio, o modelo categoriza em três classes, tais como acima, normal e sob adaptação. O segundo estágio é utilizado para prever as demandas futuras de utilização de recursos a partir de uma regressão em uma rede neural em classificação de resultados. Para isso, são considerados dois parâmetros da Máquina Virtual (VM) que são a utilização da CPU e a utilização da memória RAM para identificar cada carga da VM.

No trabalho de Yu *et al.* [120], os autores utilizaram uma abordagem baseada em *pool* de tarefas, em que o conhecimento sobre as cargas de trabalho de um grande conjunto de tarefas é usado para ajudar a prever as cargas de trabalho de novas tarefas. Em particular, desenvolvem uma abordagem de aprendizado baseada em *cluster* para realizar o conceito de *pool* de tarefas. Assim, o conjunto de tarefas é agrupado com base em suas cargas de trabalho, e uma rede neural é usada para aprender as características das cargas em cada *cluster*. Quando um novo trabalho chega, usa seu padrão de carga de trabalho inicial e parâmetros de envio para localizar o *cluster* ao qual ele pertence. Em seguida, a rede neural correspondente é usada para prever a carga de trabalho no futuro.

Um modelo de predição de tempo e dimensionamento de recursos é proposto por [100] para o contexto de nuvens federadas, obtendo bons resultados e um modelo de análise

de resíduos completo para avaliar o modelo proposto. Desta forma, utiliza-se o modelo para predizer recursos necessários por meio da meta-heurística GRASP [94] e a uma Regressão Linear Múltipla para predizer o tempo de execução de *workflows* científicos de Bioinformática. O autor propõe uma ferramenta de predição que seja integrada à plataforma BioNimbuZ [103], em sua primeira versão.

Diante do exposto, este trabalho propõe expandir a proposta de [100] a partir de métodos de AM com foco em auxiliar o usuário final na escolha eficiente de recursos em um ambiente de nuvens federadas. Para isso, será adotada a abordagem de conselho de preditores descrita na Seção 6.5.

Na Tabela 5.1 é apresentado um resumo de cada trabalho descrito anteriormente. Como pode ser observado, a maioria dos trabalhos utiliza ambiente de nuvem, e apenas este trabalho e [99] têm em suas propostas o escopo de federação de nuvens. Além disso, diferentes técnicas de predição são utilizadas, tais como AM [43, 52, 54, 122, 58, 57, 120], Métodos Estatísticos [10, 43, 99] e/ou Meta-heurísticas [99].

Tabela 5.1: Trabalhos Relacionados à Predição de Recursos.

Trabalho	**Ambiente**	**Técnica**	**Predição**	**Usuário**
Baughman *et al.*[10]	Nuvem	Regressão Linear Múltipla	Recursos	Não
Hilman *et al.*[43]	Nuvem	Séries temporais, RNA e K-NN	Tempo	Não
Kaur *et al.*[52]	Nuvem	*Pool* de Preditores com AM	Uso de CPU	Não
Kumar *et al.*[57]	Nuvem	RNA	Carga de trabalho	Não
Kim *et al.*[54]	Nuvem	*Pool* de Preditores com AM	Carga de trabalho	Não
Zhong *et al.*[122]	Nuvem	SVM Adaptado	Carga de trabalho	Não
Kumar *et al.*[58]	Nuvem	RNA	Carga de trabalho	Não
Yu *et al.*[120]	Nuvem	RNA	Carga de trabalho	Não
Rosa[99]	Nuvens Federadas	Regressão Linear Múltipla e GRASP	Tempo, Custo e Recursos	Sim

Assim sendo, apenas em [99] há um investimento direto nos interesses dos usuários, tais como custo financeiro, tempo e *feedback* da execução de *workflows* científicos de Bioinformática. Dessa maneira, há a possibilidade do usuário poder escolher entre uma execução de baixo custo ou de alto desempenho. Com isso, nota-se na Tabela 5.1 que

nenhum outro trabalho apresentado possui objetivos diretos em auxiliar o usuário final na tomada de decisão no que tange a alocação de recursos em ambiente de nuvem ou de federação de nuvens. Além disso, é possível notar na Tabela 5.1 que nenhum trabalho anterior teve foco em predição de ambiente de nuvem federada.

A maioria dos trabalhos apresentados está interessada em predizer novas cargas de trabalho em um ambiente de nuvem com viés do provedor de nuvem. Assim sendo, em Kim *et al.* [54] é proposta uma estrutura de previsão de carga de trabalho em ambiente de nuvem, além da implementação de um conselho de especialistas. Diante do exposto, este trabalho propõe a implementação de um conselho de preditores conforme proposto em [54]. Esse conselho de preditores é implementado neste trabalho a partir do desenvolvimento de uma rede neural MLP (ver Seção 6.5), na qual os pesos são definidos a partir dos resultados da predição de modelos anteriores ao conselho (ver Seção 6.4). Logo, a proposta apresentada neste trabalho se difere do conselho de especialistas apresentado por [54].

Assim sendo, nenhum desses trabalhos utiliza um modelo de conselho de preditores para ambiente de nuvens federadas como método de predição de tempo de execução de *workflows* científicos de Bioinformática.

O modelo de RNAs se adaptou melhor ao problema de predição de tempo de execução de tarefas ou *workflows* científicos de Bioinformática em um ambiente de nuvem (ver Tabela 5.1), quando comparado com os outros modelos citados. O modelo de RNAs apresenta bons resultados nos diversos trabalhos apresentados, no entanto, nenhum destes trabalhos utilizam esta implementação em conjunto com outros modelos de aprendizado de máquina com o intuito de melhorar a acurácia da predição.

Diante do exposto, este trabalho propõe uma evolução do trabalho de [99] como um serviço, no que tange o modelo de predição de tempo de *workflows* científicos de Bioinformática, além da implementação deste serviço a partir da plataforma de federação de nuvens BioNimbuZ 2. Os detalhes do serviço proposto serão descritos no Capítulo 6.

5.7 Considerações Finais

Este trabalho tem foco no aprendizado supervisionado para problemas de regressão. Assim, propõe um serviço de predição de tempo de execução de programas presentes em *workflows* científicos de Bioinformática, e para isso utiliza modelos que resolvem problemas de regressão, ou seja, quando o resultado esperado é um valor numérico contínuo. Em um segundo momento, o serviço de predição proposto utiliza a meta-heurística GRASP para predizer o recurso considerado mais adequado para o *workflow* científico de Bioinformática a ser inserido como entrada do serviço.

Deste modo, este capítulo enfatiza as diversas categorias de aprendizado de máquina, além de demonstrar os riscos na falta de análise da capacidade de generalização de modelos de AM supervisionado.

Neste sentido, alguns dos modelos de aprendizado supervisionado que resolvem problemas de regressão, vistos na Seção 5.4, estão presentes na estrutura de conselho de preditores apresentada no Capítulo 6.

Capítulo 6

sPCRAM - Serviço de Predição de Recursos Computacionais com Aprendizado de Máquina

Neste capítulo são apresentados os métodos utilizados no desenvolvimento do serviço de predição de recursos para a execução de *workflows* científicos de Bioinformática em ambiente de nuvem federada por meio de métodos de aprendizado de máquina. Para isso, a Seção 6.1 apresenta a visão geral da proposta e descreve as estruturas de serviços de predição. Em seguida, a Seção 6.2 descreve a estrutura do serviço proposto neste trabalho. A Seção 6.3 apresenta a metodologia utilizada para a geração e a coleta de dados de execuções em ambientes de nuvem, além da implementação do modelo de seleção de variáveis e a análise de multicolinearidade dessas variáveis. Em seguida, na Seção 6.4 são descritos os modelos desenvolvidos para servirem de componentes para o modelo de conselho de preditores proposto neste trabalho, descrito na Seção 6.5, assim como as métricas de avaliação dos modelos. Em seguida, são apresentados os cenários de avaliação na Seção 6.6. Também é apresentada a meta-heurística GRASP na Seção 6.7 e seus resultados na Seção 6.8. Por fim, a Seção 6.9 apresenta as considerações finais deste capítulo.

6.1 Considerações Iniciais

O Serviço de Predição de Custos e Recursos Computacionais com Aprendizado de Máquina (sPCRAM), proposto neste trabalho, tem como objetivo auxiliar o usuário final na escolha de um ambiente computacional em uma plataforma de nuvem federada, para execução adequada de seus *workflows* científicos de Bioinformática. A proposta é que o sPCRAM escolha, de maneira transparente, a infraestrutura a ser utilizada na federação,

garantindo uma relação custo x tempo que atenda às necessidades do usuário. Assim, o serviço proposto pretende fornecer ao usuário uma escolha baseada em suas necessidades, podendo oscilar entre uma execução mais barata ou mais rápida.

Assim sendo, várias características devem ser consideradas no desenvolvimento de um modelo de predição em um ambiente de nuvens federadas. Logo, entre essas características destacam-se os variados tipos de máquinas virtuais oferecidas pelos provedores, o número máximo de máquinas virtuais que podem ser alocadas por um provedor específico, o custo da comunicação entre os recursos de diferentes nuvens que compõem o ambiente federado, o custo da transmissão de dados (*download, upload*), e o custo da transmissão de dados do armazenamento, entre outras [100].

Para isso, é necessário analisar o *workflow* científico de Bioinformática para que seja possível extrair características que possam subsidiar a escolha adequada de máquinas na nuvem, objetivando diminuir as falhas por falta ou por ociosidade de uma escolha inadequada do ambiente de execução. Assim, considerando um ambiente de federação de nuvens, predizer tais elementos de um *workflow* torna-se uma tarefa bastante complexa, pois os recursos a serem escolhidos têm um crescimento exponencial em relação à quantidade e diversidade de nuvens presentes na federação [24].

Esse tipo de predição contribui para uma visão holística da execução do projeto de Bioinformática, e auxilia na tomada de decisões sobre as condições em que o projeto pode ser executado, minimizando os riscos de falhas no projeto. A predição do dimensionamento de recursos também é importante no sentido de manter a utilização dos recursos de acordo com a demanda, sem desperdícios de recursos, podendo, por exemplo, ser aplicado para garantir a Qualidade do Serviço (QoS) [53].

Para um serviço de predição é interessante que o sistema seja capaz de predizer flutuações de demanda e ações dos serviços, para que tome decisões inteligentes relacionadas a custos, tempo, qualidade de serviço, dimensionamento dinâmico, entre outras. Esta especulação da necessidade futura em termos de computação, de armazenamento, de rede, de requisições, e etc., para julgar custos, tempo e recursos, implica em uma previsão eficaz de um ambiente controlável, que aumenta a confiabilidade do sistema [45].

Diante disso, para provisionar recursos de maneira adequada à demanda de um *workflow* específico, faz-se necessária a construção de modelos que trabalham com o intuito de predizer os recursos que mais se adéquam no contexto do problema. Todavia, estimar a quantidade necessária de recursos está longe de ser uma tarefa trivial, pois as execuções se tornam uma caixa preta, com o risco de resultados imprecisos e de alto custo [20]. Com isso, a má escolha na quantidade ou até mesmo na capacidade do recurso pode produzir impactos negativos na execução do *workflow* e afetar o seu custo financeiro [100]. Este tipo de predição se torna muito importante em um ambiente de nuvens federadas, pois

os usuários demandam execuções de *workflows* de grande escala, alocando recursos e pagando somente pelo que usam. Algumas técnicas podem ser utilizadas para isso, sendo elas: Métodos Estatísticos [10, 99], Aprendizado de Máquina [43, 52, 54, 57, 58, 120, 122] e Meta-heurísticas [23, 24, 85]. Assim, a próxima seção apresenta as técnicas comumente utilizadas para a predição de recursos com foco em modelos de Aprendizado de Máquina e Estatística.

Assim sendo, este trabalho propõe um abordagem de conselho de preditores, em conjunto da meta-heurística GRASP, com o objetivo de predizer o recurso adequado que atenda as expectativas de custo e tempo para execução de *workflows* científicos de Bioinformática, em ambiente de nuvem federada.

6.2 Estrutura do sPCRAM

O sPCRAM é um serviço de predição de recursos que objetiva recomendar o recurso mais adequado para a execução de *workflows* de Bioinformática. Para isso, ele considera os interesses dos usuários, permitindo que a escolha seja dirigida a custo e/ou tempo de execução, subsidiando assim uma execução de baixo custo e/ou com menor tempo de execução. Para isso, a estrutura do serviço de predição foi dividida em quatro fases, inicialmente proposta por [100], sendo elas: Coleta de Informações, Pré-processamento, Processamento e *Feedback*, conforme apresentado na Figura 6.1.

O fluxo de execução do sPCRAM é apresentado na Figura 6.1, a qual detalha as etapas da execução do serviço proposto.

Inicialmente, na primeira fase do fluxo de execução, são coletadas as informações do usuário com o intuito de restringir a seleção de recursos em fases posteriores. O usuário deverá inserir parâmetros como custo máximo financeiro e tempo máximo de execução a ser considerada na predição. Essas informações irão especificar a lista de recursos compatíveis com as restrições impostas. Assim, a estrutura poderá variar/balancear entre uma execução de baixo custo com maior tempo de duração, ou de alto custo com menor tempo de duração. Desta forma, é possível considerar o desejo do usuário no que tange tempo e custo da execução do *workflow* repassado.

A base de dados será utilizada na fase seguinte para efetuar a predição de programas conhecidos, ou para programas ainda não conhecidos pela plataforma. No primeiro caso, será utilizado um conselho de preditores, descrito na Seção 6.5. No segundo caso, quando não há conhecimento prévio na base, será utilizado o modelo de Floresta Aleatória. Assim, esses modelos subsidiarão na predição de tempo para que posteriormente a seleção de recursos, considerando as restrições do usuário, seja feita por meio da meta-heurística GRASP.

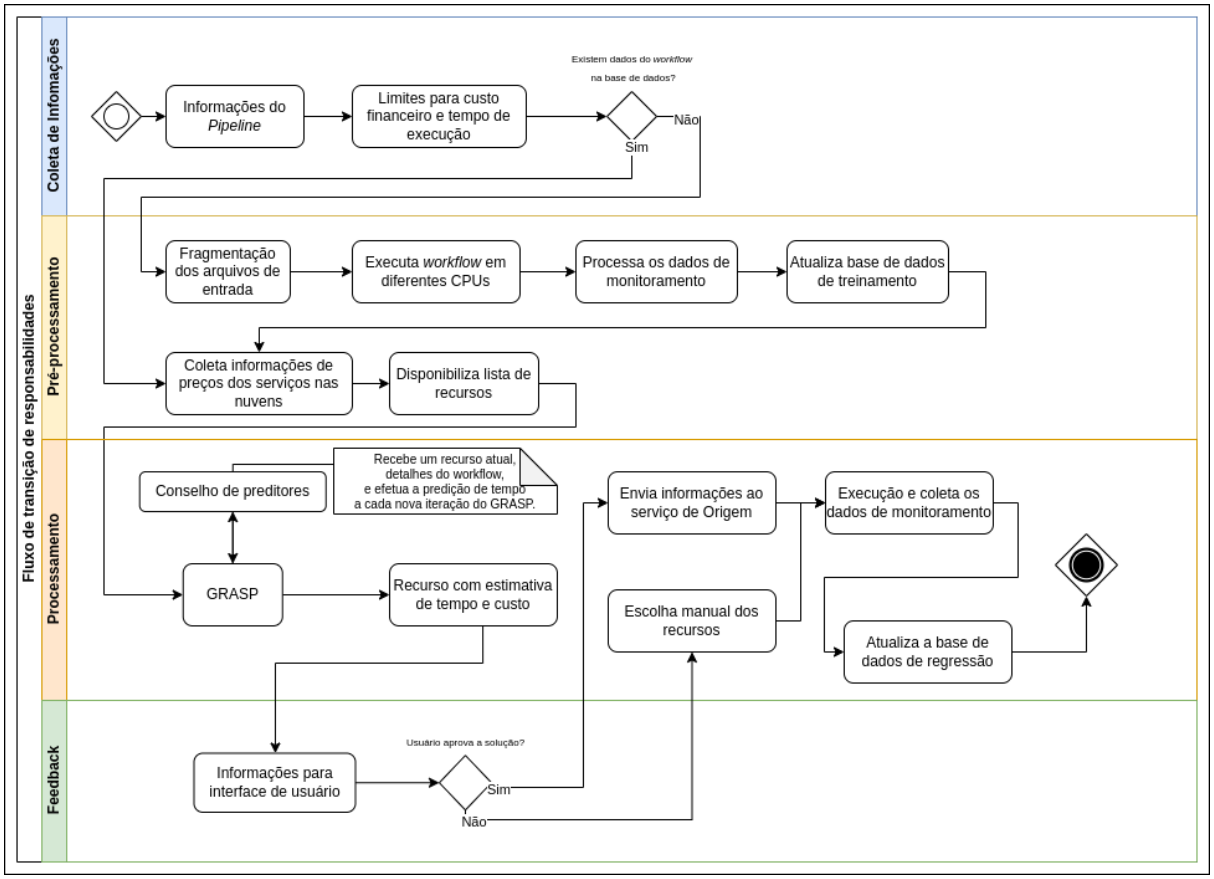

Figura 6.1: Arquitetura do Fluxo de Execução do sPCRAM, adaptado de [100].

Neste contexto, em previsões de softwares desconhecidos as estimativas podem não ser confiáveis. Assim, esta fase também engloba mecanismos que criam, a partir de pequenas porções de dados de entrada dos *workflows*, denominadas de dados sintéticos, informações de execução do software até então não conhecido na base de execuções histórica [99]. Essas execuções são replicadas em máquinas virtuais com um número diferente de núcleos, com 2, 4 e 8 núcleos. As execuções são adicionadas à base de dados histórica a fim de obter melhorar no modelo preditivo. Com isso, esta fase tem o objetivo de coletar informações descritas pelo usuário e do ambiente computacional, com o intuito de reportar as informações necessárias para a terceira fase do processo de predição.

A fase de Processamento é composta por um algoritmo baseado na abordagem GRASP, que consome as informações da fase anterior, com o objetivo de minimizar custos e tempo de execução, levando em consideração as opções descritas pelo usuário. O objetivo desta fase é disponibilizar soluções sub-ótimas, através da utilização da Meta-heurística, em tempo viável [100].

Após a construção da solução sub-ótima, o serviço de predição responde a requisição para a interface do usuário, a partir da última fase, o *feedback* da solução encontrada

e as estimativas de tempo e custo da execução. Assim, após o término da execução do *workflow*, alimenta-se a base de dados histórica, objetivando registrar os dados da predição para garantir melhorias para as próxima predições.

Após as execuções do *workflow* (ver Seção 2.4), como descrito na próxima seção (Seção 6.3), foram coletados os dados de monitoramento, por meio de *scripts*, que compuseram a base de dados utilizada para o treinamento dos modelos de aprendizado de máquina desenvolvidos neste trabalho.

6.3 Geração e Pré-processamento de Dados

Para que fosse possível predizer o tempo de execução de programas presentes em *workflows* científicos de Bioinformática foi necessário que os programas presentes na estrutura do *workflow* fossem observados em ambiente real de nuvem. Neste sentido, considerando um ambiente de nuvem federada, é possível que as características intrínsecas ao hardware como, por exemplo, a taxa de leitura e escrita de disco influencie no tempo de execução em relação ao provedor.

Diante do exposto, algumas variáveis foram mapeadas em um ambiente de nuvens para que pudessem subsidiar o esquema de predição proposto neste trabalho. Assim, este trabalho considerou as variáveis que envolvem as restrições do usuário e do ambiente de nuvem federada. Essas varáveis, baseadas no estudo de Coutinho *et al.* [23], definem a estrutura do serviço de predição, e estão descritas a seguir:

- Variáveis relacionadas ao usuário final:
  - Custo financeiro máximo permitido para a execução do *workflow*;
  - Tempo máximo de duração da execução do *workflow*;
- Variáveis relacionadas ao ambiente de nuvem federada:
  - Conjunto de máquinas virtuais oferecidas por um provedor x;
  - Custo da alocação da máquina virtual x por um determinado período de tempo;
  - Capacidade de armazenamento da máquina virtual x;
  - Capacidade de memória RAM da máquina virtual x;
  - Poder de processamento da máquina virtual x;
  - Custo do *upload* para uma máquina virtual x;
  - Custo do *download* a partir de uma máquina virtual x;
  - Custo do armazenamento dos dados transmitidos.

Com essas informações, diversas possibilidades são selecionadas a partir da infraestrutura da federação de nuvens, e assim é necessário aplicar alguma técnica com o intuito de selecionar boas escolhas em um tempo adequado para estimar uma solução viável de recursos.

Dessa maneira, para a seleção de máquinas candidatas, este trabalho utiliza uma abordagem baseada na Meta-heurística GRASP [94] para a seleção dos recursos em um ambiente de federação de nuvens. O uso desta abordagem se deu por dois motivos, o primeiro por já ter sido desenvolvida no trabalho de [99] com resultados satisfatórios para o contexto deste trabalho, e segundo por ser altamente eficiente para resolver problemas de alocação em *workflows* científicos em outros domínios, como pode ser visto no trabalho de Coutinho *et al.* [23].

Na Figura 6.2 é possível visualizar o esquema de preparação do ambiente de execução do *workflow*, conforme visto na Seção 2.4, utilizado para geração da base de dados de treinamento. Desta forma, com o objetivo de facilitar as execuções em diversas máquinas de diferentes provedores, o encadeamento do *script* de preparação deve ser executado manualmente.

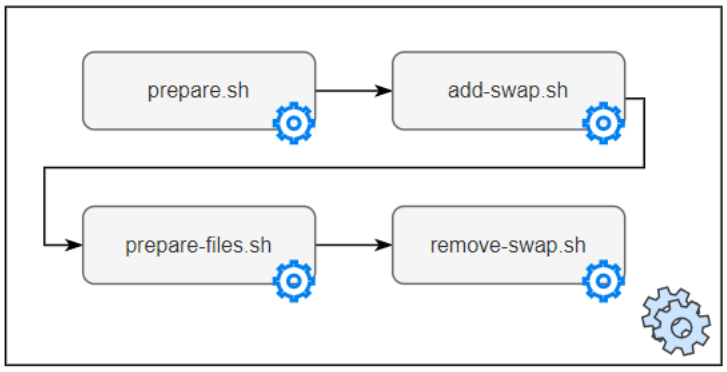

Figura 6.2: Preparação de Ambiente de Execução do *Workflow* em Ambiente de Nuvem.

Toda a cadeia é iniciada a partir da execução do *script* em "prepare.sh". Esse *script* instala e configura todos os programas necessários para a execução do *workflow* selecionado para o experimento presente neste trabalho. A partir da sua finalização, um próximo *script* responsável por ativar o arquivo de *swap* da máquina é executado. A criação do arquivo de *swap* se faz necessária pelo fato de que, em máquinas com menos do que 4GB de memória RAM, alguns programas, que fazem uso do arquivo de entrada ".fasta" ou ".fastq", não conseguem carregar o arquivo em memória RAM. Assim sendo, é preciso manter um arquivo de *swap* para o próximo passo de preparação dos arquivos de entrada, os quais serão utilizados na execução do *workflow*.

O arquivo "prepare-files.sh" é executado, e neste processo todos os arquivos necessários para a execução do *workflow* são baixados de diferentes repositórios de Bioinformática

e renomeados para a posterior utilização. Em seguida, o arquivo "remove-swap.sh" é executado para que o arquivo de *swap*, criado na segunda parte deste *workflow*, seja removido. Desta forma, todas as saídas deste processo de preparação, convergem para a montagem do ambiente para eficaz execução do *workflow* selecionado para o experimento deste trabalho.

Após a completa execução dos *scripts* que preparam o ambiente de execução, o arquivo "execution.sh" é iniciado. Cada execução do experimento foi monitorada, por meio da ferramenta *Versatile Resource Statistics Tool* (DSTAT) [119], na versão 0.7.3. A ferramenta DSTAT efetua o monitoramento dos parâmetros da máquina, tais como memória RAM, cache, utilização de CPU, disco, entre outras funções. Assim, em conjunto com o "execution.sh", um outro processo, o "dstat.sh", é executado.

Desta forma, é possível monitorar a execução do *workflow* sem interferências significativas que possam descredibilizar os valores dos dados coletados, uma vez que os processos que efetuam o monitoramento através da ferramenta DSTAT não influenciam consideravelmente nos resultados da análise dos parâmetros da máquina. Nesse contexto, a interferência ocorre, no entanto, é insignificante para o contexto geral da execução. Assim, as saídas do processo "dstat.sh" são, posteriormente, preparados para utilização na fase de treinamento dos modelos de aprendizado de máquina.

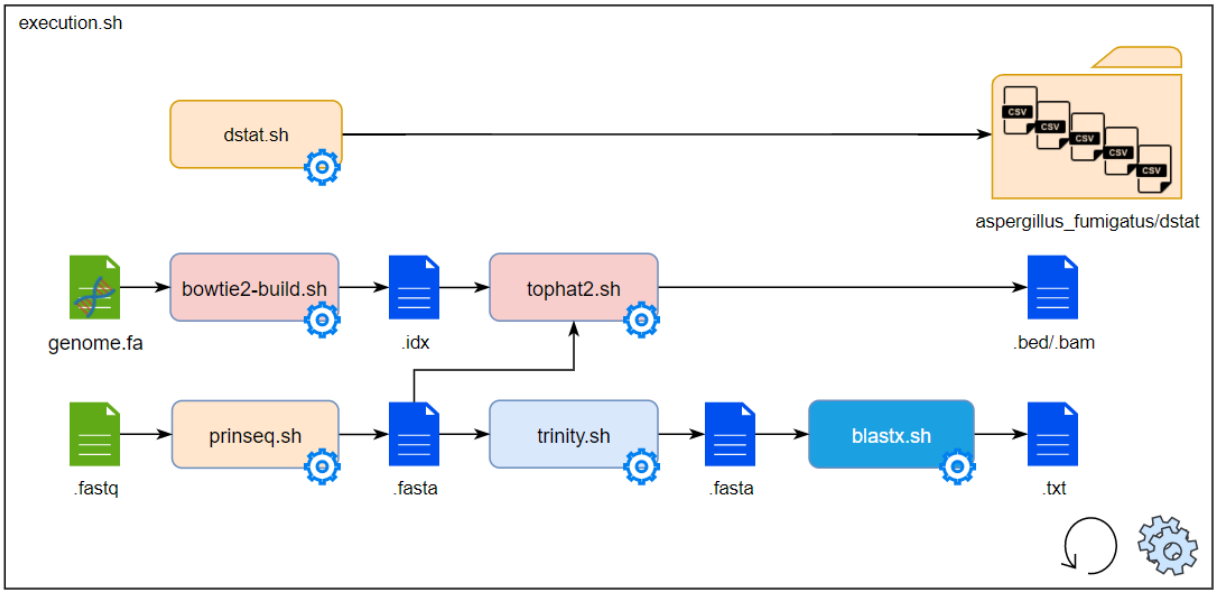

Figura 6.3: *Script* de Execução de *Workflow* em Ambiente de Nuvem.

Assim, após a conclusão do processo de execução do *workflow* de experimento, os arquivos resultantes, contendo todos os dados de monitoramento, de cada programa executado são gerados, conforme apresentado na Figura 6.3. Logo, o processo que faz o armazenamento destes arquivos é iniciado, e assim é possível efetuar análise e montagem

do conjunto de dados de entrada para os algoritmos de aprendizado de máquina. O *script* "delete-generated-files.sh" é executado para que não sejam consumidos recursos de armazenamento da máquina virtual, pois para este contexto o que interessa são os arquivos de análise da máquina gerados pela ferramenta DSTAT, enquanto o *workflow* é executado. Assim, o último *script* do *workflow* é iniciado, o arquivo "push.sh". Esse *script* efetua o envio dos arquivos de análise gerados para um repositório Git [109], conforme apresentado na Figura 6.4.

A automatização da fase de coleta dos dados de execução está estruturada para que, após cada execução, os arquivos de observações das execuções sejam enviados para esse repositório de armazenamento. Como consequência, para cada programa presente no *workflow* é gerado um arquivo em formato de Valores Separados por Vírgulas (CSV) [95] com todos os parâmetros disponíveis na ferramenta DSTAT, conforme apresentado na Figura 6.3. Assim sendo, a partir da execução dos *scripts* supracitados, os parâmetros de execuções, em período de teste gratuito, disponibilizado por cada provedor, foram coletados conforme apresentado na Tabela 6.1.

Todo esse fluxo é parametrizado para que a execução dos arquivos "execution.sh", "delete-generated-files.sh" e "push.sh" sejam executados k vezes. Neste experimento, é possível observar na Tabela 6.1 a quantidade de execuções efetuadas para cada configuração de máquina para a respectiva nuvem em período de teste gratuito. Com isso, houve a coleta de dados de execução em diferentes máquinas e em diferentes provedores de nuvens, conforme os termos de uso de cada provedor, no momento do desenvolvimento deste trabalho.

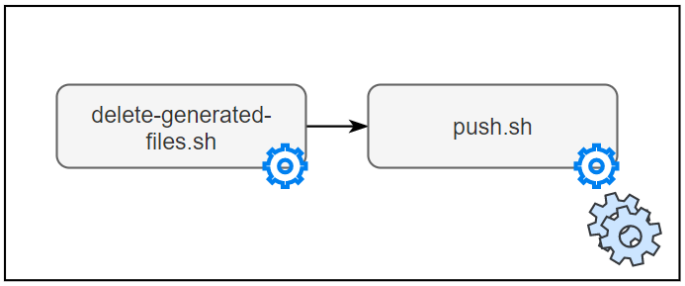

Figura 6.4: Envio de Dados de Monitoramento para Repositório Git.

Após a coleta dos parâmetros de execução em ambiente de nuvem, a estruturação dos arquivos de entrada foi feita por meio da inserção dos dados coletados em um banco de dados relacional. A intenção desta fase é o completo armazenamento dos dados do experimento, assim como a posterior utilização para o tratamento dos dados pela Linguagem de Consulta Estruturada (SQL) [31], no inglês, *Structured Query Language*.

Logo, com os dados coletados, torna-se necessário um método para a escolha das variáveis que mais influenciam no modelo e, desta forma, otimizar o número de variáveis que

Tabela 6.1: Execuções de *Workflow* em Ambiente de Nuvem.

Plataforma	CPU	RAM	Parte da entrada	Execuções
AWS	2	2	SRR836437	422
AWS	2	4	SRR836437	69
AWS	4	7	SRR836437	901
AWS	8	15	SRR836437	904
AZURE	2	4	SRR836437	300
AZURE	4	8	SRR836437	1006
GCP	1	2	SRR836437	994
GCP	2	2	SRR836437	877
GCP	4	4	SRR836437	1140
GCP	4	8	SRR836437	1005
GCP	6	6	SRR836437	1000
GCP	6	8	SRR836437	360
GCP	8	8	SRR836437	1056
GCP	8	8	SRR836438	1474

compõem a equação de regressão. O método utilizado neste trabalho foi o *Stepwise* [11], a ser descrito na Seção 6.3.1.

6.3.1 Modelo de Seleção de Variáveis

Durante a escolha das variáveis, que compõem as entradas dos modelos preditivos deste trabalho, foram selecionadas empiricamente vinte e duas variáveis relevantes observadas pelo DSTAT, conforme a Tabela 6.2. Essas variáveis representam as observações das execuções dos softwares presentes no *workflow* de experimento (ver Seção 2.4). Diante disso, é importante que a base de treinamento seja capaz de generalizar novas previsões com resultados considerados aceitáveis, para que os modelos possam obter maior acurácia na predição, em previsões futuras.

Assim, foi necessário definir um método para a escolha das variáveis que comporiam a base de treinamento dos modelos preditivos. O método de força bruta é utilizado de maneira que as variáveis são inseridas e removidas da base de treinamento de forma a se chegar no melhor conjunto possível. Entretanto, o método é extremamente oneroso computacionalmente [11]. Diante dessa limitação, foi implementado neste trabalho, a partir da linguagem Python 3.7, o método semi-automático denominado *Stepwise* [11]. Este método é utilizado para selecionar quais variáveis mais influenciam no modelo, e assim otimizar o número de variáveis a compor o modelo preditivo. Então, de maneira iterativa, o procedimento constrói modelos de regressão pela adição ou remoção de variáveis em cada etapa, por meio de um critério expresso em termos de um teste formado pela comparação de um modelo com o seu submodelo.

Tabela 6.2: Variáveis de Monitoramento.

Variável	Descrição
`usr_total_cpu_usage`	Uso de CPU pelos processos de usuário
`sys_total_cpu_usage`	Uso de CPU pelos processos de sistema
`idl_total_cpu_usage`	CPU ociosa
`wai_total_cpu_usage`	Processos em espera
`stl_total_cpu_usage`	Processos aguardando o *hypervisor*
`used_memory_usage`	Uso de memória
`free_memory_usage`	Memória livre
`buff_memory_usage`	Uso em *buffer*
`cach_memory_usage`	Uso em cache
`read_dsk_total`	Operações de leitura em disco
`writ_dsk_total`	Operações de escrita em disco
`run_procs`	Processos em execução
`blk_procs`	Processos bloqueados
`new_procs`	Processos novos
`cloud_name`	Nuvem utilizada
`cpu`	Quantidade de CPUs da máquina
`ram_gb`	Quantidade de memória RAM da máquina
`program`	*Software* observado
`extension_input_file`	Extensão do arquivo de entrada do software
`input_file_size`	Tamanho do arquivo de entrada
`input_file_lines`	Quantidade de linhas no arquivo de entrada
`total_seconds`	Tempo total de execução do *software* em segundos

Desta forma, inicialmente tem-se um modelo com uma variável preditora com a mais alta correlação com a variável que se pretende predizer, de tal forma que a cada nova adição de variável é realizada uma verificação para eliminação de variáveis que possuem alto índice de redundância com as outras variáveis presentes no momento. A finalização do procedimento ocorre quando não há mais variáveis a serem incluídas ou removidas do modelo [11]. Em determinadas situações é necessário efetuar transformações dos valores das variáveis preditoras, com o intuito de aumentar a relação de generalização do modelo.

Em [100], foram feitas algumas transformações, tais como "*log(k)*" ou "fatorCor = 4/var", na qual *var* é a variável que se pretende efetuar a transformação. Assim sendo, algumas dessas transformações foram reproduzidas também neste trabalho, com o mesmo objetivo no que tange os modelos preditivos de aprendizado de máquina. Neste sentido, as variáveis deste trabalho, estabelecidas pelo procedimento de *stepwise*, para compor a base de treinamento dos modelos preditivos podem ser vistas na Tabela 6.3. Além disso, como as transformações aplicadas neste trabalho podem ser vistas na Tabela 6.4. Nas colunas da Tabela 6.4 estão as funções de transformação aplicadas em cada variável presente nas linhas.

A função de padronização é aplicada a partir de todo o conjunto de dados de treinamento. Essa técnica de padronização ignora a forma da distribuição e transforma o dado para uma forma com média próxima de zero e um desvio padrão próximo a um. Dessa maneira, assume-se que não terá valores discrepantes nos dados. Esse processo utiliza o componente *StandardScaler* da biblioteca Scikit-learn [87]. A função logarítmica foi aplicada nas variáveis "`total_seconds`" e "`input_file_lines`" que representa o tempo em segundos que determinado programa passou em execução no *workflow*, pois, desta forma, supõe-se que o tempo segue uma distribuição exponencial [100]. A Codificação de Rótulo (no inglês, *Label Encoder*) efetua o processo de conversão dos valores categóricos em valores numéricos para que seja possível inseri-los nos modelos de predição voltados para regressão. Desta forma, foi observado empiricamente que tais transformações aumentaram a acurácia do modelo de predição proposto neste trabalho.

Tabela 6.3: Variáveis do Modelo Preditivo Selecionadas pelo Método *Stepwise*.

Variável	Descrição
`cpu`	Quantidade de CPUs da máquina
`ram_gb`	Quantidade de memória RAM da máquina
`program`	*Software* observado
`cloud_name`	Provedor de nuvem utilizado
`input_file_lines`	Quantidade de linhas no arquivo de entrada
`usr_total_cpu_usage`	Uso de CPU pelos processos de usuário

Tabela 6.4: Variáveis do Modelo Preditivo após as Transformações.

Variável	log	fatorCor	Codificação de Rótulo	Padronização
`cpu`	NÃO	SIM	NÃO	SIM
`ram_gb`	SIM	NÃO	NÃO	SIM
`program`	NÃO	NÃO	SIM	SIM
`cloud_name`	NÃO	NÃO	SIM	SIM
`input_file_lines`	SIM	NÃO	NÃO	SIM
`usr_total_cpu_usage`	NÃO	NÃO	NÃO	SIM
`total_seconds`	SIM	NÃO	NÃO	SIM

A relevância da transformação logarítmica da variável "`total_seconds`" pode ser observada nos histogramas das Figuras 6.5 e 6.6. O histograma da Figura 6.5 mostra que o impacto das observações próximas de zero é elevado, impactando no fato de não ser possível analisar precisamente qualquer outra diferença externa. Diante disso, a quantidade de observações próximas a zero fazem o restante das observações terem pouca relevância. Por outro lado, o histograma da Figura 6.6 exemplifica como a transformação na variável

"`total_seconds`" efetua um balanceamento dos pesos de cada uma das observações, o que permite que os modelos de regressão consigam predizer com melhor acurácia.

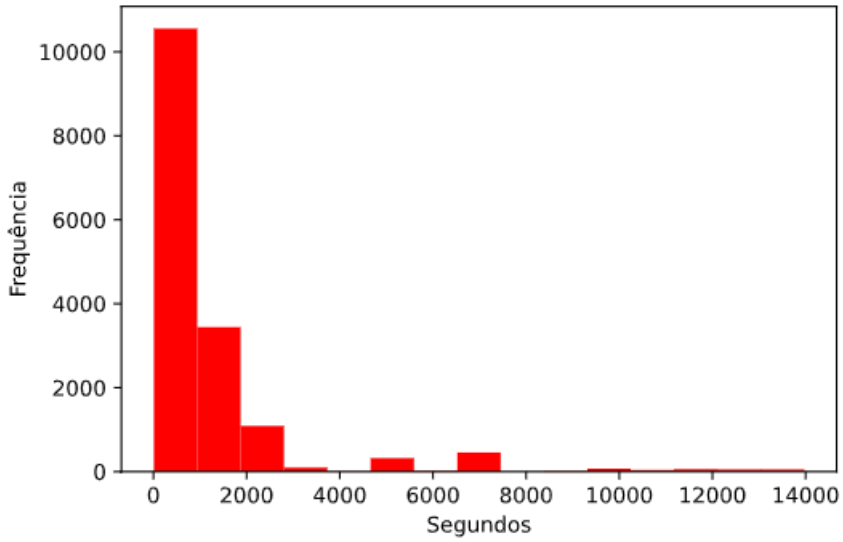

Figura 6.5: Histograma da Variável total_seconds.

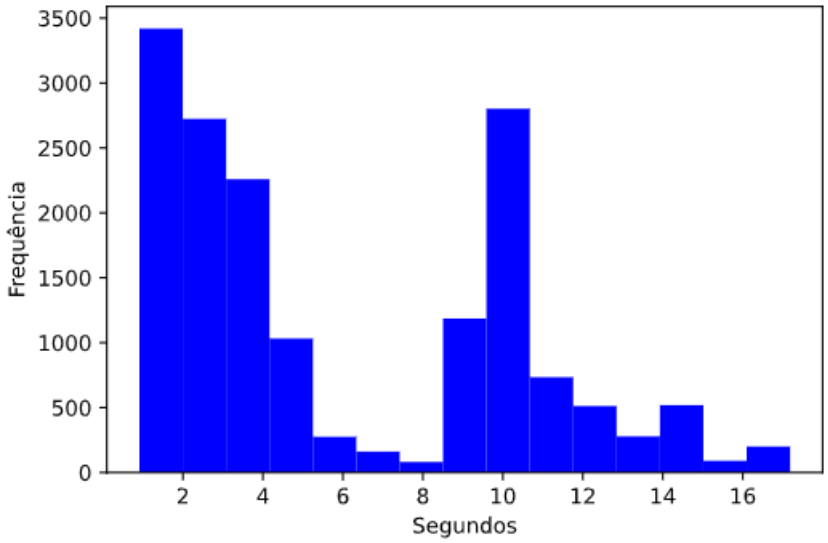

Figura 6.6: Histograma da Variável log(total_seconds).

Outra transformação que se mostrou relevante foi aplicada "`fatorCor = 4/var`", direcionada para as variáveis relacionadas a CPU, que foi criada na fase de formulação do modelo de [100]. O objetivo de aplicar essa transformação é permitir uma interpretação melhor das possíveis correlações entre as variáveis preditoras, visto que como as CPUs do conjunto avaliado variam entre 2 e 8, foi atribuído um valor intermediário, ou seja, o valor 4. Desta forma, faz com que o conjunto de CPUs varie entre 0.5 e 2, e trata-se de uma

mudança de escala que não altera as propriedades estatísticas dos modelos preditivos de regressão. Essas transformações podem ser vistas na Tabela 6.4.

Assim, concluída a fase de seleção e transformações aplicadas às variáveis preditoras presentas na base de dados de treinamento, é necessária a análise de multicolinearidade entre essas variáveis preditoras. A multicolinearidade em regressão é uma condição que ocorre quando algumas variáveis preditoras no modelo estão correlacionadas a outras variáveis preditoras. Para isso, é utilizado o Fator de Inflação de Variância (FIV) [64], descrito na Seção 6.3.2.

6.3.2 Fator de Inflação de Variância

Para auxiliar na validação do modelo preditivo foi analisada a ocorrência de multicolinearidade. Isso é importante porque refere-se à independência entre as variáveis de controle do modelo de regressão, pois quando a correlação entre as variáveis é significativa, as inferências para o modelo de regressão se mostram com baixo índice de acerto [64].

Desta forma, a existência de multicolinearidade forte em um conjunto de dados faz com que haja um aumento na variância dos coeficientes de regressão, tornando-os instáveis e com algumas consequências. Por exemplo, os coeficientes parecerem insignificantes, mesmo quando existe uma relação significativa entre a variável preditora e a resposta, ou em caso de coeficientes para preditores altamente correlacionados e que assim variam fortemente de amostra para amostra. Isso pode trazer resultados bastante imprecisos, entre outros problemas [80].

Neste sentido, alguns métodos podem ser utilizados para validação da inexistência de multicolinearidade entre variáveis preditoras em um modelo de regressão. Neste trabalho foi implementado, por meio da linguagem Python 3.7, o Fator de Inflação de Variância (FIV) [64]. Na Tabela 6.5 é possível analisar a correlação baixa e aceitável, uma vez que os respectivos valores são menores do que cinco.

Tabela 6.5: Indicação de Baixa Correlação pelo Fator de Inflação de Variância.

Variável	FIV
`program`	1,79
`ram_gb`	3,81
`cloud_name`	1,13
`input_file_lines`	3,05
`fatorCor=4/cpu`	3,69
`fatorCor=4/usr_total_cpu_usage`	2,26

As próximas seções apresentam os modelos de aprendizado de máquina desenvolvidos neste trabalho, além do conselho de preditores proposto, utilizado para predizer o

tempo e, consequentemente, o custo financeiro das execuções de *workflows* científicos de Bioinformática.

6.4 Modelos de Predição

Após os procedimentos de seleção de variáveis, além de suas transformações, consolidou-se o conjunto de dados de treinamento. Assim, a divisão entre conjunto de dados de treinamento e de teste foram feitos com tamanhos relativos a 80% para o conjunto de treinamento, e 20% para os dados de teste, sendo escolhidos de maneira aleatória entre si. Esses valores percentuais de tamanho para treinamento e teste foram definidos empiricamente conforme o princípio de Pareto, ou mais conhecido como regra 80/20 [91].

Dessa maneira, com o objetivo do completo entendimento da estrutura do modelo de conselho de preditores proposto por este trabalho, faz-se necessária a descrição de cada modelo desenvolvido. O conselho de preditores (ver Seção 6.5) está estruturado como uma *Perceptron* Multicamadas (do inglês, *Multi-layers Perceptron*) (MLP) que é treinada a partir das saídas de cada modelos selecionado para sua composição. Após a seleção das variáveis que compõem as entradas e a análise de multicolinearidade, os modelos e seus diferentes componentes são descritos brevemente a seguir.

O primeiro modelo é o de Regressão Multilinear (RML), implementado por meio do componente *LinearRegression* da biblioteca *Scikit-learn* [87]. Este modelo consiste em uma Regressão Linear de mínimos quadrados ordinários. A regressão linear ajusta-se a um modelo linear com coeficientes $w = (w_1, ..., w_n)$ para minimizar a soma residual dos quadrados entre os alvos observados no conjunto de dados, e os alvos previstos pela aproximação linear.

Em seguida, o segundo modelo desenvolvido foi o *Perceptron* Multicamadas (do inglês, *Multi-layers Perceptron*) (MLP), implementado por meio do componente *MLPRegressor* da biblioteca *Scikit-learn* [87]. Este modelo possui três camadas ocultas com 1, 3, e 5 neurônios cada, respectivamente. O treinamento deste modelo ocorreu durante 2000 épocas. É importante ressaltar que, para se chegar a essa estrutura, um procedimento de repetição foi implementado para que testes empíricos fossem efetuados com o intuito de otimizar os pesos entre diferentes quantidade de camadas e neurônios. Dessa forma, para cada execução efetuada nesse teste, as saídas para análise posterior eram salvas em diretórios para que posteriormente pudessem ser analisadas e a estrutura do modelo pudesse ser selecionada.

O terceiro modelo desenvolvido foi a Máquina de Vetores de Suporte (MVS), no inglês, *Support Vector Machine* (SVM), direcionado para problemas de regressão, também conhecido como Regressão de Vetor de Suporte (RVS), no inglês, *Support Vector Regres-*

sion (SVR). Ele foi desenvolvido também por meio do componente *SVR* da biblioteca *Scikit-learn* [87]. Este modelo não foi selecionado para o modelo de conselho de preditores por não ter demonstrado bons resultados nas métricas de avaliação, conforme pode ser visto na Tabela 6.6.

O quarto modelo desenvolvido foi a Árvore de Decisão, no inglês, *Decision Tree* (DT), por meio do componente *DecisionTreeRegressor* da biblioteca *Scikit-learn* [87]. Trata-se de um modelo que compõe a estrutura do modelo de Floresta Aleatória, o que ocasionou resultados bastante semelhantes no que tange as métricas de avaliação entre esses dois modelos. Por esse motivo, também não foi selecionado para o modelo do conselho de preditores proposto neste trabalho.

Por fim, o último modelo desenvolvido foi a Floresta Aleatória, no inglês, *Random Forest* (RF), por meio do componente *RandomForestRegressor* da biblioteca *Scikit-learn* [87]. Este modelo utiliza 10 componentes de Árvore de Decisão, e a média das saídas de cada árvore para compor os seus resultados.

Desta forma, é necessário realizar a análise gráfica dos resíduos (ver Seção 6.5), além da validação por meio das métricas de avaliação vistas na Seção 5.4.7. A análise dos resíduos é um conjunto de técnicas utilizadas para investigar a adequabilidade do modelo com base nos seus resíduos, com objetivo de verificar o erro em y não explicado pelas variáveis preditoras. Assim, quanto menor o resíduo, melhor é a modelagem de y a partir das variáveis preditoras. Dessa forma, supõe-se que os erros do modelo sejam normais e independentemente distribuídos, com desvio-padrão constante e média zero [100].

Neste sentido, a próxima seção apresenta as diferentes métricas de avaliação desenvolvidas para os modelos supracitados e para conselho de preditores (ver Seção 6.5). Isso é importante para que seja possível verificar, em conjunto com a análise gráfica dos resíduos, a adequação do modelo proposto ao problema de predição de tempo de execução de *workflows* científicos de Bioinformática em ambiente de nuvem.

6.5 Conselho de Preditores

O conselho de preditores tem como objetivo aumentar o R^2_{adj} do modelo de predição proposto neste trabalho, assim como diminuir os erros médios das observações feitas por meio das métricas de avaliação vistas na Seção 5.4.7. Assim, tem-se a intenção de obter maior acurácia na predição. Deste modo, o conselho de preditores tem como entrada o conjunto de saídas dos modelos anteriormente vistos na Seção 6.4. Logo, os resultados dos modelos anteriores e do conselho de preditores, podem ser comparados na Tabela 6.6. Essa tabela descreve os resultados das métricas de avaliação para cada

modelo apresentado neste trabalho, assim como os resultados das observações do conselho de preditores, situado na última coluna.

Assim sendo, o modelo de conselho de preditores é definido como uma Rede Neural *Perceptron* Multicamadas (do inglês, *Multi-layers Perceptron*) (MLP) desenvolvida através do componente *MLPRegressor* da biblioteca *Scikit-learn* [87], que possui três camadas ocultas com 3 neurônios cada. O seu treinamento ocorreu durante 500 épocas. Assim como no modelo de Rede Neural que compõe este conselho, para se chegar nessa estrutura neuronal, o treinamento foi efetuado com diferentes camadas e com quantidades aleatórias de neurônios, variando de 1 até 250, empiricamente. Desta forma, foi coletado o melhor modelo dentre tais execuções.

A estruturação inicial do modelo proposto pode ser visualizada na Figura 6.7. Desta forma, um modelo de Regressão Multilinear (RML), uma Rede Neural *Perceptron* Multicamadas (do inglês, *Multi-layers Perceptron*) (MLP) e uma Floresta Aleatória (FA) compõem este Conselho de Preditores (CP). Desta forma, as saídas destes modelos serviram de base para o treinamento da MLP que define o conselho. Assim, foram necessários os resultados da predição anterior dos modelos que o compõem para tornar possível a filtragem e o resultado final da predição.

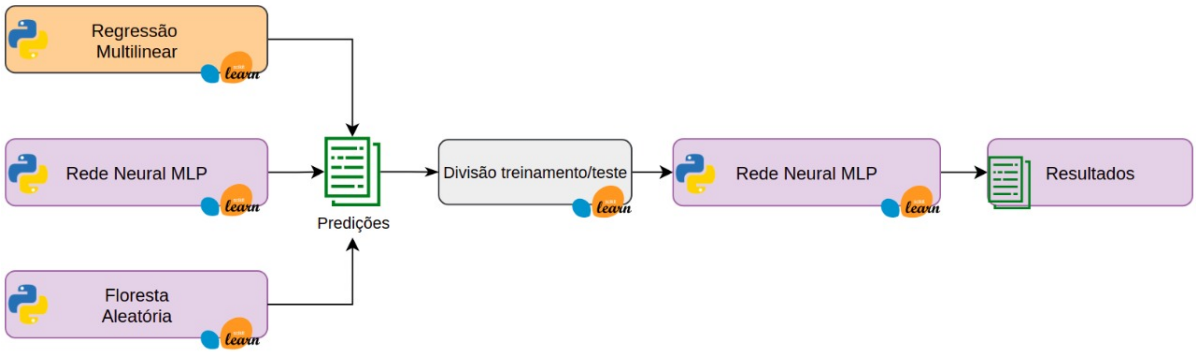

Figura 6.7: Estrutura de Treinamento do Modelo de Conselho de Preditores.

É importante ressaltar que na fase de estruturação deste conselho, cinco modelos o comporiam inicialmente, no entanto, após sua completa estruturação, tornou-se necessária a remoção do modelo Regressão de Vetor de Suporte e do modelo de Árvore de Decisão. O motivo para a retirada do modelo Regressão de Vetor de Suporte se deu pelo baixo coeficiente de determinação R^2 e R^2_{adj}. Outro motivo para sua retirada foram os resultados das métricas de avaliação (vistos na Seção 5.4.7) terem apresentado valores muito distantes de zero, conforme pode ser visto na Tabela 6.6. O segundo modelo, Árvore de Decisão, foi retirado do conselho por ter apresentado resultados nas métricas de avaliação bastante semelhantes aos do modelo de Floresta Aleatória, e, desta forma, optou-se

por diminuir a quantidade de modelos integrantes do conselho. A retirada destes dois modelos apresentou melhores coeficientes R^2 e R^2_{adj}, e menor índice de erros, conforme os resultados apresentados na Tabela 6.6.

Assim, após a descrição do modelo de conselho de preditores os resultados das métricas de avaliação, em conjunto com a análise gráfica dos resíduos, serão descritos na Seção 6.5.1.

6.5.1 Resultados do Conselho de Preditores

Nesta seção são apresentados os resultados e a análise dos dados obtidos por meio do serviço de predição proposto. A análise gráfica dos resíduos deste modelo pode ser visualizada na Figura 6.9, e o gráfico de probabilidades do modelo pode ser visto na Figura 6.8.

Assim, a Figura 6.8 mostra que a maioria das observações encontra-se sobre a linha da distribuição normal, a qual tem seus pontos no hiperplano marcados por meio da Equação 6.1. Desta forma, ocorrem indícios de que os resíduos permanecem normalmente distribuídos em torno da reta, e isso significa que os erros não possuem uma tendência forte, o que poderia inviabilizar uma melhor acurácia do modelo.

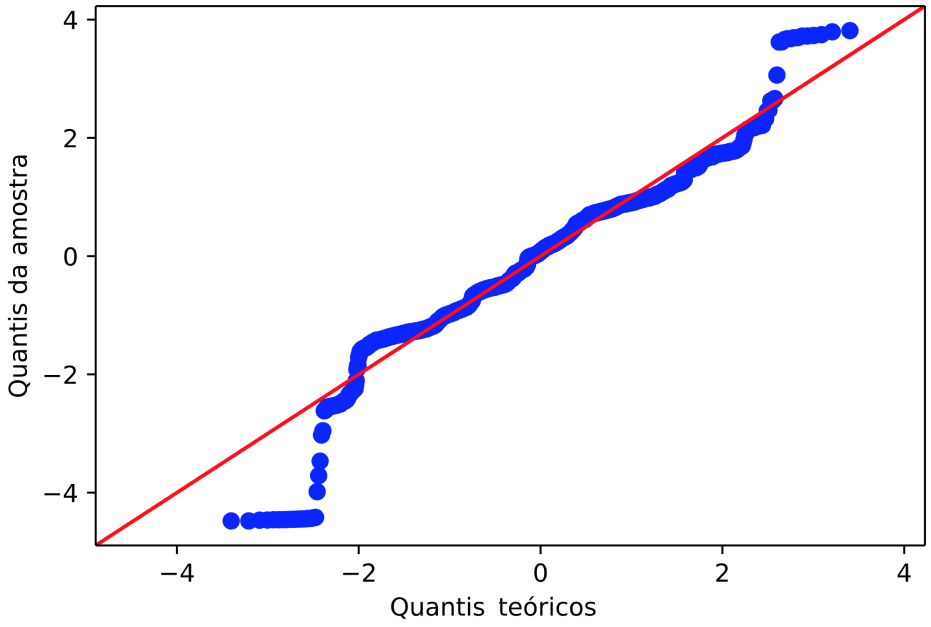

Figura 6.8: Probabilidades do Modelo de Conselho de Preditores.

$$P(y_i - \hat{y}_i) = \frac{(y_i - \hat{y}_i) - \bar{X}(y - \hat{y})}{\sigma(y - \hat{y})} \tag{6.1}$$

A Figura 6.9 apresenta em seu eixo x os valores preditos pelo modelo, e no eixo y os respectivos erros em relação ao valores realmente observados nas execuções contidas da base de dados. Essa figura mostra a distribuição do erro em relação ao conjunto de

dados testes que foi utilizado. Assim, é possível observar aleatoriedade nos resíduos, não sendo possível visualizar nenhuma espécie de tendência, tornando possível concluir que os resíduos indicam normalidade nos erros do modelo. Dessa forma, os resultados das métricas de avaliação (ver Seção 5.4.7) podem ser vistos na Tabela 6.6.

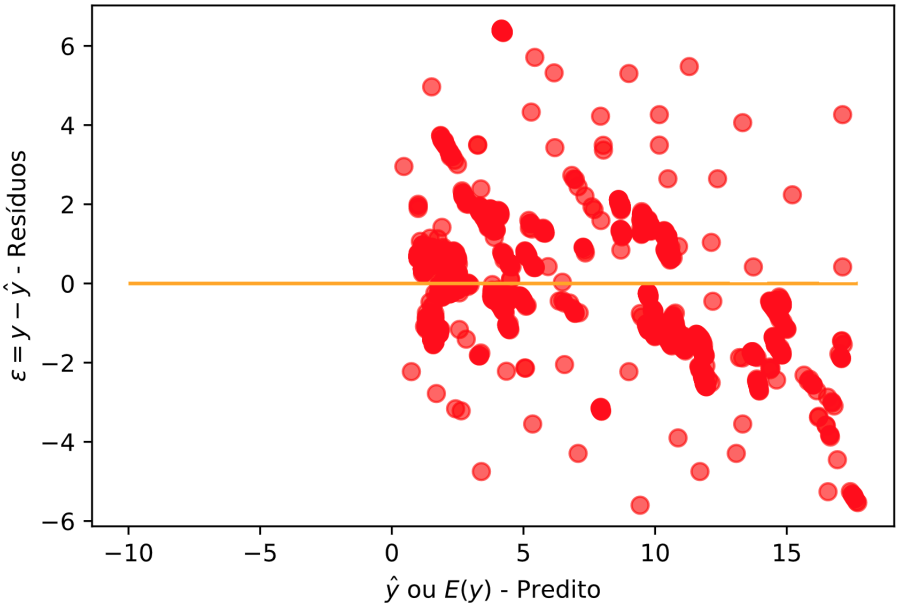

Figura 6.9: Resíduos do Modelo de Conselho de Preditores.

Tabela 6.6: Métricas de Avaliação dos Modelos de Aprendizado de Máquina.

Métrica	RML	RNA	RVS	AD	FA	CP
ME	-0.00791	-0.00601	-0.04790	0.00380	-0.00300	-0.00300
MAE	0.15171	0.01363	0.61120	0.01545	0.01486	0.00801
RMSE	0.21588	0.08600	0.65657	0.09289	0.09108	0.07361
MAPE	0.29250	0.02890	0.66245	0.07402	0.06438	0.02710
R^2	0.94122	0.98191	0.56320	0.98166	0.98172	0.98849
R^2_{adj}	0.94098	0.98182	0.56162	0.98050	0.98061	0.98704

Como pode ser notado na Tabela 6.6, o modelo de predição proposto tem um desempenho satisfatório, uma vez que os resultados da métricas ME, MAE, RMSE estão próximos de zero. A acurácia dessas métricas são dependentes do erro próximo de zero, uma vez que o mesmo é representado pela diferença entre o valor predito e o valor real. Assim sendo, a métrica EPMQ mostra que, em média, as previsões do conjunto de testes estão incorretas em apenas 2,74% dos testes efetuados. Portanto, a partir de tais informações torna-se possível concluir que o modelo de regressão baseado no conselho de preditores, com métodos de aprendizado de máquina proposto neste trabalho, pode obter

bons resultados ao ser utilizado como ferramenta para auxiliar nas estimativas de tempo de execução dos *workflows* científicos de Bioinformática.

6.6 Testes das Estimativas de Tempo

Nesta seção são avaliados os resultados obtidos nos modelos de conselho de preditores e Floresta Aleatória para estimativas de tempo, a partir dos dados do *workflow* de experimento utilizado neste trabalho. Assim sendo, as estimativas foram definidas em dois cenários:

- **Cenário 1 - Software conhecido na base de dados histórica:** fluxo no qual já existem informações de todos os programas presentes no *workflow* na base de dados histórica. Este cenário não necessita de um passo a mais para obter informações da execução parcial dos programas presentes no *workflow*;
- **Cenário 2 - Software desconhecido na base de dados histórica:** este cenário é necessário caso não existam informações prévias, na base de dados histórica, de algum programa contido no *workflow*. Desta forma, faz-se necessário instanciar um fluxo de execução parcial simulada para preencher a base de dados com informações do programa a partir de pequenas execuções, para posterior utilização no modelo de Floresta Aleatória.

É importante ressaltar que no Cenário 2 é necessário executar a predição das estimativas de tempo por meio do modelo de Regressão com Floresta Aleatória. Isso se dá pelo fato de que o treinamento dos modelos de Rede Neural possui um tempo de treinamento exorbitantemente superior quando comparado com os demais modelos descritos neste trabalho. Assim, os testes para avaliação das estimativas de tempo foram distribuídos em três máquinas virtuais distintas do provedor AWS, com as seguintes configurações:

- CPU: 2, 4, e 8;
- RAM: 7 GBytes;
- Disco: 60 GBytes;
- Sistema Operacional: Ubuntu 20.04 x64.

As execuções foram replicadas 20 vezes para cada configuração de instância, ou seja, são 60 observações para cada CPU. Os testes realizados para avaliar as estimativas de tempo, são apresentados nas próximas seções.

6.6.1 Cenário 1 - Software Conhecido

Conhecendo todos os programas presentes na base de dados é possível estimar o tempo de execução dos programas presentes no *workflow*, por meio do modelo de conselho de preditores. Nas Figuras 6.10, 6.11 e 6.12 é possível observar o resultado do modelo de conselho de preditores, por meio das estimativas de tempo dos programas TopHat2, Trinity e BlastX, respectivamente.

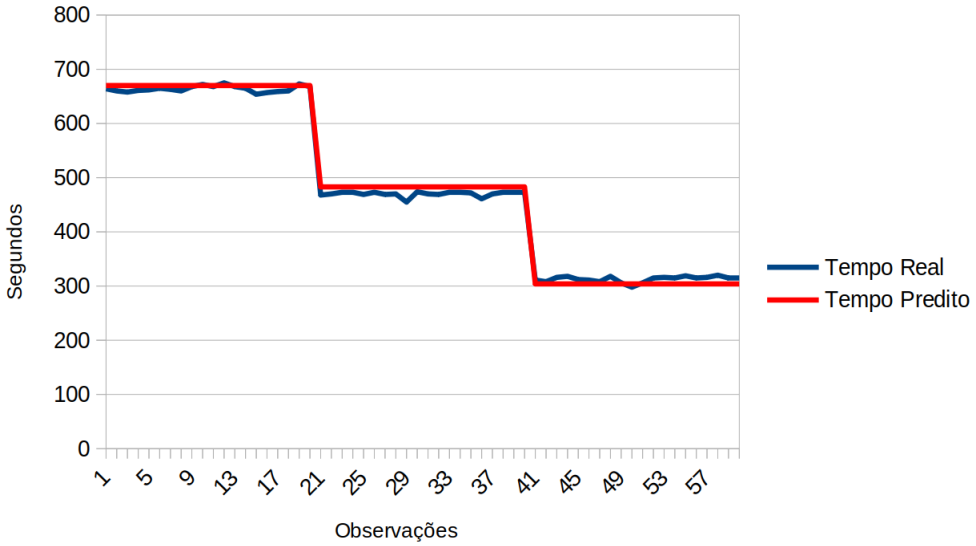

Figura 6.10: Tempo Real e Tempo Predito do Programa TopHat2.

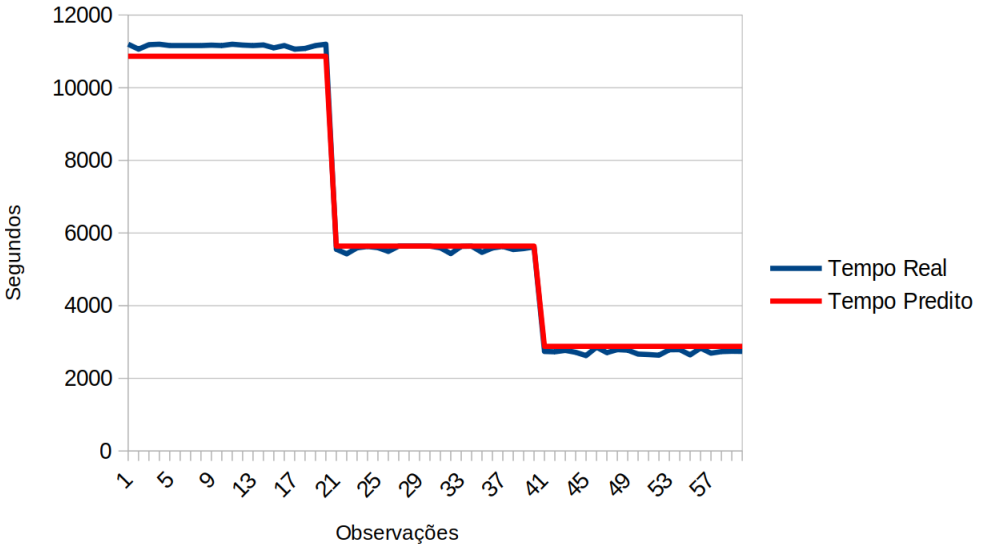

Figura 6.11: Tempo Real e Tempo Predito do Programa Trinity.

Nessas figuras o eixo x representa o tempo de execução em segundos, e o eixo y representa o número de observações de teste efetuadas neste cenário. O tempo de execução

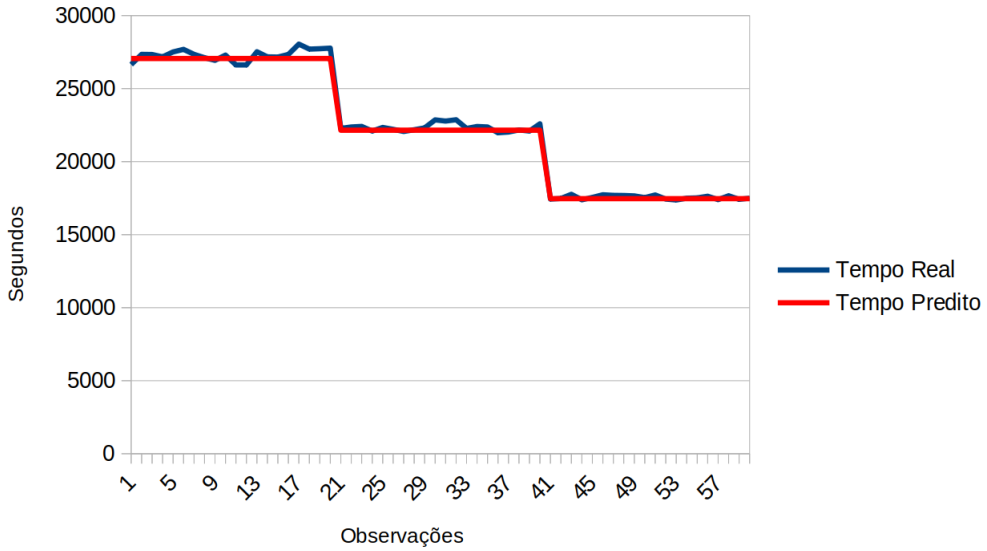

Figura 6.12: Tempo Real e Tempo Predito do Programa BlastX.

real do programa está representado em azul, enquanto o tempo estimado em vermelho. As execuções numeradas de 1 a 20 são de duas CPUs, as execuções de 21 a 40 são de quatro CPUs e, por último, as execuções de 41 a 60 são de 8 CPUs.

Nesse sentido, foi calculado uma aproximação com erro médio de 3 segundos para o TopHat2, 25 segundos para o Trinity e 170 segundos para o BlastX. É importante ressaltar que esse erro médio é calculado a partir das predições feitas para cada programa presente no *workflow* executado. Nesse sentido, alguns desses programas possuem uma estruturação não determinística, ou seja, para uma mesma entrada, é possível a obtenção de resultados diferentes, o que provoca variação no tempo de execução do programa. Sendo assim, torna-se perceptível a diferença entre o erro médio de 3 segundos do TopHat2 e de 170 do BlastX. A despeito disso, ao observar os tempos totais, tem-se que o erro é razoavelmente aceitável, e podem ser melhor considerados em conjunto com a análise dos resíduos presente na Seção 6.5.1. Um comparativo pode ser visto na Tabela 6.7.

6.6.2 Cenário 2 - Software Desconhecido

Neste cenário o modelo de conselho de preditores desconhece os valores de tempo para o programa presente no *workflow* e, por motivos de redução de tempo no que tange o treinamento do modelo com esse software não presente na base de dados histórica, optou-se direcionar a avaliação deste cenário para o modelo de Floresta Aleatória, pois este modelo possui um treinamento rápido quando comparado com o modelo de rede neural do conselho de preditores. Desta forma, será necessário retreinar o modelo de conselho de preditores em momentos específicos para que a gama de softwares conhecidos aumente.

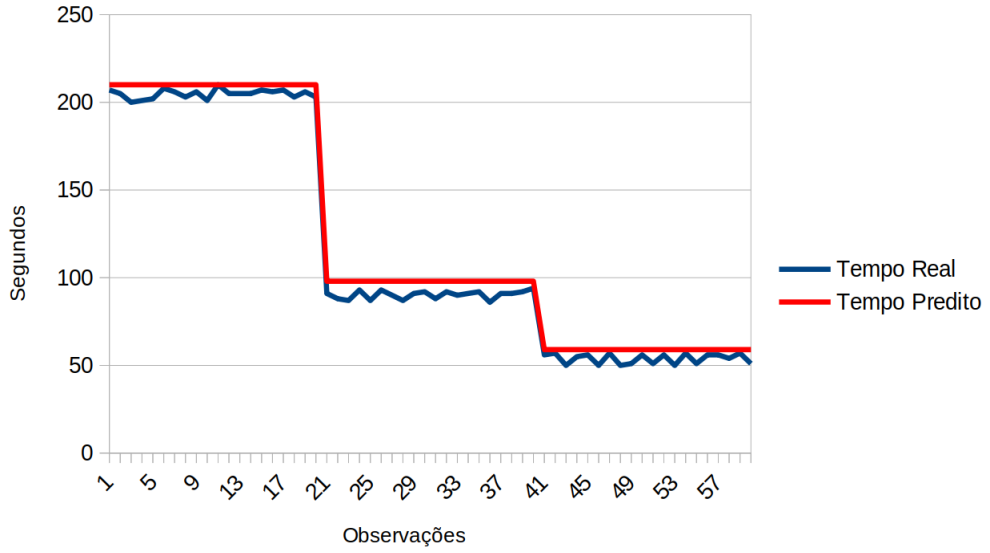

Figura 6.13: Tempo Real e Tempo Predito do Programa Bowtie2Build.

Execuções simuladas são propostas em [99], e são utilizadas como medida de contorno para quando o modelo não possui nenhum conhecimento sobre detalhes da execução do programa que se pretende predizer o tempo. Assim sendo, são realizadas execuções simuladas com uma pequena parte do arquivo de entrada. O objetivo desta fase é diminuir o erro da predição.

Desta forma, foram efetuadas 75 observações de execuções simuladas com 25.000 *reads*, assim como também 100 observações de execuções simuladas com 50.000 *reads*, do arquivo de entrada do *workflow* de experimento. São também replicadas 60 observações de execuções reais para avaliação da proposta deste cenário, conforme mostrado na Figura 6.13.

Como pode ser observado na Figura 6.13, o tempo predito implicou em erro médio de 8 segundos para o programa Bowtie2Build. Assim, foi possível reafirmar que a criação de uma base simulada é de suma importância para o modelo preditivo no contexto deste trabalho, pois as execuções simuladas expressam o comportamento do programa, mesmo com execuções de menor tempo. Esse procedimento ocasiona *overhead* de resposta, o que aumenta o tempo de execução do sPCRAM. No entanto, é possível considerar esse *overhead* irrisório se comparado com o tempo total de execução de alguns programas presentes em *workflows* de Bioinformática, sendo que, geralmente, apresentam tempo de execução em torno de horas, ou até mesmo dias, a depender do tamanho da entrada.

Na Tabela 6.7 é feito um comparativo com os resultados de [99] das execuções avaliadas do *workflow* apresentado na Seção 2.4.

Desta forma, na Tabela 6.7 é possível observar na primeira coluna os softwares testados e divididos nos dois cenários, conforme a segunda coluna. Na terceira coluna são elencados

Tabela 6.7: Erro Médio na Predição em Segundos.

Programa	Cenário	Rosa [99]	Este Trabalho
TopHat	1	68s	3s
Trinity	1	744s	25s
BlastX	1	442s	170s
Bowtie2Build	2	6s	8s

os resultados obtidos por [99] em um modelo de Regressão Linear Múltipla para a predição de tempo. Por fim, na última coluna são apresentados os erros médios em segundos obtidos neste trabalho.

Desta forma, é possível perceber que, comparado ao trabalho de [100], o modelo preditivo do conselho de preditores, na média, obteve um erro médio, contabilizado em segundos, nos testes executados, menor que o modelo apresentado no trabalho comparado, considerando os programas presentes no Cenário 1. Percebe-se também que, para o Cenário 2 apresentado, o modelo preditivo obteve um erro médio próximo do trabalho comparado.

6.7 Meta-heurística GRASP

Após estruturados os modelos de predição de tempo, torna-se necessário também o desenvolvimento de uma modelo de seleção de recursos para um ambiente de nuvens federadas. A meta-heurística1 (do inglês, *Greedy Randomized Adaptive Search Procedure*) é utilizada neste trabalho com o intuito de disponibilizar uma solução viável de recursos para execução dos *workflows* em ambiente de nuvens federadas.

A meta-heurística GRASP consiste na busca iterativa, aleatorizada, e gulosa de uma solução que seja compatível com as intenções dos usuários da ferramenta de orquestração de nuvens. Esse método foi utilizado inicialmente em [99], e apresentou resultados satisfatórios no que tange o processo de predição de recursos em um ambiente de nuvens federadas. Neste sentido, este trabalho adaptou essa solução como modelo de seleção de recursos em ambiente de nuvem federada em conjunto com o modelo de predição de tempo presente na Seção 6.4.

Para descrever a meta-heurística são necessárias notações para a alocação das máquinas virtuais presentes nos diferentes provedores de nuvem participantes da federação. Essa notação é definida em [100], e reproduzida neste trabalho para entendimento da implementação da meta-heurística GRASP. Desta forma, P é definido como o conjunto de todas as máquinas virtuais de um determinado provedor j. C_m e T_m são definidos, respectivamente, como o custo financeiro máximo e o tempo máximo de execução, e são

relativos aos desejos dos usuários. Além disso, são considerados também os requisitos relacionados ao *workflow*, como a capacidade de armazenamento em disco D, a capacidade de memória necessária M e o poder de processamento N_c.

Assim, cada tipo de máquina virtual $p \in P$ possui um custo financeiro associado c_p. O custo de alocação de uma máquina virtual, por um determinado período de tempo, pode ser tarifado entre 1 minuto ou 1h, e depende do provedor de nuvem sendo considerado. Tais provedores têm recursos de computação como armazenamento em disco d_p, capacidade de memória m_p, quantidade de núcleos de processamento n_p.

Deve também ser considerado na notação o custo do *upload* uc_p, *download* dc_p de uma máquina virtual, o custo do armazenamento dos dados transmitidos cd_p, e o limite máximo de máquinas virtuais N_h que podem ser alocadas por usuários e que também varia de acordo com o provedor de nuvem.

A notação usada neste trabalho está resumida por meio da Tabela 6.8. Essa notação dá suporte para definir a função de custo, também chamada de função objetivo, e pode ser observada por meio da Equação 6.2, adaptada de [100].

Tabela 6.8: Notação para Ambiente de Nuvem Federada [100].

Notação	Descrição
P_j	Conjunto de tipos de máquinas virtuais oferecidas pelo provedor j
C_m	Custo máximo financeiro definido pelo usuário
T_m	Tempo máximo de execução definido pelo usuário
D	Capacidade de armazenamento em disco
M	Capacidade de memória
N_c	Poder de processamento da máquina virtual
c_p	Custo da máquina virtual $_p$
d_p	Capacidade de armazenamento em disco da máquina virtual $_p$
m_p	Capacidade de memória da máquina virtual $_p$
n_p	Quantidade de núcleos da máquina virtual $_p$
N_h	Limite de máquinas virtuais alocadas por usuário
uc_p	Custo do *upload* para uma máquina virtual $_p$
dc_p	Custo do *dowload* a partir de uma máquina virtual $_p$
cd_p	Custo do armazenamento dos dados transmitidos
L_s	Lista de softwares utilizados no *workflow*
L_p	Lista ordenada de máquina virtual $_p$ por meio da Equação 6.3

A meta-heurística GRASP é composta das fases de construção e busca local. Na fase de construção uma solução é inicialmente encontrada e pode ser melhorada na fase de busca local. Para melhor entendimento da meta-heurística, faz-se necessária uma notação adicional. Assim, uma solução $s \in S$ é factível se s não viola os requisitos de usuário definidos na fase de coleta de informações (ver Seção 6.2). O tempo máximo *tp(s)* é o

tempo total de execução estimado pelo modelo preditivo para s, e c_{udd} como o custo de *upload*, *download* e de armazenamento dos dados de uma máquina virtual p.

$$
\begin{aligned}
F(s) = &\Big(\alpha_1\big(c_p + c_{udd}\big) + \alpha_2 tp(s)\Big) \\
&+ \lambda_1\Big(abs\big(tp(s) - T_m\big)\Big) \\
&+ \lambda_2\Big(abs\big(c_p - C_m\big)\Big)
\end{aligned}
\tag{6.2}
$$

A função de custo, descrita na Equação 6.2, calcula a qualidade da solução atual e é utilizada para retornar a necessidade de troca da melhor solução encontrada até o momento. Essa função tem como objetivo a minimização dos custos financeiros e do tempo total de execução de um *workflow*, relaxamento das desigualdades, e penalização da diferença em relação ao tempo máximo e o custo financeiro estipulados pelo usuário, inicialmente. Os coeficientes de penalidade associados à violação dos requisitos de tempo, λ_1, e de custo, λ_2, são utilizados como a diferença absoluta entre o coeficiente e custo ou tempo.

Algoritmo 1: GRASP [100].

Dados: $maxIter, P, C_m, T_m, L_s, \alpha_1, \alpha_2, \lambda_1, \lambda_2$
Resultado: s^*

```
1  s* ← vazio
2  F(s*) ← ∞
3  i ← 0
4  enquanto i <= maxIter faça
5  |  s ← construcao(P, C_m, T_m, L_s, α_1, α_2, λ_1, λ_2)
6  |  s ← buscaLocal(s, P, C_m, T_m, L_s, α_1, α_2, λ_1, λ_2)
7  |  se (F(s) < F(s*)) então
8  |  |  s* ← s
9  |  |  i ← 0
10 |  fim
11 |  i ← i + 1
12 fim
13 retorna s*
```

O Algoritmo 1 recebe um conjunto de máquinas virtuais P, o custo total da execução C_m, o tempo máximo de execução T_m, uma lista L_s que contém os softwares que compõem o *workflow*, os parâmetros que definem o custo, o tempo e as penalidades de cada um, sendo eles α_1, α_2, λ_1 e λ_2, respectivamente.

O parâmetro *maxIter* indica o número máximo de iterações sem melhoria da solução encontrada. A primeira tarefa de cada iteração do Algoritmo 1 é construir uma solução inicial, de forma randomizada e gulosa, por meio do Algoritmo 2.

Na fase de Construção, Algoritmo 2, é necessária uma lista ordenada L_p com todas as máquinas virtuais $p \in P$, de tal forma que os elementos estejam em ordem descendente do custo financeiro e do poder de processamento, calculado a partir da Equação 6.3 [100].

$$L_p = (\alpha_1 c_p + \alpha_2 n_p) \tag{6.3}$$

A cada iteração do laço de repetição do Algoritmo 2, é feita uma escolha de uma máquina virtual p^* entre as primeiras β máquinas virtuais a serem consideradas, e quando uma máquina é escolhida a mesma é removida da lista para que não seja novamente considerada na iteração. Dessa forma, o parâmetro β define o grau de aleatoriedade que a fase de construção terá e o processo termina somente quando houver uma solução construída. E essa solução encontrada possui o mínimo de memória necessária para a execução do *workflow* parametrizado por meio do monitoramento das execuções dos softwares presentes no mesmo.

Algoritmo 2: Construção.

Dados: $P, C_m, T_m, L_s, \alpha_1, \alpha_2, \lambda_1, \lambda_2$
Resultado: s^*

```
1  s ← vazio
2  L_p ← ordenar(P)
3  i ← 0
4  enquanto (s == vazio ou i <= size(L_p)) faça
5  |  k = escolha a máquina virtual p (índice i) randomicamente entre os
6  |  primeiros β elementos de L_p ainda não visitados
7  |  se (memória de k é suficiente) então
8  |  |  s ← k
9  |  senão
10 |  |  remova k de L_p
11 |  fim
12 |  i ← i + 1
13 fim
14 retorna s
```

O método de construção, Algoritmo 2, não garante o retorno de uma solução viável ou localmente ótima em relação à sua vizinhança. Nesse cenário, a solução pode ser melhorada através do procedimento de busca local, conforme Algoritmo 3.

O método de busca local inicia com a solução fornecida pela fase de construção e, iterativamente, substitui a solução atual quando s obtém melhoria em relação à solução

Algoritmo 3: Busca Local.

Dados: $P, C_m, T_m, L_s, \alpha_1, \alpha_2, \lambda_1, \lambda_2$
Resultado: s

```
para todo s* ∈ (N_e(s) ⋃ N_d(s)) faça
    se (F(s*) < F(s)) então
        s ← s*
    senão
        remova s* de L_p
    fim
fim
retorna s
```

sendo observada na iteração. Logo, isso ocorre quando a solução da vizinhança encontrada possui menor resultado avaliado pela função de custo (ver a Equação 6.2) em relação à solução atual.

Assim, no Algoritmo 3, a vizinhança da solução atual, encontrada na fase de construção, é retornada com um conjunto de máquinas L_p, sendo considerada k_e elementos à esquerda e k_d elementos à direita do índice k de L_p.

Na Figura 6.14 é possível observar a estrutura da meta-heurística GRASP desenvolvida no sPCRAM. O método recebe uma lista de recursos disponíveis na federação, e os detalhes do *workflow* a ser executado pelo projetista de Bioinformática. O método executa as funções de construção e busca local $maxIter$ vezes, sendo $maxIter$ o número de repetições da meta-heurística GRASP. Nesse processo, chamadas ao microsserviço de predição de tempo são feitas para que o tempo do *workflow*, para aquele recurso iterado, seja estimado. Ao fim do método GRASP, o recurso predito é retornado ao sPCRAM.

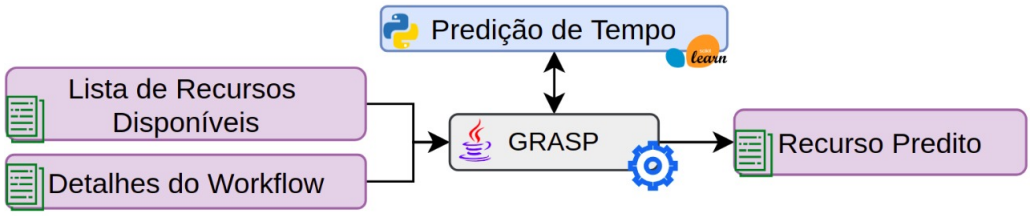

Figura 6.14: Estrutura da Meta-heurística GRASP no sPCRAM.

Na próxima seção são apresentados os resultados da junção da abordagem preditiva, vista na Seção 6.4, com a meta-heurística GRASP. O objetivo é indicar eficientemente, dentre as várias opções de uma nuvem federada, o recurso necessário para o *workflow*, além do custo e do tempo demandado.

6.8 Resultados da Meta-heurística GRASP

Nesta seção são apresentados os resultados obtidos por meio da meta-heurística GRASP (ver Seção 6.7) em ambiente de nuvens federadas. Para isso, como estudo de caso, foi desenvolvido um *plugin* para ser utilizado em conjunto da plataforma BioNimBuZ 2 (ver Seção 4).

Esta avaliação tem como principal objetivo a demonstração do comportamento da proposta deste trabalho, de custo financeiro e de tempo, por meio da interação com os parâmetros de execução do sPCRAM.

Para tais avaliações, os provedores de nuvem *Google Cloud Platform* (GCP) [39] e *Amazon Web Services* (AWS) [4] foram utilizados, pois ambos disponibilizam recursos de forma gratuita por tempo limitado, e já são implementados na arquitetura de microsserviços do BioNimbuZ 2. É importante ressaltar que foi considerado um total de 2.413 máquinas presentes na federação para os dois provedores supracitados, sendo 1.456 para AWS e 957 para GCP. Assim como também não foram consideradas as máquinas preemptivas com o intuito de eliminar possíveis interferências no monitoramento, caso elas fossem requisitadas em algum momento [100].

Para avaliar a qualidade da solução proposta neste trabalho, foram efetuados experimentos em diferentes cenários de avaliação, conforme apresentado na Tabela 6.9.

Tabela 6.9: Cenários de Avaliação.

Cenário	α_1	α_2	T_m	C_m
1	1	0	∞	∞
2	0	1	∞	∞
3	0.9	0.1	∞	5
4	0.1	0.9	10k	∞

As colunas dois e três representam pesos da função objetivo de custo e de tempo, respectivamente. A coluna quatro representa o tempo máximo de execução definido pelo usuário e, por último, a coluna cinco representa o custo máximo que o usuário deseja pagar pela execução.

Conforme pode ser visto na Tabela 6.9, os cenários podem ser lidos da seguinte forma: o primeiro cenário caracteriza como uma execução de baixo orçamento; o segundo cenário é definido como uma execução de alto desempenho e alto orçamento; o terceiro cenário considera que existe um custo financeiro máximo definido pelo usuário; e, por fim, o último cenário limita o tempo máximo de execução, definido também pelo usuário.

Os resultados experimentais são a verificação do tempo de execução predito pelo modelo preditivo, assim como da seleção de recursos feita pela meta-heurística GRASP, em

termos de qualidade da solução da proposta. Os testes foram efetuados com os programas que compõem o *workflow* de Bioinformática adotado neste trabalho (ver Seção 2.4).

Desta forma, deve ser encontrado um recurso com o tempo total de execução e custo financeiro que esteja dentro do limite estabelecido pelo usuário. Os parâmetros utilizados nesta avaliação podem ser vistos na Tabela 6.10, e representam os casos em que as variáveis alteram seus pesos na Equação 6.2, assim como seus respectivos valores. Estes valores foram definidos empiricamente após diversos experimentos.

Além disso, também foram definidas as variáveis de usuário como o custo máximo (C_m) a ser gasto, e o tempo máximo (T_m) de preferência do usuário, distribuídos nos cenários descritos na Tabela 6.9.

Tabela 6.10: Parâmetros de Execução de Avaliação da Meta-heurística GRASP.

Parâmetro	Valor
λ_1	100
λ_2	100
β	soma do número máximo de máquinas virtuais que o usuário pode alocar para cada provedor j
maxIter	100

Na Tabela 6.11 é possível observar as instâncias utilizadas nesta avaliação. Na primeira coluna está o nome da instância selecionada; na segunda coluna o número de núcleos de processamento da instância selecionada; na terceira coluna é informada a memória RAM da instância; e, por fim, a última coluna representa o valor da máquina por hora em dólar americano (USD), na data do experimento efetuado neste trabalho. O sistema operacional utilizado nessas instâncias permaneceu o Ubuntu 20.04 x64 para todos os casos. O cálculo do custo descrito neste trabalho é atrelado ao valor da máquina calculado em minutos, uma vez que é dependente da implementação feita na plataforma BioNimbuZ 2.

Tabela 6.11: Instâncias utilizadas nos Cenários de Avaliação do sPCRAM.

Instância	Provedor	CPU	RAM (GB)	Custo (USD) por Hora
t2.large	AWS	2	8	$ 0.0928
a1.xlarge	AWS	4	8	$ 0.102
a1.2xlarge	AWS	8	16	$ 0.204
n1-standard-2	GCP	2	7.5	$ 0.095
n1-standard-4	GCP	4	15	$ 0.190
n1-standard-16	GCP	16	60	$ 0.760

Também é apresentado na Tabela 6.12 o conjunto de testes realizados pelo sPCRAM. Na primeira coluna, é descrito o cenário de avaliação, na segunda coluna é descrita qual a

instância selecionada, conforme Tabela 6.11, a terceira especifica o software executado no teste. A quarta coluna representa o tempo predito pelo modelo preditivo, proposto neste trabalho, em segundos. A quinta coluna informa o tempo de execução real, também em segundos, da execução do recurso selecionado pela meta-heurística GRASP. Nessa linha, o custo total presente na sexta coluna representa o valor financeiro, o qual é calculado a partir do tempo previsto. É importante ressaltar que caso o software termine antes do tempo mínimo de tarifação, o provedor levará em conta no custo esse tempo mínimo além dos outros encargos, como por exemplo o armazenamento em disco.

Tabela 6.12: Resultados dos Cenários de Avaliação das Execuções do sPCRAM.

Cenário de Avaliação	Instância	Software	Tempo Real	Tempo Predito	Custo (USD)
1	t2.large	Bowtie2build	199s	210s	$ 0.35
2	a1.xlarge	Tophat2	470s	479s	$ 0.42
3	a1.2xlarge	Trinity	2,740s	2,761s	$ 0.904
4	a1.2xlarge	Blastx	17,677s	17,709s	$ 2.428
1	n1-standard-2	Bowtie2build	204s	215s	$ 0.42
2	n1-standard-2	Tophat2	683s	699s	$ 0.54
3	n1-standard-4	Trinity	5,545s	5,558s	$ 1.028
4	n1-standard-16	Blastx	9,732s	9,812s	$ 1.760

Na Figura 6.15 é possível observar a predição efetuada pelo modelo preditivo proposto neste trabalho, e o tempo real necessário para execução do programa nos quatro cenários de avaliação apresentados na Tabela 6.9. O eixo x representa as execuções de cada programa de determinada instância especificados no *workflow*. O eixo y apresenta o tempo, em segundos, de cada programa. A coluna em azul representa o tempo real de execução, e a coluna em vermelho o tempo estimado pelo modelo preditivo do sPCRAM. Como pode ser notado, houve uma diferença média percentual de 0.51%, totalizando 193 segundos de diferença, entre o tempo real e o predito.

Além disso, para que fosse possível analisar a qualidade da seleção de recursos proposta neste trabalho, por meio da meta-heurística GRASP, foi implementada a técnica de força bruta. Essa técnica objetiva selecionar a melhor solução dentre todas as instâncias disponíveis no ambiente da federação de nuvens. Ela utiliza a mesma função objetivo da meta-heurística GRASP, com a diferença de percorrer todas as máquinas disponíveis nesse ambiente. Dessa forma, essa técnica é capaz de obter a melhor solução para os quatro cenários apresentados. No entanto, é uma técnica que possui tempo de execução maior quando comparado com as técnicas implementadas por meio de meta-heurísticas [100].

Diante disso, são apresentadas na Tabela 6.13 um comparativo entre a técnica de força bruta e a meta-heurística GRASP. Assim, a primeira coluna equivale ao cenário testado,

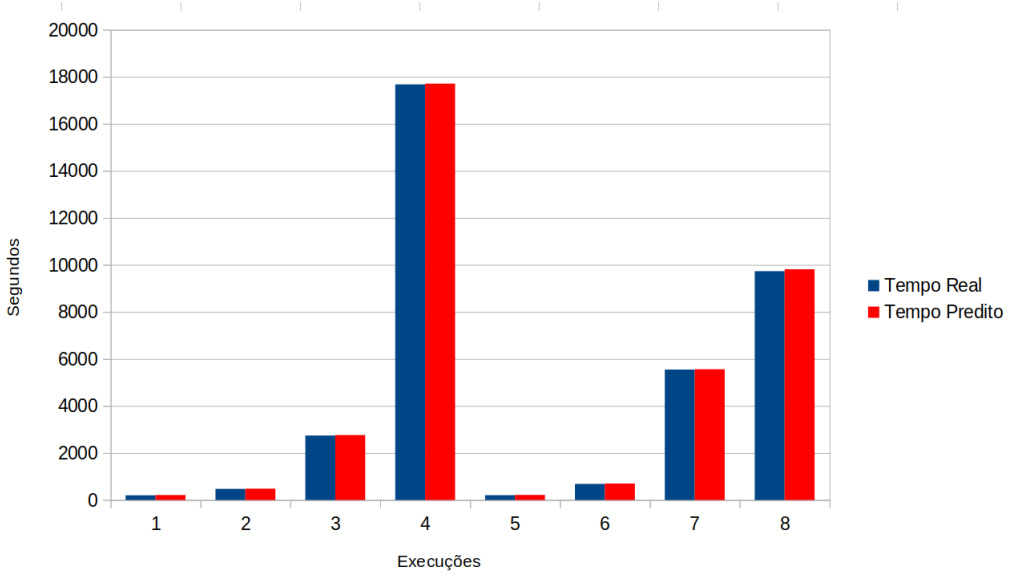

Figura 6.15: Resultados das Execuções dos Cenários de Avaliação do sPCRAM.

a segunda o método observado, a terceira o custo financeiro demandado no cenário, a quarta descreve o tempo de execução predito para o cenário observado, e, por fim, a última coluna descreve o tempo de execução da técnica de seleção de recursos, ou seja, o tempo de duração para se chegar na solução.

Tabela 6.13: Comparativo dos Resultados do GRASP *versus* Força Bruta.

Cenário de Avaliação	Método	Custo (USD)	Tempo Real	Tempo de Execução
1	GRASP	$ 0.392	30,444s	99s
1	Força Bruta	$ 0.379	30,399s	4523s
2	GRASP	$ 8.649	10,242s	102s
2	Força Bruta	$ 8.593	10,175s	4509s
3	GRASP	$ 1.590	30,134s	109s
3	Força Bruta	$ 1.418	30,021s	4511s
4	GRASP	$ 13.049	10,146s	104s
4	Força Bruta	$ 12.769	9,993s	4518s

Conforme é possível observar na Tabela 6.13, em todos os cenários a meta-heurística GRASP é, na média, 97.70% mais rápida do que a técnica de força bruta para os casos avaliados, e com diferença percentual relacionado ao custo financeiro de apenas 2.2% entre GRASP e força bruta, para os cenários de execução avaliados. Tais percentuais são calculados a partir da diferença entre o tempo real e o predito para os cenários considerados na tabela supracitada.

As Figuras 6.16 e 6.17 apresentam as estimativas de tempo e de custo financeiros para as execuções entre GRASP e força bruta, respectivamente. A Figura 6.16 tem em seu

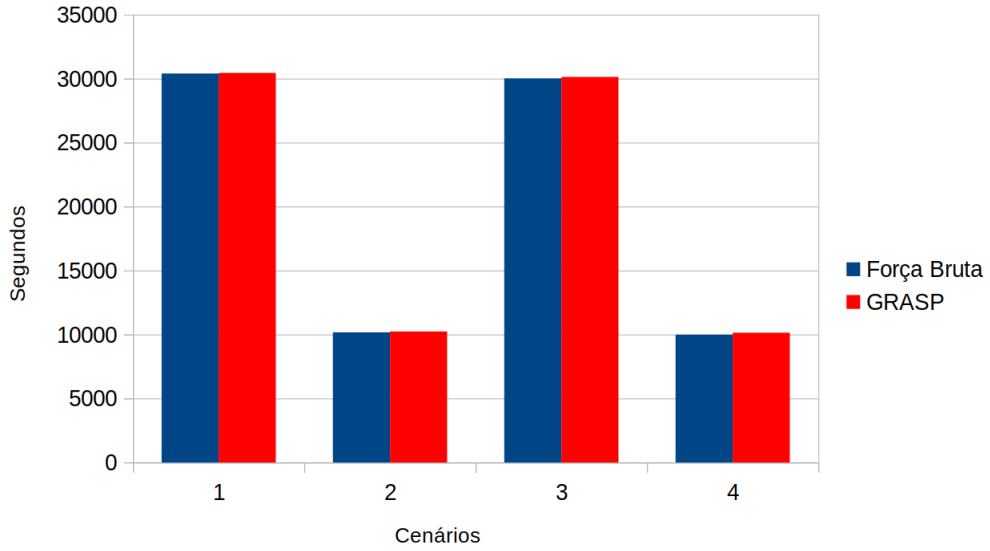

Figura 6.16: Comparativo das Estimativas de Tempo do GRASP *versus* Força Bruta.

eixo x o tempo em segundos, e no eixo y os cenários aplicados. Na Figura 6.17, o eixo x representa o custo financeiro, e no eixo y cada um dos cenários testados. Diante do exposto, é possível observar que o sPCRAM, proposto neste trabalho, foi capaz de estimar o recurso necessário em um ambiente de federação de nuvens, considerando as restrições de usuário propostas inicialmente com eficiência.

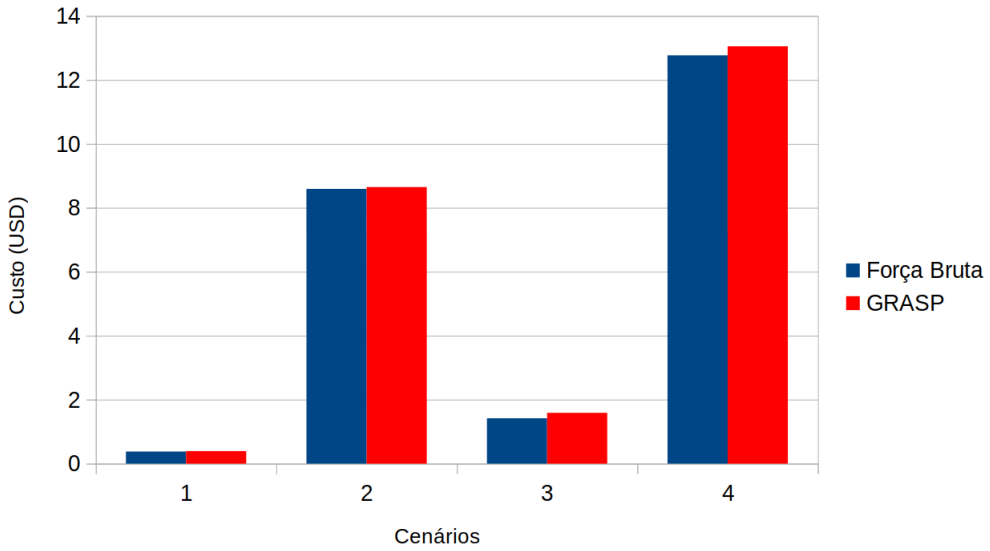

Figura 6.17: Comparativo da Estimativa de Custo do GRASP *versus* Força Bruta.

Neste sentido, é possível afirmar que o serviço proposto neste trabalho é adequado para utilização, no que tange a predição de tempo e de recursos para *workflows* científicos em ambiente de nuvens federadas, pois foi possível, a partir da lista de recursos de ambos

os provedores utilizados para os testes, estimar o tempo e os recursos adequados para os cenários apresentados.

Tabela 6.14: Trabalhos Relacionados à Predição de Recursos.

Trabalho	**Ambiente**	**Técnica**	**Predição**	**Usuário**
Baughman *et al.*[10]	Nuvem	Regressão Linear Múltipla	Recursos	Não
Hilman *et al.*[43]	Nuvem	Séries temporais, RNA e K-NN	Tempo	Não
Kaur *et al.*[52]	Nuvem	*Pool* de Preditores com AM	Uso de CPU	Não
Kumar *et al.*[57]	Nuvem	RNA	Carga de trabalho	Não
Kim *et al.*[54]	Nuvem	*Pool* de Preditores com AM	Carga de trabalho	Não
Zhong *et al.*[122]	Nuvem	SVM Adaptado	Carga de trabalho	Não
Kumar *et al.*[58]	Nuvem	RNA	Carga de trabalho	Não
Yu *et al.*[120]	Nuvem	RNA	Carga de trabalho	Não
Rosa[99]	Nuvens Federadas	GRASP e Regressão Linear Múltipla	Tempo, Custo e Recursos	Sim
Este Trabalho	**Nuvens Federadas**	**GRASP e conselho de preditores com AM**	**Tempo, Custo e Recursos**	**Sim**

A Tabela 6.14 mostra que os modelos propostos na literatura possuem, em sua grande maioria, foco em solucionar um problema presente no contexto do provedor de nuvens, como a previsão eficiente de cargas de trabalho. Em contrapartida, este trabalho se interessa em fornecer, ao usuário final, a possibilidade de escolha entre instâncias que efetuarão uma execução mais rápida ou mais barata. Sendo assim, este trabalho tem foco na realidade do usuário final, ou seja, de um projetista de *workflows* de Bioinformática. Possui estruturação por meio da integração da arquitetura de microsserviços do BioNimbuZ 2. Utiliza uma abordagem por meio do modelo de conselho de preditores com o objetivo de aumentar a acurácia das predições.

Conforme novos programas forem sendo executados, retreinamentos da estrutura poderão ser feitos para a melhoria das predições do modelo de conselho de preditores. Neste sentido, quando o programa não é conhecido, o mesmo efetua sua predição por meio do Modelo de Árvore aleatória, após diversas execuções simuladas. No entanto, para

esse caso, não está restrito a esse modelo, pois é possível parametrizar nas configurações do serviço de integração com o BioNimbuZ 2 qual modelo, dentre os disponíveis, será utilizado. Possibilita, ainda, a integração com outros serviços por meio de chamadas à sua API, sendo o único, dentre os trabalhos relacionados, que possui uma arquitetura distinta e independente, ou seja, que não necessita do serviço de origem, o BioNimbuZ 2, por exemplo, para efetuar a predição de recursos.

Este serviço compartilha de algumas características com o serviço proposto por [99], como a integração com a versão mais atual da Plataforma BioNimbuZ, a utilização no contexto de nuvens federadas, no qual fez uso das nuvens já presentes no ambiente do BioNimbuZ 2, compartilhou também do modelo de predição de recursos, a meta-heurística GRASP, com uma nova proposta da função objetivo para adequação à realidade deste trabalho, expandiu os modelos quando utiliza três diferentes modelos além do modelos estruturado como conselho desses mesmos três modelos de preditores, sendo a principal característica que o difere do trabalho de [99].

6.9 Considerações Finais

Neste capítulo foi apresentada a estrutura do sPCRAM, assim como o *workflow* utilizado para a geração de dados históricos de execuções. Além disso, foram apresentados os resultados obtidos pelo modelo preditivo proposto e os testes com a meta-heurística GRASP para a seleção de recursos em um ambiente de federação de nuvens. Assim sendo, o próximo capítulo apresenta as considerações finais e os trabalhos futuros.

Capítulo 7

Conclusão

O uso da computação em nuvem federada supre as demandas por recursos dos *workflows* científicos de Bioinformática, aumentando a quantidade de recursos disponíveis de acordo com a demanda do usuário. Todavia, a escolha adequada desses recursos não é uma tarefa simples para os usuários que trabalham com esses *workflows*.

Dessa forma, este trabalho apresentou um serviço de predição de recursos por meio de métodos de aprendizado de máquina supervisionado que possibilitou a mensuração do tempo de execução do *workflow* científico de Bioinformática. Esse modelo utilizou uma estrutura de conselho de preditores, o qual utiliza a agregação de saídas de três modelos anteriores que serve como entrada para um modelo de RNA para gerar a saída final do tempo de execução do *workflow* de entrada. Assim, foi possível subsidiar a escolha dos recursos que visem uma boa relação custo x benefício. Além disso, apresentou também uma implementação da meta-heurística GRASP para uma recomendação eficiente de seleção dos recursos presentes nesse ambiente de federação de nuvens.

Nesse sentido, o modelo preditivo proposto possibilita a predição do tempo execução dos programas conhecidos presentes nos *workflows* científicos de Bioinformática, assim como possui um fluxo de contorno quando o programa não é conhecido anteriormente pelo modelo. Sendo assim, o fluxo segue com uma criação de execuções simuladas com o objetivo de obter informações e ter o conhecimento mínimo do comportamento do programa presente no *workflow* e que até o momento não é conhecido pelo modelo de conselho de preditores.

No procedimento de seleção de recursos foi utilizada a meta-heurística GRASP. Com esta combinação, do modelo preditivo e a GRASP, é possível encontrar uma solução viável de recurso para a execução do *workflow* científico de Bioinformática. Nesse método, uma busca iterativa, aleatória e gulosa é feita com o intuito de encontrar uma solução que atenda os desejos do usuário propostos na fase de coleta de informações do serviço proposto, como, por exemplo, o tempo e custo monetário máximos que o usuário pretende

dispender para a execução deste *workflow*.

Diante do exposto, com o Serviço de Predição proposto neste trabalho, foi possível estimar recursos para um *workflow*, com estimativas que se adéquam aos cenários definidos pelo usuário. Comprovou-se que a premissa inicial de evitar o desperdício de recursos, e ainda recomendar soluções viáveis e eficientes para escolhas de recursos pelos usuários foi alcançada neste trabalho.

No que tange os trabalhos futuros, é interessante que seja possível a exploração de outras técnicas de seleção de recursos, como por exemplo os algoritmos genéticos, assim como a implementação do modelo de predição para ambientes de programação paralela, considerando o custo e o tempo de execução. Além disso, é interessante implementar um banco de dados distribuído integrado à plataforma BioNimbuZ 2 com o objetivo de aprimorar o acesso às informações da base de dados histórica.

Referências

[1] Altintas, I., Barney, O., and Jaeger-Frank, E. Provenance collection support in the kepler scientific workflow system. In *International Provenance and Annotation Workshop* (2006), Springer, pp. 118–132. 5

[2] Andrade, E., Nogueira, B., Matos, R., Callou, G., and Maciel, P. Availability modeling and analysis of a disaster-recovery-as-a-service solution. *Computing. Springer 99*, 10 (2017), 929–954. 16

[3] Armstrong, J. S., and Overton, T. S. Estimating nonresponse bias in mail surveys. *Journal of marketing research. SAGE Publications Sage CA: Los Angeles, CA 14*, 3 (1977), 396–402. 50

[4] AWS. Amazon elastic compute cloud. `https://aws.amazon.com/pt/ec2/`. Acessado online em 27 de Janeiro de 2020. 14, 87

[5] AWS. Cloudformation. `https://aws.amazon.com/pt/cloudformation/`. Acessado online em 09 de Dezembro de 2020. 23

[6] Azevedo, D. R., and Freitas Júnior, T. B. d. Biocirrus: uma nova política de armazenamento para a plataforma bionimbuz de nuvem federada. *Monografia (Licenciatura em Ciência da Computação) - Universidade de Brasília - UnB* (2015). 24

[7] Barril, J. F. H., Ruyter, J., and Tan, Q. A view on internet of things driving cloud federation. In *2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)* (2016), IEEE, pp. 221–226. 18, 21

[8] Baudat, G., and Anouar, F. Kernel-based methods and function approximation. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)* (2001), vol. 2, pp. 1244–1249. 44

[9] Bauer, M., Fahrenkrog-Petersen, S. A., Koschmider, A., Mannhardt, F., van der Aa, H., and Weidlich, M. Elpaas: Event log privacy as a service. In *BPM (PhD/Demos)* (2019), pp. 159–163. 16

[10] Baughman, M., Chard, R., Ward, L. T., Pitt, J., Chard, K., and Foster, I. T. Profiling and predicting application performance on the cloud. In *UCC* (2018), IEEE/ACM, pp. 21–30. 55, 57, 62, 92

[11] Bendel, R. B., and Afifi, A. A. Comparison of stopping rules in forward "stepwise" regression. *Journal of the American Statistical association. Taylor & Francis Group 72*, 357 (1977), 46–53. 68, 69

[12] Bharathi, S., Chervenak, A., Deelman, E., Mehta, G., Su, M.-H., and Vahi, K. Characterization of scientific workflows. In *2008 third workshop on workflows in support of large-scale science* (2008), IEEE, pp. 1–10. xii, 7

[13] Brikman, Y. *Terraform: Up & Running: Writing Infrastructure as Code*. 2019. 23

[14] Buyya, R., Ranjan, R., and Calheiros, R. N. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In *International Conference on Algorithms and Architectures for Parallel Processing* (2010), Springer, pp. 13–31. 18, 19

[15] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems. Elsevier 25*, 6 (2009), 599–616. 1, 12

[16] Caballer, M., Segrelles, D., Moltó, G., and Blanquer, I. A platform to deploy customized scientific virtual infrastructures on the cloud. *Concurrency and Computation: Practice and Experience. Wiley Online Library 27*, 16 (2015), 4318–4329. 23

[17] Carrasco, J., Durán, F., and Pimentel, E. Trans-cloud: Camp/tosca-based bidimensional cross-cloud. *Computer Standards & Interfaces. Elsevier 58* (2018), 167–179. 23

[18] Celesti, A., Mulfari, D., Fazio, M., Villari, M., and Puliafito, A. Improving desktop as a service in openstack. In *2016 IEEE Symposium on Computers and Communication (ISCC)* (2016), IEEE, pp. 281–288. 16

[19] Celesti, A., Tusa, F., Villari, M., and Puliafito, A. How to enhance cloud architectures to enable cross-federation. In *2010 IEEE 3rd international conference on cloud computing* (2010), IEEE, pp. 337–345. xii, 18, 19

[20] Chaisiri, S., Lee, B.-S., and Niyato, D. Optimization of resource provisioning cost in cloud computing. *IEEE transactions on services Computing. IEEE 5*, 2 (2011), 164–177. 2, 8, 61

[21] CLOUD FOUNDRY. Open source cloud application platform. `https://www.cloudfoundry.org/`. Acessado online em 09 de Dezembro de 2020. 15

[22] Costa, H. H. d. P. M. Controle de acesso na plataforma de nuvem federada bionimbuz. *Monografia (Bacharelado em Ciência da Computação) - Universidade de Brasília - UnB* (2015). 24

[23] Coutinho, R. d. C., Drummond, L. M., and Frota, Y. Optimization of a cloud resource management problem from a consumer perspective. In *European Conference on Parallel Processing* (2013), Springer, pp. 218–227. 62, 64, 65

[24] Coutinho, R. d. C., Drummond, L. M., Frota, Y., and de Oliveira, D. Optimizing virtual machine allocation for parallel scientific workflows in federated clouds. *Future Generation Computer Systems. Elsevier 46* (2015), 51–68. 61, 62

[25] Cristianini, N., Shawe-Taylor, J., et al. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000. 42, 43

[26] Curino, C., Jones, E. P., Popa, R. A., Malviya, N., Wu, E., Madden, S., Balakrishnan, H., and Zeldovich, N. Relational cloud: A database-as-a-service for the cloud. 16

[27] de Carvalho, L. R. Framework node2faas: Uma abordagem eficiente para conversão automática de aplicações nodejs para *Function as a Service*. *Dissertação (Mestrado em Informática) - Universidade de Brasília - UnB* (2020). 16

[28] Deelman, E., Gannon, D., Shields, M., and Taylor, I. Workflows and e-science: An overview of workflow system features and capabilities. *Future generation computer systems. Elsevier 25*, 5 (2009), 528–540. 1

[29] Deelman, E., Peterka, T., Altintas, I., Carothers, C. D., van Dam, K. K., Moreland, K., Parashar, M., Ramakrishnan, L., Taufer, M., and Vetter, J. The future of scientific workflows. *The International Journal of High Performance Computing Applications. SAGE Publications Sage UK: London, England 32*, 1 (2018), 159–175. 5

[30] Duan, Y., Fu, G., Zhou, N., Sun, X., Narendra, N. C., and Hu, B. Everything as a service (xaas) on the cloud: origins, current and future trends. In *2015 IEEE 8th International Conference on Cloud Computing. IEEE* (2015), IEEE, pp. 621–628. 16

[31] Elmasri, R., Navathe, S. B., et al. *Sistemas de banco de dados*, vol. 6. 2005. 67

[32] Emeakaroha, V. C., Łabaj, P. P., Maurer, M., Brandic, I., and Kreil, D. P. Optimizing bioinformatics workflows for data analysis using cloud management techniques. In *Proceedings of the 6th workshop on Workflows in support of large-scale science* (2011), pp. 37–46. 10

[33] Faria, H. M. d. A backup-as-a-service (baas) software solution. *Dissertação (Mestrado Profissional em Computação Aplicada) - Universidade de Brasília - UnB* (2018). 16

[34] for Biotechnology Information (US), N. C., and Camacho, C. *BLAST (r) Command Line Applications User Manual*. 2008. 9

[35] Foster, I., Zhao, Y., Raicu, I., and Lu, S. Cloud computing and grid computing 360-degree compared. In *2008 grid computing environments workshop* (2008), Ieee, pp. 1–10. 11

[36] Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS 28*, 13 (2009), 2009. 11

[37] Goiri, I., Guitart, J., and Torres, J. Characterizing cloud federation for enhancing providers' profit. In *2010 IEEE 3rd International Conference on Cloud Computing* (2010), IEEE, pp. 123–130. 18, 21

[38] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. xii, 53

[39] Google. Google cloud platform. `https://cloud.google.com/`. Acessado online em 27 de Janeiro de 2020. 14, 15, 87

[40] Grabherr, M. G., Haas, B. J., Yassour, M., Levin, J. Z., Thompson, D. A., Amit, I., Adiconis, X., Fan, L., Raychowdhury, R., Zeng, Q., et al. Trinity: reconstructing a full-length transcriptome without a genome from rna-seq data. *Nature biotechnology. NIH Public Access 29*, 7 (2011), 644. 9

[41] Gravesteijn, B. Y., Nieboer, D., Ercole, A., Lingsma, H. F., Nelson, D., Van Calster, B., Steyerberg, E. W., Åkerlund, C., Amrein, K., Andelic, N., et al. Machine learning algorithms performed no better than regression models for prognostication in traumatic brain injury. *Journal of clinical epidemiology. Elsevier* (2020). 38

[42] Haykin, S. *Redes neurais: princípios e prática*. Bookman Editora, 2007. xii, 48, 49, 50

[43] Hilman, M. H., Rodriguez, M. A., and Buyya, R. Task runtime prediction in scientific workflows using an online incremental learning approach. In *2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC)* (2018), IEEE, pp. 93–102. 55, 57, 62, 92

[44] Ibrahim, I. A., and Khatib, T. A novel hybrid model for hourly global solar radiation prediction using random forests technique and firefly algorithm. *Energy Conversion and Management. Elsevier 138* (2017), 413–425. xii, 48

[45] Islam, S., Keung, J., Lee, K., and Liu, A. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems. Elsevier 28*, 1 (2012), 155–162. 61

[46] Jin, C., De-Lin, L., and Fen-Xiang, M. An improved id3 decision tree algorithm. In *2009 4th International Conference on Computer Science & Education* (2009), IEEE, pp. 127–130. 47

[47] Jones, M., Bradley, J., and Sakimura, N. Json web token (jwt). *Tech. Rep.* (2015). 30

[48] Júnior, B., and de Oliveira, W. Escalonador de tarefas para o plataforma de nuvens federadas bionimbuz usando beam search iterativo multiobjetivo. *Monografia (Bacharelado em Engenharia da Computação) - Universidade de Brasília - UnB* (2016). 24

[49] Junqueira, F., and Reed, B. *ZooKeeper: distributed process coordination.* O'Reilly Media, Inc., 2013. 28, 29

[50] JWT. Rfc 7519. `https://jwt.io/`. Acessado online em 12 de Novembro de 2020. 27

[51] Kandel, E., Schwartz, J., Jessell, T., Siegelbaum, S., and Hudspeth, A. *Princípios de neurociências-5.* AMGH Editora, 2014. xii, 49

[52] Kaur, G., and Bala, A. An efficient resource prediction–based scheduling technique for scientific applications in cloud environment. *Concurrent Engineering. SAGE Publications Sage UK: London, England 27*, 2 (2019), 112–125. 55, 57, 62, 92

[53] Kianpisheh, S., and Charkari, N. M. A grid workflow quality-of-service estimation based on resource availability prediction. *The Journal of Supercomputing. Springer 67*, 2 (2014), 496–527. 61

[54] Kim, I. K., Wang, W., Qi, Y., and Humphrey, M. Cloudinsight: Utilizing a council of experts to predict future cloud application workloads. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)* (2018), IEEE, pp. 41–48. 20, 55, 57, 58, 62, 92

[55] Kim, E. Everything you wanted to know about the kernel trick (but were too afraid to ask). `http://www.erickim.net/eric-kim-net/posts/1/kernel_trick.html`. Acessado online em 25 de Janeiro de 2020. xii, 44, 45

[56] Kovács, J., and Kacsuk, P. Occopus: a multi-cloud orchestrator to deploy and manage complex scientific infrastructures. *Journal of Grid Computing. Springer 16*, 1 (2018), 19–37. 23

[57] Kumar, J., Saxena, D., Singh, A. K., and Mohan, A. Biphase adaptive learning-based neural network model for cloud datacenter workload forecasting. *Soft Computing. Springer* (2020), 1–18. 20, 55, 57, 62, 92

[58] Kumar, K. D., and Umamaheswari, E. Ewptnn: An efficient workload prediction model in cloud computing using two-stage neural networks. *Procedia Computer Science. Elsevier 165* (2019), 151–157. 56, 57, 62, 92

[59] Lama, P., and Zhou, X. Aroma: Automated resource allocation and configuration of mapreduce environment in the cloud. In *Proceedings of the 9th international conference on Autonomic computing* (2012), pp. 63–72. 44

[60] Langmead, B., Trapnell, C., Pop, M., and Salzberg, S. L. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome biology. BioMed Central 10*, 3 (2009), R25. 9

[61] Latgé, J.-P. Aspergillus fumigatus and aspergillosis. *Clinical microbiology reviews. Am Soc Microbiol 12*, 2 (1999), 310–350. xii, 8, 9

[62] Le, D.-H., Truong, H.-L., Copil, G., Nastic, S., and Dustdar, S. Salsa: a framework for dynamic configuration of cloud services. In *2014 IEEE 6th International Conference on Cloud Computing Technology and Science. IEEE* (2014), IEEE, pp. 146–153. 23

[63] Lightbend. Playframework. `https://www.playframework.com/`. Acessado online em 12 de Novembro de 2020. 27

[64] Lin, F.-J. Solving multicollinearity in the process of fitting regression model using the nested estimate procedure. *Quality & Quantity. Springer 42*, 3 (2008), 417–426. 72

[65] Liu, C., Liu, C., Shang, Y., Chen, S., Cheng, B., and Chen, J. An adaptive prediction approach based on workload pattern discrimination in the cloud. *Journal of Network and Computer Applications. Elsevier 80* (2017), 35–44. 20

[66] Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., and Leaf, D. Nist cloud computing reference architecture. *NIST special publication 500*, 2011 (2011), 1–28. xii, 12, 13, 14, 16, 17

[67] Mallat, S. G. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE transactions on pattern analysis and machine intelligence 11*, 7 (1989), 674–693. 56

[68] Manwani, N., and Sastry, P. Geometric decision tree. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics). IEEE 42*, 1 (2011), 181–192. xii, 46, 47

[69] Marill, K. A. Advanced statistics: linear regression, part ii: multiple linear regression. *Academic emergency medicine. Wiley Online Library 11*, 1 (2004), 94–102. xii, 41, 42

[70] Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., and Ghalsasi, A. Cloud computing - the business perspective. *Decision support systems. Elsevier 51*, 1 (2011), 176–189. 12

[71] Masse, M. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. "O'Reilly Media, Inc.", 2011. 29

[72] Mattoso, M., Dias, J., Costa, F., de Oliveira, D., and Ogasawara, E. Experiences in using provenance to optimize the parallel execution of scientific workflows steered by users. In *Workshop of Provenance Analytics* (2014), vol. 1. 21

[73] McCulloch, W. S., and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics. Springer 5*, 4 (1943), 115–133. 36, 50

[74] Mell, P., Grance, T., et al. The nist definition of cloud computing. 12

[75] Mendes, F. L. d. S. Bionimbuz 2: uma plataforma de federação de nuvens em uma arquitetura orientada a microsserviços. xii, 2, 6, 18, 21, 22, 23, 24, 25, 26, 27, 29, 30, 32, 33

[76] Meng, S., and Liu, L. Enhanced monitoring-as-a-service for effective cloud management. *IEEE Transactions on Computers. IEEE 62*, 9 (2012), 1705–1720. 17

[77] MICROSOFT. Azure. `https://azure.microsoft.com`. Acessado online em 09 de Dezembro de 2020. 15

[78] Mitchell, T. M. *Machine Learning*, vol. 47. McGraw-Hill, New York, 1997. 35

[79] Monard, M. C., and Baranauskas, J. A. Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações. Manole Ltda 1*, 1 (2003), 32. xii, 37

[80] Montgomery, D. C., Peck, E. A., and Vining, G. G. *Introduction to linear regression analysis*, vol. 821. John Wiley & Sons, 2012. 35, 40, 41, 72

[81] Moura, B. R. d. Arquitetura de um controlador de sla para ambiente de nuvens federadas. 5, 6, 14, 24

[82] Moura, B. R. d. Arquitetura de um controlador de sla para ambiente de nuvens federadas. 11

[83] Norvig, P., and Russell, S. *Inteligência Artificial: Uma Abordagem Moderna*, vol. 1, Tradução da 3a Edição. Elsevier Brasil, 2014. xii, 3, 36, 37, 38, 39, 40, 41, 42, 43, 45, 50, 53

[84] Ostermann, F., and Cavalcanti, C. d. H. Teorias de aprendizagem. *Evangraf. Porto Alegre* (2011). 39

[85] Pandey, S., Wu, L., Guru, S. M., and Buyya, R. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *2010 24th IEEE international conference on advanced information networking and applications* (2010), IEEE, pp. 400–407. 62

[86] Paula, R. d. Proveniência de dados em workflows de bioinformática. 8

[87] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research. JMLR. org 12* (2011), 2825–2830. 50, 70, 73, 74, 75

[88] Pham, L. M., Tchana, A., Donsez, D., De Palma, N., Zurczak, V., and Gibello, P.-Y. Roboconf: a hybrid cloud orchestrator to deploy complex applications. In *2015 IEEE 8th International Conference on Cloud Computing. IEEE* (2015), IEEE, pp. 365–372. 23

[89] Pivotal Software. Spring boot. `https://spring.io/projects/spring-boot`. Acessado online em 13 de Novembro de 20200. 28

[90] Polat, K., and Güneş, S. A novel hybrid intelligent method based on c4. 5 decision tree classifier and one-against-all approach for multi-class classification problems. *Expert Systems with Applications. Elsevier 36*, 2 (2009), 1587–1592. 47

[91] Pressman, R. S. *Software engineering: a practitioner's approach*. Palgrave macmillan, 2005. 73

[92] Ramos, V. d. A. Um sistema gerenciador de workflows científicos para a plataforma de nuvens federadas bionimbuz. *Monografia (Bacharelado em Engenharia da Computação) - Universidade de Brasília - UnB* (2016). 24

[93] Rao, C., Liu, M., Goh, M., and Wen, J. 2-stage modified random forest model for credit risk assessment of p2p network lending to "three rurals" borrowers. *Applied Soft Computing. Elsevier 95* (2020), 106570. 47

[94] Resende, M. G., and Ribeiro, C. C. Greedy randomized adaptive search procedures: advances and extensions. In *Handbook of metaheuristics*. Springer, 2019, pp. 169–220. 57, 65

[95] RFC 4180. Common format and mime type for comma-separated values (csv) files. `https://tools.ietf.org/html/rfc4180`. Acessado online em 22 de Janeiro de 2020. 67

[96] Rimal, B. P., Choi, E., and Lumb, I. A taxonomy and survey of cloud computing systems. In *2009 Fifth International Joint Conference on INC, IMS and IDC* (2009), Ieee, pp. 44–51. 17

[97] Rochwerger, B., Breitgand, D., Levy, E., Galis, A., Nagin, K., Llorente, I. M., Montero, R., Wolfsthal, Y., Elmroth, E., Caceres, J., et al. The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development. IBM 53*, 4 (2009), 4–1. 1

[98] Rosa, M., Moura, B., Vergara, G., Santos, L., Ribeiro, E., Holanda, M., Walter, M. E., and Araújo, A. Bionimbuz: A federated cloud platform for bioinformatics applications. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (2016), IEEE, pp. 548–555. 1, 24

[99] Rosa, M. J., Ralha, C. G., Holanda, M., and Araujo, A. P. Computational resource and cost prediction service for scientific workflows in federated clouds. *Future Generation Computer Systems. Elsevier* (2021). 8, 57, 58, 62, 63, 65, 81, 82, 92, 93

[100] Rosa, M. J. F. Predição de tempo e dimensionamento de recursos para workflows científicos em nuvens federadas. *Dissertação (Mestrado em Informática) - Universidade de Brasília - UnB* (2017). xiii, xiv, 1, 2, 3, 10, 20, 24, 29, 42, 44, 56, 57, 61, 62, 63, 69, 70, 71, 74, 82, 83, 84, 85, 87, 89

[101] Roy, S., Pattnaik, P. K., and Mall, R. A cognitive approach for evaluating the usability of storage as a service in cloud computing environment. *International Journal of Electrical & Computer Engineering (2088-8708) 6*, 2 (2016). 16

[102] Rutkowski, L., Jaworski, M., Pietruczuk, L., and Duda, P. The cart decision tree for mining data streams. *Information Sciences. Elsevier 266* (2014), 1–15. 47

[103] Saldanha, H. V. Bionimbus: uma arquitetura de federação de nuvens computacionais híbrida para a execução de workflows de bioinformática. *Dissertação (Mestrado em Informática) - Universidade de Brasília - UnB* (2012). 7, 10, 23, 24, 57

[104] Sapankevych, N. I., and Sankar, R. Time series prediction using support vector machines: a survey. *IEEE Computational Intelligence Magazine. IEEE 4*, 2 (2009). 3

[105] Schmieder, R., and Edwards, R. Quality control and preprocessing of metagenomic datasets. *Bioinformatics. Oxford University Press 27*, 6 (2011), 863–864. 9

[106] Scikit-learn: Machine learning in Python. Metrics and scoring: quantifying the quality of predictions. `https://scikit-learn.org/stable/modules/model_evaluation.html`. Acessado online em 28 de Janeiro de 2020. 50, 51, 52

[107] Setubal, J. C., Meidanis, J., and Setubal-Meidanis, . *Introduction to computational molecular biology.* No. 04; QH506, S4. PWS Pub. Boston, 1997. 7

[108] Siqueira-Batista, R., and da Silva, E. Notas sobre os fundamentos matemáticos da inteligência artificial. *Revista de Ciência, Tecnologia e Inovação 4*, 6 (2020). 38

[109] Software Freedom Conservancy, Inc. Git: Documentation. `https://git-scm.com/docs`. Acessado online em 22 de Janeiro de 2020. 67

[110] Springenberg, J. T. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390* (2015). 39

[111] Sutton, R. S., and Barto, A. G. *Reinforcement learning: An introduction.* 2018. 39

[112] Toosi, A. N., Calheiros, R. N., and Buyya, R. Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Computing Surveys (CSUR) 47*, 1 (2014), 1–47. xii, 16, 18, 21

[113] Trapnell, C., Pachter, L., and Salzberg, S. L. Tophat: discovering splice junctions with rna-seq. *Bioinformatics. Oxford University Press 25*, 9 (2009), 1105–1111. 9

[114] Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. A break in the clouds: towards a cloud definition, 2008. xii, 14, 15

[115] Vergara, G. F. Arquitetura de um controlador de elasticidade para nuvens federadas. 24

[116] VMWARE. Introducing vmware cloud assembly, vmware code stream and vmware service broker. `https://blogs.vmware.com/management/2018/08/introducing-cloud-automation.html`. Acessado online em 09 de Dezembro de 2020. 23

[117] Vu, Q. H., Pham, T.-V., Truong, H.-L., Dustdar, S., and Asal, R. Demods: A description model for data-as-a-service. In *2012 IEEE 26th International Conference on Advanced Information Networking and Applications* (2012), IEEE, pp. 605–612. 16

[118] Wang, X., Liu, Z., Qi, Y., and Li, J. Livecloud: A lucid orchestrator for cloud datacenters. In *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings. IEEE* (2012), IEEE, pp. 341–348. 23

[119] Wieers, D. Dstat: Versatile resource statistics tool. `http://dag.wiee.rs/home-made/dstat/`. Acessado online em 19 de janeiro de 2020. 66

[120] Yu, Y., Jindal, V., Bastani, F., Li, F., and Yen, I.-L. Improving the smartness of cloud management via machine learning based workload prediction. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)* (2018), vol. 2, IEEE, pp. 38–44. 56, 57, 62, 92

[121] Zawoad, S., Dutta, A. K., and Hasan, R. Seclaas: secure logging-as-a-service for cloud forensics. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security* (2013), pp. 219–230. 16

[122] Zhong, W., Zhuang, Y., Sun, J., and Gu, J. A load prediction model for cloud computing using pso-based weighted wavelet support vector machine. *Applied Intelligence. Springer 48*, 11 (2018), 4072–4083. 20, 56, 57, 62, 92