Definição de Parâmetros do Spark por meio de Aprendizado de Máquina: um Estudo com *Dataflows* de Astronomia

Luís Felipe Oliveira, Cristina Boeres, Daniel de Oliveira

[1] Instituto de Computação – Universidade Federal Fluminense (IC/UFF)

lfxaoliveira@id.uff.br, {boeres,danielcmo}@ic.uff.br

Resumo. *O Apache Spark tem se mostrado um framework promissor para auxiliar na execução de experimentos científicos baseados em simulação e que demandam execuções em ambientes de alto desempenho. Entretanto, o Spark possui mais de 180 parâmetros para serem configurados, o que torna a tarefa de configuração entediante e propensa a erros. O presente artigo explora a utilização de múltiplos métodos de aprendizado de máquina para auxiliar na configuração dos parâmetros do Spark. Os modelos foram treinados a partir de um dataset com dados de proveniência de mais de 500 execuções de dataflows de astronomia. Os resultados mostraram que o uso de aprendizado de máquina nesse contexto é promissor. Além disso, os resultados mostraram que a estratégia de partição dos dados de entrada é o atributo que que tem maior relevância na obtenção de menores tempos de execução e que as Redes Neurais Artificiais são o método que traz os melhores resultados.*

Abstract. *Apache Spark shows to be a promising framework to aid in the execution of simulation-based scientific experiments that demand execution in high-performance environments. However, it has more than 180 parameters to be configured, making its configuration a tedious task and prone to errors, if performed manually. This paper explores the use of multiple machine learning methods to help configure Spark parameters. Such models were trained on the Orange platform and later incorporated into the SpaCE tool, proposed in a previous work. The models were trained from a dataset with provenance data from over 500 astronomy dataflow executions. The results show that the use of machine learning methods in this context is promising and that the dataflow input data partitioning strategy is the attribute that has greater relevance in obtaining shorter execution times, as well as the fact that Artificial Neural Networks are the machine learning method that brings the best results.*

1. Introdução

Nas últimas décadas, o mundo tem enfrentado o que chamamos de dilúvio de dados [Hey et al. 2009], sendo que, em especial, na área científica, múltiplos experimentos têm gerado um volume de dados sem precedentes [de Oliveira et al. 2019]. Muitos desses experimentos são executados por meio de complexas simulações computacionais (*i.e.*, experimentos *in silico*). Embora sejam modelados em sua grande maioria como *workflows* científicos ou *scripts*, o uso de *frameworks Big Data* tem ganhado popularidade nos últimos anos para executar os *dataflows* associados a esses experimentos [de Oliveira et al. 2019].

Esses *frameworks* se apresentam como uma solução interessante devido principalmente à possibilidade de modelagem das etapas de um experimento (*i.e.* atividades) de uma

forma simples, sendo que o próprio *framework* se encarrega de coordenar a execução paralela. Existem diversos *frameworks* como o Hadoop, o Storm e o Spark [Zaharia et al. 2010]. Esse último se tornou o padrão *de fato* para processamento de grandes volumes de dados, já que apresenta um bom desempenho em múltiplos cenários. Uma das grandes vantagens do Spark é a possibilidade do armazenamento dos dados em memória principal, o que leva a um desempenho de execução ordens de grandeza melhor quando comparado a outros *frameworks* [Gottin et al. 2018].

Embora o Spark represente um avanço, a configuração de seus parâmetros ainda é um desafio [Herodotou et al. 2011]. O Spark possui mais de 180 parâmetros que podem/-devem ser configurados para otimizar o desempenho (ou mesmo tornar a execução viável), considerando cada aplicação específica. Tais parâmetros podem ser configurados manualmente, por meio de tentativa e erro, mas este é um processo entediante e pouco eficiente [Wang et al. 2016]. Em um trabalho anterior [de Oliveira et al. 2021] utilizamos técnicas de aprendizado de máquina para criar um modelo preditivo que fosse capaz de definir as combinações de valores de parâmetros do Spark, em conjunto com parâmetros específicos do domínio da aplicação, que levassem aos menores tempos de execução. Tal modelo foi treinado a partir de um *dataset* de dados de proveniência [Freire et al. 2008] com dados de mais de 500 execuções de experimentos, e acoplado a uma ferramenta chamada SpaCE [de Oliveira et al. 2021] (ferramenta para recomendação de parâmetros no Spark). Para que o modelo fosse interpretável no SpaCE, foram utilizadas árvores de decisão [Han et al. 2011]. Porém, outros métodos podem ser utilizados nesse contexto, mesmo que o modelo gerado não seja interpretável.

Nesse contexto, o artigo é explora diferentes métodos de aprendizado de máquina (*i.e*, redes neurais artificiais, KNN, máquina de vetor de suporte e Naive Bayes) para configurar os parâmetros do Apache Spark. Esses métodos devem ser capazes de "rotular" uma determinada combinação de parâmetros informada de acordo com seu tempo de execução. Para comparar os modelos gerados, utilizamos como estudo de caso três experimentos da área da astronomia: O SkyMap, o *Constellation Queries* e o Montage. O artigo se organiza em 5 seções além da introdução. Na Seção 2 é apresentado o referencial teórico. Na Seção 3 definimos formalmente o problema tratado nesse artigo. Na Seção 4 discutimos os resultados obtidos. Na seção 5 são apresentados os trabalhos relacionados, e finalmente, a Seção 6 apresenta as conclusões desse artigo.

2. Referencial Teórico

2.1. Apache Spark

O Spark [Zaharia et al. 2010] é um *framework* para processamento paralelo de aplicações em ambientes de Computação Escalável Intensiva de Dados (DISC), sendo bastante interessante para a execução de aplicações que reutilizam dados durante a execução. O Spark faz uso da memória para armazenar dados intermediários das atividades do *dataflow* e ainda, o HDFS (*Hadoop Distributed File System*) pode ser utilizado para armazenar dados de entrada/saída. O Spark provê mais operações que o Hadoop (*e.g.*, *filter*, *join*, *groupBy*), e essa variedade de operações permite uma melhor modelagem dos experimentos. A arquitetura do Spark consiste em um programa *Driver* e um conjunto de executores distribuídos pelos nós do *cluster*.

Seguindo o paradigma mestre/escravo, o *Driver* executa sobre o nó principal e é responsável pela gerência da execução da aplicação. Cada *executor*, processa as tarefas nos nós

do *cluster*, devendo ser configurado com o número de núcleos computacionais para executá-las e a quantidade de memória disponível. A inicialização destes processos no *cluster* é da responsabilidade do gerenciador do *cluster*. Nesse artigo, os experimentos foram realizados utilizando o YARN - um gerenciador de pacotes para aplicar comandos prontos ao código de uma aplicação - sendo que, as duas principais características de Spark e YARN consideradas são CPU e memória. Embora os recursos de E/S, *e.g.* disco e rede, também tenham impacto no desempenho da aplicação no Spark, estes recursos não foram considerados. Assumimos que Spark e YARN executam em um *cluster* onde a latência da rede não tem impacto no desempenho do experimento. Desta forma, apenas foram avaliados os parâmetros do Spark relacionados com CPU e Memória: (i) Número de Executores, (ii) Número de Núcleos por *Executor*, (iii) Quantidade de Memória por *Executor* e (iv) Número Máximo de Tarefas por *Executor*.

2.2. Conceitos de Aprendizado de Máquina

Nos últimos anos observou-se um aumento na quantidade de dados gerados e disponibilizados em diversos seguimentos da sociedade. Nesse contexto, surge a necessidade de desenvolver métodos que sejam capazes de utilizar tais dados que representam experiências passadas, para predizer eventos futuros. Segundo [Han et al. 2011], esses métodos podem ser classificados em preditivos ou descritivos. O primeiro caso tem por objetivo encontrar uma função, na qual, a partir dos dados de treinamento, seja capaz de prever um rótulo ou valor que caracterize um novo exemplo. Já nos métodos descritivos, o objetivo é explorar ou descrever um conjunto de dados e, por exemplo, encontrar grupos de objetos semelhantes. Nesse artigo estamos interessados em modelos preditivos, em especial os que realizam a tarefa de classificação. Problemas de classificação são um importante campo da área de ciência de dados. À partir de um conjunto de treinamento com dados rotulados, um modelo é treinado e utilizado para predizer uma nova observação considerando valores de seus atributos e incluí-la em uma das classes existentes [Han et al. 2011]. A seguir apresentamos resumidamente os métodos preditivos considerados nesse artigo.

O método de classificação com base nos k-vizinhos mais próximos (KNN) é considerado bastante simples apesar da sua robustez, consistindo na rotulação de um novo dado com o mesmo rótulo de seus k vizinhos mais próximos. Nesse caso, a proximidade é medida por meio da distância euclidiana em um espaço n-dimensional [Han et al. 2011]. O KNN utiliza o método indutivo para classificação de objetos, no qual objetos com características semelhantes pertencem ao mesmo grupo. O classificador Naive Bayes se aplica a situações em que cada tarefa x pode ser descrita como um conjunto de atribuição de valores no qual o objetivo da função $f(x)$ pode assumir qualquer valor de um conjunto finito V. O método Naive Bayes se vale de uma função de distribuição de probabilidades para estimar a classificação de um determinado objeto. Nele é considerada a hipótese de que os valores dos atributos são independentes de sua classe e todas as probabilidades necessárias à obtenção do classificador são computadas a partir dos dados de treinamento.

Uma Árvore de Decisão usa a estratégia de dividir um problema complexo em problemas menores e mais simples. Então as soluções dos problemas menores são compiladas para formar a solução do problema original [Han et al. 2011]. O método se divide em duas fases: construção e poda. A construção é realizada a partir da divisão recursiva do conjunto de treinamento com base em um critério até que a maioria, ou todas, as tuplas possuam um rótulo de classe. A fase seguinte consiste na poda, usada para melhorar a generalização do algoritmo, e onde são "podados" os vetores responsáveis pela classificação de poucos registros

[Han et al. 2011]. As Redes Neurais Artificiais (RNAs) têm como fonte de inspiração o modelo inteligente de comportamento do cérebro humano. Dessa forma sua composição se dá a partir de unidades de processamento simples e interconectadas onde são processadas funções matemáticas – essas unidades são conhecidas como neurônios artificiais [Han et al. 2011]. As RNAs se caracterizam por dois aspectos básicos: a arquitetura, que está relacionada ao tipo e a quantidade de unidades de processamento e ao aprendizado, que se refere aos ajustes dos parâmetros, ou seja, a definição dos pesos associados as conexões de rede que fazem com que o modelo obtenha o melhor desempenho [Han et al. 2011]. [Han et al. 2011] argumentam que Máquina de Vetores de Suporte (SVM do inglês *Support Vector Machine*) vem recebendo bastante atenção nos últimos anos em tarefas de reconhecimento de padrões. O embasamento das SVMs vem da Teoria do Aprendizado Estatístico pela qual são estabelecidos princípios que norteiam a obtenção de classificadores com boa capacidade de generalização [Han et al. 2011]. Dentre as características que justificam a utilização do método se destacam: a boa capacidade de generalização – uma vez que o método é capaz de ter bons resultados com dados que não pertençam ao seu conjunto de treinamento; a robustez em objetos de grandes dimensões; a convexidade da função objetivo, ou seja nas SVM a função quadrática é otimizada e há apenas um mínimo global; e por fim, uma teoria bem definida baseada em matemática e estatística [Han et al. 2011].

3. Definição do Problema

O problema é formalmente definido em relação aos aspectos relevantes relacionados com a estrutura do *dataflow*, dados de entrada, parâmetros de configuração e o ambiente DISC, seguindo o formalismo de [de Oliveira et al. 2021]. Nesse artigo, um *dataflow* é modelado como um grafo acíclico dirigido (DAG) $W = (A, Dep)$, onde A são as atividades representadas pelos vértices, e Dep o conjunto de dependências de dados entre as atividades. Dada uma atividade $a_i \in A$, assumamos $D(a_i)$ como o conjunto de dados de entrada e $O(a_i)$ como o conjunto de dados de saída de a_i. A dependência entre as atividades a_i, a_j é denotada por $(a_i, a_j) \in Dep \leftrightarrow \exists\ d_k \in D(a_j) \mid d_k \in O(a_i)$. Como cada atividade a_i pode ser executada em paralelo, ela pode ser decomposta em uma série de tarefas. Assim, o conjunto $T = \{t_1, ..., t_n\}$ de tarefas é criado para representar a execução paralela das atividades de A. Note que, uma atividade a_i pode ser executada múltiplas vezes e cada t_j processa um subconjunto dos dados de entrada.

Um dado *dataflow* $W = (A, Dep)$ de entrada deve ser executado em um ambiente distribuído composto por um conjunto de máquinas $R = \{r_1, \ldots, r_k\}$. Para tal execução, o conjunto de valores de parâmetros $S(W, D, R) = \{pv_1, pv_2, ..., pv_m\}$ devem ser especificados, sendo cada valor pv_i um dos parâmetros considerados nesse artigo: número de núcleos por executor; número de executores; quantidade de memória por executor; e tipo de particionamento dos dados. Esse último parâmetro é específico de domínio. Dado o *dataflow* W, um conjunto de dados de entrada D e um conjunto de máquinas R, o objetivo desse artigo é encontrar a parametrização $S^*(W, D, R)$ de forma que o tempo de execução de W seja o menor possível. Formalmente, $\exists\ S^*(W, D, R)$ de forma que $ET(W, D, R, S^*(W, D, R)) = \min\limits_{\forall S(W,D,R)} ET(W, D, R, S(W, D, R))$, onde $ET(W, D, R, S(W, D, R))$ é o tempo de execução do *dataflow* W consumindo D e executando em R com os valores de parâmetros $S(W, D, R)$. É importante mencionar que o valor $S^*(W, D, R)$ é obtido por meio do modelo preditivo a ser treinado.

4. Avaliação Experimental

O *dataset* sobre o qual foi realizado o experimento contém o histórico de execução (proveniência) de três *dataflows* da astronomia: o SkyMap, o *Constellation Queries* e o Montage. O SkyMap visa produzir um histograma de uma região do céu investigada. A localização de um planeta, estrela e outros elementos no céu é baseada em um sistema de coordenadas. Este sistema consiste em uma grade bidimensional, com posições indicadas pelos atributos Ascensão Reta (RA) e Declinação (DEC). O SkyMap produz o histograma do céu pintando cada um dos objetos celestes no catálogo em uma representação $2D$ da região celeste de interesse. O processo é realizado modelando um *dataflow* com 3 atividades (*LoadData, SkyMap e SkyMapAdd*). O *dataflow Constellation Queries* tem como objetivo encontrar o padrão cruzado de Einstein no catálogo de constelações, que é especificado por um conjunto de pontos (A, B, C e D) e seus atributos. Portanto, o catálogo é processado para encontrar conjuntos de pontos cuja forma é semelhante à da amostra (A, B, C e D). Os dados a serem processados devem ser particionados seguindo as estratégias *Equi-Depth* ou Hierárquica. Em [Porto et al. 2017], o particionamento é descrito em mais detalhes. O *Constellation Queries* é um *dataflow* com três atividades (*Load Data, Matching e Collect*). O Montage [Berriman et al. 2003] é um *dataflow* bem conhecido que monta imagens astronômicas em mosaicos utilizando arquivos do tipo FITS (*Flexible Image Transport System*). Estes arquivos incluem um sistema de coordenadas e o tamanho da imagem, rotação, e projeção do mapa WCS (*World Coordinate System*). O Montage é composto por 9 atividades (*ListFITS, Projection, SelectProjections, CreateUncorrectedMosaic, CalculateOverlap, ExtractDifferences, CalculateDifferences, FitPlane*, e *CreateMosaic*). Nas execuções consideradas no *dataset* seguimos a mesma configuração utilizada em [Silva et al. 2016]. Esses *dataflows* foram selecionados de modo a apurar a qualidade do modelo preditivo em cada um dos métodos selecionados, visto que apresentam características distintas tanto do ponto de vista de complexidade quanto do volume de dados e a necessidade de recursos computacionais.

O *dataset* utilizado contém 500 execuções dos *dataflows* supracitados e é composto por nove atributos, sendo um deles o tempo de execução, que é o que se pretende predizer. O tempo de execução é um atributo meta, e foi discretizado em três faixas: alto, médio e baixo. O objetivo é verificar se uma determinada configuração de parâmetros vai gerar uma execução com tempo de execução baixo. Os outros 7 atributos são: (i) Tipo de Particionamento: define a estratégia usada para particionar os dados. Nesse experimento foram definidas duas estratégias, a *Equi-Depth* e a randômica, (ii) Número de Partições: define o número de partições criadas para os dados de entrada do *dataflow*, (iii) Número de Executores: define quantos processos são criados e alocados para executar a aplicação no Spark, e seu valor pode variar de 1 a 200, (iv) Número de Núcleos por *Executor*: controla o número de núcleos computacionais disponíveis para cada *executor* durante a execução da aplicação. Esse parâmetro pode variar de 1 a 32, (v) Quantidade de Memória por *Executor*: define a quantidade máxima de memória disponível para cada *executor*, variando entre 1 e 64 GB, (vi) Número Máximo de Tarefas: define a quantidade máxima de tarefas a ser executada em paralelo por cada *executor* no Spark, (vii) Tamanho dos Dados de Entrada: esse atributo define o tamanho dos dados de entrada do *dataflow*, variando de 1 a 24 GB. A escolha dos parâmetros se deu em linha com o trabalho anterior [de Oliveira et al. 2021].

Os *workflows* foram implementados em Python 2.7 e foram executados em um *cluster* equipado com Intel Xeon E5-2630 V3 2.4GHz, RAM 96GB e CentOS 6.7 (64 bits). Todo o treinamento e avaliação dos modelos treinados foram realizados na ferramenta Orange

[Demšar et al. 2004]. O Orange permite desenvolver *scripts* Python para prototipar novos algoritmos e também se mostra indicado para usuários menos experientes que podem se valer de uma interface visual de utilização simples. Utilizamos os seguintes métodos no experimento: KNN, Naive Bayes, Redes Neurais Artificiais e Máquina de Vetor de Suporte (SVM). Os resultados foram comparados com o *baseline* (Árvores de Decisão) utilizado anteriormente em [de Oliveira et al. 2021]. A primeira análise realizada foi verificar quais atributos oferecem o maior ganho de informação. O ganho de informação é calculado comparando a entropia do *dataset* antes e depois de uma transformação. A Tabela 1 apresenta o ganho de informação para cada um dos atributos do *dataset*. Podemos perceber que o tipo de particionamento é o atributo que traz maior ganho de informação, seguido do Número Máximo de Tarefas e Número de Partições. Isso se dá uma vez que a utilização do programa *Spatial Catalog FRAgmeNter* (FRANCE) para gerar particionamento *Equi-Depth* sempre gerou tempos de execução baixos ou médios. O FRANCE é uma aplicação iterativa que particiona os dados em histogramas de mesma profundidade [de Oliveira et al. 2021]. Além disso, o número máximo de tarefas define a concorrência em um mesmo *executor*.

Tabela 1. Análise do Ganho de Informação.

Atributo	**Ganho de Informação**
Tipo de Particionamento	0,730
Número Máximo de Tarefas	0,606
Número de Partições	0,606
Número de Executores	0,118
Quantidade de Memória por *Executor*	0,118
Número de Núcleos por *Executor*	0,067
Tamanho do *Dataset*	0,000

Para avaliar os modelos gerados, utilizamos o método de validação cruzada *holdout*, que consiste em dividir o conjunto total de dados em dois subconjuntos mutuamente exclusivos, um para treinamento e outro para teste (validação). A proporção utilizada foi $2/3$ dos dados para treinamento e o $1/3$ restante para teste. Em relação às métricas de desempenho dos classificadores, utilizamos: Precisão, *Recall*, Acurácia e a medida F1. A Tabela 2 apresenta os valores das métricas para cada um dos classificadores avaliados. A *Precisão* mostra proporção de execuções classificadas corretamente em um determinado grupo considerando todos que foram classificados naquele grupo. No experimento realizado, as Redes Neurais Artificiais apresentaram o melhor resultado em relação à precisão (0,981). Já o Naive Bayes obteve o pior desempenho (0,866) nessa métrica. O *Recall* calcula a taxa que o método classifica a execução como pertencente a um determinado grupo (baixa, média ou alta) em relação a todos que de fato pertencem ao grupo em questão. De acordo com os resultados do experimento as Redes Neurais Artificiais obtiveram o melhor desempenho, seguido pelo KNN, Árvores de Decisão, SVM e Naive Bayes. A *Acurácia* mede a taxa com que o classificador classifica como baixas, as execuções que de fato tiveram tempos de execução baixos em relação a todo o *dataset*. No experimento realizado, as Redes Neurais Artificiais apresentaram também a melhor acurácia. Finalmente, a *métrica F1* combina precisão e *recall* de modo a trazer um número único que indique a qualidade geral do modelo treinado. Essa métrica é interessante pois trabalha bem com conjuntos de dados que possuem classes desproporcionais, o que acontece nesse *dataset*, uma vez que temos mais execuções com tempo de execução alto em comparação ao total de execuções com tempos médios e baixos. No caso da métrica F1, as Redes Neurais Artificiais também apresentaram o melhor resultado.

Nota-se que os resultados para todos os modelos foram bastante próximos, isso se deve ao *dataset* ser comportado (e possuir poucos *outliers*) e ter um número relativamente reduzido de instâncias (< 1.000). Além disso, é importante salientar que ao utilizarmos redes neurais, perdemos a capacidade de interpretação dos resultados, se comparado com as árvores de decisão. Dessa forma, o uso de árvores de decisão dentro do SpaCE ainda se mostra como uma alternativa interessante.

Tabela 2. Avaliação dos modelos treinados

Modelo	Acurácia	F1	Precisão	*Recall*
Redes Neurais Artificiais	0,981	0,981	0,981	0,981
KNN	0,979	0,979	0,979	0,979
Árvore de Decisão	0,977	0,977	0,977	0,977
SVM	0,961	0,961	0,961	0,961
Naive Bayes	0,866	0,868	0,896	0,866

5. Trabalhos Relacionados

Algumas abordagens já propõem a otimização de parâmetros no Apache Spark [Yigitbasi et al. 2013, Herodotou et al. 2011, Wang et al. 2016]. Estas abordagens podem ser classificadas como "*white-box*" ou "*black-box*". [Herodotou et al. 2011] propõem um método baseado no custo de otimização "*white-box*". Este método requer um conhecimento profundo do funcionamento interno do sistema, *e.g.,* como interagir com a máquina virtual Java (JVM). Assim, torna-se difícil ser utilizado por usuários que não estejam familiarizados com tais detalhes (*e.g.* astrônomos). Por outro lado, [Yigitbasi et al. 2013] e [Wang et al. 2016] propõem abordagens "*black-box*"para este problema de otimização. [Yigitbasi et al. 2013] usa a Regressão Vetorial de Apoio (SVR) para construir o modelo preditivo de configuração de parâmetros para o Hadoop. [Wang et al. 2016] propõem um método para otimizar aplicações Spark por meio de algoritmos e técnicas de aprendizado de máquina. Embora estas abordagens "*black-box*" representem um avanço, consideram apenas os parâmetros padrão do Spark sem otimizar os parâmetros específicos de domínio (*e.g.* o particionamento de dados astronômicos). Ainda, há na literatura trabalhos que buscam investigar abordagens existentes no ajuste de parâmetros para sistemas de processamento de dados em lote e *stream*, [Herodotou et al. 2020] propõem uma taxonomia com as abordagens pesquisadas além de sumarizar os prós e contras das mesmas.

6. Conclusões e Trabalhos Futuros

O Spark tem sido considerado um *framework* promissor para gerenciar a execução de experimentos em paralelo em ambientes de alto desempenho. A vantagem do Spark é que ele esconde do usuário a complexidade da execução paralela da aplicação. Entretanto, o Spark possui mais de 180 parâmetros para serem configurados, o que torna a tarefa de configuração complexa. O presente artigo mostrou a aplicação de métodos de aprendizado de máquina em um *dataset* de dados de proveniência coletados a partir de múltiplas execuções de de 3 *dataflows* de domínio da astronomia processados pelo Spark, com o objetivo de definir os atributos tanto do Spark quanto da aplicação para obter tempos menores de processamento. A análise experimental mostrou que a estratégia de partição dos dados de entrada do *dataflow* é o atributo que tem o maior peso para obter menores tempos de processamento, sendo a aplicação *Spatial Catalog FRAgmeNter* (FRANCE) responsável por tais resultados eficientes. Ainda, foi apresentado o resultado dos classificadores KNN, Naive Bayes, Árvores de

Decisão, SVM e Redes Neurais Artificiais em relação às métricas de desempenho: Acurácia, *Recall*, F1 e Precisão, onde observou-se que o método das Redes Neurais Artificiais apresenta os melhores resultados para o *dataset* utilizado nesse artigo. Como trabalho futuro sugere-se estender o experimento fazendo o uso de novos classificadores, bem como utilizar outras estratégias de amostragem e de particionamento dos dados de entrada dos *dataflows*.

Referências

Berriman, G., Good, J., Laity, A., Bergou, A., Jacob, J., Katz, D., Deelman, E., Kesselman, C., Singh, G., Su, M.-H., and Williams, R. (2003). Montage: A grid enabled image mosaic service for the national virtual observatory. In *ADASS XIII*.

de Oliveira, D. C. M., Liu, J., and Pacitti, E. (2019). *Data-Intensive Workflow Management: For Clouds and Data-Intensive and Scalable Computing Environments*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers.

de Oliveira, D. E. M., Porto, F., Boeres, C., and de Oliveira, D. (2021). Towards optimizing the execution of spark scientific workflows using machine learning-based parameter tuning. *Concurr. Comput. Pract. Exp.*, 33(5).

Demšar, J., Zupan, B., Leban, G., and Curk, T. (2004). Orange: From experimental machine learning to interactive data mining. In *ECML PKDD*, pages 537–539. Springer.

Freire, J., Koop, D., Santos, E., and Silva, C. T. (2008). Provenance for computational tasks: A survey. *Comput. Sci. Eng.*, 10(3):11–21.

Gottin, V., Pacheco, E., Dias, J., Ciarlini, A., Costa, B., Vieira, W., Souto, Y. M., Pires, P., Porto, F., and Rittmeyer, J. G. (2018). Automatic caching decision for scientific dataflow execution in apache spark. In *Proc. of BeyondMR*.

Han, J., Kamber, M., and Pei, J. (2011). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition.

Herodotou, H., Chen, Y., and Lu, J. (2020). A survey on automatic parameter tuning for big data processing systems. *ACM Computing Surveys (CSUR)*, 53(2):1–37.

Herodotou, H., Lim, H., Luo, G., Borisov, N., Dong, L., Cetin, F. B., and Babu, S. (2011). Starfish: A self-tuning system for big data analytics. In *CIDR 11*, pages 261–272.

Hey, T., Tansley, S., and Tolle, K., editors (2009). *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, Redmond, Washington.

Porto, F., Khatibi, A., Nobre, J. R., Ogasawara, E. S., Valduriez, P., and Shasha, D. E. (2017). Constellation queries over big data. *CoRR*, abs/1703.02638.

Silva, V., de Oliveira, D., Valduriez, P., and Mattoso, M. (2016). Analyzing related raw data files through dataflows. *Concurrency and Computation: Practice and Experience*, 28(8):2528–2545.

Wang, G., Xu, J., and He, B. (2016). A novel method for tuning configuration parameters of spark based on machine learning. In *IEEE HPCC/SmartCity/DSS*, pages 586–593.

Yigitbasi, N., Willke, T., Liao, G., and Epema, D. (2013). Towards machine learning-based auto-tuning of MapReduce. In *IEEE MASCOTS*, pages 11–20.

Zaharia, M., Chowdhury, M., Franklin, M., Shenker, S., and Stoica, I. (2010). Spark: Cluster Computing with Working Sets. In *Proc. of the HotCloud'10*, pages 10–10, Berkeley, CA, USA. USENIX Association.