UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GABRIEL PAIVA MEDEIROS

AVALIAÇÃO DE MODELOS DE AGRUPAMENTO PARA DETECÇÃO DE COMPORTAMENTO DE APLICAÇÕES EM TERMOS DE DEMANDA E USO DE RECURSOS

CAMPINA GRANDE - PB

2023

GABRIEL PAIVA MEDEIROS

AVALIAÇÃO DE MODELOS DE AGRUPAMENTO PARA DETECÇÃO DE COMPORTAMENTO DE APLICAÇÕES EM TERMOS DE DEMANDA E USO DE RECURSOS

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : Professor Dr. Fábio Jorge Almeida Morais

CAMPINA GRANDE - PB

2023

GABRIEL PAIVA MEDEIROS

AVALIAÇÃO DE MODELOS DE AGRUPAMENTO PARA DETECÇÃO DE COMPORTAMENTO DE APLICAÇÕES EM TERMOS DE DEMANDA E USO DE RECURSOS

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Fábio Jorge Almeida Morais

Orientador – UASC/CEEI/UFCG

Professor Dr. Cláudio de Souza Baptista

Examinador – UASC/CEEI/UFCG

Professor Dr. Francisco Vilar Brasileiro

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 28 de JUNHO de 2023.

CAMPINA GRANDE - PB

RESUMO

Vem se tornando cada vez mais comum a utilização de técnicas de observação de aplicações. Observar uma aplicação gera dados importantes sobre o seu funcionamento e da infraestrutura onde ela está inserida. Analisar o comportamento de aplicações é um elemento chave que permite entender e provisionar recursos computacionais, otimizando o uso da infraestrutura em que se executam as aplicações. Embora haja o reconhecimento comportamental das aplicações em relação ao uso de recursos computacionais a partir de decisões humanas, a detecção de comportamentos de alto e baixo consumo de memória, por exemplo, através de modelos preditivos ainda não é muito comum, o que abre oportunidades de estudos nesta área. O presente trabalho se propõe a detectar comportamentos de uma aplicação a partir de diferentes algoritmos de agrupamento. Os resultados mostram que é possível detectar cada comportamento para facilitar a compreensão e alocação eficiente de recursos de computação.

Evaluation of Clustering Models for Detecting Application Behavior in Terms of Resource Demand and Use

ABSTRACT

The use of application observation techniques has become increasingly common. Observing an application generates important data about its functioning and the infrastructure where it is inserted. Analyzing the behavior of applications is a key element that allows understanding and provisioning computational resources, optimizing the use of the infrastructure on which the applications run. Although there is behavioral recognition of applications in relation to the use of computational resources based on human decisions, the detection of high and low memory consumption behaviors, for example, through predictive models is still not very common, which opens opportunities for studies in this area. The present work proposes to detect application behaviors from different clustering algorithms. The results show that it is possible to detect each behavior to facilitate understanding and efficient allocation of computing resources.

Avaliação de Modelos de Agrupamento Para Detecção de Comportamento de Aplicações em Termos de Demanda e Uso de Recursos

Gabriel Paiva Medeiros
Universidade Federal de Campina Grande
Campina Grande, Paraíba
gabriel.medeiros@ccc.ufcg.edu.br

Fabio Jorge Almeida Morais
Universidade Federal de Campina Grande
Campina Grande, Paraíba
fabio@computacao.ufcg.edu.br

RESUMO

Vem se tornando cada vez mais comum a utilização de técnicas de observação de aplicações. Observar uma aplicação gera dados importantes sobre o seu funcionamento e da infraestrutura onde ela está inserida. Analisar o comportamento de aplicações é um elemento chave que permite entender e provisionar recursos computacionais, otimizando o uso da infraestrutura em que se executam as aplicações. Embora haja o reconhecimento comportamental das aplicações em relação ao uso de recursos computacionais a partir de decisões humanas, a detecção de comportamentos de alto e baixo consumo de memória, por exemplo, através de modelos preditivos ainda não é muito comum, o que abre oportunidades de estudos nesta área. O presente trabalho se propõe a detectar comportamentos de uma aplicação a partir de diferentes algoritmos de agrupamento. Os resultados mostram que é possível detectar cada comportamento para facilitar a compreensão e alocação eficiente de recursos de computação.

Palavras-Chave

Agrupamento, séries temporais, padrões de comportamento, otimização de infraestrutura.

1. INTRODUÇÃO

No campo da tecnologia, a detecção do comportamento das aplicações em termos de demanda e uso de recursos é um desafio crucial. Com o crescente papel das aplicações computacionais em diversos setores, compreender seus padrões de comportamento tornou-se essencial para otimizar recursos, melhorar a eficiência e reduzir custos. Em Ciência da Computação, isso é conhecido como observação de aplicações, que envolve a capacidade de entender, medir e analisar o estado interno de um sistema em tempo real. A observabilidade é especialmente importante em um cenário cada vez mais complexo de aplicações distribuídas e escaláveis, pois garante que as aplicações estejam operando corretamente e ajuda a identificar problemas e gargalos antes que eles afetem negativamente os usuários finais. Portanto, a observação de aplicações é uma abordagem crucial para monitorar o desempenho, a eficiência e a qualidade das aplicações, permitindo a otimização de recursos e uma melhor experiência para os usuários finais.

Nesse contexto, é possível observar aplicações para a detecção de características e comportamentos que podem ajudar nas decisões sobre o provisionamento de recursos para otimizar a infraestrutura em que a aplicação se encontra. Desta forma, constata-se a necessidade de avaliar soluções que contribuam para uma melhor compreensão acerca das aplicações. No entanto, o grande volume de dados provenientes da observação torna análises manuais inviáveis, tornando métodos e modelos computacionais necessários.

Portanto, a ciência da computação pode ser utilizada para auxiliar nesse problema por meio de algoritmos de aprendizado de máquina aplicados aos dados observados, produzindo resultados que gerem os benefícios para a detecção de características e comportamentos das aplicações de forma automática. Com base nisso, o presente trabalho utilizou os dados de aplicações para avaliação de técnicas de aprendizagem de máquina e detecção de comportamentos de aplicações. Dados reais sobre utilização de CPU, memória, requests e latência de aplicações foram empregados para a construção de modelos de agrupamento e com isso, permitiram fazer a detecção comportamental. Foram considerados três modelos de

agrupamento do estado da arte para agrupar dados das aplicações: DBSCAN, Mean Shift e K-Means.

Os algoritmos foram analisados a partir do coeficiente da silhueta, que avalia a compactação de clusters individuais (distância intra cluster) e separação entre clusters (distância entre clusters) para medir uma pontuação representativa geral de quão bem nosso algoritmo de clustering foi executado. Os algoritmos também foram analisados por análise visual dos grupos de comportamentos detectados. Como resultado, verificou-se que o algoritmo K-Means foi o que apresentou melhor desempenho quando lidam com dados suavizados, que consistem em etapas de pré-processamento aplicados aos dados para a remoção de ruído.

2. METODOLOGIA

A solução tem a função de fazer a detecção do comportamento atual de uma aplicação com base em comportamentos passados de uma determinada aplicação.

2.1 Base de Dados e Variáveis

Para a criação dos modelos, foi necessário adquirir os dados e definir o escopo que seria analisado. Tendo isso em vista, os dados a serem utilizados são dados de aplicações que estavam em produção em um ambiente virtualizado de uma empresa parceira da UFCG, no contexto de projetos de PDI. Foram selecionadas aplicações executando em um ambiente real e não simulado, visto que os dados teriam uma variabilidade maior em relação aos comportamentos a serem detectados pelo modelo.

2.1.1 Obtenção de Dados

Os dados foram coletados de 2 aplicações e foram coletados pela ferramenta Dynatrace, utilizada para monitorar aplicativos e infraestrutura de execução. As informações adquiridas estão no formato csv e correspondem ao estado de cpu, memória, número de requisições e latência das aplicações no período de outubro de 2022 até janeiro de 2023.

Em seguida os dados foram pré-processados, por meio da renomeação das colunas do dataframe e a seleção dos dados a serem utilizados no agrupamento. Isso foi feito através do Google Collaboratory, que consiste em um serviço de nuvem gratuito hospedado pelo próprio Google para incentivar a pesquisa de Aprendizado de Máquina e Inteligência Artificial utilizando a linguagem Python.

2.1.2 Descrição dos Dados

A etapa anterior produziu um conjunto de dados de 5 colunas (variáveis) e 2161 linhas (registros) para cada aplicação. Nesse conjunto temos 4 variáveis numéricas e 1 textual. A Figura 1 mostra uma das bases de dados com as variáveis utilizadas para construção dos modelos.

	Date	Process CPU usage	Process memory usage	Request count	Reponse time
0	2022-10-21 12:00	3.83	6.26	316000.0	0.198
1	2022-10-21 13:00	5.01	6.27	436000.0	0.252
2	2022-10-21 14:00	4.38	6.29	376000.0	0.230
3	2022-10-21 15:00	4.34	6.29	378000.0	0.200
4	2022-10-21 16:00	3.77	6.30	312000.0	0.191

Figura 1: Base de dados comum às outras.

A base de dados pré-processada possui as as variáveis “Process CPU usage” e “Process memory usage”, que são variáveis numéricas e representam, respectivamente, porcentagens de uso de cpu e memória. Em relação às variáveis “Response Time” e “Request Count”, seus valores são numéricos e representam, respectivamente, o tempo médio de resposta de requisições e o número de requisições. A variável “Date” corresponde a data e hora em que o registro foi coletado da aplicação a ser analisada.

2.1.3 Transformação dos Dados

Antes da elaboração dos modelos de agrupamento, os dados adquiridos foram transformados por meio da aplicação de técnicas de normalização e suavização de dados, com o objetivo de remover ruídos das séries temporais.

A normalização dos dados foi realizada no agrupamento multidimensional, a fim de deixar todas as métricas em uma mesma escala de valores. A suavização foi realizada com a aplicação de uma média móvel com uma janela de tamanho variável. Tal tratamento visa suavizar os dados da série temporal e consequentemente otimizar o agrupamento.

Os dados transformados foram organizados em dados de treino e teste. Essa estrutura é empregada para a construção dos modelos (por meio dos dados de treino) e avaliação dos mesmos (dados de teste). De modo geral, os dados das aplicações foram divididos da seguinte forma: 80% para treino e 20% para teste.

2.2 Algoritmos de Agrupamento

O agrupamento é uma técnica de mineração de dados multivariados que através de métodos numéricos, tem por objetivo agrupar automaticamente por aprendizado não supervisionado os casos da base de dados em grupos, geralmente disjuntos denominados clusters ou agrupamentos.

Com isso, foram escolhidos três algoritmos do estado da arte sobre problemas de agrupamento: k-means, mean shift e DBSCAN. Cada um deles empregou o conjunto de dados de treino definido anteriormente para a construção dos modelos, que foram examinados em relação aos dados de teste, de modo a entender a capacidade de agrupamento de cada um com relação aos dados.

2.2.1 Métricas de avaliação

Durante a geração dos modelos, os dados de treino são fornecidos para que o modelo seja treinado. Após o treinamento, o modelo é capaz de fazer a predição de novas entradas. Porém, os dados de treino possuem características diferentes dos dados de teste. Tais diferenças acarretam um desempenho menor nos dados de teste. Dessa maneira, são necessárias métricas bem definidas para avaliar os modelos em relação aos dados que eles estão tentando agrupar.

O coeficiente de Silhueta pode auxiliar na avaliação dos agrupamentos. O coeficiente é calculado usando a distância média intra-cluster e a distância média do cluster mais próximo para cada amostra. Outra forma de avaliar é analisando de forma manual e visual os clusters detectados.

2.2.2 K-Means

É um algoritmo de agrupamento disponível na biblioteca Scikit-Learn que utiliza técnicas iterativas para agrupar casos de um conjunto de dados em clusters que contenham características semelhantes. Esses agrupamentos são úteis para explorar dados, identificar anomalias nos dados e criar previsões. Modelos de agrupamento também podem ajudar a identificar relações em um conjunto de dados que você pode não derivar logicamente por meio de navegação ou simples observação.

Ao configurar um modelo de agrupamento usando o método K-means, é especificado um número de destino K que indica o número de centróides que você deseja no modelo. Tal número pode ser encontrado utilizando o método do cotovelo. O centróide é um ponto representativo de cada cluster. O algoritmo K-means atribui cada ponto de dados de entrada a um dos clusters minimizando a soma de quadrados dentro do cluster.

Quando ele processa os dados de treinamento, o algoritmo K-means começa com um conjunto inicial de centróides escolhidos aleatoriamente. Os centróides servem como pontos de partida para os clusters e aplicam o algoritmo de Lloyd para refinar seus locais iterativamente. O algoritmo K-means para de criar e refinar os clusters quando ele atende a uma ou mais das seguintes condições: os centróides estabilizam, significando que as atribuições de cluster para pontos individuais não tem mais alteração e, portanto, o algoritmo convergiu em uma solução ou o algoritmo é concluído executando o número especificado de iterações.

2.2.3 DBSCAN

É um algoritmo de agrupamento disponível na biblioteca Scikit-Learn baseado em densidade, que é significativamente efetivo para identificar clusters de formato arbitrário e de diferentes tamanhos, identificar e separar os ruídos dos dados e detectar clusters “naturais” e seus arranjos dentro do espaço de dados, sem qualquer informação preliminar sobre os grupos. O método requer dois parâmetros de entrada, mas dá suporte para determinar um valor apropriado para eles.

A ideia chave do método DBSCAN é que, para cada ponto de um cluster, a vizinhança para um dado raio contém, no mínimo, certo número de pontos, ou seja, a densidade na vizinhança tem que exceder um limiar.

Os dois parâmetros necessários para aplicação do modelo DBSCAN são o eps, que é o raio máximo em que dois pontos podem estar para serem considerados do mesmo cluster e o minpts referente ao número mínimo de pontos em uma região necessários para garantir uma densidade desejada.

2.2.4 Mean Shift

Mean Shift é um algoritmo de agrupamento não supervisionado que visa descobrir bolhas em uma densidade suave de amostras. Ele também é conhecido por atribuir os pontos de dados aos clusters através do algoritmo de deslocamento médio. Nesse algoritmo, cada ponto tenta encontrar seu grupo movendo-se em direção à média ponderada de sua área local em cada etapa. O

destino de cada ponto será o centróide do cluster de dados ao qual o ponto pertence. Em seguida, todos os pontos de dados com o mesmo ponto de destino podem ser rotulados com o mesmo cluster. Portanto, ao contrário dos K-Means, não precisamos escolher o número de clusters nós mesmos.

3. RESULTADOS

A partir dos dados coletados e utilização da linguagem de programação Python, que possui ferramental e bibliotecas para análise de agrupamentos, para a aplicação dos algoritmos e técnicas apresentados, foi possível obter modelos resultantes de cada um dos algoritmos selecionados. Os modelos foram avaliados com os dados das aplicações monitoradas em um contexto de projeto de parceria com a Dell e por questões de privacidade vamos chamar de Aplicação DevOps e Aplicação Service API. Desta forma, é possível observar a qualidade desses algoritmos e a capacidade de agrupamento dos modelos produzidos.

Analisando os resultados, é possível observar que o desempenho de um modelo está bastante relacionado com a métrica a ser agrupada. As Tabelas 1, 2, 3 e 4 resumem os resultados referentes às métricas de CPU, memória, requisições e latência, respectivamente.

3.1 CPU

Analisando os resultados da Tabela 1, é possível observar que o dbscan obteve o melhor coeficiente de silhueta para as duas aplicações, tendo assim a menor distância intra cluster. Porém, o mesmo não conseguiu ter um bom desempenho na divisão da série temporal em clusters, retornando clusters com baixa significância, como podemos ver na Figura 2. Em relação ao mean shift, para aplicação DevOps e para Service API, ele obteve um coeficiente de silhueta razoável (alta distância intra-cluster) com uma divisão da série temporal em clusters significativos. No que diz respeito ao k-means, o algoritmo mostrou, para as duas aplicações, uma baixa distância intra-cluster. Além disso, de acordo com a Figura 2, mostrou uma divisão significativa da série temporal em clusters.

Figura 2: Agrupamento da CPU

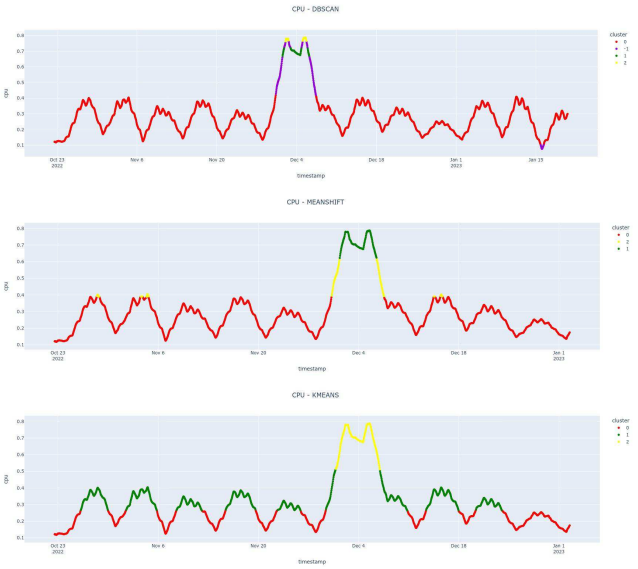

Tabela 2: Resultados dos modelos para CPU

Aplicação	Algoritmos		
	Mean shift	K-means	DBSCAN
DevOps	0,23	0,6	0,67
Service API	0,71	0,45	0,69

3.2 Memória

Ao analisar os resultados da Tabela 2, nota-se que o modelo mean shift apresentou o melhor desempenho nas duas aplicações. Além disso, de acordo com a Figura 3, a divisão das séries temporais em clusters retornou clusters significativos, resultando em clusters com alta relevância. Quanto ao k-means, para as aplicações DevOps e Service API, ele obteve uma alta distância intra-cluster e uma divisão significativa da série temporal em clusters. Em relação ao dbscan, o algoritmo não conseguiu ter um bom desempenho na divisão da série temporal em clusters.

Figura 3: Agrupamento da memória

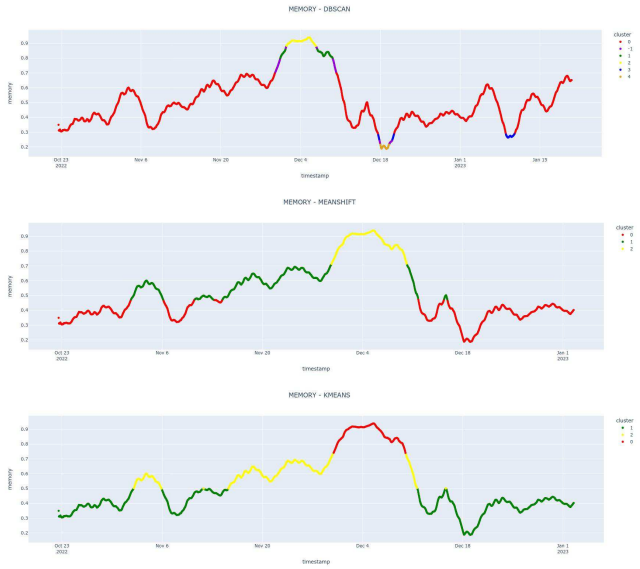

Tabela 2: Resultados dos modelos para memória

Aplicação	Algoritmos		
	Mean shift	K-means	DBSCAN
DevOps	0,55	0,54	0,23
Service API	0,6	0,45	0,23

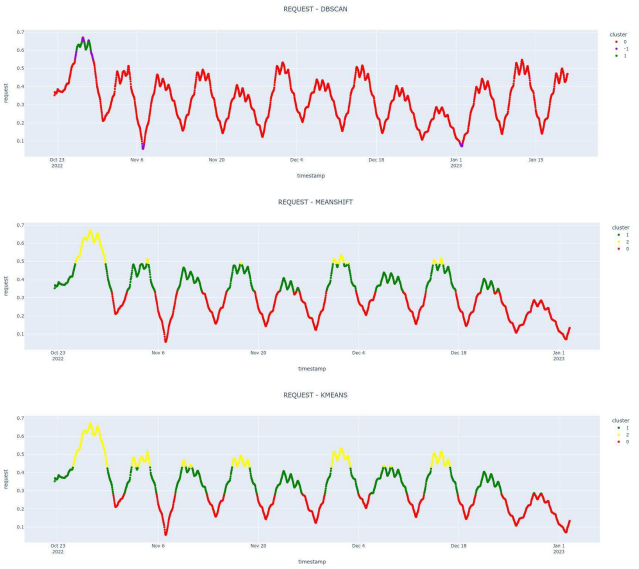

Tabela 3: Resultados dos modelos para request

Aplicação	Algoritmos		
	Mean shift	K-means	DBSCAN
DevOps	0,51	0,58	0,4
Service API	0,49	0,56	0,42

3.3 Request

Examinando os resultados da Tabela 3, percebe-se que o k-means apresentou um desempenho superior em ambas as aplicações. Além disso, sua eficácia na segmentação das séries temporais em clusters foi notável, resultando em clusters altamente relevantes. No que diz respeito ao algoritmo mean shift, para as aplicações DevOps e Service API, ele não obteve um coeficiente de silhueta muito distante do k-means. Além disso, ele conseguiu dividir significativamente a série temporal em clusters. Por outro lado, o algoritmo DBSCAN não demonstrou uma baixa distância intra-cluster e falhou em apresentar um desempenho satisfatório na segmentação da série temporal em clusters, de acordo com a Figura 4.

Figura 4: Agrupamento da request

3.4 Latência

Após analisar os dados apresentados na Tabela 4, é perceptível que o algoritmo que apresentou o melhor desempenho para ambas as aplicações foi o dbscan. No entanto, de acordo com a Figura 5, sua eficácia na divisão da série temporal em clusters foi limitada, resultando em clusters com pouca relevância para as duas aplicações. Já o mean shift obteve uma baixa distância intra-cluster para ambas, conseguindo realizar uma divisão significativa das séries temporais em clusters. Em relação ao k-means, esse algoritmo demonstrou um coeficiente de silhueta significativo e uma divisão adequada da série temporal em clusters para ambas as aplicações.

Figura 5: Agrupamento da latência

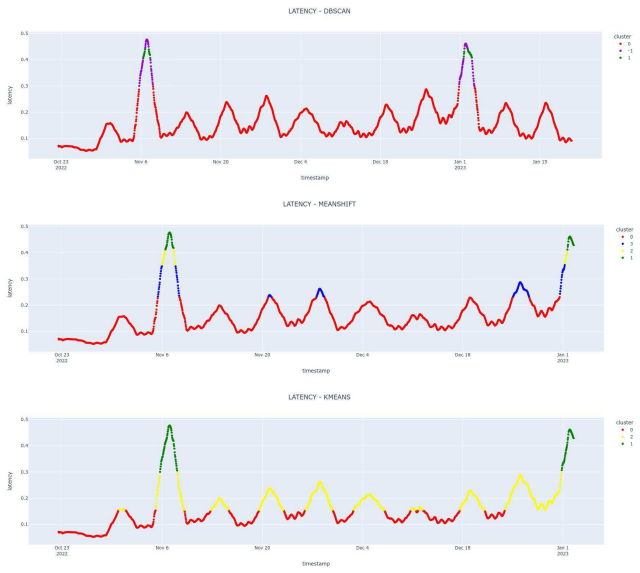

Tabela 4: Resultados dos modelos para latência

Aplicação	Algoritmos		
	Mean shift	K-means	DBSCAN
DevOps	0,5	0,66	0,69
Service API	0,54	0,46	0,69

4. CONCLUSÕES

A utilização de algoritmos de machine learning na criação de modelos abre caminho para a solução de desafios em variados contextos, incluindo a detecção de comportamentos de aplicações em séries temporais. Os resultados produzidos mostram que nem sempre um modelo de agrupamento com um silhouette score alto (baixa distância intra-cluster) apresenta o melhor agrupamento de uma série temporal em termos visuais. Sendo assim, o k-means e o mean shift apresentaram desempenhos satisfatórios com relação ao coeficiente de silhueta e a divisão da série temporal em clusters. No entanto, o algoritmo k-means obteve o melhor desempenho, demonstrando que esse modelo produz resultados satisfatórios dentro de um cenário real. Ainda verificou-se que a suavização das séries temporais foi bastante importante para produzir melhores modelos. Análises anteriores, sem a suavização, apresentaram resultados com um baixo coeficiente de silhueta, o que representa uma alta distância intra cluster.

Como trabalhos futuros, outros modelos podem ser avaliados dentro do escopo deste trabalho, como o MiniBatchKMeans, um algoritmo conhecido que usa lotes de dados pequenos, aleatórios e de tamanho fixo para armazenar na memória e, a cada iteração, uma amostra aleatória dos dados é coletada e usada para atualizar os clusters. Além disso, é possível desenvolver os modelos de forma mais específica, procurando processamentos mais complexos a serem feitos nas séries temporais.

5. REFERÊNCIAS

Dynatrace, 2023. Disponível em https://www.dynatrace.com/. Acesso em 9 de jun. de 2023.

SANTOS, T. Google Colab: o que é, tutorial de como usar e criar códigos, 2023. Disponível em https://www.alura.com.br/artigos/google-colab-o-que-e-e-como-usar. Acesso em 9 de jun. de 2023.

CASSIANO, Keila. Análise De Séries Temporais Usando Análise Espectral Singular (SSA) e Clusterização De Suas Componentes Baseada em Densidade. Orientador: Reinaldo Castro Souza. 2015. p. 172. Dissertação (Doutorado em Engenharia Elétrica) - PUC-RIO - Pontifícia Universidade Católica do Rio De Janeiro, 2014.

Compreendendo o k-means, 2023. Disponível em https://learn.microsoft.com/pt-br/azure/machine-learning/component-reference/k-means-clustering. Acesso em 9 de jun. de 2023.

Understanding Mean Shift Clustering and Implementation with Python, 2022. Disponível em https://towardsdatascience.com/understanding-mean-shift-clustering-and-implementation-with-python-6d5809a2ac40. Acesso em 9 de jun. de 2023.

sklearn.metrics.silhouette_score. Disponível em https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html. Acesso em 9 de jun. de 2023.

Clustering - Conceitos básicos, principais algoritmos e aplicações. Disponível em https://medium.com/turing-talks/clustering-conceitos-básicos-principais-algoritmos-e-aplicação-ace572a062a9. Acesso em 13 de jun. de 2023.