UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE CIÊNCIAS, TECNOLOGIAS E SAÚDE DO CAMPUS ARARANGUÁ
DEPARTAMENTO DE COMPUTAÇÃO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Vinícius Camozzato Vaz

Obtenção de Batimetria em Estuários Através da Aplicação de Métodos de Aprendizado de Máquina em Imagens Multiespectrais de Satélites.

Araranguá
2021

Obtenção de Batimetria em Estuários Através da Aplicação de Métodos de Aprendizado de Máquina em Imagens Multiespectrais de Satélites

Machine-Learning Approach for Satellite Derived Bathymetry Using Multispectral Imagery

Vinicius Camozzato Vaz * Ricardo Santacattarina †

Luis Pedro Melo de Almeida ‡ Antonio Henrique da Fontoura Klein §

Antonio Carlos Sobieranski ¶

2021, Maio

Resumo

A Batimetria Derivada de Satélite (*Satellite Derived Bathymetry* - SDA) é um meio eficiente e de baixo custo para obtenção da profundidade da coluna d'água, principalmente pela facilidade de ser aplicada em grandes regiões e também em locais de difícil acesso como zonas rasas de estuários. O presente trabalho foi desenvolvido com o objetivo de aplicar, avaliar e validar a utilização de métodos de aprendizado de máquina para realizar a batimetria por satélite na Baía da Babitonga, um estuário localizado no norte do estado de Santa Catarina - Brasil. Como fonte primária de dados utilizou-se as imagens multiespectrais do satélite Sentinel-2, bandas 1 à 8 relacionadas por data com um levantamento batimétrico monofeixe categoria B de acordo com a NORMAM-25, realizado no local de interesse. Na primeira etapa do trabalho realizou-se a preparação dos dados, etapa esta que envolve a aquisição e processamento inicial, para então prosseguir com uma análise exploratória dos dados das imagens, analisando principalmente o domínio do valor de cada banda e suas respectivas distribuições. Com base no conjunto de dados, as devidas manipulações realizadas e o conjunto pronto para ser utilizado na calibração de modelos, foram propostos alguns algoritmos de *machine-learning* a serem testados e validados de acordo com métricas predefinidas. Os resultados experimentais para o melhor modelo apresentaram como *baseline* um r^2 *score* de 0,78 evoluindo para 0,95 após as etapas de otimização de hiperparâmetros e *feature engineering*. Todos os modelos passaram por etapas de validação avançada, incluindo *cross-validation e* validação de janela móvel, para garantirem a validade dos resultados obtidos. Dentre os algoritmos propostos, os modelos baseados em árvores de decisão se mostraram mais eficazes quando comparados aos demais. Os resultados dos modelos são considerados satisfatórios para estimação da batimetria em ambientes estuarinos similares à Baía da Babitonga, porém, devido ao processo de validação ter sido realizado no mesmo local, a capacidade de extrapolação do modelo para outras regiões não foi verificada.

Palavras-chaves: Batimetria. *Machine-Learning*. Sentinel. Satélite. Reconhecimento de Padrões

*vinicvaz95@gmail.com - Universidade Federal de Santa Catarina, Departamento de Computação

†ricardo.santacattarina@gmail.com - Universidade Federal de Santa Catarina, Departamento de Computação

‡melolp@gmail.com - +Atlantic LVT, Portugal - Edifício LACS Estrada da Malveira da Serra 920, Cascais, Portugal

§ antono.klein@ufsc.br - co-orientador - Universidade Federal de Santa Catarina UFSC - Coordenadoria Especial de Oceanografia

¶a.sobieranski@ufsc.br - orientador - Universidade Federal de Santa Catarina, Departamento de Computação

Obtenção de Batimetria em Estuários Através da Aplicação de Métodos de Aprendizado de Máquina em Imagens Multiespectrais de Satélites

Machine-Learning Approach for Satellite Derived Bathymetry Using Multispectral Imagery

Vinicius Camozzato Vaz * Ricardo Santacattarina †

Luis Pedro Melo de Almeida ‡ Antonio Henrique da Fontoura Klein §

Antonio Carlos Sobieranski ¶

2021, Maio

Abstract

Satellite-derived bathymetry (SDA) is an efficient and low-cost way of measuring the depth of a water column. SDA is easy to apply in large regions that are difficult to access, such as shallow water areas. This work was developed with the objective of evaluating, validating and implementing the use of machine learning methods to perform satellite bathymetry in Babitonga Bay, an estuary located in the north of the state of Santa Catarina, Brazil. The primary data sources were multispectral images from the Sentinel-2 satellite, bands 1 to 8. These images were acquired in a survey category B according to NORMAM-25. The first stage of work was data preparation. This involved the acquisition and initial data processing. This stage was followed by an exploratory analysis of the image data which focused mainly on the value domain of each band and their respective distributions. Based on the dataset, the necessary manipulations performed, and the dataset ready to be used in the calibration of models, some machine-learning algorithms have been proposed to be tested and validated. The experimental results for the best model presented an r^2 score of 0.78 as a baseline, evolving to 0.95 after hyperparameter optimization and feature-engineering stages. All models went through advanced validation stages, including cross-validation and sliding window validation, in order to guarantee the validity of the results obtained. Among the proposed algorithms, models based on decision tree ensembles proved to be more effective. The results of the models are considered satisfactory for estimating bathymetry in estuarine environments similar to Babitonga Bay. However, because the validation process was carried out in the same place, the extrapolation capacity of the models compared to other regions has not yet been verified.

Key-words: Bathymetry. Machine-Learning. Sentinel. Satellite. Pattern Recognition.

*vinicvaz95@gmail.com - Universidade Federal de Santa Catarina, Departamento de Computação

†ricardo.santacattarina@gmail.com - Universidade Federal de Santa Catarina, Departamento de Computação

‡melolp@gmail.com - +Atlantic LVT, Portugal - Edifício LACS Estrada da Malveira da Serra 920, Cascais, Portugal

§ antono.klein@ufsc.br - co-orientador - Universidade Federal de Santa Catarina UFSC - Coordenadoria Especial de Oceanografia

¶a.sobieranski@ufsc.br - orientador - Universidade Federal de Santa Catarina, Departamento de Computação

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Camozzato Vaz, Vinícius
Obtenção de Batimetria em Estuários Através da Aplicação de Métodos de Aprendizado de Máquina em Imagens Multiespectrais de Satélites. / Vinícius Camozzato Vaz ; orientador, Antonio Carlos Sobieranski, coorientador, Antonio da Fontoura Klein, 2021.
38 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Araranguá, Graduação em Engenharia de Computação, Araranguá, 2021.

Inclui referências.

1. Engenharia de Computação. 2. machine-learning. 3. batimetria por satélite. 4. reconhecimento de padrões. 5. inteligência artificial. I. Sobieranski, Antonio Carlos. II. da Fontoura Klein, Antonio. III. Universidade Federal de Santa Catarina. Graduação em Engenharia de Computação. IV. Título.

Documento assinado digitalmente
Fabricio de Oliveira Ourique
Data: 21/05/2021 14:57:09-0300
CPF: 916.167.860-00
Verifique as assinaturas em https://v.ufsc.br

Prof. Fabrício De Oliveira Ourique, Dr.
Coordenador do Curso

Banca Examinadora:

Documento assinado digitalmente
Antonio Carlos Sobieranski
Data: 21/05/2021 15:06:46-0300
CPF: 005.305.809-77
Verifique as assinaturas em https://v.ufsc.br

Prof. Antônio Carlos Sobieranski, Dr.
Orientador

Documento assinado digitalmente
Antonio Henrique da Fontoura Klein
Data: 21/05/2021 15:25:31-0300
CPF: 696.210.710-20
Verifique as assinaturas em https://v.ufsc.br

Prof. Antonio Henrique da Fontoura Klein, Dr.
Co-Orientador

Documento assinado digitalmente
Anderson Luiz Fernandes Perez
Data: 21/05/2021 16:04:46-0300
CPF: 020.803.459-58
Verifique as assinaturas em https://v.ufsc.br

Prof. Anderson Luiz Fernandes Perez, Dr.
Avaliador

Luís Pedro Melo de Almeida

Prof. Luis Pedro Melo de Almeida, Dr.
Avaliador

Prof. Fábio Rodrigues De La Rocha, Dr.
Avaliador Suplente

1 Introdução

Batimetria é a medição da profundidade da coluna d'água em meios fluviais e oceânicos. Esse tipo de levantamento é essencial para realizar a representação e visualização da topografia submersa, prover segurança e eficiência na navegação, auxiliar a indústria da construção civil em obras costeiras, entre outras finalidades socioeconômicas (GAGG, 2016). No entanto, obter uma batimetria exata é um desafio no estudo de processos costeiros. Para estuários esse desafio se intensifica, uma vez que estes consistem em uma zona de transição entre o ambiente marinho e fluvial, e isso torna o ambiente de análise mais complexo e difícil de modelar através de um modelo estatístico. A Baía da Babitonga é um destes complexos ambientes de análise, sendo o maior estuário de Santa Catarina. Além da importância ambiental, esse ambiente também está associado às atividades econômicas da região. (MAZZER; GONÇALVES, 2011)

Historicamente os dados batimétricos vem sido obtidos através da utilização de ecobatímetros, que consiste em um aparelho que se baseia na medição do tempo transcorrido entre a emissão de um pulso sonoro e a recepção do mesmo sinal após ser refletido pelo fundo (FERREIRA, 2013). Porém, essa abordagem é limitada à capacidade de navegação na área de interesse e aos altos custos para realização. Devido a essas dificuldades observadas nos métodos tradicionais, nota-se a necessidade de otimização desse processo.

A Batimetria Derivada de Satélite (BDS) é uma técnica alternativa capaz de estimar profundidades para ambientes costeiros de profundidade rasa. Existem diversos algoritmos para realização da BDS, em grande parte encetam de um princípio físico simples que consiste na atenuação da luz pela água, com isso consegue-se criar uma relação entre a reflectância de cada banda espectral com a profundidade (PHILPOT, 1989). Estes métodos exigem alta precisão da imagem de entrada, ou seja, são altamente sensíveis à correção atmosférica utilizada para converter a radiação total recebida pelo satélite para a radiação real transmitida pela coluna da água (HEDLEY et al., 2016) (KUTSER, T. et al. 2013).

Com o surgimento de tecnologias de reconhecimento de padrões e aprendizado de máquina (*machine-learning*), abordagens computacionais têm sido propostas para estimação da profundidade de ambientes subaquáticos a partir de imagens de satélite (MANESSA, M. D. M. et al.2016). De uma forma geral, pode-se observar principalmente implementações para ambientes marinhos rasos. Os ambientes deposicionais estuarinos e lagunares possuem uma abordagem complexa devido a alta taxa de ressuspensão de material particulado proveniente do aporte fluvial continental. Das técnicas de predição utilizadas, podem ser elencadas principalmente abordagens lineares baseadas em regressões lineares simples ou múltiplas (ALVES et al. 2018) (FILIPPI, 2020). Porém, devido à alta complexidade e peculiaridade de alguns ambientes, as técnicas lineares podem ser incapazes de representar suas características, além de tornar o modelo extremamente dependente da região que foi calibrado.

O presente trabalho aborda a implementação, validação e comparação de diferentes algoritmos supervisionados de ***machine-learning*** para obtenção da BDS para a região da Baía da Babitonga, utilizando como conjunto de dados as imagens do satélite Sentinel-2 associadas à medição *in situ*, na mesma data da aquisição do satélite, utilizando ecobatímetro. Os resultados experimentais demonstram a viabilidade do *workflow* proposto, tendo seus melhores resultados a partir de modelos baseados em conjuntos (*ensembles)* de árvores de decisão, com um r^2 de 0.95 e erro absoluto médio de 0.54m, se mostrando superior quando comparados aos outros modelos testados, baseados em técnicas lineares.

O restante deste artigo está organizado da seguinte forma: Na Seção 2 são apresentados os trabalhos correlatos com enfoque em métodos aplicados para obtenção de batimetria através de imagens de satélites. A Seção 3 apresenta a fundamentação teórica básica necessária para melhor compreensão das seções seguintes. A Seção 4 descreve a metodologia empregada para o desenvolvimento da solução proposta, mostrando as análises realizadas, os algoritmos utilizados, as métricas e indicadores a serem otimizados. A descrição detalhada dos processos de

validação e os resultados experimentais obtidos nestes processos são encontrados na Seção 5. A Seção 6 apresenta as discussões e conclusões obtidas acerca do trabalho desenvolvido, assim como possíveis trabalhos futuros.

2 Trabalhos Correlatos

A partir de uma revisão da literatura, foram verificadas abordagens computacionais para a batimetria a partir de imagens de satélite. Nota-se que os principais métodos utilizados são os métodos lineares em conjunto com os dados do satélite Landsat-7 (ALVES, D. C. L. et al. 2018). Foram selecionados trabalhos publicados a partir de 2005, contendo as seguintes *keywords*: Batimetria Derivada de Satélite, Sensoriamento Remoto, Landsat, Sentinel. Os trabalhos mais relevantes ao contexto de aplicação supracitado são elencados abaixo e na Tabela 1.

- KRUG e NOERNBERG. (2006): Este trabalho foi realizado no Complexo Estuarino de Paranaguá, que consiste em um complexo com 75% de profundidade inferior a 5m. Os dados utilizados foram do satélite Landsat-7, adquiridos para a data de 26 de setembro de 1999. O algoritmo proposto foi a uma regressão polinomial de grau 2 sobre o índice de diferença normalizada da água (NDWI) das bandas 2 e 4. O coeficiente de determinação r^2 foi de 0,746.

- MANESSA, M. D. M, et al. (2016): Neste trabalho foi realizada a BDS para duas regiões da Indonésia, sendo elas, Ilhas Gili Mantra e Ilha Panggang. As imagens utilizadas são provenientes do satélite Worldview-2. Este trabalho compara o método *Random Forest* com os métodos tradicionais de Múltiplas Regressões Lineares. O r^2 obtido para a melhor combinação de bandas para a região das Ilhas Gili Mantra foi de 0.874, e para a Ilha Panggang de 0.858.

- ALVES, D. C. L. et al. (2018): Neste trabalho foram utilizados dados do satélite Landsat-8 para a região do canal do porto de Rio Grande (RS-Brasil), situado no estuário da Lagoa dos Patos. A batimetria medida *in situ* foi realizada no mês de Novembro de 2015, e foram utilizadas imagens de Fevereiro de 2015 que resultaram em um r^2 de 0.817 e também imagens de dezembro de 2015 que resultaram em um r^2 de 0.965. O algoritmo proposto é uma regressão utilizando o logaritmo natural do *NDWI* das bandas 1 e 5 (Stumpf et al. 2003).

- GEYMAN E. C. & MALOOF A. C (2019): Este trabalho utiliza imagens do satélite *RapidEye* relacionadas com a medição da batimetria *in situ* para regiões com diferentes tipos de fundos dos Bancos das Bahamas. O objetivo foi comparar a eficácia na estimação da profundidade para regiões de fundos distintos com diversos algoritmos. O r^2 obtido no melhor algoritmo foi de 0.90.

- FILIPPI, B. (2020): Este trabalho foi realizado para a Baía da Babitonga, com dados do satélite Sentinel-2A adquiridos para a data de 20 de maio de 2018. O algoritmo proposto foi uma regressão polinomial de grau 2 sobre o NDWI das bandas 5 e 3 para regiões com índice de material particulado em suspensão inferior a 10g/cm³ e profundidade inferior a 20m. Nestes casos, o r^2 obtido foi de 0.68.

Tabela 1 - Tabela comparativa dos métodos encontrados na literatura com relação ao trabalho proposto.

Autor	Ano	Método	Local	Satélite	r^2
Lilian Krug e Mauricio Noernberg	2006	Regressão Polinomial	Complexo Estuarino de Paranaguá	Landsat-7	0.746
Manessa et al.	2016	Random Forest	Ilhas Gili Mantra e Ilha Panggang	Worldview-2	0.874
Alves et al.	2018	Stumpf et al. (2003)	Lagoa dos Patos	Landsat-8	0.960
Geyman, E. C. & Maloof, A. C	2019	CBR	Great Bahama Bank	RapidEye	0.900
Bianca Filippi	2020	Regressão Polinomial	Baía da Babitonga	Sentinel-2	0.680
Camozzato et al.	2021	LGBM	Baía da Babitonga	Sentinel-2	0.948

Fonte: Elaborado pelo autor.

3 Fundamentação Teórica

3.1 *Machine-Learning* (Aprendizado de Máquina)

O aprendizado de máquina, ou *machine-learning*, é o campo de estudo que dá aos computadores a habilidade de “aprender” através de modelos computacionais, sem serem explicitamente programados, definido por Samuel (1959). Dentro do *machine-learning* tem-se diferentes tipos de aprendizados, sendo os principais: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço. Além dos tipos de aprendizado, pode-se dividir os problemas gerais a serem resolvidos em dois tipos: regressão e classificação. Um modelo ideal apresenta baixo viés (*low bias*) e baixa variância (*low variance*), ou seja, alta acurácia nos dados de treino e também nos dados de teste, que consiste em dados nunca antes visto pelo modelo.

Os tipos de aprendizados mencionados são subcategorias dentro do aprendizado de máquina. O aprendizado supervisionado é a tarefa de aprendizado de máquina que consiste em aprender uma função que mapeia uma entrada para uma saída com base em pares de entrada-saída de exemplo (RUSSELL; NORVIG. 1995). O aprendizado não supervisionado consiste na subcategoria que aprende padrões sobre conjunto de dados não anotados (HINTON; SEJNOWSKI. 1999). O aprendizado por reforço utiliza agentes inteligentes que realizam ações em um ambiente visando maximizar a recompensa cumulativa (HU et al. 2020), ou seja, a partir do resultado de cada época de treinamento os agentes são recompensados ou punidos de acordo com suas ações no ambiente de controle buscando sempre maximizar a recompensa.

A regressão é uma ferramenta estatística utilizada para modelar relacionamentos entre quaisquer variáveis independentes e variáveis dependentes contínuas, como por exemplo, salário, peso, profundidade, preço. Classificação é a abordagem estatística utilizada para modelar relacionamentos entre as variáveis independentes e variáveis dependentes discretas (FREEDMAN D. 2009), ou seja, que podem ser separadas em classes, como por exemplo, a classificação binária de um email entre *spam* ou não.

A Batimetria Derivada de Satélite consiste em um problema de regressão pois a profundidade assume valores contínuos. No desenvolvimento deste trabalho foi utilizado o aprendizado supervisionado, utilizando como variáveis independentes (*features*) para calibração do modelo as informações contidas nas imagens de satélite e variável dependente (*target*) a profundidade previamente anotada (*label*) com a medição *in situ*.

3.2 Formato dos Dados

As imagens de sensoriamento remoto usualmente são armazenadas no formato geoTIFF, que consiste em um arquivo com extensão *.tif,* com conjunto de metadados geográficos embutidos. Os dados medidos *in situ* são armazenados com extensão *.xyz*, que representa em um arquivo tabular contendo as coordenadas geográficas da batimetria e a profundidade para este ponto. Essas coordenadas podem estar disponíveis em Universal Transversa de Mercator (UTM) ou em Latitude Longitude. No levantamento batimétrico tradicional utilizado neste trabalho, o formato UTM foi o escolhido. Esses dados geográficos, tanto da imagem como da batimetria medida, são necessários para a preparação inicial dos dados, pois a escala da batimetria medida pelo método tradicional e a escala das bandas dos satélites são diferentes, necessitando uma conversão de posição geográfica para *pixel* equivalente na imagem.

3.3 Frameworks e Tecnologias

Para realizar a prototipagem do trabalho foi utilizada a linguagem de programação *Python* no ambiente de desenvolvimento *jupyter*, que consiste em uma página web interativa com suporte nativo às linguagens de programação *Python*, *Julia* e *R*. Nesse ambiente instalou-se algumas bibliotecas de código aberto (*open-source)*[1] específicas de acordo com a necessidade do projeto, entre elas, as mais relevantes:

- Scikit-learn: biblioteca que provê uma *Application Programming Interface* (API)[2] fácil e intuitiva para modelos clássicos de *machine-learning*, processamento e transformação de dados, estatística, entre outras.
- XGBoost: XGBoost é uma biblioteca que fornece um *framework* de aumento de gradiente (*gradient boosting)* para diversas linguagens de programação, incluindo Python. De acordo com a descrição do projeto, tem objetivo de proporcionar uma biblioteca de *gradient boosting* escalável, portátil e distribuída.
- LightGBM: *Light Gradient Boosting Machine* (LGBM), é uma estrutura de aumento de gradiente distribuída gratuita e de código aberto para aprendizado de máquina desenvolvida originalmente pela Microsoft. Também disponível para diversas linguagens de programação, assim como o XGBoost.
- Numpy: biblioteca matemática voltada principalmente para álgebra linear e geometria, provê uma API fácil e rápida para manipulação de arrays N dimensionais.
- Pandas: biblioteca rápida e poderosa contendo diversos recursos para análise e manipulação de dados e arquivos, estatística descritiva, entre outros
- Matplotlib: biblioteca que tem uma API simples voltada principalmente para visualização de dados e estatística descritiva.
- Seaborn: biblioteca baseada na *matplotlib*, provê uma maneira fácil e rápida de gerar gráficos atrativos e informativos com pouco código.
- SHAP: biblioteca baseada na teoria dos jogos para explicar a tomada de decisão para de qualquer modelo de machine-learning.
- Tifffile: biblioteca para leitura, escrita e manipulação de arquivos com extensão tif.

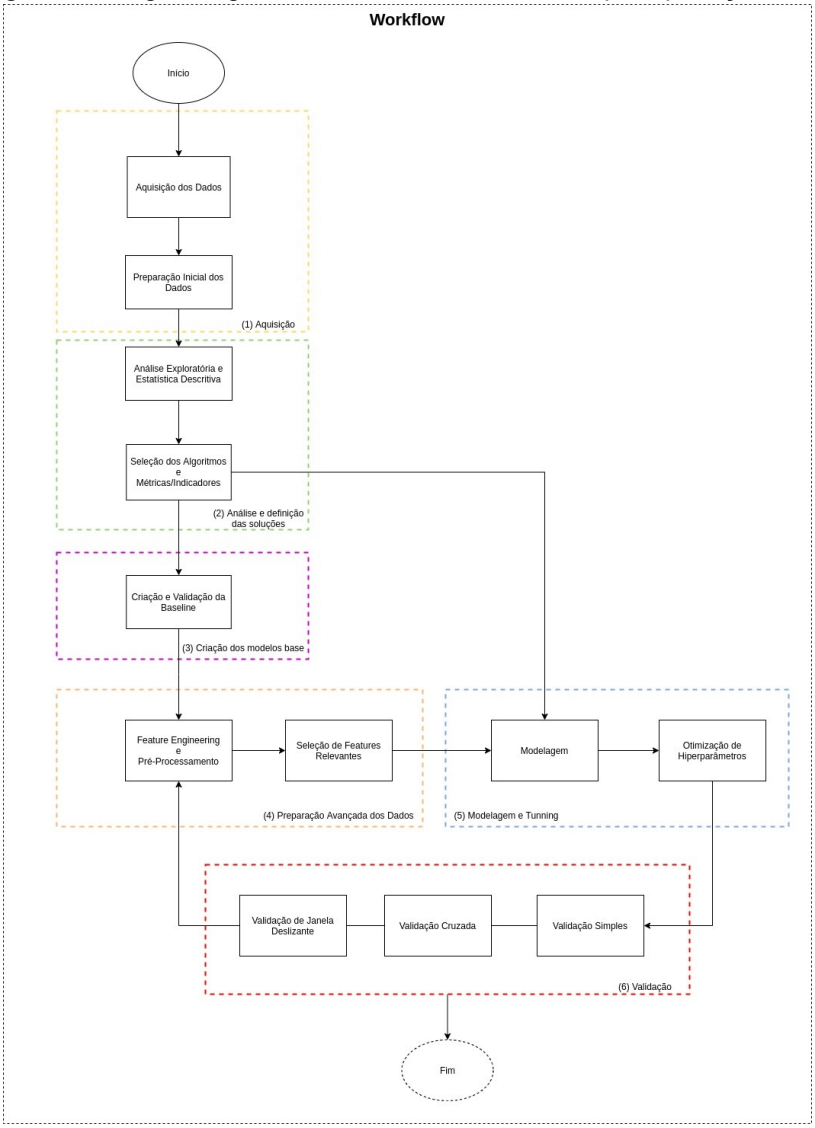
[1] Códigos com direito autoral que permite estudo, modificação e utilização para qualquer finalidade.

[2] Conjunto de normas que possibilita a comunicação entre plataformas, bibliotecas, e serviços através de uma série de padrões e protocolos.

4 Metodologia

Os recursos de máquina disponíveis no desenvolvimento do fluxo de trabalho (*workflow*) proposto foram 8 GB RAM 3200MHz, processador AMD RyzenTM 5 4600H 3.0GHz e placa de processamento gráfico NVIDIA GeForce GTX 1650. Todos os códigos desenvolvidos durante o trabalho estão disponíveis no repositório do *Github*[3], assim como os *frameworks open-source* utilizados referenciados na subseção 3.3. A Figura 1 apresenta o fluxograma utilizado para o desenvolvimento deste trabalho, e cada etapa descrita na figura está detalhada nas subseções seguintes.

Figura 1 - Diagrama geral do *workflow* desenvolvido para predição BDS.



Fonte: Elaborado pelo autor

[3] Plataforma para hospedagem e versionamento de código fonte. Disponível em <https://github.com/> Acesso em 28/04/2021

4.1 Aquisição de Dados

A aquisição dos dados medidos *in situ* se fez por meio de um levantamento batimétrico monofeixe de Categoria B de acordo com as Normas da Autoridade Marítima para Levantamentos Hidrográficos (NORMAN-25). O levantamento ocorreu no período entre 14 e 24 de maio de 2018. Datas escolhidas para coincidir com a passagem do satélite Sentinel-2 sobre a área de estudo (FILIPPI, 2020). A área sondada foi previamente determinada por linhas com espaçamento de 1 km totalizando 225 km, assim cobrindo todas áreas navegáveis da baía. As imagens foram adquiridas através da plataforma *Copernicus Hub* (ESA. c. 2020-2021) com *download* manual sobre a missão Sentinel-2A, filtrando a data e região de interesse e exportadas em formato GeoTIFF. A missão Sentinel possui dois satélites idênticos, Sentinel-2A e Sentinel-2B, com sensor multiespectral, tamanho de faixa de 290km, a aquisição se dá a cada cinco dias com precisão radiométrica de 12-bits. A Tabela 2 mostra detalhadamente cada banda do satélite Sentinel-2A com suas respectivas resoluções espaciais, comprimento de onda e largura de banda.

Tabela 2 - Composição de bandas multiespectrais do satélite Sentinel-2A

Sentinel-2A Bands	**Resolução Espacial**	**Comprimento de Onda (nm)**	**Largura de Banda (nm)**
1 - Aerosol Costeiro	60	442,7	21
2 - Azul	10	492,4	66
3 - Verde	10	559,8	36
4 - Vermelho	10	664,6	31
5 Red Edge 1	20	704,5	15
6 - Red Edge 2	20	740,5	15
7 - Red Edge 3	20	782,8	20
8 - NIR	10	832,8	106
8A - Narrow NIR	20	864,7	21
9 - Vapor D'água	60	945,1	20
10 - Cirrus	60	1373,5	31
11 - SWIR1	20	1613,7	91
12 - SWIRD2	20	2202,4	175

Fonte:Copernicus Hub

As imagens obtidas tem largura 2742 *pixels* e altura 2094 *pixels*, e todas as bandas ao serem baixadas passaram por um processo de reamostragem (*resampling)*[4] para igualar a resolução espacial em 10m/*pixel*.

[4] Método que consiste em extrair amostras repetidas das amostras de dados originais

4.2 Preparação dos Dados

O levantamento batimétrico realizado tem resolução espacial de 1 metro, sendo assim necessárias algumas manipulações sobre o *dataset* para transformá-lo à mesma dimensão espacial das imagens. Para isso, com a informação da posição UTM inicial das bandas, agregadas à escala espacial da imagem, converteu-se as posições geográficas da batimetria em posições equivalentes em *pixel* da imagem. A equação matemática que descreve esse processamento é expressa por:

$$pixel_x = \frac{(coord_x - start_x)}{scale_x}\ ,\ pixel_y = \frac{(coord_y - start_y)}{scale_y} \quad (1)$$

Onde:

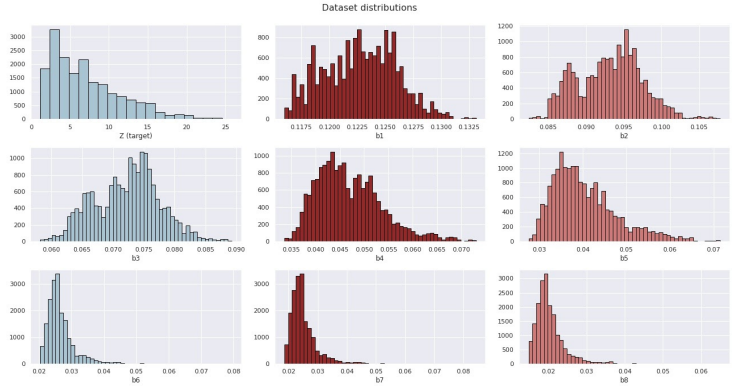
- $coord_x$ e $coord_y$: coordenadas UTM ***x*** e ***y*** da batimetria medida;
- $start_x$ e $start_y$: coordenadas UTM ***x*** e ***y*** iniciais da imagem;
- $scale_x$ e $scale_y$: escala metros/*pixel* da imagem.

Como cada *pixel* da imagem contém diversos pontos de batimetria medidos em campo, agrupou-se por *pixel* as batimetrias medidas, extraindo a média dos pontos coincidentes e aplicando um filtro de variância para evitar pontos onde a profundidade entre os pontos de um único *pixel* fossem muito divergentes.

4.3 Análise Exploratória e Estatística Descritiva

Após ter o conjunto de dados preparado, foram realizadas análises estatísticas para melhor compreender o conjunto de dados. O histograma da Figura 2 mostra as distribuições de frequências dos dados brutos da batimetria (**z**) e o domínio do valor das imagens das bandas após o filtro de variância aplicado.

Figura 2 - Distribuição de frequência dos valores da profundidade e das bandas 1 a 8

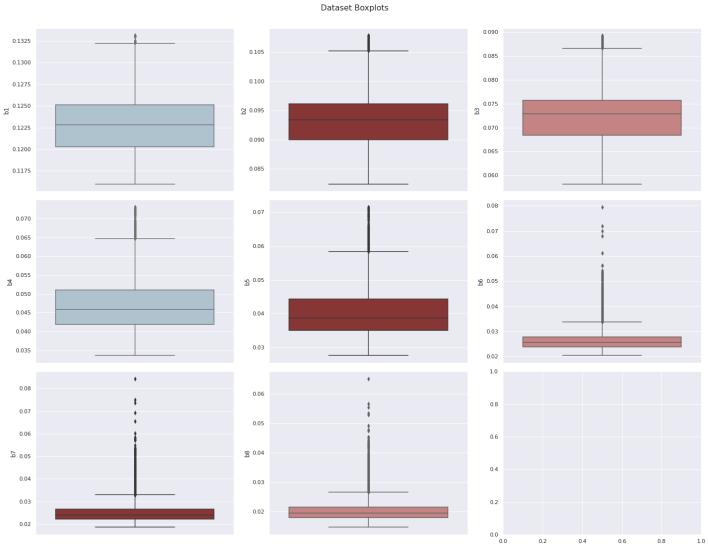


Fonte: Elaborada pelo autor

Nota-se que a variável dependente ***Z*** tem uma assimetria à direita, tendo a maior parte da sua distribuição contida na faixa de 0 a 10 metros de profundidade. Isso pode influenciar na capacidade de alguns modelos de extrapolarem para valores fora desse *range*. Além da assimetria à direita contida na variável Z, também nota-se uma

assimetria positiva mais intensificada nos valores das bandas 6, 7 e 8. Esse tipo de assimetria ocorre normalmente quando os limites inferiores são muito baixos se comparados ao restante dos dados. Nota-se em b6, por exemplo, que a moda está em torno de 0.025 com a média um pouco maior que este valor. Porém, mesmo que grande parte da distribuição estando contida em volta desse valor, ainda existem diversos valores, potencialmente *outliers*, distribuídos à direita crescendo até 0.08. Isso faz com que essa distribuição tenha uma cauda longa a direita. Os *boxplots* da Figura 3 mostram claramente a quantidade de pontos distantes do centro da distribuição nas bandas 6 a 8.

Figura 3 - Análise de dispersão dos valores das bandas 1 a 8

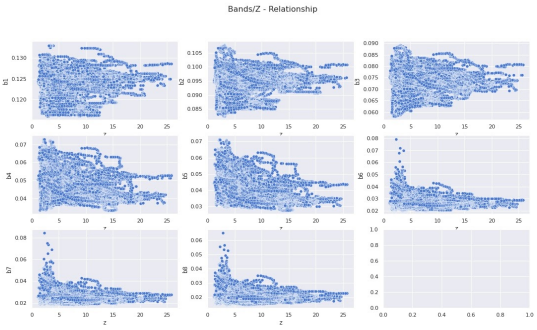


Fonte: Elaborada pelo autor

Esse tipo de assimetria pode indicar a necessidade de algum pré-processamento nos dados de calibração do modelo, como por exemplo normalização, ou remoção de *outliers*. Isso será discutido mais à frente na etapa de *feature engineering* e pré-processamento posterior a criação do modelo baseline.

Além disso, a Figura 4 mostra a relação entre os dados brutos das bandas 1 a 8 com a variável de interesse Z. Percebe-se a não linearidade dos dados em todas as relações. Esse tipo de comportamento ruidoso do dado pode ser explicado devido à complexidade do ambiente de estudo e todos os processos morfodinâmicos presentes na região.

Figura 4 - Relação dos valores das bandas 1 a 8 com a profundidade



Fonte: Elaborada pelo autor

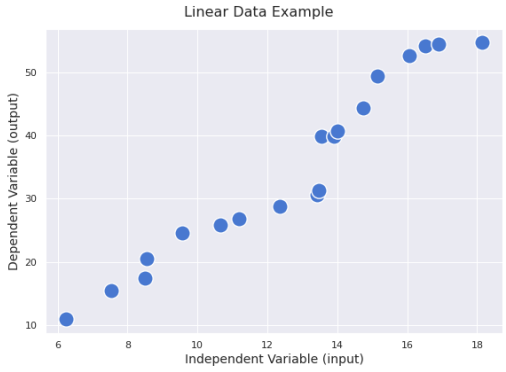
Adicionalmente à exploração estatística demonstrada nesta subseção, foram obtidas evidências que modelos baseados em técnicas lineares podem apresentar sua performance reduzida devido a presença de ruído e não linearidade nos dados.

4.4 Algoritmos Propostos

4.4.1 Regressão Linear

A regressão linear consiste em tentar encontrar um hiperplano que melhor se ajuste aos dados de entrada (HASTIE *et al*. 2009). A Figura 5 mostra um conjunto de dados de duas dimensões que apresenta uma certa correlação linear, assim podendo ser modelado com um modelo de regressão linear.

Figura 5 - Exemplo de dados com tendência linear



Fonte: Elaborada pelo autor

A regressão linear usa descidas em gradiente (*gradient descent)* sobre a função de perda (*loss function*) para encontrar mínimos globais que correspondem ao melhor ajuste da curva, a função de perda é definida de acordo com a implementação, a mais comum é o erro médio quadrático. Exemplificando matematicamente o espaço onde é esperado que o *target* seja uma combinação linear de outras *features* tem-se:

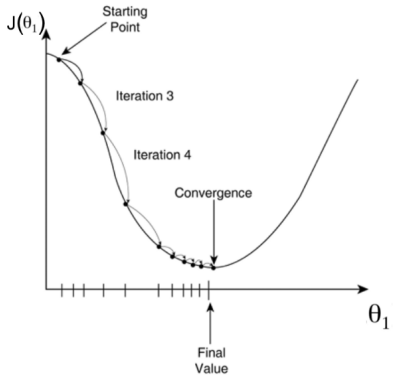
$$\hat{y}(w, x) = w_0 + w_1 x_1 + \ldots + w_n x_n \quad (2)$$

A técnica utilizada na implementação de Pedregosa (2011) é conhecida como método dos mínimos quadrados e parte do princípio que minimizando a soma do quadrado dos resíduos das iterações é possível encontrar os coeficientes **w** que associados as *features* **x** melhor representem o *target* **y**. Matematicamente a equação reduzida utilizada na implementação assume a seguinte forma:

$$min \quad ||Xw - y||_2^2 \quad (3)$$

A Figura 6 representa uma função de perda (*loss function)* genérica e as iterações até a convergência para o mínimo global, mostrando de maneira aproximada como funciona o processo de *gradient descent* e treinamento de uma regressão linear.

Figura 6 - Exemplo do método de descida em gradiente

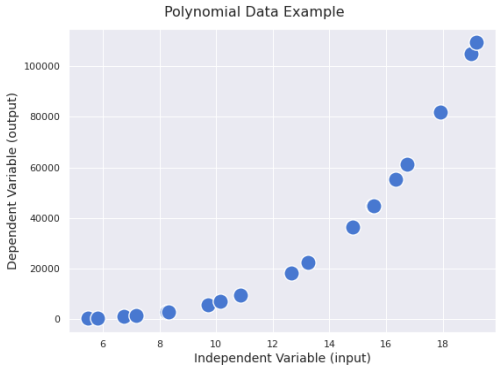


Fonte: LIU, C. 2020

4.4.2 Regressão Polinomial

A regressão polinomial é um caso especial da regressão linear, com a diferença que enquanto na linear busca-se a melhor reta/plano que se ajuste ao conjunto de dados, na polinomial busca-se a melhor curva/superfície de grau N, sendo este, o grau do polinômio da regressão (HASTIE *et al*. 2009). Este tipo de regressão é usada quando a relação entre a variável dependente e a variável independente tem comportamento curvilíneo, podendo ser modelada por um polinômio de grau N, como por exemplo na Figura 7.

Figura 7 - Exemplo de dados com tendência polinomial de grau 2



Fonte: Elaborada pelo autor.

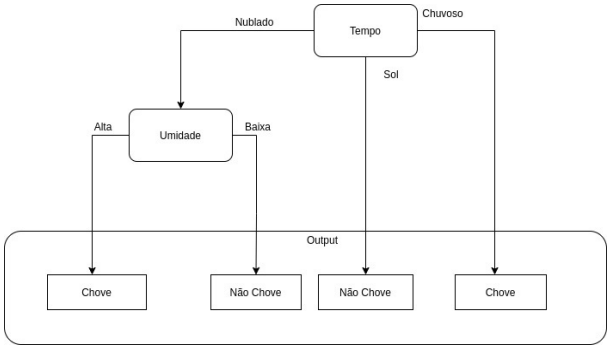
O modo como o modelo se ajusta aos dados é o mesmo da regressão linear simples, utilizando descidas em gradiente na função de perda.

4.4.3 Random Forest

Random Forests (RF) ou árvores aleatórias é um algoritmo de *machine-learning* baseado em árvores de decisão (*decision trees*) e para entendê-lo, precisa-se conhecer primeiramente as árvores de decisão. Estas árvores usam regras de decisão simples para chegarem a um *output* e pode ser exemplificado através da Figura 8 em um

exemplo de árvore de decisão para classificação de “chover ou não chover” de acordo com entradas do tempo.

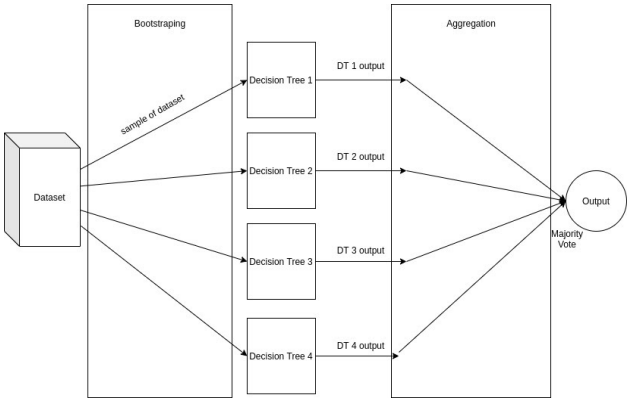
Figura 8 - Exemplo de Árvore de Decisão Não Binária Simples



Fonte: Elaborada pelo autor

Tipicamente, modelos de árvore de decisão tem baixo viés (*bias*) e alta variância (*variance*), eles tendem a se ajustar muito aos dados de treino e não conseguem generalizar para conjuntos novos (*overfitting*). A *Random Forest* foi criada para solucionar o problema de alta variância das *decisions trees*. Este algoritmo utiliza uma técnica de *ensemble*, que significa combinar diversos modelos e no caso da *Random Forest*, a técnica utilizada é o *bagging* (*bootstrap aggregation*). Esta técnica consiste, de maneira resumida, em utilizar múltiplas árvores de decisão, passando a cada uma delas uma amostra aleatória do *dataset* original e treiná-las individualmente (*bootstrapping*). Após o treinamento individual, cada árvore realiza sua previsão e o *output* final será a maioria das saídas de cada árvore para classificação, ou a média das saídas de cada árvore para regressão, técnica nomeada de *majority vote* (HASTIE *et al*. 2009). Na Figura 9 é apresentado uma exemplificação da técnica de *bagging* aplicada para gerar a *Random Forest*.

Figura 9 - Exemplo da técnica de bagging.



Fonte: Elaborada pelo autor

A abordagem matemática do algoritmo utilizado neste trabalho (PEDREGOSA *et al.* 2011*)* pode ser escrita da seguinte maneira:

Tendo um conjunto de vetores de features $x_i \varepsilon R$, $i = 1 \rightarrow I$ e um vetor de labels**,** o algoritmo tende a particionar os dados recursivamente de maneira que as amostras (Q)

do *dataset* (D) com *labels* similares sejam agrupadas em Q. Escolhendo arbitrariamente um nó *m,* os N dados neste nó podem ser representados como Q_m. Para cada possível divisão $\theta = (j, t_m)$, onde *j* é a *feature*, e t_m o ponto de corte (*threshold*) que será utilizado para particionar o dado em subconjuntos da direita e da esquerda. Exemplo do processo anteriormente mencionado pode ser caracterizado pela equação:

$$Q_m^{left}(\theta) = \{(x, y) \mid x_j <= t_m\} \ , \ Q_m^{right}(\theta) = Q_m \setminus Q_m^{left}(\theta) \tag{3}$$

Isso mostra que o subconjunto do nó filho da esquerda será composto pela *feature* ***j*** até o *threshold* definido e o subconjunto do nó filho da direita será o subconjunto de Q_m excluindo o subconjunto da esquerda.

Para cada árvore de decisão é calculada a importância dos seus nós que assumem apenas dois filhos (árvore binária). A função de impureza, ou função de perda (*loss function*) ***H*** utilizada depende do tipo de aplicação, no caso do problema deste trabalho, regressão.
O método genérico, pode ser escrito da seguinte maneira:

$$G(Q_m, \theta) = \frac{N_m^{left}}{N_m} H(Q_m^{left}(\theta)) + \frac{N_m^{right}}{N_m} H(Q_m^{right}(\theta)) \tag{5}$$

Onde:

- $G(Q_m, \theta)$ **=** importância do nó **m;**
- $H(Q_m^X(\theta))$ = impureza no nó **m** e no lado X (direita ou esquerda);
- $\frac{N_m^X}{N_m}$ = Número ponderado de amostras no nó m.

A implementação padrão de Pedregosa *(*2011*)* utiliza como função de perda padrão o cálculo da redução da variância utilizando o erro médio quadrático, assumindo a seguinte forma:

$$H(Q_m) = \frac{1}{N_m} \sum_{i=1}^{N_m} (y_i - \hat{y})^2 \tag{6}$$

Onde:

- y_i é a label para o exemplo;
- N_m o número de exemplos para o subconjunto;
- ŷ a média dada por $\frac{1}{N_m} \sum_{i=1}^{N_m} y_i$ **.**

O objetivo final de cada árvore de decisão é reduzir a impureza, logo:

$$\theta^* = argmin\ G(Q_m, \theta) \tag{7}$$

Todo o processo descrito anteriormente é realizado diversas vezes para cada árvore de decisão da *Random Forest*. Os valores são repartidos recursivamente até atingir a profundidade máxima definida pelo algoritmo. A importância de cada *feature*, ou peso, é o valor utilizado para gerar a previsão do modelo. Na Random Forest o valor

de *output* de um modelo de regressão é a média dos *outputs* de todas as árvores independentes.

4.4.4 XGBOOST

O XGBoost é uma implementação eficiente de algoritmos baseados em árvores de decisão com *gradient boost* (CHEN; GUESTRIN. 2016). Consiste em um método que tenta otimizar a função de perda aplicando algumas técnicas de regularização. Computacionalmente o XGBoost é otimizado para tentar utilizar melhor os recursos da máquina, incluindo a capacidade de usar o disco para manipular conjuntos de dados grandes que ultrapassam os limites da memória principal.

Similar ao método de *bagging*, *boosting* também é uma técnica de *ensemble* (HASTIE *et al*. 2009). A principal diferença dessa técnica é que as árvores de decisão serão treinadas de forma aditiva, ou seja, a partir dos resíduos gerados pelas iterações anteriores, sendo o resíduo inicial a diferença entre o valor real e a média dos valores ou de um valor predefinido. O XGBoost também costuma utilizar o erro quadrático como *loss function* na regressão para criar as árvores, porém, a função final tem um termo extra que é a regularização, que penaliza a complexidade do modelo. O intuito da regularização é evitar alta complexidade nos modelos internos, fazendo com que utilize árvores mais simples para previsão, reduzindo assim o *overfitting.* A equação que descreve a *loss function* e a regularização por Chen (2016) é a seguinte:

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \gamma T + \frac{1}{2}\lambda||O||^2 \quad (8)$$

Onde:

- *l* = loss function;
- γ = termo penalizante para encorajar a “poda” da árvore;
- T = número total de nós finais (folhas);
- $\frac{1}{2}\lambda||O||^2$ = termo regularizador;
- *i* = instância;

O objetivo é encontrar um valor ***O*** para a folha que minimize toda a equação. Para encontrar o melhor valor de saída ***O*** para um nó, XGBoost (CHEN; GUESTRIN. 2016) utiliza o método de aproximação por polinômio de Taylor de ordem 2. A forma já reduzida da aproximação pode ser escrita como:

$$L^{(t)} \simeq \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i O + \frac{1}{2} h_i O^2] + \Omega\lambda||O||^2 \quad (9)$$

Onde:

- *l* = *loss function;*
- g = primeira derivada da *loss function;*
- h = segunda derivada da *loss function.*

Expandindo a série, só se tem referência à ***O*** nos termos multiplicados pela primeira e segunda derivada, logo o termo de ordem zero não tem efeito na otimização do valor **O**. Para encontrar o valor capaz de minimizar a função, deve-se derivá-la e igualar a zero e resolver para este valor, no caso ***O***.

$$\frac{d}{dO}(g_1 + g_2 + \dots g_n)O + \frac{1}{2}(h_1 + h_2 + \dots h_n + \lambda)O^2 = 0 \quad (10)$$

Resolvendo esta equação se tem:

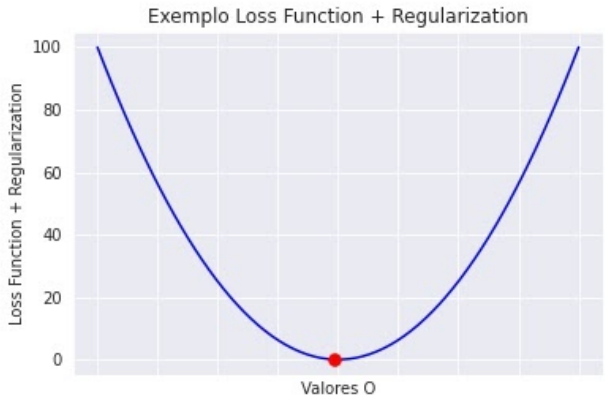
$$O = \frac{-(g_1 + g_2 + \dots g_n)}{(h_1 + h_2 + \dots + h_n + \lambda)} \quad (11)$$

Substituindo ***g*** pela primeira derivada da loss function utilizada e ***h*** pela segunda derivada tem-se:

$$O = \frac{\sum resíduos}{Número\ total\ de\ resíduos + \lambda} \quad (12)$$

Observando a Figura 10, o ponto vermelho corresponde ao resultado otimizado do valor ***O***, que mostra o ponto no eixo X de coordenadas onde a curva, ou seja, a *loss function*, atinge seu valor mínimo.

Figura 10 - Exemplo genérico de *Loss Function* com regularização para Valores ***O***



Fonte: Elaborado pelo autor

No crescimento da árvore é necessário ter a pontuação de similaridade (*similarity score)*. Com isso, o algoritmo multiplica a expansão da Equação 9 por -1, invertendo assim a concavidade da curva da Figura 10. De acordo com o *paper* original do XGBoost (CHEN; GUESTRIN. 2016), o *similarity score* será o ponto máximo em *y* desta curva, porém, na implementação computacional, o valor adotado para o *similarity* score é de 2*y.* A equação final para obter o *similarity score* é:

$$Similarity\ score = \frac{\sum(resíduos)^2}{Número\ total\ de\ resíduos + \lambda} \quad (13)$$

A partir da similaridade é possível encontrar a contribuição relativa de cada feature para o modelo, ou ganho, como referenciado no artigo original.

$$gain = score_{left} + score_{right} - score_{father} \quad (13)$$

Com o ganho é possível encontrar as melhores divisões para as árvores binárias, iterando sobre os valores das *features* e buscando sempre a divisão que proporciona o maior ganho. Este valor deve ser sempre positivo e maior que o hiperparâmetro *min_split_gain*. Esse algoritmo mencionado se chama *Exact Greedy Algorithm*, a complexidade de tempo é O(n*m) onde *n* é o número de amostras de treino e *m* a dimensão do conjunto de *features* (CHEN; GUESTRIN. 2016). O *framework* XGBoost provê diversas otimizações para situações onde o conjunto de dados é grande, porém a exemplificação de funcionamento e matemática destes casos não está no escopo deste trabalho (CHEN; GUESTRIN. 2016).

4.4.5 LightGBM (LGBM)

Conforme a documentação oficial, o *LightGBM* ou LGBM também é um algoritmo baseado em *Gradient Boosting Decision Trees* (*GBDT*), porém a maneira como é implementado é um pouco diferente, tendo como principal objetivo otimizar o tempo de treinamento e os resultados. Diferentemente de outras técnicas de *boosting*, que usam por padrão algoritmos baseados em pré-ordenação (*pre-sort based algorithms*), o LGBM usa algoritmos baseados em histogramas (*histogram-based algorithms*), que agrupam as *features* contínuas em intervalos discretos (*bins*). Isso faz com que o treinamento se torne mais rápido e consuma menos memória.

Tomando o custo computacional, os algoritmos baseados em histogramas tem complexidade de tempo *O(dados)*, porém, a operação envolvida neste processo é apenas um somatório, e uma vez realizado, a complexidade de tempo se torna *O(bins)*, onde *bins* é menor do que *dados.* Além disso, como *bins*, é um valor discreto e pequeno, é possível armazenar os dados em tipos que necessitem menos memória.

Na otimização da performance, o LGBM adota um método diferente para crescer as árvores de decisão, que é o *leaf-wise growth (best-first)*. Isso significa que a divisão dos dados se dá sempre no nó com o maior diferença na perda, ou seja, no nó onde o delta da função de perda é maior. Neste caso, diferentemente do método utilizado no XGBoost (*level-wise growth)*, a árvore cresce de maneira não balanceada, como mostra a Figura 11. O problema deste método é que pode causar *overfitting* facilmente, quando conjuntos pequenos de dados são utilizados (KE. et al. 2017) (MICROSOFT. c. 2021).

Figura 11 - Comparação de crescimento *leaf-wise* e *level-wise*.

Fonte: LGBM Documentation

4.4.6 Cluster Based Regression (CBR)

O método *CBR* é também um método de *ensemble*, porém os modelos são treinados em subconjuntos dos dados. Os *k* subconjuntos são divididos utilizando o algoritmo não supervisionado *K-Means*, e para cada subconjunto é treinado um modelo de regressão. O *CBR* foi implementado utilizando *Random Forest*, Regressão Linear e Regressão Polinomial, onde, $k \in [2,\ 10]$. O valor final estimado para um *pixel* é a média ponderada das *k* predições onde os pesos da média ponderada são o inverso da distância entre o pixel sendo previsto e o centróide de cada classe. A equação 15 descreve a matemática para a previsão de uma profundidade para um *pixel* X utilizando o método CBR

$$z(pixel) = \sum_{1}^{k} [modelOutput_k(pixel) \ . \ \frac{1}{dist_k} \sum_{1}^{i} \frac{1}{dist_i}] \quad (15)$$

Podendo ser simplificado em:

$$z(pixel) = \sum_{1}^{k} w_k \ . \ modelOutput_n(pixel) \quad (16)$$

Onde:

- ***z(pixel)*** = profundidade para o *pixel*;
- w_k = peso da classe k para a previsão;
- $modeOutput_k(pixel)$ = previsão do modelo k para o *pixel*.

A previsão dos modelos independentes para o *pixel* são obtidas da mesma forma descritas anteriormente de acordo com o tipo de modelo utilizado no CBR.

4.5 Métricas e Indicadores

Escolher a métrica de problema a ser avaliada e otimizada é fundamental em qualquer processo de *machine-learning*. A partir da(s) métrica(s) definida(s) que será possível analisar e avaliar os resultados dos modelos construídos, verificando assim a eficiência destes no problema proposto. A escolha errada das métricas e indicadores pode afetar diretamente na capacidade de tomada de decisão a partir dos resultados do modelo. As principais métricas deste trabalho são:

- Avaliar a capacidade de correlacionar as reflectâncias com a profundidade
- Avaliar as melhores combinações de dados que minimizem o erro

Tudo que será realizado neste trabalho será voltado a melhorar estas métricas, porém é preciso ter cautela pois, como definido por Charles Goodhart (1983), quando uma métrica se torna um alvo, ela deixa de ser uma boa métrica, logo tornando-a menos confiável. Os indicadores, ou métricas de avaliação, são medidas matemáticas utilizadas relacionadas às métricas de negócio. Existem diversos indicadores, e cada um fornece uma visão diferente sobre o comportamento do modelo, sendo os principais utilizados neste trabalho:

- **R² Score:** Medida estatística que representa a proporção da variância da variável dependente que é explicada por um modelo linear simples que representa a média dos valores da variável dependente (GLATZ. 1999). Apesar do nome sugerir ser uma medida sempre positiva, o r² score pode assumir valores negativos, isso significa que o modelo utilizado para prever a variável dependente tem resultado inferior que a média dos valores.

$$R^2 = 1 - \frac{\sum_I (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \quad (17)$$

Onde:
- y_i = valor previsto;
- $\hat{y}_i$ = valor esperado;
- $\bar{y}$ = média dos valores esperados.

- **Erro Absoluto Médio (MAE):** Medida estatística que calcula a média dos erros entre o previsto e o esperado, ou *l1-norm loss*.

$$MAE = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |y_i - \hat{y}_i| \quad (18)$$

Onde:
- $n_{samples}$ = número de amostras;
- y_i = valor previsto;
- $\hat{y}_i$ = valor esperado.

- **Erro Médio Quadrático (MSE):** Medida estatística que calcula a média do quadrado dos erros entre o previsto e o esperado, ou *l2-norm loss*. Esse indicador é interessante quando altos erros não são tolerados, pois a potência faz com que qualquer erro relativamente grande faz com que o MSE cresça rapidamente.

$$MSE = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |y_i - \hat{y}_i|^2 \quad (19)$$

Onde os argumentos são os mesmos do MAE.

- **Raiz do Erro Quadrático Médio (RMSE)**: Devido a potência do MSE, este está em dimensão diferente de **y**, o RMSE é uma forma de melhorar a interpretabilidade do MSE, trazendo a mesma dimensão de y.

$$RMSE = \sqrt{\frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |y_i - \hat{y}_i|^2} \quad (20)$$

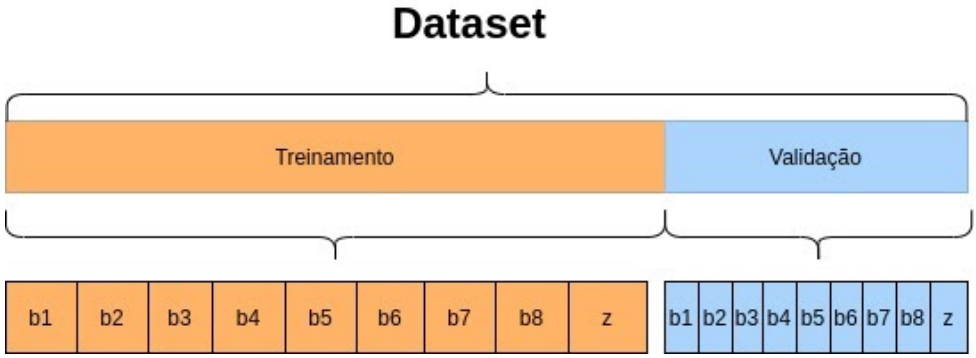
Onde os argumentos são os mesmos do MAE e do MSE.

4.6 *Baseline*

Criar um modelo *baseline* é uma etapa importante em um *pipeline* de *machine-learning*. A partir deste modelo pode-se fazer comparações para verificar se os modelos mais robustos estão conseguindo ter performance superior aos mais simples. Um modelo *baseline* pode ser desde uma medida de dispersão simples, como a média, até modelos mais robustos sem *feature engineering* e *tunning* de hiperparâmetros realizados. A ideia da baseline é criar uma solução rápida e fácil para prosseguir com as análises.

Para realizar a modelagem é necessário dividir (*split*) os dados já preparados em treino e validação. A divisão definida na etapa de treinamento e validação única foi de 70% dos dados para treino e 30% para validação, com divisão randômica. As *features* utilizadas nesta etapa foram os dados sem processamento adicional das bandas 1 a 8 relacionados com a variável de interesse da profundidade.

Figura 12 - Exemplo de divisão de dados 70/30



Fonte: Elaborada pelo autor

Com a divisão realizada, o treinamento da baseline pode ser executado e os resultados obtidos nesta etapa são anotados para serem comparados com etapas futuras. A Tabela 3 mostra os resultados com split simples dos modelos utilizados como baseline.

Tabela 3 - Resultados comparativos dos modelos utilizados como *baseline*

Modelo	**R^2**	**MAE**	**MSE**	**RMSE**
Random Forest	0.802	1.316	3.988	1.99
XGBoost	0.754	1.568	4.955	2.226
LGBM	0.719	1.721	5.662	2.379
Regressão Linear	0.431	2.596	11.469	3.366

Fonte: Próprio Autor

Além da validação simples com *split* randômico, foi realizada a validação cruzada simples (*cross-validation*) para proporcionar maior robustez a validação da *baseline* e garantir que os conjuntos de treino e validação não estão proporcionando

algum viés ao modelo. Foram definidas 10 divisões sobre o conjunto de dados para realizar a *cross-validation* e a variância dos resultados obtidos em cada *split* para cada modelo apresentaram valores próximos a zero garantindo a qualidade dos valores dos indicadores da tabela. O processo de *cross-validation* será descrito detalhadamente na Seção 5.

4.7 *Feature Engineering* e Pré-Processamento

Todo modelo de *machine-learning* usa uma representação numérica das informações (*features*) que tenham relação com a variável de interesse (*target*). A qualidade dos resultados do modelo está diretamente ligada à qualidade das *features* utilizadas para previsão.

Feature engineering consiste em, a partir dos dados brutos, extrair novas características que possam prover informações aos modelos preditivos que os ajudem a realizar as previsões de maneira mais assertiva (ZHENG A).

Diferentemente da *feature-engineering*, que busca criar novas features, o pré-processamento dos dados consiste em aplicar transformações, filtros, entre outras operações que façam sentido sobre o conjunto de dados brutos para melhorar a qualidade dos dados.

4.7.1 Feature Engineering - Razão do Log

O algoritmo da Transformação da Razão das Bandas (21) desenvolvido por Stumpf et al. (2003), utiliza a razão do logaritmo natural entre a reflectância de duas bandas para estimar a profundidade da coluna d'água.

$$ST = \left(\frac{ln(nRw(\lambda_i)}{ln(nRw(\lambda_j)}\right) \tag{21}$$

Onde:

- ST = Modelo proposto por Stumpf ainda sem referência vertical;
- n = constante para garantir que a razão permaneça positiva;
- $Rw(\lambda_i)$ = reflectância da banda i;
- $Rw(\lambda_j)$ = reflectância da banda j.

Foi realizada a permutação das bandas 1 a 8 aplicando o algoritmo da Equação 21 unindo a informação resultante ao conjunto de dados original.

4.7.1 Feature Engineering - NDWI

A relação de bandas do algoritmo NDWI de McFeeters (1996), também foi testado como *feature* adicional. Este algoritmo foi criado para determinar e diferenciar regiões de água e de terra a partir da relação entre as bandas verde e infravermelho próximo (NIR). Neste trabalho, aplicou-se a relação mostrada na equação 22 permutada em todas as bandas.

$$NDWI = \frac{Rw(\lambda_i) - Rw(\lambda_j)}{Rw(\lambda_i) + Rw(\lambda_j)} \tag{22}$$

Onde:

- *NDWI* **=** modelo proposto por McFeeters (1996) ainda sem referência vertical;
- $Rw(\lambda_i)$ = reflectância da banda i;
- $Rw(\lambda_j)$ = reflectância da banda j.

4.7.3 *Feature Engineering* - Material Particulado em Suspensão (MPS)

O material particulado em suspensão (MPS) é o conjunto de partículas que ficam dispersas na coluna da água. Essas partículas influenciam diretamente na reflectância medida pelos sensores do satélite uma vez que elas são capazes de interferir na penetração da luz no corpo d'água. Essa informação é potencialmente relevante para o modelo uma vez que, a presença em média ou grande quantidade de MPS pode levar o modelo a prever erroneamente estas zonas. A equação 23 descreve o algoritmo proposto por Liu et al. (2017), que utiliza a banda 7 (Red Edge 3) como base para realizar o cálculo do MPS presente em uma determinada região,

$$MPS = 2950\ (Rw(\lambda_7)^{1.357}) \tag{23}$$

Com essa informação é possível também filtrar regiões onde o valor de material particulado pode causar interferência na qualidade do modelo.

4.7.4 Transformações

Os modelos, em geral, tendem a lidar bem quando os dados são numéricos, sendo assim, possível utilizá-los de forma bruta. Porém alguns modelos, como a regressão linear, podem ter seus resultados prejudicados de acordo com a estabilidade do dado utilizado para treinamento. Para resolver este problema pode-se aplicar transformações para tornar o dado mais estável, sendo as mais normais a transformada logarítmica ou quadrática. Aplicou-se a transformada logarítmica e também a transformada quadrática sobre a variável de interesse para mudar a relação entre as variáveis de entrada e saída.

Além de transformações simples, como a utilizada apenas sobre o conjunto unidimensional de dados de interesse, é possível aplicar transformações espaciais que levam em consideração as vizinhanças. Para reduzir o ruído presente nos dados brutos das bandas, aplicou-se um filtro de mediana com janela móvel de 15x15 sobre todas as bandas. Essa operação é uma convolução matricial onde o *pixel* de interesse é a mediana dos **n** *pixels* vizinhos, sendo **n** é definido pelo tamanho do kernel, neste caso, 15x15. Para salvar processamento, essa operação foi realizada apenas nos *pixels* da imagem onde há levantamento batimétrico georreferenciado. Futuramente, para previsão de toda área submersa, o algoritmo deve ser aplicado sobre esta, removendo as regiões de terra para evitar processamento desnecessário.

4.8 Seleção de *Features*

É importante notar que nem sempre o aumento da quantidade de *features* ou transformadas refletem necessariamente na melhora da performance do modelo. Isso se dá pois, em alguns casos, as *features* criadas podem ser redundantes ou porque o modelo escolhido não consegue lidar bem com múltiplas colunas de features. Por este motivo é importante selecionar com cautela as *features* a serem usadas, pois além de não adicionar nenhuma relevância ao modelo, adicionam custo computacional. Existem dois processos mais comuns para seleção de *features*, sendo eles:

- ***Backward Selection*:** A partir de N *features*, reduzir o número realizando treinamentos procurando obter o menor conjunto que maximize a qualidade dos resultados do modelo (GUYON; ELISSEEFF 2003).
- ***Forward Selection*:** A partir de uma *feature*, aumentar o conjunto de *features* de modo a obter o melhor conjunto de menor tamanho que maximize a qualidade dos resultados do modelo (GUYON; ELISSEEFF 2003).

O método escolhido para seleção de *features* foi o *Forward Selection*. Este método consiste em começar realizando o treinamento para cada modelo utilizando cada *feature* de maneira isolada e a partir da instância que gerou o melhor resultado na validação, fixar a mesma e adicionar em seguida uma segunda, iterando sobre as restantes. Este processo é repetido iterativamente, aumentando o conjunto de *features* até chegar ao número máximo de instâncias que não altera o resultado do modelo (GUYON; ELISSEEFF 2003).

Para otimizar o tempo de execução dos treinamentos nesta etapa, o processo de seleção de features foi realizado utilizando o XGBoost e a Regressão Linear. Foram escolhidos estes modelos pois a regressão polinomial é uma variação da regressão linear e o XGBoost é um algoritmo de árvore com comportamento similar aos outros modelos não lineares escolhidos, porém com custo de treinamento menor quando comparado à Random Forest e similar comparado ao LGBM.

A melhor combinação de *features* encontrada foi os dados das 8 bandas com a mediana espacial de janela móvel aplicada associado ao dado bruto da batimetria medida. As transformações logarítmica e quadrática não tiveram efeito significativo na performance do modelo, mantendo valores similares aos obtidos na *baseline*. A adição de novos dados utilizando o NDWI, Razão de Log e MPS também não se mostrou justificável trazendo previsões similares à utilização das bandas brutas. Isso pode ser justificado pois esses métodos trazem redundância ao conjunto de dados, uma vez que estes são criados a partir de informações já presentes no dataset.

4.9 Otimização de Hiperparâmetros (*Tunning*)

Durante o processo de treinamento do modelo existem parâmetros que são otimizados automaticamente. Além destes, existem também os que devem ser otimizados manualmente, este grupo de parâmetros é costumeiramente chamado de hiperparâmetros. Existem os hiperparâmetros específicos do modelo utilizado, como por exemplo a *learning rate* de uma rede neural ou a profundidade máxima das árvores em

uma *Random Forest*, porém, também pode-se considerar hiperparâmetros a serem otimizados argumentos das funções de pré-processamento e *feature selection*. Essa grande quantidade de variáveis torna o problema de otimização mais complexo e demorado.

Diversos métodos de otimização de hiperparâmetros podem ser aplicados, sendo os mais comuns *Grid Search*, *Random Search* e *Bayesian Optimization* (Otimização Bayesiana), sendo o último utilizado neste trabalho. Todos os métodos partem de um espaço de hiperparâmetros definido e vão iterando criando diversos modelos buscando sempre o que retorne o melhor resultado, o que os difere é como a iteração acontece.

Diferentemente do *Grid Search*, que consiste em uma busca exaustiva sobre o conjunto total de hiperparâmetros, e do *Random Search* que é uma busca em N iterações com subconjuntos aleatórios do espaço de hiperparâmetros. A Otimização Bayesiana adiciona o princípio Bayesiano à iteração, criando um modelo probabilístico sobre a função objetivo que está sendo otimizada, explorando este modelo para definir qual amostra do espaço de hiperparâmetros tomar (OSBORNE, 2010). A ideia deste método é usar todas informações das iterações anteriores e não depender apenas do gradiente local para convergir. A otimização bayesiana foi adotada pois, apesar de não percorrer todo o espaço de hiperparâmetros, ela tende a convergir mais rapidamente à valores satisfatórios, otimizando também o tempo de execução quando comparada aos outros métodos.

Considerando que a otimização é uma função fixa, bem definida e sem forma conhecida na primeira iteração, é possível afirmar que caso houvesse recurso computacional ilimitado bastaria computar todas as combinações existentes de hiperparâmetros para convergir ao melhor conjunto. Como isto é impossível, na otimização bayesiana algumas amostras são adotadas ao início do processo. A partir destas amostras se cria o modelo substituto (*surrogate model*) para aproximar a função objetivo. Um modelo substituto pode ser definido como a representação probabilística da função objetivo para um certo conjunto de hiperparâmetros, ou seja, $P(função\ objetivo \mid hiperparâmetros)$. Para selecionar a próxima amostra cria-se a função de aquisição e os hiperparâmetros escolhidos na próxima iteração serão os que correspondem ao ponto máximo dessa função. Esse processo é repetido até alcançar o número definido de iterações. Existem diversos tipos de modelos substitutos e funções de aquisição, porém os mais comuns são, respectivamente, *Gaussian Process Model* e *Expected Improvement*, porém isso varia de acordo com a implementação e o objetivo (OSBORNE, 2010).

Neste trabalho, foi realizado Otimização Bayesiana para todos os modelos baseados em árvores, ou seja, para Random Forest, LGBM e XGBoost. Os *batches* de treinamento foram realizados com 100 iterações para cada modelo, buscando otimizar os seguintes hiperparâmetros para cada modelo RF (PEDREGOSA *et al.* 2011), XGBoost (KE. et al. 2017) e LGBM (MICROSOFT. c. 2021):

- ***learning_rate*** (LGBM e XGBoost): Taxa de aprendizado usada durante o boosting. Este hiperparâmetro influencia diretamente no processo de boosting pois as descidas em gradiente estão relacionadas ao seu valor. Caso seja muito

grande, a chance da função não ser capaz de encontrar o mínimo global aumenta, caso seja muito pequeno, o tempo de processamento aumenta.

- ***max_depth*** (RF, LGBM e XGBoost): Profundidade máxima das árvores de decisão do modelo. Caso este parâmetro seja mal ajustado, pode fazer com que as árvores de decisão se ajustem muito aos dados de treino causando *overfitting*.

- ***subsample*** (LGBM e XGBoost): Proporção da amostra utilizada para treinamento. Essa divisão acontece em toda iteração. Reduzir este valor significa pegar aleatoriamente menos dados para criar as árvores durante o treinamento, podendo reduzir o *overfitting*.

- ***colsample_bytree*** (LGBM e XGBoost): Proporção da amostra de colunas ao construir cada árvore. Essa divisão também acontece em toda iteração, e define a proporção da quantidade de colunas dos dados de treino utilizadas para construção de cada árvore.

- ***n_estimators***: (RF, LGBM e XGBoost): Número total de árvores do modelo. Aumentar este número até certo ponto tende a deixar o modelo mais robusto pois mais árvores estão sendo usadas para chegar à previsão final, porém também aumenta o custo computacional.

5 Validação e Resultados Experimentais

Após o treinamento de um modelo é necessário avaliar sua performance para novos conjuntos de dados, sendo esta etapa normalmente chamada de validação. Para realizar uma validação confiável é necessário que o conjunto de dados utilizados neste estágio seja diferente do conjunto usado para treinamento do modelo. Os dados devem ser separados previamente ao treinamento, sendo esta divisão determinada pelo método de validação proposto. Em cada processo ou iteração são extraídos os indicadores definidos na Seção 4.5 para ser possível mensurar a qualidade do modelo e gerar comparações com outros. Neste trabalho foram implementados 3 métodos de validação diferentes, sendo eles:

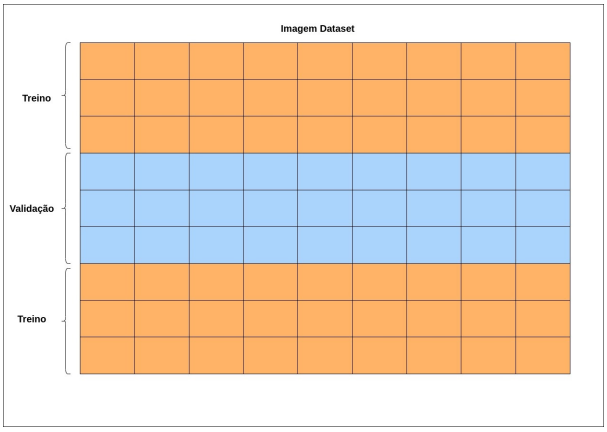
1. Validação Simples: Idêntico ao processo descrito na Seção 4.6 durante a criação da Baseline. Este processo consistiu no *split* aleatório dos dados entre treino e validação com 70% dos dados sendo utilizados para treinamento e 30% para validação. Este método foi utilizado apenas para ter resultados parciais da qualidade dos modelos, uma vez que o conjunto de dados de validação é fixo e pode apresentar algum viés em seus valores invalidando o resultado do processo.

2. Validação Cruzada K-Fold (K-Fold Cross-Validation): Também idêntico ao processo de validação cruzada descrito na Seção 4.6. A validação cruzada K-Fold consiste em dividir o conjunto total de dados em K subconjuntos de mesmo tamanho. Em cada iteração um subconjunto é utilizado para validação e os restantes utilizados

para treinamento (KOHAVI 1995). A partir das métricas obtidas em cada K iteração é possível ter uma estimativa mais confiável da qualidade dos resultados encontrados.

3. Validação de Janela Deslizante: Diferentemente dos outros métodos, a escolha dos dados para treino e validação não será aleatória neste caso. Para realizar este processo é definido uma janela que será deslizada sobre a imagem e os dados utilizados para treino e validação são escolhidos a partir dessas janelas. A representação ilustrada na Figura 13 exemplifica a escolha dos dados para uma iteração deste processo.

Figura 13 - Exemplo de Escolha de Janela



Fonte: Elaborada pelo autor

Os resultados obtidos para a validação simples e a média das K iterações na validação cruzada após a etapa de otimização de hiperparâmetros foram muito similares. A Tabela 4 contém a comparação dos resultados obtidos entre os modelos nestas duas validações. A Tabela 5 mostra os indicadores medidos utilizando a validação de janela deslizante.

Tabela 4 - Resultados comparativos dos modelos após a otimização dos hiperparâmetros para a validação simples e a validação cruzada (K-Fold).

Modelo	**R^2**	**Mean Absolute Error**	**Mean Squared Error**	**Root Mean Squared Error**	**Bias**
Random Forest	0.943	0.544	1.148	1.071	0.033
XGBoost	0.911	0.863	1.788	1.337	0.040
LGBM	0.948	0.545	1.041	1.020	0.028
Regressão Linear	0.479	2.485	10.508	3.241	0.042
Regressão Polinomial	0.624	2.009	7.566	2.750	0.029
CBR RF	0.872	1.176	2.570	1.603	0.109
CBR Linear	0.528	2.359	9.516	3.084	-0.186
CBR Polinomial	0.569	2.142	8.675	2.945	-0.312

Tabela 5 - Resultados comparativos dos modelos após a otimização dos hiperparâmetros para a validação de janela deslizante 10x10 px.

Modelo	R^2	Mean Absolute Error	Mean Squared Error	Root Mean Squared Error	Bias
Random Forest	0.664	1.621	6.294	2.508	-0.037
XGBoost	0.664	1.703	6.285	2.507	-0.084
LGBM	0.688	1.594	5.843	2.417	-0.118
Regressão Linear	0.470	2.416	9.926	3.150	-0.005
Regressão Polinomial	0.631	1.943	6.903	2.627	0.097
CBR RF (n = 2)	0.859	1.282	2.880	1.697	0.109
CBR Linear (n=4)	0.505	2.432	10.135	3.183	-0.150
CBR Poly (n=3)	0.569	2.165	9.733	3.119	-0.390

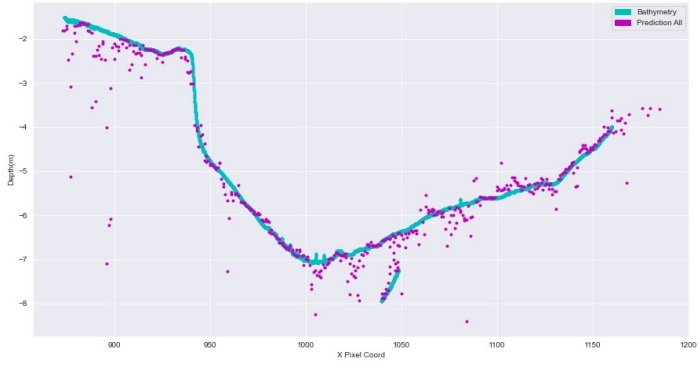
Nota-se que os modelos lineares têm resultados similares aos obtidos nas outras validações, porém os modelos de árvores tendem a apresentar piora nos resultados. Uma das possíveis razões para estes valores estarem razoavelmente diferente dos obtidos nos outros processos é que, os modelos de árvore não são bons em capturar tendências, ou seja, se o dado a ser extrapolado está muito fora do *range* utilizado para treinamento este modelo pode não ser capaz de representar o mesmo de maneira adequada. Como a janela de 10 *pixels* representa uma área de aproximadamente 100x100 metros, pode ser que exista um *gap* de dados significativo para o treinamento que incapacita o modelo a generalizar. Apesar dos modelos não lineares simples terem apresentado resultados inferiores, o conjunto de modelos não lineares com CBR RF não teve alteração significativa no resultado da validação por janela deslizante. Acredita-se que isto ocorre pois, ao treinar um modelo para cada cluster, gera-se um modelo "especialista" para cada região ou tipo de dado. Logo quando um ponto a ser previsto está fora do *range* de domínio de um *modelo A*, a distância desse ponto para o centróide do cluster em que o *modelo A* está contido tende a crescer, fazendo com que a influência deste modelo diminua sobre a previsão.

Para melhor observação dos resultados dos modelos foram gerados mapas de um transecto comparando a batimetria medida com a batimetria estimada pelos algoritmos. O transecto selecionado para observação está marcado na Figura 14 com a cor verde e resolução de 10m/*pixel*.Os resultados obtidos através da BDS para todos os pontos do transecto com treinamento com divisão aleatória dos dados podem ser vistos nas Figuras 15 a 22.

Figura 14 - Transectos batimetria medida e transecto selecionado

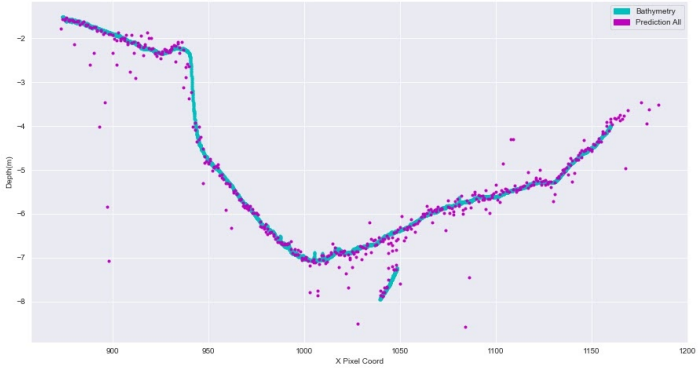
Fonte: Elaborada pelo autor

Figura 15 - Previsão Random Forest



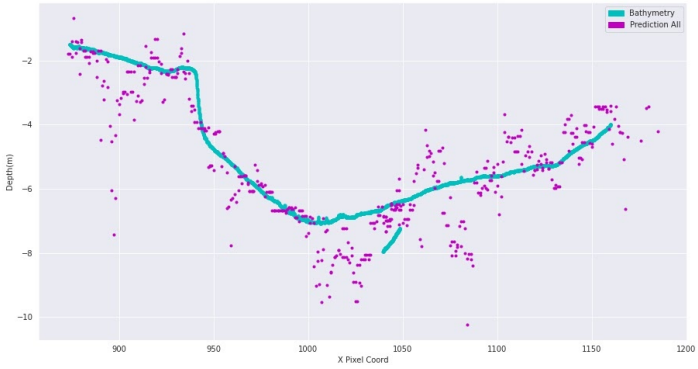
Fonte: Elaborada pelo autor

Figura 16 - Previsão LGBM



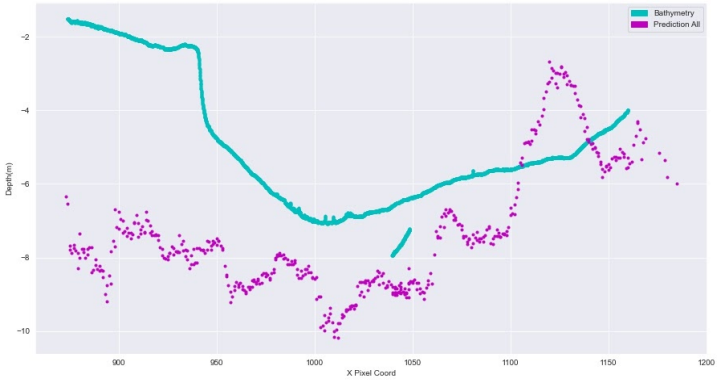
Fonte: Elaborada pelo autor

Figura 17 - Previsão XGBoost



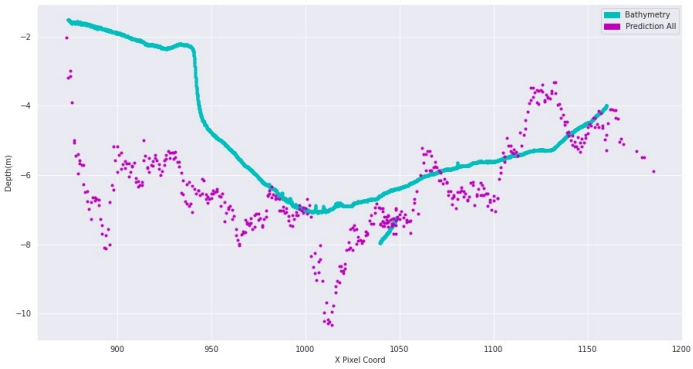
Fonte: Elaborada pelo autor

Figura 18 - Previsão Regressão Linear



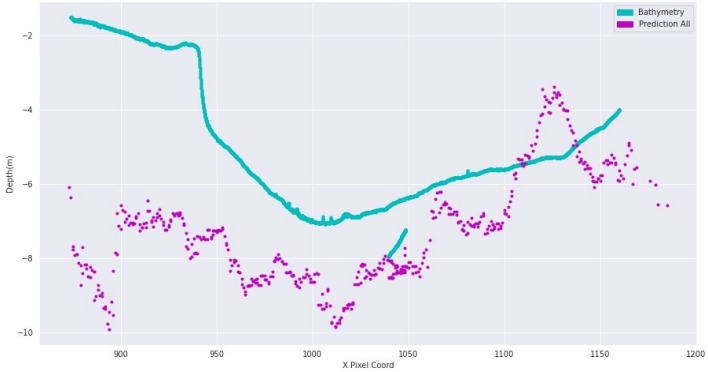
Fonte: Elaborada pelo autor

Figura 19 - Previsão Regressão Polinomial grau 2



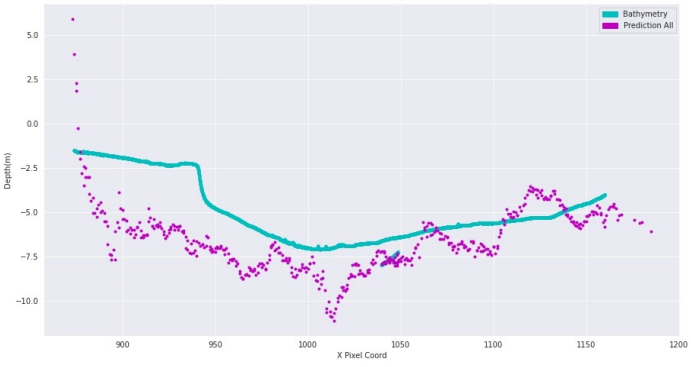
Fonte: Elaborada pelo autor

Figura 20 - Previsão CBR (n=4) Regressão Linear



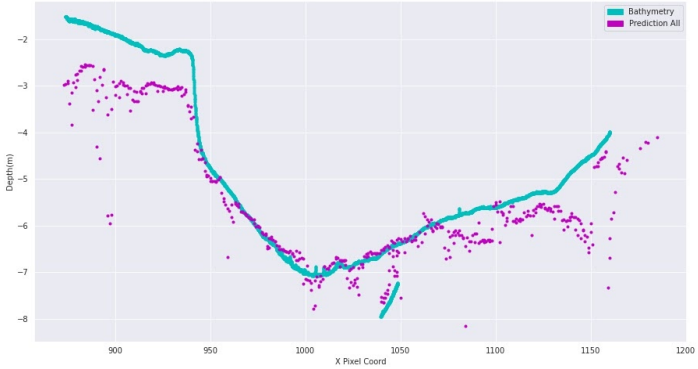
Fonte: Elaborada pelo autor

Figura 21 - Previsão CBR (n=3)
Regressão Polinomial grau 2



Fonte: Elaborada pelo autor

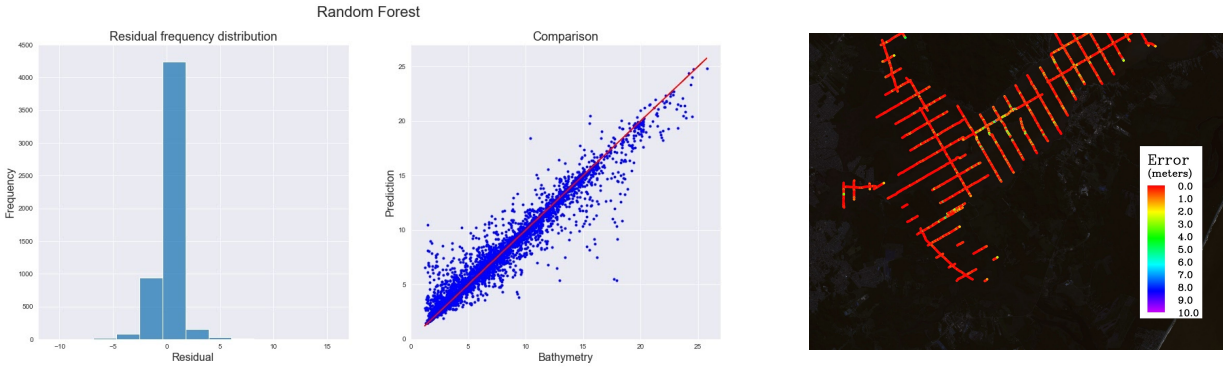
Figura 22 - Previsão CBR (n=2) Random Forest



Fonte: Elaborada pelo autor

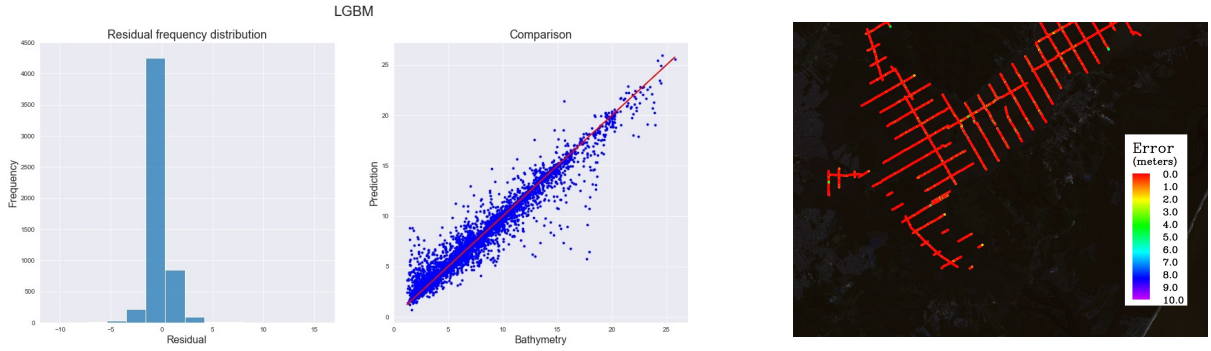
Além dos gráficos dos transectos, foram gerados também gráficos comparativos entre a previsão a partir das imagens de satélite e o medido pelo ecobatímetro, histogramas e mapas de resíduo para cada algoritmo.

Figura 23 - Resultados Random Forest



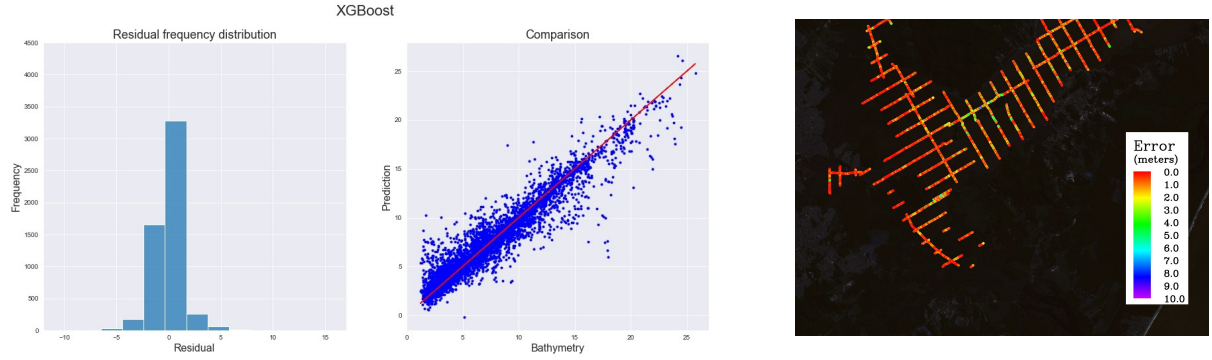
Fonte: Elaborada pelo autor

Figura 24 - Resultados LGBM



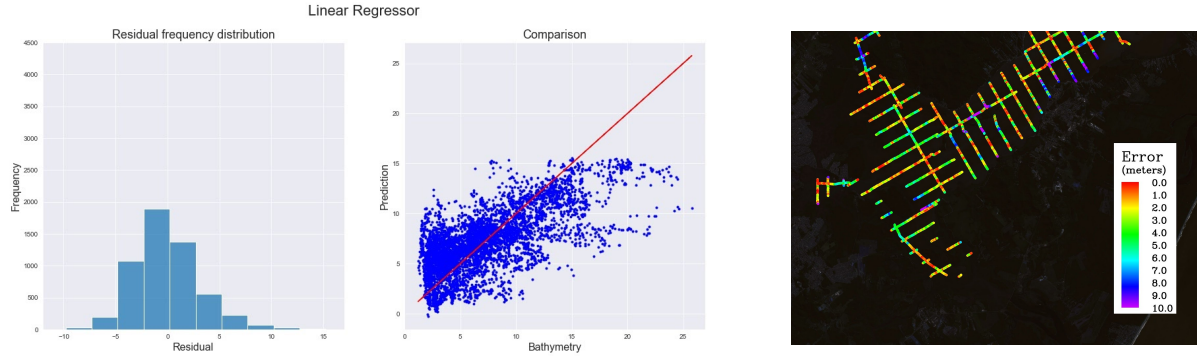
Fonte: Elaborada pelo autor

Figura 25 - Resultados XGBoost



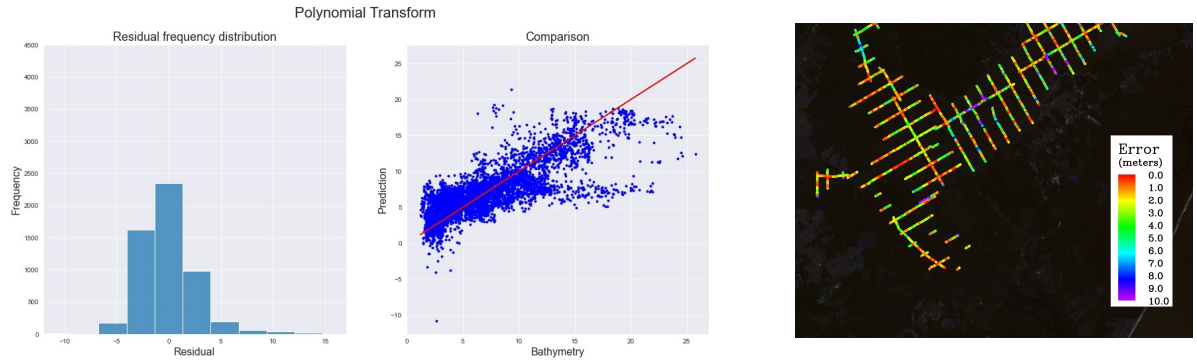
Fonte: Elaborada pelo autor

Figura 26 - Resultados Regressão Linear



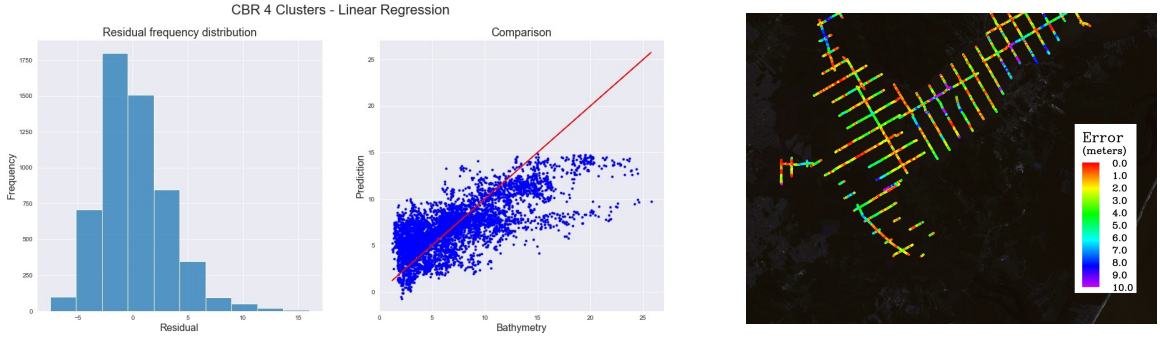
Fonte: Elaborada pelo autor

Figura 27 - Resultados Regressão Polinomial



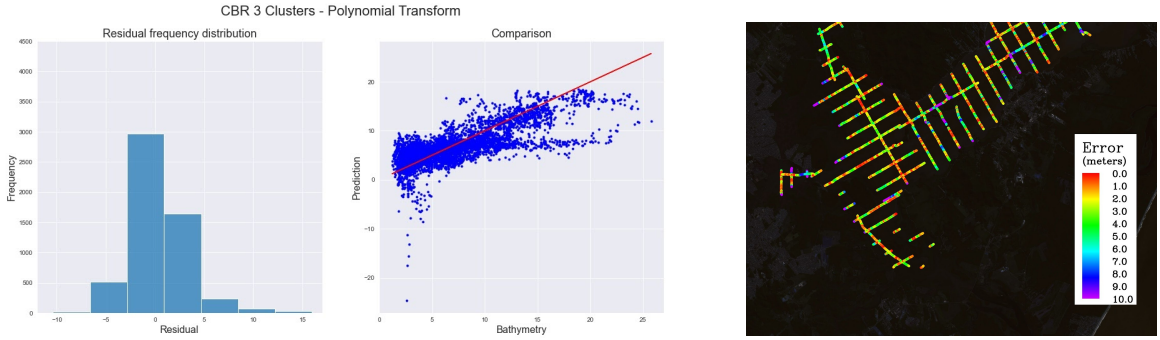
Fonte: Elaborada pelo autor

Figura 28 - Resultados CBR Regressão Linear (n=4)



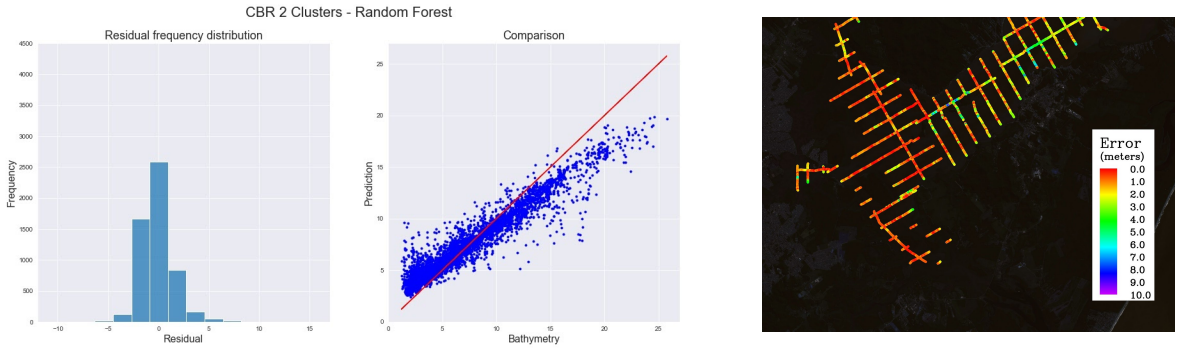
Fonte: Elaborada pelo autor

Figura 29 - Resultados CBR Regressão Polinomial (n=3)



Fonte: Elaborada pelo autor

Figura 30 - Resultados CBR Random Forest (n=2)



Fonte: Elaborada pelo autor

A partir dos gráficos e mapas gerados é possível comparar visualmente a eficácia dos algoritmos. Nota-se que os modelos baseados em *ensemble* de árvores de decisão tem a maior frequência de erros perto do valor 0 e conseguem manter consistência nos resultados mesmo com o aumento da profundidade. Por outro lado, os modelos lineares simples não conseguiram ter a mesma eficácia, principalmente com profundidades superiores à 17~18m. Para o *ensemble* de *Random Forests* ponderando as predições a partir de clusters (CBR), notou-se também a dificuldade de predição sobre profundidades superiores à 17~18m, porém, a distribuição do erro foi significativamente inferior quando comparado aos modelos lineares utilizando esta técnica.

6 Conclusões e Trabalhos Futuros

O presente trabalho apresentou diversos algoritmos capazes de estimar a profundidade de ambientes estuarinos a partir de valores de reflectâncias obtidas através de imagens do satélite Sentinel-2A. Dentro do ambiente de testes, os algoritmos que se mostraram mais eficazes foram os constituídos por conjuntos de árvores de decisão, como LGBM e *Random Forest*. Resultados demonstraram coeficiente de determinação (r^2) de 0.95 e erro absoluto médio de 0.54m, utilizando como dados de entradas os valores das reflectâncias das bandas 1 a 8 do satélite Sentinel-2A com filtro de mediana espacial de janela móvel 15x15 pixels.

Apesar do resultado expressivo, é necessário ter cautela na utilização do método descrito pois os dados de testes constituem em sub-regiões do ambiente de calibração dos modelos, podendo gerar algum viés para o tipo de dado da região. Para realização de uma validação mais robusta, é necessário um ou mais conjuntos de dados com batimetria medida por ecobatímetro para regiões diferentes da Baía da Babitonga. Apesar dos melhores indicadores terem sido obtidos a partir destes modelos, o modelo *CBR Random Forest* se mostrou uma boa opção para situações onde a amostra de dados de treinamento é menor ou não tão representativa se comparada com o conjunto total.

Uma vez realizada a validação para outras áreas, confirmando a capacidade de generalização dos modelos, a implementação proposta pode ser usada para criação de representações da topografia submarina de ambientes costeiros e estuarinos.

Também é desejável desvincular o sistema desenvolvido do ambiente experimental e tornar o *pipeline* de fácil acesso. Uma alternativa é o desenvolvimento de uma *interface* amigável que possibilite aos usuários das áreas beneficiadas, que não possuem conhecimento em programação, utilizarem o sistema de maneira intuitiva. Para isso é possível migrar para um sistema *web*, desacoplando todo *pipeline* proposto em um *backend* escalável responsável por todas etapas de processamento necessárias para criação e validação de um modelo. Com isso, o usuário fica responsável apenas por selecionar a região do mapa onde a batimetria tradicional foi realizada e entrar com um arquivo *.xyz* contendo estes dados.S

Além do benefício de usabilidade do sistema proposto, também é possível armazenar os dados batimétricos entrados pelos diversos usuários, criando uma base de dados robusta, podendo ser utilizada na calibração de um algoritmo genérico para diversas regiões do globo.

REFERÊNCIAS:

ZHENG, A. **Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists.** p. 630. 2018.

ALVES, D. C. L. et al. **Bathymetry Estimation by Orbital Data of OLI Sensor: A Case Study of the Rio Grande Harbor, Southern Brazil.** BioOne Complete. Journal Of Coastal Research. v. 85. p. 51-55. 2018.

CHEN, T.; GUESTRIN, C. **XGBoost: A Scalable Tree Boosting System.** p. 795-794. Ago. 2016.

FREEDMAN D. **Statistical Models: Theory and Practice**. Cambridge University Press 458 p. 2009.

DIRETORIA DE HIDROGRAFIA E NAVEGAÇÃO - DHN. **Normas da Autoridade Marítima para Levantamentos Hidrográficos – NORMAM-25**. Marinha do Brasil. 2011.

EUROPEAN SPACE AGENCY - ESA. **Copernicus Hub.** c. 2000 - 2021. Disponível em <https://sentinels.copernicus.eu/web/sentinel/home> Acesso em 06/05/2021

FERREIRA, I. O. **Coleta, Processamento e Análise de Dados Batimétricos visando a Representação Computacional do Relevo Submerso utilizando Interpoladores Determinísticos e Probabilísticos.** Viçosa. Minas Gerais. 70 p. 2013.

FILIPPI, B. **Obtenção de batimetria em estuários através de imagens de satélite: estudo de caso na Baía da Babitonga, SC .** Repositório Institucional da UFSC. 2020.

GAGG, G. **Apostila de Levantamentos Hidrográficos – Noções Gerais.** IGEO – Instituto De Geociências. Departamento de Geodésia. UFRGS. 41 p. 2016. Disponível em <https://lume.ufrgs.br/bitstream/handle/10183/157210/001020445.pdf?sequence=1&isAllowed=y> Acesso em 08/04/2021.

GEYMAN, E. C.; MALOOF, A. C. **A Simple Method for Extracting Water Depth From Multispectral Satellite Imagery in Regions of Variable Bottom Type.** Department of Geosciences. Princeton University. USA. p. 527-537. 2019.

GLANTZ, STANTON A.; SLINKER, B. K. **Primer of Applied Regression and Analysis of Variance**. McGraw-Hill. 949 p. 1990.

GOODHART C. A. E. **Problems of Monetary Management: The UK Experience.** Palgrave Macmillan. London. 288 p. Dec. 1983.

GUYON I. ELISSEEFF A. **An Introduction to Variable and Feature Selection.** Journal of Machine Learning Research 3. p. 1157-1182. 2003.

HAMYLTON, S. M,; HEDLEY, J. D.; BEAMAN, R. J. **Derivation of High-Resolution Bathymetry from Multispectral Satellite Imagery: A Comparison of Empirical and Optimisation Methods through Geographical Error Analysis.** Remote Sensing. v. 7. n. 12. Dez. 2015.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning**. **Data Mining, Inference, and Prediction.** 2009. In: **Decision Trees.** Scikit-learn. c. 2007-2020.

HINTON, G.; SEJNOWSKI, T. **Unsupervised Learning: Foundations of Neural Computation.** MIT Press. 398 p. 1999.

HU, J.; NIU, H.; CARRASCO, J.; LENNOX, B.; ARVIN, F. **Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning.** IEEE Transactions on Vehicular Technology. v. 69. n. 12. p. 14413-14423. Dez. 2020.

HUNTER, J. et al. **Matplotlib: Visualização com Python.** Versão 3.4.1. Mar. 2021. Disponível em <https://matplotlib.org/> Acesso em 06/05/2021.

JAGALINGAM, P.; AKSHAYA, B. J.; HEGDE, A. V. **Bathymetry Mapping Using Landsat 8 Satellite Imagery**. Department of Applied Mechanics and Hydraulics. National Institute of Technology Karnataka, Surathkal. India. p. 560-566. 2015.

KAM HO, T. **Random Decision Forests.** Internet Archive. 2011-2018. Disponível em **<**https://web.archive.org/web/20160417030218/http://ect.bell-labs.com/who/tkh/publications/papers/odt.pdf> Acesso em 06/05/2021.

KE, G. et al. **LightGBM: A Highly Efficient Gradient Boosting Decision Tree.** 31st Conference on Neural Information Processing Systems. Long Beach. CA. USA. 9 p. 2017.

KOHAVI, R. **A study of cross-validation and bootstrap for accuracy estimation and model selection**. In: International joint Conference on artificial intelligence. v. 14. p. 1137–1145. 1995.

KRUG, L. A.; NOERNBERG, M. A. **O sensoriamento remoto como ferramenta para determinação de batimetria de baixios na Baía das Laranjeiras, Paranaguá - PR.** Scielo. Revista Brasileira de Geofísica. v. 25. São Paulo. 2007.

LIU, C. **Figura Gradiente Descent.** KDNuggets. 2020. Disponível em <https://www.kdnuggets.com/2020/05/5-concepts-gradient-descent-cost-function.html> Acesso em 06/05/2021.

LIU, H. et al. **Application of Sentinel 2 MSI Images to Retrieve Suspended Particulate Matter Concentrations in Poyang Lake.** Remote Sensing. 2017.

MANESSA, M. D. M., et al. **Satellite-derived Bathymetry Using Random Fnumporest Algorithm And Worldview-2 Imagery.** Geoplanning Journal of Geomatics and Planning. v 3. n. 2. p. 117-126. 2016.

MAZZER, A. M.; GONÇALVES, M. L. **Aspectos Geomorfológicos da Baía da Babitonga, Santa Catarina, Brasil: Caracterização Morfométrica.** Revista Brasileira de Geomorfologia. v. 12. n. 3. p. 115-120. 2011

McFEETERS, S. K. **The use of the Normalized Difference Water Index (NDWI) in the delineation of open water features.** Remote Sensing. International Journal of Remote Sensing. v. 17. p. 1425-1432. 1996.

MICROSOFT CORPORATION. **SHAP.** c. 2021. Disponível em <https://github.com/slundberg/shap> Acesso em 28/04/2021.

MICROSOFT CORPORATION. **LightGBM.** c. 2021 Disponível em <https://lightgbm.readthedocs.io/en/latest/Features.html> Acesso em 24/04/2021

MUNDOGEO. **Levantamentos Batimétricos.** Editora MundoGeo Ltda. Paraná. 2012. Disponível em <https://mundogeo.com/2005/09/27/levantamentos-batimetricos/> Acesso em 06/05/2021.

NIROUMAND-JADIDI, M.; VITTIA, A.; LYZENGAD, D. R. **Multiple Optimal Depth Predictors Analysis (MODPA) for river bathymetry: Findings from spectroradiometry, simulations, and satellite imagery.** ScienceDirect. v. 218. p. 132-147. Dec. 2018.

NUMPY. **NumPy v 1.20.0. Type annotation support - Performance improvements through multi-platform SIMD.** 30 jan. 2021. Disponível em <https://numpy.org/> Acesso em 06/05/2021.

NumFOCUS. **PANDAS.** Versão 1.2.4. Abr. 2021. Disponível em <https://pandas.pydata.org/> Acesso em 06/05/2021.

OSBORNE, M. A. **Bayesian Gaussian processes for sequential prediction, optimisation and quadrature**. Oxford University. UK. 2010

PEDREGOSA, F. et al. **Scikit-learn: Machine Learning in Python.** Journal of Machine Learning Research. v. 12. p. 2825-2830. 2011.

PHILPOT, W. D. **Bathymetric mapping with passive multispectral imagery.** Applied Optics. v. 28. n. 8. p. 1569-1578. 1989.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. Prentice Hall. Series in Artificial Intelligence. New Jersey. 946 p. 1995.

SAGAWA, T. et al. **Satellite Derived Bathymetry Using Machine Learning and Multi-Temporal Satellite Images**. Remote Sensing. 14 Maio 2019.

SAMUEL, A. L. **Some Studies in Machine Learning Using the Game of Checkers.** IBM Journal of Research and Development. v. 3. 3 ed. p. 535-554. 1959.

SCIKIT-LEARN. **Machine Learning in Python.** Versão 0.24.1. Jan. 2021. Disponível em <https://scikit-learn.org/stable/> Acesso em 06/05/2021.

STUMPF, R. P.; HOLDERIED, K.; SINCLAIR, M. **Determination of water depth with high-resolution satellite imagery over variable bottom types.** Limnology And Oceanography. v. 48. n. 12. p. 547-556. Jan. 2003.

GOHLKE, C. **TIFFFILE 2021.4.8.** Laboratory for Fluorescence Dynamics. University of California, Irvine. Apr. 2021. Disponível em <https://pypi.org/project/tifffile/> Acesso em 06/05/2021.

KUTSER, T.; VAHTMÄE, E.; PAAVEL, B.; KAUER, T. **Removing glint effects from field radiometry data measured in optically complex coastal and inland waters.** Remote Sensing of Environment v. 133. p. 85-89. 15 jun. 2013.

VENNERS, B. **The Making of Python.** A Conversation with Guido van Rossum. Artima. Jan. 2003. Disponível em < https://www.artima.com/articles/the-making-of-python> Acesso em 06/05/2021

WASKOM, M. **Seaborn: statistical data visualization.** Seaborn. c. 2012-2020. Disponível em <https://seaborn.pydata.org/> Acesso em 06/05/2021

XAVIER, M.; HARRIS, P.; CALOCA, S.; CAHALANE, C. **Spatial Prediction of Coastal Bathymetry Based on Multispectral Satellite Imagery and Multibeam Data.** Remote Sensing. 2015.

XGBOOST. **Introduction to Boosted Trees.** XGBoost Tutorials c. 2020. Disponível em <https://xgboost.readthedocs.io/en/latest/tutorials/model.html> Acesso em 06/05/2021