Predição de Falhas em *Workflows* Científicos em Nuvens baseada em Aprendizado de Máquina

Daniel Pinheiro da Silva Junior [1]*, **Aline Paes**[1], **Daniel de Oliveira**[1]

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

danieljunior@id.uff.br, {alinepaes,danielcmo}@ic.uff.br

Resumo. *Os cientistas cada vez mais têm se apoiado em ferramentas computacionais para executar e analisar experimentos científicos, almejando reduzir esforços e custos para comprovar ou refutar hipóteses. Entretanto, recursos podem ser desperdiçados se os parâmetros usados nas aplicações fizerem com que a execução do experimento falhe. Assim, para diminuir a quantidade de execuções que resultam em falha, este artigo propõe a integração de uma técnica de Aprendizado de Máquina com um Sistema de Gerência de Workflows Científicos para induzir um modelo preditivo de falhas, a partir de dados de proveniência. Resultados experimentais mostram que o modelo é capaz de identificar corretamente casos de falha no Workflow Científico SciPhy.*

1. Introdução

Os cientistas de áreas como a biologia e a astronomia executam experimentos científicos a fim de comprovar ou refutar suas hipóteses. Muitos desses experimentos são modelados como *Workflows* Científicos [Mattoso et al. 2010]. Um *workflow* é composto por atividades que possuem dependências de dados entre si, onde cada atividade invoca programas de computador. Os *workflows* científicos são gerenciados por Sistemas de Gerência de *Workflows* Científicos (SGWfC). Existem diversos SGWfCs como o Swift/T [Wozniak et al. 2013] e o SciCumulus [de Oliveira et al. 2010], e a maioria deles, além e gerenciar a execução, capturam dados históricos do *workflow*, chamados de dados de proveniência [Freire et al. 2008].

Um mesmo *workflow* pode ser executado diversas vezes, variando-se parâmetros e dados de entrada a fim de confirmar uma hipótese. Em muitos cenários, os cientistas não sabem quais são os valores e parâmetros que produzem resultados satisfatórios, logo, os mesmos tem que explorar uma gama de valores, similar a um processo de tentativa e erro. A grande dificuldade desse processo é que algumas combinações de parâmetros podem gerar erros de execução ou não produzir resultados úteis. Em ambos os casos, tais problemas podem comprometer todo o experimento. Esse cenário torna-se ainda mais complexo se considerarmos que muitos dos *workflows* são executados em ambientes de processamento de alto desempenho que possuem custos associados, como as nuvens públicas (*e.g.* Amazon AWS). Neste caso, tais erros podem gerar além de atrasos na execução (que já são indesejáveis), prejuízo financeiro. Dessa forma, se torna fundamental que a execução dos *workflows* seja minimamente impactada por falhas que sejam produtos de valores de parâmetros e dados de entrada mal definidos pelos cientistas.

*Este artigo foi financiado parcialmente pelo CNPq, FAPERJ e CAPES

Entretanto, não é trivial identificar *a priori* quais dados ou valores de parâmetros podem gerar erros de execução do *workflow*. Mesmo com todo o histórico de execuções passadas de *workflows* disponível (*i.e.* dados de proveniência) ainda é uma tarefa árdua e propensa a erros identificar manualmente padrões ou extrair regras que possam ser utilizados pelos SGWfC para prever falhas e informar ao usuário que um determinado valor de parâmetro tem chance de gerar erros de execução. Acreditamos que algoritmos de Aprendizado de Máquina [Mitchell 1997] podem ser adaptados e aplicados a este tipo de problema, sendo capazes de induzir modelos preditivos a partir de dados de proveniência.

Assim, nesse artigo, propomos um componente que aplica o método de aprendizado de máquina chamado de Máquinas de Vetores de Suporte [Cortes and Vapnik 1995] (SVM), para induzir um modelo que seja capaz de prever uma falha no *workflow*, a partir dos dados de proveniência que incluem características dos programas e dados da infraestrutura utilizada na execução. Adicionalmente, como é possível que falhas no *workflow* não tenham sido registradas no banco de dados de proveniência (*i.e.* nenhuma execução passada apresentou falhas de atividades), propomos o uso de SVM tanto para diferenciar os casos de sucesso e os de falha, como para criar um modelo que seja capaz de detectar um comportamento anômalo no *workflow*, a partir apenas de casos de sucesso fornecidos como exemplo (SVM *One Class*). O componente desenvolvido foi acoplado ao SGWfC SciCumulus de forma que o próprio SGWfC fosse capaz de verificar quais combinações de parâmetros podem gerar erro ou não, de forma a ajustar a execução para evitar falhas. Resultados experimentais com o modelo induzido para o *workflow* SciPhy [Ocaña et al. 2011] mostram que a abordagem proposta é capaz de identificar dados e parâmetros de entrada que produzem falhas na execução do *workflow* de forma satisfatória.

O artigo está organizado em 4 seções além da introdução. A Seção 2 revisão de conceitos e trabalhos relacionados. A Seção 3 apresenta a abordagem proposta, incluindo os componentes de coleta de dados, aprendizado do SVM e visualização dos resultados. A Seção 4 apresenta os resultados experimentais obtidos a partir do uso do componente. Finalmente, a seção 5 conclui o trabalho e apresenta direções para trabalhos futuros.

2. Referencial Teórico e Trabalhos Relacionados

2.1. Aprendizado de Máquina e Máquinas de Vetores de Suporte

A maneira convencional que usamos o computador é por meio de aplicações desenvolvidas com algum paradigma de programação onde as instruções do que o computador deve fazer são explícitas. A área de *Aprendizado de Máquina* [Mitchell 1997], na tentativa de resolver problemas que são difíceis de serem programados de forma explícita, procura generalizar uma informação, induzir um modelo, ou construir uma hipótese por meio de exemplos ou fatos.

Algoritmos de Aprendizado de Máquina necessitam de dados de entrada para serem executados. A informação utilizada como entrada é na forma de uma base de dados composta por um conjunto de *exemplos* (no contexto desse artigo, os exemplos são os dados de proveniência). Ou seja, considerando que a base de dados é uma matriz N x M, Cada linha $0 < i \leq N$ representa um exemplo, e cada coluna $0 < j \leq M$ é um *atributo* do i-ésimo exemplo, ou seja cada exemplo é descrito por um conjunto de atributos.

O aprendizado pode ser *supervisionado*, em que o conjunto de exemplos possui tanto as entradas como as saídas esperadas para uma tarefa, ou *não-supervisionado*, onde apenas as entradas são fornecidas para o algoritmo. Dois exemplos de classes de problemas no aprendizado supervisionado são: *Regressão*, quando o valor de resposta do exemplo é contínuo, e a *Classificação*, quando o valor de resposta para o exemplo é um valor discreto. Existem diversos algoritmos de Aprendizado de Máquina. Nesse artigo, focamos no algoritmo que aprende *Vetores de Suporte*, chamado de *Support Vector Machines(SVM)* [Cortes and Vapnik 1995, Steinwart and Christmann 2008], que podem ser usados tanto para Classificação quanto para Regressão.

Sob a perspectiva da Classificação, a ideia do SVM é encontrar o hiperplano que separa os dados das classes do domínio, com a maior margem de distância entre os exemplos [Cortes and Vapnik 1995]. O SVM possui 3 componentes principais: *Vetores de Suporte, Vetor de pesos e Kernel*. Os Vetores de Suporte são um subconjunto do conjunto de exemplos que são utilizados como a base de criação dos vetores ou curvas que separam as classes, denominados *fronteira de decisão*. O Vetor de pesos é o grau de contribuição que cada atributo dos vetores de suporte tem na classificação. O *Kernel* é uma função que tem como objetivo mapear um dado espaço em outro, usado principalmente em casos onde os dados não são linearmente separáveis. Assim, o hiperplano que divide os dados pode ser representado pela Equação $f(x) = w \cdot x + b = 0$ (1), onde o termo *w*, é referente ao vetor de pesos para cada atributo dos vetores de suporte. O termo *x* é um vetor de atributos de cada instância do conjunto de exemplos e o *b* é o viés que função tem, no caso da reta, geometricamente interpretado como o coeficiente angular.

One-Class Support Vector Machines Existem casos onde executar a tarefa de Classificação da forma que vimos até o momento para determinados tipos de problemas é inviável a partir dos algoritmos tradicionais de construção de hipóteses, visto que existem poucos ou nenhum exemplo para representar uma classe (*e.g.* a falta de falhas em *workflows* na base de proveniência citada anteriormente). Neste cenário, podem ser usados algoritmos de *a Detecção de Anomalias*, que diferem levemente de abordagens para classificação. Um dos modelos propostos sob esta perspectiva é o *One-Class Support Vector Machines* [Amer et al. 2013], que é uma variação dos algoritmos de SVM, e pode ser visto como uma abordagem de Aprendizado Não-Supervisionado, pois é pressuposta a existência de dados de apenas uma classe não explicitada nos dados. O *One-Class* SVM utiliza o mesmo conceito de *Kernel* de SVMs tradicionais, para transformar o espaço original dos dados de treinamento em um outro espaço. Porém, essa transformação não tem a ver com a separação dos dados entre si, uma vez que todos pertencem a mesma classe. A fronteira de decisão do *One-Class* é encontrada a partir do hiperplano que está mais distante da origem, de modo que entre a origem e o hiperplano não existam ocorrências de registros. Essa restrição pode ser relaxada introduzindo variáveis de folga e conseguindo uma margem maior de separação.

2.2. Trabalhos Relacionados

A maioria dos trabalhos relacionados ao uso de técnicas de Aprendizado de Máquina para apoiar os usuários de SGWfCs está na composição dos *workflows*. As tarefas focam em recomendar, na maior parte das vezes, qual sequência de atividades escolher [Yeo and Abidi 2013]. Outros poucos trabalhos focam em utilizar algoritmos de aprendizado de máquina para predizer falhas em execuções.

[Câmara et al. 2015] propõe uma abordagem para predição de falhas em *workflows* que utiliza árvores de decisão [Mitchell 1997]. Entretanto, árvores de decisão usualmente produzem classificadores inferiores aos SVMs [Mitchell 1997]. Em [Bala and Chana 2015] Bala desenvolveu um trabalho semelhante, no entanto usou técnicas de aprendizado diferentes: Naive Bayes, Regressão Logística, Redes Neurais Artificiais e Random Forest. Em [Si et al. 2016], é proposto um algoritmo de predição temporal de exceções onde as predições são baseados no estado de execução do workflow.

Utilizando uma abordagem Não-Supervisionada, usando o algoritmo de clusterização *K-Means*, Samak *et al.* em [Samak et al. 2011], propõem que os atributos necessários para a verificação de semelhança do *workflow* atual com algum dos *clusters* previamente extraídos por meio de dados históricos sejam recomputados, de acordo com a produção de dados de monitoramento das etapas de execução do *workflow*.

Gaikwad desenvolveu um trabalho que envolve a Detecção de Anomalias, porém voltado para a infraestrutura do ambiente de execução do *workflow*. Em [Gaikwad et al. 2016], procura-se detectar a degradação de desempenho de execução e falhas de execução dos *workflows* em um ambiente de nuvem. Para a detecção, foi desenvolvido um algoritmo baseado em modelos Autoregressivos, porém tal método não funciona bem se a maioria dos exemplos é positivo ou negativo.

3. Um Componente para Predição de Falhas em *Workflows* Científicos

Neste artigo, implementamos um componente chamado *PredFail* que foi acoplado ao SGWfC SciCumulus para prever falhas e execuções anômalas em *workflows* científicos, visando economizar recursos que poderiam ser desperdiçados em tais situações. Tal componente é baseado no aprendizado de modelos por meio de SVMs e se apoia no grande volume de dados de proveniência gerado pelo SciCumulus em suas execuções. Os módulos do componente, bem como suas integrações, são apresentados na Figura 3.

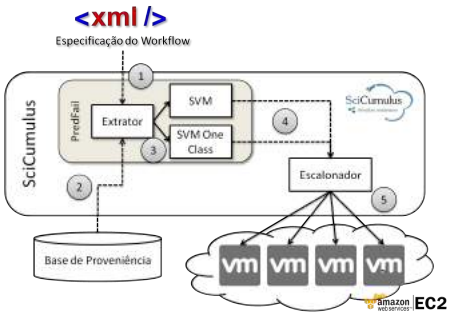

Figura 1. Arquitetura proposta - *PredFail* acoplado ao SGWfC SciCumulus. As setas pontilhadas representam transferências de dados e as setas cheias indicam a ordem de execução.

Uma vez que a especificação do *workflow* é submetida ao SciCumulus, o SGWfC envia a mesma ao *PredFail* antes mesmo de iniciar sua execução (etapa 1 na Figura 1). O *PredFail* analisa quais são as combinações de parâmetros utilizadas e qual o *workflow* a ser executado e importa do banco de dados de proveniência (etapa 2) as informações relativas a execuções passadas do mesmo *workflow*. Esses dados são convertidos no formato

CSV e enviados aos algoritmos de SVM (etapa 3). Então, os algoritmos de SVM podem construir o modelo (de detecção de falhas e de anomalia) a partir dos dados extraídos. Finalmente, a partir dos modelos construídos, o *PredFail* pode realizar as predições com os valores de parâmetros informados. Se algum valor de parâmetro for classificado como que vai produzir erros, essa execução é abortada. Somente combinações que não produzirão erros é que são enviadas ao escalonador (etapa 4) para serem efetivamente executados na nuvem (etapa 5). O *PredFail* foi implementado em Python, usando o padrão *Model-View-Controller* (MVC). A seguir apresentamos cada módulo do *PredFail* com mais detalhes.

3.1. Extração da Base de Dados a partir do Banco de Dados de Proveniência

Para extrair os dados de proveniência do Banco de Dados gerenciado pelo SciCumulus, de forma que eles estejam em um formato manuseável por um algoritmo de Aprendizado de Máquina, são executadas uma série de consultas SQL. É permitido que o usuário personalize as consultas SQL que o *PredFail* executa, de forma que mudanças no *schema* do banco possam ser facilmente absorvidas.

A consulta SQL utilizada nos experimentos desse artigo extrai dados relativos ao *id* de atividades, *id* da máquina virtual onde a atividade foi executada, a quantidade de processadores que esta máquina possui e o *exitstatus* da execução da atividade, representando o sucesso ou falha da execução da mesma. A variável de classe é sempre considerada como a última coluna extraída pela consulta SQL, *i.e.* se a atividade falhou ou não. Caso haja um erro na consulta SQL, o componente não permite que o usuário continue o processo e envia todas as combinações de parâmetros para o SciCumulus, mesmo as que podem produzir falhas. O *backup* da base de proveniência do SciCumulus e as consultas SQL utilizadas nos experimentos podem ser obtidos em https://github.com/UFFeScience/PredFail.

3.2. Detecção de Falhas com Máquinas de Vetores de Suporte

Com a base de dados devidamente formatada, o componente constrói os modelos de predição de falhas e detecção de anomalias com SVMs. Para tanto, o componente é integrado à Biblioteca de Aprendizado de Máquina SciKit-Learn [Pedregosa et al. 2011].

Podem existir casos onde os dados de proveniência das execuções dos *workflows* possua apenas dados de execuções de sucesso. Nestes casos, modelos supervisionados de classificação não são capazes de serem treinados, devido à existência de dados de apenas uma classe. No contexto deste artigo, foi usado o *SVM One-Class* para lidar com este tipo de situação. Nesse caso, um conjunto de valores de atributos que fogem aos padrões descobertos pelo algoritmo será considerado um caso de falha.

Quanto aos dados, a única diferença da predição de falhas com SVM e da detecção de anomalias com o *SVM One-Class*, é que o primeiro requer uma variável de classe. Ambos os algoritmos implementados na biblioteca SciKit requerem que dados categóricos sejam codificados em numéricos, e para tanto usamos a codificação *One-Hot* [Coates and Ng 2011], que transforma cada possível valor de um atributo categórico em binário.

4. Avaliação Experimental

A fim de avaliar experimentalmente o componente proposto, foram utilizados os dados de proveniência coletados pelo SGWfC SciCumulus durante execuções do *workflow* ci-

Tabela 1. Resultados: SVM

kernel	grau	Média Acurácia	Média Recall	Média F1-Score	Média Precisão
linear	-	0.43	0.52	0.11	0.07
poly	1	0.53	0.37	0.09	0.06
poly	2	0.62	0.25	0.07	0.04
poly	3	0.76	0.32	0.15	0.11
rbf	-	0.94	0.0	0.0	0.0
sigmoid	-	0.94	0.0	0.0	0.0

Tabela 2. Resultados: SVM One-Class

kernek	grau	Média Acurácia	Média Recall	Média F1-Score	Média Precisão
linear	-	0.5	0.68	0.16	0.09
poly	1	0.52	0.68	0.16	0.09
poly	2	0.52	0.68	0.16	0.09
poly	3	0.52	0.68	0.16	0.09
rbf	-	0.68	0.1	0.01	0.0
sigmoid	-	0.94	0.0	0.0	0.0

entífico SciPhy [Ocaña et al. 2011] no ambiente de nuvem da Amazon AWS. O SciPhy é um *workflow* científico da área da bioinformática desenvolvido para construir árvores filogenéticas a partir de sequências de DNA, RNA e aminoácidos. Para mais informações, favor consultar Ocaña *et al*.

Metodologia Experimental A consulta para extração dos dados em uma base de 150 execuções do SciPhy retornou 400 tuplas, sendo 376 exemplos de sucesso de execução e 24 exemplos de falha. Para executar o *SVM One-class* apenas os dados de sucesso foram utilizados durante o treinamento, mas todos os dados foram utilizados para fins de teste. Para o treinamento, foi utilizado um computador com processador *Celeron(R) Dual-Core CPU T3300 @ 2.00GHz x 2*, memória de *4GB DDR2*, e taxa de transferência de *300 MBps*, e SO Linux-Ubuntu. Para a obtenção dos resultados quantitativos, foi utilizada a abordagem *Stratified 10-Fold Cross Validation*[Refaeilzadeh et al. 2009], onde os exemplos são divididos em conjuntos, mas obedecendo a distribuição original das classes. A cada iteração, $k - 1$ conjuntos são usados para treinamento e o conjunto restante é usado para validação dos resultados.

Resultados do Classificador SVM Para a predição de falhas com o SVM foram experimentados os *kernels* (1) Sigmoid, (2) Linear, (3) Polinomial e (4) *Radial Basis Function* (RBF). No terceiro caso, é necessário definir um parâmetro *degree*, que é o grau do polinômio usado para a fronteira de decisão do modelo. No quarto caso, que assume uma distribuição Gaussiana para os exemplos, é necessário definir o parâmetro *gamma*, que indica o quanto um único exemplo afeta o modelo. Este parâmetro será fixo com o valor padrão *auto*. Todos os *kernels* requerem o parâmetro *C*, responsável por analisar se o modelo está sendo aprimorado a cada iteração, e que será fixado com o valor 1. A Tabela 1 apresenta as médias para os conjuntos de validação das métricas *Acurácia, Recall, F1-Score* e *Precisão* [Han et al. 2012], obtidas a partir da execução do componente configurado com diferentes parâmetros do classificador SVM. Em uma primeira análise, pode parecer que os *kernels rbf* e *sigmoid* apresentam os melhores resultados no que diz respeito a acurácia dos modelos (94%). No entanto, as demais métricas mostram que, devido ao desbalanceamento do conjunto de treinamento, o modelo construído a partir de todos os *kernels* acabou por favorecer a classe majoritária (casos de execuções de sucesso). Assim, podemos concluir que o modelo construído com o *kernel polinomial* e *degree* = 3, foi o que obteve o melhor resultado, pois junto à *acurácia* de 76%, observou-se também um *recall* de 32%.

Resultados do Detector de Anomalias One-Class O Detector de Anomalias *One-Class*, possui muitos parâmetros similares ao SVM, exceto pelo parâmetro C. Por outro lado, no *SVM One-class* existe o parâmetro $\backslash\sqcap$, fixado aqui $0,5$, e que representa um limite superior para o erro de treinamento e para a quantidade de vetores de suporte, ambos em fração. A Tabela 2 apresenta as mesmas métricas da Tabela 1. No entanto, para o modelo *SVM One-Class* o conjunto de treinamento utilizado continha apenas dados de execuções de sucesso, e a porção de dados de testes usada no processo de validação cruzada foram retirados do mesmo conjunto de dados que o SVM tradicional, contendo casos de sucesso e de falha de execuções, onde a falha é a classe positiva. Os modelos induzidos com o *kernel linear* e *polinomial* obtiveram resultados semelhantes, tendo uma acurácia média baixa. Porém, esses modelos foram os que retornaram os melhores valores de *Recall*, isto é, obtiveram melhor capacidade de generalizar o caso de falha, mesmo a partir de um número reduzido de exemplos nesta classe. Observamos ainda uma melhora nos resultados de *recall* em relação ao SVM tradicional, o que indica que o modelo consegue detectar casos anômalos com sucesso.

5. Conclusões e Trabalhos Futuros

Workflows Científicos e SGWfCs incrementaram a produtividade dos cientistas em suas áreas de pesquisa, devido ao apoio à gerência de execuções e armazenamento de dados. Contudo, com os experimentos cada vez mais complexos, a escolha de parâmetros a serem usados na execução de um *workflow* ainda é uma tarefa árdua em muitos casos, agravada pelo fato de que uma escolha incorreta pode resultar em desperdício de recursos.

O componente proposto neste artigo, chamado de *PredFail*, apoia a execução de *workflows*, ao utilizar Aprendizado de Máquina para induzir um modelo preditivo de falhas. Os modelos induzidos pelos dados de proveniência, que podem conter ou não os casos de falhas, mostraram que é possível prever se o conjunto de parâmetros utilizados resulta ou não em uma execução com falha no *workflow*. Mesmo com uma base de treinamento desbalanceada, o SVM e o *SVM One-Class* foram capazes de obter resultados significativos, particularmente em termos do *recall* obtido pelo último. *PredFail* foi projetado para operar em dados de proveniência armazenados em uma base de dados relacional, permitindo que ele possa ser utilizado também com dados que não foram gerados pelo SciCumulus.

Futuramente, os modelos induzidos pelo *PredFail* serão adaptados automaticamente, caso surjam novos casos de falhas. Além disso, as falhas serão discriminadas em tipos mais específicos e outros tipos de classificadores também serão utilizados.

Referências

Amer, M., Goldstein, M., and Abdennadher, S. (2013). Enhancing one-class support vector machines for unsupervised anomaly detection. *Proc. of the ACM SIGKDD Workshop on Outlier Detection and Description*, pages 8–15.

Bala, A. and Chana, I. (2015). Intelligent failure prediction models for scientific workflows. *Expert Systems with Applications*, 42(3):980 – 989.

Coates, A. and Ng, A. Y. (2011). The importance of encoding versus training with sparse coding and vector quantization. In *Proc. of the 28th Int. Conf. in Machine Learning*, pages 921–928.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Câmara, R. V., Paes, A., and de Oliveira, D. (2015). Aplicação de Árvores de Decisão para Recomendação de Parâmetros em Workflows Científicos. *Brazilian e-Science Workshop*.

de Oliveira, D., Ogasawara, E., Baião, F., and Mattoso, M. (2010). SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows. *Proceedings of the 3rd IEEE Int. Conf. on Cloud Computing*.

Freire, J., Koop, D., Santos, E., and Silva, C. T. (2008). Provenance for Computational Tasks: A Survey. *Computing in Science & Engineering*, pages 20–30.

Gaikwad, P., Mandal, A., Ruth, P., Juve, G., Król, D., and Deelman, E. (2016). Anomaly Detection for Scientific Workflow Applications on Networked Clouds. *International Conference on High Performance Computing & Simulation (HPCS)*.

Han, J., Kamber, M., and Pei, J. (2012). *Data Mining: Concepts and Techniques, Third Edition*. Elsevier.

Mattoso, M., Werner, C., Travassos, G. H., Braganholo, V., Murta, L., Ogasawara, E., de Oliveira, D., da Cruz, S. M. S., and Martinho, W. (2010). Towards Supporting the Life Cycle of Large-scale Scientific Experiments. *Int. Journal of Business Process Integration and Management*, pages 79–92.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Science/Engineering/Math.

Ocaña, K., de Oliveira, D., Ogasawara, E., Davila, A., Lima, A., and Mattoso, M. (2011). SciPhy: A Cloud-based Scientific Workflow for Phylogenetic Analysis of Drug Targets in Protozoan Species. *Brazilian Simposium of Bioinformatics*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Refaeilzadeh, P., Tang, L., and Liu, H. (2009). *Cross-Validation*, pages 532–538. Springer US, Boston, MA.

Samak, T., Gunter, D., Goode, M., Deelman, E., Juve, G., Mehta, G., Silva, F., and Vahi, K. (2011). Online Fault and Anomaly Detection for Large-Scale Scientific Workflows. *IEEE Int. Conf. on High Performance Computing and Communications*.

Si, Y.-W., Hoi, K.-K., Biuk-Aghai, R. P., Fong, S., and Zhang, D. (2016). Run-based exception prediction for workflows. *Journal of Systems and Software*, 113:59 – 75.

Steinwart, I. and Christmann, A. (2008). *Support vector machines*. Springer Science & Business Media.

Wozniak, J. M., Armstrong, T. G., Wilde, M., Katz, D. S., Lusk, E., and Foster, I. T. (2013). Swift/t: Scalable data flow programming for many-task applications. *SIGPLAN Not.*, 48(8):309–310.

Yeo, P. and Abidi, S. S. (2013). Dataflow oriented similarity matching for scientific workflows. *IEEE 27th International*, page 2091–2100.