Utilização de Dataflow para previsão de aceitação de respostas no fórum StackOverflow.com

Using Dataflow to predict answers' acceptance in the StackOverflow.com forum

Talita Albuquerque de Araújo[1] orcid.org/0000-0003-4002-8238
Jairson Barbosa Rodrigues[2] orcid.org/0000-0003-1176-3903

[1] Escola Politécnica de Pernambuco, Universidade de Pernambuco, Recife, PE, Brasil
[2] Colegiado de Engenharia da Computação, Universidade do Vale do São Francisco, Juazeiro, BA, Brasil

E-mail do autor principal: taa2@ecomp.poli.br

Resumo

Nos últimos anos processar dados em larga escala tem sido um grande desafio, sendo, para isso, necessária a utilização de sistemas de alto desempenho para esse processamento. Este trabalho tem como objetivo apresentar um framework que permita que seja desempenhada essa função de forma rápida e simples, tirando proveito da estrutura do *DataFlow* para processamento de Big Data. A análise realizada é do tipo preditiva, em uma base disponibilizada on-line. A partir dela, será mostrado o uso do framework e se procurará verificar se o modelo gerado teve sucesso ou não. Os indicadores usados para essa comprovação serão a acurácia, a curva ROC, a especificidade e a sensitividade. Como resultado, espera-se extrair conhecimento sobre a aplicação do framework *DataFlow* para análise de grandes quantidades de dados e mostrar algumas vantagens no seu uso prático.

Palavras-chave: Big Data; Aprendizado de Máquina; *Dataflow*; Apache Beam;

Abstract

In recent years large-scale data processing has been a major challenge, requiring the use of high-performance systems for this processing. This work aims to present a framework that allows this function to be performed quickly and easily, taking advantage of the DataFlow structure for Big Data processing. The analysis performed is of the predictive type, in a database made available online. From this, it will be shown the use of the framework and will try to verify if the model generated was successful or not. The indicators used for this verification will be the accuracy, the ROC curve, the specificity and the sensitivity. As a result, it is expected to extract knowledge about the application of the DataFlow framework for analyzing large amounts of data and to show some advantages in its practical use.

***Key-words**: Big Data; machine learning; Dataflow; Apache Beam;*

 http://dx.doi.org/10.25286/repa.v3i3.942

1 Introdução

O presente artigo realiza, como estudo de caso, uma análise preditiva na base de dados do site *StackOverflow* **[1]**— disponibilizada na *BigQuery*, um *Data Warehouse* do *Google Cloud Plataform* **[2]** —, empregando o *framework Dataflow*, usado para processamento de *Big Data*.

É esperado que mais dados sejam gerados nos próximos 5 anos do que nos últimos 5.000 anos **[3]**. Tal previsão reforça a necessidade de ferramentas de análise de dados em escala *Big Data*. O *framework Hadoop* apresentou-se, então, como uma evolução natural nesse cenário; todavia, há casos como os de análises em tempo real e análises com Aprendizagem de Máquina nos quais o *Hadoop* não é plenamente satisfatório **[4]**.

E como isso apresenta-se o *framework Dataflow* que pode ser apresentado como um modelo de programação unificado e um serviço gerenciado que desenvolve e executa uma grande variedade de padrões de processamento de dados, essa análise será realizada sobre os dados do fórum *StackOverflow*, no qual desenvolvedores postam perguntas referentes a tecnologia em busca de ajuda ou explicações para as suas dúvidas. Por ser um fórum, existe muita interação entre os usuários, através de *posts* com respostas, comentários ou até novas dúvidas. O usuário que postou a pergunta inicial deve definir uma das respostas ou comentários como aquele (a) que resolveu o seu problema, e essa interação é marcada como a resposta válida à pergunta realizada no fórum.

Na seção 4 do presente artigo, serão expostos alguns trabalhos nos quais houve uso do *framework* Dataflow. Já os conceitos importantes para a compreensão do tema serão esclarecidos na seção 5, que corresponde ao Referencial Teórico. Daí, parte-se para a explicação do caminho percorrido na elaboração do trabalho, quando, na seção 6, será descrito como alguns dos conceitos especificados foram utilizados para a realização da análise na base de dados escolhida. Os resultados verificados e as conclusões a que se chegou são apresentados e interpretados na seção 7 e, ao final, algumas ideias para trabalhos futuros se encontram na seção 8.

2 Contexto Atual

Sabe-se que houve avanços quanto às ferramentas utilizadas no processamento de *Big Data*, e a evolução do modelo monoestágio do *MapReduce* para o modelo de processamento multiestágio do *framework Spark* consiste em um dos casos mais conhecidos **[4]**, **[5]**, **[6]**. Já trabalhos comparativos entre *Spark* e *Dataflow* podem ser encontrados em **[7]**, **[8]** e **[9]**. Destas referências, as duas primeiras foram criadas pela empresa Google, que desenvolveu o *Dataflow* como um *framework* integrado com outras ferramentas na sua plataforma *Google Cloud*. Ao se gerar um código usando o *framework* para processar dados em paralelo (em fluxo contínuo) não se faz necessário gerar um novo código para se trabalhar com dados em batch: o mesmo código serve para os dois tipos de processamento, facilitando a análise de grandes quantidades de dados.

> [...] nenhum outro modelo de programação paralela de dados em grande escala fornece a enorme capacidade e a facilidade de uso do Dataflow/Beam **[8]**.

Pelo fato de o *Dataflow* ser apresentado como um sucessor do *MapReduce* **[10]** e trazer soluções diferentes das propostas pelo *Spark*, e ainda por ser disponibilizado de maneira simples pelo Google Cloud, foi decidido elaborar este artigo com foco no processamento de dados com *Dataflow*, pois ele oferece um *SDK* (Kit de Desenvolvimento de Software) de rápida curva de aprendizado que permite a construção de *Jobs* em vários motores de tempos de execução (*runtime engine*), conforme é descrito na proposta de Onofre e Jagielski **[10]**.

3 Objetivos

- Criar um modelo preditivo usando *DataFlow* para definir se um *post* no fórum *StackOveflow* terá uma resposta marcada como aceita; e
- Demonstrar o potencial de aplicação de novas ferramentas e novos paradigmas de

Aprendizado de Máquina em grandes volumes de dados.

4 Trabalhos Relacionados

A análise de dados através de *DataFlow* é demonsytada em **[11]**, que explica o *framework* em profundidade, descrevendo seu modelo de programação e o funcionamento do SDK, não sedo feita a analise de uma base, que mostraria o uso de forma pratica do *DataFlow*. Já em [12], encontra-se um estudo comparativo das funcionalidades do *framework Dataflow* com *Spark* e *Flink*, analisando-se o processamento de cada um, o mecanismo de tolerância a falhas e suas forças e fraquezas, mostrando, a partir de outros *frameworks* utilizados no mercado, os atributos do *Dataflow*.

5 Referencial Teórico

Nesta seção, será apresentada uma introdução acerca dos conceitos e tecnologias utilizadas neste trabalho — especificamente: *Big Data*, *Hadoop*, *MapReduce*, *Spark,* além do *DataFlow* e das bibliotecas *TensorFlow* e *Scikit-learn*.

5.1 Big Data

Apesar de ser um termo bastante usado na atualidade, o conceito de *Big Data* pode variar de autor para autor ou quando se têm objetivos diferentes. Doug Laney **[13]** define *Big Data* com foco nas três dimensões de um conjunto dos dados. Segundo o autor, uma massa de dados pode ser caracterizada como *Big Data* quando supre os requisitos de: volume (ou seja, a quantidade de dados); velocidade (em que as informações são geradas); e variedade (em relação às fontes e tipos de dados — de diversos formatos, estruturados ou não estruturados).

Com o passar do tempo, muitas outras dimensões foram adicionadas a esse conceito. A título de exemplo: veracidade **[14]** e valor **[15]**, dentre outros **[16]**. Muitos desses termos se referem ao resultado do emprego de técnicas de ciência de dados, não refletindo essencialmente a atribuição de novas características. Sob uma ótica, essas classificações adicionais podem ser percebidas como um esforço para se entender melhor o conceito *Big Data*; por outra perspectiva, pode-se incorrer em falta de precisão ou mesmo publicidade extravagante sobre o tema.

Dados em escala *Big Data* não podem ser manipulados através de ferramentas e métodos tradicionais. Dessa forma, ferramentas e técnicas específicas têm sido desenvolvidas para seu tratamento adequado perante os desafios impostos na atualidade.

5.2 Hadoop

O *Hadoop* é um *framework* usado para armazenamento distribuído e para processamento de grandes conjuntos de dados **[17]**. Dentre seus componentes principais, encontram-se o HDFS (*Hadoop Distributed File System*) e a máquina de execução MapReduce.

O *HDFS* trata do armazenamento distribuído de dados em *hardware* de forma escalável e confiável. A sua arquitetura faz com que a informação seja armazenada em blocos, que são distribuídos em diversas máquinas ao longo de um *cluster*. Na arquitetura *HDFS*, existe uma máquina que controla as demais, chamada *namenode*; os dados em si são armazenados nos *datanodes,* e em cada nó do *cluster* pode existir um ou mais *datanodes*. **[18]**. Essa arquitetura foi desenhada para ser executada em *hardware* de baixo custo.

O *MapReduce*, por sua vez, é um modelo de programação desenvolvido pela Google usado no processamento de vastas informações em *clusters* **[19]**. O seu modelo de processamento é paralelo e escalável, o que gera um grande conjunto de dados organizados em forma de blocos, que são distribuídos em todos os nós do *cluster* utilizando duas funções: *map* e *reduce* **[5]**.

Esse modelo funciona bem quando submetido a processamento de dados em lote, lidos de forma sequencial, e por isso a velocidade não é essencial, já que o *MapReduce* tem que buscar os dados armazenados pelo *HDFS*, ler esses dados e realizar a operação que foi designada. Após isso, o resultado deve ser salvo no *armazenamento* HDFS. O procedimento é repetido até que todo o trabalho seja finalizado. Tais operações tornam o processamento lento devido ao excessivo número de operações de entrada e saída. Em cenários de tratamento de dados iterativos ou em um fluxo

http://dx.doi.org/10.25286/repa.v3i3.942

contínuo de dados, fazem-se necessárias abordagens que diminuam o gargalo do *Hadoop*.

Dessa necessidade, surgiu o *Spark*, um novo *framework* que trabalha como um motor de computação distribuída em memória **[20]** resultando em velocidade de processamento maior.

5.3 Spark

Em se tratando de processamento de tarefas iterativas e distribuídas (tais como Aprendizagem de Máquina), o *framework Apache Spark* é uma solução largamente adotada, como se pode ver pela lista, disponibilizada em sua página oficial, de empresas que atualmente utilizam o *Spark* **[21]**. Com esta plataforma, é possível manipular dados de vários tipos — texto, grafos e imagens — e que vêm de diferentes origens — em lote ou fluxo contínuo de dados em tempo real. Entre as suas vantagens, encontra-se o fato do *Spark* ser escalável, rápido e possuir curva de aprendizado simples, permitindo codificação de tarefas em *Python, Java, R* ou *Scala* **[22]**.

O armazenamento em memória é possível graças a uma estrutura de dados paralela e tolerante a falhas chamada RDD — Conjunto de Dados Distribuídos Resilientes **[23]**, que salva os resultados intermediários na memória do nó correspondente; assim, podem ser executados algoritmos iterativos de Aprendizagem de Máquina com maior eficiência.

Esse conjunto de dados não é salvo em armazenamentos físicos, e sim na memória volátil dos nós do *cluster*; isso torna sua recuperação e escrita muito rápidas e, assim, pode-se reutilizar esses resultados parciais para uso em operações paralelas do tipo *MapReduce* **[24]**.

O programador pode criar diversos *pipelines*, que contêm em si várias etapas de diferentes complexidades, e os dados que se encontram nesses *pipelines* podem ser compartilhados, permitindo que os *Jobs* do *Spark* possam trabalhar com os mesmos dados. Os resultados do processamento dos dados são retidos em memória, minimizando a escrita no HDFS ou outro armazenamento que possa ser escolhido.

Assim, para aplicações que exigem processamento rápido e análises em tempo real, o *Apache Spark* supera o seu predecessor, o *Hadoop*.

5.3 Apach Beam e Processamento *Big Data*

O *Dataflow*, disponibilizado pela Google na plataforma *Google Cloud,* permite que sejam desenvolvidos modelos de análise de *Big Data*, que possa ser integrada com diversos produtos disponibilizados pela plataforma para o trabalho com *Big Data*, como: o *Big Query*, para trabalhos com bases *Big Data*; o *Storage*, para armazenamento de grandes quantidades de dados; o *Dataproc*, um serviço de gerenciamento do *Spark* e do *Hadoop*; dentre outros. Enquanto no *Google Cloud* é disponibilizada a versão 1.0 do *Dataflow*, existe uma versão mais atual, 2.x, que é chamada de *Apache Beam*.

O *Apache Beam* é mantido pela Apache e tem distribuição gratuita do *SDK* utilizado pelos programadores para gerar os códigos pertinentes.

O *SDK* permite a criação de *pipelines* de processamento, que poderão ser executados em diversas plataformas, inclusive na nuvem. A arquitetura suporta linguagens como *Java* e *Python*.

Para fins de compreensão, ressalta-se que, ao se usar o termo *Dataflow* neste artigo, estará sendo mencionado o *framework* disponibilizado pela Google na sua plataforma *Google Cloud* — cuja infraestrutura foi usada para maior rapidez no processamento — e, quando for mencionado *Apache Beam*, o que estará sendo citado é o *SDK*, que foi usado para o desenvolvimento do código.

O *Apache Beam* — que contém parte do modelo de programação/*SDK* do *Google Cloud Dataflow*, o qual foi criado a partir de uma evolução do *MapReduce* e incubado pela Apache — tem seu nome originado da junção de duas palavras, que nomeiam os dados "Batch" e o processamento "strEAM". Disso, infere-se que o *Apache Beam* pode ser usado para dados em *batch* — que são dados separados em lotes, sendo processado um lote de cada vez —, para um *stream* contínuo de dados e para vários tipos de processamento de dados, incluindo ETL. Tudo isso pode ocorrer de forma simplificada em motores variados, como *Spark Runner* ou *Google Cloud Dataflow*, e em qualquer escala **[10]**.

Além do *SDK*, outro grande diferencial do Apache Beam é o seu modelo de programação unificado. Em relação ao modelo, existem quatro conceitos principais de nível superior que devem ser pensados ao se construir o *Job* de

processamento, que são: *Pipeline*, *PCollection*, *PTransform* e *Pipeline Runner* **[10]**.

O *Pipeline* representa um *Job* que recebe dados de entrada, realiza alguma computação e armazena a saída, seja um resultado ou os próprios dados tratados na saída. O *PCollection* representa o conjunto de dados dentro do *Pipeline*; esse conjunto possui tamanho praticamente ilimitado, podendo ser trabalhado com dados em lote ou cuja origem tenha atualização contínua, criando-se, assim, um *Job* para um *stream* de dados. Vários tipos de *Jobs* podem ser feitos dentro do *Pipeline*, como construção de histogramas e criação de modelos de Aprendizagem de Máquina, e é através do *PCollection* que *Jobs* paralelos podem acontecer **[25]**.

O *PTransform* é a computação em si, que transforma dados de entrada em dados de saída **[26]**. Ele pode executar transformações em elementos, agregar vários elementos em conjunto ou ser uma combinação composta de outros *PTransforms* **[25]**. Observe a Figura 1.

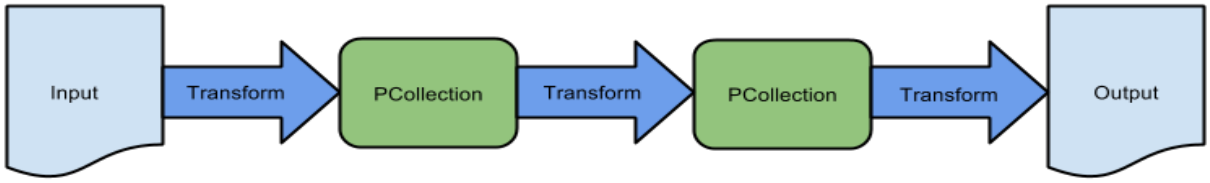

Figura 1: Funcionamento de Jobs do Spark. Fonte:**[45]**

Ao se criar programas com o *SDK* do *Apache Beam*, um *Job* de processamento é gerado e será executado por um dos serviços de *backend* de processamento distribuído e múltiplo, através do que é chamado de executor de *Pipelines*: os *PipelineRunners* **[27]**. Existem várias opções; algumas são: *DirectRunner*, que é executado diretamente em uma máquina local, após a instalação do *Beam* na máquina; *DataflowRunner*, no qual o *Pipeline* a ser executado é submetido no *Google Cloud Data Flow* (executor utilizado neste artigo) — esta opção provê uma integração com outros sistemas da plataforma; e *SparkRunner*, que executa o *Pipeline* em um *Apache Spark Cluster* **[28]**. Este fornece um *framework* de computação cluster *open-source*, além de poder ser executado localmente ou na nuvem.

5.5 Bibliotecas de Aprendizagem de Máquina

O treinamento e teste do modelo preditivo foi desenvolvido usando o *Tensor Flow* [29], uma biblioteca para Aprendizagem de Máquina de código aberto. As demais atividades do projeto foram desenvolvidas usando a biblioteca *Scikit-learn* [30].

O *Tensor Flow* disponibiliza ferramentas para definir modelos novos, não oferecendo soluções prontas de Aprendizagem de Máquina. Assim, o desenvolvedor com conhecimento técnico pode criar modelos flexíveis com um conjunto extenso de funções e classes e realizar os cálculos a partir de chamadas, podendo o *Tensor Flow* também ser usado para execução de códigos matemáticos complexos [29].

Além disso, o *Tensor Flow* permite implantar aplicativos em clusters distribuídos, em estações de trabalho locais, em dispositivos móveis e em aceleradores customizados. Pode-se trabalhar com o *Tensor Flow* da maneira que mais se adequa ao caso do projeto **[31]**.

Uma computação *Tensor Flow* é descrita por um gráfico de fluxo de dados, que é composto por um conjunto de nós. Esse gráfico representa todos os cálculos feitos, as operações matemáticas, os parâmetros e suas regras e o pré-processamento de entrada. Vale a pena salientar que o modelo suporta várias execuções simultâneas sobre o processamento geral. Cada nó possui zero ou mais entradas e saídas e representa a instanciação de uma operação **[32]**.

A biblioteca *Scikit-learn* **[33]** possui código aberto e é específica para a linguagem *Python*. Ela contém um conjunto bastante amplo de algoritmos para modelos estatísticos e implementa muitos

http://dx.doi.org/10.25286/repa.v3i3.942

algoritmos de Aprendizagem de Máquina, o que visa a facilitar a vida do desenvolvedor **[34]**. A biblioteca é focada na modelagem dos dados, sendo usada em conjunto com outras bibliotecas *Python* para carregamento, manipulação e sumarização dos dados.

5.6 Métricas no modelo preditivo

A base de dados do fórum *StackOverflow* se encontra armazenada no *Big Query* **[35]** na área denominada pela plataforma como *bigquery-public-data* **[36]** e contém as interações entre os usuários e outras informações dessa comunidade on-line. A base sempre reflete o conjunto atual de dados compartilhados da comunidade.

O *Big Query* permite a análise e a consulta dos dados através de *scripts* SQL; com isso, podem ser realizados tratamentos na base e, assim, utilizá-la para responder diversas perguntas — no caso deste estudo, analisando as variáveis dos *posts*, determinar se as perguntas cadastradas no fórum *StackOverflow* terão uma resposta aceita ou não pelo usuário que fez a pergunta. Com a base escolhida, foi realizada uma análise preditiva para tentar responder à questão.

A análise preditiva é diferente de outros tipos de análise de dados (como a descritiva ou a prescritiva) por usar padrões do passado — como apontado por **[37]** — para determinar eventos ou respostas futuras, não se utilizando de conhecimento empírico baseado somente na experiência prévia e pessoal do analista. A análise preditiva tem, assim, o objetivo de usar dados estatísticos e históricos para decidir as melhores ações ou mesmo saber como uma determinada situação irá ocorrer. Em uma abordagem de negócios, por exemplo, a análise preditiva pode ser utilizada para várias atividades, como: identificar tendências; prever comportamentos; entender as reais necessidades de clientes; determinar se um paciente possui determinada doença; ou mesmo se um voo irá atrasar ou não.

O modelo de Aprendizagem de Máquina escolhido foi a regressão logística. Uma das razões dessa escolha foi sua capacidade para trabalhar com modelos binários — no caso da base analisada, *true* ou *false*. A regressão logística calcula a relação entre a variável categórica e todas as outras variáveis dependentes. Um bom modelo deve avaliar quais dessas características realmente fazem a diferença na contagem de probabilidade. A regressão logística funciona fazendo suposições sobre os dados analisados para o aprendizado sobre a base e, dessa forma, prevê probabilidades, e não somente as categorias possíveis **[39]**.

Para desenvolvimento do modelo, os dados devem ser divididos em grupos de treinamento e de teste. Para tal, podem ser usados vários métodos; no caso concreto, foi usado o *Cross Validation.* Os conjuntos de dados são usados, em um momento, para treinamento e, em outro, para teste, e assim não há o risco de o modelo aprender apenas sobre uma categoria específica **[38]**.

As métricas descritas a seguir foram utilizadas para definir se o modelo foi bem treinado.

A matriz de confusão mostra o número de classificações corretas em comparação com as classificações que foram previstas para as classes existentes na base. Bases com categorias binárias, do tipo *true* ou *false*, geram uma matriz de confusão menor e de melhor compreensão do que se tivessem mais categorias. Esse tipo de visualização (apresentado na Tabela 2) é mais simples e foi gerado para a matriz de confusão da análise da base do *StackOverflow*.

Com os valores definidos na matriz, algumas métricas foram calculadas, como a acurácia, que consiste na proporção de predições corretas, observando-se o acerto total. Por utilizar todos os valores de desempenho gerados pela matriz de confusão, ela é considerada um classificador geral.

Outra métrica utilizada foi a curva ROC. No campo da Ciência de Dados, essa curva é útil para se trabalhar com domínios cujas classes estão desbalanceadas, pelo fato de ela ser baseada nas taxas médias de VP e FP, que não dependem da distribuição das classes.

A curva ROC é construída ordenando-se as linhas, validando-se os valores (se positivos ou negativos) e construindo-se a curva de baixo para cima. Quando o valor é positivo, a curva sobe; quando negativo, ela inclina-se para a direita.

A curva ROC é baseada nas métricas VP e FP, mas não nos valores absolutos, e sim nas suas taxas, podendo estas serem chamadas de Taxa VP/Taxa FP (*VP rate/FP rate*) ou identificadas por seus nomes específicos, que são: sensitividade, ou *recall*, e especificidade, respectivamente. Essas métricas são obtidas das seguintes formas:

- Sensitividade: porcentagem de amostras classificadas como positivas e corretas, sobre o total de amostras positivas.

- Especificidade: porcentagem de amostras negativas identificadas corretamente, sobre o total de amostras negativas.

6 Metodologia

Os dados do *StackOverflow* utilizados para o estudo de caso totalizam um tamanho de 174 GB, com 786.588 amostras, contendo variáveis diversas, descritas na Tabela 1, tais como: quais foram as perguntas feitas, quem são os usuários, quais foram as interações com os *posts*, dentre outros dados.

A base total do fórum *StackOverflow* não está totalmente disponível no *Big Query*, por isso não foram usadas na pesquisa todas as informações existentes no fórum, apenas as já se encontravam na base disponibilizada no *Big Query*. Além disso, deve-se observar que, pela natureza do fórum, a base tende a crescer de maneira constante, exigindo do programador um maior conhecimento em análise de *stream* contínuo. No entanto, esse não foi o foco do trabalho e sim demonstrar o potencial da ferramenta e a aplicabilidade do modelo, mesmo que não manipulando dados em volumes que se encaixem plenamente no conceito *Big Data* em relação ao seu tamanho.

A base possui desbalanceamento de 21% para perguntas respondidas e 79% para posts que não tiveram respostas aceitas. Apesar disso, não houve tratamento de balanceamento, pois, uma vez que esses 21% não incorrem em eventos raros, como foi demonstrado em [39], o resultado não seria alterado, havendo impacto apenas na métrica acurácia que foi complementada com outras métricas — curva ROC, sensitividade e especificidade. A curva ROC é particularmente útil em domínios em que existe uma grande desproporção entre as classes **[40]**.

Durante o pré-processamento da base foram constatados dados quantitativos — numéricos, e variáveis textuais de identificação dos *posts* — como o título e o próprio texto relacionado. As variáveis numéricas podem ser classificadas como discretas ou contínuas **[41]**: as primeiras são baseadas em contagens — no caso da base em questão, número de views de um *post* ou número de vezes em que ele foi marcado como favorito; já as segundas consistem de variáveis numéricas que podem assumir qualquer valor em um determinado intervalo, podendo ser data/hora, por exemplo (no caso da base escolhida, a data em que foi cadastrada a pergunta e a data da última interação com o *post* eram variáveis contínuas, mais difíceis de se trabalhar, então foi criada uma variável que registra o tempo que o *post* permaneceu ativo no fórum, nomeada *days_it_has_been_active*).

http://dx.doi.org/10.25286/repa.v3i3.942

Tabela 1: Esquema da base de dados analisada.

Nome da Coluna	Tipo	Descrição
Answer_count	Int	Número de respostas sugeridas
Comment_count	Int	Número de comentários feitos no *post*
Favorite_count	Int	Número de vezes em que o *post* foi marcado como favorito
Score	Int	Pontuação dada ao *post* pela plataforma
View_count	Int	Número de vezes em que o *post* foi visualizado
Days_it_has_been_active	Int	Número de dias corridos entre a data da criação do *post* e a última data ativa
Accepted_answe_id	Int	ID do *post* aceito como resposta; caso não possua, o valor é zero
Has_accepted_answer	Bool	Variável criada a partir da anterior; caso tenha resposta, o valor é true (1); caso não possua, é false (0)

Fonte: o autor.

As variáveis textuais foram retiradas da base usada pelo modelo, pois não forneciam informações de classificação (funcionando apenas como um Identificador Único do objeto cadastrado na linha da base); com isso, foram utilizadas apenas as variáveis numéricas da tabela já existente, sem uso de categorias textuais.

Não foram observadas variáveis categóricas que pudessem ser extraídas a partir das outras existentes, ficando a base com as variáveis discretas e com a variável alvo que identifica se um *post* possui uma resposta aceita ou não (*Accepted Answer ID*). No entanto, como não se pôde trabalhar com IDs específicos, passou-se a utilizar a variável criada has_accepted_answer, como mostrado na Tabela 1

Essa variável define se existe ou não uma resposta ao *post*; esse tipo de variável lida com situações em que o valor final é categorizado em um número finito de possibilidades mutuamente exclusivas **[42]**. Por ser uma variável binária, o 0 representa *posts* sem resposta aceita, e o 1, os que possuem resposta aceita, como mostrado na Tabela 1.

No fim, as variáveis utilizadas foram: quantidade de respostas sugeridas; número de comentários feitos no *post*; quantas vezes a pergunta foi marcada como favorita; pontuação do *post* dado pela plataforma; número de vezes em que a postagem foi visualizada; ID da postagem aceita como resposta; e as duas novas criadas — dias em que a postagem passou ativa e uma variável booleana, que informa se houve resposta aceita. Tais variáveis se encontram relacionadas na Tabela 1.

Com a criação da variável *days_it_has_been_active*, esperava-se poder auxiliar o modelo a identificar melhor se um *post* teria uma resposta válida ou não, mas foi observado que o tempo de vida de um *post* não influenciou a análise probabilística, havendo apenas uma pequena variação das métricas utilizadas pelo artigo, não sendo realmente considerada como uma melhora no modelo.

Com as alterações na base descritas antes e com a escolha do modelo preditivo, procurou-se prever o resultado quanto à aceitação de uma resposta ao *post* — dadas as informações da Tabela 1, sem as variáveis *has_accepted_answer* e *accepted_answer*_id — e, assim, responder à pergunta proposta no artigo.

Para executar o modelo produzido, o *SDK* ajudou na criação do *Pipeline*, que busca a base hospedada no Storage com as novas variáveis já geradas. Além do *SDK*, também foram usadas as bibliotecas de Aprendizagem de Máquina citadas anteriormente — *Tensor Flow* e *Scikit-learn* — especializadas e disponibilizadas em *Pyhton*. Por ser uma base grande e a máquina local utilizada não possuir um grande poder de processamento, foi utilizado o executor do *Google Data Flow* e, assim, a computação foi feita na nuvem, tendo sido geradas as métricas explicadas no Referencial Teórico para analisar se o modelo foi treinado com sucesso.

7 Resultados e Conclusões

A partir do modelo produzido e da análise feita partindo das variáveis obtidas, foi possível criar um modelo com resultados altos para as métricas consideradas e, assim, realizar a predição desejada.

A aplicação do SDK Apache Beam demonstrou facilidade de uso e praticidade quanto à criação de Jobs no executor de Pipeline utilizado para a avaliação descrita neste estudo. O fato de as ferramentas serem distribuídas de maneira gratuita e separadas umas das outras — como é o caso do *SDK* ou do *Runner* — possibilita a geração da computação desacoplada de qualquer serviço, ficando o desenvolvedor livre para escolher como montar uma arquitetura de processamento que seja ideal para o seu contexto específico. Por exemplo, o usuário pode escolher se usará o *SDK* com o Hadoop ou com todo o conjunto do Apache *Beam*, além de poder utilizar o *Pipeline Runner* local ou no Apache Spark Cluster.

A fase de pré-processamento depende muito da capacidade do analista para identificar o que mais se adequa ao estudo **[43]** e para verificar quais variáveis farão com que o modelo seja bem-sucedido.

Ao fim dessa fase e após o modelo gerado ser rodado obteve-se a Tabela 2, a matriz de confusão, da análise feita.

Tabela 2: Matriz de confusão gerada pela análise

		Valor Previsto	
		0	1
Valor Verdadeiro	0	484.462	103.234
	1	72.530	126.362

Fonte: o autor.

A partir da matriz, é possível obter o valor da acurácia, que, para o modelo testado, foi de 0,85385. Porém, a acurácia não é uma métrica confiável para o caso de bases desbalanceadas **[44]**. Isso acontece porque o classificador acaba se mostrando tendencioso para o lado com mais classes; por essas serem as que mais existem na base, a probabilidade tende a ser maior.

Contudo, esse tipo de tendência não influencia a curva ROC, porque ela é invariante quanto à proporção de exemplos entre as classes — desde que seja assumida a distribuição de uma classe como característica do domínio **[40]** — , e, por isso, a curva ROC é a métrica mais importante na análise e ajuda a complementar o valor informado pela acurácia. O valor da curva foi de 0,92 e pode ser observado na Figura 2. Pode-se observar também, nessa imagem, a linha randômica, que mostra quando o modelo não consegue prever a categoria, atuando como um modelo de adivinhação.

A área abaixo da curva ROC (AUC) está associada ao poder discriminante da análise de avaliar o modelo, pois quanto maior a área, maior a assertividade do modelo.

Os valores obtidos para a sensitividade e para a especificidade foram, respectivamente: 0,9181 e 0,8249. Esses valores altos demonstram o nível de acerto do modelo, já que ambos analisam a proporção das vezes que o modelo acertou a variável categórica analisada (has_accepted_answer) — seja negativamente ou positivamente — sobre o número dos valores reais positivos e negativos encontrados na base.

http://dx.doi.org/10.25286/repa.v3i3.942

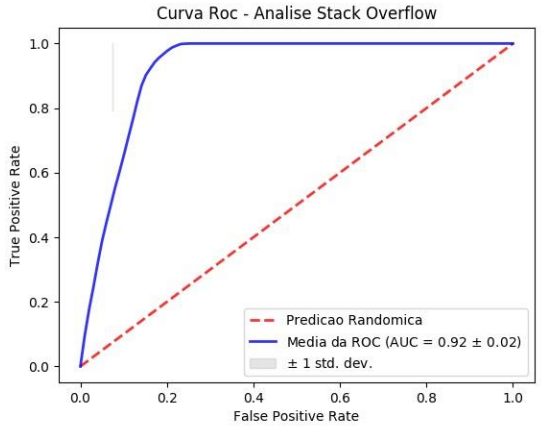

Figura 2: Curva ROC da análise preditiva
Fonte: o autor.

O uso da arquitetura montada executando um modelo de Regressão Logística, a geração do código aplicando o *Beam* e a escolha do *DataflowRunner* para o processamento se mostraram aceitáveis para o desenvolvimento de um modelo preditivo que pode realizar a validação probabilística.

Com base nos resultados obtidos, pode-se confirmar que o modelo gerado para realizar a análise preditiva nos dados coletados teve sucesso, e, com isso, foi possível prever se uma pergunta no fórum teria ou não uma resposta marcada como.

8 Projetos Futuros

A análise descrita no artigo poderia se beneficiar de um balanceamento da base para a verificação da existência de uma melhora nas métricas, e poderia ser feita uma análise com rede neural. Gerando um novo modelo que se beneficiaria da estrutura paralela que este tipo de análise apresenta e da sua habilidade de aprender e generalizar.

Esse modelo pode ser empregado para buscar o mesmo tipo de resposta que foi encontrada para a análise da base do *StackOverflow* em outros fóruns de colaboração. Esse tipo de análise pode dar mais conhecimento aos usuários do fórum uma vez que saberão se podem esperar ou não por uma resposta com base nas interações no *post* da sua pergunta.

Assim, seria possível obter um uso prático para o modelo, além de uma aplicação em uma variedade de cenários.

Referências

[1] STACK OVERFLOW - Where Developers Learn, Share, & Build Careers. Disponível em: < https://stackoverflow.com/>. Acesso em: 18 abr. 2018.

[2] Computação em nuvem, serviços de hospedagem e APIs do Google. Disponível em: <https://cloud.google.com/>. Acesso em: 18 abr. 2018.

[3] PRAMANA, Setia et al. Big data for government policy: Potential implementations of bigdata for official statistics in Indonesia. In: INTERNATIONAL WORKSHOP ON BIG DATA AND INFORMATION SECURITY, 2017, Jakarta. **Proceedings**...Jakarta: IEEE, 2017. p. 17-21.

[4] AGNEESWARAN, Vijay S. **Big Data Analytics Beyond Hadoop**. New Jersey: Perason, 2014.

[5] ZHAO, Disheng. Performance comparison between Hadoop and HAMR under laboratory environment. **Procedia Computer Science**, v. 111, p. 223–229, 2017.

[6] AGNEESWARAN, Vijay S. **Big Data Analytics Beyond Hadoop**: Real-Time Applications with Storm, Spark, and More Hadoop Alternatives. FT Press, 2014.

[7] GONZALEZ, Jose Ugia; KRISHNAN, S. P. T. **Building Your Next Big Thing with Google Cloud Platform: A Guide for Developers and Enterprise Architects**. New York: Apress, 2015.

[8] AKIDAU, Tyler; PERRY, Frances. **DATAFLOW/Beam e Spark**: uma comparação de modelo de programação. Apach Beam Commiters, 2016. Disponível em: <https://cloud.google.com/dataflow/blog/dataflow-beam-and-spark-comparison?hl=pt-br>. Acesso em: 9 mar. 2018.

[9] MORGAN, Timothy Prickett. **Google Pits Dataflow Against Spark**. The Next Plataform, 2016. Disponível em: <https://www.nextplatform.com/2016/05/03/google-pits-dataflow-spark/>. Acesso em: 13 mar. 2018.

[10] ONOFRE, J. B.; JAGIELSKI, J. **BeamProposal - Incubator Wiki**. Disponível em: <https://wiki.apache.org/incubator/BeamProposal>. Acesso em: 26 dez. 2017.

[11] AKIDAU, Tyler et al. The Dataflow Model: a Practical Approach to Balancing Correctness, Latency and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing, **Proceedings of the VLDB Endowmen**, v. 8, n. 12, p. 1792-1803, 2015.

[12] SALEM, Farouk. **Comparative Analysis of Big Data Stream Processing Systems**. Aalto University, 2016.

[13] LANEY, D. 3D Data Management: Controlling Data Volume, Velocity, and Variety Application Delivery Strategies. **META group research note**, v. 6, n. 70, p. 1, 2001.

[14] KEPNER, Jeremy et al. Computing on masked data: A high performance method for improving big data veracity. In: HIGH PERFORMACE EXTREME COMPUTING CONFERENCEi, 8., 2014, Massachusetts. **Proceedings...** Massachuttes: IEE**E**,2014. p. 1–6.

[15] LIU, S. M. et al. Big Data A Survey. **Mobile Networks and Applications,** New York 2014.

[16] GRADY, Nancy; CHANG, Wo (ed.) N**IST Big Data Interoperability Framework**: Volume 1, Definitions. Gaithersburg: NIST, 2015. p. 32.

[17] UZUNKAYA, C.; ENSARI, T.; KAVURUCU, Y. Hadoop Ecosystem and Its Analysis on Tweets. **Procedia - Social and Behavioral Sciences**, s.l., v. 195, p. 1890–1897, 2015.

[18] APACHE Hadoop 3.0.0 – HDFS Architecture. Disponível em: <http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html#NameNode_and_DataNodes>. Acesso em: 22 fev. 2018.

[19] MINER, Donald; SHOOK, Adam. **MapReduce design patterns**. Sebastopol: O'Reilly, 2012.

[20] BENGFORT, B.; KIM, J. **Analítica de dados com Hadoop**. São Paulo: Novatec, 2016.
[21] APACHE Spark. Disponível em: <http://spark.apache.org/powered-by.html>. Acesso em: 8 mar. 2018.

[21] KANE, F. **Frank Kane's Taming Big Data with Apache Spark and Python**. Birmingham: Packt Publishing, 2017.

[22] ZAHARIA, M.; CHOWDHURY, M.; DAS, T.; DAVE, A. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing, **Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation**. USENIX Association, 2012.

[23] ZAHARIA, Matei. et al. **Spark**: Cluster Computing with Working Sets. In: Conferece on Hot topics in Cloud Computer, 2., 2010, Boston. **Proceedings**... Boston: USENIX, 2010.

[24] THE WORLD beyond batch: Streaming 102 - O'Reilly Media. Disponível em: <https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-102>. Acesso em: 15 fev 2018.

[25] MODELO de programação do Dataflow. Disponível em: <https://cloud.google.com/dataflow/model/programming-model>. Acesso em: 15 fev. 2018.

[26] PIPELINERUNNER (Google Cloud Dataflow SDK 1.9.1 API). Disponível em: <https://cloud.google.com/dataflow/java-sdk/JavaDoc/com/google/cloud/dataflow/sdk/runners/PipelineRunner>. Acesso em: 14 mar. 2018.

http://dx.doi.org/10.25286/repa.v3i3.942

[27] APACHE Beam. Disponível em: <https://beam.apache.org/>. Acesso em: 15 fev 2018.

[28] TENSORFLOW. Disponível em: <https://www.tensorflow.org/>. Acesso em: 10 mar. 2018.

[29] SCIKIT-LEARN: machine learning in Python. Disponível em: <http://scikit-learn.org/stable/>. Acesso em: 14 mar. 2018.

[30] ABADI, M. et al. TensorFlow: A System for Large-Scale Machine Learning. In: USENIX CONFERENCE ON OPERATING SYSTEMS DESIGN AND IMPLEMENTATION, 12., 2016, Geórgia. **Proceedings**... Geórgia: Berkeley, 2016. p. 265–284.

[31] MARTIN, A. et al. **TensorFlow**: Large-scale machine learning on heterogeneous systems. Disponível em: < https://www.tensorflow.org/>. Acesso: 14 mar. 2018.

[32] GÉRON, A. **Hands-On Machine Learning with Scikit-Learn and TensorFlow**. Sebastopol: O'Reilly Media, 2017.

[33] ABRAHAM, A. et al. Machine Learning for Neuroimaging with Scikit-Learn. F**rontiers in neuroinformatics**, v.8, p.14, 2014.

[34] GOOGLE BigQuery. Disponível em: <https://bigquery.cloud.google.com/dataset/bigquery-public-data:stackoverflow>. Acesso em: 23 fev. 2018.

[35] CONJUNTOS de dados públicos do Google BigQuery. Disponível em: <https://cloud.google.com/bigquery/public-data/>. Acesso: 14 mar. 2018.

[36] PYNE, S.; PRAKASA; B.L.S.; RAO, S. B. **Big Data Analytics**: Methods and Applications. New Delhi: Springer, 2016.

[37] TRAIN/TEST Split and Cross Validation in Python – Towards Data Science. Disponível em: <https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6>. Acesso em: 15 fev. 2018.

[38] KING, Gray; ZENG, Langche. Logistic Regression in Rare Events Data. **Political Analysis**, v. 9, n. 2, p. 137–163, 2001.

[39] PRATI, R. C.; BATISTA, GEAPA; MONARD, M. C. Curvas ROC para avaliação de classificadores. **IEEE Lat. Am. Trans.**, s.l., v. 6, n. 2, p. 215–222, 2008.

[40] LE BLANC, D. C. **Statistics**: Concepts and Applications for Science. Sudbury: Jones and Bartlett, 2004. v. 2.

[41] HAIDER, M. **Getting Started with Data Science: Making sense of Data with Analytics**, IBM Press, Indianápolis, 2016.

[42] ENRIQUE, G.; PRADO, D. A.; BATISTA, A. Pré-processamento de Dados em Aprendizado de Máquina Supervisionado. **Instituto de Ciências Matemáticas e de Computação**, São Carlos, 2003.

[43] GRIGOREV, A. **Mastering Java for data science** : building data science applications in Java. Birmingham: Packt Publishing, 2017.

[44] GOOGLE Dataflow And Apache Beam (I). Disponível: <http://www.datio.com/development/google-dataflow-and-apache-beam-i/>. Acesso em: 14 mar. 2018.