

JOÃO PEDRO NUNES OLIVEIRA

**OTIMIZAÇÃO DE PERFORMANCE EM FUNCIONALIDADES
CRÍTICAS DO PLUGIN *SMART-MAP* PARA O QGIS**

Trabalho de Conclusão de Curso apresentado ao
Câmpus Muriaé, do Instituto Federal de Educação
Ciência e Tecnologia do Sudeste de Minas Gerais,
como parte das exigências do Curso de Graduação
de Tecnologia em Gestão da Tecnologia da
Informação para a obtenção do título de Tecnólogo.

**MURIAÉ
MINAS GERAIS – BRASIL
2024**

JOÃO PEDRO NUNES OLIVEIRA

**OTIMIZAÇÃO DE PERFORMANCE EM FUNCIONALIDADES
CRÍTICAS DO PLUGIN *SMART-MAP* PARA O QGIS**

Trabalho de Conclusão de Curso apresentado ao
Câmpus Muriaé, do Instituto Federal de Educação
Ciência e Tecnologia do Sudeste de Minas Gerais,
como parte das exigências do Curso de Graduação
de Tecnologia em Gestão da Tecnologia da
Informação para a obtenção do título de Tecnólogo.

Orientador: Prof. Dr. Gustavo Willam Pereira

**MURIAÉ
MINAS GERAIS – BRASIL
2024**

JOÃO PEDRO NUNES OLIVEIRA

**OTIMIZAÇÃO DE PERFORMANCE EM FUNCIONALIDADES
CRÍTICAS DO PLUGIN *SMART-MAP* PARA O QGIS**

Trabalho de Conclusão de curso apresentado ao
Câmpus Muriaé, do Instituto Federal de Educação
Ciência e Tecnologia do Sudeste de Minas Gerais,
como parte das exigências do curso de graduação em
Tecnologia em Gestão da Tecnologia da Informação
para a obtenção do título de tecnólogo.

APROVADO:

Prof. Gustavo Willam Pereira (Orientador)

Prof. Diego Rossi

Prof. Marcus Vinicius Souza Costa

Prof. Paulo Vinicius Moreira Dutra

**MURIAÉ
MINAS GERAIS – BRASIL
2024**

Com imensa gratidão e carinho, dedico este trabalho aos meus pais, Edneia e Lécio. Agradeço por todo o sacrifício e apoio incondicional ao longo desta jornada acadêmica. Sem o suporte e amor que sempre me proporcionaram, este momento não seria possível.

AGRADECIMENTOS

A Deus, pela minha vida e por ter me acompanhado ao longo desta jornada.

Aos meus pais, Lécio Nunes Vieira e Edneia Silva de Oliveira Vieira, por sempre me apoiarem e pelo profundo compromisso com a minha educação.

Ao professor Carlos Antônio Baldanza, por me ensinar que desistir nunca é uma opção, tanto através de seus ensinamentos no esporte quanto pela sua história e pelo exemplo que é como pessoa.

À Escola São Paulo e a todos os professores da instituição, por todo apoio durante a minha formação básica, a qual possibilitou que eu chegasse até esse momento.

Aos professores do curso de Gestão da Tecnologia da Informação do IF SUDESTE-MG, por todo apoio, companheirismo e conhecimentos adquiridos nas disciplinas.

Ao professor Gustavo Willam Pereira, por sempre ter acreditado em meu potencial, pela amizade, confiança, ensinamentos e conselhos.

Tecnologia em Gestão da Tecnologia da Informação

OTIMIZAÇÃO DE PERFORMANCE EM FUNCIONALIDADES CRÍTICAS DO PLUGIN *SMART-MAP* PARA O QGIS

RESUMO

João Pedro Nunes Oliveira

Dezembro, 2024

Orientador: Prof. Dr. Gustavo Willam Pereira

O *Smart-Map* é um plugin do QGIS, *software* para geração de mapas, que possibilita a interpolação de atributos do solo através do método geoestatístico Krigagem Ordinária e do algoritmo de *Machine Learning Support Vectors Machine*. Entretanto, o *software* conta com algumas limitações em relação ao número de pontos amostrados e tempo de execução. Dessa forma, pretendeu-se melhorar a performance do *plugin* na duração do processamento e na quantidade de dados a serem utilizados na realização das operações. As principais mudanças feitas na versão original foram a troca do tipo de validação usada, antes *Leave-One-Out*, agora *K-Fold* e alteração de códigos menos eficientes em termos de carregamento. Com esses ajustes, foi possível observar um aumento significativo na velocidade de execução das funcionalidades do *plugin*, bem como o melhoramento do limite de dados usados para realizar as funcionalidades da ferramenta, na versão original 5.000 registros, com as atualizações, de acordo com as configurações do sistema computacional utilizados, até 50.000.

Palavras-chave: *Machine Learning*. Performance. Validação Cruzada. *Smart-Map*.

Technical Degree in Information Technology Management

PERFORMANCE OPTIMIZATION IN CRITICAL FEATURES OF THE SMART-MAP PLUGIN FOR QGIS

SUMMARY

João Pedro Nunes Oliveira

December, 2024

Academic Advisor: Dr. Gustavo Willam Pereira

Smart-Map is a plugin of QGIS, software to generate maps, which provides the ground's attributes interpolation by the geostatistical method Ordinary Kriging and by the machine learning algorithm called Support Vectors Machine. However, the application has some limitations related with the number of sampled points used and execution time. In this way, the aim was to improve the performance of the plugin in terms of process duration and amount of data used. The main changes made in the original version were the alteration of the type of validation utilized, before Leave-One-Out, now K-Fold and the change of less efficient codes in terms of loading. With these adjustments, it was possible to observe a significant increase in the execution velocity of the plugin's functionalities, as well as the improvement of the limit of the amount of data used during the tool's operations, in the original version 5,000 registers, with the updates, according to the settings of the computational system utilized, until 50,000.

Keywords: Machine Learning. Performance. Cross-Validation. Smart-Map.

LISTA DE ILUSTRAÇÕES

Figura 1. Exemplo das etapas de um projeto de <i>Machine Learning</i>	17
Figura 2. Representação do hiperplano, margem máxima e vetores de suporte	18
Figura 3. Exemplo da transformação da dimensão dos dados pelo <i>Kernel Trick</i>	19
Figura 4. Exemplo de diferentes valores para o parâmetro C	20
Figura 5. Exemplo de diferentes valores para o parâmetro <i>Gamma</i>	20
Figura 6. Representação dos hiperplanos do SVM e SVR	21
Figura 7. Representação do <i>Leave-One-Out Cross-Validation</i>	22
Figura 8. Representação do <i>K-Fold Cross-Validation</i>	23
Figura 9. Exemplo de Paginação de Dados	25
Figura 10. Etapas ajustadas do projeto destacadas em vermelho	26
Figura 11. Interface de importação dos dados	28
Figura 12. Visualização dos dados selecionados	29
Figura 13. Interface para validação dos algoritmos e geração dos mapas	30
Figura 14. Versão utilizando o método nativo de leitura de dados vetoriais do QGIS	31
Figura 15. Versão utilizando o método da biblioteca Pandas	32
Figura 16. Processo original de injeção dos dados na tabela nativa do QGIS	33
Figura 17. Uso de paginação para gerenciar a injeção de dados na tabela do QGIS	34
Figura 18. Variação original utilizando o LOOCV na Krigagem Ordinária	35
Figura 19. Variação implementada utilizando o <i>K-Fold</i> na Krigagem Ordinária	35
Figura 20. Variação original utilizando o LOOCV no <i>Support Vector Machines</i>	36
Figura 21. Variação implementada utilizando o <i>K-Fold</i> no <i>Support Vector Machines</i>	36
Figura 22. Lógica e número de <i>folds</i> usados originalmente	37
Figura 23. Lógica e número de <i>folds</i> alterados	37

LISTA DE ABREVIATURAS E SIGLAS

ML - *Machine Learning*

IA - Inteligência Artificial

SVM - *Support Vector Machines*

SVR - *Support Vector Regression*

KO - krigagem Ordinária

CV - *Cross-Validation*

LOOCV - *Leave-One-Out Cross-Validation*

SUMÁRIO

1. INTRODUÇÃO	14
1.1 OBJETIVOS	15
1.1.1 Objetivo Geral	15
1.1.2 Objetivos Específicos	15
1.2 JUSTIFICATIVA	16
1.3 ORGANIZAÇÃO DO TEXTO	16
2. REVISÃO BIBLIOGRÁFICA	17
2.1 <i>MACHINE LEARNING</i>	17
2.1.1 <i>SUPPORT VECTOR MACHINES (SVM)</i>	18
2.1.2 <i>SUPPORT VECTOR REGRESSION (SVR)</i>	21
2.2 <i>CROSS-VALIDATION (CV)</i>	21
2.2.1 <i>LEAVE-ONE-OUT CROSS-VALIDATION (LOOCV)</i>	22
2.2.2 <i>K-FOLD CROSS-VALIDATION</i>	23
2.3 PAGINAÇÃO DE DADOS	24
3. METODOLOGIA	25
3.1 TECNOLOGIAS UTILIZADAS	27
3.2 AMBIENTE DE DESENVOLVIMENTO	27
3.3 PROCESSOS	27
4. RESULTADOS	31
4.1 OTIMIZAÇÃO NO CARREGAMENTO DOS DADOS	31
4.2 ALTERAÇÃO DO MÉTODO DE <i>CROSS-VALIDATION</i> UTILIZADA	34
4.3 OTIMIZAÇÃO NA CRIAÇÃO DO MAPA INTERPOLADO	37
5. CONSIDERAÇÕES FINAIS	38
REFERÊNCIAS BIBLIOGRÁFICAS	39

1. INTRODUÇÃO

Nos últimos anos, a integração de algoritmos de *Machine learning (ML)* no mapeamento digital de atributos do solo tem se mostrado uma alternativa eficaz aos métodos convencionais e geoestatísticos. Esses modelos, reconhecidos por sua capacidade de integrar diversas camadas de informação como covariáveis, oferecem mais flexibilidade e precisão em análises geoespaciais. Porém, a complexidade e a variedade de algoritmos disponíveis podem ser um obstáculo para sua adoção por usuários finais, principalmente aqueles sem um conhecimento profundo em Aprendizado de Máquina.

Dessa forma, o *plugin Smart-Map* surge como uma ferramenta valiosa, sendo uma extensão para o Sistema de Informação Geográfica (SIG) QGIS. Este trabalho se propõe a aprimorá-lo em aspectos essenciais para a experiência positiva do usuário. Foram realizadas melhorias significativas no processamento dos dados e na velocidade de carregamento de alguns dos processos da aplicação, tornando o uso do *app* mais eficiente e acessível.

O *Smart-Map* combina o método geoestatístico Krigagem Ordinária (OK) com o modelo de ML *Support Vector Machines (SVM)*, possibilitando a criação de mapas interpolados com maior eficiência e precisão. Os ajustes feitos não só aumentam a utilidade prática do *plugin*, mas também reforçam seu papel como uma ferramenta crucial para profissionais que atuam com geociências e mapeamento digital de solos. Além, é claro, de ser *Open Source*, o que contribui com o crescimento da comunidade de desenvolvedores e pesquisadores.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

- Aprimorar a performance de execução de processos, como: visualização de dados em formato de tabela, treinamento do algoritmo utilizado, processos internos da Krigagem e criação de mapas interpolados.

1.1.2 Objetivos Específicos

- Compreender o funcionamento do *software* para melhorar a precisão na identificação e resolução de eventuais problemas, possibilitando uma maior assertividade na resolução dos problemas.
- Otimizar o processamento de dados, reduzindo o tempo de execução das operações.
- Aumentar a eficiência no carregamento dos dados, minimizando o tempo necessário para a visualização de grandes volumes de registros.
- Melhorar o tempo de execução do algoritmo SVM e da Krigagem Ordinária para uma melhor experiência do usuário.
- Reduzir o tempo de criação do mapa interpolado quando apresentado grandes volumes de dados sem perder a qualidade da imagem.
- Avaliar o impacto das otimizações no uso do Smart-Map por meio de estudos de caso aplicados em diferentes tipos e tamanhos de dados.

1.2 JUSTIFICATIVA

As adaptações realizadas no plugin *Smart-Map* se justificam pela necessidade de otimizar o processamento e a velocidade de execução das operações suportadas pela ferramenta, primordiais para uma usabilidade mais eficiente e ágil no QGIS. Essas melhorias tornam o *plugin* mais acessível e útil para profissionais que necessitam de análises geoespaciais precisas, mas que lidam com limitações em termos de desempenho e tempo de resposta, devido ao grande volume de dados.

1.3 ORGANIZAÇÃO DO TEXTO

O trabalho está estruturado da seguinte maneira:

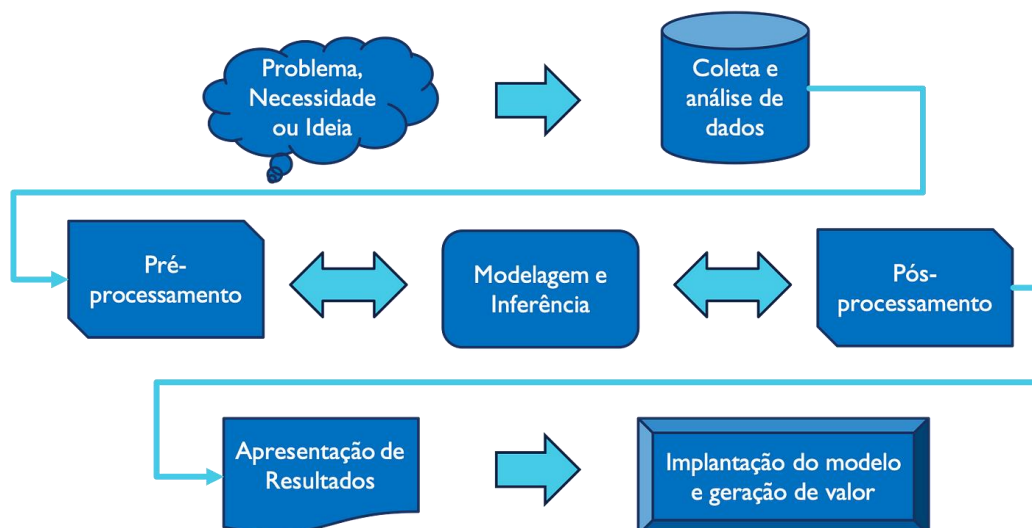
1. Introdução: Apresenta o contexto geral do trabalho, os objetivos da pesquisa e a justificativa para as melhorias efetuadas no plugin *Smart-Map*.
2. Revisão Bibliográfica: Explora os conceitos de Machine Learning, Cross-Validation e Paginação na linguagem de programação Python, fornecendo o embasamento teórico necessário para compreender as técnicas aplicadas e os ajustes implementados.
3. Metodologia: Detalha as funcionalidades e processos do sistema que foram alvo de melhorias, justificando o motivo destas otimizações.
4. Resultados: Apresenta e analisa os resultados obtidos com as melhorias, comparando as metodologias desenvolvidas para o plugin originalmente e depois com as alterações implementadas, apresentando o impacto das mudanças na eficiência do software.
5. Considerações Finais: Resumo das principais descobertas do projeto, destaca a contribuição das otimizações efetuadas e propõe direções para futuras pesquisas e desenvolvimento no contexto do *Smart-Map*.

2. REVISÃO BIBLIOGRÁFICA

2.1 MACHINE LEARNING

Conforme descrito no site da Oracle (2024), “*Machine Learning* é o subconjunto da inteligência artificial (IA) que se concentra na construção de sistemas que aprendem, e melhoram o desempenho, com base nos dados que consomem” (Oracle, 2024). O conceito de ML tem ganhado muito espaço em diversas áreas do conhecimento, devido à sua capacidade de lidar com grandes quantidades de dados e identificar padrões complexos que seriam difíceis de identificar utilizando abordagens tradicionais.

Figura 1. Exemplo das etapas de um projeto de *Machine Learning*

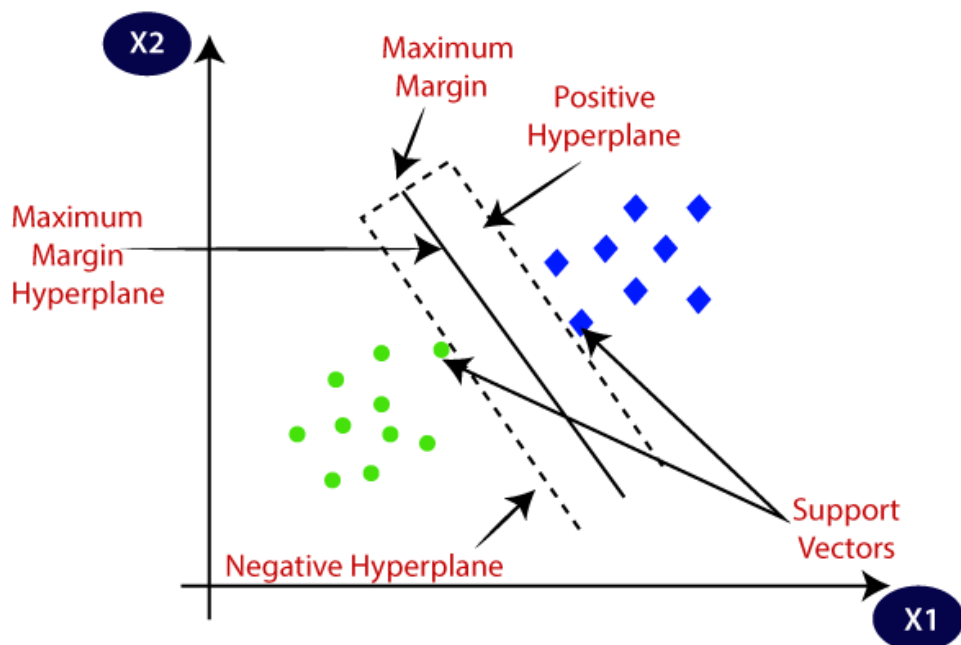


Fonte: ESCOVEDO & KOSHIYAMA (2020)

2.1.1 SUPPORT VECTOR MACHINES (SVM)

No cenário do mapeamento digital de atributos do solo, Aprendizado de Máquina se mostra especialmente eficaz. A possibilidade de unir diferentes camadas de informação, como dados vetoriais e raster, torna os algoritmos de ML ideais para previsões geoespaciais. Um exemplo comum é o uso do *Support Vector Machines* (SVM), um modelo supervisionado que é eficaz em classificar dados em espaços de alta dimensionalidade, devido à sua capacidade de encontrar a melhor fronteira de separação entre rótulos possível para um certo conjunto de dados, sendo chamada, para o SVM, de hiperplano (Matheus Remigio, 2020).

Figura 2. Representação do hiperplano, margem máxima e vetores de suporte

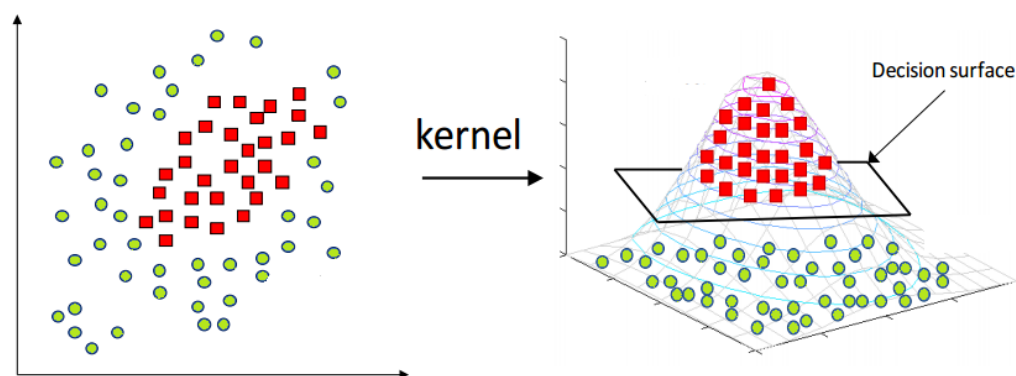


Fonte: ANSHUL SAINI (2024)

Porém, a imagem acima reflete um hiperplano encontrado numa base de dados em que as classes são linearmente separáveis, contudo, é possível trabalhar com esse modelo em problemas não lineares através de técnicas desenvolvidas para tratar de problemas mais complexos, como o *kernel Trick* e a otimização dos hiper parâmetros para tornar o algoritmo mais flexível (Matheus Remigio, 2020).

O *kernel Trick* é uma técnica que aplica uma transformação não linear no espaço, através de uma função, em que é alterada a dimensão original dos dados para uma maior, com o objetivo de tornar a distribuição dos dados linear para facilitar a descoberta do melhor hiperplano (Drew Wilimitis, 2018).

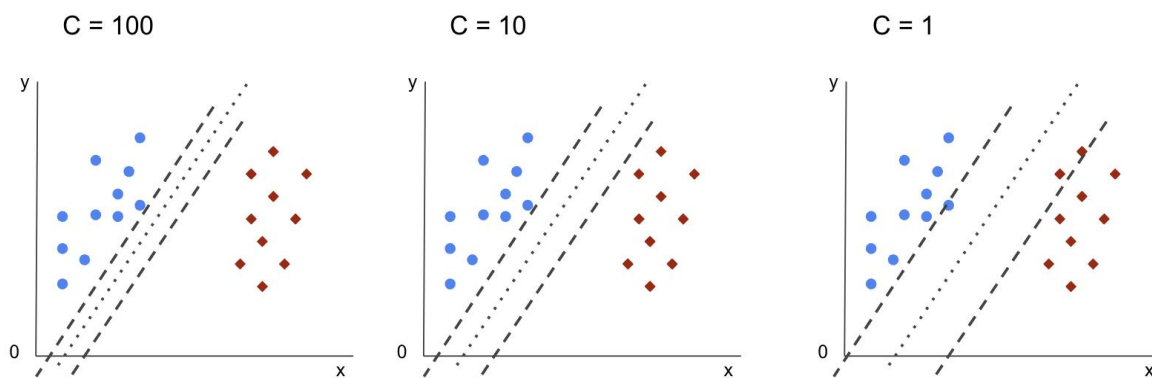
Figura 3. Exemplo da transformação da dimensão dos dados pelo *Kernel Trick*



Fonte: GRACE ZHANG (2018)

A otimização dos hiper parâmetros pode ser uma técnica essencial para a melhora de performance do modelo na identificação de padrões mais difíceis, já que são passados diretamente para o algoritmo tendo uma relação direta com os resultados encontrados. O SVM traz dois principais parâmetros de controle: C e Γ .

O Parâmetro C é inversamente proporcional ao tamanho da margem, ou seja, valores maiores para C ocasionam um espaço menor entre os vetores de suporte, o contrário também é verdade, números menores para C produzem margens maiores. Pode ser ajustável em qualquer tipo de *Kernel* sendo responsável por controlar o grau de penalização para erros na classificação das amostras de treinamento, influenciando diretamente a precisão do algoritmo (Cássia Sampaio, 2023).

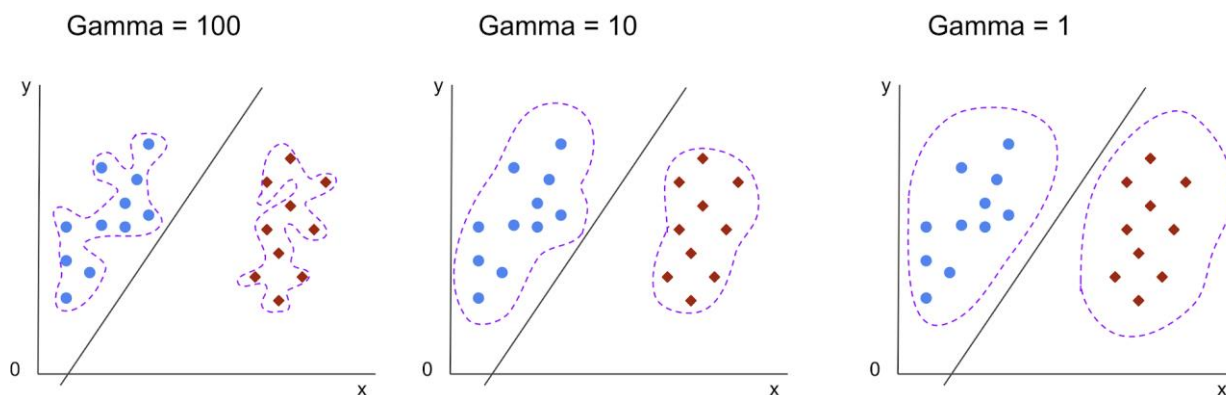
Figura 4. Exemplo de diferentes valores para o parâmetro C

Fonte: Cássia Sampaio (2023)

Existem inúmeras possibilidades de fronteiras de decisão (hiperplano), e enquanto algumas conseguirão separar as classes de forma eficaz, outras não. No momento de definir uma fronteira de decisão eficiente é preciso levar em conta se serão considerados apenas pontos mais próximos de cada classe ou incluir também pontos mais distantes. No *Support Vector Machines*, essa regulação de alcance é controlada pelo hiperparâmetro *Gamma*.

Assim como o parâmetro C, o *Gamma* é inversamente proporcional à distância, quando seu valor é alto, o modelo abrange apenas os pontos mais próximos ao definir a fronteira de Decisão, por outro lado, quando seu valor é baixo, pontos distantes também são considerados na escolha do hiperplano. Dessa forma, a escolha do valor adequado para gamma é fundamental para equilibrar a complexidade do modelo e sua capacidade de generalização.

Figura 5. Exemplo de diferentes valores para o parâmetro *Gamma*

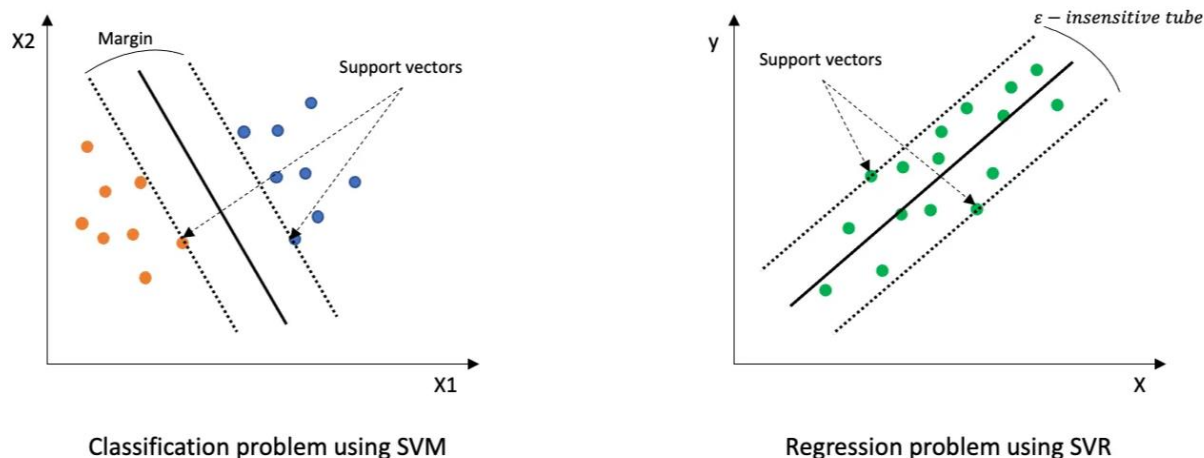


Fonte: Cássia Sampaio (2023)

2.1.2 SUPPORT VECTOR REGRESSION (SVR)

No desenvolvimento do *plugin Smart-Map*, foi utilizado o *Support Vector Regression (SVR)*, o qual é uma adaptação do *Support Vector Machines* voltada para problemas de regressão. Enquanto o SVM busca a melhor separação entre as classes com uma margem maximizada, o SVR tende a otimizar uma função de regressão de modo que a diferença entre as previsões e os valores reais fiquem dentro de uma margem de erro definida. Ambos os algoritmos citados acima utilizam hiperparâmetros como C e Γ , porém o SVR introduz o parâmetro ϵ -insensitive tube para gerenciar a margem de erro permitida.

Figura 6. Representação dos hiperplanos do SVM e SVR



Fonte: Niousha Rasifaghihi (2023)

2.2 CROSS-VALIDATION (CV)

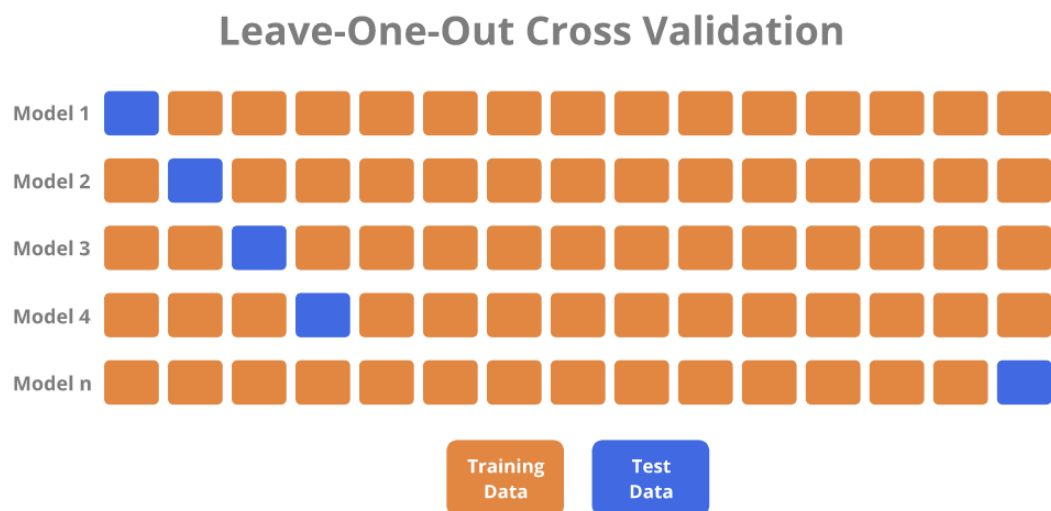
Uma técnica essencial em aprendizado de máquina e estatística para avaliar o desempenho de um modelo na generalização de novos dados é o *Cross-Validation*, que ao contrário do método tradicional de simplesmente dividir o conjunto de dados em partes fixas para treinamento e teste, envolve a divisão dos dados em múltiplos subconjuntos, permitindo que o modelo seja testado em várias amostras diferentes, proporcionando uma maior confiabilidade nas métricas escolhidas para avaliar a performance do algoritmo, auxiliando na identificação de possíveis problemas de *overfitting* ou *underfitting*.

2.2.1 LEAVE-ONE-OUT CROSS-VALIDATION (LOOCV)

O Leave-One-Out Cross-Validation é uma variação mais simples da Validação Cruzada. Neste método, o conjunto de dados é dividido em tantos subconjuntos quantas forem as observações disponíveis. Cada iteração é criada levando em consideração todas as amostras, exceto uma que é usada para teste. Esse processo se repete até que cada ponto tenha sido utilizado como conjunto de teste uma vez. Embora essa técnica possa fornecer estimativas muito precisas sobre o desempenho

de um algoritmo, ela pode ser extremamente custosa computacionalmente, principalmente em *datasets* com grande quantidade de registros, pois exige o treinamento do modelo tantas vezes quanto o número de observações (Sklearn).

Figura 7. Representação do *Leave-One-Out Cross-Validation*



Fonte: BIJEN PATEL (2020)

2.2.2 K-FOLD CROSS-VALIDATION

Uma das variações mais comuns usadas de *Cross-Validation* é o *K-Fold Cross-Validation*, por ser menos intensivo computacionalmente do que o LOOCV e ainda oferecer ótimas estimativas de generalização. Nesse método, os dados são separados em k subconjuntos, ou *folds*. O modelo é treinado k vezes, e em cada uma delas utilizando $k-1$ *folds* para treino e o fold restante para teste. Os resultados obtidos de cada iteração são combinados para fornecer uma estimativa média do algoritmo. O valor de k fica a critério do usuário, porém não é possível ultrapassar o número de observações do banco. Geralmente, para equilibrar a quantidade de iterações e o tamanho dos subconjuntos, valores como $k=5$ ou $k=10$ são amplamente usados (Sklearn).

Figura 8. Representação do *K-Fold Cross-Validation*



Fonte: Wikipedia (2024)

2.3 PAGINAÇÃO DE DADOS

O conceito de paginação no contexto de grandes conjuntos de dados envolve dividir os dados em partes menores e mais gerenciáveis, chamadas de páginas ou blocos (*chunks*). Essa técnica pode ser especialmente útil, pois processar blocos menores de registros é mais rápido do que carregar o *dataset* por completo, além de reduzir a carga computacional no sistema (Teamcode, 2023).

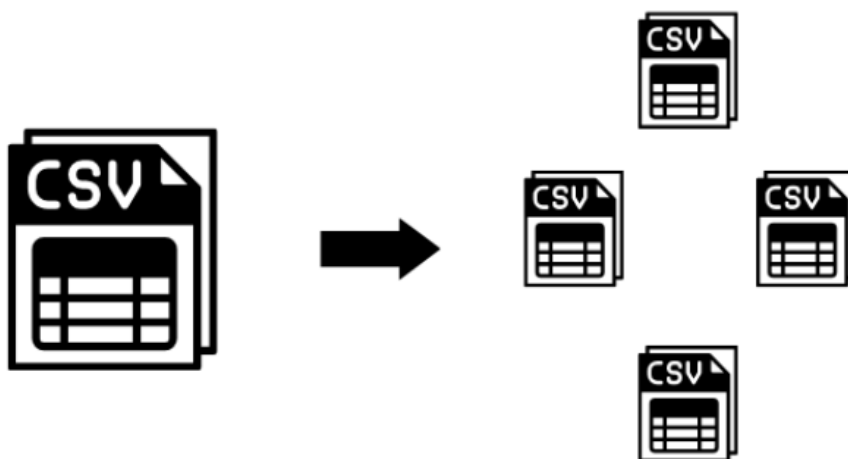
Ademais, o conceito do carregamento em *chunks* é comumente usado em diversos *softwares*, melhorando a experiência do usuário ao evitar longos tempos de carregamento e tornando a navegação pelos dados mais dinâmica. Ao trabalhar com paginação, é preciso compreender alguns termos:

- O **tamanho da página** refere-se à quantidade de linhas que serão carregadas e processadas por vez. Essa variável define o volume de dados que será apresentado ao usuário num determinado momento,

sendo um fator essencial para eficácia e a experiência do usuário no acesso a grandes volumes de dados.

- O **número total de itens** corresponde ao tamanho completo da quantidade de registros de um *dataset*. Esse valor geralmente é utilizado para calcular o número ideal de páginas necessárias para exibir todos os elementos.
- A **página atual indica** qual parte do processo está sendo visualizada ou acessada pelo usuário numa dada fase do carregamento. Isso ajuda a garantir que os dados não sejam perdidos ao percorrer o arquivo em etapas.

Figura 9. Exemplo de Paginação de Dados



Fonte: Elaborado pelo autor (2024)

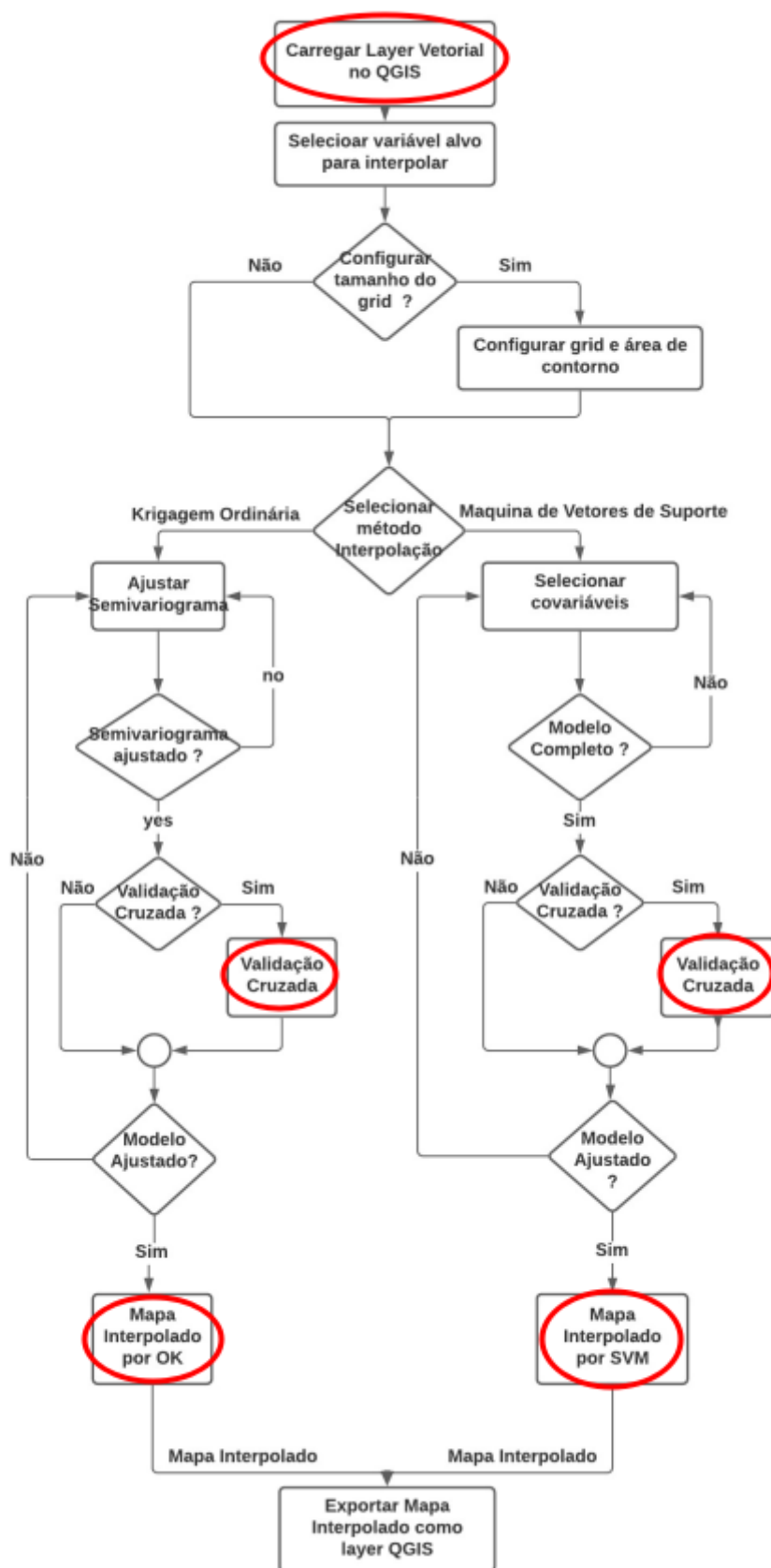
3. METODOLOGIA

A metodologia aplicada neste projeto envolveu uma análise detalhada das funcionalidades existentes no *plugin Smart-Map*, seguida pela identificação dos pontos críticos que poderiam ser otimizados. Com base nesse estudo, foram feitas alterações estratégicas que buscaram não apenas corrigir potenciais limitações, mas também aprimorar a capacidade da ferramenta em processar e carregar grandes volumes de dados.

Esta seção será dividida em subseções que abordam os principais aspectos do processo metodológico, incluindo a descrição das funcionalidades revisadas e a justificativa para cada uma das melhorias realizadas.

Esses ajustes foram projetados para oferecer uma solução mais robusta e eficaz para os usuários do *Smart-Map*, a fim de facilitar o uso do *app* em diferentes contextos do mapeamento digital de atributos do solo, sem comprometer a precisão dos resultados.

Figura 10. Etapas ajustadas do projeto destacadas em vermelho



3.1 TECNOLOGIAS UTILIZADAS

O desenvolvimento das otimizações do *plugin Smart-Map* contou com a utilização de linguagens e bibliotecas, como:

- **Python:** Linguagem principal para a implementação das melhorias do sistema.
- **Scikit-learn:** Biblioteca em Python amplamente utilizada para lidar com desenvolvimento e criação de processos que envolvam *Machine Learning*.
- **Pandas:** Biblioteca em Python amplamente utilizada e otimizada para lidar com leitura e processos que envolvam dados vetoriais.

3.2 AMBIENTE DE DESENVOLVIMENTO

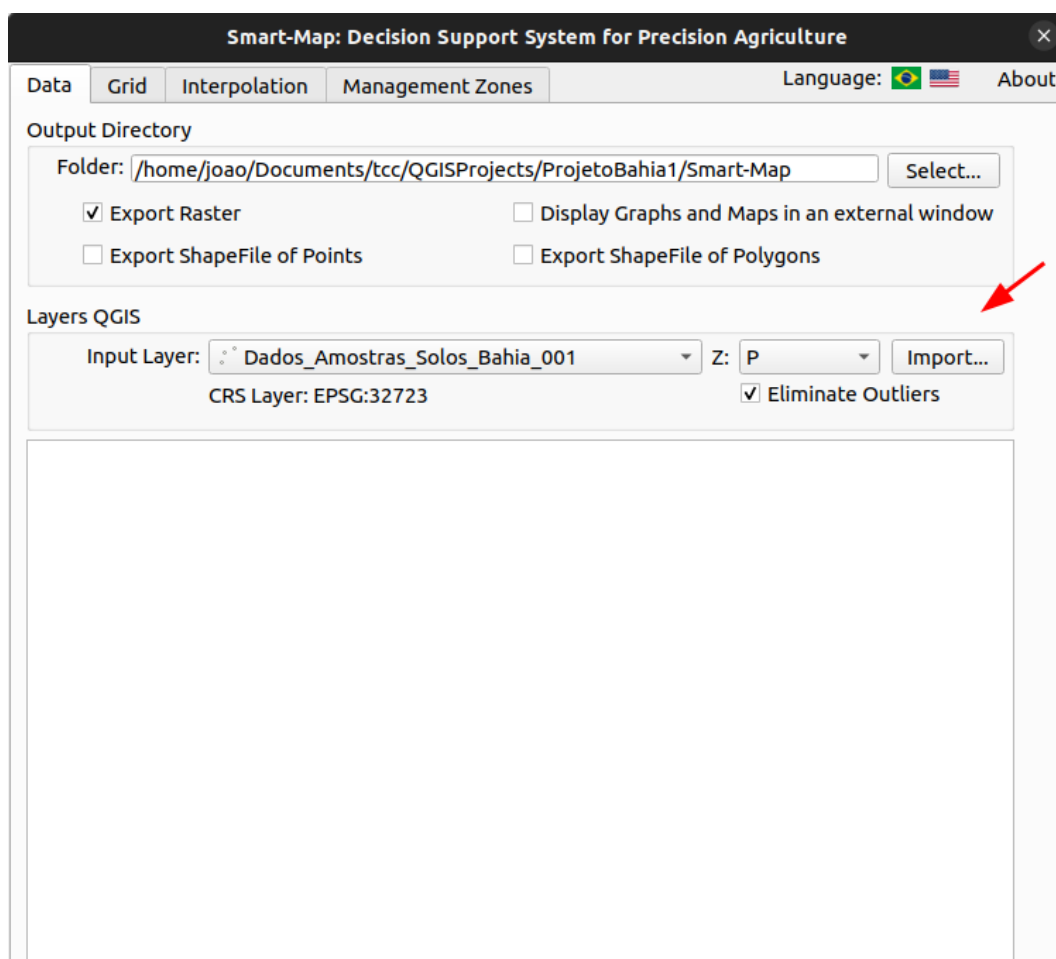
O desenvolvimento das otimizações do *plugin* consistiu no uso da IDE Visual Studio Code. O código foi desenvolvido em um computador Acer Aspire A515-57, equipado com um processador de 12th Gen Intel® Core™ i5-12450H x 12, 20GB de RAM e um SSD de 500GB. O sistema operacional utilizado foi o Linux, especificamente na distribuição Ubuntu na versão 22.0.4.

3.3 PROCESSOS

A. IMPORTAÇÃO DOS DADOS

O processo de importação dos dados inicia-se com a escolha da variável alvo para análise e a seleção dos pontos da propriedade coletados para a geração do mapa interpolado futuramente. Como observado na imagem abaixo.

Figura 11. Interface de importação dos dados



Fonte: elaborado pelo autor (2024)

Quando o usuário clicar em *import* ou importar será inicializada uma sequência de passos que irá processar os dados, e posteriormente, apresentá-los em formato de tabela.

Figura 12. Visualização dos dados selecionados

Smart-Map: Decision Support System for Precision Agriculture

Language: About

Data | Grid | Interpolation | Management Zones

Output Directory

Folder:

☒ Export Raster ☐ Display Graphs and Maps in an external window

☐ Export ShapeFile of Points ☐ Export ShapeFile of Polygons

Layers QGIS

Input Layer: Z:

CRS Layer: EPSG:32723 ☒ Eliminate Outliers

	ID	Coord X	Coord Y	P
1	1	445397.730	8627930.190	21.600
2	2	445302.130	8627900.710	20.600
3	3	445206.650	8627871.220	13.200
4	4	445236.070	8627775.620	9.000
5	5	445331.660	8627805.110	10.200
6	6	445427.150	8627834.590	12.800
7	7	445522.740	8627864.080	11.100
8	8	445618.340	8627893.560	12.300
9	9	445713.830	8627923.050	17.100
10	10	445809.430	8627952.530	14.900
11	11	445838.840	8627856.930	17.300
12	12	445743.360	8627827.440	6.500

Fonte: elaborado pelo autor (2024)

Nessa fase, já se identifica um gargalo significativo: os dados precisam ser lidos e organizados de forma vetorial em um *grid* (matriz x,y), uma vez que cada ponto da propriedade ocupa uma posição específica nos eixos x e y do espaço. Caso o número de amostras ultrapasse 5000, o sistema aciona uma funcionalidade de reamostragem para limitar os elementos a essa quantidade máxima. Essa medida visa conter o travamento e o longo tempo de carregamento da interface, o que evidencia a limitação do processo e a necessidade de otimização.

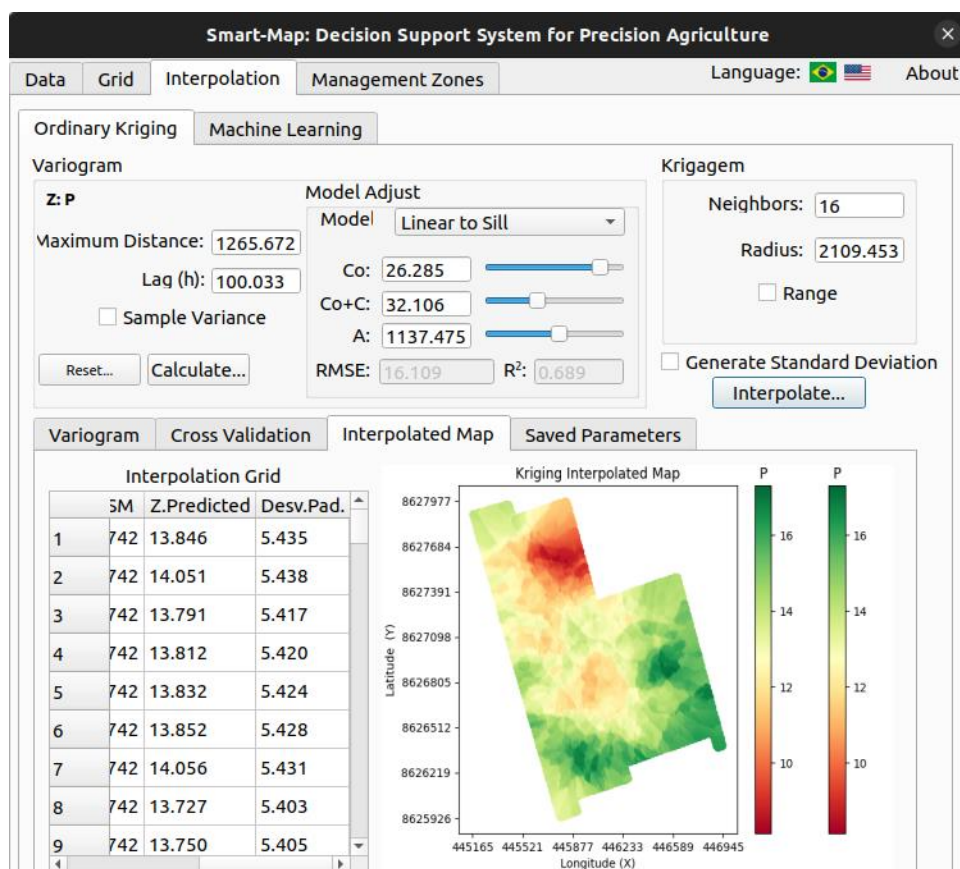
B. INTERPOLAÇÃO

Após o carregamento e importação dos dados, o próximo passo é a geração do mapa interpolado, que pode ser realizada utilizando o método da Krigagem

Ordinária ou o algoritmo de *Machine Learning Support Vector Machines* (SVM). No entanto, um dos principais obstáculos nesta etapa é o tempo de validação de ambas as técnicas, especialmente durante o processo de *Cross-Validation*. Essa lentidão é acentuada pela grande quantidade de dados, pela variação da validação cruzada adotada e o número de *folds* configurados para a criação dos mapas interpolados, o que impacta diretamente o tempo de execução dessas operações.

A técnica de validação cruzada aplicada foi a *Leave-One-Out (LOOCV)*, uma das mais onerosas em termos computacionais, conforme discutido anteriormente. Esse alto custo computacional, aliado à elevada quantidade de dados, ressalta a necessidade de otimização do processo para melhorar a eficiência e reduzir o tempo total de processamento.

Figura 13. Interface para validação dos algoritmos e geração dos mapas



Fonte: elaborado pelo autor (2024)

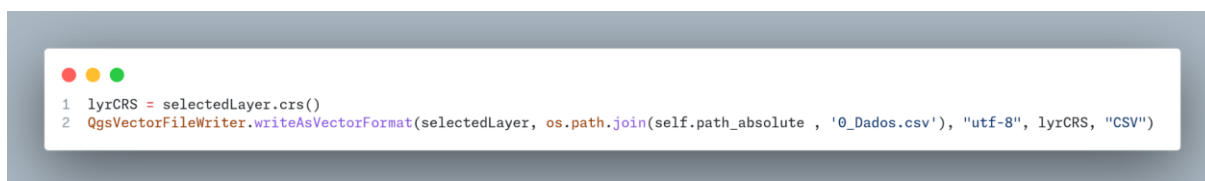
4. RESULTADOS

Nesta seção, serão discutidos os efeitos das melhorias implementadas no *plugin Smart-Map*, com foco nos ajustes realizados para otimizar o processamento de dados e ampliar a capacidade geral do sistema. Embora não haja muitos dados quantitativos específicos para medir a diferença de performance, as mudanças foram direcionadas para abordar problemas críticos identificados, como a lentidão no carregamento e processamento de grandes volumes de dados, além do alto custo computacional durante a validação cruzada.

Os resultados serão apresentados por meio de exemplos de código que ilustram as soluções antes e depois das otimizações, destacando como essas modificações visam aprimorar o desempenho do *app*. Mesmo sem uma quantificação exata, as melhorias refletem a aplicação de práticas reconhecidas, que por meio de testes usando a versão antiga e a nova com os códigos alterados comprovou-se a diferença na eficiência do sistema.

4.1 OTIMIZAÇÃO NO CARREGAMENTO DOS DADOS

Figura 14. Versão utilizando o método nativo de leitura de dados vetoriais do QGIS



Fonte: elaborado pelo autor (2024)

Figura 15. Versão utilizando o método da biblioteca Pandas

```
1
2 fields = selectedLayer.fields()
3 data = []
4
5 for feature in selectedLayer.getFeatures():
6     feature_data = {}
7     for field in fields:
8         feature_data[field.name()] = feature[field.name()]
9     data.append(feature_data)
10
11 df = pd.DataFrame(data)
12
13 df.to_parquet(os.path.join(self.path_absolute, '0_Dados.parquet'), compression='gzip', index=None)
```

Fonte: elaborado pelo autor (2024)

A implementação original que utilizava a função ***QgsVectorFileWriter.writeAsVectorFormat*** do QGIS foi substituída por uma abordagem que utiliza a biblioteca *Pandas*, o que resultou num desempenho significativamente mais rápido, devido a troca do tipo de arquivo trabalhado, antes *.csv* depois *.parquet* que possui métodos de compressão eficientes que contribuem para uma maior velocidade de processamento, além das otimizações internas do *Pandas* para manipulação de dados. Esses fatores combinados tornam o processo de leitura e escrita de dados mais eficiente, especialmente com grandes volumes de dados, mostrando que, mesmo com uma implementação ligeiramente mais complexa, a eficiência desse processo foi aperfeiçoada.

Além dessas alterações, foi ajustado o método de injeção de dados na tabela nativa do QGIS. Inicialmente, todos os registros eram processados de uma vez para o sistema de *grid* do software de informação geográfica, o que gerava problemas de desempenho quando grandes quantidades de dados eram manipulados nesse processo. Com as modificações, passou-se a utilizar paginação para gerenciar essa operação de forma mais eficiente, permitindo um carregamento mais fluido e evitando ao máximo sobrecargas no uso de memória e potencial travamento da interface.

Figura 16. Processo original de injeção dos dados na tabela nativa do QGIS

```
1 cont = 1
2 try:
3     for i in range(len(df1.index)):          #linhas
4         for j in range(len(df1.columns)):    #colunas
5             valor = df1.iloc[i,j]
6
7             if j == 0: #Coluna[0] -> ID
8                 valor = '%.0f' % valor
9             else:
10                 if valor.dtype == "float64":
11                     valor = '%.3f' % valor
12
13                 valor = QTableWidgetItem(str(valor))
14
15                 if i in self.list_index_outlier:
16                     valor.setForeground(QBrush(QColor(255, 0, 0)))
17
18
19                 self.dlg.datatable_atributos.setItem(i,j, valor)
20                 cont = cont + 1
21                 progress.setValue(cont)
22                 if progress.wasCanceled():
23                     progress.close()
24                 return
25                 #break
```

Fonte: elaborado pelo autor (2024)

Figura 17. Uso de paginação para gerenciar a injeção de dados na tabela do QGIS

```

1  try:
2
3      page_size = 5000
4      total_rows = len(df1)
5      num_pages = -(-total_rows // page_size)
6
7      num_cols = df1.shape[1]
8      self.dlg.datatable_atributos.setColumnCount(num_cols)
9
10     for page in range(num_pages):
11
12         start_index = page * page_size
13         end_index = min((page + 1) * page_size, total_rows)
14
15         page_data = df1.iloc[start_index:end_index].values
16         num_rows = page_data.shape[0]
17
18         self.dlg.datatable_atributos.setRowCount(start_index + num_rows)
19
20         for row_index, row in enumerate(page_data):
21             for col_index, value in enumerate(row):
22                 if col_index == 0:
23                     formatted_value = f'{value:.0f}'
24                 elif isinstance(value, float):
25                     formatted_value = f'{value:.3f}'
26                 else:
27                     formatted_value = str(value)
28
29                 item = QTableWidgetItem(formatted_value)
30                 if start_index + row_index in self.list_index_outlier:
31                     item.setForeground(QBrush(QColor(255, 0, 0)))
32
33                 self.dlg.datatable_atributos.setItem(start_index + row_index, col_index, item)
34
35
36         progress.setMaximum(total_rows)
37         progress.setValue(end_index)
38         if page % 10 == 0:
39             QApplication.processEvents()

```

Fonte: elaborado pelo autor (2024)

4.2 ALTERAÇÃO DO MÉTODO DE *CROSS-VALIDATION* UTILIZADA

Devido ao elevado custo computacional causado pelo método de validação cruzada utilizada originalmente, o *Leave-One-Out*, foi implementada a técnica do *K-Fold Cross-Validation* tanto no processo da OK quanto SVM. Essa alteração resultou em uma significativa redução de tempo de execução, haja vista a possibilidade de um controle muito maior do número de treinamentos realizados utilizando o número de *k* ideal para cada situação, possibilitando uma utilização de memória menor do que com o LOOCV.

Figura 18. Variação original utilizando o LOOCV na Krigagem Ordinária

```

1      for cont in (range(len(self.xy))):
2
3
4          coordx = self.xy.iloc[cont][0]      #concatena o único ponto (x), que será utilizado na validação cruzada
5          coordy = self.xy.iloc[cont][1]      #concatena o único ponto (y), que será utilizado na validação cruzada
6
7
8          xy2 = self.xy.drop(cont)            #deleta a linha cont do dataframe xy
9          z2 = self.z.drop(cont)              #deleta a linha cont da Series z
10
11         OK = kriging.OrdinaryKriging(xy2, z2, variogram_model=Model, variogram_parameters = var_params)
12
13
14         #busca de vizinhança será feita por Raio de Busca -> se n° de pontos exceder o número de vizinhos permitido,
15         #será utilizado o n° máximo de vizinhos permitido.
16         coordxy = [[coordx, coordy]]
17
18         z_est_py, ss = OK.execute(coordxy, n_closest_points=n_neig, radius=raio_busca)
19
20
21         if cont == 0:
22             labels_OK_CV = np.copy(z_est_py[0])
23         else:
24             labels_OK_CV = np.vstack((labels_OK_CV, z_est_py[0]))      #concatena após ultima linha.
25
26
27         progress.setValue(cont)
28         if progress.wasCanceled():
29             progress.close()
30         return

```

Fonte: elaborado pelo autor (2024)

Figura 19. Variação implementada utilizando o *K-Fold* na Krigagem Ordinária

```

1  splits = len(self.xy)
2  if splits > 500:
3      splits = 2
4
5  kf = KFold(n_splits=splits, shuffle=True, random_state = 42)
6
7  labels_OK_CV = np.zeros(len(self.xy))
8
9  for train_index, test_index in kf.split(self.xy):
10     xy_train, xy_test = self.xy.iloc[train_index], self.xy.iloc[test_index]
11     z_train, z_test = self.z.iloc[train_index], self.z.iloc[test_index]
12
13     OK = kriging.OrdinaryKriging(xy_train, z_train, variogram_model=Model, variogram_parameters=var_params)
14
15     coordxy = [[xy_test.iloc[0][0], xy_test.iloc[0][1]]]
16     z_est_py, ss = OK.execute(coordxy, n_closest_points=n_neig, radius=raio_busca)
17
18     labels_OK_CV[test_index] = z_est_py[0]

```

Fonte: elaborado pelo autor (2024)

Figura 20. Variação original utilizando o LOOCV no *Support Vector Machines*

```

1  for cont in (range(len(features))):
2
3
4      train_features = np.copy(features)
5      train_labels = np.copy(labels)
6
7      test_features = train_features[cont:cont+1,:] #copia a linha cont da matriz train_features(features)
8
9      train_features = np.delete(train_features, (cont), axis=0) #deleta a linha cont da matriz - train_features
10     train_labels = np.delete(train_labels, (cont), axis=0) #deleta a linha cont da matriz - train_features
11
12
13     self.norm = self.norm.fit(train_features)
14     #self.norm = self.norm.fit(test_features)
15
16     train_features = self.norm.transform(train_features)
17
18     test_features = self.norm.transform(test_features)
19
20     self.svr.fit(train_features, train_labels)
21
22     predictions = self.svr.predict(test_features)
23
24     if cont == 0: #inicia a matriz de covariaveis p
25         labels_SVM_CV = np.copy(predictions)
26     else:
27         labels_SVM_CV = np.vstack((labels_SVM_CV, predictions)) #concatena após ultima linha.
28
29
30     progress.setValue(cont+2)
31     if progress.wasCanceled():
32         progress.close()
33         return

```

Fonte: elaborado pelo autor (2024)

Figura 21. Variação implementada utilizando o *K-Fold* no *Support Vector Machines*

```

1  self.svr = svm.SVR(kernel = 'rbf', C = C_average, gamma = gamma_average)
2
3
4  #####
5  #Validação Cruzada
6
7  features = self.norm.fit_transform(features)
8
9  kf = KFold(n_splits=3, shuffle=True, random_state=42)
10
11  predictions = cross_val_predict(self.svr, features, labels, cv=kf, n_jobs=-1)
12
13  data_CV_SVM = np.column_stack((features[:, 0], features[:, 1], labels, predictions))

```

Fonte: elaborado pelo autor (2024)

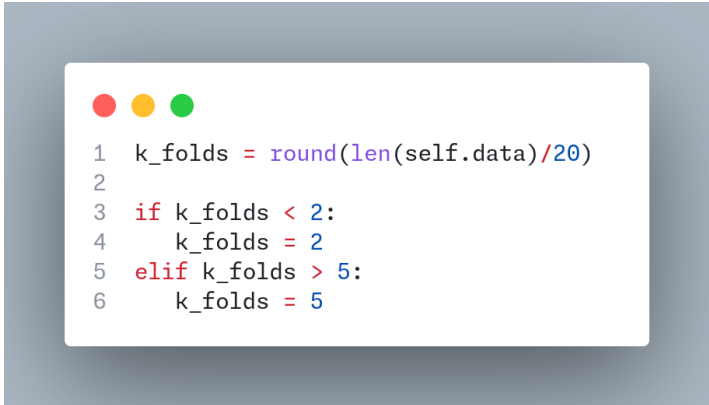
Nesta técnica, destaca-se o uso da função ***cross_val_predict*** que possui um parâmetro chamado *n_jobs* que é responsável por acionar todos os processadores disponíveis no sistema para realizar a operação quando seu valor é igual a -1, o que

contribui significativamente para o desempenho dos processos internos realizados por essa funcionalidade.

4.3 OTIMIZAÇÃO NA CRIAÇÃO DO MAPA INTERPOLADO

Para mitigar a lentidão no tempo de execução durante o processo de criação dos mapas interpolados, uma alternativa encontrada foi simplesmente limitar ainda mais o número de *folds* que estavam sendo usados na operação. Antes da geração dos mapas em si, é preciso encontrar os melhores parâmetros para os modelos com auxílio do *Grid Search* - uma forma de encontrar a combinação ideal de hiper parâmetros para um algoritmo de *Machine Learning*. Porém, essa abordagem é computacionalmente inviável devido ao grande número de tentativas e combinações possíveis geradas.

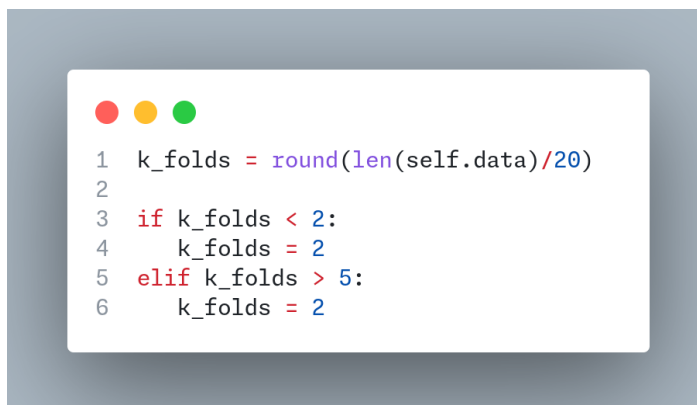
Figura 22. Lógica e número de *folds* usados originalmente



```
1 k_folds = round(len(self.data)/20)
2
3 if k_folds < 2:
4     k_folds = 2
5 elif k_folds > 5:
6     k_folds = 5
```

Fonte: elaborado pelo autor (2024)

Figura 23. Lógica e número de *folds* alterados



```
1 k_folds = round(len(self.data)/20)
2
3 if k_folds < 2:
4     k_folds = 2
5 elif k_folds > 5:
6     k_folds = 2
```

Fonte: elaborado pelo autor (2024)

5. CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo implementar melhorias de performance em funcionalidades críticas no plugin Smart-Map, focando na otimização do processamento dos dados e na redução dos tempos de carregamento. A adoção da biblioteca Pandas para orquestrar processos que envolvem dados vetoriais, a implementação do conceito de paginação do projeto, a alteração da variação usada de validação cruzada e a limitação de folds utilizadas na procura pelos melhores parâmetros do algoritmo mostraram-se eficazes para aumentar a eficiência do sistema.

Um avanço notável que realmente evidencia que as alterações surtiram grandes efeitos positivos nas operações internas da aplicação foi a extensão do limite de *resampling* de pontos citada anteriormente. Inicialmente, esse limite era de 5.000 registros, agora, inclusive em computadores com menor capacidade de processamento, acredita-se que possa ser estendido para aproximadamente 18.000 registros. Contudo, em testes realizados na máquina no qual foram desenvolvidas as soluções apresentadas, o máximo atingido foi de 50.000 pontos, porém comprometendo a performance do sistema.

Apesar dos avanços, ainda há desafios, como a necessidade de equilibrar precisão e custo computacional. As otimizações feitas contribuem para tornar o Smart-

Map uma ferramenta mais eficiente e robusta, com potencial para futuras melhorias e desenvolvimentos.

REFERÊNCIAS BIBLIOGRÁFICAS

ANALYTICS VIDHYA. *Support Vector Machines (SVM) – A Complete Guide for Beginners*. 2021. Disponível em: <https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/>. Acesso em: 22 ago. 2024.

ESCOVEDO, T.; KOSHIYAMA, A. S. *Introdução a Data Science — Algoritmos de Machine Learning e métodos de análise*. São Paulo: Ed. Casa do Código, 2020.

ESCOVEDO, T. *Machine Learning: conceitos e modelos*. Medium, 2020. Disponível em: <https://tatianaesc.medium.com/machine-learning-conceitos-e-modelos-f0373bf4f445>. Acesso em: 22 ago. 2024.

FARAMARZI, N. *Support Vector Regressor: Theory and Coding Exercise in Python*. Medium, 2020. Disponível em: <https://medium.com/@niousha.rf/support-vector-regressor-theory-and-coding-exercise-in-python-ca6a7dfa927>. Acesso em: 22 ago. 2024.

MEDIUM. *The Kernel Trick*. Medium, 2020. Disponível em: <https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f>. Acesso em: 22 ago. 2024.

ORACLE. *What is Machine Learning?*. Disponível em: <https://www.oracle.com/br/artificial-intelligence/machine-learning/what-is-machine-learning/>. Acesso em: 22 ago. 2024.

PATEL, B. *Resampling Methods*. Bijen Patel, 2022. Disponível em: <https://www.bijenpatel.com/guide/islr/resampling-methods/>. Acesso em: 22 ago. 2024.

PEREIRA, G. W. Ferramentas computacionais para suporte a decisão no mapeamento de atributos do solo. 2021. Tese – Universidade Federal de Viçosa, Viçosa, 2021. Disponível em: <https://locus.ufv.br/handle/123456789/29872>. Acesso em: 22 ago.2024.

REMÍGIO, M. *Máquinas de vetores de suporte (SVM)*. Medium, 2020. Disponível em: <https://medium.com/@msremigio/m%C3%A1quinas-de-vetores-de-suporte-svm-77bb114d02fc>. Acesso em: 22 ago. 2024.

SCIKIT-LEARN. *Cross-Validation*. Disponível em: https://scikit-learn.org/stable/modules/cross_validation.html. Acesso em: 22 ago. 2024.

STACKABUSE. *Understanding SVM Hyperparameters*. 2021. Disponível em: <https://stackabuse.com/understanding-svm-hyperparameters/>. Acesso em: 22 ago. 2024.

TEAMCODE. *A Step-by-Step Guide for Pagination in Python*. Medium, 2023. Disponível em: <https://medium.com/@teamcode20233/a-step-by-step-guide-for-pagination-in-python-f7da5f06767d>. Acesso em: 22 ago. 2024.

WIKIPEDIA. *Cross-validation (statistics)*. 2024. Disponível em: [https://en.wikipedia.org/wiki/Cross-validation_\(statistics\)](https://en.wikipedia.org/wiki/Cross-validation_(statistics)). Acesso em: 22 ago. 2024.

ZHU, X. *What is the Kernel Trick? Why is it Important?*. Medium, 2021. Disponível em: <https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d>. Acesso em: 22 ago. 2024.