Um Arcabouço Flexível para Integração de Análise Preditiva e Prescritiva, com Atuação

Flexible Framework for Predictive and Prescriptive Analytics with Actuation

Marcos de Aguiar
Universidade Federal da Bahia
Departamento de Ciência da Computação
Salvador, Bahia
marcos.deaguiar@gmail.com

Fabíola Greve
Universidade Federal da Bahia
Departamento de Ciência da Computação
Salvador, Bahia
fabiola@dcc.ufba.br

Genaro Costa
Universidade Federal da Bahia
Instituto de Humanidades, Artes e Ciências
Salvador, Bahia
genaro@dcc.ufba.br

RESUMO

A Internet das Coisas (IoT), aliada a diversas outras tendências, tem proporcionado uma geração massiva de dados. Todo esse volume de dados gera oportunidades para extração de conhecimento e agregação de valor. Nesse contexto, a integração da análise de dados preditiva com a análise prescritiva pode auxiliar usuários e a Indústria a serem mais produtivos e bem sucedidos. Esse artigo apresenta um arcabouço genérico para fazer predição, prescrição e atuação, permitindo com que o desenvolvedor integre esses conceitos e facilidades em seus experimentos. O arcabouço apresenta arquitetura de micro serviços flexível e escalável, promove eficácia, tolerância a falhas e exibe bom desempenho em cenários de IoT e computação em nuvem. Uma implementação da proposta é apresentada e resultados significativos são obtidos.

Palavras-Chave

Análise Preditiva, Análise Prescritiva, Machine Learning, Internet das Coisas

ABSTRACT

The Internet of Things (IoT), coupled with other trends, is being generating large amounts of data. All this data volume bring opportunities for knowledge extraction and value creation. The concept of integrating predictive and prescriptive may help the industry and users to be more productive and successful. This work presents a generic framework for prediction, prescription and actuation, enabling developers to easily integrate these concepts and functionalities in their experiments. The framework has a flexible, micro services based, scalable architecture, that provides efficiency, fault tolerance and displays a good performance in IoT and cloud computing scenarios. An implementation of the proposal is presented and significant results are obtained.

SBSI 2017 June 5^{th} – 8^{th}, 2017, Lavras, Minas Gerais, Brazil

CCS Concepts

•Computing methodologies → Supervised learning by classification; •Applied computing → Service-oriented architectures; Event-driven architectures;

Keywords

Predictive Analytics, Prescriptive Analytics, Machine Learning, Internet of Things

1. INTRODUÇÃO

A Internet das Coisas (IoT) tem mostrado uma forte tendência a ser a próxima grande onda de tecnologia; alguns autores afirmam que em breve mais dados serão gerados de dispositivos do que por humanos [1]. Esse cenário, aliado ao uso em larga escala já estabelecido da Internet, está gerando uma quantidade massiva de dados, de tal forma que alguns já estão considerando esses dados como o petróleo do Século vinte um. A oportunidade está em, não somente utilizar esses dados para propósitos operacionais e de serviço, mas, sobretudo para extrair conhecimento e valor dos mesmos.

A Indústria dos negócios já está se beneficiando da análise de dados para extração de conhecimento, a fim de tirar proveito das novas oportunidades. O conhecimento específico, fundamentado em dados, permite que as empresas foquem onde existe a maior probabilidade de benefícios. Inúmeros são os exemplos, de consumidores que tendem a comprar sempre um mesmo conjunto de produtos (que podem ser colocados em estantes próximas), a sistemas de recomendação na web; essas técnicas são bem conhecidas e constantemente refinadas.

Muito mais desafiador (e compensador) é a utilização de dados para análise preditiva. Esse tipo de análise possui diversas aplicações: de empresas que detectam tendências antes do tempo e se preparam para obter vantagens e benefícios, a sistemas que detectam defeitos em máquinas antes que algum dano seja causado. A lista de projetos que aplicam esses princípios é grande, seguem alguns deles: (i) a Ford está desenvolvendo um carro que dectecta se o motorista está intoxicado e dirigindo de forma perigosa e gera alarme [8]; (ii) a Target desenvolveu um sistema que detecta se as clientes estão possívelmente grávidas e enviam catálogos de produtos especializados para as mesmas; (iii) muitas empresas utilizam análise preditiva para ver quais consumidores tendem a responder melhor a certos anúncios, a fim de

economizar em custos de impressão e envio, maximizando o retorno do investimento [8].

O passo seguinte é a análise prescritiva, que consiste em um modelo que utiliza o resultado da predição para fazer uma recomendação, tomar uma decisão, ou atuar automaticamente no mundo real. Modelos que utilizam análise preditiva e emitem ordem de compra e venda de ações para maximizar lucros ou minimizar risco são um exemplo de análise prescritiva.

Esse artigo apresenta o projeto e implementação de um arcabouço (*framework*) que permite que pesquisadores possam rodar experimentos, utilizando análise preditiva e prescritiva, com atuação em tempo real. Ao nosso conhecimento, diferentemente de outras propostas na literatura, o arcabouço não só incorpora modelos preditivos, mas permite que o usuário forneça um modelo prescritivo no qual a resposta pode se tornar uma atuação em tempo real.

O arcabouço utiliza modelos baseados em aprendizagem de máquina e é genérico suficiente para integrar-se com sistemas de software já existentes, através de protocolos já bem estabelecidos na Indústria. Adicionalmente a esta flexibilidade, sua arquitetura promove eficácia, tolerância a falhas e bom desempenho. Apesar de o usuário poder executar diretamente o sistema, bastando estender algumas poucas classes, toda arquitetura foi feita para que outras tecnologias possam ser incorporadas, a depender da infra estrutura que o usuário possua. Um experimento teste, a partir de simulações, foi realizado, demonstrando com sucesso a viabilidade da proposta, inclusive para cenários com alta carga de dados, como os da IoT e da computação em nuvem.

O restante do artigo é organizado nas seguintes seções: a Seção 2 apresenta trabalhos relacionados; a Seção 3 exibe os conceitos utilizados no arcabouço; a Seção 4 apresenta a solução desenvolvida; a Seção 5 apresenta um experimento com o sistema e os resultados obtidos; a Seção 6 apresenta as conclusões e trabalhos futuros.

2. TRABALHOS RELACIONADOS

Barber and Sharkey [2] utilizam análise preditiva para identificar estudantes com maior risco de abandono de curso, exibindo bons resultados, o que torna uma boa ferramenta de suporte e decisão. Diferentemente da nossa proposta, o cenário do trabalho não é de sistema em tempo real.

Fulop et al. [5] apresentam um arcabouço genérico que combina processamento de eventos complexos e análise preditiva, realizando um experimento de prova de conceito que mostra a sinergia entre essas duas tecnologias. Esse projeto pode servir de base para outros sistemas, porém o experimento demonstrado utiliza análise preditiva para prever os próprios eventos, e não para a tomada de decisão. O trabalho apresentado também não é de tempo real, possui uma taxa de atualização de vinte e cinco minutos, e os autores também não mostraram nenhuma evidência de escalabilidade e de funcionamento interno da solução.

Huang et al. [7] utilizam predição para suporte à decisão de redes complexas. O modelo é utilizado para criar gráficos que ajudam humanos a ajustar redes de distribuição de energia em caso de falha, de forma a minimizar os impactos na rede. O trabalho apresenta um ótimo caso para análise preditiva, mas não possui um modelo de prescrição e nem atuação em tempo real.

Tönjes et al. [9] projetaram um arcabouço genérico para cidades inteligentes, que processa um fluxo de dados vindo de dispositivos conectados via Internet das Coisas (IoT), faz análises sobre esses dados, e provém uma interface de programação para serviços que precisam dessa informação. O autores citam alguns projetos em andamento que utilizarão essa plataforma, para viabilizar melhor transporte, segurança pública, e inovações que melhorem a qualidade e vida dos cidadãos.

3. CONCEITOS, CONSTRUÇÃO E VALIDAÇÃO DE MODELOS

A especificação do modelo é um passo importante para atingir bons resultados. As escolhas nessa fase devem ser feitas por alguém com bom conhecimento em ciência dos dados, para tomar decisões corretas em relação à metodologia e validação do trabalho.

Primeiramente, é necessário decidir exatamente o que o modelo irá predizer; pode ser algum objetivo de negócio específico, como, quais usuarios estão em risco de cancelar um serviço, quais empréstimos não irão ser pagos. O modelo preditivo é escolhido a depender da pergunta sendo feita e dos dados disponíveis para responder. Normalmente, os modelos são agrupados nas seguintes categorias [11]:

- Classificação: Consiste em rotular resultados. Um exemplo seria classificar estudantes por sua probabilidade de abandonar um curso; as classificações poderiam ser: “Baixo Risco”, “Médio Risco”, “Alto Risco”;
- Pontuação: São tarefas que possuem um valor numérico e contínuo como saída. Um exemplo é a predição do preço de uma casa, baseado em localização, quantidade de quartos e área em metros quadrados. O algoritmo aprende o relacionamento entre as variáveis de entrada e pode predizer a saída para novas instâncias de dados. Normalmente, modelos baseados em regressão linear são usados para essas tarefas;
- Agrupamento (Clustering): Conhecido também como modelo não supervisionado, são algoritmos que agrupam os dados através de algum critério de similaridade. É bastante utilizado para encontrar padrões ainda desconhecidos em grupos de dados, na fase de análise exploratória.

O modelo define o que será respondido ao usuário, já o algoritmo define como essas questões serão respondidas. Para cada tipo de modelo pode existir mais de um algoritmo de treinamento; normalmente, o mesmo tipo de problema pode ser resolvido por diversos algoritmos, e o cientista de dados deve testar e ajustar para decidir qual algoritmo ou combinação de algoritmos são mais adequados para construção do modelo. Apesar de existir diferentes tipos de modelos e algoritmos, o fluxo de trabalho que deve ser seguido na construção de um modelo é sempre o mesmo. As etapas são:

1. Aquisição e limpeza dos dados que serão utilizados para treinar e validar o modelo;
2. Treinamento de modelos, e escolha do melhor;
3. Ajuste do modelo, removendo variáveis desnecessárias;
4. Validação do modelo ajustado.

A validação na fase quatro é de extrema importância, é mandatório treinar e testar o modelo com dados reais para garantir que o mesmo tenha o desempenho esperado quando entrar em produção.

3.1 Dados de Treinamento, Teste e Validação

Quando um modelo é construído, os dados são o componente mais importante, o mesmo deve ser uma boa representação do fenômeno que o cientista quer fazer a predição. Se os dados são imprecisos e inconsistentes, fica bastante difícil para o modelo convergir, e ainda que isso aconteça o desempenho não será satisfatório no ambiente de produção. Uma vez que o conjunto de dados é adquirido para treinar o modelo, o mesmo deve ser particionado em duas ou três partes, a depender do caso e da quantidade de dados disponíveis. Essas partições são:

- Dados de treinamento: O conjunto de dados que será utilizado para treinar o modelo. Pode ser reutilizado diversas vezes, até o modelo ficar bem ajustado;
- Dados de teste: Deverá ser usado somente uma vez; serve como uma simulação do ambiente de produção, e dados reais devem ser utilizados;
- Dados de validação (opcional): Pode ser utilizado como teste de sanidade para revalidar o modelo. Em alguns casos, quando o modelo não teve um bom desempenho com os dados de teste, pode servir como um segundo caso de teste quando o modelo é corrigido;

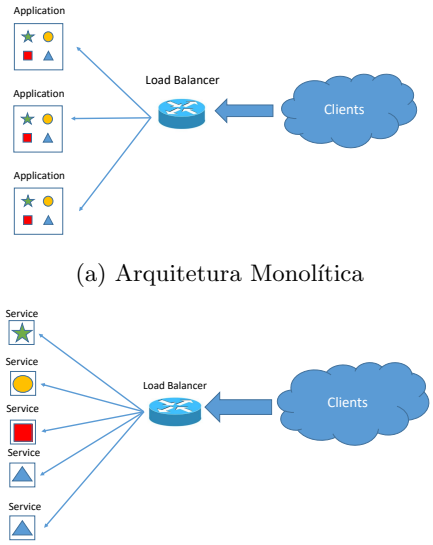
Normalmente, a opção de ter um conjunto de dados de validação, depende da quantidade de dados disponíves para criação do modelo. Se não existir muitos dados disponíveis, a maior parte deve ser utilizada para treinar o modelo e garantir uma eficiência aceitável. Quanto maior a quantidade de dados, mais bem ajustado será o modelo.

3.2 Arquitetura de Micro Serviços

Micro serviços é uma metodologia de arquitetura de software que tem como sua pedra fundamental o conceito de quebrar uma aplicação em diversos serviços menores e independentes ao invés de grandes aplicações monolíticas [4]. Em arquitetura de aplicações convencionais, uma aplicação possui todos os serviços embutidos no seu pacote de implantação; assim, quando a aplicação precisa ser replicada para aumentar disponibilidade de serviço, todos os serviços são replicados juntos, tendo cópias completas da aplicação em cada instância, como pode ser visto na Figura 1(a).

No caso de micro serviços, partes da aplicação podem ser replicadas de forma independente, de acordo com a demanda para um determinado serviço, como pode ser visto na Figura 1(b). Essa estratégia permite um uso mais eficiente e especializado de recursos, ideal para cenários de computação em nuvem. Existem também outras vantagens em utilizar essa metodologia arquitetural, a saber:

- Cada micro serviço pode ter seu próprio ciclo de desenvolvimento e publicação independente, não precisando aguardar o ciclo completo de testes do resto da aplicação;
- Os serviços podem ser desenvolvidos utilizando liguagens de programação e plataformas diversas, provendo mais flexibilidade aos desenvolvedores que podem utilizar a tecnologia que melhor se adequa ao problema;



(a) Arquitetura Monolítica

(b) Arquitetura de Micro Serviços

Figura 1: Padrões de Arquitetura.

- É transparente ao usuário final;
- Times diferentes podem ser responsáveis por cada serviço, trabalhando de forma independente, com um conjunto de habilidades específicas.

4. ARQUITETURA DO SISTEMA

A contrução de modelos e seu uso é a parte preditiva da solução. A outra parte é a estratégia de análise prescritiva; ou seja, é a parte de atuação, em reação a um fluxo de dados, que é feita em tempo real. O sistema de atuação consulta o modelo preditivo e prescritivo para a tomada de decisão. Alguns exemplos de aplicação para esse conceito são:

- Monitoramento de atividades esportivas em tempo real, que se baseiam em modelos treinados e podem recomendar limites seguros de esforço para os praticantes, prevenindo, por exemplo, ataques cardíacos;
- Casas inteligentes, que aprendem os hábitos dos moradores e emitem um alarme caso algo fora do padrão aconteça, sugerindo, por exemplo, uma possível invasão;
- Sistemas de compra e venda de ativos, que podem analisar fluxo de dados em tempo real e emitir ordens de compra e venda que podem ser configuradas para maximizar lucros ou minimizar riscos;
- Sistemas que podem detectar padrões de tráfego de rede, e ajustar as configurações para o melhor uso de recursos e manutenção de qualidade de serviço.

A Figura 2 apresenta a macro arquitetura do sistema. Os clientes acessam a aplicação através de uma interface REST (*Representational State Transfer*) e o conteúdo da mensagem é codificado no formato JSON (*JavaScript Object Notation*). Para a atuação uma requisição REST pode ser feita ativamente pelo sistema, e para notificações assíncronas, o

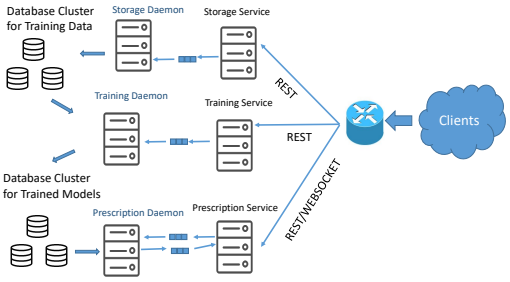


Figura 2: Macro Arquitetura do Sistema.

protocolo *websocket* pode ser utilizado, mantendo uma conexão aberta com o cliente. Como pode ser visto na Figura 2, a arquitetura possui três camadas, a saber:

- *Front-end*: Servidores web utilizando REST e *websocket* para interagir com aplicações clientes;
- *Back-end*: Serviços de processamento que implementam toda a lógica da aplicação;
- Sistema de armazenamento: O sistema assume que existem dois sistemas de armazenamento, um para os dados de treinamento e outro para salvar o modelo gerado.

O *front-end* e *back-end* são conectados através de fila de mensagens que possuem diferentes padrões de comunicação. O arcabouço também utiliza o conceito de micro serviços, assim cada serviço pode ser replicado de forma independente para aumentar a disponibilidade e qualidade de serviço. Se o gargalo estiver no serviço de processamento e o servidor web não estiver congestionado, mais processos "trabalhadores" podem ser replicados. Para o código cliente, existem três serviços: armazenamento de dados de treinamento, serviço de treinamento, e o serviço de prescrição.

O uso de fila de mensagens também possui o benefício de tolerância a falhas. Como pode ser visto na Figura 3, se um processo de *back-end* falha durante o processamento de uma mensagem, a mesma é entregue a outro processo, e é reprocessada sem que o cliente perceba. Em um contexto de computação em nuvem, é esperado que processos e servidores falhem. Para melhora de desempenho do sistema de armazenamento, o *framework* conta com as soluções já disponíveis. Assim, a implementação padrão vem com suporte ao banco de dados Cassandra[1] and `\alignauthor` para os dados de treinamento, e Redis[2] para salvar os modelos. No entanto, essas implementações podem ser facilmente trocadas para suportar outras plataformas.

Essa abordagem flexível, que permite que o desenvolvedor expanda o arcabouço de forma a suportar outras tecnologias, é adotada em todas as camadas do sistema como será visto nas subseções a seguir.

4.1 Serviço de Armazenamento

O serviço de armazenamento é responsável por armazenar os dados de treinamento para uma aplicação. É importante mencionar que a arquitetura desse módulo serve de base para

[1]http://cassandra.apache.org
[2]https://redis.io

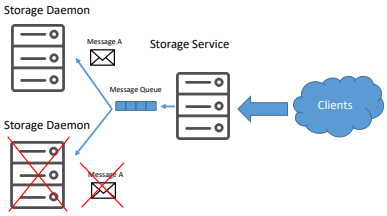


Figura 3: Exemplo de Tolerância a Falhas no *Back-End*.

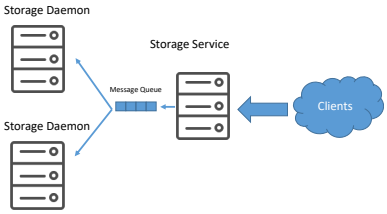


Figura 4: Arquitetura do Sistema de Armazenamento.

a implantação dos outros serviços. Como pode ser visto na Figura 4, o servidor web se comunica com o *back-end* através de fila de mensagens. Na implementação realizada o sistema suporta o uso do RabbitMQ[3], mas como pode ser visto no projeto de classes da Figura 5, essa parte pode também ser alterada para suportar outras tecnologias.

Cada instância CSV (Valores Separados por Vírgulas) é armazenada em uma tabela com o nome da aplicação. Para regressão e classificação cada instância deve ser única para o resultado; do ponto de vista de treinamento de modelos não faz sentido ter instâncias idênticas com resultados diferentes, essa regra é aplicada pelo próprio banco de dados. No *front-end*, o servidor escolhido para a interface REST foi o NodeJS[4] devido à facilidade de implantação e por possuir diversos pacotes de integração com outros sistemas. Tanto o *front-end* quanto o *back-end* podem ser vistos como micro serviços.

Para cada aplicação, esse serviço suporta dois comandos REST para o código cliente, a saber:

- Armazenar instância CSV: Adiciona uma instância CSV com a classificação ou o resultado da regressão correspondente;
- Limpar dados: Apaga todos os dados de treinamento de uma certa aplicação; esse efeito é desejado quando o usuário não quer utilizar dados mais antigos para treinar um novo modelo.

4.2 Serviço de Treinamento

Esse serviço é responsável por utilizar os dados de treinamento, aplicar transformações automáticas, descartar variáveis irrelevantes, treinar o modelo com diferentes algoritmos e escolher o melhor para o conjunto de dados disponibilizado. A arquitetura geral do módulo é análoga a do serviço de armazenamento de dados. Esse micro serviço também provê somente duas chamadas REST para os clientes, a saber:

[3]https://www.rabbitmq.com
[4]https://nodejs.org

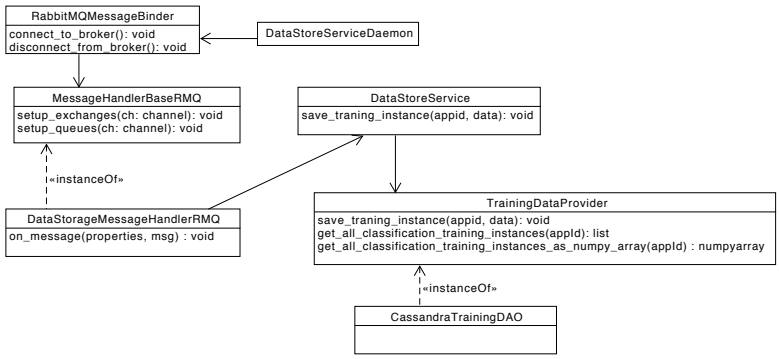


Figura 5: Desenho Interno do Serviço de Armazenamento.

- Treinar modelo: Inicia o treinamento de modelos para uma aplicação e salva o melhor modelo;
- Ler status de treinamento: Retorna se o treinamento foi bem sucedido, se está em progresso, ou falhou.

O treinamento é automático e envolve diversas etapas, que correspondem ao fluxo normal da geração de um modelo, liberando o pesquisador dessa tarefa. Usando a filosofia do arcabouço, a comunidade pode prover outras lógicas de fluxo de treinamento. Os seguintes passos ocorrem quando é iniciado o fluxo de treinamento:

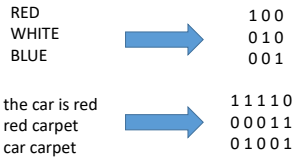
- Verificação de colunas que precisam de transformação, como datas no formato ISO (Figura 6(a)), variáveis categóricas, e instâncias de texto, como demonstrado na Figura 6(b);
- Salvar os índices a serem codificados e os codificadores correspondentes;
- Normalizar os dados e salvar o normalizador gerado;
- Treinar o modelo com diferentes algoritmos e salvar o que obtiver melhor resultado;
- Salvar os indíces de dados de entradas a serem removidos por ter sido detectados como irrelevantes;
- Salvar o modelo no sistema de armazenamento de modelos.

Na implementação padrão, o Redis é utilizado para armazenar os modelos gerados, mas como pode ser visto no desenho de classes da Figura 7, o arcabouço permite que a comunidade possa adicionar implementações para suportar outras tecnologias. Para a geração de modelo a biblioteca scikit-learn[5] está sendo utilizada, junto com os algoritmos de classificação Random Forest [3] e SVM (Support Vector Machines) [6], esses também podem ser facilmente substituídos ou terem mais opções adicionadas.

4.3 Serviço de Prescrição

Esse serviço é responsável por receber um fluxo de dados e gerar prescrições. Como pode ser observado na Figura 8, a arquitetura é um pouco diferente em comparação aos outros

(a) Extraindo Dados de Datas no Formato ISO.



(b) Transformação de Variáveis Categoricas e de Texto em Vetores.

Figura 6: Transformações de Dados.

serviços, existe uma fila extra de retorno para os servidores web, pois esse serviço funciona como um RPC (Remote Procedure Call).

Até o momento, quatro padrões de comunicação são suportados.

- Comunicação REST *request reply*: O cliente faz uma chamada REST para o servidor web e recebe a prescrição como resposta, como descrito na Figura 9(a);
- Comunicação *Websocket request reply*: Funciona de maneira análoga ao REST, só que utilizando o protocolo *websocket*;
- Atuação REST: O cliente envia os dados por REST ou *websocket* e o servidor faz uma chamada REST a um serviço fornecido pelo cliente, que é a parte da atuação. A Figura 9(b) exemplifica esse padrão de comunicação. O usuário do arcabouço precisa também implementar a construção da chamada REST;

[5]http://scikit-learn.org

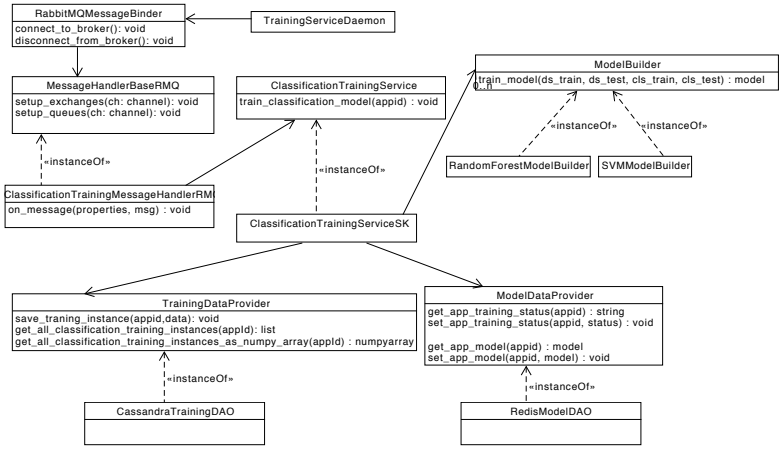


Figura 7: Desenho Interno do Serviço de Treinamento.

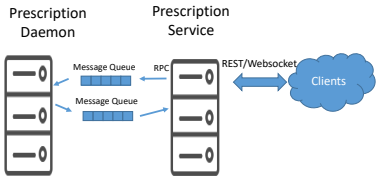


Figura 8: Arquitetura do Serviço de Prescrição.

- Atuação *Websocket Publish-Subscribe*: Nesse padrão, quando o cliente faz a requisição por REST ou *websocket*, a prescrição vai para todos os clientes conectados na aplicação como visto na Figura 9(c).

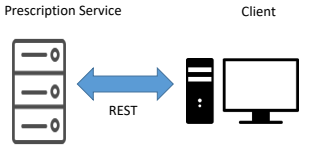
Na Figura 10 pode ser visto o desenho interno do serviço de prescrição; as classes e métodos que precisam ser implementadas pelo usuário serão descritas na próxima subseção.

4.4 Modelo de Aplicação

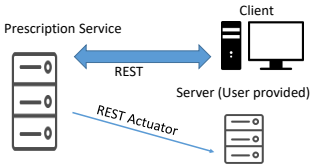
A Figura 11 exibe as classes mais importantes para o usuário do arcabouço.

- Classe `Application`: Essa classe precisa ser estendida pelo usuário, com as configurações de padrão de comunicação, nome de aplicação e outros detalhes. Como pode ser observado na Figura 10, mais de uma aplicação pode ser utilizada por servidor;
- Classe `ApplicationPrescriber`: É onde o usuário coloca o modelo de prescrição, que vai receber o resultado da predição;
- ApplicationActuator: É a parte final, onde o resultado da predição e prescrição estão disponíveis, e o usuário pode construir a mensagem *websocket*, ou o atuador REST.

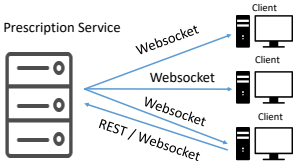
Estendendo essas poucas classes, o arcabouço permite a construção de aplicações orientadas a modelos de dados bastante poderosas, que podem ser facilmente integradas com outros sistemas de software, e podem ser implantadas em



(a) Prescrição via REST.



(b) Atuador REST.



(c) Atuação *Publish-subscribe*.

Figura 9: Padrões de Comunicação.

ambiente de nuvem computacional, podendo ser replicadas de forma horizontal para atender uma maior demanda de clientes quando necessário.

5. EXPERIMENTO COM USO DO ARCABOUÇO

Com o intuito de validar a arquitetura, um experimento foi desenvolvido para mostrar como essa estrutura pode ser utilizada em um cenário real. Como os dados são o elemento principal para a construção de modelos, um conjunto de dados disponíveis publicamente de outro estudo foi utilizado [10]. A Tabela 1 contém a configuração de *hardware* e as versões dos sistemas de software utilizados no experimento. A utilização do sistema foi feita da mesma maneira

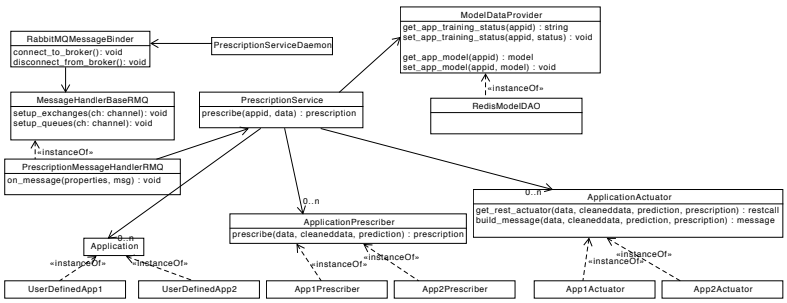


Figura 10: Desenho Interno do Serviço de Prescrição.

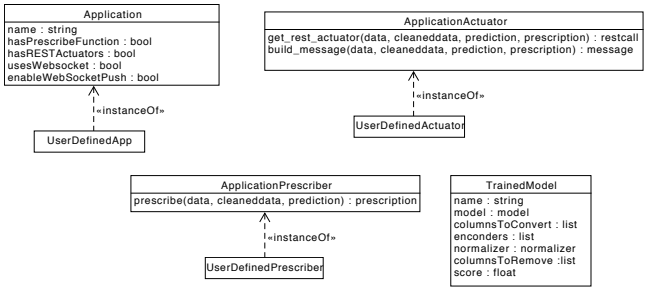


Figura 11: Modelo de Aplicação.

Tabela 1: Configuração do Sistema onde o Experimento foi Executado.

Processador:	Intel(R) Xeon(R) CPU E5-1607 v2 3.00GHz
RAM:	16Gb DDR3 1800Mhz
Sistema operacional:	CentOS 7.2
Python:	2.7.5
NodeJS:	4.4.3
RabbitMQ:	3.6.1
Cassandra:	3.0.5
Redis:	3.0.7
Scikit-learn:	0.17.0

que um sistema cliente teria interagido com o sistema: utilizando as chamadas REST.

5.1 Os Dados

Velloso et al. [10] conduziram uma série de experimentos que utilizaram aprendizagem de máquina para quantificar a qualidade de execução de um exercício físico, nesse caso, o levantamento lateral de peso. Eles utilizaram sensores no cinto, no braço, na luva e no peso. Os voluntários executaram os exercícios de forma correta e em quatro posturas incorretas. Os dados disponibilizados contêm uma série de medições dos sensores durante essas posturas, além da classificação da postura [10].

5.2 O Teste

O teste realizado emula a idéia de que, no futuro, pessoas terão sensores embutidos na roupa e esses se comunicarão com dispositivos mais complexos, como um relógio inteligente. O dispositivo inteligente, por sua vez, fará a comunicação com o serviço de prescrição. Finalmente, esse aparato de dispositivos e serviço irá ajudar as pessoas a não executar os exercícios de forma incorreta. Como resposta, o dispositivo poderia vibrar ou emitir sons, caso o usuário esteja executando os exercícios de forma incorreta.

Tabela 2: Prescrições Possíveis.

Corrigir a posição do ombro
Levantar mais o peso
Descer mais o peso
Corrigir a posição da cintura

Dos dados disponibilizados, 70% foi separado para o servidor e 30% para simular as chamadas do cliente. Todo o fluxo de limpeza de dados e escolha de entradas relevantes é feita automaticamente pelo sistema. Os dados foram fornecidos através de chamadas REST, como um cliente faria. O melhor modelo de classificação foi escolhido automaticamente pelo sistema, sendo o Random Forest [3].

O modelo de prescrição nesse caso é bem simples: para cada postura incorreta, o sistema emite uma mensagem recomendando o que precisa ser feito para corrigir a postura; essa mensagem apareceria no relógio inteligente do usuário. A Tabela 2 mostra as prescrições possíveis. Um cliente em Python foi desenvolvido para utilizar os 30% de dados que forão separados, realizar as chamadas e receber as prescrições.

5.3 Resultados

Levando em conta uma abordagem caixa preta, a aplicação enviou os dados, requisitou a criação do modelo e forneceu o modelo de prescrição. Nesse caso, deve-se avaliar se as prescrições foram geradas corretamente ou não, como pode ser observado na Tabela 3.

Para fazer essa análise, os seguintes casos foram considerados:

- Caso 1: em que foi medido quantas prescrições eram es-

Tabela 3: Resultado do Experimento de Prescrição.

Total de Prescrições	5.886
Prescrições Corretas	5.749 (98%)
Falsos Positivos	12
Falsos Negativos	26

Tabela 4: Resultados de Desempenho do Experimento.

Armazenamento de instâncias CSV por segundo	1056
Prescrições por segundo	82

peradas. Assim, caso o usuário estivesse fazendo exercício de forma incorreta, ele seria notificado a se corrigir. Essas são consideradas as prescrições corretas;

- Caso 2: determina os falsos positivos e falsos negativos. Quando o usuário está executando o exercício da forma correta e é notificado para fazer alguma correção é configurado o falso positivo. Quando o usuário faz o exercício de forma incorreta e não é notificado, é configurado um falso negativo;

- Caso 3: quando o usuário recebe uma prescrição incorreta.

Fazendo uma análise interna do que ocorreu, foi identificado que o sistema primeiro treinou um modelo utilizando o algoritmo SVM (Support Vector Machine) [6] e obteve uma precisão de somente 0.274, que seria inviável para uma aplicação. Em seguida, testou o algoritmo Random Forest [3] que obteve uma precisão de 0.98, e disponibilizou esse modelo para a aplicação.

Outro aspecto importante observado foram as métricas de desempenho do sistema, já que é esperado que mesmo venha a ser utilizado em situações de grande tráfego de dados, como o cenário de Internet das Coisas e computação em nuvem. Os resultados obtidos para o teste podem ser vistos na Tabela 4. A quantidade de 82 prescrições por segundo é de um servidor web com um trabalhador de *back-end*. Caso seja necessário, mais trabalhadores podem ser adicionados e, quando o servidor web estiver saturado, todo esse conjunto pode ser replicado horizontalmente, oferencendo uma escalabilidade praticamente infinita.

6. CONCLUSÃO E TRABALHOS FUTUROS

A análise dos dados torna-se uma ferramenta preponderante em cenários de geração de grandes volumes de dados, como a IoT. Esse trabalho apresentou um arcabouço genérico e flexível, que permite que desenvolvedores incorporem análise preditiva e prescritiva, ao mesmo tempo que permite atuação em tempo hábil para o usuário da aplicação. O arcabouço foi desenvolvido com tecnologias que favorecem a sua modularidade, integração, tolerância a falhas e desempenho. Toda a arquitetura externa e interna do sistema foi apresentada de forma a mostrar toda a flexibilidade que o arcabouço possui para criação de aplicações de análise prescritiva em tempo real. A arquitetura de micro serviços adotada possibilita melhoria de desempenho ao permitir a disponibilidade de poder computacional, justamente onde os gargalos acontecem, otimizando assim a utilização de recursos. Um experimento teste, a partir de simulações, foi realizado, demonstrando a eficácia e desempenho do arcabouço, considerando cenários com alta carga de dados.

Como trabalhos futuros, planeja-se a implementação da solução para aplicações de produção com sensores reais, além de servidores remotos, a fim de testar o desempenho do fluxo de dados nessas condições. Outro teste interessante a ser feito seria a replicação horizontal dos serviços, com o intuito de verificar o aumento de transações por segundo da aplicação, validando melhor a arquitetura para cenários de Big Data e IoT. Uma série de trabalhos futuros podem também envolver a utilização de outras tecnologias de construção de modelos como redes neurais, aplicações de reconhecimento de imagem, além de suporte a outros sistemas de armazenamento.

7. REFERÊNCIAS

[1] L. Atzori, A. Iera, G. Morabito, and A. Diee. The Internet of Things : A survey. *Computer Networks*, 54(15):2787–2805, 2010.

[2] R. Barber and M. Sharkey. Course correction: using analytics to predict course success. In *2nd international conference on learning analytics and knowledge*, pages 259–262, 2012.

[3] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina. Microservices: yesterday, today, and tomorrow. *arXiv preprint arXiv:1606.04036*, 2016.

[5] L. J. Fülöp, Á. Beszédes, G. Tóth, H. Demeter, L. Vidács, and L. Farkas. Predictive complex event processing: a conceptual framework for combining complex event processing and predictive analytics. In *Fifth Balkan Conference in Informatics*, pages 26–31, 2012.

[6] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems*, 13:18–28, 1998.

[7] Z. Huang, P. C. Wong, P. Mackey, Y. Chen, J. Ma, K. Schneider, and F. L. Greitzer. Managing complex network operation with predictive analytics. In *AAAI Spring Symposium: Technosocial Predictive Analytics*, pages 59–65, 2009.

[8] E. Siegel. *Predictive analytics: The power to predict who will click, buy, lie, or die*. John Wiley & Sons, 2013.

[9] R. Tönjes, P. Barnaghi, M. Ali, A. Mileo, M. Hauswirth, F. Ganz, S. Ganea, B. Kjærgaard, D. Kuemper, S. Nechifor, et al. Real time iot stream processing and large-scale data analytics for smart city applications. In *poster session, European Conference on Networks and Communications*, 2014.

[10] E. Velloso, A. Bulling, H. Gellersen, W. Ugulino, and H. Fuks. Qualitative activity recognition of weight lifting exercises. In *4th Augmented Human International Conference*, pages 116–123, 2013.

[11] N. Zumel, J. Mount, and J. Porzak. *Practical data science with R*. Manning, 2014.