Implementação e validação de modelos de aprendizado não supervisionado na plataforma LEMONADE

Pesquisa Tecnológica

Luiz Henrique da Silva Gonçalves[1]
Orientador: Adriano César Machado Pereira[2]
Coorientador: Walter dos Santos Filho[3]

[1]Bacharelado em Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)
Minas Gerais – MG – Brasil

luiz10hdsg@ufmg.br

adrianoc@dcc.ufmg.br

walter@dcc.ufmg.br

Resumo. *O projeto concluído teve como proposito consolidar e validar as tarefas de aprendizado não supervisionado na plataforma LEMONADE. Os esforços do trabalho, foram centrados na implementação de algoritmos de clustering e na melhoria da interface de resultados da plataforma no módulo de Construção de Modelos chamado MODEL BUILDER, com adição de componentes que auxiliam os usuários na validação dos experimentos. Foi feito um amplo estudo de viabilidade em relação a quais elementos e algoritmos poderiam ser adicionados a plataforma, com proposito de tornar o processo de treinamentos de modelos mais eficiente e intuitivo. A finalidade do projeto foi contribuir de forma ativa na construção de um interface versátil e funcional para usuários com diferentes níveis de conhecimento em programação e aprendizado de máquina. No processo de implementação, foi feito uma analise dos modelos de aprendizado não supervisionado compatíveis com o pipeline do LEMONADE e também avaliando as limitações do framework utilizado. Testes abrangentes e correções foram realizados, visando garantir robustez na execução do treinamento dos modelos. Foram adicionados quatro algoritmos de clustering, K-means, Gaussian mixture, Latent Dirichlet allocation (LDA), Bisecting k-means. Além disso, foi feito uma analise de quais componentes poderiam ser incorporados na interface dos resultados, para ajudar na interpretação dos experimentos para a tarefa de agrupamento. Por fim, foi realizado uma serie de experimentos práticos para consolidação do funcionamento dos algoritmos e componentes implementados. O projeto fortaleceu o LEMONADE, proporcionando um ambiente mais poderoso e acessível para a construção eficaz de modelos de aprendizado de máquina em diferentes contextos.*

1. Introdução

A constante evolução das tecnologias de processamento de dados e aprendizado de máquina tem desempenhado um papel crucial na resolução de desafios complexos em diversas áreas. Dentro desse cenário, a plataforma LEMONADE emerge como uma proposta inovadora para a computação distribuída e o desenvolvimento de aplicações voltadas ao processamento de dados e aprendizado de máquina. Este trabalho concentra-se na otimização e expansão dos modelos de aprendizado não supervisionado, no módulo de construção de modelos, conhecido como MODEL BUILDER.

1.1. Caracterização do problema

O advento do LEMONADE introduz uma plataforma visual para computação distribuída, direcionada a usuários com distintos níveis de conhecimento em programação e aprendizado de máquina. Contudo, para consolidar plenamente a eficácia dessa ferramenta, há a necessidade de expandir e aprimorar o MODEL BUILDER, com a adição da tarefa de aprendizado não supervisionado. Assim, o objetivo é tornar o LEMONADE uma plataforma ainda mais versátil, suportando uma ampla gama de analises e experimentos de diversas áreas do conhecimento.

1.2. Motivação

A motivação principal do projeto reside na tentativa de democratizar o processo de treinamento de modelos de aprendizado não supervisionado. O desenvolvimento desses modelos, realizado por cientistas de dados, é uma tarefa complexa e multidisciplinar, que envolve muitos conhecimentos prévios, como programação e estatística. Envolve desde a preparação dos dados até a seleção cuidadosa de parâmetros, passando pela avaliação constante dos resultados obtidos. A complexidade e dificuldade do processo de treinamento de modelos de aprendizado de maquina processo é o ponto focal deste trabalho, a ideia é tentar tornar todo esse desenvolvimento mais intuitivo para qualquer tipo de usuário.

1.3. Objetivos

O objetivo deste trabalho foi consolidar e validar as tarefas de aprendizado não supervisionado na plataforma LEMONADE, adicionando novos algoritmos de *clustering*, avaliando a possibilidade da adição de novos modelos, novas métricas de analise e adicionar elementos visuais de validação dos experimentos na interface de resultados. Por fim, realizar uma serie de testes, visando demostrar o funcionamento pleno dessa tarefa no MODEL BUILDER. O proposito geral foi tornar essa ferramenta mais abrangente, capaz de suportar uma variedade de estudos e análises em diferentes domínios.

1.4. Estrutura do Trabalho

O trabalho desenvolvido no projeto seguiu primeiramente com um estudo detalhado de quais algoritmos *clustering* poderiam ser incorporados ao LEMONADE, levando em consideração algumas limitações em relação a compatibilidade do pipeline da plataforma e também limitações de implementação e escalabilidade dos *frameworks* disponíveis atualmente como o *Apache Spark* e o *Scikit Learn*, dentre outras bibliotecas de *machine learning*. Em seguida foram feitos analises de uma variedade métricas de avaliação disponíveis para a tarefa de aprendizado não supervisionado. Foram feitas pontuações em relação a limitações de uso dessas métricas no LEMONADE, em relação a escalabilidade e disponibilidade desses métodos no framework utilisado. Por fim, também foi feito um estudo de componentes visuais e elementos estatísticos que poderiam ser incorporados na interface de resultados. A próxima etapa, se seguiu com a implementação dos novos algoritmos e melhorarias na interface de resultados, com a adição de novos componentes descritivos. Na ultima etapa,foi realizado testes de certificação, verificando alguns fatores da interface como usabilidade, campos de configuração dos experimentos e disponibilidade das *features* das bases de dados. Na mesma etapa de testes, foi realizado testes de execução, com a geração e verificação do código da execução gerada dos experimentos, com o intuito de demostrar de fato o funcionamento e eficacia dos novos modelos implementados, com experimentos conduzidos e verificação dos resultados do treinamento para a tarefa de *clustering*, foram analisados o funcionamento de todos os componentes novos adicionados.

2. Referencial Teórico

Nesta seção, apresentaremos os conceitos e aspectos teóricos essenciais relacionados ao projeto implementado.

2.1. Aprendizado de Máquina

O aprendizado de máquina (*Machine Learning*) é um subcampo da inteligência artificial que se concentra no desenvolvimento de algoritmos que permitem aos computadores aprenderem a partir de dados. Ele visa capacitar os sistemas a identificar padrões e fazer previsões ou decisões sem serem explicitamente programados para realizar tarefas específicas (Tufail et al. 2023). Existem diferentes abordagens para o aprendizado de máquina, incluindo o aprendizado supervisionado, não supervisionado e por reforço (Figura 1). O aprendizado supervisionado envolve treinar um modelo em um conjunto de dados rotulados, onde cada entrada tem uma saída desejada, com o objetivo de aprender uma função que mapeia entradas para saídas corretas, permitindo a previsão de novos dados não vistos. O aprendizado por reforço, por outro lado, baseia-se em agentes que

aprendem a tomar decisões através de interações com um ambiente, recebendo recompensas ou punições com base nas ações realizadas (Goodfellow e Bengio 2020).

No contexto deste projeto, o foco está no aprendizado não supervisionado, que é utilizado para descobrir padrões ocultos ou agrupamentos intrínsecos nos dados sem a necessidade de rótulos pré-definidos. Esse tipo de aprendizado é amplamente utilizado em diversas aplicações práticas. Por exemplo, a segmentação de clientes permite agrupar consumidores com características e comportamentos semelhantes, facilitando campanhas de marketing direcionadas e a personalização de produtos e serviços. Na análise de imagens e vídeos, é utilizado para segmentação de imagem, reconhecimento de padrões e agrupamento de vídeos similares, melhorando a precisão na classificação de imagens e facilitando a busca de imagens similares (Brown et al. 2020).

Na bioinformática, o aprendizado não supervisionado é empregado para analisar dados genômicos e proteômicos, identificando grupos de genes ou proteínas com funções semelhantes e descobrindo padrões em dados biológicos, o que pode acelerar a descoberta de novos medicamentos e melhorar a compreensão de processos biológicos. Técnicas de redução de dimensionalidade, como a Análise de Componentes Principais (PCA), simplificam a complexidade dos dados, mantendo a estrutura essencial para visualização e interpretação, facilitando a visualização de dados de alta dimensão e ajudando na remoção de ruído.

A detecção de fraudes é outra aplicação importante, onde o aprendizado não supervisionado é usado para identificar atividades anômalas em transações financeiras e outras áreas propensas a fraudes, aumentando a precisão na detecção de fraudes e melhorando a segurança. Em manutenção preditiva, é utilizado para monitorar equipamentos e prever falhas antes que ocorram, reduzindo custos de manutenção, evitando paradas não planejadas e prolongando a vida útil dos equipamentos.

Essas aplicações ilustram o grau de importância desse trabalho mostrando de forma plena como o aprendizado não supervisionado pode ser uma ferramenta poderosa para extrair valor de dados não rotulados, proporcionando *insights* valiosos e apoiando a tomada de decisões informadas em diversas áreas. No projeto LEMONADE, a integração desses algoritmos dentro do Model Builder visa simplificar e otimizar esses processos, tornando o aprendizado de máquina acessível e eficiente para usuários de diferentes níveis de especialização.

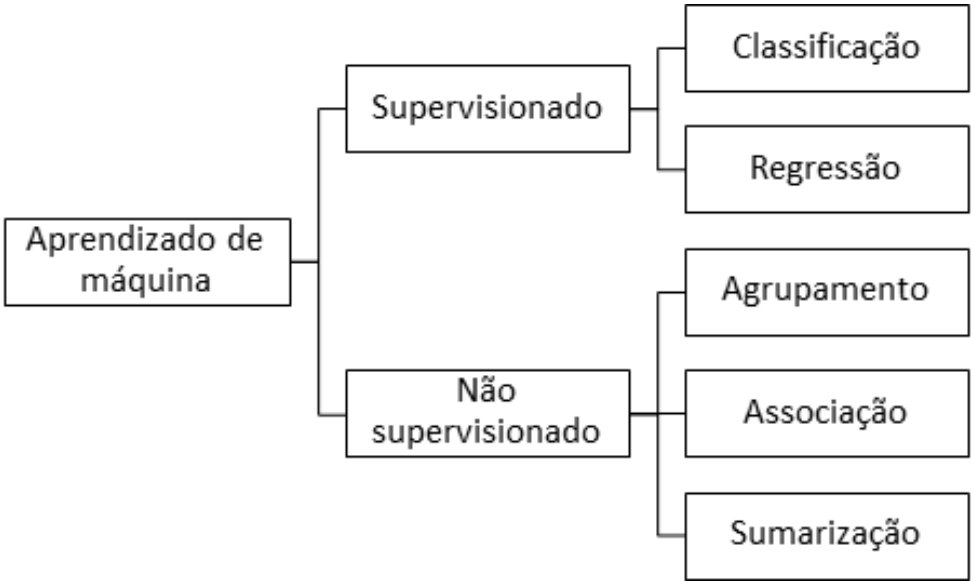

Figura 1. Aprendizado de Maquina.

2.2. A Plataforma LEMONADE

O LEMONADE *Documentação do Lemonade* s.d. é uma plataforma visual para computação distribuída, destinada a permitir a implementação, experimentação, teste e implantação de aplicações de processamento de dados e aprendizado de máquina. Ele fornece um nível mais alto de abstrações, chamado de operações, para os usuários criarem fluxos de processamento usando uma interface gráfica da Web (Figura 2). Usando tecnologias de alto desempenho e escalonáveis, como COMPSs, Ophidia e Spark, o LEMONADE pode processar uma grande quantidade de dados, ocultando toda a complexidade do *backend* para os usuários e permitindo que eles se concentrem principalmente na construção da solução.

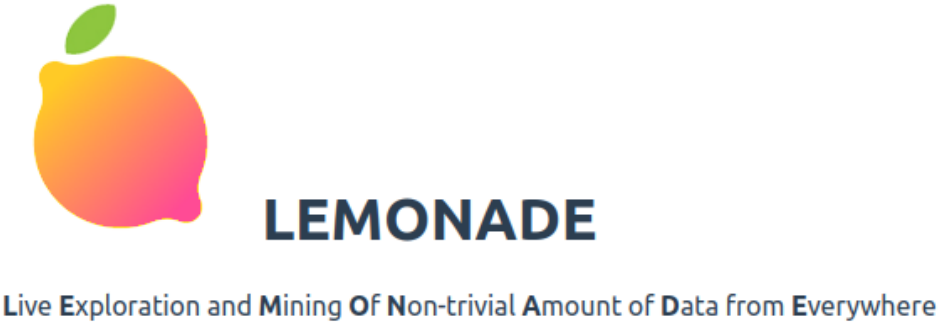

Figura 2. LEMONADE

2.3. Model Builder

O MODEL BUILDER é o modulo de construção modelos de aprendizado de máquina do LEMONADE (Figura 3). O foco principal desse modulo é permitir a análise de métricas relacionadas à construção do modelo. A partir de uma tarefa, o usuário pode escolher um ou mais algoritmos a serem executados. Cada algoritmo, conjuntos de parâmetros de execução e métricas de avaliação poderão ser definidos. Ao confirmar a criação do modelo de aprendizado de máquina, o usuário poder escolher quais modelos são os mais adequados segundo seus critérios.

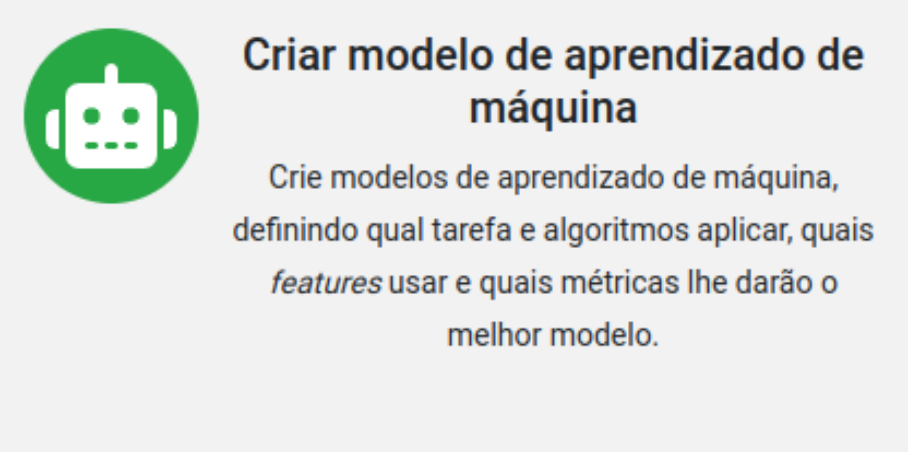

Figura 3. Gerador de modelos de aprendizado de maquina.

2.4. Arquitetura do Lemonade

O Lemonade é composto por vários módulos, que colaboram entre si por meio de APIs e possui uma arquitetura baseada em microsserviços (Figura 4). Para o contexto do projeto desenvolvido, apenas dois serviços foram trabalhados nas atividades, os seguintes serviços *Juicer* e *Tahiti*.

O Juicer tem quatro responsabilidades principais: Receber uma especificação de fluxo de trabalho (JSON) do Stand e convertê-la em código executável (transpilar operações para o respectivo código de tecnologia subjacente). Executar o código gerado, controlando o fluxo de execução. Reportar status de execução para o Stand. Interagir com a API do Limonero para criar novos conjuntos de dados e registrar os seus metadados. O Juicer irá gerar o código direcionado a uma plataforma de processamento distribuída, tal como o Spark.

O Tahiti gerencia os metadados associados às operações. As operações são a menor unidade de processamento e representam uma tarefa de granularidade alta executada em um dos backends suportados. Atualmente, a LEMONADE suporta extração, transformação, carregamento (ETL) e algumas operações de aprendizagem de máquina.

Metadados incluem nome da operação, descrição, parâmetros e portas. Portas são pontos de comunicação que possuem direção (entrada e saída), multiplicidade (quantas conexões suportadas) e devem "implementar" interfaces para garantir a compatibilidade entre as operações. Cada operação tem um conjunto de parâmetros agrupados como formulários. Os formulários são organizados em classes diferentes, por exemplo, execução, segurança e privacidade, qualidade de serviço, aparência, relatório e registro.

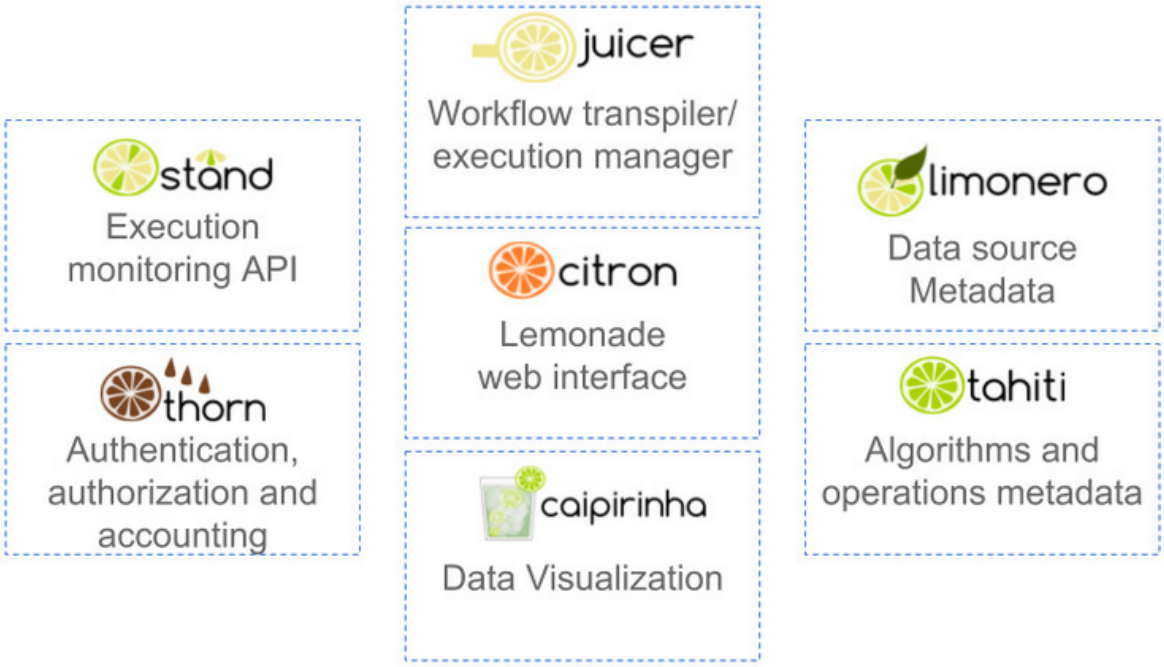

Figura 4. Organização em micro-serviços

2.5. Apache Spark

O Apache Spark (*Apache Spark* s.d.), é uma poderosa plataforma de processamento de dados em larga escala, desempenha um papel central na implementação do trabalho. Essa estrutura distribuída permite manipular grandes conjuntos de dados com eficiência, proporcionando um ambiente ideal para o processamento de aprendizado de máquina (ML). No contexto deste projeto, o Spark é essencial para aprimorar a capacidade do LEMONADE.

2.6. Clustering

Clustering é uma técnica de aprendizado não supervisionado que envolve a segmentação de um conjunto de dados em grupos homogêneos. Cada grupo, ou *cluster*, contém itens que são mais semelhantes entre si do que com itens de outros grupos (Figura 5). A tarefa

de agrupamento tem diversas utilidades, com aplicações na segmentação de clientes para personalizar estratégias de marketing (personas), agrupamento de documentos para organizar bibliotecas digitais, análise de dados genéticos para identificar padrões biológicos e muitos outros.

Na implementação desse projeto, vários algoritmos de *clustering* foram analisados cada um com suas próprias características e aplicações. (Jiang, Li e Zhang 2022.).

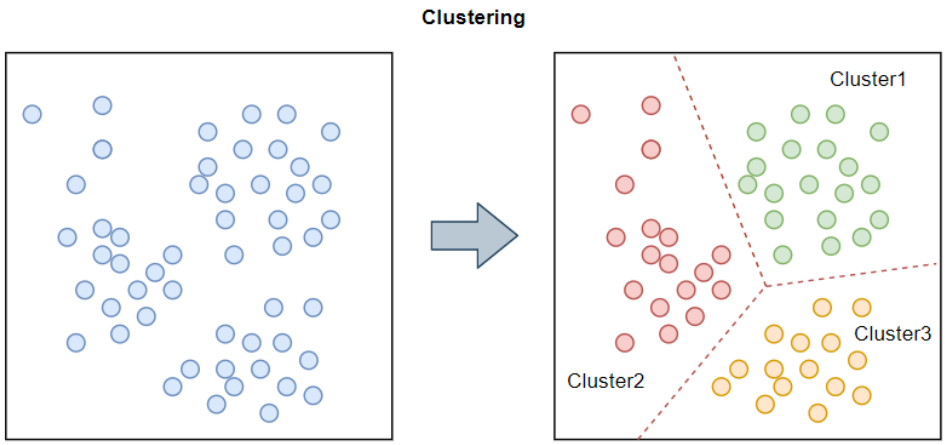

Figura 5. Aplicação da tarefa de clustering.

2.6.1. K-means

O *K-means* é um dos algoritmos mais populares e amplamente utilizados para clustering. Ele funciona particionando os dados em k *clusters*, onde cada ponto de dados pertence ao *cluster* com o centróide mais próximo (Figura 7). O algoritmo começa com a escolha inicial de k centróides, que podem ser selecionados aleatoriamente ou usando algum método de inicialização, como o *K-means++*. Em seguida, os pontos de dados são atribuídos ao centróide mais próximo, formando os clusters iniciais. Depois, os centróides são recalculados como a média dos pontos de dados em cada cluster, e o processo de atribuição e recalculação continua iterativamente até que a convergência seja alcançada, ou seja, até que os centróides não mudem significativamente de uma iteração para outra (Na, Xumin e Yong 2010).

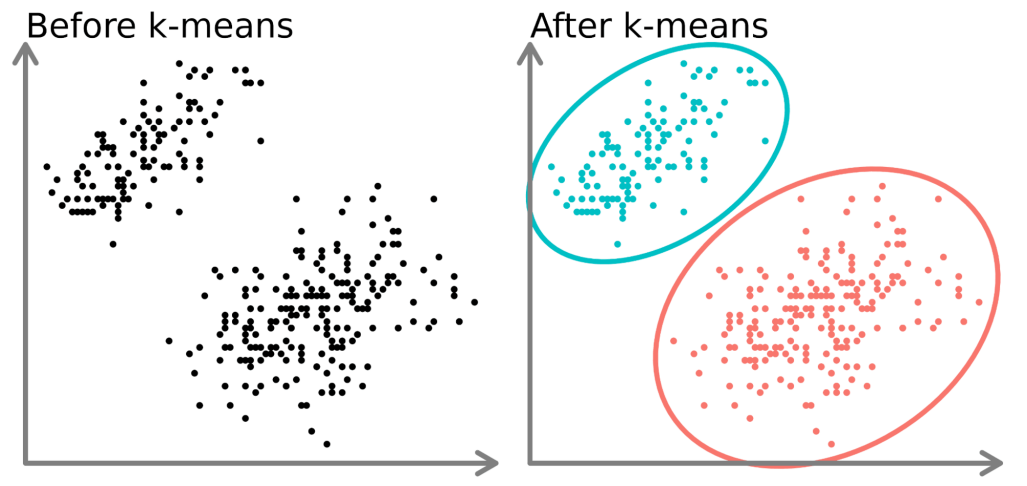

Figura 6. Execução do Kmeans.

Uma das principais vantagens do *K-means* é sua simplicidade e eficiência computacional, especialmente em termos de tempo de execução, o que o torna adequado para grandes conjuntos de dados. No entanto, o algoritmo tem algumas limitações. Ele assume que os clusters são de forma esférica e de tamanho similar, o que pode não ser o caso em muitos problemas do mundo real. Além disso, *K-means* é sensível a *outliers* e a inicialização dos centróides, que pode levar a diferentes resultados. Para mitigar o problema de inicialização, o método *K-means++* pode ser usado para selecionar centróides iniciais de forma mais cuidadosa, melhorando a convergência e a qualidade dos *clusters* (Celebi, Kingravi e Vela 2013).

2.6.2. GMM

A identificação de padrões e *clusters* entre grandes volumes de dados é frequentemente necessária no aprendizado de máquina e na análise de dados. As técnicas tradicionais de agrupamento, como o agrupamento *k-means*, têm dificuldades na detecção de grupos de formas e tamanhos variados. Modelos de mistura gaussiana (GMMs) podem ajudar nessas limitações.

Esses modelos são usados para categorizar os dados em vários grupos, dependendo da distribuição de probabilidade. Os modelos de mistura gaussiana têm diversas aplicações, incluindo finanças, marketing e muito mais.

Modelos de mistura gaussiana (GMM) são modelos probabilísticos usados para modelar conjuntos de dados do mundo real. GMMs são generalizações de distribuição gaussiana que podem ser usadas para descrever qualquer conjunto de dados que possa ser

agrupado em múltiplas distribuições gaussianas. O GMM é um modelo probabilístico no qual todos os pontos de dados são considerados criados por uma mistura de distribuições gaussianas com parâmetros desconhecidos. *Clustering*, ou agrupamento de um conjunto de pontos de dados em *clusters*, pode ser realizado usando um modelo de mistura gaussiana.

O *Gaussian Mixture Model* (GMM) estende o *K-means* ao permitir que os *clusters* tenham formas elípticas, modelando os dados como uma mistura de distribuições gaussianas. O GMM utiliza o algoritmo de *Expectation-Maximization* (EM) para encontrar os parâmetros das distribuições que melhor descrevem os dados. Este método é mais flexível que o *K-means* e pode capturar a variabilidade dos dados de maneira mais precisa (Figura 7).

O processo de ajuste do GMM envolve duas etapas principais: a fase de Expectation (E) e a fase de Maximization (M). Na fase E, o algoritmo calcula a probabilidade de cada ponto de dados pertencer a cada componente gaussiano. Na fase M, ele ajusta os parâmetros das distribuições gaussianas para maximizar a probabilidade conjunta dos dados observados. Este processo é iterado até a convergência (Wan et al. 2019).

Uma das principais vantagens do GMM é sua capacidade de modelar dados que têm uma distribuição não esférica e variâncias diferentes. Além disso, ele pode determinar a probabilidade de um ponto de dados pertencer a cada *cluster*, permitindo uma interpretação probabilística das atribuições de *clusters*. No entanto, o GMM é computacionalmente mais intensivo que o *K-means* e pode ser difícil de ajustar em termos de número de componentes, pois um número inadequado de componentes pode levar a uma modelagem excessiva ou insuficiente dos dados.

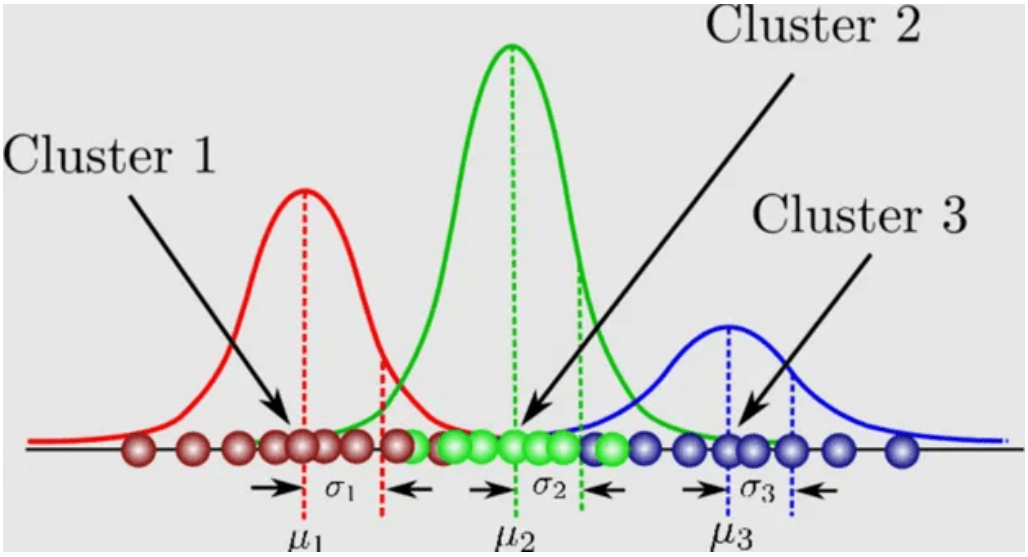

Figura 7. Distribuição dos clusters com GMM.

2.6.3. LDA

O *Latent Dirichlet Allocation* (LDA) é um modelo generativo para agrupamento de dados textuais. Ele representa documentos como distribuições sobre tópicos, e tópicos como distribuições sobre palavras, permitindo identificar temas subjacentes em grandes corpora de texto. LDA é amplamente utilizado em processamento de linguagem natural para tarefas como categorização de documentos e descoberta de tópicos, oferecendo uma maneira poderosa de explorar grandes conjuntos de dados textuais, embora possa ser computacionalmente exigente e sensível ao número de tópicos especificados.

O algoritmo Latent Dirichlet Allocation (LDA) é uma técnica popular usada no processamento de linguagem natural e no aprendizado de máquina para descobrir temas ou tópicos subjacentes em uma coleção de documentos. A modelagem de tópicos é uma abordagem de modelagem estatística que revela os tópicos latentes presentes em uma coleção de documentos. Esse método fornece uma maneira de representar dados de texto de maneira estruturada, permitindo que pesquisadores e profissionais explorem e analisem eficientemente grandes volumes de informações textuais. A modelagem de tópicos permite a organização e categorização de documentos com base em seus temas subjacentes, facilitando a navegação e a pesquisa em vastos dados textuais. Ao identificar os principais tópicos dentro de uma coleção de documentos, a modelagem de tópicos pode gerar resumos concisos que capturam a essência do conteúdo. A modelagem de tópicos facilita a recomendação de conteúdo personalizado, compreendendo os tópicos de interesse de usuários individuais.

O algoritmo Latent Dirichlet Allocation (LDA) funciona de forma iterativa para identificar temas em um conjunto de documentos. Inicialmente, no passo 1, cada palavra em um documento é aleatoriamente atribuída a um dos K tópicos. No passo 2, ocorre a inferência iterativa, onde as atribuições de tópicos das palavras são reavaliadas com base nas distribuições de tópicos-palavras e documentos-tópicos. Esse processo continua repetidamente até a convergência do modelo ou até que um critério de parada pré-definido seja alcançado. Após a convergência, o LDA fornece as distribuições de tópicos-palavras e documentos-tópicos, que podem ser analisadas para a interpretação e exploração dos tópicos identificados (Figura 8).

Em resumo, o Latent Dirichlet Allocation (LDA) é uma ferramenta poderosa e amplamente utilizada no processamento de linguagem natural e aprendizado de máquina para a descoberta de temas subjacentes em grandes coleções de documentos. Representando documentos como distribuições sobre tópicos e tópicos como distribuições sobre palavras, o LDA permite a organização, categorização e análise eficiente de grandes volumes de dados textuais. Apesar de ser computacionalmente exigente e sensível ao número

de tópicos especificados, sua capacidade de identificar e estruturar informações textuais de maneira intuitiva facilita a navegação, pesquisa e recomendação de conteúdo personalizado.

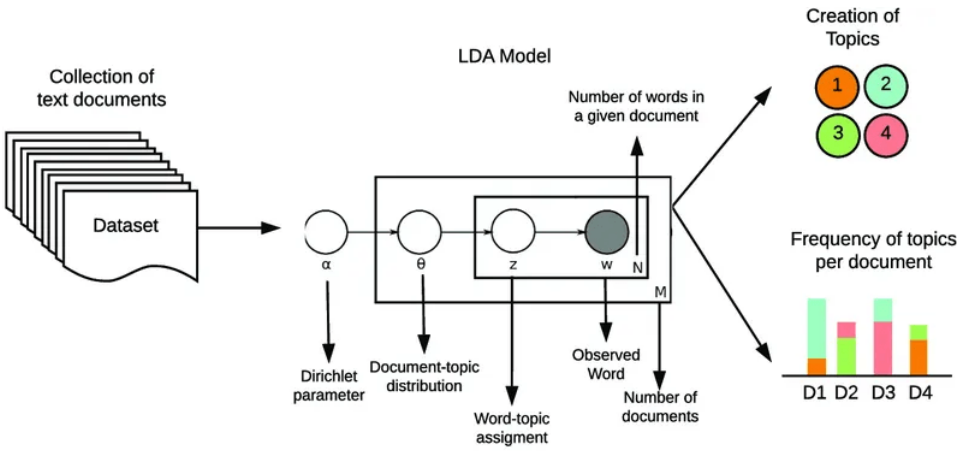

Figura 8. Execução do LDA.

2.6.4. Bisecting K-means

O *Bisecting k-means* (Rohilla et al. 2019) é uma abordagem híbrida entre Clustering Hierárquico Divisivo (*clustering* de cima para baixo) e *Clustering K-means*. Em vez de particionar o conjunto de dados em K *clusters* em cada iteração, o algoritmo o *Bisecting k-means* divide um *cluster* em dois *subclusters* em cada etapa de bissecção (usando k-means) até que k *clusters* sejam obtidos (Figura 9).

O *Bisecting K-means* pode produzir melhores resultados que o *K-means* tradicional em termos de qualidade de clusters, especialmente em datasets onde a estrutura hierárquica dos dados é relevante. A abordagem divisiva permite que o algoritmo se adapte melhor à forma e densidade dos dados, proporcionando uma maior flexibilidade na formação dos clusters. Além disso, a estrutura hierárquica resultante pode ser útil para análises que requerem uma decomposição mais detalhada dos dados.

No entanto, como o *K-means*, o *Bisecting K-means* também pode ser sensível a *outliers* e inicializações diferentes. A escolha inicial dos pontos de divisão pode influenciar significativamente os resultados finais, e a presença de *outliers* pode distorcer a formação dos *subclusters*. Métodos de pré-processamento de dados, como a remoção de *outliers* e a normalização, podem ajudar a mitigar esses problemas.

A *Bisecting K-means* é mais eficiente quando K é grande. Para o algoritmo *kmeans*, o cálculo envolve todos os pontos de dados do conjunto de dados e k centróides. Por outro lado, em cada etapa *Bisecting* de *Bisecting k-means*, apenas os pontos de dados de

um *cluster* e dois centróides estão envolvidos no cálculo. Assim, o tempo de cálculo é reduzido. A bissecção do *k-means* produz *clusters* de tamanhos semelhantes, enquanto *k-means* é conhecido por produzir *clusters* de tamanhos amplamente diferentes.

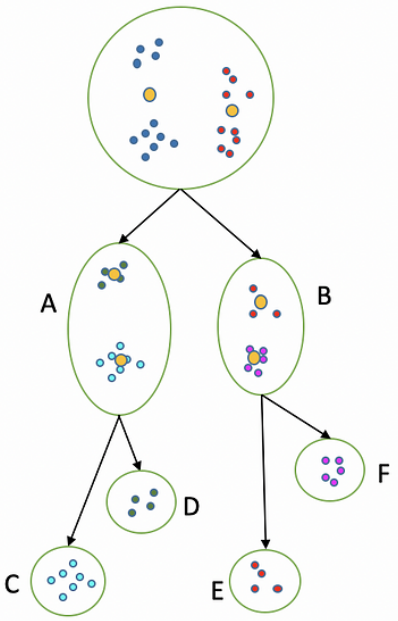

Figura 9. Execução do Bisecting K-means.

2.6.5. PIC

O *Power Iteration Clustering* (PIC) é um método eficiente para agrupar grandes conjuntos de dados, baseado na iteração na técnica de *Power Iteration* (iteração de potência) para encontrar as componentes principais dos dados. Este algoritmo é particularmente útil para problemas de *clustering* espectral, onde a estrutura dos dados pode ser representada como um grafo. PIC é eficiente em termos de memória e tempo, tornando-o adequado para grandes volumes de dados, mas sua eficácia depende da escolha apropriada dos parâmetros de inicialização (Lin e Cohen 2010).

O PIC funciona iterativamente, começando com a escolha de um vetor inicial aleatório. Na execução do algoritmo, primeiro, o PIC constrói uma matriz de similaridade que representa a estrutura dos dados como um grafo, onde os nós correspondem aos dados e as arestas refletem a similaridade entre eles. Em seguida, o algoritmo aplica o método da iteração de potência para encontrar os autovalores e autovetores principais dessa matriz de similaridade. Este processo envolve multiplicar repetidamente a matriz de similaridade pelo vetor inicial, normalizando-o a cada passo, até que o vetor convirja para o principal

autovetor da matriz. Esse autovetor principal representa as componentes principais dos dados e é utilizado para identificar os agrupamentos. Finalmente, o algoritmo atribui os dados a *clusters* com base nas componentes principais identificadas, agrupando dados semelhantes juntos. Esse método é eficiente em termos de memória e tempo, tornando-o adequado para grandes conjuntos de dados (Figura 10).

A principal vantagem do PIC reside na sua capacidade de escalar bem com o aumento do tamanho dos dados, permitindo a aplicação em contextos onde outros métodos de *clustering* podem falhar devido a limitações de recursos computacionais. Além disso, por ser baseado na iteração por Power Iteration, o PIC pode identificar de forma eficiente as estruturas subjacentes dos dados, revelando agrupamentos naturais que podem não ser aparentes a partir de métodos de *clustering* tradicionais.

Contudo, é importante notar que o desempenho do PIC pode ser significativamente influenciado pela qualidade dos dados de entrada e pela escolha dos parâmetros de inicialização. Parâmetros mal escolhidos podem levar a resultados sub-ótimos, destacando a necessidade de uma compreensão sólida do domínio dos dados e de experimentação cuidadosa para ajustar o algoritmo às necessidades específicas do problema em questão.

Além disso, enquanto o PIC é robusto e eficiente, ele pode não capturar bem a complexidade de dados altamente não-lineares ou quando os *clusters* possuem formas complexas. Nestes casos, pode ser necessário combinar PIC com outras técnicas de pré-processamento ou de pós-processamento para melhorar a qualidade dos resultados obtidos.

O *Power Iteration Clustering* (PIC) é uma ferramenta poderosa para o agrupamento de grandes volumes de dados, oferecendo eficiência e escalabilidade. Sua aplicação bem-sucedida requer atenção aos detalhes na escolha dos parâmetros de inicialização e uma compreensão profunda da natureza dos dados a serem analisados.

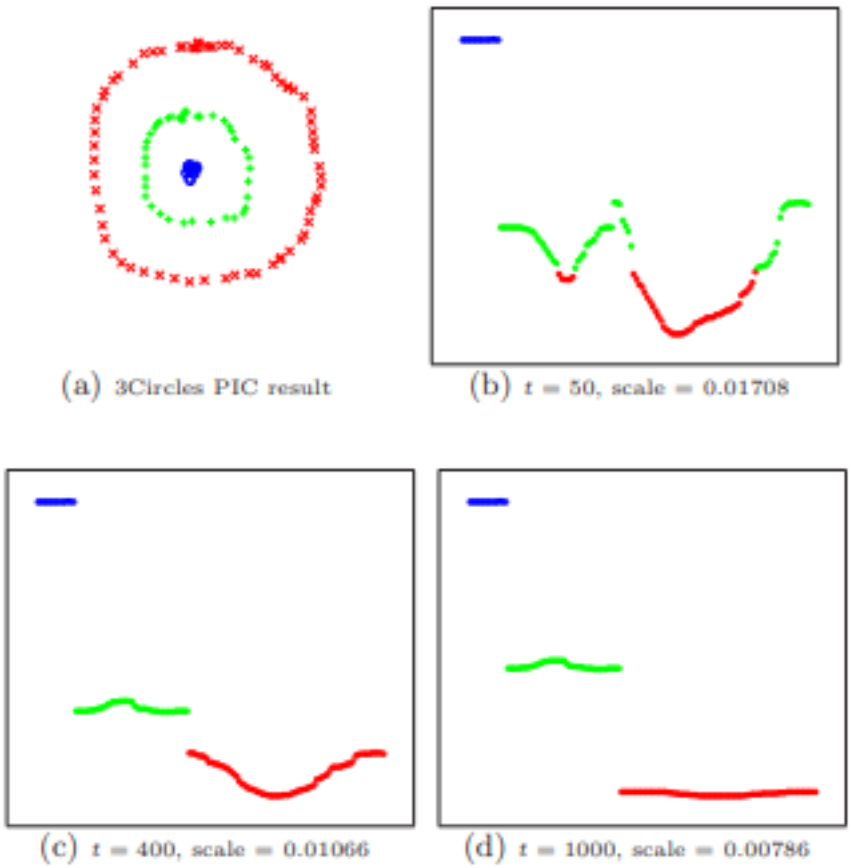

Figura 10. Resultado do agrupamento e a incorporação fornecida por v_t para o conjunto de dados 3Circles. Em (b) até (d), o valor de cada componente de v_t é plotado contra seu índice. Os gráficos (b) a (d) são reescalados de modo que o maior valor esteja sempre no topo e o menor valor na parte inferior, com a escala sendo a diferença entre o valor máximo e o mínimo.

2.7. Métricas de avaliação

Na tarefa de *clustering*, é essencial avaliar a qualidade dos agrupamentos gerados para entender a estrutura dos dados e a eficácia do algoritmo de *clustering* utilizado. A seguir, é apresentado uma descrição de algumas métricas que foram analisadas durante o desenvolvimento do trabalho.

2.7.1. Índice de Silhueta (*Silhouette Score*)

O Índice de Silhueta mede a coesão e a separação dos *clusters* formados. É calculado para cada ponto de dados e fornece uma avaliação de quão semelhante um ponto é ao seu

próprio *cluster* em comparação com outros *clusters*. A fórmula do Índice de Silhueta para um ponto i é dada por:

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Onde:

- $a(i)$ é a distância média entre i e todos os outros pontos do mesmo *cluster*.
- $b(i)$ é a distância mínima entre i e todos os pontos de qualquer outro *cluster*.

O coeficiente é gerado entre o intervalo de -1 a +1, onde quanto mais próximo a "+1" representa maior distância entre os *clusters*; quando for "0" significa que estão muito próximos do ponto de decisão entre os *clusters* (ou seja, está na dúvida a qual pertence). Quando os valores forem negativos isso significa que possivelmente os dados estão no *cluster* errado. Também deve ser levado em consideração na análise o valor médio do coeficiente, sendo que, no cenário ideal, o coeficiente de silhueta de cada *cluster* deve ser maior que o valor médio do mesmo e as espessuras de cada um no gráfico devem ser semelhantes entre si.

2.7.2. Índice de *Davies-Bouldin*

O Índice de *Davies-Bouldin* mede a dispersão dentro dos *clusters* e a separação entre os *clusters*. É definido como a média dos rácios de similaridade para cada *cluster* com relação ao *cluster* que mais se assemelha a ele. A fórmula do Índice de *Davies-Bouldin* é:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right)$$

Onde:

- σ_i é a dispersão média do *cluster* i.
- $d(c_i, c_j)$ é a distância entre os centróides dos *clusters* i e j.

Valores mais baixos do Índice de *Davies-Bouldin* indicam *clusters* mais compactos e melhor separados. O objetivo dessa metrica é avaliar quão "bem separados" foram os *clusters*. O resultado índice é similaridade padrão entre os grupos, cujo resultado pode variar de 0 até o infinito positivo. Como queremos que nossos grupos sejam o mais bem particionados quanto possível, quanto mais próximo de 0 for o nosso resultado, melhor.

2.7.3. Índice de *Calinski-Harabasz*

O Índice de *Calinski-Harabasz*, também conhecido como Critério de Variância Razão, mede a proporção entre a soma da dispersão entre *clusters* e a soma da dispersão dentro dos *clusters*. É definido como:

$$CH = \frac{(N-k)}{(k-1)} \cdot \frac{\sum_{i=1}^{k} n_i \|c_i - \bar{x}\|^2}{\sum_{i=1}^{k} \sum_{x \in C_i} \|x - c_i\|^2}$$

Onde:

- N é o número total de pontos de dados.
- k é o número de *clusters*.
- n_i é o número de pontos no cluster i.
- c_i é o *centróide* do *cluster* i.
- $\bar{x}$ é a média de todos os pontos de dados.

Valores mais altos do Índice de *Calinski-Harabasz* indicam *clusters* mais densos e bem separados. Em termos abstratos, podemos dizer que o índice mede, simultaneamente duas coisas: quão distantes os *clusters* estão um dos outros (dispersão entre *clusters*) e quão densos estão os *clusters* em si (dispersão dentro dos *clusters)*. Seu valor varia de 0 até o infinito positivo, e quanto maior, mais próximo da solução ideal você está.

2.7.4. *Perplexity* (LDA)

No contexto da Latent Dirichlet Allocation (LDA), uma técnica de clustering usada para modelagem de tópicos, a Perplexity é uma métrica que mede a capacidade do modelo de prever um conjunto de palavras. Ela avalia a qualidade dos tópicos gerados pelo modelo LDA. A Perplexity é definida como:

$$\text{Perplexity}(D) = \exp\left(-\frac{\sum_{d=1}^{M} \log p(w_d)}{\sum_{d=1}^{M} N_d}\right)$$

Onde:

- M é o número de documentos.
- N_d é o número de palavras no documento d.
- $p(w_d)$ é a probabilidade do conjunto de palavras no documento d.

Valores mais baixos de *Perplexity* indicam um modelo de tópicos melhor, pois representam uma incerteza menor na previsão das palavras.

2.7.5. *Coherence Score* (LDA)

O *Coherence Score* é uma métrica usada para avaliar a interpretabilidade dos tópicos gerados pelo modelo LDA. Ele mede a semântica de coesão entre as palavras em cada tópico. Existem várias variantes dessa métrica, como UMass, UCI e NPMI. Um exemplo comum é a *Coherence Score* baseada em pares de palavras:

$$\text{Coherence}(T) = \sum_{t=1}^{T} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \log \frac{P(w_i, w_j) + \epsilon}{P(w_i)P(w_j)}$$

Onde:

- T é o número de tópicos.
- N é o número de palavras em um tópico.
- $P(w_i, w_j)$ é a probabilidade conjunta de palavras w_i e w_j aparecerem no mesmo documento.
- ϵ é uma pequena constante para evitar logaritmo de zero.

Valores mais altos de *Coherence Score* indicam tópicos mais coerentes e interpretáveis.

2.7.6. *Log Verossimilhança* (LDA)

A Log Verossimilhança é uma métrica que mede a probabilidade dos dados observados sob o modelo de tópicos LDA. É uma indicação de quão bem o modelo LDA explica os dados observados. A fórmula da *Log Verossimilhança* é:

$$\log P(D|\alpha, \beta) = \sum_{d=1}^{M} \log \int_{\theta} \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} \theta_{z_{dn}} \beta_{z_{dn}, w_{dn}} \right) p(\theta|\alpha) d\theta$$

Onde:

- D é o conjunto de documentos.
- α e β são os hiperparâmetros do modelo.
- θ é a distribuição de tópicos para um documento.
- z_{dn} é o tópico atribuído à n-ésima palavra no documento d.
- w_{dn} é a n-ésima palavra no documento d.

Valores mais altos de *Log Verossimilhança* indicam um modelo melhor, pois representam uma probabilidade maior dos dados serem gerados pelo modelo.

3. Desenvolvimento do Trabalho

Neste capítulo, apresentaremos as principais contribuições do projeto.

3.1. Aprendizado não supervisionado

A principal contribuição do projeto foi estabelecer o funcionamento das tarefas de aprendizado não supervisionado no MODEL BUILDER e também consolidar um ambiente de visualização dos resultados mais intuitivo e eficiente, projetando melhorias, ao analisar, a adição de novos componentes visuais e estatísticos que poderiam ser incorporados na interface de execução dos modelos. As atividades concluídas, construíram uma versão estável e funcional do MODEL BUILDER, com a adição de novos modelos para a tarefa de agrupamento e correção de erros de execução.

3.1.1. Analise

No primeiro momento, as atividades do projeto se concentraram na avaliação dos modelos para a tarefa de agrupamento disponíveis nos frameworks *Apache Spark* e *Scikit Learn*. Formam analisados os seguintes algoritmos: *K-means*, *Gaussian mixture model (GMM)*, *Power iteration clustering (PIC)*, *Latent Dirichlet allocation (LDA)*, *Bisecting k-means* e *Density-based spatial clustering of applications with noise(DBSCAN)*. A analise se deu com um estudo detalhado do funcionamento desses algoritmos, avaliando cada aspecto pratico e teórico da sua implementação. Foram analisados varias características, como escalabilidade, capacidade de generalização, capacidade de aprendizado continuo, visualização de resultados, medida de similaridade utilizada e medidas de avaliação e validação de cada procedimento.

3.1.2. Validação

Para cada abordagem, foi realizada uma validação experimental prática, implementando esses algoritmos com o propósito de investigar as vantagens e benefícios de tais abordagens dentro do escopo do projeto. Essa etapa prática permitiu identificar os pontos fortes e as limitações de cada algoritmo, fornecendo uma base sólida para a escolha das técnicas mais adequadas para diferentes cenários de aplicação.

Diante do que foi analisado, o passo seguinte consistiu em validar tais algoritmos dentro do LEMOMADE, avaliando a compatibilidade e limitações de uso dessas abordagens na estrutura e pipeline do MODEL BUILDER. O proposito dessa etapa, foi ter um entendimento amplo das possibilidades de quais novos modelos de aprendizado não

supervisionado poderiam ser implementados no LEMONADE, contribuindo assim, para a expansão das capacidades e funcionalidades da plataforma, permitindo a inclusão de técnicas mais avançadas e diversificadas de *clustering*

3.1.3. Implementação

No âmbito da implementação, os esforços se deram no *backend* do MODEL BUILDER, nos serviços *Tahiti* e *Juicer* (2.4), o foco da implementação foi o desenvolvimento das tarefas de aprendizado não supervisionado para o agrupamento. Para tal, foi utilizado o *framework Apache Spark*, em virtude da sua compatibilidade com a plataforma e alta performance e eficiência processando de grandes volumes de dados. Dado a analise de compatibilidade feita na etapa anterior, foram implementados e adicionado os seguintes algoritmos: *K-means, Gaussian mixture, Latent Dirichlet allocation (LDA), Bisecting k-means*.

A implementação desses algoritmos envolveu algumas etapas para garantir que cada algoritmo fosse adaptado corretamente ao ambiente do LEMONADE e às necessidades dos usuários finais. Toda a implementação foi feita utilizando a linguagem Python e a interface PySpark(*Pyspark* s.d.). Cada algoritmo foi configurado utilizando a biblioteca nativa do Spark para *machine learning*, a MLlib(*Mllib* s.d.). A MLlib fornece várias funcionalidades para criar, treinar e avaliar modelos de aprendizado de máquina em um ambiente distribuído, facilitando a integração de algoritmos de *machine learning* com pipelines de processamento de dados em larga escala.

3.1.4. Algoritmos

Os algoritmos (Figura 11) de *clustering* implementados (*K-means*, *Gaussian mixture model (GMM)*, *Latent Dirichlet allocation (LDA)*, *Bisecting k-means*), foram oriundos da biblioteca MLlib, cada algoritmo foi encapsulado em módulos específicos. O desenvolvimento de tais módulos, incluiu a criação de componentes de software que encapsulam a lógica dos algoritmos de *clustering*, assegurando que cada módulo pudesse ser facilmente integrado ao Juicer.

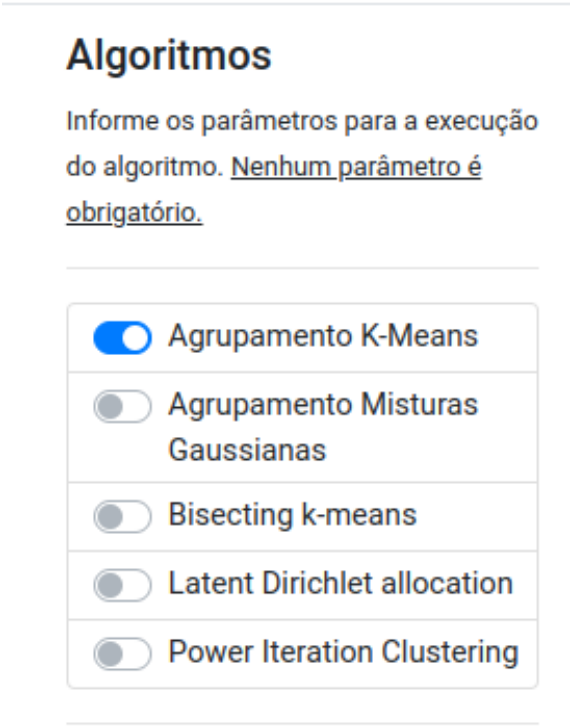

Figura 11. Algoritmos implementados no LEMONADE.

Para criar os campos de interação dos usuários na interface de geração de modelos, foi necessário criar um arquivo de migração, utilizando as bibliotecas Python Alembic(*Alembic* s.d.) e Flask (*Flask* s.d.). Tal arquivo de migração permite aplicar todas as alterações do banco de dados decorrentes do desenvolvimento, possibilitando que os usuários configurem facilmente os parâmetros dos algoritmos na interface do MODEL BUILDER. Dessa forma, com a associação do script de migração às novas operações do MODEL BUILDER, é possível definir de forma intuitiva a associação dos parâmetros na interface, como o número de clusters para K-means (Figura 13) e Bisecting K-means, as variâncias e médias iniciais para o GMM, e os hiperparâmetros para LDA.

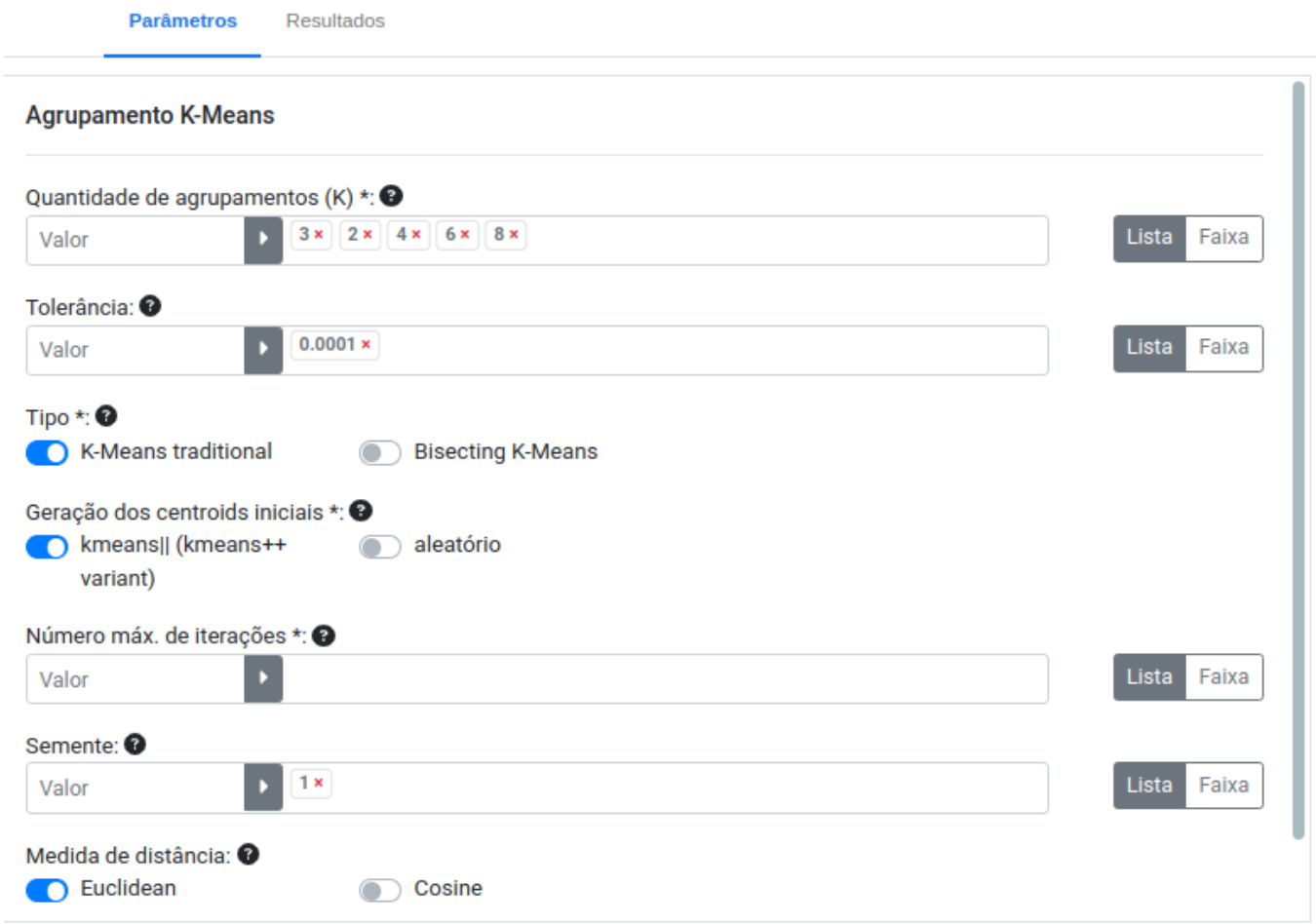

Figura 12. Interface de parametrização do MODEL BUILDER para o K-Means.

Por fim, os algoritmos de agrupamento foram integrados ao sistema de metadados do LEMONADE, permitindo que os usuários selecionem fontes de dados e configurem tarefas de *clustering* com base nos atributos dos dados e finalmente, podendo estes, executarem o treinamento do modelo escolhido, gerando o código da execução e os resultados do experimento.

3.1.5. Execução experimental

Para validar o funcionamento correto dos algoritmos implementados, foi conduzida uma série de execuções experimentais. Esses experimentos tinham como objetivo garantir que cada algoritmo de *clustering* operasse conforme esperado e que os resultados obtidos fossem precisos e consistentes.

Inicialmente, selecionamos um conjunto alguns *datasets* sintéticos e reais, abran-

gendo diferentes características e complexidades. Esses *datasets* incluíam conjuntos de dados bem conhecidos da literatura de aprendizado de máquina, como o *Iris*, *Wine*, e conjuntos de dados textuais para validação do *LDA*.

Para cada algoritmo, foi configurado uma série de parâmetros experimentais, variando desde o número de *clusters* para o *K-means* e *Bisecting K-means*, até o número de componentes Gaussianos para o GMM e o numero de tópicos para o LDA. Essas configurações foram ajustadas para garantir que os algoritmos fossem testados em diferentes cenários operacionais. Para cada experimento executado no LEMONADE, foi avaliado o desempenho dos algoritmos em termos de tempo de processamento, escalabilidade e analise de métricas de qualidade de agrupamento, que no caso foi utilizada a métrica de Indice de Silhueta.

Após a execução dos experimentos, analisamos os resultados obtidos para identificar possíveis problemas ou inconsistências. Verificamos a corretude dos clusters formados, comparando-os com os resultados esperados ou conhecidos da literatura. Para os algoritmos probabilísticos como o *GMM* e o *LDA*, verificamos se as distribuições geradas estavam em conformidade com as propriedades estatísticas esperadas (Figura 13).

Os resultados dos experimentos mostraram que os algoritmos de *clustering* implementados no LEMONADE, funcionam conforme o esperado, mantendo uma alta qualidade nos resultados e a performance operacional.

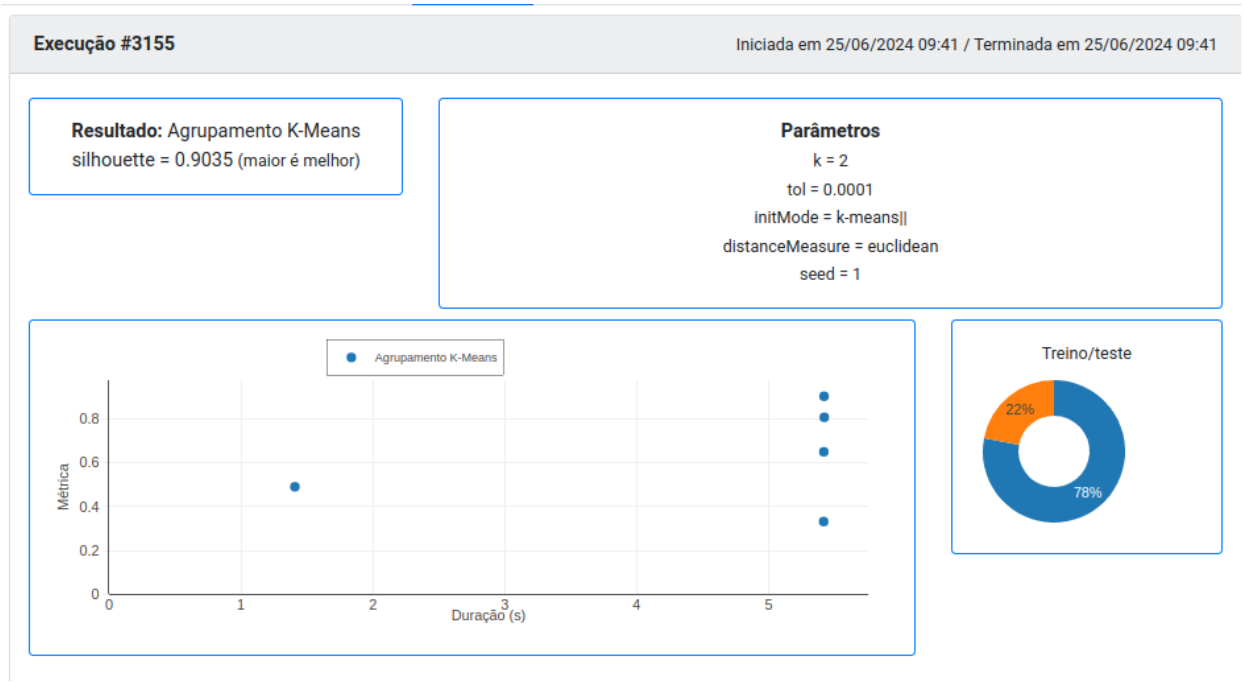

Figura 13. Interface com os resultados, com os resultados obtidos e parâmetros utilizados .

4. Conclusões e Trabalhos Futuros

O trabalho teve por objetivo consolidar o funcionamento do MODEL BUILDER, com a implementação de novos algoritmos de agrupamento e propor melhorias na interface de resultados dos experimentos. Com base nas atividades realizadas e nos resultados obtidos, pode-se certificar que toda abordagem prática, teórica e técnica foi aplicada no desenvolvimento desse projeto, concluindo com êxito assim os objetivos estabelecidos.

Durante o desenvolvimento do trabalho, foi feita um estudo detalhado (3.1.1) das técnicas usadas no aprendizado não supervisionado, analisando o funcionamento dos algoritmos de *clustering*, limitações de execução, cenário de aplicação, métricas de avaliação da qualidade e desempenho de execução e aplicações de analise em contextos práticos. Esse estudo ajudou a ter um entendimento amplo da técnica de agrupamento e como esse método de analise poderia ser incorporado ao LEMONADE.

Na implementação (3.1.3), foi adicionado novos algoritmos de clustering, tornando o MODEL BUILDER e por consequência o LEMONADE, uma ferramenta mais robusta, versátil e consistente, fortalecendo a capacidade da plataforma de suportar uma ampla gama de analises e estudos utilizando para tal o aprendizado de maquina.

A execução e teste experimental (3.1.5) confirmou que os algoritmos operam de

forma eficiente e precisa no ambiente de computação distribuída do LEMONADE, demonstrando a aplicabilidade do aprendizado não supervisionado e robustez e escalabilidade da plataforma ao lidar com grandes volumes de dados.

Para trabalhos futuros, diversas direções podem ser exploradas para ampliar ainda mais as capacidades da plataforma. Um dos objetivos inicias do trabalho era explorar a interface de resultados dos experimentos, porém por limitações de tempo, dificuldades de processos e próprias limitações do *framework* utilizado, optou-se apenas em analisar os componentes que poderiam ser melhorados e quais elementos poderiam ser incorporados a esse interface, com intuído de torna o processo de validação e entendimento dos resultados mais consistente.

Dessa forma, superando essas limitações citadas, como sugestão de melhorias para trabalhos futuros, primeiramente, podemos propor a adição de novos tarefas de aprendizado não supervisionado como modelos de itens sets frequentes e sumarização. Outra proposta de melhoria, seria utilizar outras métricas de avaliação além do Índice de Silhueta, métricas como Indice de Davies-Bouldin e Indice de Calinski-Harabasz que foram analisados no trabalho poderiam ser incorporadas utilizando outro *framework* além do *Apache Spark* que possui poucos métodos implementados de avaliação. O *framerwork scikit learn* poderia proporcionar mais métricas de avaliação para essa tarefa, uma analise da escalabilidade das métricas no scikit learn poderia ser feita para testar a viabilidade.

Outro aspecto importante para futuros desenvolvimentos é trabalhar na melhoria contínua da interface de visualização dos resultados. Na analise feita no projeto, avaliamos que poderia ser incluído mais elementos visuais, como gráficos de dispersão para visualizar efetivamente os *clusters* formados, gráficos de linhas para verificar a eficacia do treinamento da tarefa de agrupamento dado o numero de *clusters* e também adicionar tabelas com alguma informação estatística para um entendimento mais apurado dos resultados. Para o algoritmo LDA, poderia ser incorporado um tipo de visualização interativa, chamada nuvens de palavras, onde é possível verificar as palavras mais citadas e relevantes em um conjunto textual especifico. Para o mesmo algoritmo poderia ser informado na propria interface as tabelas de índices e termos mais importantes, possibilitando o usuário avaliar diretamente a importância relativa de cada termo nos tópico dos documentos.

Por fim, seria interessante fazer uma documentação abrangente para auxiliar os usuários na utilização dos novos elementos da plataforma e também validar os modelos implementados em cenários de aplicações reais. Essas melhorias contínuas garantirão que a plataforma continue a atender às necessidades dos usuários.

Em suma, podemos concluir que o projeto finalizado cumpriu seus propósitos e abre caminho para uma série de aprimoramentos futuros, consolidando o LEMONADE

como uma ferramenta essencial para análise de dados em larga escala e aplicação em aprendizagem de maquina.

Referências

Alembic (s.d.). URL: ⟨https://alembic.sqlalchemy.org/⟩.

Apache Spark (s.d.). URL: ⟨https://spark.apache.org/⟩.

Brown, Tom B. et al. (2020). "Language Models are Few-Shot Learners". Em: *arXiv preprint arXiv:2005.14165*. URL: ⟨https://arxiv.org/abs/2005.14165⟩.

Celebi, M. Emre, Hassan A. Kingravi e Patrick A. Vela (2013). "A comparative study of efficient initialization methods for the k-means clustering algorithm". Em: *Expert Systems with Applications* 40.1, pp. 200–210.

Documentação do Lemonade (s.d.). URL: ⟨https://docs.lemonade.org.br/pt-br/⟩.

Flask (s.d.). URL: ⟨https://flask.palletsprojects.com/en/3.0.x/⟩.

Goodfellow, Ian e Yoshua Bengio (2020). "Generative Adversarial Networks: An Overview". Em: *Communications of the ACM* 63.11. Published: 12 November 2020, pp. 121–130. DOI: ⟨10.1145/3422622⟩. URL: ⟨https://dl.acm.org/doi/10.1145/3422622⟩.

Jiang, Fan, Shuncheng Li e Chuan Zhang (2022). "Deep Clustering: A Comprehensive Survey". Em: *arXiv preprint arXiv:2006.10614*. URL: ⟨https://arxiv.org/abs/2006.10614⟩.

Lin, Frank e William W. Cohen (2010). "Power Iteration Clustering". Em: *International Conference on Machine Learning*. URL: ⟨https://api.semanticscholar.org/CorpusID:861386⟩.

Mllib (s.d.). URL: ⟨https://spark.apache.org/mllib/⟩.

Na, Shi, Liu Xumin e Guan Yong (2010). "Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm". Em: *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, pp. 63–67. DOI: ⟨10.1109/IITSI.2010.74⟩.

Pyspark (s.d.). URL: ⟨https://spark.apache.org/docs/latest/api/python/index.html⟩.

Rohilla, Vinita et al. (2019). "Data Clustering using Bisecting K-Means". Em: *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pp. 80–83. DOI: ⟨10.1109/ICCCIS48478.2019.8974537⟩.

Tufail, Shahid et al. (2023). "Advancements and Challenges in Machine Learning: A Comprehensive Review of Models, Libraries, Applications, and Algorithms". Em: *Electronics* 12.8. Received: 16 February 2023 / Revised: 29 March 2023 / Accepted: 5 April 2023 / Published: 10 April 2023, p. 1789. DOI: ⟨10.3390/electronics12081789⟩. URL: ⟨https://www.mdpi.com/2079-9292/12/8/1789⟩.

Wan, Huan et al. (2019). "A Novel Gaussian Mixture Model for Classification". Em: *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pp. 3298–3303. DOI: ⟨10.1109/SMC.2019.8914215⟩.