# INTEGRAÇÃO DA ANÁLISE DE IMAGEM BASEADA EM OBJETOS (OBIA) E APRENDIZAGEM DE MÁQUINA EM AMBIENTE DE COMPUTAÇÃO DISTRIBUÍDA

## *INTEGRATION OF OBJECT BASED IMAGE (OBIA) ANALYSIS AND MACHINE LEARNING IN A DISTRIBUTED COMPUTER ENVIRONMENT*

Rodrigo Rodrigues Antunes

**Resumo**

O objetivo principal deste artigo é apresentar novo modelo de integração de sistema de classificação de imagem de sensoriamento remoto com biblioteca de aprendizagem de máquina em ambiente de computação distribuída (nuvem). Nesse caso específico, são apresentadas e utilizadas ferramentas de computação em nuvem, de código livre, em área específica da Geociências, Análise de Imagem Baseada em Objetos (OBIA). As principais ferramentas e serviços de nuvem utilizados para esta pesquisa foram: InterCloud, AWS (Amazon Web Services), Hadoop HDFS, Apache Spark, Apache Hive e Apache Zeppelin. Destacam-se o InterCloud, que é uma plataforma distribuída para interpretação automática de imagens baseada em Hadoop, Pig e modelo de programação MapReduce, o Apache Zeppelin, que é um *notebook web* para leitura de códigos e análise interativa (gráficos), o Apache Hive para manipulação de tabelas e o MLib Spark para disponibilizar biblioteca de aprendizagem de máquina. Com o método de integração apresentado foi possível, em tempo real, expor os dados, fazer consultas SQL e criar modelos estatísticos de interpretação e conhecimento em ambiente de computação distribuída. Foi possível, por exemplo, gerar gráficos de dispersão, identificar limites e valor médio de *pixel* de cada classe/atributo do modelo de classificação.
**Palavras-chave:** OBIA; InterCloud; Computação em nuvem; Aprendizagem de máquina.

***ABSTRACT***

*The main objective of this article is to present a new integration model of a remote sensing image classification system through the learning library machine in a distributed computing environment(cloud). In this specific case, we present and use free-code cloud computing tools in a specific area of Geosciences, Object-Based Image Analysis (OBIA). The main cloud tools and services used for this research were: InterCloud, AWS (Amazon Web Services), Hadoop HDFS, Apache Spark, Apache Hive and Apache Zeppelin. We highlight InterCloud, which is a distributed platform for automatic image interpretation based on Hadoop, Pig and MapReduce programming model, Apache Zeppelin, which is a web notebook for code reading and interactive analysis (graphics), Apache Hive for tables manipulation and MLib Spark to make learning library machine available. The integrating method presented made it possible, in real time, to expose the data, make SQL queries and create statistical models of interpretation and knowledge in a distributed computing environment. It was possible, for example, to generate scatter plots, to identify boundaries, to identify the average pixel value of each class / attribute of the classification model.*
***Keywords****: OBIA. InterCloud. Cloud Computing. Machine Learning.*

## INTRODUÇÃO

Atualmente, percebe-se evoluções e transformações contínuas de várias áreas em curto espaço de tempo. Por exemplo, a medicina e o combate a novas doenças, a tecnologia e o tratamento de grande conjunto de dados (Big Data), o agronegócio sendo desafiado pelas mudanças climáticas e novas pragas, a educação disponibilizando e adaptando-se ao modelo de ensino à distância, dentre outros. Junto com as transformações, surge a necessidade de ferramentas eficientes de pesquisas, solução de problemas ou aprimoramento de cada área específica.

No que concerne a Geociências, destaca-se o avanço dos sistemas sensores de alta resolução espacial e espectral, atualmente, assumindo relevância na representação de informações sobre a superfície terrestre (LIU, 2015), vinculados à utilização da Análise de Imagem Baseada em Objeto (OBIA – Object Based Image Analysis), reconhecida por Blaschke et al. (2014) como novo paradigma em sensoriamento remoto, sendo possível a identificação mais precisa dos alvos urbanos/intra-urbanos. A diversidade e a realidade dos objetos inferidas por meio desses sensores podem ser mais bem interpretadas com o uso de OBIA, devido à possibilidade de se introduzir o conhecimento do analista e atribuir vários parâmetros além da já consagrada resposta espectral.

De acordo com Blaschke, Burnett e Pekkarinen (2004), vários dos aspectos que envolvem a geo-informação não podem ser obtidos por informação somente de *pixel*, conforme ocorre nas classificações *pixel* a *pixel*, mas em um contexto de vizinhança dos objetos de interesse. OBIA permite a inclusão de informações adicionais para orientar os processos de classificação e modelagem. Essas informações podem ser: refletância média do objeto; desvio-padrão da refletância do objeto; valor máximo de *pixel* do objeto, valor mínimo do *pixel* e média de refletância; área, forma e textura dos objetos; localização de objetos em relação a outros objetos, constituindo uma combinação complexa de parâmetros individuais (BLASCHKE, 2013) a ser inserida em regras de decisão a partir da construção de redes semânticas e árvores de decisão.

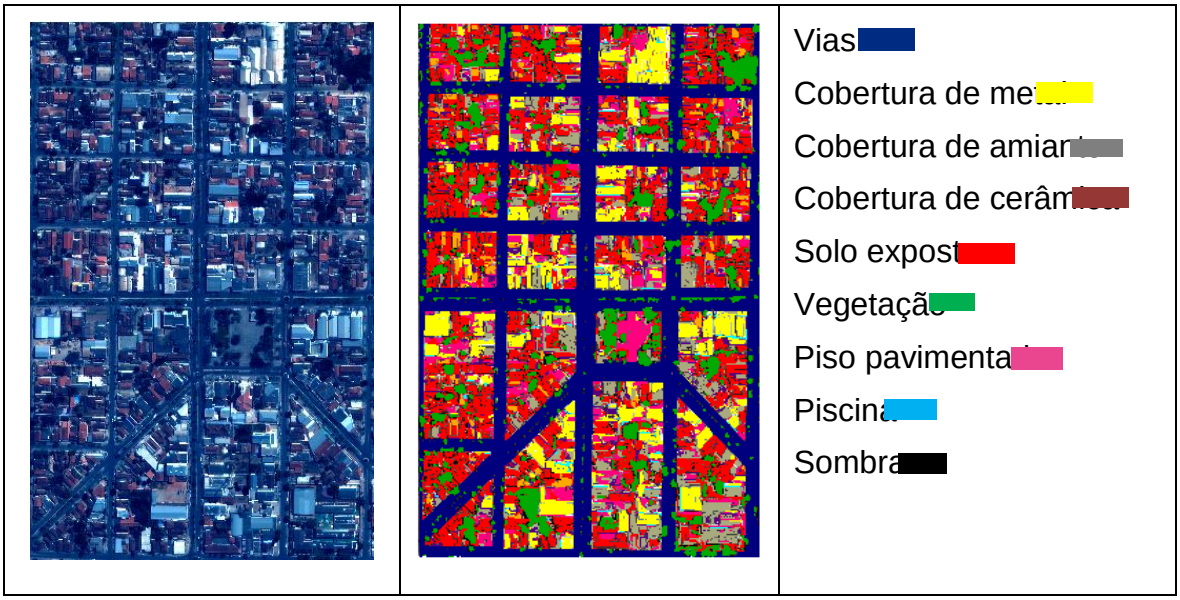
Contudo, entra-se em uma era de grande volume de dados (LIU, 2015), e *petabytes* de dados são gerados diariamente (TSAI et al., 2016). Com o crescimento do volume de dados espaciais em curto espaço de tempo e a necessidade de classificar esses dados para obtenção de visão espacial para as diversas aplicações necessárias à tomada de decisões, busca-se desenvolver mecanismos que expandam a utilização de OBIA em ambiente de computação distribuída.

O InterCloud é uma plataforma livre para interpretação de imagens projetadas para executar *grids* de computadores (*cluster* físico ou infraestrutura de computação em nuvem) e permite a interpretação de grandes conjuntos de dados de sensoriamento remoto de forma eficiente (ANTUNES, 2015). Esse sistema possui operadores para aplicar OBIA na classificação.

É importante destacar que o InterCloud é uma evolução do InterIMAGE, *software* de código aberto, escrito em C++, desenvolvido em 2010 e faz parte de um projeto de cooperação científica internacional liderado pelo Laboratório de Visão Computacional do Departamento de Engenharia Elétrica da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) e pelas divisões de Processamento de Imagens e de Sensoriamento Remoto do Instituto Nacional de Pesquisas Espaciais (INPE). O InterIMAGE (2010) baseia-se no aplicativo GeoAIDA, desenvolvido pelo Instituto de Tecnologia de Informação da Universidade de Hannover, na Alemanha, e herdou desse sistema sua característica funcional básica, além de estruturas de

conhecimento e mecanismos de controle. GeoAIDA é a sigla de Geo Automatic Image Data Analyzer e designa, geralmente, projeto com ferramentas para a avaliação de dados de sensoriamento remoto. Especificamente, GeoAIDA consiste em procedimento que, com ajuda de conhecimento de modelos, interpreta automaticamente dados de sensoriamento remoto (PAHL, 2008). O sistema InterIMAGE foi desenvolvido para plataforma *Desktop*, possuindo limitação para interpretação de grande conjunto de dados. É difícil definir o valor exato do tamanho da imagem suportado por esse sistema, pois depende de vários fatores, como a capacidade do processador, a quantidade de memória RAM disponível, a complexidade da rede semântica, entre outros. Mesmo com essas dificuldades, testes mostraram (InterIMAGE, 2010) que o sistema Desktop interpreta com sucesso imagens de até 9 *megapixels* (3.000 x 3.000 *pixels*). Exemplo de classificação por meio do sistema InterIMAGE é apresentado na Figura 1.

**Figura 1 - Resultado de classificação de imagem orbital (área urbana) por meio *software* INTERIMAGE**




Fonte: Antunes et al., 2018.

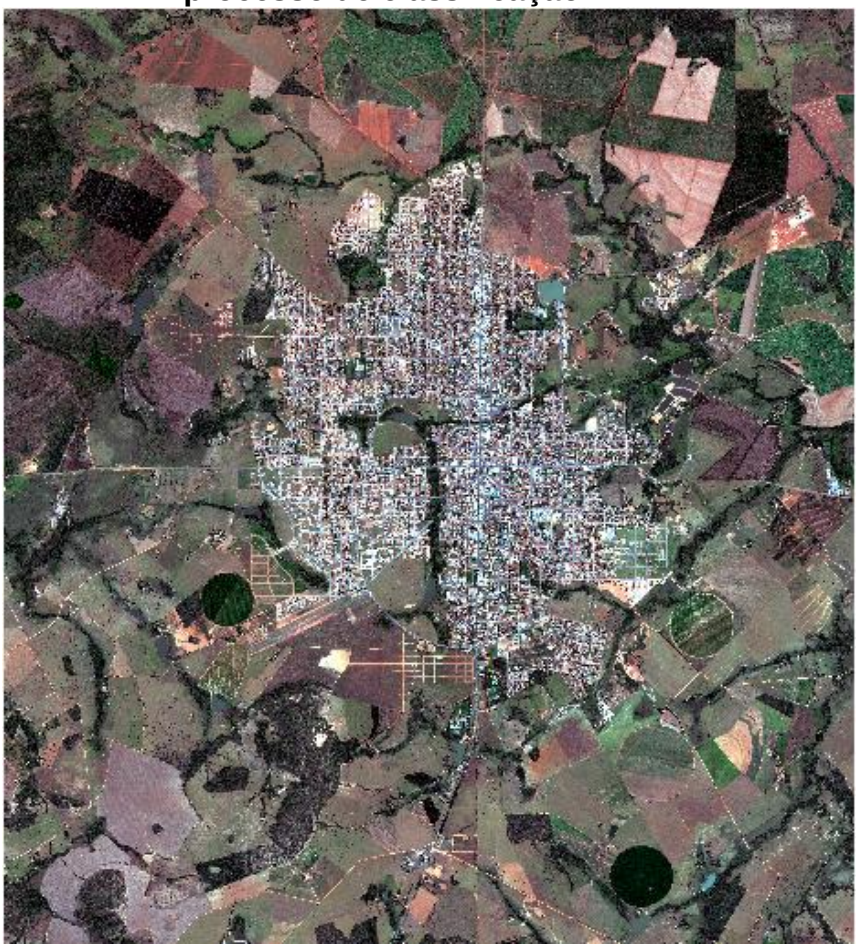
O presente estudo enfoca novo método de integração de sistema distribuído de classificação de imagem orbital com biblioteca de aprendizado de máquina para criar modelos estatísticos de interpretação e conhecimento baseado em OBIA em ambiente de computação distribuída. Para a implementação do método foram utilizadas a plataforma distribuída de interpretação de imagem InterCloud (*open source*) e a biblioteca de aprendizagem de máquina MLlib Spark. O método de classificação utilizado foi de árvore de decisão. O *framework* Apache Hive permitiu criar e manipular tabelas virtuais em *cluster* e o Apache Zeppelin, que é um *notebook web*, permitiu visualizar e modelar os dados com valores de *pixels* por meio de gráficos.

## MATERIAL E MÉTODO

A imagem utilizada para classificação é do município de Goianésia, localizado no centro-norte do estado de Goiás, com população de 65.767 habitantes. A cidade está localizada a 168 km da capital estadual, Goiânia, e a 265 km de Brasília, capital

do Distrito Federal. A imagem de 2013 é do sensor GeoEye-1 e tem 3 GB, 103 km³, 19.404 por 21.360 *pixels*, resolução de 0,5 m e quatro faixas espectrais (Figura 2).

**Figura 2 - Imagem do sensor GEOEYE-1 utilizada como dado de entrada no processo de classificação**



Fonte: Departamento de Geoprocessamento. Município de Goianésia, Goiás.

O conjunto de dados utilizado neste trabalho está armazenado em repositório HDFS Hadoop e é oriundo de processo de segmentação multiresolução distribuída (HAPP, 2016) e extração de características (classes e atributos) espectrais e morfológicas. Para Brites et al. (2012), a segmentação de imagens consiste em processo de agrupamento de *pixels* que possuem características semelhantes. É a divisão de imagem em regiões homogêneas, separadas em espaço contínuo (BLASCHKE; BURNETT; PEKKARINEN, 2004).

Os dados, com valores de *pixels*, foram utilizados para classificação de árvore de decisão em ambiente de computação totalmente distribuído e em tempo real.

Para a implementação foram utilizados os pacotes de *softwares* e serviços apresentados no Quadro 1.

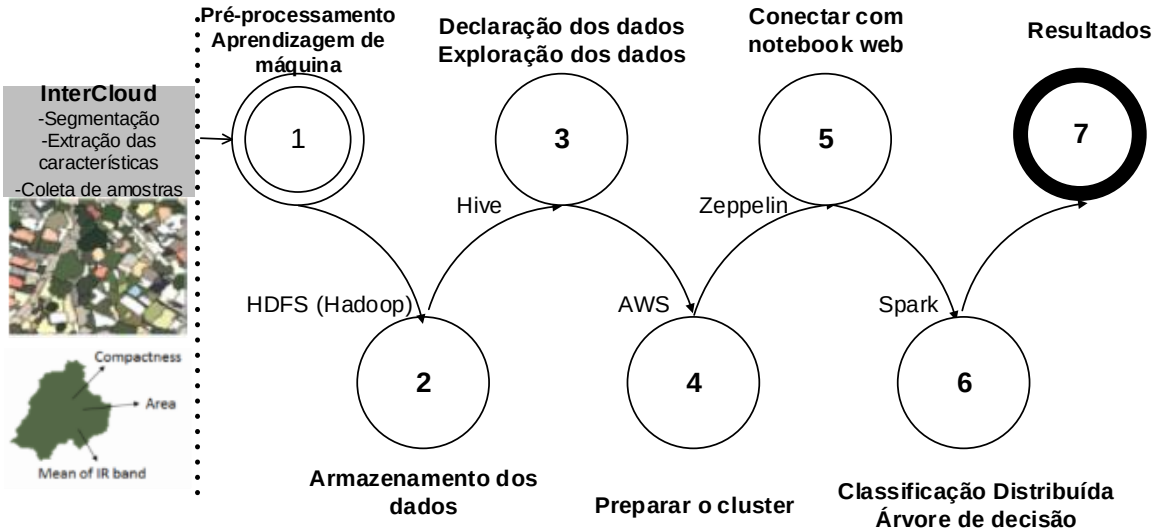
**Quadro 1 - Ferramentas de computação distribuída utilizadas na pesquisa**

| | |
|---|---|
| | InterCloud: Plataforma distribuída para interpretação automática de imagens baseada em objetos. Nesse sistema foram executadas a segmentação distribuída e a extração das características das feições que serão mapeadas. |
| amazon web services | AWS - Amazon Web Services: Serviços de computação em nuvem com armazenamento escalável (S3), servidores virtuais (EC2) e Amazon Elastic MapReduce (EMR), tais como Hadoop, Spark, Hive e outros serviços. |
| PuTTY | PuTTy: *Software* de acesso via terminal. Suporta SSH (Secure Shell), acesso remoto de modo seguro, com criptografia. |
| hadoop HDFS | Hadoop-HDFS: Sistema de arquivos distribuído, para armazenamento de dados em *cluster* (sob demanda). |
| APACHE Spark | Apache Spark: Sistema de computação em *cluster*, com foco em velocidade. Possui biblioteca de aprendizado de máquina na nuvem (MLlib). |
| Apache HIVE | Apache Hive: Sistema para gerenciamento de conjunto de dados no repositório em ambiente distribuído (*in the Cloud*). |
| Apache Zeppelin | Apache Zeppelin: *notebook web* que permite análise interativa por meio de gráficos no ambiente distribuído. |

Fonte: Elaborado pelo autor.

A Figura 3 apresenta o fluxo de trabalho desenvolvido para esta pesquisa.

**Figura 3 - Esquema das etapas do trabalho**




Fonte: Elaborado pelo autor.

*PRÉ-PROCESSAMENTO (APRENDIZAGEM DE MÁQUINA)*

O conjunto de dados (amostras) resultado do processo de segmentação distribuída e extração de características executado no InterCloud foi convertido em **.csv**, que é o formato a ser lido pelo classificador.

O propósito do pré-processamento é transformar os dados de entrada brutos em formato apropriado para as análises subsequentes. Os passos envolvidos no pré-processamento de dados incluem a fusão de dados de múltiplas fontes, a limpeza dos dados para remoção de ruídos, observações duplicadas, a seleção de registros e características que sejam relevantes à tarefa de mineração de dados. Por causa das muitas formas por meio das quais os dados podem ser coletados e armazenados, o pré-processamento de dados talvez seja o passo mais trabalhoso e demorado no processo geral da descoberta de conhecimento (TAN; STEINBACK; KUMAR, 2005).

*ARMAZENAMENTO DISTRIBUÍDO DOS DADOS*

O conjunto de dados, em formato **.csv,** foi enviado para diretório na nuvem (HDFS) e conta com 11 classes e 24 atributos (Quadro 2).

**Quadro 2 - Classes e atributos**

| Classes | Característica espectral | Característica morfológica |
|---|---|---|
| Amianto | Brilho, média dos valores das bandas 1, 2, 3 e 4, valor máximo do pixel das bandas 1, 2, 3 e 4, valor mínimo do pixel das bandas 1, 2, 3 e 4, razão das bandas 1, 2, 3 e 4, divisão da banda 4 por banda 1, e divisão da banda 4 por banda 3 | Área, ângulo, compacidade, retângulo mínimo e círculo. |
| Vias | | |
| Solo exposto | | |
| Cerâmica escura | | |
| Cerâmica clara | | |
| Metálica | | |
| Piso Pavimentado | | |
| Piscina | | |
| Vegetação arbórea | | |
| Vegetação rasteira | | |
| Sombra | | |

Fonte: Elaborado pelo autor.

*DECLARAÇÃO DOS DADOS*

O Apache Hive foi desenvolvido pelo Facebook em 2008 e é baseado em Hadoop, projetado para permitir fácil síntese de dados, consulta e análise *ad-hoc* de grandes volumes de dados (APACHE HIVE, 2019). A linguagem de consulta do Hive é a DDL (Data Definition Language). Para as consultas necessárias neste trabalho, foram declarados, conforme Quadro 3, cada atributo e classes disponíveis no armazenamento distribuído (HDFS).

**Quadro 3 - Declaração dos atributos e classes para criar tabela no Hive**

| Atributos | Declaração | Descrição |
|---|---|---|
| Brilho, média dos valores das bandas 1, 2, 3 e 4, valor máximo do pixel das bandas 1, 2, 3 e 4, valor mínimo do pixel das bandas 1, 2, 3 e 4, razão das bandas 1, 2, 3 e 4, divisão da banda 4 por banda 1, e divisão da banda 4 por banda 3, área, ângulo, compacidade, retângulo mínimo e círculo. | FLOAT | Representa números em notação em precisão simples de 32 bits em conformidade com a norma IEEE 754-1985. |
| Classes | STRING | Sequência de caracteres |

Fonte: Elaborado pelo autor.

*Preparação dos cluster e exposição dos dados*

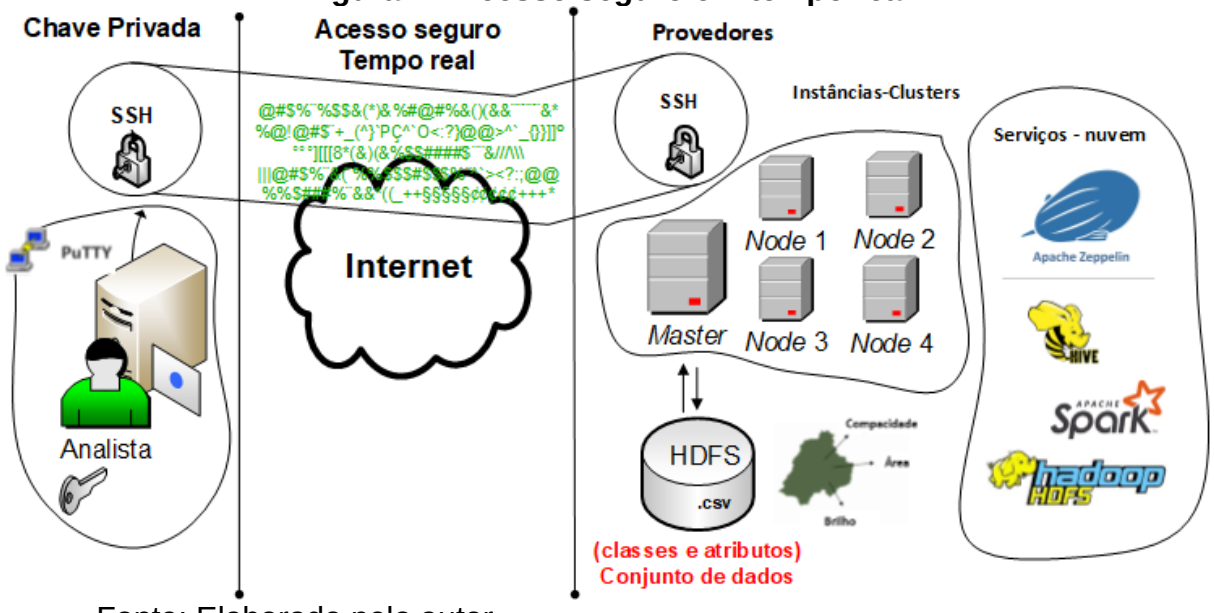
Foram preparados *clusters* para execução dos serviços Hadoop (versão 2.7.3), Spark (versão 2.1.0), Hive (versão 2.1.1) e Zeppelin (versão 0.7.0).

Os recursos utilizados foram:

a) *cluster* com 2 *nodes*;
b) processadores Intel Xeon E5-2670 v2 - 8 vCPU (CPU virtual);
c) armazenamento em SSD (alto desempenho de E/S) – 80 GB;
d) memória – 15 GiB.

Não foram realizados procedimentos de *upload* e/ou *download*. Os procedimentos foram executados na nuvem, em tempo real. Visando realizar acesso seguro com o *cluster*, foi criado um par de chaves de criptografia. O acesso no *node master* foi por meio do protocolo de criptografia SSH e o sistema de acesso de terminal PuTTY.

**Figura 4 - Acesso seguro em tempo real**




Fonte: Elaborado pelo autor.

*CONECTANDO COM NOTEBOOK WEB*

Para visualizar no modo gráfico os dados com valores de *pixels* classificados, foi criada conexão com *cluster* por meio de interface *web* do Apache Zeppelin (APACHE ZEPPELIN, 2019). Iniciado seu desenvolvimento em 2013 pela comunidade *open source*, o Apache Zeppelin é uma ferramenta colaborativa de análise e visualização de dados para sistemas de processamento de dados distribuídos.

*CLASSIFICAÇÃO DISTRIBUÍDA "APRENDIZAGEM DE MÁQUINA"*

A aprendizagem de máquina distribuída refere-se aos algoritmos de aprendizagem de máquina (multi-node) e sistemas projetados para melhorar o desempenho e aumentar a precisão e escala para tamanhos maiores de dados de entrada (GALAKATOS et al., 2014). Para Peteiro-Baral e Guijarro-Berdinass (2013), a aprendizagem de máquina pode tirar proveito da computação distribuída para gerenciar grandes volumes de dados ou para aprender sobre dados inerentemente distribuídos.

O método de classificação utilizado neste trabalho foi de árvore de decisão, por meio da biblioteca de aprendizagem de máquina MLlib do Apache Spark, desenvolvido em 2010 pela AMPLab da Universidade da Califórnia, Berkeley (MENG et al., 2016). O Apache Spark é um *framework* para computação em *cluster* com foco em alta velocidade e fornece APIs de alto nível em Java, Scala, Python e R. Além da MLlib, o Apache Spark suporta conjunto de ferramentas de alto nível, incluindo SQL - Structured Query Language (APACHE SPARK, 2019).

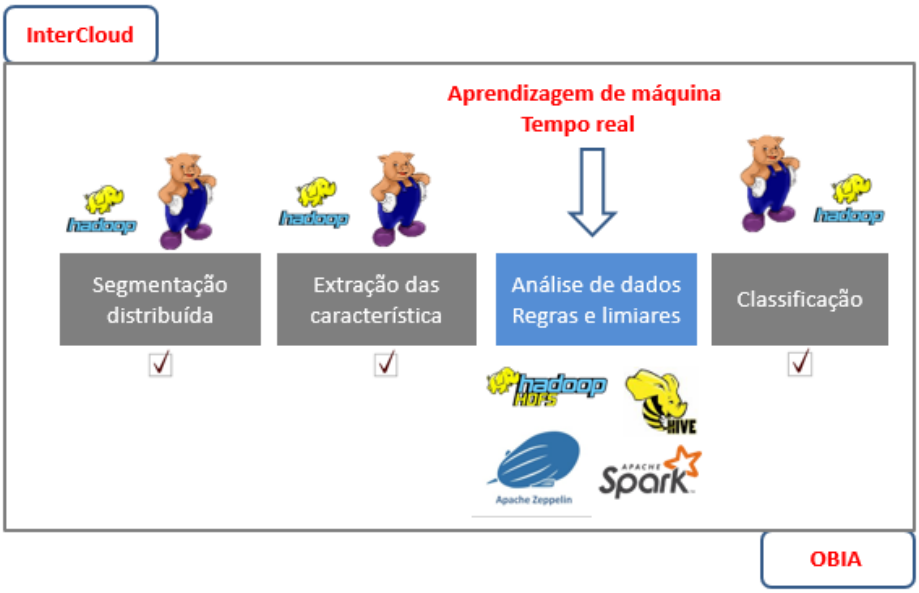
## RESULTADOS E DISCUSSÃO

Atualmente, os métodos de classificação no InterCloud são implementados usando funções do WEKA (Waikato Environment for Knowledge Analysis) para construção de árvores de decisão e baseiam-se unicamente na execução de algoritmos de aprendizagem de máquina. Não há modelo de interpretação/conhecimento nem visualização gráfica dos dados.

Com o método proposto, novos *frameworks* foram suportados e integrados no processo de classificação: Apache Hive, Apache Spark e Apache Zeppelin.

A Figura 5 apresenta o novo método de classificação (Aprendizagem de Máquina) para interpretação baseada no conhecimento, proposto para integrar processo de classificação de imagem no InterCloud.

**Figura 5 - Método proposto de integração do Intercloud com biblioteca de aprendizagem de máquina do Spark Mllib**




Fonte: Elaborado pelo autor.

O Hive permitiu, em tempo real, criar nova tabela em *cluster*, denominada obia_hive, com classes e atributos oriundos do conjunto de dados em formato **.csv** armazenado em HDFS. A declaração e instrução utilizadas são apresentadas no Quadro 4.

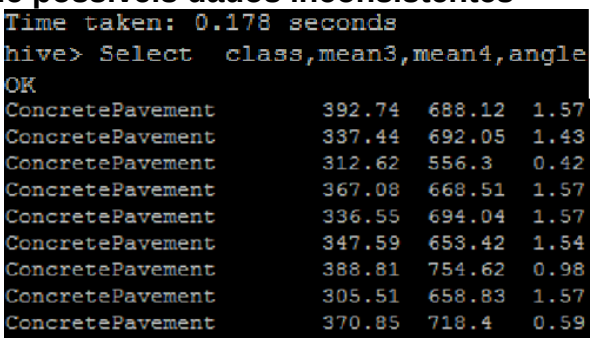
**Quadro 4 - Declaração e instrução para criar tabela virtual OBIA_HIVE**

```
CREATE EXTERNAL TABLE obia_hive
(compactness FLOAT,
brightness FLOAT,
area FLOAT,
minPixVal3 FLOAT,
minPixVal2 FLOAT,
minPixVal1 FLOAT,
roundness FLOAT,
bandDiv43 FLOAT,
bandDiv41 FLOAT,
minPixVal4 FLOAT,
ratio2 FLOAT,
ratio1 FLOAT,
rectangle FLOAT,
mean3 FLOAT,
mean4 FLOAT,
angle FLOAT,
mean1 FLOAT,
ratio4 FLOAT,
ratio3 FLOAT,
mean2 FLOAT,
maxPixVal1 FLOAT,
maxPixVal4 FLOAT,
maxPixVal2 FLOAT,
maxPixVal3 FLOAT,
class STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
LOCATION 's3n://icpunb/interimage/DataMining/';
 MSCK REPAIR TABLE obia_hive;
```

Fonte: Elaborado pelo autor.

Os valores, em *pixels*, foram disponibilizados para visualização em forma de tabela e uma primeira checagem nos dados para validação pode ser feita pelo analista. A Figura 6 apresenta exemplo da seleção da classe *ConcretePavement* e atributos da tabela **obia_hive**. É possível analisar, na tabela, possíveis "ruídos" e artefatos no conjunto de dados e também checar valores inconsistentes.

**Figura 6 - Seleção das classes e atributos da tabela OBIA_HIVE para checagem de possíveis dados inconsistentes**

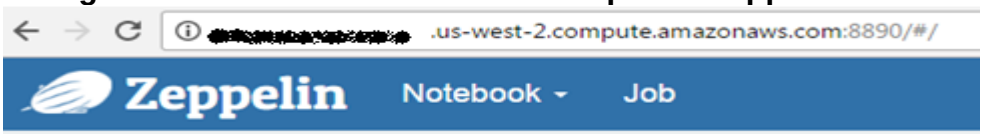


```
Time taken: 0.178 seconds
hive> Select  class,mean3,mean4,angle
OK
ConcretePavement        392.74  688.12  1.57
ConcretePavement        337.44  692.05  1.43
ConcretePavement        312.62  556.3   0.42
ConcretePavement        367.08  668.51  1.57
ConcretePavement        336.55  694.04  1.57
ConcretePavement        347.59  653.42  1.54
ConcretePavement        388.81  754.62  0.98
ConcretePavement        305.51  658.83  1.57
ConcretePavement        370.85  718.4   0.59
```

Fonte: Elaborado pelo autor.

O Zeppelin foi conectado ao *cluster* para disponibilizar interface gráfica de interpretação de comandos do Spark e visualização dos resultados da classificação. A conexão foi feita por meio de extensão FoxyProxy do navegador Firefox, que permitiu conexão personalizada direto com o *cluster* via http (Hypertext Transfer Protocol). A porta utilizada para acesso ao Zeppelin é 8890, conforme apresentado na Figura 7.

**Figura 7 - Conexão do *cluster* com Apache Zeppelin**




Fonte: Elaborado pelo autor.

Foi utilizada a biblioteca de aprendizagem de máquina *Spark.mllib* para classificação de árvores de decisão. A implementação permitiu o treinamento distribuído com várias instâncias. O algoritmo de aprendizagem de máquina utilizado no código está disponível na MLlib por meio de APIs (APACHE SPARK, 2017). Foi possível combinar vários algoritmos em um único fluxo de trabalho (*Pipeline*).

O código criado na linguagem Scala (Figura 8), suportado pelo Spark, permitiu acessar os registros do conjunto de dados na tabela obia_hive, em *cluster* e em tempo real.

**Figura 8 - Código utilizado da biblioteca SPARK.MLLIB para treinamento dos dados oriundos da tabela OBIA_HIVE**

```
Line 1 - import org.apache.spark.ml.classification.DecisionTreeClassifier
Line 2 - import org.apache.spark.ml.feature.{StringIndexer, IndexToString, VectorIndexer, VectorAssembler}
Line 3 - import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
Line 4 - import org.apache.spark.sql.functions._
Line 5 - import org.apache.spark.sql.Row
Line 6 - import org.apache.spark.sql.types._
Line 7 - val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)
Line 8 - val dfraw = sqlContext.sql("select compactness,brightness,area,minPixVal3,minPixVal2,minPixVal1,
roundness,bandDiv43,bandDiv41,minPixVal4,ratio2,ratio1,rectangle,class,mean3,mean4,angle,mean1,ratio4,ratio3,
mean2,maxPixVal1,maxPixVal4,maxPixVal2,maxPixVal3 from obia_hive")
Line 9 - val df = dfraw.select(
Line 10 - $"compactness".as("compactness").cast(FloatType),
Line 11 - $"brightness".as("brightness").cast(FloatType),
Line 13 - $"area".as("area").cast(FloatType),
Line 14 - $"minPixVal3".as("minPixVal3").cast(FloatType),
Line 15 - $"minPixVal2".as("minPixVal2").cast(FloatType),
Line 16 - $"minPixVal1".as("minPixVal1").cast(FloatType),
Line 17 - $"roundness".as("roundness").cast(FloatType),
Line 18 - $"bandDiv43".as("bandDiv43").cast(FloatType),
Line 19 - $"bandDiv41".as("bandDiv41").cast(FloatType),
Line 20 - $"minPixVal4".as("minPixVal4").cast(FloatType),
Line 21 - $"ratio2".as("ratio2").cast(FloatType),
Line 22 - $"ratio1".as("ratio1").cast(FloatType),
Line 23 - $"rectangle".as("rectangle").cast(FloatType),
Line 24 - $"class".as("class").cast(StringType),
Line 25 - $"mean3".as("mean3").cast(FloatType),
Line 26 - $"mean4".as("mean4").cast(FloatType),
Line 27 - $"angle".as("angle").cast(FloatType),
Line 28 - $"mean1".as("mean1").cast(FloatType),
Line 29 - $"ratio4".as("ratio4").cast(FloatType),
Line 30 - $"ratio3".as("ratio3").cast(FloatType),
Line 31 - $"mean2".as("mean2").cast(FloatType),
Line 32 - $"maxPixVal1".as("maxPixVal1").cast(FloatType),
Line 33 - $"maxPixVal4".as("maxPixVal4").cast(FloatType),
Line 34 - $"maxPixVal2".as("maxPixVal2").cast(FloatType),
Line 35 - $"maxPixVal3".as("maxPixVal3").cast(FloatType)
Line 36 - )
Line 35 - df.printSchema()
Line 35 - df.show(510)
```

Fonte: Elaborado pelo autor.

Algoritmos e tarefas de avaliação necessários para compor o fluxo do trabalho (APACHE SPARK, 2017) foram importados do MLlib Spark, conforme apresentados nas linhas de 1 a 6 e descritos na Figura 9.

**Figura 9 - Algoritmos e tarefas importados do Mllib Spark**

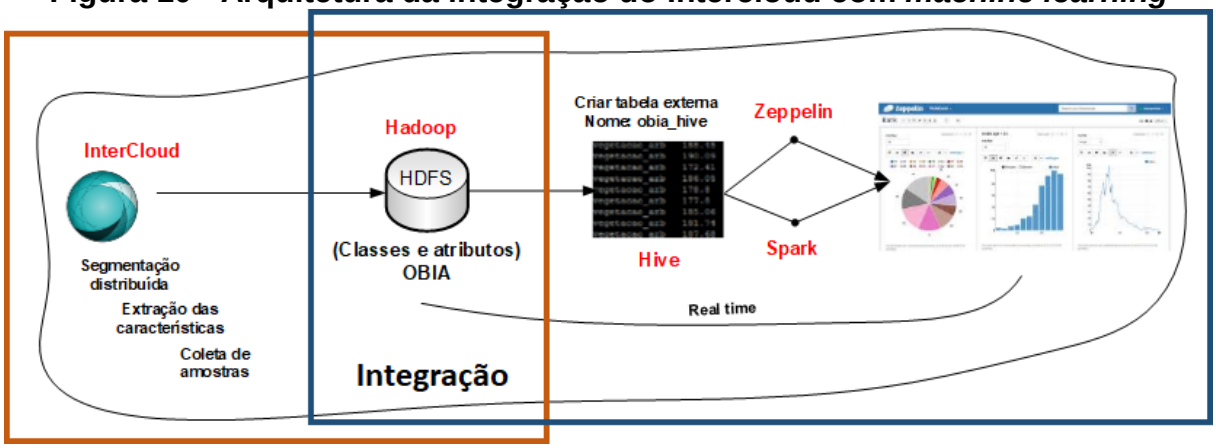
| |
|---|
| **Line 1 - org.apache.spark.ml.classification.DecisionTreeClassifier** |
| Algoritmo de aprendizagem de árvore de decisão para classificação. Suporta classificação binárias e multiclass, bem como recursos contínuos e categóricos. |
| **Line 2 - org.apache.spark.ml.feature.{StringIndexer, IndexToString, VectorIndexer, VectorAssembler}** |
| Algoritmos para extrair, transformar e selecionar recursos |
| **Line 3 - org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator** |
| Avaliador para classificação multiclass, que espera duas colunas de entrada: previsão e rótulo. |
| **Line 4 - org.apache.spark.sql.functions** |
| Define funções do dataframe |
| **Line 5 - org.apache.spark.sql.Row** |
| Representa uma linha de saída de um operador relacional. |
| **Line 6 org.apache.spark.sql.types._** |
| Tipos de declarações. Exemplo: FloatType |

Fonte: Elaborado pelo autor.

A linha 7 (Figura 8) representa o ponto de entrada (SQLcontext) para trabalhar com dados estruturados (linhas e colunas) no Apache Spark. O SQLcontext permitiu a criação de objetos DataFrame, bem como a execução de consultas SQL. As linhas 8 e 9 (Figura 8) foram selecionados e declarados todos atributos e classes, do tipo FloatType e String, da tabela virtual obia_hive.

Com o código apresentado, foi possível treinar um classificador Decision Tree e gerar valores de *pixels* previstos em cada registro no conjunto de dados. A Figura 10 apresenta a nova arquitetura da integração do InterCloud com aprendizagem de máquina por meio dos *frameworks* Hive, Zeppelin e Spark.

**Figura 10 - Arquitetura da integração do Intercloud com *machine learning***

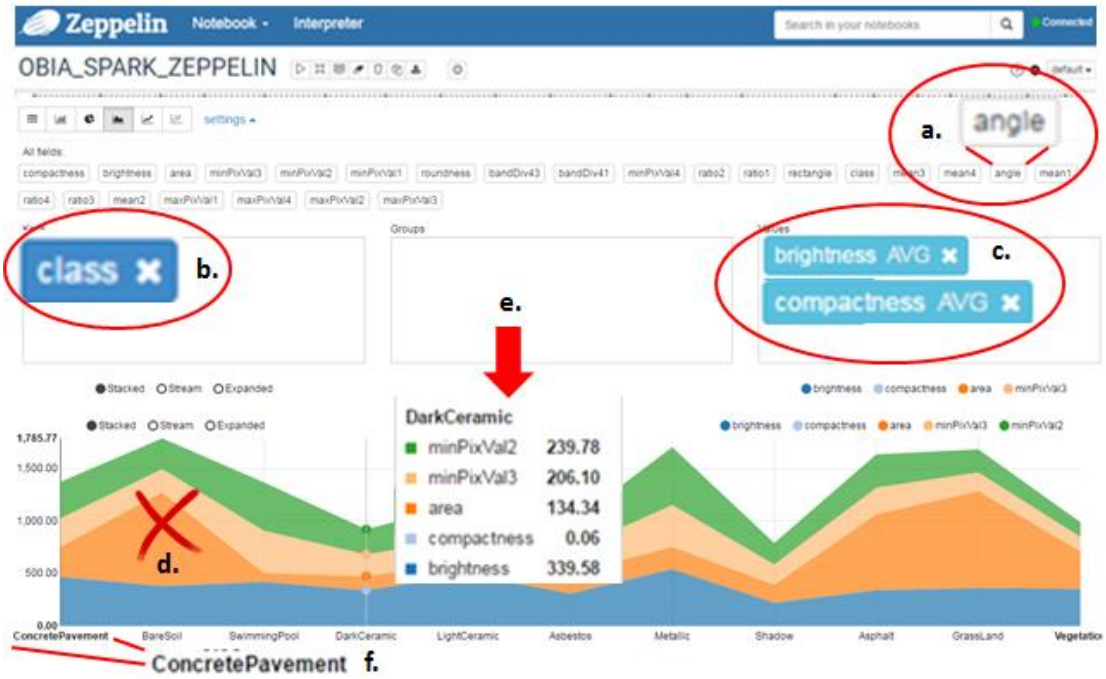



Nota: *Frameworks* utilizados: Hive, Zeppelin e Spark.
Fonte: Elaborado pelo autor

Com os resultados do método proposto é possível criar modelo estatístico de interpretação baseado no conhecimento para classificação da imagem no InterCloud. Pode-se, por exemplo, criar modelo de definição de limites ou mesmo definir valor médio de *pixel* para cada classe/atributo.

Foi possível visualizar, por meio da interface do Zeppelin, a mesma tabela com atributos e classes disponibilizada no Apache Hive e, por meio de interpretador de comando SQL, foram gerados gráficos estatísticos com valores de *pixels*, conforme apresentado no exemplo da Figura 11.

**Figura 11 - Exemplo de gráfico estatístico para modelo de interpretação**




Nota: (a) são relacionados todos os atributos disponíveis; (b) representa a variável alvo; (c) onde são selecionados todos os atributos; (d) e (e) representam o gráfico e os dados.
Fonte: Elaborado pelo autor.

Um *notebook web* criado com nome OBIA_SPARK_ZEPPELIN permitiu visualizar e analisar de maneira interativa os dados por meio de gráficos. Todas as variáveis (classes e atributos) foram listadas no painel (a) e (f). Em (b) é possível observar a variável alvo (*class*) como chave principal. Todas as variáveis foram selecionadas, conforme apresentado em (c), para compor o modelo de aprendizado. Em (d) é apresentado o resultado da classificação em forma de gráfico e os valores podem ser observados em (e).

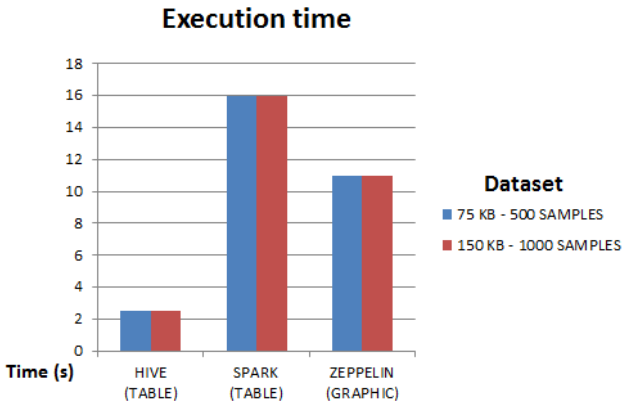
Foi possível simular vários tipos de gráficos com funções agregadas, incluindo *sum, count, average, min, max*. A Figura 12 exemplifica o atributo *brightness* com as funções agregadas disponíveis.

**Figura 12 - Funções (*sum, count, avg, min* e *max*) agregadas no atributo *brightness***


Fonte: Elaborado pelo autor.

O Gráfico 1 e a Tabela 1 mostram que o tempo de processamento é executado em poucos segundos para cada processo do Hive, Spark e Zeppelin. Utilizando-se dois conjuntos de dados, o primeiro com 75 KB e 500 amostras e o outro com 150 KB e 1.000 amostras, foi possível perceber que o tempo de processamento permanece o mesmo para cada processo, mostrando que o recurso de *hardware* disponível, de modo escalável, é suficiente para o processamento dos conjuntos de dados processados. Por permitir processamento muito rápido na execução dos processos, não foram feitos testes com quantidades maiores de instâncias. Considerando dois nós, foi possível obter tempo de processamento de 2,5 segundos para o processo de criar tabela no Hive para os dois conjuntos de dados. Para o processo de importação dos algoritmos e criação de tabela com Spark, o tempo de processamento para os dois conjuntos de dados foi de 16 segundos e para gerar gráficos no Zeppelin (classificação) 11 segundos.

**Gráfico 1 - Tempo de execução dos processos da classificação**




Fonte: Elaborado pelo autor.

**Tabela 1 - Tempo de processamento dos processos no Hive, Spark e Zeppelin**

| DATA | 75 KB - 500 SAMPLES | 150 KB - 1000 SAMPLES |
|---|---|---|
| HIVE (TABLE) | 2,5 | 2,5 |
| SPARK (TABLE) | 16 | 16 |
| ZEPPELIN (GRAPHIC) | 11 | 11 |

Fonte: Elaborado pelo autor.

## CONCLUSÃO E PERSPECTIVAS

Neste trabalho foi apresentado novo método de integração, em tempo real, de plataforma distribuída de classificação de imagem com operadores de OBIA e algoritmo de classificação (*Decision Tree*) de aprendizado de máquina para exposição dos dados, consultas SQL de classes e atributos e criação de modelos estatísticos de interpretação e conhecimento. Com dados, tabelas e gráficos disponíveis por meio de console de interpretação, é possível, por exemplo, o analista de OBIA gerar gráficos de dispersão, identificar limites e valor médio de *pixel* de cada classe/atributo, etc.

A integração (usuário x máquina) ainda depende de um bom conhecimento do analista em razão do uso de diferentes linguagens de programação, tais como DDL do Hive e Scala do Spark/Zeppelin. É necessário o desenvolvimento de interface para inserção e manipulação dos dados, pois até o momento, toda a entrada de dados é realizada por meio de códigos fontes.

Além do Hadoop e pig e funções do WEKA, que são a base do InterCloud, outros *frameworks* como Hive, Spark e Zeppelin foram utilizados para integrar e modelar classificação para OBIA. Outras conexões com outros interpretadores e *frameworks* para futuros projetos também são suportadas, tais como: R, Phyton, PostgreeSQL, Flink e outros.

O processo de integração do InterCloud com Hive, Spark e Zeppelin, em tempo real, foi satisfatório por apresentar agilidade no processo de classificação. Antunes et al. (2016) tiveram tempo de processamento de 44 segundos, com os mesmos 2 *nodes cluster* para classificação de árvore de decisão com InterCloud utilizando funções do WEKA. Além de utilizar procedimentos que retarda o processo, tais como, separar manualmente dados de treinamento e teste e fazer procedimentos de *upload* e *download* (*script pig*), o tempo de processamento foi superior ao do método proposto neste trabalho (11 segundos), conforme apresentado no Gráfico 1 e Tabela 1.

O MLlib do Spark provou ser excelente opção no processo de classificação (OBIA) em ambiente de computação distribuída. Outros métodos de classificação podem ser testados por meio da biblioteca, tais como: Linear Support Vector Machines (SVMs), random forests, gradient-boosted trees e Naive Bayes.

O Apache Zeppelin mostrou ser excelente plataforma para análise visual dos gráficos. Os dados em valores de *pixels* foram disponibilizados para modelagem estatística de interpretação e conhecimento, mas mostraram necessidade de, por exemplo, induzir (visualização) uma árvore de decisão com regras e valores de limiares.

O método de integração, com ferramentas de computação distribuída (Apache Hive, Apache Spark e Apache Zeppelin), apresentado nesta pesquisa, também pode ser utilizado em outras áreas do conhecimento, tais como: medicina, agronegócio, engenharias e outros.

## REFERÊNCIAS

ANTUNES, R. R.; HAPP, P.N. BIAS, E.S.; BRITES R.S. e COSTA, A.O.P. An object-based image interpretation application on cloud computing infrastructure. In: **GEOBIA 2016:** solutions and synergies. Enschede, The Netherlands: University of Twente, Faculty of Geo-Information and Earth Observation (ITC), 2016.

ANTUNES, Rodrigo Rodrigues et al. Análise de Integração de Mineradores de Dados com a Plataforma InterIMAGE–Qual a Melhor Solução?. Revista Brasileira de Cartografia, v. 70, n. 4, p. 1470-1509, 2018.

ANTUNES, Rodrigo Rodrigues et al. Object-Based Analysis for Urban Land Cover Mapping Using the Interimage and the Sipina Free Software Packages. **Boletim de Ciências Geodésicas**, v. 24, n. 1, p. 1-17, 2018.

APACHE HIVE. Getting Started. 2017. Disponível em: <https://hive.apache.org/>. Acesso em: 02 mar. 2019.

APACHE SPARK. Documentation. 2019. Disponível em: <https://spark.apache.org/>. Acesso em: 02 mar. 2019.

APACHE ZEPPELIN. Documentation. 2019. Disponível em: <https://zeppelin.apache.org/>. Acesso em: 02 mar. 2019.

BLASCHKE, T.; BURNETT, C.; PEKKARINEN, A. New contextual approaches using image segmentation for object-based classification. In: JONG, S. M.; VAN DER MEER, F. D. (eds), **Remote sensing image analysis:** induding the spatial domain. Elsevier, Hannover, Germany: Kluver Academic Publishers Dordrecht, 2004, p 211-236.

BLASCHKE, T. et al. Geographic object-based image analysis-towards a new paradigm. **ISPRS Journal of Photogrammetry and Remote Sensing**, v. 87, 2014, p. 180-191.

BLASCHKE, T. Object based image analysis: a new paradigm remote sensing? **ASPRS, Anual Conference**. Baltimore, Maryland, 2013.

FERREIRA, R. S. **InterIMAGE cloud platform:** the architecture of a distributed platform for automatic, object-based image interpretation. [Tese de doutorado em engenharia elétrica pela Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio]. Rio de Janeiro, 2015.

GALAKATOS, A.; CROTTY, A.; KRASKA, T. **Distributed Machine Learning.** Providence RI: Brown University, 2014. Disponível em: <http://cs.brown.edu/~agg/EDBS.pdf>. Acesso em: 03 mar. 2019.

HAPP, P. N. et al. A cloud computing strategy for region-growing segmentation. **IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing**, v. 9, 2016, p. 5294-5303.

INTERIMAGE. Manual do usuário. 2010. Site: <http://www.lvc.ele.pucrio.br/projects/interimage/pt-br/documentacao/> Acessado em março de 2018.

LIU, P. A survey of remote-sensing big data. **Frontiers in Environmental Science**, v. 3, 2015, p. 45.

MENG, X. et al. Mllib: machine learning in apache spark. **Journal of Machine Learning Research**, v. 17, n. 34, 2016, p. 1-7.

PETEIRO-BARRAL, D.; GUIJARRO-BERDIÑAS, B. A survey of methods for distributed machine learning. **Progress in Artificial Intelligence**, v. 2, n. 1, 2013, p. 1-11.

TAN, P. N.; STEINBACH, M.; KUMAR, V. **Introduction to data mining**. Boston-USA: Addison-Wesley Longman Publishing, 2005.

TSAI, C.; LINB, W.; KEC, S. Big data mining with parallel computing: A comparison of distributed and MapReduce methodologies. **The Journal of Systems and Software**, v. 122, 2016, p. 83-92.