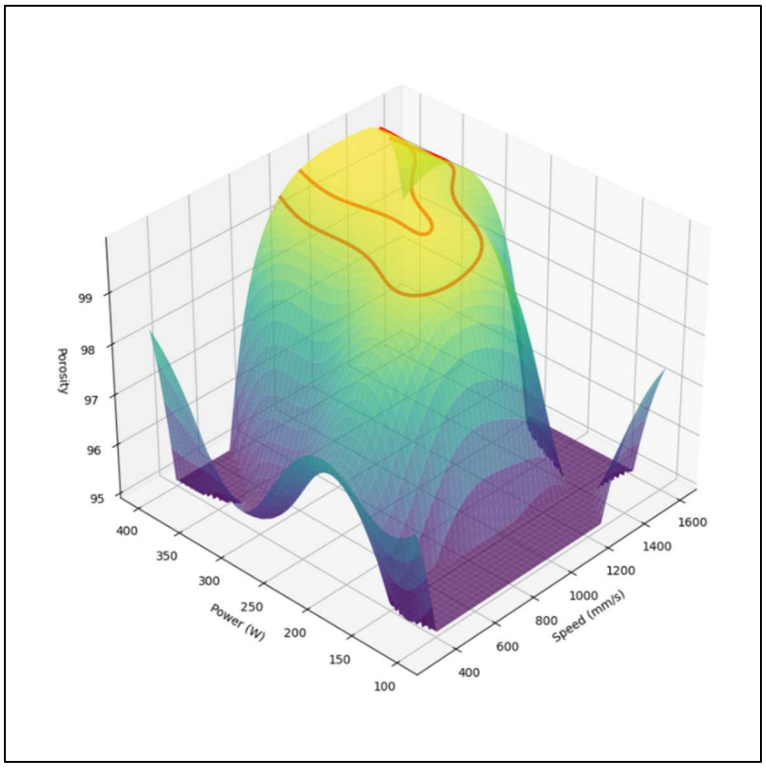
Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

PROCESS OPTIMIZATION IN ADVANCE MANUFACTURING



Estudiante: Alonso Cuende, Adrián

Director/Directora: Herrero Villalibre, Saioa

Curso: 2023-2024

Fecha: Bilbao, 15, octubre, 2024

1. Summary

This research project explores the application of machine learning to optimize the Selective Laser Melting (SLM) process for the production of titanium alloy (Ti6Al4V) components, with a specific focus on minimizing porosity to improve part quality. Porosity is a critical factor affecting the mechanical properties and reliability of parts manufactured via additive manufacturing (AM) methods such as SLM. The traditional approach to optimizing SLM parameters—typically reliant on trial-and-error experimentation—can be time-consuming, costly, and inefficient. Therefore, this research proposes a data-driven solution using machine learning to streamline the optimization process and enhance manufacturing outcomes.

Three machine learning models—Linear Regression, Random Forest Regression, and Gaussian Process Regression (GPR)—were selected for this research based on their strengths and suitability for different aspects of the SLM process. Linear Regression served as a baseline model, offering initial insights into how process parameters affect porosity. Random Forest Regression provided a more advanced approach, capturing non-linear interactions between parameters and identifying which factors most significantly influenced porosity. Gaussian Process Regression (GPR) emerged as the most effective model due to its flexibility and ability to provide uncertainty estimates, making it particularly valuable for applications with complex, non-linear relationships and limited data availability.

A notable innovation in this project was the use of transfer learning, which enabled the integration of data from multiple sources to improve model accuracy and generalizability. This approach effectively addressed data scarcity issues, a common challenge in additive manufacturing research, by allowing the models to leverage insights from a broader range of experimental conditions.

Este proyecto de investigación explora la aplicación del aprendizaje automático para optimizar el proceso de fusión selectiva por láser (SLM, por sus siglas en inglés) en la producción de componentes de aleación de titanio (Ti6Al4V), con un enfoque específico en minimizar la porosidad para mejorar la calidad de las piezas. La porosidad es un factor crítico que afecta las propiedades mecánicas y la fiabilidad de las piezas fabricadas mediante métodos de manufactura aditiva (AM) como el SLM. El enfoque tradicional para optimizar los parámetros de SLM —que generalmente depende de la experimentación por prueba y error— puede resultar lento, costoso e ineficiente. Por lo tanto, esta investigación propone una solución basada en datos utilizando aprendizaje automático para agilizar el proceso de optimización y mejorar los resultados de fabricación.

Tres modelos de aprendizaje automático —Regresión Lineal, Regresión de Bosques Aleatorios y Regresión de Procesos Gaussianos (GPR)— fueron seleccionados para esta investigación, en función de sus fortalezas y adecuación para diferentes aspectos del proceso SLM. La Regresión Lineal sirvió como modelo base, ofreciendo una comprensión inicial de cómo los parámetros del proceso afectan la porosidad. La Regresión de Bosques Aleatorios proporcionó un enfoque más avanzado, capturando interacciones no lineales entre parámetros e identificando los factores que influyen más significativamente en la porosidad. La Regresión de Procesos Gaussianos (GPR) resultó ser el modelo más efectivo debido a su flexibilidad y capacidad para proporcionar estimaciones de incertidumbre.

Una innovación destacada en este proyecto fue el uso del aprendizaje por transferencia, que permitió la integración de datos de múltiples fuentes para mejorar la precisión y la generalización del modelo. Este enfoque abordó eficazmente los problemas de escasez de datos, un desafío común en la investigación de manufactura aditiva, al permitir que los modelos aprovecharan conocimientos de una gama más amplia de condiciones experimentales.

Ikerketa-proiektu honek ikaskuntza automatikoaren aplikazioa aztertzen du, laser bidezko fusio selektiboko prozesua (SLM, ingelesezko sigletan) titanio-aleazioko osagaien ekoizpenean optimizatzeko (Ti6Al4V), piezen kalitatea hobetzeko porositatea minimizatzeko ikuspegi espezifiko batekin. Porositatea faktore kritikoa da, eta SLM bezalako manufaktura gehigarriko metodoen bidez fabrikatutako piezen propietate mekanikoei eta fidagarritasunari eragiten die. SLM parametroak optimizatzeko ikuspegi tradizionala — normalean proba eta errore bidezko esperimentazioaren araberakoa — motela, garestia eta ez-eraginkorra izan daiteke. Horrenbestez, ikerketa honek datuetan oinarritutako soluzio bat proposatzen du, ikaskuntza automatikoa erabiliz optimizazio-prozesua arintzeko eta fabrikazio-emaitzak hobetzeko.

Ikasketa automatikoko hiru eredu — Erregresio Lineala, Baso Aleatorioen Erregresioa eta Prozesu Gaussiarren Erregresioa (GPR) — aukeratu ziren ikerketa honetarako, beren indarguneen eta SLM prozesuaren hainbat alderditarako egokitzapenaren arabera. Erregresio linealak oinarrizko eredu gisa balio izan zuen, prozesuaren parametroek porositateari nola eragiten dioten hasierako ulermena eskainiz. Ausazko basoen erregresioak ikuspegi aurreratuagoa eman zuen, parametroen arteko interakzio ez-linealak atzemanez eta porositatean eragin handiena duten faktoreak identifikatuz. Prozesu Gaussiarren Erregresioa (GPR) eredu eraginkorrena izan zen malgutasunagatik eta ziurgabetasun-estimazioak emateko gaitasunagatik.

Proiektu honetan nabarmendu zen berrikuntza bat transferentzia bidezko ikaskuntzaren erabilera izan zen. Horri esker, iturri askotako datuak integratu ahal izan ziren, ereduaren zehaztasuna eta orokortzea hobetzeko. Ikuspegi horrek modu eraginkorrean heldu zien datu-eskasiari, manufaktura gehigarriaren ikerketan erronka komuna baitzen, ereduek baldintza esperimentalen sorta zabalago baten ezagutzak aprobetxatzea ahalbidetuz.

2. Index

3. List of tables, illustrations, graphs and acronyms

4. MEMORY

4.1. Introduction

Additive Manufacturing (AM), often referred to as 3D printing, has revolutionized various industries by enabling the production of complex parts with high precision and minimal material waste. Unlike traditional manufacturing methods that rely on subtractive processes, AM builds parts layer by layer, offering unparalleled design flexibility. This flexibility has led to its widespread adoption in industries such as aerospace, automotive, and biomedical, where lightweight, custom-designed parts are in high demand. Among the various AM techniques, Selective Laser Melting (SLM) has emerged as a particularly effective method for producing high-strength metal parts, especially those made from titanium alloys such as Ti6Al4V.

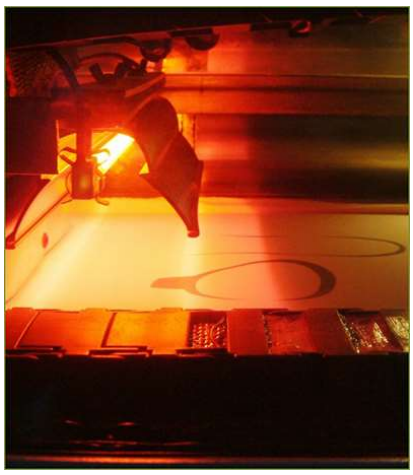


Figure 1: Laser Bed Powder Fusion

In the aerospace industry, for instance, the demand for high-performance parts that can withstand extreme conditions has led to the extensive use of SLM. Components such as turbine blades, aircraft brackets, and structural elements benefit from the precision and material efficiency that SLM provides. However, despite its advantages, SLM and other AM processes face several challenges, particularly in terms of achieving optimal part quality.

Figure 2: Manufactured parts

One of the most significant challenges in additive manufacturing is the control of porosity, which directly impacts the mechanical properties of the manufactured parts. High porosity levels can reduce the strength, durability, and overall performance of a part, rendering it unsuitable for critical applications. In industries like aerospace, where component failure is not an option, achieving low porosity levels—ideally less than 0.5%—is crucial. However, controlling porosity in the AM process is not straightforward due to the multitude of parameters that influence it, such as laser power, scan speed, hatch spacing, and layer thickness. The complexity of these interactions makes manual tuning of process parameters highly inefficient, time-consuming, and costly.

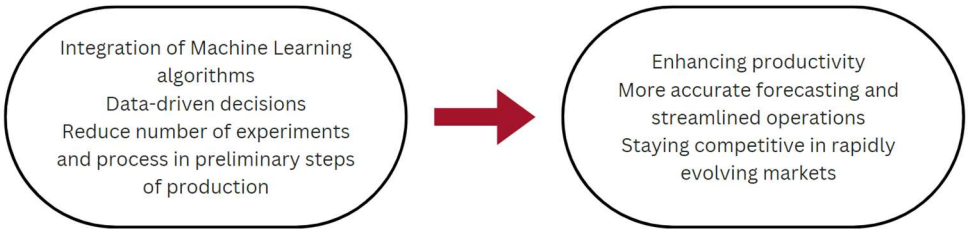


Figure 3: Benefits of machine learning

This is where machine learning (ML) comes into play. Machine learning offers a powerful toolset for optimizing complex processes like SLM by automatically identifying patterns in the data and adjusting parameters to achieve desired outcomes. By leveraging historical data and predictive models, machine learning can help manufacturers optimize process parameters without the need for extensive trial-and-error experimentation. This leads to improved part quality, reduced production costs, and faster development times.

In recent years, machine learning has gained significant traction in the field of additive manufacturing. Techniques such as Bayesian optimization, Random Forest Regression, and Gaussian Process Regression (GPR) have been applied to optimize various aspects of the AM process, including print speed, energy density, and material properties. These models are particularly effective in handling small datasets, which are common in high-cost manufacturing environments where large-scale experimentation is impractical.

In this research project, the focus is on optimizing three critical parameters in the SLM process: laser power, scan speed, and hatch spacing. The goal is to reduce porosity in Ti6Al4V alloy parts while maximizing the efficiency of the manufacturing process. To achieve this, a combination of machine learning models—Linear Regression, Random Forest Regression, and Gaussian Process Regression—are applied to experimental data collected from both internal and external sources. Additionally, Bayesian optimization techniques are used to fine-tune the models and identify the optimal printing parameters.

This project not only aims to demonstrate the effectiveness of machine learning in optimizing additive manufacturing processes but also highlights the challenges and limitations of these techniques, particularly in handling complex, multi-parameter systems. Through this research, we aim to contribute to the growing body of knowledge on the use of machine learning in advanced manufacturing and provide a framework for future studies in this field.

4.2. Context

4.2.1. Overview of Additive Manufacturing and Its Significance

Additive Manufacturing (AM), commonly known as 3D printing, has significantly disrupted conventional manufacturing methods, offering unprecedented design freedom, cost-effectiveness, and material efficiency. By fabricating parts layer by layer,

AM eliminates the constraints of traditional subtractive processes, allowing for complex geometries and intricate internal structures that would be impossible or extremely costly to achieve through conventional means. This capability has made AM particularly valuable in industries such as aerospace, automotive, biomedical, and defense, where components often require lightweight structures, custom designs, and material optimization.

Among various AM technologies, Selective Laser Melting (SLM) stands out as a preferred method for producing high-strength metal parts. SLM utilizes a high-energy laser to fuse powdered metal particles layer by layer, enabling the production of fully dense, mechanically robust components. This technique is highly suitable for manufacturing parts made from titanium alloys (e.g., Ti6Al4V), which are widely used in aerospace applications due to their excellent strength-to-weight ratio, corrosion resistance, and high-temperature performance. The ability to produce such components with minimal waste makes SLM a highly attractive option, particularly for industries seeking to reduce costs and enhance sustainability.

However, despite its advantages, the SLM process is not without challenges. One of the most significant issues is achieving consistent part quality, particularly in terms of porosity. Porosity, which refers to the presence of voids or pores within the material, can significantly affect the mechanical properties of a part, reducing its strength, fatigue life, and overall performance. In critical applications, such as aerospace or medical implants, high levels of porosity are unacceptable, making it essential to optimize the SLM process to produce parts with minimal defects.

4.2.2. Challenges in Achieving Optimal Quality in Additive Manufacturing

The quality of parts produced by SLM is influenced by a multitude of process parameters, including laser power, scan speed, hatch spacing, layer thickness, and

powder flow rate. These parameters interact in complex ways, making it difficult to predict the outcome of the process without extensive experimentation. For example, increasing the laser power can enhance the fusion of metal particles, leading to denser parts, but it can also result in overheating, causing defects such as keyhole porosity or thermal distortion. Similarly, adjusting the scan speed can affect the heat input and cooling rate, impacting the microstructure and mechanical properties of the final part.

In practice, manufacturers often rely on a trial-and-error approach to identify the optimal set of parameters for a given material and part geometry. This approach, however, is time-consuming, costly, and inefficient, particularly for high-value applications where even minor defects can result in significant losses. Moreover, the variability inherent in the SLM process makes it challenging to achieve consistent results, even when using the same set of parameters. Factors such as variations in powder quality, machine calibration, and environmental conditions can all contribute to deviations in part quality, further complicating the optimization process.

Given these challenges, there is a growing need for more systematic and data-driven approaches to optimize the SLM process. This is where machine learning offers a promising solution, enabling manufacturers to leverage existing data to predict outcomes, identify optimal parameters, and ultimately achieve higher-quality parts with fewer experiments.

4.2.3. Machine Learning: A Solution to Additive Manufacturing Challenges

Machine learning (ML) is a subset of artificial intelligence that involves developing algorithms that can learn patterns from data and make predictions or decisions without being explicitly programmed. In the context of additive manufacturing, machine learning can be used to analyze the complex relationships

between process parameters and part quality, enabling manufacturers to optimize their processes more efficiently.

Machine learning models, such as Linear Regression, Random Forest Regression, and Gaussian Process Regression (GPR), have shown great potential in predicting and optimizing outcomes in various manufacturing processes. These models can handle large datasets with multiple variables, making them well-suited for analyzing the multi-parameter nature of SLM. By training these models on historical data, manufacturers can gain insights into how different process parameters affect part quality and use this knowledge to identify the optimal conditions for producing high-quality parts.

Purposes of the Model

Method	Purpose	Approach
Compositional modelling	**Find patterns within data**	Passive
Exploration	**Learn function as quickly as possible**	Active

Figure 4: Purposes of GPR model

For example, Linear Regression can be used to identify linear relationships between process parameters and porosity, providing a simple yet effective way to understand how changes in laser power or scan speed impact part quality. Random Forest Regression, on the other hand, can capture more complex interactions between parameters by building an ensemble of decision trees, each representing a different aspect of the process. Gaussian Process Regression, a more advanced machine learning technique, offers the ability to model uncertainty and make predictions with confidence intervals, making it particularly valuable for applications where data is scarce or noisy.

The application of machine learning in additive manufacturing is not limited to parameter optimization. It can also be used for real-time process monitoring, defect detection, and predictive maintenance. For instance, by analyzing sensor data collected

during the printing process, machine learning models can detect anomalies or deviations from the desired process conditions, enabling operators to intervene and correct issues before they affect the final part quality.

4.2.4. Current State of Research in Machine Learning for Additive Manufacturing

In recent years, there has been a surge in research exploring the use of machine learning to optimize additive manufacturing processes. Studies have demonstrated the effectiveness of machine learning models in predicting key outcomes such as porosity, surface roughness, and mechanical properties based on process parameters. For example, the application of Gaussian Process Regression (GPR) in optimizing laser power and scan speed has shown promising results in reducing porosity and improving the overall quality of metal parts. Other studies have utilized Random Forest models to identify the most influential parameters in the printing process, allowing for more targeted optimization efforts.

Despite these successes, challenges remain in applying machine learning to additive manufacturing. One of the primary challenges is the limited availability of high-quality data, as generating experimental data in additive manufacturing is both time-consuming and costly. This makes it difficult to train machine learning models effectively, particularly for more complex algorithms that require large datasets to achieve accurate predictions.

Another challenge is the lack of standardized methods for integrating machine learning into the additive manufacturing workflow. While many studies have demonstrated the potential of machine learning for optimization, there is still a need for practical guidelines and frameworks that manufacturers can use to implement these techniques in real-world settings.

4.2.5. Transfer Learning: A Path Forward

Transfer learning is an emerging approach that offers a solution to the data scarcity problem in additive manufacturing. Transfer learning involves leveraging knowledge gained from one task or dataset to improve the performance of a model on a related but different task or dataset. In the context of this research, transfer learning can be used to apply insights gained from optimizing one set of printing parameters to another, potentially reducing the need for extensive experimentation.

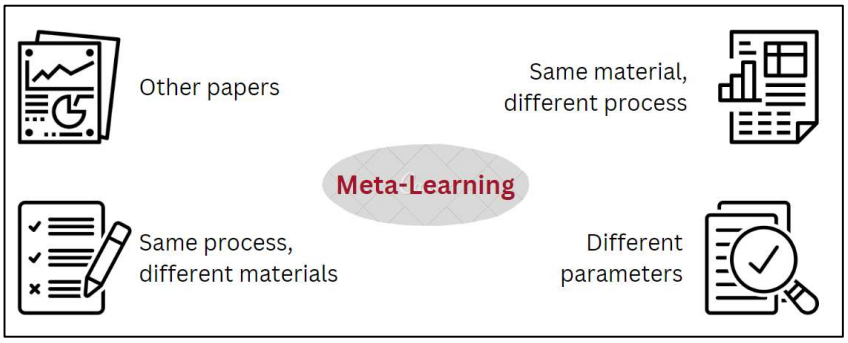


Figure 5: Meta-Learning Scheme

For example, data collected from experiments using one type of alloy can be used to inform the optimization process for a different alloy, thereby accelerating the learning process and improving model accuracy. This approach is particularly valuable in additive manufacturing, where generating new data can be prohibitively expensive.

By combining traditional machine learning techniques with transfer learning, this research project aims to develop a more efficient and effective method for optimizing the SLM process, ultimately contributing to the advancement of additive manufacturing technologies.

4.3. Objectives and scope of the work

4.3.1. Objectives of the Research

The primary objective of this research is to leverage machine learning techniques to optimize the Selective Laser Melting (SLM) process parameters for manufacturing

titanium alloy (Ti6Al4V) parts, with a specific focus on minimizing porosity and improving part quality. By applying machine learning algorithms, this study aims to develop a data-driven approach that can identify the optimal combinations of process parameters, thereby reducing the need for extensive trial-and-error experimentation.

The specific objectives of the research are as follows:

1. **Optimization of Process Parameters:**
   - Identify the optimal levels of key SLM process parameters, including laser power, scan speed, and hatch spacing, to achieve the lowest possible porosity in Ti6Al4V parts.
   - Understand the impact of these parameters on the overall energy density and how they influence the formation of defects such as pores or voids.
2. **Application of Machine Learning Models:**
   - Evaluate the performance of different machine learning models, such as Linear Regression, Random Forest Regression, and Gaussian Process Regression (GPR), in predicting porosity based on process parameters.
   - Determine which machine learning model offers the best predictive accuracy and robustness for the optimization of SLM processes.
3. **Implementation of Transfer Learning:**
   - Utilize transfer learning techniques to combine datasets from different sources (e.g., DLR and IIT datasets) to improve the predictive capabilities of the machine learning models.
   - Assess the effectiveness of transfer learning in enhancing the optimization process, especially when dealing with limited experimental data.

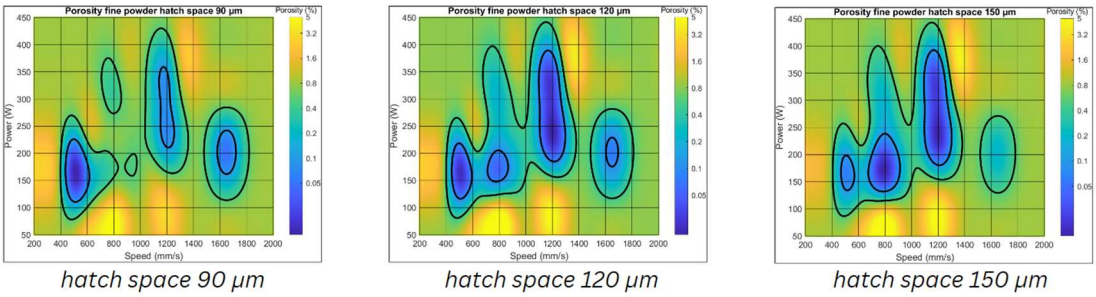


Figure 6: Result of Transfer Learning

4. **Development of a Framework for Process Optimization:**
   - Create a systematic and replicable framework that can be used by manufacturers to optimize SLM process parameters using machine learning.
   - Provide practical guidelines and recommendations for integrating machine learning into the additive manufacturing workflow, enabling real-time parameter optimization and process control.
5. **Evaluation of Results and Practical Implications:**
   - Compare the results obtained from machine learning models with experimental findings to validate their accuracy and effectiveness in optimizing the SLM process.
   - Highlight the potential benefits of implementing machine learning-driven optimization in industrial settings, such as reduced production costs, improved part quality, and shortened development times.

4.3.2. Scope of the Work

The scope of this research encompasses the application of machine learning techniques to optimize the SLM process parameters for Ti6Al4V alloy parts. The study will primarily focus on the following aspects:

1. **Data Collection and Preprocessing:**
   - The research will utilize experimental data collected from both the IIT and DLR datasets, which contain information on process parameters (laser power, scan speed, hatch spacing) and corresponding porosity measurements.
   - Preprocessing techniques will be applied to clean and standardize the data, ensuring that it is suitable for machine learning analysis.
2. **Machine Learning Model Implementation:**
   - The study will implement multiple machine learning models, including Linear Regression, Random Forest Regression, and Gaussian Process Regression (GPR), to predict porosity based on process parameters.
   - Bayesian optimization will be used to fine-tune the hyperparameters of these models, ensuring optimal performance.

3. **Transfer Learning Techniques:**
   - The research will explore transfer learning by combining insights gained from different datasets (e.g., DLR and IIT) to improve model accuracy and predictive capabilities.
   - This approach aims to demonstrate how transfer learning can address data scarcity issues in additive manufacturing and accelerate the optimization process.
4. **Comparison and Analysis of Results:**
   - The study will compare the predictions generated by machine learning models against actual experimental results, evaluating their accuracy and reliability.
   - The analysis will include the creation of contour plots, heatmaps, and other visualizations to illustrate the relationships between process parameters and porosity.
5. **Practical Recommendations and Guidelines:**
   - Based on the findings, the research will provide practical recommendations for manufacturers looking to implement machine learning techniques in their additive manufacturing processes.
   - The study will discuss the potential challenges, limitations, and future research directions for optimizing SLM processes using machine learning.

4.3.3. Research Questions

To guide the research process and ensure a focused approach, the following research questions have been formulated:

- How do laser power, scan speed, and hatch spacing affect the porosity of parts manufactured using the SLM process?
- Which machine learning model (Linear Regression, Random Forest Regression, or Gaussian Process Regression) is most effective in predicting and optimizing porosity based on process parameters?
- How can transfer learning techniques improve the predictive accuracy of machine learning models when dealing with limited experimental data?

- What are the key factors that contribute to the successful implementation of machine learning in optimizing additive manufacturing processes?
- What practical guidelines can be developed to assist manufacturers in applying machine learning for real-time process optimization in SLM?

By addressing these research questions, this study aims to provide valuable insights into the potential of machine learning as a tool for optimizing additive manufacturing processes. The findings will contribute to the broader field of advanced manufacturing and offer practical solutions for improving part quality and production efficiency.

4.4. Benefits of the work

The application of machine learning to optimize additive manufacturing processes offers numerous benefits, particularly in the context of Selective Laser Melting (SLM) for producing titanium alloy parts. One of the most significant advantages is the potential to dramatically improve part quality by reducing porosity, which is a critical factor in determining the mechanical properties and performance of the final product. In industries such as aerospace, automotive, and biomedical engineering, achieving parts with minimal porosity is essential to ensure reliability, strength, and longevity. By using machine learning to predict and optimize process parameters, manufacturers can produce parts that meet these stringent quality requirements, ultimately enhancing the safety and effectiveness of components used in high-stress applications.

Another key benefit is the reduction in time and costs associated with process optimization. Traditionally, identifying the optimal set of parameters for the SLM process involves a labor-intensive trial-and-error approach, where engineers must conduct numerous experiments to determine how different settings impact part quality. This method is not only time-consuming but also costly, as it requires the consumption of materials, machine time, and labor resources. Machine learning offers a more efficient alternative by leveraging existing data to build predictive models that can identify optimal parameters without the need for extensive experimentation. This data-driven approach significantly shortens the optimization process, enabling manufacturers to bring products to market more quickly and with lower production costs.

The implementation of machine learning also contributes to the overall sustainability of additive manufacturing. By optimizing process parameters, manufacturers can minimize material waste, energy consumption, and the number of defective parts produced during the manufacturing process. For example, using

predictive models to fine-tune laser power, scan speed, and hatch spacing ensures that only the necessary amount of energy and material is used to create each layer of the part, reducing excess consumption. This not only leads to cost savings but also aligns with environmental sustainability goals, making the manufacturing process more eco-friendly.

Furthermore, machine learning provides manufacturers with valuable insights into the underlying relationships between process parameters and part quality. This deeper understanding allows for more informed decision-making and process control, enabling manufacturers to anticipate and address potential issues before they arise. For instance, if a model indicates that a certain combination of laser power and scan speed is likely to result in high porosity, adjustments can be made in real-time to prevent defects, thereby improving the overall efficiency and consistency of the production process.

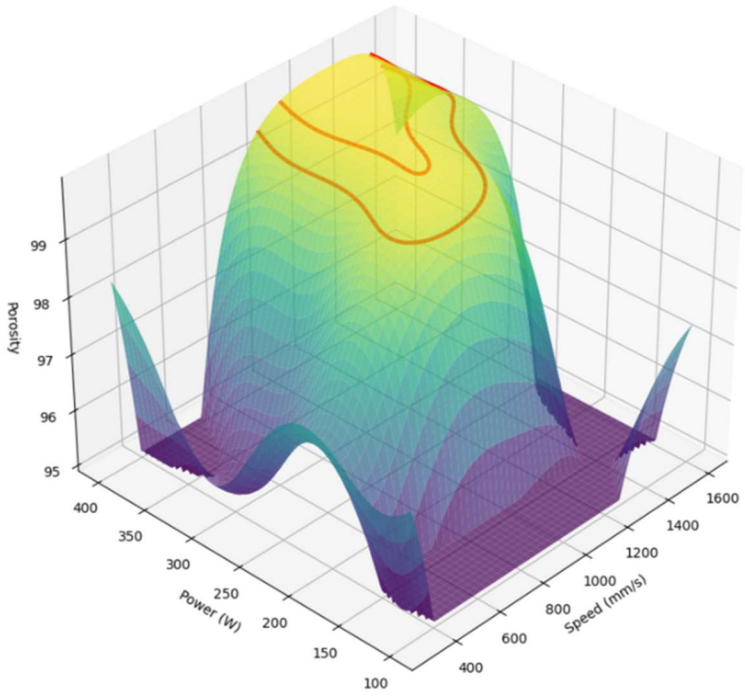


Figure 7: 3D graph of optimal areas

Another significant benefit of this research is its potential to serve as a foundation for developing more advanced optimization techniques in additive manufacturing. By demonstrating the effectiveness of machine learning models such as Gaussian Process Regression (GPR) and Random Forest Regression in predicting and optimizing process outcomes, this study opens the door for further exploration into more sophisticated methods, such as deep learning or reinforcement learning. These advanced techniques could potentially offer even greater accuracy and predictive power, further enhancing the ability to optimize complex manufacturing processes.

Finally, the integration of machine learning into the additive manufacturing workflow can lead to improved scalability and adaptability in production. As manufacturers increasingly adopt Industry 4.0 principles, which emphasize automation, data exchange, and smart manufacturing technologies, machine learning offers a critical tool for achieving flexible, adaptable, and autonomous production systems. By enabling real-time monitoring and adjustment of process parameters, machine learning allows manufacturers to adapt quickly to changing conditions, material variations, or design modifications, ensuring consistent quality across different production runs.

In summary, the benefits of applying machine learning to optimize additive manufacturing processes are multifaceted. They range from enhancing part quality and reducing production costs to promoting sustainability and providing valuable insights into the manufacturing process. These advantages not only make machine learning an invaluable tool for manufacturers seeking to remain competitive in an increasingly complex market but also position it as a key enabler for the future of advanced manufacturing.

4.5. Selection/Description of the proposed solution

The challenge of optimizing additive manufacturing parameters, particularly for the Selective Laser Melting (SLM) process of titanium alloys, necessitates a sophisticated approach due to the complex interactions that influence porosity and part quality. Traditional methods, often reliant on extensive trial-and-error experimentation, are not only inefficient but also fall short of capturing the intricate relationships between process parameters. To address this, the research adopts a machine learning-driven strategy, leveraging advanced models to predict and optimize key parameters, with the ultimate goal of minimizing porosity and enhancing part quality.

A key insight in this project is the identification of energy density parameters—specifically, Linear Energy Density (LED), Global Energy Density (GED), and Volumetric Energy Density (VED)—as critical factors in predicting porosity outcomes. These energy densities serve as comprehensive indicators of how different process variables affect the final product. For instance, LED, which represents the ratio of laser power to scan speed, provides insight into the rate of energy input into the powder bed. Similarly, GED and VED capture the effects of hatch spacing and layer thickness, respectively, on the overall energy distribution during the SLM process.

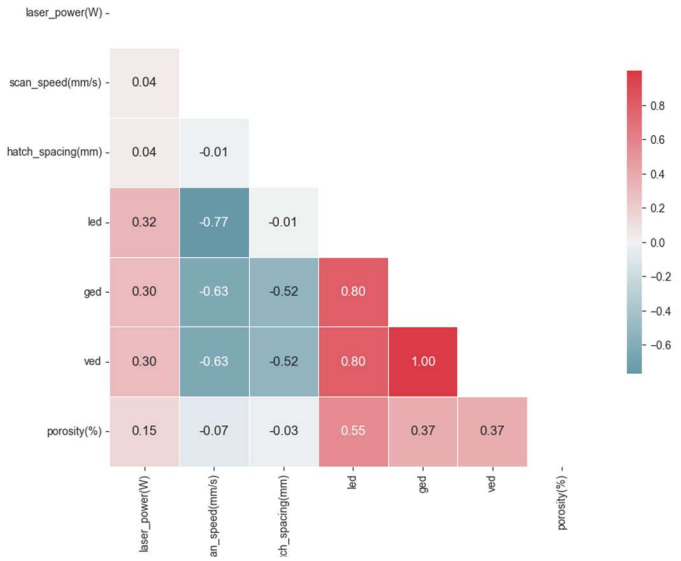


Figure 8: Correlation chart

The research builds on these insights by incorporating LED, GED, and VED as input features in machine learning models. This approach enables the development of a predictive framework capable of identifying the optimal set of process parameters, ensuring the production of parts with minimal porosity. By modeling the interactions between energy densities and porosity, the research aims to establish a data-driven method for achieving high-quality manufacturing outcomes.

To achieve this, the study employs a combination of machine learning models, including Linear Regression, Random Forest Regression, and Gaussian Process Regression (GPR). Each model offers unique advantages in handling the complex nature of the SLM process. Linear Regression serves as a starting point, providing a straightforward method for identifying linear relationships between process parameters and porosity. Its simplicity makes it an effective tool for gaining initial insights, although it may not fully capture the non-linear interactions inherent in the SLM process.

Random Forest Regression, on the other hand, offers a more advanced approach by constructing an ensemble of decision trees, each representing different aspects of the process. This model is particularly adept at handling non-linear relationships, making it suitable for capturing the complex dependencies between laser power, scan speed, hatch spacing, and porosity. Random Forest Regression has demonstrated considerable accuracy in predicting porosity, especially when working with datasets that include a variety of process conditions.

Gaussian Process Regression (GPR) was chosen as the primary model due to its flexibility and ability to provide uncertainty estimates alongside predictions. GPR is especially well-suited for modeling non-linearities and capturing complex patterns in the data, making it an ideal choice for predicting outcomes in additive manufacturing processes. Unlike other models, GPR defines a distribution over possible functions that fit the data, allowing it to adapt to intricate relationships between process parameters

and porosity. This adaptability is particularly valuable when working with datasets that are limited in size or contain noise, as it enables the model to account for uncertainty and make more informed predictions.

In addition to implementing these machine learning models; Bayesian optimization techniques were utilized to fine-tune the models' hyperparameters. Bayesian optimization is a probabilistic method that iteratively explores the solution space, identifying the combination of hyperparameters that yield the best predictive performance. This approach enhances the accuracy and efficiency of the machine learning models, ensuring that they are well-equipped to handle the complexities of the SLM process.

The integration of transfer learning into the optimization strategy represents another key component of the proposed solution. Transfer learning involves leveraging knowledge gained from one dataset or task to improve model performance on a related dataset or task. This approach was applied to combine insights from multiple datasets, enabling the machine learning models to learn from a broader range of process conditions. By incorporating transfer learning, the research effectively addresses the challenges associated with limited data availability, a common issue in additive manufacturing where generating new data can be time-consuming and costly.

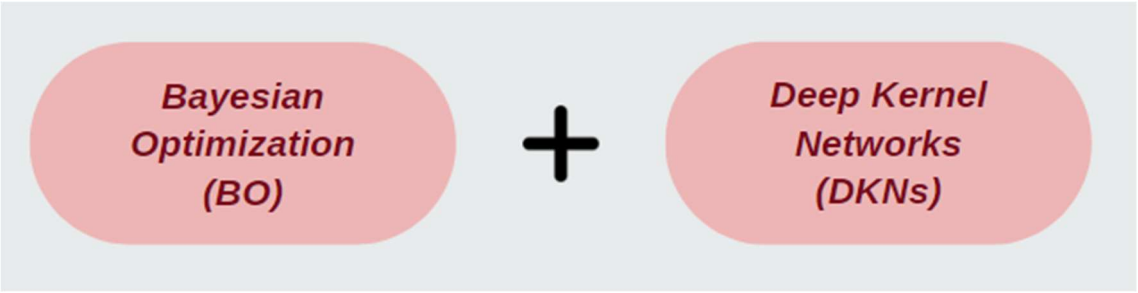


Figure 9: Transfer learning algorithm

The application of transfer learning proved particularly beneficial when working with the Gaussian Process Regression model. By training the GPR model on a

combination of datasets, it was possible to achieve a more comprehensive understanding of how different process parameters influence porosity. This, in turn, led to more accurate predictions and a more effective optimization process.

Data preprocessing and feature engineering played a crucial role in ensuring the success of the machine learning models. Before the data could be used for modeling, it underwent several preprocessing steps, including normalization, handling missing values, and the calculation of derived features such as LED, GED, and VED. These preprocessing steps ensured that the input data was clean, standardized, and ready for analysis, ultimately improving the predictive performance of the models.

In summary, the proposed solution integrates advanced machine learning techniques with data-driven insights to optimize the SLM process parameters for titanium alloy parts. By utilizing Linear Regression, Random Forest Regression, and Gaussian Process Regression, combined with Bayesian optimization and transfer learning, the research establishes a comprehensive framework for minimizing porosity and enhancing part quality. This approach represents a significant advancement in the field of additive manufacturing, offering a more efficient and effective method for process optimization.

5. METHODOLOGY FOLLOWED

5.1. Description of tasks, phases, or procedures

The methodology for this research project was structured meticulously, ensuring a systematic approach to optimizing the Selective Laser Melting (SLM) process for titanium alloy (Ti6Al4V) parts. This structured approach was essential due to the complexity and multi-dimensionality of the problem, as it involved multiple interacting process parameters. The tasks were organized into distinct phases, each addressing a critical aspect of the optimization process and building upon the findings of the previous phase to form a cohesive and comprehensive research workflow.

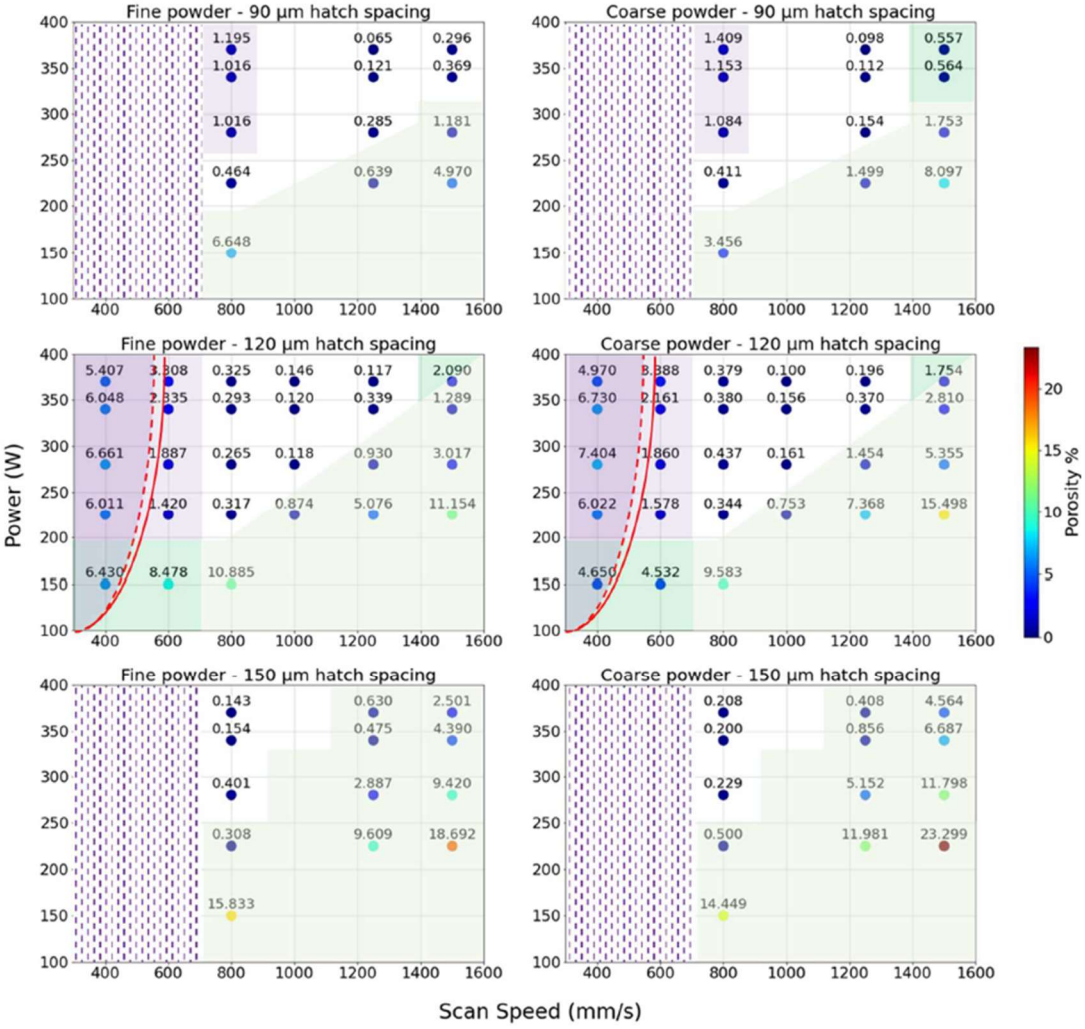


Figure 10: Experimental data used

The initial phase involved an extensive literature review and data collection process. This step aimed to understand the current state of research in additive manufacturing and identify gaps where machine learning could provide significant value. The literature review included an examination of existing optimization techniques, energy density calculations, and machine learning applications in manufacturing. By reviewing studies on Bayesian optimization, Gaussian Process Regression (GPR), and other machine learning models, the project established a foundation for selecting the appropriate methodologies for optimization.

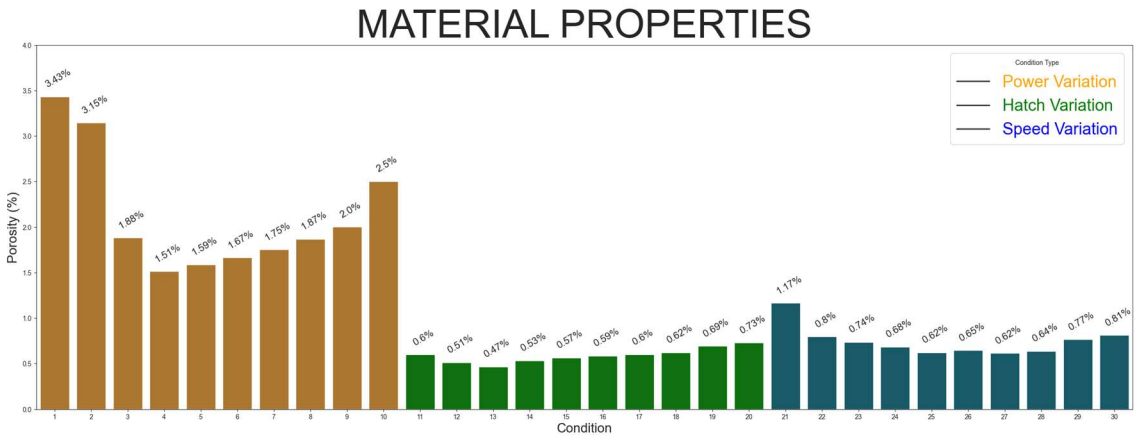


Figure 11: Porosities of dataset

Data collection was a crucial part of this phase. The research utilized datasets from multiple sources, with the primary experimental data obtained from controlled experiments and external studies (e.g., DLR and IIT datasets). These datasets contained detailed records of various SLM process parameters, such as laser power, scan speed, hatch spacing, and the resulting porosity measurements. Collecting high-quality data was vital, as the predictive power of machine learning models depends heavily on the quality and comprehensiveness of the input data. The collected datasets were carefully curated to ensure they captured a broad range of experimental conditions, making it possible to train the machine learning models on diverse scenarios.

The second phase focused on data preprocessing, a step that involved transforming the raw data into a format suitable for machine learning analysis. This process required addressing missing values, detecting outliers, and normalizing the data to ensure consistency across different variables. In many cases, the experimental data contained missing entries or anomalies due to machine errors or variations in experimental setups. These issues were addressed by employing techniques such as linear interpolation and imputation, ensuring that the datasets remained robust and complete. The careful preprocessing of data not only improved the quality of the machine learning models but also helped prevent inaccuracies that could arise from noisy or incomplete datasets.

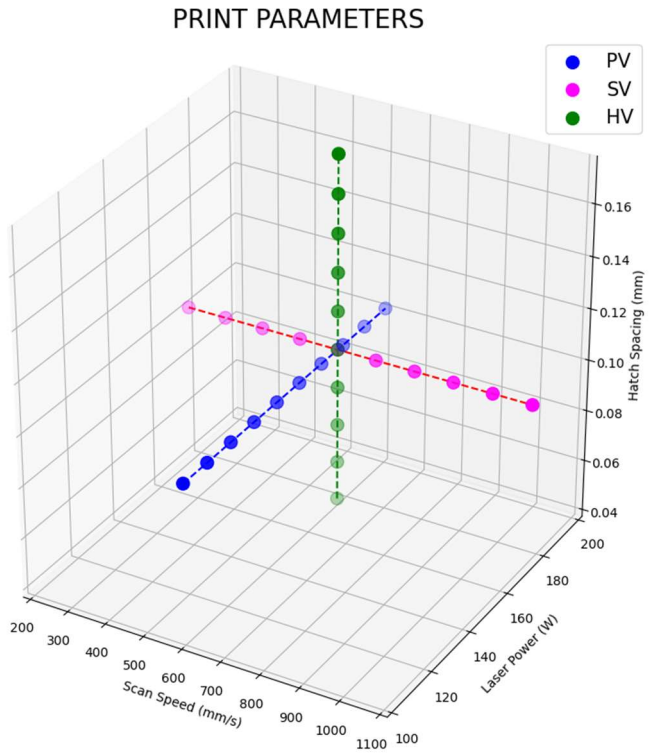


Figure 12: Dataset mapping

Feature engineering was another critical component of this phase. The calculation of energy densities—Linear Energy Density (LED), Global Energy Density (GED), and Volumetric Energy Density (VED)—was central to understanding the

influence of process parameters on porosity. These energy densities were derived from fundamental equations that captured the relationship between laser power, scan speed, hatch spacing, and layer thickness. Calculating these values provided a more comprehensive understanding of how energy input affected the melting and solidification of the titanium powder, directly influencing the formation of porosity.

The third phase involved implementing and training machine learning models. This phase required a deep understanding of different machine learning algorithms and their suitability for the SLM optimization problem. The project implemented Linear Regression, Random Forest Regression, and Gaussian Process Regression (GPR), each offering unique advantages and challenges. These models were trained using the preprocessed datasets, and their performance was evaluated using various metrics, such as Root Mean Squared Error (RMSE) and R-squared (R^2). This process involved iterative training and validation to refine the models and improve their predictive accuracy.

The fourth phase focused on hyperparameter optimization using Bayesian techniques. This step was crucial in fine-tuning the models to achieve optimal performance. Bayesian optimization was chosen for its efficiency in exploring complex, multi-dimensional search spaces. By iteratively adjusting hyperparameters, such as learning rates, tree depths (for Random Forest), and kernel parameters (for GPR), the models were optimized to minimize prediction errors. This process significantly enhanced the accuracy of the models and ensured they could capture the complex relationships between process parameters and porosity.

Finally, the fifth phase involved the analysis and interpretation of results. The trained models were used to predict porosity outcomes across different combinations of process parameters. These predictions were compared against actual experimental data to validate the models' accuracy. The analysis also included generating

visualizations, such as contour plots, heatmaps, and three-dimensional scatter plots, which provided valuable insights into the optimal printing conditions. These visualizations served as a practical tool for identifying the regions in the parameter space where porosity was minimized, guiding manufacturers in setting up their SLM processes more efficiently.

5.2. Preprocessing of data

Data preprocessing was a critical and detailed step that ensured the machine learning models received high-quality input data. This step involved several key activities, each designed to address specific challenges associated with the experimental datasets.

The first aspect of data preprocessing was data cleaning, which addressed issues such as missing values, inconsistencies, and outliers. Missing values are a common problem in experimental data due to various reasons, such as sensor failures, interruptions in the manufacturing process, or incomplete data recording. In this project, missing values were handled using interpolation methods, where the missing data points were estimated based on surrounding values. For instance, if a data point for laser power was missing, the average value from similar experimental conditions was used to fill the gap. This ensured that the dataset remained comprehensive, and that no valuable information was lost.

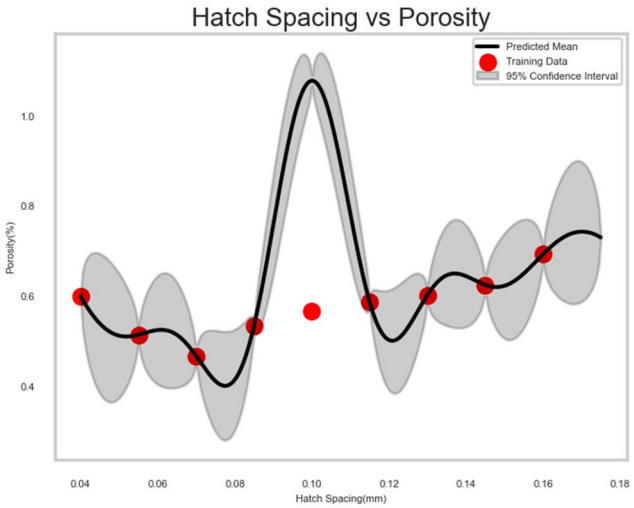


Figure 13: Error in dataset

Outlier detection was another essential component of data cleaning. Outliers, which are data points that deviate significantly from the expected range, can skew the results and negatively impact the performance of machine learning models. These

outliers were identified using statistical techniques, such as Z-scores, and were either removed or corrected to prevent them from influencing the models' predictions.

Feature scaling was another important preprocessing step. Since the machine learning models used in this study are sensitive to the scale of input features, it was necessary to standardize all parameters to a common scale. For example, laser power values ranged from 150 to 370 watts, while hatch spacing values ranged from 0.09 to 0.15 millimeters. By standardizing these features to have a mean of zero and a standard deviation of one, the project ensured that each parameter contributed equally to the model training process, preventing any single feature from dominating the predictions.

The calculation of energy densities (LED, GED, and VED) represented a significant feature engineering effort. These derived features encapsulated the combined effects of multiple process parameters, providing a more holistic view of the energy input during the SLM process. For instance, LED represents the energy input per unit length, providing insights into how the laser interacts with the material as it moves across the powder bed. Similarly, GED accounts for the energy input per unit area, while VED represents the energy input per unit volume. By incorporating these energy densities into the dataset, the machine learning models could better understand the interplay between laser power, scan speed, hatch spacing, and layer height, resulting in more accurate porosity predictions.

5.3. Calculations, algorithms

The research employed three main machine learning models: Linear Regression, Random Forest Regression, and Gaussian Process Regression (GPR). Each model was chosen based on its strengths and suitability for handling the complexity and variability inherent in the Selective Laser Melting (SLM) process. This section will discuss the theory behind each of these models, explaining how they function and why they were appropriate for predicting porosity in the SLM process.

5.3.1. Linear Regression

Linear Regression is one of the most fundamental machine learning algorithms and is commonly used for predictive modeling when a linear relationship is presumed between independent variables (predictors) and a dependent variable (target). The underlying theory of Linear Regression is based on the concept of fitting a straight line, known as the regression line, through the data points in a way that minimizes the overall distance between the observed values and the predicted values.

The equation for a simple linear regression model can be expressed as:

$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon$ where:

- y is the predicted dependent variable (in this case, porosity),
- $x_1, x_2, \ldots, x_n$ are the independent variables (process parameters such as laser power, scan speed, and hatch spacing),
- β_0 is the intercept of the regression line,
- $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients representing the impact of each predictor on the dependent variable,
- ϵ is the error term, representing the difference between the observed and predicted values.

Linear Regression assumes that the relationship between the predictors and the target variable is linear, meaning that any change in a predictor results in a proportional

change in the target variable. In this study, Linear Regression was initially applied to establish a baseline understanding of how the process parameters affect porosity. Despite its simplicity and interpretability, Linear Regression has limitations, particularly when dealing with complex, non-linear relationships, which are typical in manufacturing processes like SLM.

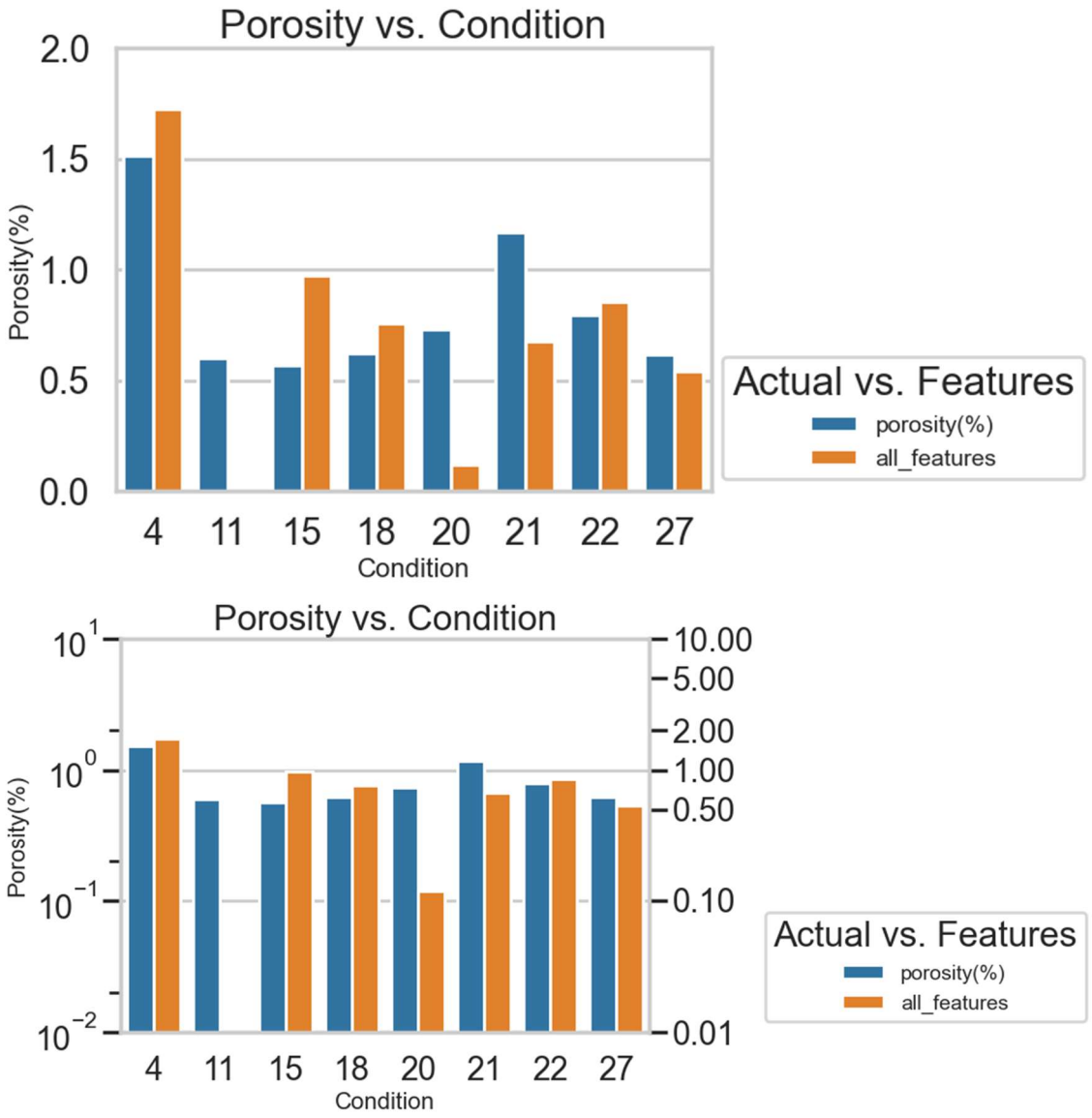


Figure 14: Porosity predictions linear regression

5.3.2. Random Forest Regression

Random Forest Regression is a more sophisticated machine learning technique that addresses the limitations of Linear Regression by capturing non-linear relationships between variables. It is an ensemble learning method that combines the predictions of multiple decision trees to arrive at a more accurate and robust prediction.

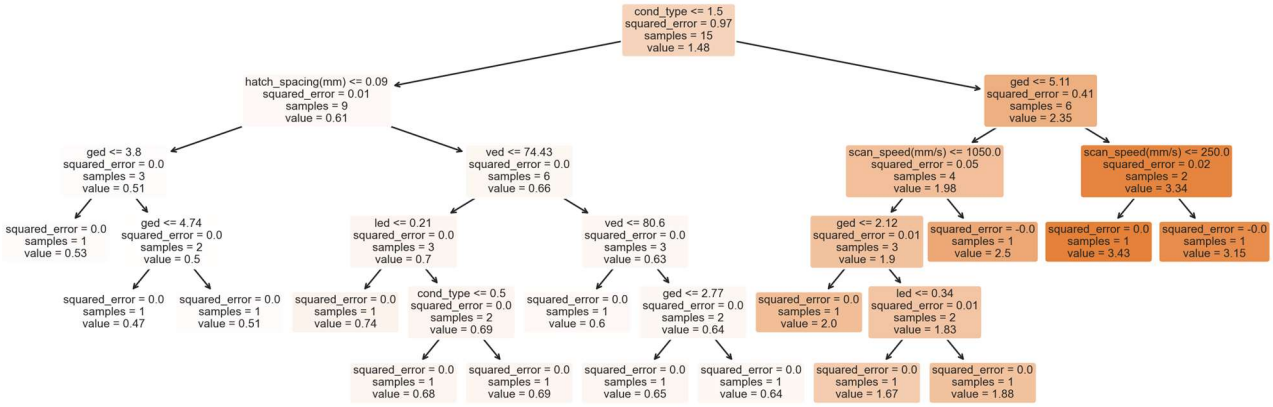


Figure 15: Random forest example

The theory behind Random Forest lies in the concept of "bootstrapping" and "aggregation" (also known as "bagging"). The process involves training several decision trees on different subsets of the dataset, where each subset is generated by sampling the data with replacement (bootstrapping). Each decision tree in the forest makes predictions independently, and the final prediction is obtained by averaging the outputs from all the trees, which helps reduce overfitting and improves accuracy.

A decision tree itself is a flowchart-like structure where each internal node represents a decision based on a specific feature, each branch represents the outcome of that decision, and each leaf node represents a final prediction. During training, the tree splits the data at each node by selecting the feature and threshold that best separates the data points according to a metric such as mean squared error (for regression tasks).

Random Forest's ability to handle complex interactions and non-linear relationships makes it well-suited for predicting porosity in the SLM process, where multiple parameters interact in intricate ways. It also has the advantage of handling high-dimensional data and providing feature importance scores, which offer insights into which process parameters most significantly influence porosity.

5.3.3. Gaussian Process Regression (GPR)

Gaussian Process Regression (GPR) is a probabilistic, non-parametric model that provides a flexible and powerful approach to regression tasks, especially in cases where the relationship between variables is complex and highly non-linear. GPR is fundamentally different from other machine learning models because it does not make direct assumptions about the functional form of the data (such as a linear or polynomial relationship). Instead, it defines a distribution over possible functions that could explain the data and selects the most likely function based on the observed data points.

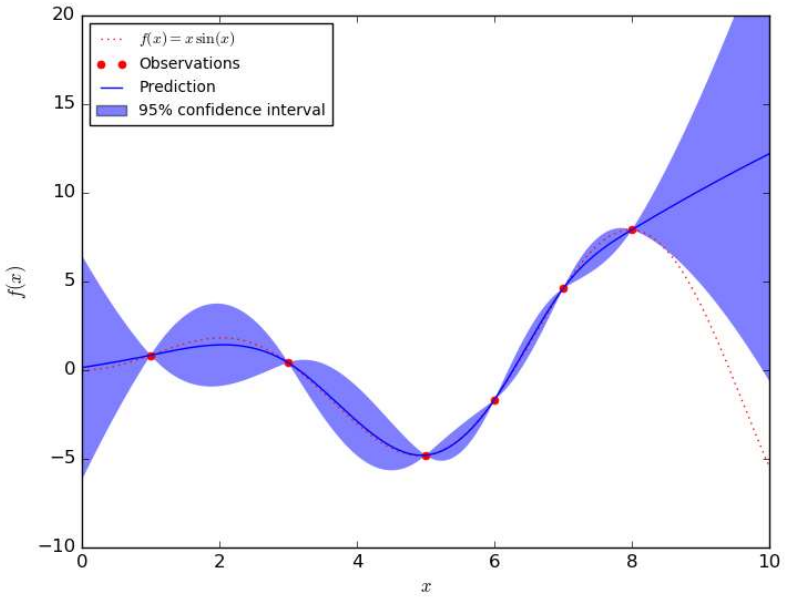


Figure 16: Gaussian Process Regression

The theory behind GPR is based on the Gaussian Process, which is a generalization of the Gaussian (normal) distribution to functions. A Gaussian Process can be thought of as a distribution over functions, characterized by a mean function $m(x)$ and a covariance function (kernel) $k(x,x')$, where: $f(x) \sim GP(m(x),k(x,x'))$

- $f(x)$ is the predicted output,
- $m(x)$ represents the mean of the process, usually assumed to be zero in practice,
- $k(x,x')$ is the kernel function that defines the similarity between any two input points x and x' .

The kernel function is the heart of GPR, as it encodes assumptions about the data's underlying structure. Common kernels include the Radial Basis Function (RBF) kernel and the Matern kernel. The RBF kernel assumes that data points that are closer in input space are more similar, while the Matern kernel allows for more flexibility in capturing roughness in the data. During the training process, GPR uses the kernel to calculate the covariance matrix, which represents the relationships between all pairs of data points.

The predictive capability of GPR is enhanced by its ability to provide uncertainty estimates for its predictions. Unlike other models that provide a single-point estimate, GPR outputs a mean prediction along with a variance, indicating the confidence level in that prediction. This characteristic is particularly useful in scenarios where data is sparse or noisy, as it allows for more informed decision-making.

In the context of this research, GPR was employed to model the highly non-linear relationships between the SLM process parameters and porosity. Its probabilistic nature and flexibility in capturing complex patterns made it the most effective model for this task, outperforming both Linear Regression and Random Forest Regression in terms of accuracy and robustness. The uncertainty estimates provided by GPR were also valuable in identifying regions of the parameter space where predictions were less certain, guiding further experimentation and data collection efforts.

5.3.4. Comparison and Justification for Model Selection

The choice of these three models was deliberate and based on their ability to handle different levels of complexity in the data. Linear Regression provided a simple, interpretable baseline, allowing us to understand the fundamental trends in how process parameters influenced porosity. However, its assumption of linearity made it less suitable for capturing the full complexity of the SLM process.

Random Forest Regression, with its ensemble approach and ability to model non-linear relationships, proved more adept at handling the interactions between multiple parameters. Its feature importance scores offered valuable insights into which parameters had the greatest influence on porosity, which is particularly useful in a manufacturing context where understanding these relationships is critical for process optimization.

Finally, Gaussian Process Regression (GPR) represented the most advanced model, capable of capturing the intricate, non-linear relationships inherent in the SLM process. Its probabilistic framework allowed for uncertainty quantification, which was particularly valuable when working with limited experimental data. This capability made GPR not only the most accurate model but also the most informative, providing insights into where additional data collection could be most beneficial.

By integrating these three models, the research ensured a comprehensive approach to predicting and optimizing the SLM process. The combination of Linear Regression, Random Forest Regression, and GPR allowed for a nuanced analysis of the relationships between process parameters and porosity, leading to a more effective and data-driven optimization strategy.

5.4. Description / Analysis of results

The analysis of the results from the machine learning models—Linear Regression, Random Forest Regression, and Gaussian Process Regression (GPR)—provided a comprehensive understanding of how the process parameters of the Selective Laser Melting (SLM) technique influence porosity in Ti6Al4V alloy parts. The results highlighted the strengths and limitations of each model, demonstrating the varying degrees of success in predicting and optimizing porosity based on the input features.

5.4.1. Linear Regression Analysis

Linear Regression served as the baseline model for this study, offering an initial insight into the relationship between the key process parameters (laser power, scan speed, hatch spacing) and the porosity levels of the printed parts. The simplicity of Linear Regression made it an excellent starting point for understanding how changes in these parameters might impact porosity, as it provided a clear, interpretable set of coefficients representing the influence of each parameter.

When analyzing the results from the Linear Regression model, it became evident that certain parameters had a more direct and predictable impact on porosity. For instance, an increase in laser power generally led to a decrease in porosity up to a certain point, after which excessive laser power began to introduce defects such as keyhole porosity due to overheating. Similarly, the scan speed exhibited a more complex relationship with porosity; slower scan speeds allowed for better melting of the powder particles, resulting in lower porosity, but excessively slow speeds led to thermal distortions and irregularities.

Despite providing useful insights into these trends, the limitations of Linear Regression became apparent when attempting to capture the non-linear interactions between parameters. The model's performance, as measured by the Root Mean

Squared Error (RMSE) and R-squared (R^2) values, was suboptimal, with RMSE values indicating a significant deviation between the predicted and actual porosity values. The R^2 score, which measures how well the model explained the variance in the data, was relatively low, suggesting that the linear model could not fully capture the complexities of the SLM process.

The Linear Regression model's inability to accommodate the non-linear dependencies between parameters highlighted the need for more sophisticated models capable of understanding these intricate relationships. However, its results were still valuable in confirming that certain process parameters were indeed influential and needed to be considered in more advanced modeling efforts.

5.4.2. Random Forest Regression Analysis

The Random Forest Regression model represented a significant advancement over Linear Regression by offering the capability to capture non-linear relationships and interactions between the process parameters. By constructing an ensemble of decision trees, the Random Forest model could analyze the data from multiple perspectives, ultimately generating more accurate and reliable predictions for porosity.

One of the key advantages of the Random Forest model was its ability to provide feature importance scores. These scores quantified the contribution of each process parameter (e.g., laser power, scan speed, hatch spacing, LED, GED, VED) to the overall prediction, offering valuable insights into which parameters had the most significant impact on porosity. The results showed that laser power and scan speed were consistently the most influential factors, while hatch spacing and the derived energy densities (LED, GED, VED) also played a substantial role in determining porosity outcomes.

The Random Forest model achieved a much lower RMSE compared to Linear Regression, indicating a closer match between the predicted and actual porosity values. The R^2 score was also significantly higher, suggesting that the model was capable of explaining a greater proportion of the variance in the data. This improvement in performance demonstrated the model's ability to handle the complex, multi-dimensional nature of the SLM process, making it a powerful tool for predicting porosity.

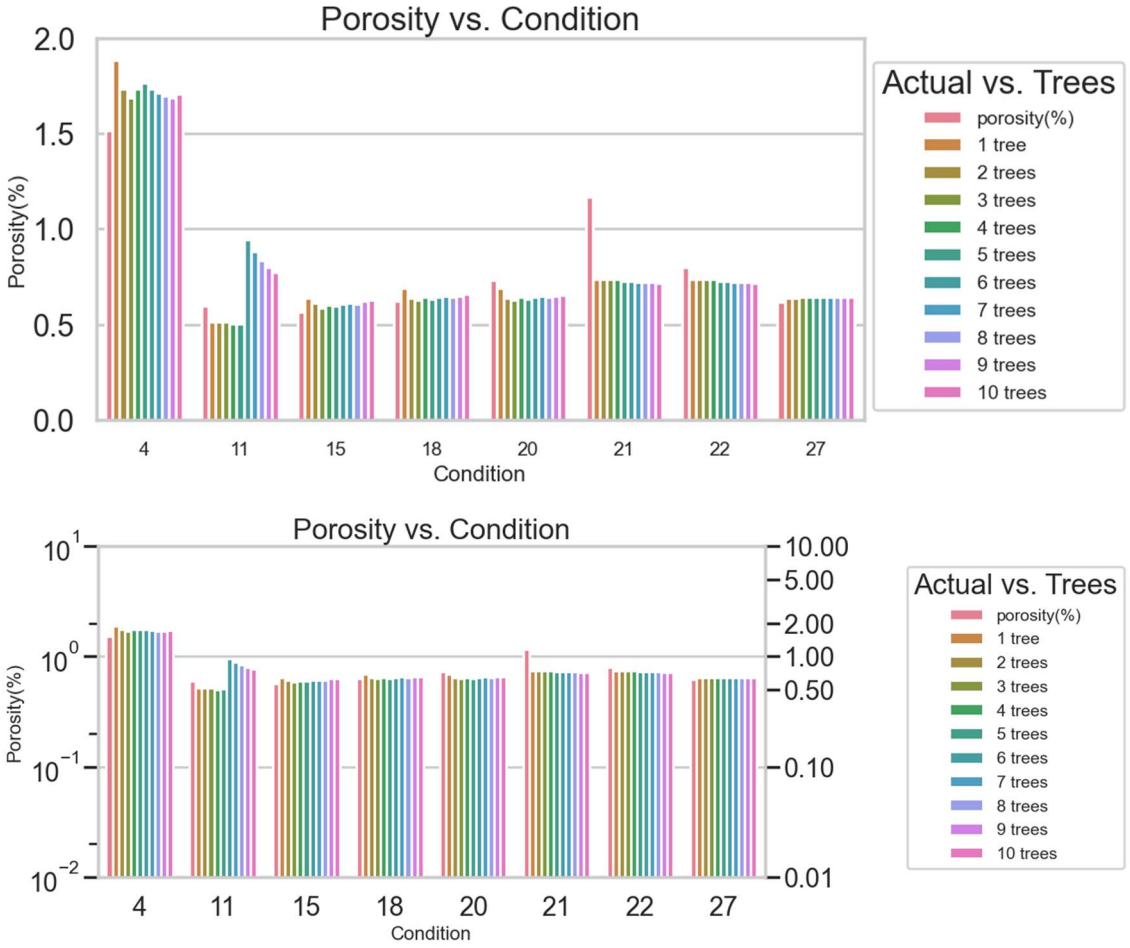


Figure 17: Random Forest Predictions

Additionally, the Random Forest model's robustness to overfitting was noteworthy. Due to its ensemble nature, the model averaged predictions across multiple trees, reducing the likelihood of being influenced by noise or anomalies in the data. This characteristic made it particularly effective in generating accurate predictions

across a wide range of experimental conditions, ensuring that it could adapt to different combinations of process parameters.

The visualizations generated from the Random Forest model, such as feature importance plots and partial dependence plots, provided deeper insights into how specific parameters influenced porosity. For instance, the partial dependence plots revealed that there was a threshold effect for laser power, where increasing the power beyond a certain point led to diminishing returns in reducing porosity. These insights are crucial for practitioners aiming to optimize the SLM process, as they highlight the parameter ranges where adjustments are most effective.

5.4.3. Gaussian Process Regression (GPR) Analysis

Gaussian Process Regression (GPR) emerged as the most effective model for this study, offering a high level of predictive accuracy and the ability to model uncertainty in the predictions. GPR's strength lies in its flexibility and its probabilistic nature, which allows it to adapt to complex patterns in the data. Unlike the other models, GPR provides not only a mean prediction but also an estimate of uncertainty, indicating how confident the model is in its predictions across different regions of the parameter space.

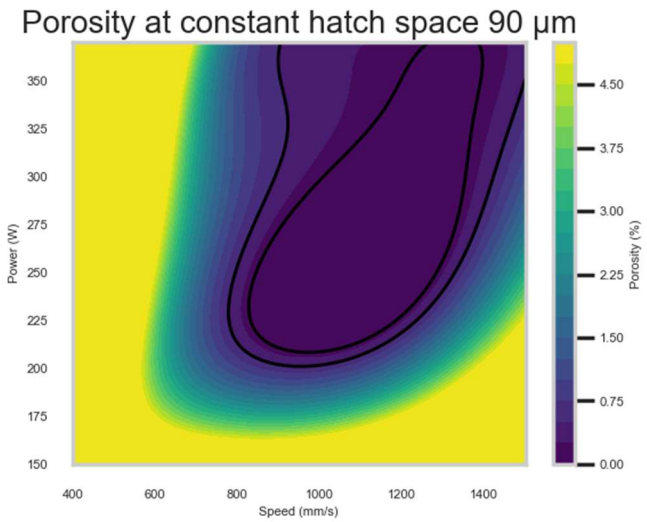


Figure 18: Porosity mapping 90um

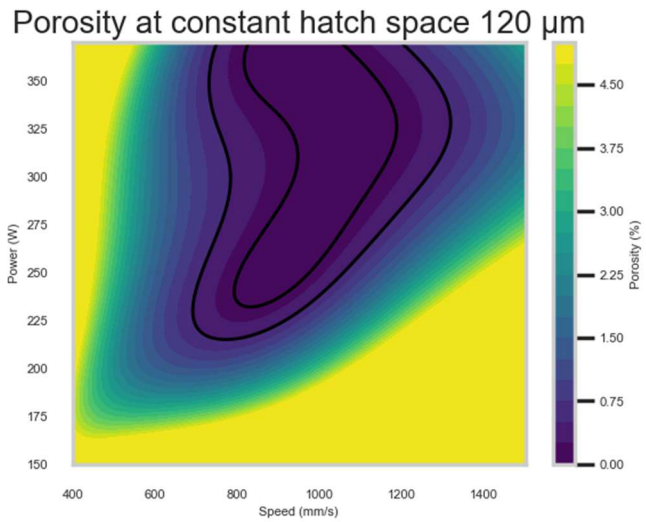


Figure 19: Porosity mapping 120um

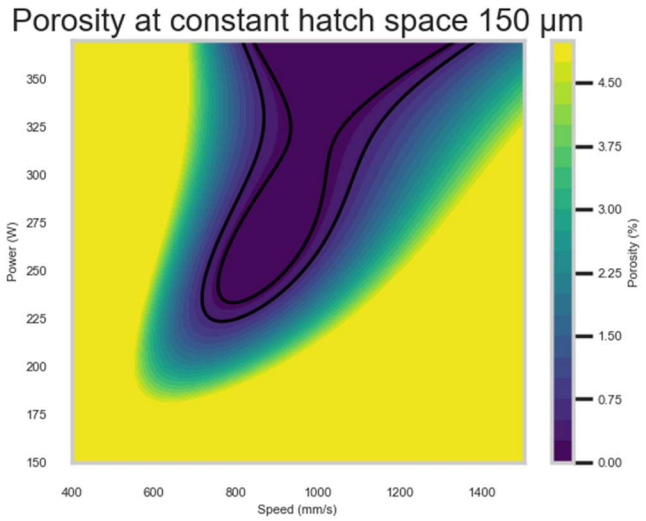


Figure 20: Porosity mapping 150um

The application of GPR to the SLM process data revealed several important trends. The model was able to capture intricate, non-linear relationships between the process parameters and porosity that were not evident in the simpler models. For example, GPR effectively modeled the interaction between laser power and scan speed, demonstrating that optimal porosity levels were achieved within a specific range of these parameters. Deviating from this range in either direction led to a significant increase in porosity, indicating that a fine balance between these parameters is essential for high-quality part production.

The uncertainty estimates provided by GPR were particularly valuable in identifying areas where the model's predictions were less certain. This information is critical for manufacturing practitioners, as it indicates where additional experimental data might be needed to improve the model's accuracy. For instance, in regions of the parameter space where uncertainty was high, further experimentation could help refine the model's understanding and lead to more reliable predictions.

The RMSE values for the GPR model were the lowest among all the models tested, and the R^2 scores were the highest, indicating that GPR was the most accurate

and reliable model for predicting porosity. These results demonstrate the model's ability to generalize well, even when working with complex, multi-dimensional data. The ability of GPR to handle uncertainty and provide probabilistic predictions makes it an ideal tool for optimizing the SLM process, particularly in scenarios where experimental data is limited or noisy.

The contour plots generated using GPR's predictions offered clear visualizations of the optimal parameter regions for minimizing porosity. These plots showed how different combinations of laser power, scan speed, and hatch spacing influenced porosity, providing a practical guide for adjusting process parameters. For instance, the plots revealed an optimal "sweet spot" for laser power and scan speed, where porosity was minimized, offering actionable insights for fine-tuning the SLM process.

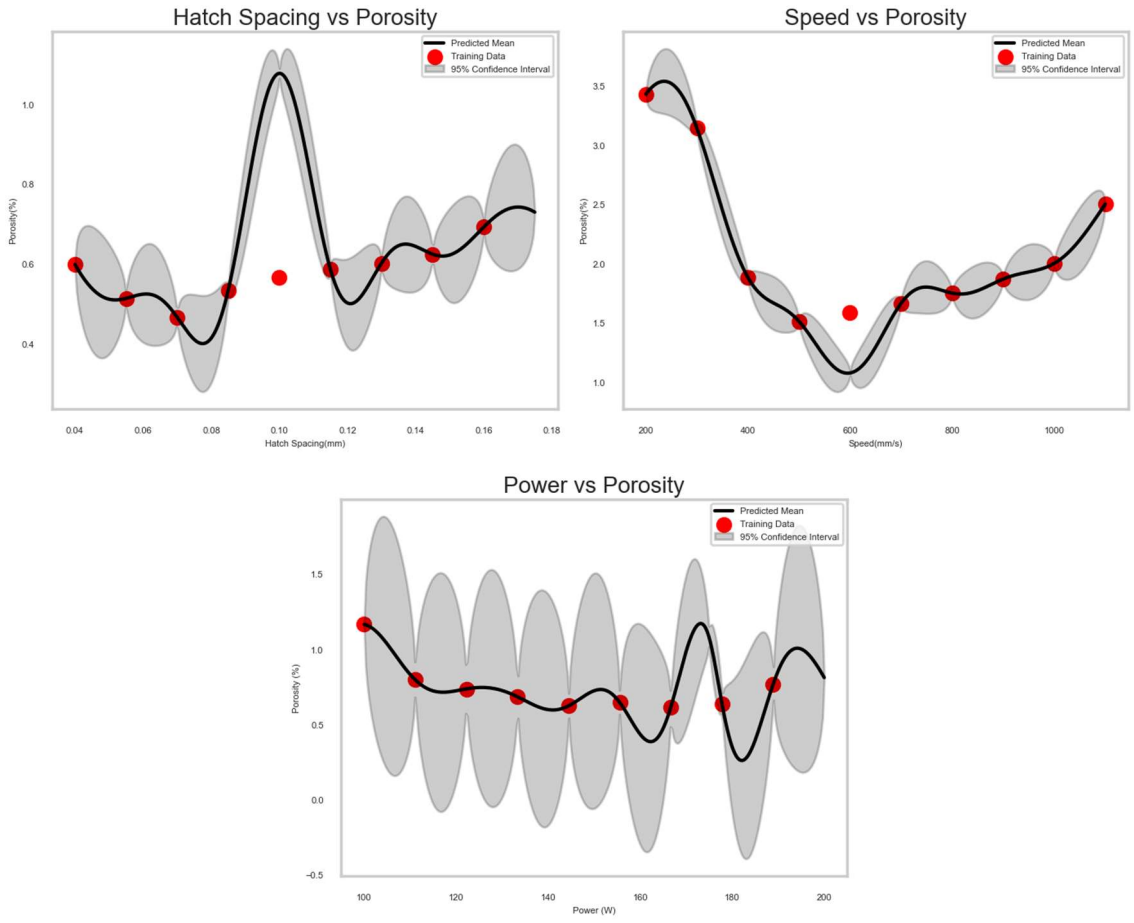


Figure 21: GPR in single variation

5.4.4. Transfer Learning and Its Impact on Results

The application of transfer learning played a crucial role in enhancing the predictive capabilities of the machine learning models, particularly GPR. By leveraging data from multiple sources (e.g., DLR and IIT datasets), the models were able to learn from a broader range of experimental conditions, which improved their ability to generalize to new data. Transfer learning was especially effective in addressing the challenge of limited data availability, a common issue in additive manufacturing studies where generating experimental data can be expensive and time-consuming.

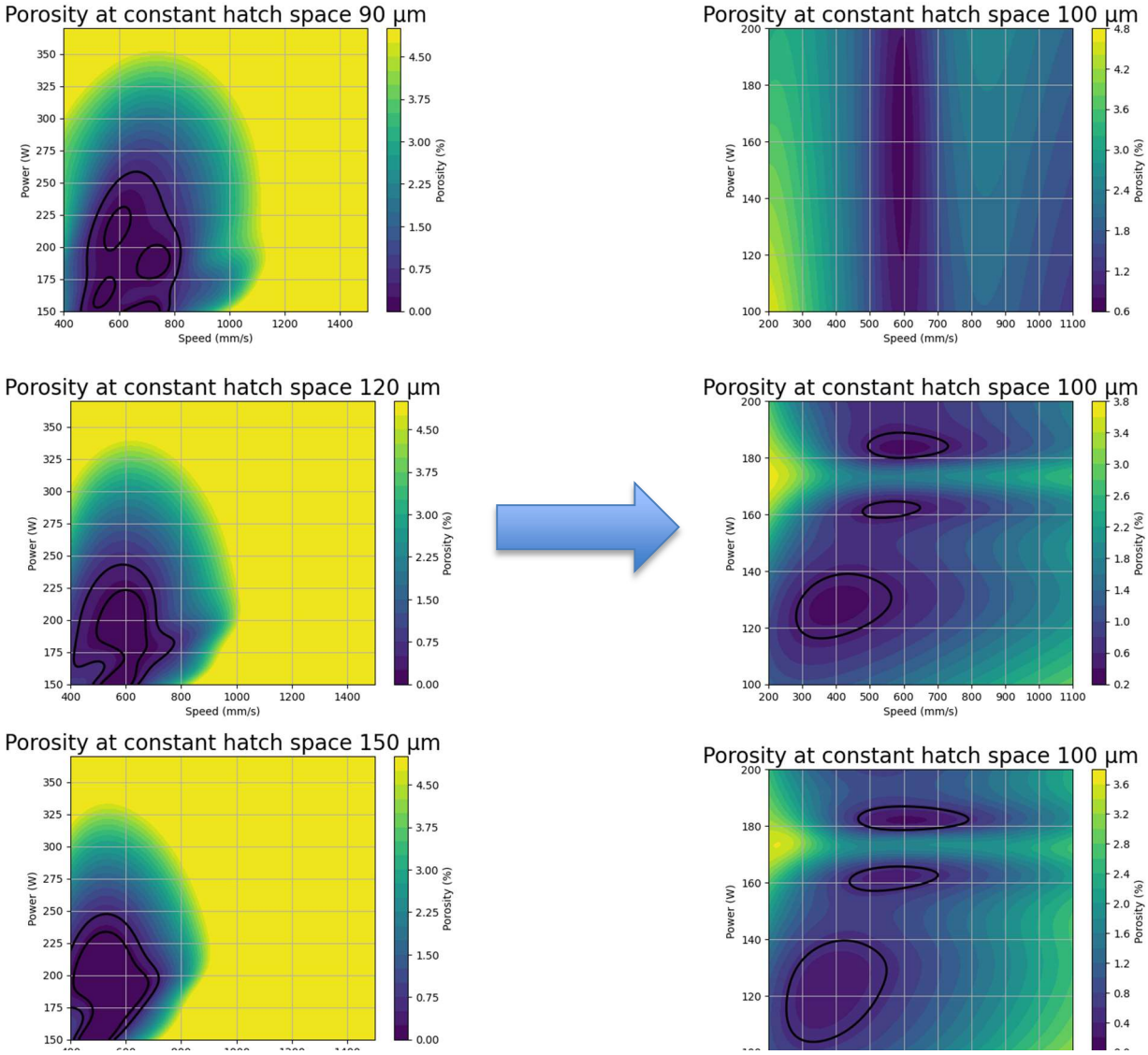


Figure 22: Transfer learning process

The impact of transfer learning was evident in the improved accuracy of the GPR model, as well as in its ability to make more confident predictions across different regions of the parameter space. By incorporating knowledge from different datasets, the model developed a more nuanced understanding of how process parameters influence porosity, which led to more reliable optimization recommendations.

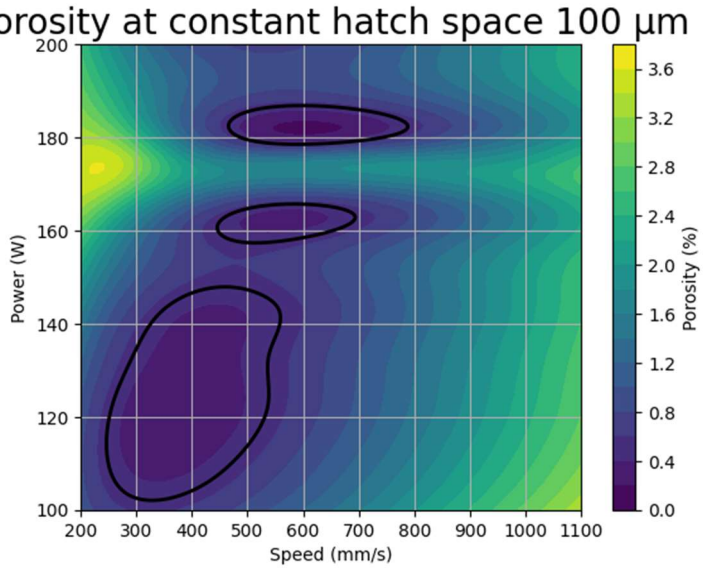


Figure 23: Original mapping for example

5.4.5. Practical Implications of the Results

The findings from this research have significant practical implications for additive manufacturing practitioners. By demonstrating the effectiveness of machine learning models—especially GPR—in predicting and optimizing porosity, the study provides a clear pathway for integrating data-driven approaches into the SLM process. The insights gained from the feature importance analysis, partial dependence plots, and contour visualizations offer actionable guidelines for adjusting process parameters, enabling manufacturers to produce parts with minimal porosity and improved mechanical properties.

Moreover, the ability of machine learning models to handle complex, multi-dimensional data means that they can be used to optimize other aspects of the SLM

process, such as surface finish, mechanical strength, or production efficiency. The integration of transfer learning further enhances the potential for these models to be applied across different materials and manufacturing setups, making them a versatile tool for the broader additive manufacturing industry.

In conclusion, the analysis of the results confirmed that machine learning, particularly Gaussian Process Regression, is a powerful tool for optimizing the SLM process. The ability to predict porosity with high accuracy and provide uncertainty estimates offers a significant advantage, enabling more informed decision-making and process optimization. These findings pave the way for the broader adoption of machine learning in additive manufacturing, ultimately contributing to more efficient, cost-effective, and high-quality production processes.

6. CONCLUSIONS

This research project set out to explore and implement machine learning techniques to optimize the Selective Laser Melting (SLM) process for the manufacturing of titanium alloy (Ti6Al4V) parts, with the ultimate goal of minimizing porosity and enhancing part quality. The study has demonstrated the significant potential of machine learning in addressing the challenges of additive manufacturing, providing a data-driven approach to optimizing complex, multi-parameter processes.

The investigation began with a thorough analysis of how critical process parameters, such as laser power, scan speed, and hatch spacing, influence the porosity of parts manufactured using the SLM technique. The study also incorporated energy density metrics—Linear Energy Density (LED), Global Energy Density (GED), and Volumetric Energy Density (VED)—as derived features, which were found to be crucial in capturing the combined effects of these parameters on part quality. By integrating these metrics into the machine learning models, the research provided a more holistic view of the process, leading to more accurate predictions and optimizations.

Three primary machine learning models—Linear Regression, Random Forest Regression, and Gaussian Process Regression (GPR)—were employed to predict porosity based on the process parameters. Each model demonstrated varying levels of effectiveness, highlighting their respective strengths and limitations:

- ***Linear Regression*** provided an initial understanding of the linear relationships between process parameters and porosity. While its simplicity offered valuable insights, it was ultimately insufficient for capturing the non-linear interactions that characterize the SLM process. The model's limited performance underscored the complexity of the problem and the need for more sophisticated algorithms capable of handling non-linearity and interaction effects.

- ***Random Forest Regression*** significantly improved predictive accuracy by capturing the non-linear dependencies and interactions between parameters. Its ability to handle high-dimensional data and generate feature importance scores made it a valuable tool for understanding which process parameters most strongly influenced porosity. The insights gained from the Random Forest model contributed to identifying key optimization strategies, but the model still fell short in certain areas, particularly in providing uncertainty estimates and handling more subtle patterns in the data.
- ***Gaussian Process Regression (GPR***) emerged as the most effective model for predicting and optimizing porosity in the SLM process. GPR's flexibility, combined with its ability to provide uncertainty estimates, allowed it to model the complex, non-linear relationships inherent in the data accurately. The probabilistic nature of GPR made it uniquely suited for this application, especially when data was limited or noisy, as it provided not only predictions but also insights into the confidence level of those predictions. This capability is particularly valuable in additive manufacturing, where obtaining extensive datasets can be challenging due to the high cost and time required for experiments.

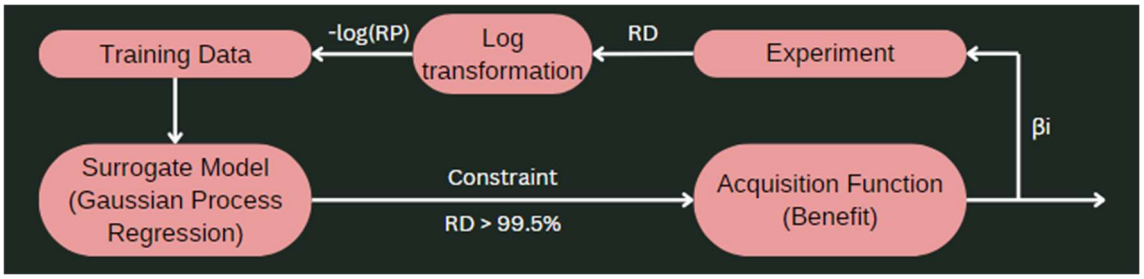


Figure 24: Optimization process

One of the most significant contributions of this study was the successful application of transfer learning to enhance the performance of the machine learning models, particularly GPR. By combining data from multiple sources (e.g., DLR and IIT datasets), the research demonstrated that transfer learning could address the challenge of data scarcity in additive manufacturing, enabling more accurate predictions and better generalization. This approach opens the door for future applications of transfer

learning in similar manufacturing contexts, where data availability is often a limiting factor.

The visualizations generated through this research, such as contour plots, heatmaps, and feature importance analyses, provided actionable insights into the optimal process parameter ranges for achieving minimal porosity. These tools not only confirmed the effectiveness of the machine learning models but also offered practical guidance for manufacturers seeking to optimize their SLM processes. The identification of optimal "sweet spots" for laser power, scan speed, and hatch spacing is particularly valuable, as it enables practitioners to make data-driven adjustments that lead to improved part quality and reduced material waste.

This study's findings have several important implications for the field of additive manufacturing. First, the demonstrated effectiveness of machine learning models, especially GPR, shows that these techniques can significantly improve process optimization, resulting in parts with superior mechanical properties and lower defect rates. The ability to predict and minimize porosity is a crucial step toward the broader adoption of additive manufacturing for high-performance applications, such as aerospace, biomedical, and automotive industries, where part quality and reliability are paramount.

Second, the use of machine learning models provides a more efficient and cost-effective alternative to traditional trial-and-error experimentation. By leveraging existing data to build predictive models, manufacturers can reduce the number of physical experiments required to identify optimal process parameters, leading to shorter development cycles and reduced production costs. This efficiency gain is particularly important in industries where material costs and machine time are significant factors.

However, the research also highlighted certain limitations and challenges that must be addressed in future studies. One of the primary challenges is the quality and availability of data. While transfer learning helped mitigate the impact of limited data availability, the accuracy and generalizability of machine learning models could be further improved with access to larger, more diverse datasets. This points to the need for collaborative data-sharing initiatives within the additive manufacturing community, where researchers and practitioners can pool their data to develop more robust models.

Additionally, while GPR proved to be the most effective model for this study, it is computationally intensive, especially as the dataset size increases. Future research could explore more scalable machine learning techniques, such as deep learning or hybrid models, which may offer similar levels of accuracy with reduced computational overhead. These advanced models could also be integrated into real-time process monitoring systems, enabling adaptive control of the SLM process as conditions change.

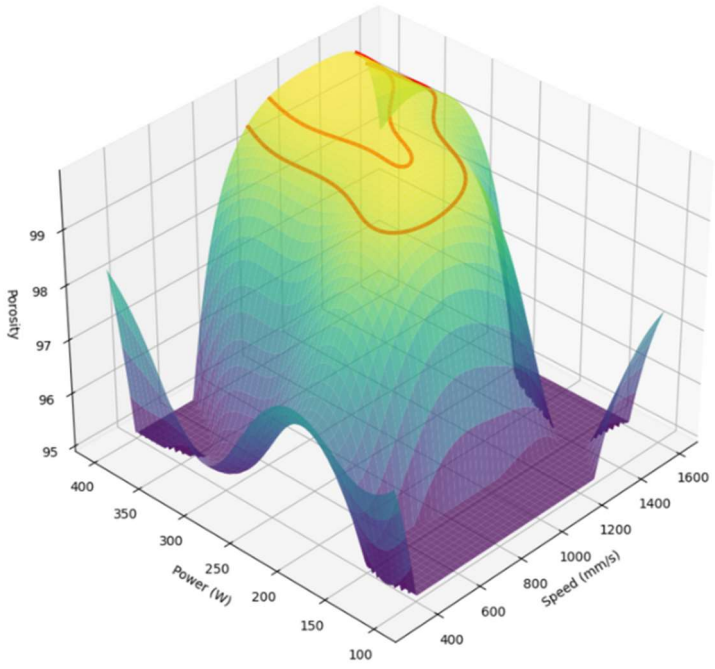


Figure 25: 3D porosity mapping

In conclusion, this research has demonstrated the transformative potential of machine learning in optimizing additive manufacturing processes. By applying Linear Regression, Random Forest Regression, and Gaussian Process Regression models to predict and minimize porosity, the study has provided a valuable framework for data-driven process optimization in the SLM technique. The successful implementation of transfer learning further underscores the potential for machine learning to adapt to various manufacturing contexts, even in the face of data limitations.

The insights gained from this study pave the way for broader applications of machine learning in additive manufacturing, from optimizing other quality metrics such as surface roughness and mechanical strength to enabling real-time process adjustments based on predictive analytics. As the field continues to evolve, the integration of machine learning into additive manufacturing holds the promise of unlocking new levels of efficiency, precision, and quality, ultimately driving the adoption of this technology across a wide range of industries. This research represents an important step toward realizing that potential, demonstrating that machine learning is not just a theoretical tool but a practical, impactful solution for the challenges faced in modern manufacturing.

7. BIBLIOGRAPHY

Asherloo, Mohammadreza, et al. “Laser-beam powder bed fusion of cost-effective non-spherical hydride-dehydride ti-6al-4v alloy.” *Additive Manufacturing*, vol. 56, Aug. 2022, p. 102875, https://doi.org/10.1016/j.addma.2022.102875.

Craig, Miles. “Increase AM Yield: Final Report.” *GitHub Project*, 2019.

Dong, Y.P., et al. “Cost-affordable ti-6al-4v for additive manufacturing: Powder Modification, compositional modulation and laser in-situ alloying.” *Additive Manufacturing*, vol. 37, Jan. 2021, p. 101699, https://doi.org/10.1016/j.addma.2020.101699.

Du Plessis, Anton. “Effects of process parameters on porosity in laser powder bed fusion revealed by X-ray tomography.” *Additive Manufacturing*, vol. 30, Dec. 2019, p. 100871, https://doi.org/10.1016/j.addma.2019.100871.

Frazier, Peter. “Tutorial: Optimization via simulation with Bayesian statistics and Dynamic Programming.” *Proceedings Title: Proceedings of the 2012 Winter Simulation Conference (WSC)*, vol. 2, Dec. 2012, pp. 1–16, https://doi.org/10.1109/wsc.2012.6465237.

Iranzo Juan, Ignacio. “Bayesian Optimization applied to Hybrid Electric Vehicle.” *Universitat Politècnica de València (UPV)*, 2022.

Kasperovich, Galina, et al. “Correlation between porosity and processing parameters in tial6v4 produced by Selective Laser Melting.” *Materials & Design*, vol. 105, Sept. 2016, pp. 160–170, https://doi.org/10.1016/j.matdes.2016.05.070.

Makrygiorgos, Georgios, et al. “Performance-oriented model learning for control via multi-objective bayesian optimization.” *Computers & Chemical Engineering*, vol. 162, June 2022, p. 107770, https://doi.org/10.1016/j.compchemeng.2022.107770.

Shahriari, Bobak, et al. “Taking the human out of the loop: A review of Bayesian optimization.” *Proceedings of the IEEE*, vol. 104, no. 1, Jan. 2016, pp. 148–175, https://doi.org/10.1109/jproc.2015.2494218.

Shang, Wenjie, et al. “Hybrid data-driven discovery of high-Performance Silver selenide-based thermoelectric composites.” *Advanced Materials*, vol. 35, no. 47, 16 Oct. 2023, https://doi.org/10.1002/adma.202212230.

Smoqi, Ziyad, et al. “Monitoring and prediction of porosity in laser powder bed fusion using physics-informed meltpool signatures and machine learning.” *Journal of Materials Processing Technology*, vol. 304, June 2022, p. 117550, https://doi.org/10.1016/j.jmatprotec.2022.117550.

Theckel Joy, Tinu, et al. “A flexible transfer learning framework for Bayesian optimization with Convergence Guarantee.” *Expert Systems with Applications*, vol. 115, Jan. 2019, pp. 656–672, https://doi.org/10.1016/j.eswa.2018.08.023.

8. ANNEX

8.1. Annex I: Poster Presentation

8.2. Annex II: Python Code

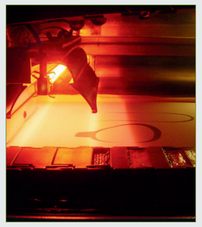
Process Optimization in Advance Manufacturing

Student: Adrian Alonso Cuende
Advisor: Dr. Baisravan HomChaudhuri
Department of Mechanical, Materials, and Aerospace Engineering
Illinois Institute of Technology

Project Objectives & Goals

Apply Bayesian Optimization to process parameters in Laser-beam Powder Bed Fusion.

- Ensure defect-free parts for several applications and industries.
- Critical components in aerospace applications require relative densities greater than **99.9%**.
- Most industrial applications, a relative density of **99.5%** is sufficient.
- Maximize building rate of PBF, to improve producibility of the process.
- Optimization of energy density taking into account density constraint.



Process characteristics

- *Process name:* Laser-Beam Powder Bed Fusion.
- *Powder material:* Ti-6Al-4V alloy

Fixed parameters

- *Layer thickness* = 60 µm
- *Powder flow and focal distance*

Variable parameters

- *Laser speed*
- *Laser power*
- *Hatching space*

Machine Learning and Optimization have become increasingly essential in various fields due to the **high costs** associated with **traditional experimentation**.

Integration of Machine Learning algorithms
Data-driven decisions
Reduce number of experiments and process in preliminary steps of production

→

Enhancing productivity
More accurate forecasting and streamlined operations
Staying competitive in rapidly evolving markets

Background & State of the Art

Gaussian Regression Model

Factors to Consider When Choosing this Model

- Unknown shape of function.
- Complex analytical evaluation of the parameter influence.
- Elevated design cost avoiding great repeatability.

Purposes of the Model

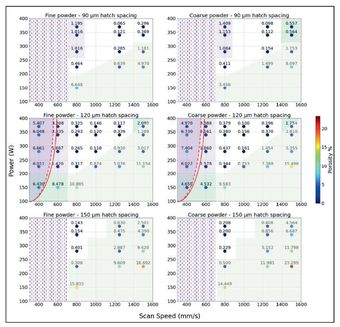
Method	Purpose	Approach
Compositional modelling	**Find patterns within data**	Passive
Exploration	**Learn function as quickly as possible**	Active

Several predefined functions are already implemented in software

Statistics and Machine Learning Toolbox → *fitrgp()*

Scikit-Learn Library → *GaussianProcessRegressor()*

Experimental Data



Source of data

M. Asherloo, ... Amir Mostafei, "*Laser-beam powder bed fusion of cost-effective non-spherical hydride-dehydride Ti-6Al-4V alloy",* Elsevier B.V. 2022

Data gives values of porosity attending to hatch spacing, laser power and laser speed; dividing the analysis between coarse and fine powder.

Other papers

Same material, different process

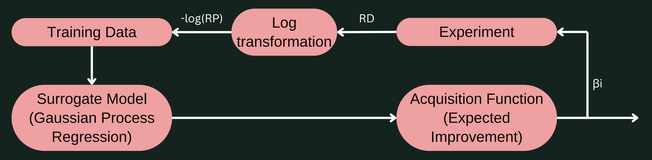
Meta-Learning

Same process, different materials

Different parameters

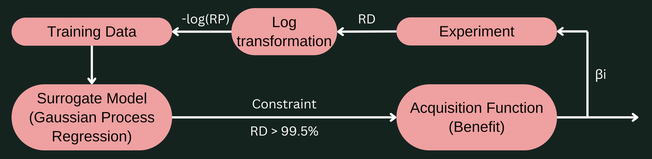
Data & Results

Maximization of the relative density



- Fit the Gaussian Process Regression model using available data for both coarse and fine powder.
- Utilize the acquisition function to suggest the next experiment.
- Conduct a new experiment to collect data.
- Update the model with the new acquired data, the process repeats.

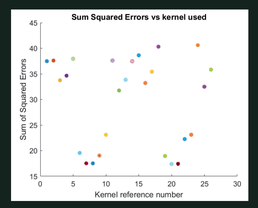
Multiobjective build rate and energy optimization



- Objective function is fully known in this case.
- The optimization of a virtual benefit function involves both energy density and build rate.

$$Benefit\ function = \frac{Build\ Rate}{Energy\ Density} = \frac{(v\ L\ H)^2}{P}$$

Kernel selection



To ensure effectiveness of Bayesian Optimization, select appropriate kernel function and set accurately hyperparameter values for GPR. Procedure:

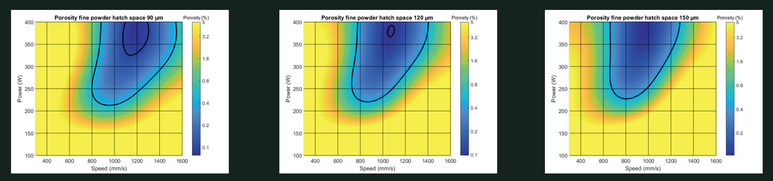
- Evaluation of results according to Sum of Squared Errors (SSE).
- Short data set of 53 points = 52 training data + 1 testing point. Therefore, 53 iterations per model.
- 13 kernel functions available, each with and without hyperparameter optimization.

Selected kernel: ***ARD Matern 3/2***

Kernel analysis and first mapping of the porosity function were performed by previous research student ***Ignacio Iranzo Juan****.*

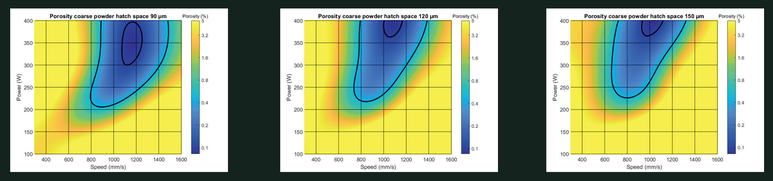
Mapping the Porosity Function

Fine Powder



hatch space 90 µm *hatch space 120 µm* *hatch space 150 µm*

Coarse Powder



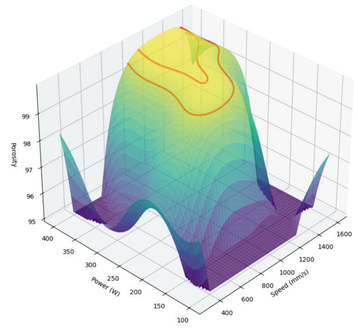
hatch space 90 µm *hatch space 120 µm* *hatch space 150 µm*

Significant improvements in all process properties depending on the specific goals and requirements of the additive manufacturing project:

- MAX Build Rate 5.88 times
- min Energy Density 5.30 times
- MAX Benefit 5.93 times

} nominal value improvement of the benefit metric

Conclusions & Discussions



Out of the numerous parameter changes in PBF, only a ***specific combination*** of them is effective in producing high-density manufactured parts.

Bayesian Optimization has proven to be an incredibly powerful and versatile optimization technique ***reducing the number of experiments*** to characterize process behavior.

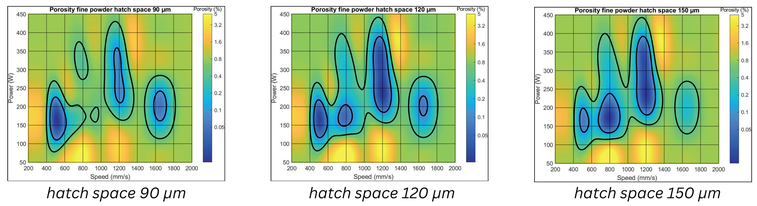
Adaptability of Bayesian Optimization in multi-objective problems, black-box functions and lack of huge data sets.

Meta-Learning from other experiments

Due to the absence of wide datasets for a specific manufacturing process, it becomes necessary to merge multiple experiments.

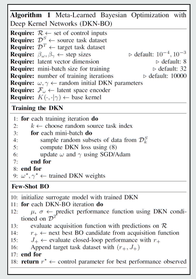
Sharing of experimental numerical results

Next graphs show the porosity function mapping generated from the raw combination of data from multiple sources.



hatch space 90 µm *hatch space 120 µm* *hatch space 150 µm*

Advancement on the path



Algorithm 1 Meta-Learned Bayesian Optimization with Deep Kernel Networks (DKN-BO)

Require: $\mathcal{R}$ ← set of control inputs
Require: $\mathcal{D}^S$ ← source task dataset
Require: $\mathcal{D}^T$ ← target task dataset
Require: $\{\zeta_\omega, \zeta_\gamma\}$ ← step sizes ▷ default: 10^{-4}, 10^{-3}
Require: latent vector dimension ▷ default: 8
Require: mini-batch size for training ▷ default: 32
Require: number of training iterations ▷ default: 10000
Require: ω_0, γ_0 ← random initial DKN parameters
Require: $\mathcal{F}_\omega$ ← latent space encoder
Require: $K(\cdot,\cdot|\gamma)$ ← base kernel

Training the DKN

1: **for** each training iteration **do**
2: $k \leftarrow$ choose random source task index
3: **for** each mini-batch **do**
4: sample random subsets of data from $\mathcal{D}_k^S$
5: compute DKN loss using (8)
6: update ω and γ using SGD/Adam
7: **end for**
8: **end for**
9: $\omega^*, \gamma^* \leftarrow$ trained DKN weights

Few-Shot BO

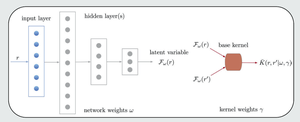
10: initialize surrogate model with trained DKN
11: **for** each DKN-BO iteration **do**
12: $\mu_t, \sigma \leftarrow$ predict performance function using DKN conditioned on $\mathcal{D}_T^0$
13: evaluate acquisition function with predictions on $\mathcal{R}$
14: $r_t \leftarrow$ next best BO candidate from acquisition function
15: $J_t \leftarrow$ evaluate closed-loop performance with r_t
16: Append target task dataset with (r_t, J_t)
17: **end for**
18: **return** $r^* \leftarrow$ control parameter for best performance observed

DKN-BO approach

Proposal of the use of meta-learning to generate initial surrogate model based on data collected from performance optimization tasks performed on a ***variety of systems*** that are different to the target system.

Bayesian Optimization (BO) + ***Deep Kernel Networks (DKNs)***

- Unexplored potential of ***combining*** fine and coarse ***powders*** for their interaction analysis.
- Introduction of ***new variables*** to the process.
- Incorporation of ***new mechanical properties*** to establish a bigger dataset library.



References

[1] Iranzo, I. "Bayesian Optimization for HEV and Additive Manufacturing". *IIT-UPV*, 2023.
[2] Asherloo, M., Mostafaei, A. et al. "Laser-beam powder bed fusion of cost-effective non-spherical hydride-dehydride Ti-6Al-4V alloy". *Elsevier, Additive Manufacturing*, 2022.
[3] Frazier, P. I. "A Tutorial on Bayesian Optimization". 2018.
[4] Chakrabarty, A. "Optimizing Closed-Loop Performance with Data from Similar Systems: A Bayesian Meta-Learning Approach". *Conference on Decision and Control*, 2022.
[5] du Plessis, A. "Effects of process parameters on porosity in laser powder bed fusion revealed by X-ray tomography". *Elsevier, Additive Manufacturing*, 2019.

```
# %% [markdown]
# # Python Packages

# %%
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from functools import reduce # to combine final dfs into a single df
from sklearn.tree import plot_tree

# %% [markdown]
# # Constants

# %%
# Constants used throughout the project

# colors use in presentation and graphs
colors = {'blue': '#0b6374', 'orange': '#c0791b', 'gray': '#666666', 'green': '#008000'}

# plot title font size
tt_fs = 20

# the layer height(mm) for the print was held constant
layer_height = 0.03

# %% [markdown]
# # Print Parameters (PP)

# %%
# read print parameters file
pp = pd.read_csv('print_parameters_DLR.csv', index_col=0)

# set column type to category
pp['cond_type'] = pp['cond_type'].astype('category')

# calculate energy densities in pp df
# LED = linear laser energy density
# GED = global energy density
# VED = volumetric laser energy density

# calculate LED
# LED = LP/SS
pp['led'] = (pp['laser_power(W)'] / pp['scan_speed(mm/s)']).round(2)

# calculate GED
# GED = LP/(SS*HS)
pp['ged'] = (pp['laser_power(W)'] / \
    (pp['scan_speed(mm/s)'] * pp['hatch_spacing(mm)'])).round(2)

# calculate VED
# VED = LP/(SS*HS*LH)
pp['ved'] = (pp['laser_power(W)'] / \
    (pp['scan_speed(mm/s)'] * pp['hatch_spacing(mm)'] * layer_height )).round(2)

print('PRINT PARAMETERS')
pp

# %% [markdown]
# # Material Properties (MP)

# %%
# Read Material Properties (MP) file
# Material properties file containing porosity

mp = pd.read_csv('porosity_DLR.csv', index_col=0)
print('MATERIAL PROPERTIES')
```

```
mp

# %% [markdown]
# # Final DataFrames for Analysis

# %% [markdown]
# ## Print Parameters (PP)

# %%
pp.info()

# %%
pp.describe()

# %% [markdown]
# ## Material Properties (MP)

# %%
mp.info()


# %%
mp.describe()

# %% [markdown]
# ## Combine (PP, MP) into a Single DF

# %%
# Combine pp, mpm, and mp into a single df

tables = [pp, mp]

combined_df = reduce(lambda  left,right: pd.merge(left, right, on=['condition'], how='outer'),
tables)

combined_df

# %% [markdown]
# # Data Story Telling

# %% [markdown]
# ## Print Parameter Plot

# %%

fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')

# Define colors for conditions
pp_colors = {'PV':colors.get('blue'), 'SV':colors.get('orange'), 'HV':colors.get('green')}
condition_colors = {'PV': 'blue', 'SV': 'magenta', 'HV': 'green'}

# Plot 3D scatter plot
for condition, color in condition_colors.items():
    subset = pp[pp['cond_type'] == condition]
    ax.scatter(subset['scan_speed(mm/s)'], subset['laser_power(W)'],
subset['hatch_spacing(mm)'],
               c=color, label=condition, s=100)

# Set labels and title
ax.set_xlabel('Scan Speed (mm/s)')
ax.set_ylabel('Laser Power (W)')
ax.set_zlabel('Hatch Spacing (mm)')
ax.set_title('PRINT PARAMETERS', fontsize=20)

# Set limits
ax.set_xlim(200, 1100)
```

```
ax.set_ylim(100, 200)
ax.set_zlim(0.04, 0.175)  # Assuming hatch spacing ranges between 0 and 10 mm

# Add legend
ax.legend(fontsize=15)

# Add straight lines parallel to each axis at given positions
ax.plot([600, 600], [175, 175], [0.04, 0.175], color='green', linestyle='--')
ax.plot([600, 600], [100, 200], [0.1, 0.1], color='blue', linestyle='--')
ax.plot([200, 1100], [175, 175], [0.1, 0.1], color='red', linestyle='--')
ax.set_box_aspect([2, 2, 2])

plt.show()

# %% [markdown]
# ## Material Property Plot

# %%
# Plot material properties, porosity vs. condition

sns.set_style("ticks")
plt.figure(figsize=(30, 10))
ax = sns.barplot(x=combined_df.index, # Horizontal Axis
                 y='porosity(%)', # Vertical Axis
                 data=combined_df, # Data Source
                 hue='cond_type', # Color Markers
                 palette=pp_colors) # Color Dictionary

_ = plt.title('MATERIAL PROPERTIES', fontsize=tt_fs*3)
_ = plt.ylim(0, 4)
_ = plt.legend(['Power Variation','Hatch Variation', 'Speed Variation'],
           title='Condition Type',
               loc='upper right',
               frameon=True,
               labelcolor=['orange','green','blue'],
               fontsize=25)
plt.xlabel('Condition', fontsize=20)
plt.ylabel('Porosity (%)', fontsize=20)

for index, row in combined_df.iterrows():
    x = index - 1
    y = row['porosity(%)']
    ax.text(x+0.1, y+0.1,
            str(round(y,2))+'%',
            ha='center',
            fontsize=15,
            rotation=30)


# %% [markdown]
# ## Create Dot Plot Function

# %%

# This function is to be reused to create dot graphs

def create_lmplot(plot_dict):

    sns.set_style("ticks")

    sns.lmplot(x=plot_dict['x_data'], # Horizontal Axis
               y=plot_dict['y_data'], # Vertical Axis
               data=plot_dict['table'], # Data Source
               hue=plot_dict['hue'], # Color Markers
               scatter_kws={'s': 100}, # Set Marker Size
               fit_reg=True, # Don't Fix a Regression Line
               legend=False # Remove Default Legend
```

```
            )
    plt.title(plot_dict['title'], fontsize=tt_fs)
    plt.xlim(plot_dict['x_lim'][0], plot_dict['x_lim'][1])
    plt.ylim(plot_dict['y_lim'][0], plot_dict['y_lim'][1])
    plt.legend(title=plot_dict['legend_title'],
               loc='center left',
               bbox_to_anchor=(1.0, 0., 1.0, 1.0)
              )

# %% [markdown]
# ## Porosity vs. Laser Power & Speed & Hatch Spacing

# %%
# plot laser power vs. porosity

mpm_pp_mp_pv = combined_df[combined_df['cond_type']=='PV']

sns.set_style("ticks")

_ = sns.lmplot(x='laser_power(W)', # Horizontal Axis
           y='porosity(%)', # Vertical Axis
           data=mpm_pp_mp_pv, # Data Source
           palette=pp_colors, # Color Dictionary
           scatter_kws={'s': 100}, # Set Marker Size
           fit_reg=True, # Don't Fix a Regression Line
           legend=False # Remove Default Legend
          )

_ = plt.title('Porosity vs. Laser Power', fontsize=tt_fs)
_ = plt.xlim(100, 200)
_ = plt.ylim(0, 4)

# %%
# plot scan speed vs. porosity

mpm_pp_mp_sv = combined_df[combined_df['cond_type']=='SV']

sns.set_style("ticks")

_ = sns.lmplot(x='scan_speed(mm/s)', # Horizontal Axis
           y='porosity(%)', # Vertical Axis
           data=mpm_pp_mp_sv, # Data Source
           palette=pp_colors, # Color Dictionary
           scatter_kws={'s': 100}, # Set Marker Size
           fit_reg=True, # Don't Fix a Regression Line
           legend=False # Remove Default Legend
          )

_ = plt.title('Porosity vs. Scan Speed', fontsize=tt_fs)
_ = plt.xlim(200, 1100)
_ = plt.ylim(0, 4)

# %%
# plot hatch spacing vs. porosity

mpm_pp_mp_sv = combined_df[combined_df['cond_type']=='HV']

sns.set_style("ticks")

_ = sns.lmplot(x='hatch_spacing(mm)', # Horizontal Axis
           y='porosity(%)', # Vertical Axis
           data=mpm_pp_mp_sv, # Data Source
           palette=pp_colors, # Color Dictionary
           scatter_kws={'s': 100}, # Set Marker Size
           fit_reg=True, # Don't Fix a Regression Line
           legend=False # Remove Default Legend
          )
```

```
_ = plt.title('Porosity vs. Hatch Spacing', fontsize=tt_fs)
_ = plt.xlim(0.04, 0.175)
_ = plt.ylim(0, 4)

# %% [markdown]
# # Correlation Plots

# %% [markdown]
# ## PP vs. MP

# %%
pp_mp = pd.concat([pp,mp], axis=1)
pp_mp.drop(columns=['cond_type'], inplace=True)

# Compute the correlation matrix
corr = pp_mp.corr()

# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(10,10))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
_ = sns.heatmap(corr, mask=mask, cmap=cmap, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5}, annot=True, fmt='.2f',
annot_kws={"size": 12})

# %% [markdown]
# # MODELING

# %%
model_df = combined_df.copy()
model_df['cond_type'] = model_df['cond_type'].map({'PV':0, 'HV':1, 'SV':2}) # can only contain
numerical values
model_df.loc[model_df['porosity(%)'] < 0, 'porosity(%)'] = 0 # technically porosity can't be
less than zero
model_df.head()

# %% [markdown]
# ## Import Libraries

# %%
from sklearn.model_selection import train_test_split
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import mean_squared_error, r2_score

# setup seaborn
sns.set_style("whitegrid")
sns.set_context("poster")

# %% [markdown]
# ## Data Preparation

# %%
# split data into train/test

# split data table into X and y
X = model_df.drop(['porosity(%)'], axis=1)
y = pd.DataFrame(model_df['porosity(%)'])

# split data into train and test
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.05, random_state=1)

# %% [markdown]
# ## Error: Root Mean Squared

# %%
# calculate the root mean squared error function

from sklearn.metrics import mean_squared_error
from math import sqrt

def calc_rmse(y_actual, y_predicted):
    print('Actual:', '\n', y_actual, '\n')
    print('Predicted:', '\n', y_predicted, '\n')
    rmse = sqrt(mean_squared_error(y_actual, y_predicted))
    print('Root Mean Squared Error: ', rmse, '\n')
    return rmse

# %% [markdown]
# ## Error: R-Squared

# %%
# calculate the r-squared error

from sklearn.metrics import r2_score

# %% [markdown]
# # Model 1: Linear Regression

# %%
from sklearn.linear_model import LinearRegression

# %%
# linear regression function

def lin_reg(X_tr, X_te, y_tr, y_te):

    # initialize a LinearRegression model
    lm = LinearRegression()

    # fit the model with the training data
    lm.fit(X_tr, y_tr)

    # record the feature coefficients
    feat_coeff = pd.DataFrame({'feature': X_tr.columns, 'coef': lm.coef_[0]})[['feature',
'coef']]

    print('Feature Coefficients')
    print(feat_coeff, '\n')

    # predict the output (y_test) with the X_test data
    y_pred_o = lm.predict(X_te)

    print('Predicted Porosity')
    print(y_pred_o, '\n')

    # if the model predicts a negative porosity, set the value to 0
    y_pred = y_pred_o.clip(min=0)

    print('Predicted Porosity (non-neg)')
    print(y_pred, '\n')

    return y_pred

# %% [markdown]
# ## Record All y Values
```

```
# %%
# table to collect all y values (both actual and predictions)
y_vals_lr = pd.DataFrame(y_test.copy())
# record rmse values
lr_rmse_ls = []
# record r-squared values
lr_r_sq_ls = []

# %%
# use all the features to create and test a linear regression model

y_pred = lin_reg(X_train, X_test, y_train, y_test)

# %%
# calculate the rmse
lr_rmse_ls.append(calc_rmse(y_test['porosity(%)'], y_pred))
# record the predicted y values
y_vals_lr['all_features'] = y_pred

# %%
# calculate the r-squared value

r_sq = r2_score(y_test, y_pred)
lr_r_sq_ls.append(r_sq)
print('The R-Squared Value is', round(r_sq,3))

# %% [markdown]
# ## Comparison

# %%
# df to compare all the y output values
y_vals_lr

# %%
# melt the table so it can be plotted
# make a temp df for the plot
temp = y_vals_lr.copy()
# convert the index (labels) into a column
temp.reset_index(inplace=True)
# melt the df into labels > campaign > value
temp = pd.melt(temp, id_vars=['condition'])


# %%
# plot of porosity vs. condition (linear scale)

_ = sns.barplot(x='condition',
                y='value',
                data=temp,
                hue='variable')

_ = plt.title('Porosity vs. Condition')
_ = plt.xlabel('Condition', fontsize=16)
_ = plt.ylabel('Porosity(%)', fontsize=16)
_ = plt.ylim( (0,2) )
_ = plt.yscale('linear')
_ = plt.legend(loc=3, bbox_to_anchor=(1, 0.0), fontsize = 14, title='Actual vs. Features')

# %%
# plot of porosity vs condition (log scale)

from matplotlib.ticker import ScalarFormatter, NullFormatter

fig = plt.figure()
ax1 = fig.add_subplot(111)

ax1 = sns.barplot(x='condition',
```

```
                y='value',
                data=temp,
                hue='variable')
ax1.set_title('Porosity vs. Condition')
ax1.set_xlabel('Condition', fontsize=16)
ax1.set_ylabel('Porosity(%)', fontsize=16)

ax1.set_ylim(10**-2,10**1)
ax1.set_yticks([10**x for x in range(-2,2,1)])
ax1.set_yscale('log', subs=[2])
ax1.legend(loc=3, bbox_to_anchor=(1.2, 0.0), fontsize = 14, title='Actual vs. Features')

ax2 = ax1.twinx()

ax2 = sns.barplot(x='condition',
                y='value',
                data=temp,
                hue='variable')
ax2.set_ylabel('', fontsize=16)
ax2.legend_.remove()

ax2.set_ylim(10**-2,10**1)
ax2.set_yscale('log', subs=[])
ax2.axes.yaxis.set_ticklabels([])
ax2.grid(False)
ax2.axes.yaxis.set_major_formatter(ScalarFormatter())
ax2.set_yticks([0.01, 0.1, 0.5, 1, 2, 5, 10])
ax2.yaxis.set_minor_formatter(NullFormatter())

# %%
# plot rmse values

lr_error_dict = {'data': ['all features'],
             'rmse':lr_rmse_ls,
             'r_squared':lr_r_sq_ls}
lr_error_df = pd.DataFrame.from_dict(lr_error_dict)
lr_error_df

# %% [markdown]
# # Model 2: Random Forest

# %%
from sklearn.ensemble import RandomForestRegressor

# %%
# random forest function

def rand_for(X_tr, X_te, y_tr, y_te, n_tree):

    # initialize a LinearRegression model
    rf = RandomForestRegressor(n_estimators=n_tree,
                               random_state=1,
                               bootstrap=True)

    # fit the model with the training data
    rf.fit(X_tr, y_tr.values.ravel())

    # predict the output (y_test) with the X_test data
    y_pred_o = rf.predict(X_te)

    print('Predicted Porosity')
    print(y_pred_o, '\n')

    # if the model predicts a negative porosity, set the value to 0
    y_pred = y_pred_o.clip(min=0)

    print('Predicted Porosity (non-neg)')
```

```
    print(y_pred, '\n')

    # Print the feature ranking
    print("Feature Ranking:")

    fi = pd.DataFrame(rf.feature_importances_, index=X_tr.columns,
columns=['importance']).sort_values('importance', ascending=False)
    feature_importances = fi[fi.importance != 0]
    print(feature_importances)

    print('------------------')

    return y_pred

# %%
# create random forest models with n number of trees

def n_rand_for(X_tr, X_te, y_tr, y_te, n_min, n_max):

    # number of trees used in the random forest model
    trees = [x for x in range(n_min, n_max,1)]

    for n_tree in trees:
        # for loop to create random forest models with varying number of trees
        print('tree =', n_tree)

        # create a random forest model with n number of trees
        y_pred = rand_for(X_tr, X_te, y_tr, y_te, n_tree)

        # create column label
        if n_tree == 1:
            col = str(n_tree) + ' tree'
        else:
            col = str(n_tree) + ' trees'

        # record predicted y values in df
        y_vals_rf[col] = y_pred

        # calculate the rmse value
        rf_rmse_ls.append([n_tree, calc_rmse(y_test['porosity(%)'], y_pred)])

        # calculate the r_squared value
        r_sq = r2_score(y_te, y_pred)
        rf_r_sq_ls.append([n_tree, r_sq])

# %%
# table to collect all y values (both actual and predictions)
y_vals_rf = pd.DataFrame(y_test.copy())
# record rmse values
rf_rmse_ls = []
# record r_squared values
rf_r_sq_ls = []

# %% [markdown]
# ## All features

# %%
# new nested function
n_rand_for(X_train, X_test, y_train, y_test, 1, 11)

# %%
y_vals_rf

# %%
# number of trees vs rmse
rf_rmse_df = pd.DataFrame(rf_rmse_ls, columns=['n_trees', 'rmse'])
rf_rmse_df
```

```
# %%
# number of trees vs r_squared
rf_r_sq_df = pd.DataFrame(rf_r_sq_ls, columns=['n_trees', 'r_sq'])
rf_r_sq_df

# %%
# melt the table so it can be plotted
# make a temp df for the plot
temp = y_vals_rf.copy()
# convert the index (labels) into a column
temp.reset_index(inplace=True)
# melt the df into labels > campaign > value
temp = pd.melt(temp, id_vars=['condition'])

# %%
# plot of porosity vs. condition (linear scale)
fig = plt.figure(figsize=(10, 5))
_ = sns.barplot(x='condition',
                y='value',
                data=temp,
                hue='variable')

_ = plt.title('Porosity vs. Condition')
_ = plt.xlabel('Condition', fontsize=16)
_ = plt.ylabel('Porosity(%)', fontsize=16)
_ = plt.xticks(fontsize = 14)
_ = plt.ylim((0,2))
_ = plt.yscale('linear')
_ = plt.legend(loc=3, bbox_to_anchor=(1, 0.0), fontsize = 14, title='Actual vs. Trees')

# %%
# plot porosity vs condition (log scale)

from matplotlib.ticker import ScalarFormatter, NullFormatter

fig = plt.figure(figsize=(10, 5))
ax1 = fig.add_subplot(111)

ax1 = sns.barplot(x='condition',
                y='value',
                data=temp,
                hue='variable')

ax1.set_title('Porosity vs. Condition')
ax1.set_xlabel('Condition', fontsize=16)
ax1.set_ylabel('Porosity(%)', fontsize=16)
ax1.set_ylim(10**-2,10**1)
ax1.set_yticks([10**x for x in range(-2,2,1)])
ax1.set_yscale('log', subs=[2])
ax1.legend(loc=3, bbox_to_anchor=(1.2, 0.0), fontsize = 14, title='Actual vs. Trees')

ax2 = ax1.twinx()

ax2 = sns.barplot(x='condition',
                y='value',
                data=temp,
                hue='variable')

ax2.set_ylabel('', fontsize=16)
ax2.legend_.remove()
ax2.set_ylim(10**-2,10**1)
ax2.set_yscale('log', subs=[])
ax2.axes.yaxis.set_ticklabels([])
ax2.grid(False)
ax2.axes.yaxis.set_major_formatter(ScalarFormatter())
ax2.set_yticks([0.01, 0.1, 0.5, 1, 2, 5, 10])
```

```
ax2.yaxis.set_minor_formatter(NullFormatter())

# %%
rf_rmse_r_sq = rf_rmse_df.merge(rf_r_sq_df, on='n_trees')
# melt the table so it can be plotted
# make a temp df for the plot
temp = rf_rmse_r_sq.copy()
# melt the df into labels > campaign > value
temp = pd.melt(temp, id_vars=['n_trees'])

# %%
# RMSE of the random forest per # of trees

_ = sns.barplot(x='n_trees',
                y='value',
                data=temp,
                hue='variable')

_ = plt.title('RMSE & R-Squared vs. # of Trees', y=1.05)
_ = plt.xlabel('# of Trees', fontsize=16)
_ = plt.ylabel('RMSE / R-Squared', fontsize=16)
_ = plt.ylim(0,1)
_ = plt.legend(loc=3, bbox_to_anchor=(1, 0.0), title='Error')

# %% [markdown]
# # Best Models

# %% [markdown]
# ## Best Linear Regression

# %%
# create model and display feature coefficient

# initialize a LinearRegression model
lm = LinearRegression()

# fit the model with the training data
lm.fit(X_train, y_train)

# record the feature coefficients
feat_coeff = pd.DataFrame({'Feature': X_train.columns, 'Coefficient': lm.coef_[0]})
[['Feature', 'Coefficient']]

feat_coeff = feat_coeff.sort_values('Coefficient', ascending=False)

feat_coeff

# %%
# plot feature coefficients

_ = sns.barplot(x='Feature',
                y='Coefficient',
                data=feat_coeff,
                hue='Feature')

_ = plt.title('Coefficient vs. Feature', y=1.05)
_ = plt.xlabel('Feature', fontsize=16)
_ = plt.ylabel('Coefficient', fontsize=16)
_ = plt.xticks(rotation=45, fontsize=12)


# %% [markdown]
# ## Best Random Forest Model

# %%
# create model and display feature importance
```

```
# intialize model
rf = RandomForestRegressor(n_estimators=3,
                           random_state=1,
                           bootstrap=True)

# fit model
rf = rf.fit(X_train, y_train.values.ravel())

fi = pd.DataFrame(rf.feature_importances_,
                  index=X_train.columns,
                  columns=['Importance']).sort_values('Importance',
                                                      ascending=False)
feature_importances = fi[fi.Importance != 0]

feature_importances['Feature'] = feature_importances.index
feature_importances = feature_importances.reset_index(drop=True)
feature_importances

# %%
# plot feature importance
plt.figure(figsize=(12, 6))

_ = sns.barplot(x='Feature',
                y='Importance',
                data=feature_importances,
                hue='Feature')

_ = plt.title('Importance vs. Feature', y=1.05, fontsize=20)
_ = plt.xlabel('Feature', fontsize=16)
_ = plt.ylabel('Importance', fontsize=16)
_ = plt.xticks(rotation=30, fontsize=16)
_ = plt.ylim(10**-3,1)
_ = plt.yscale('log', subs=[2])

# %% [markdown]
# ## Display Tree Model

# %%
model = RandomForestRegressor(n_estimators=3, random_state=1)

# Train
model.fit(X_train, y_train)
# Extract single tree
estimator = model.estimators_[0]

plt.figure(figsize=(30, 10))
plot_tree(estimator, feature_names=list(X_train.columns), filled=True, rounded=True,
fontsize=15, precision=2)
plt.show()

# %% [markdown]
# # Model 3: Gaussian Process Regression

# %% [markdown]
# Define Expected Improvement for a prediction

# %%
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF, Matern
from sklearn.preprocessing import StandardScaler

plt.rcParams.update({
    'font.size': 8,          # Main font size
    'axes.labelsize': 8,     # Label size for x and y axes
    'axes.titlesize': 8,     # Title size
    'xtick.labelsize': 8,     # Label size for x-axis ticks
    'ytick.labelsize': 8,     # Label size for y-axis ticks
```

```
    'legend.fontsize': 8,      # Legend font size
    'figure.titlesize': 30     # Title size for the whole figure
    })

# %%
Data_y = np.array(mp['porosity(%)'])
Data_x = np.array(pp[['laser_power(W)', 'scan_speed(mm/s)', 'hatch_spacing(mm)']])
np.random.seed(0)
scaler = StandardScaler()
Data_x_standardized = scaler.fit_transform(Data_x)

# %%
import itertools

# Define the range of nu values and length scale parameters to try
nu_values = np.linspace(0.1, 3.0, 12)
length_scale_values = np.linspace(0.1, 3.0, 12)

# Generate combinations of length scale parameters
length_scale_combinations = list(itertools.product(length_scale_values, repeat=3))

# Initialize variables to store the best model and minimum RMSE
best_rmse = float('inf')
best_model = None
best_nu = None
best_length_scale = None
best_y_pred = None

# Standardize the input data
scaler = StandardScaler()
Data_x_standardized = scaler.fit_transform(Data_x)

# Split the data into training and testing sets
X_G_train, X_G_test, y_G_train, y_G_test = train_test_split(Data_x_standardized, Data_y,
test_size=0.25, random_state=42)
y_G_pred = np.zeros_like(y_G_test)

# Loop through each nu value and length scale parameter combination
for nu in nu_values:
    for length_scale in length_scale_combinations:
        # Define the Matern kernel with the current nu value and length scale parameters
        kernel = Matern(length_scale=length_scale, nu=nu)

        # Fit the Gaussian Process Regression model
        gpr_model = GaussianProcessRegressor(kernel=kernel, optimizer=None,
normalize_y=False).fit(X_G_train, y_G_train)

        # Make predictions on the testing set
        y_G_pred = gpr_model.predict(X_G_test)

        # Calculate RMSE
        rmse = np.sqrt(mean_squared_error(y_G_test, y_G_pred))

        # Check if this model has the lowest RMSE so far
        if rmse < best_rmse:
            best_rmse = rmse
            best_model = gpr_model
            best_nu = nu
            best_length_scale = length_scale
            best_y_pred = y_G_pred

# Output the best nu value, length scale, and the corresponding minimum RMSE
print("Best nu value:", best_nu)
print("Best length scale:", best_length_scale)
print("Minimum RMSE:", best_rmse)
r_sq_G = r2_score(y_G_test, best_y_pred)
print(r_sq_G)
```

```
# %%
# Generate combinations of length scale parameters
length_scale_combinations = list(itertools.product(length_scale_values, repeat=3))

# Initialize variables to store the best model and minimum RMSE
best_rmse_RBF = float('inf')
best_model_RBF = None
best_nu_RBF = None
best_length_scale_RBF = None
y_G_pred_RBF = np.zeros_like(y_G_test)
best_y_pred_RBF = None

# Loop through each nu value and length scale parameter combination
for nu in nu_values:
    for length_scale in length_scale_combinations:
        # Define the RBF kernel with the current length scale parameters
        kernel = nu * RBF(length_scale=length_scale)

        # Fit the Gaussian Process Regression model
        gpr_model = GaussianProcessRegressor(kernel=kernel, optimizer=None,
normalize_y=False).fit(X_G_train, y_G_train)

        # Make predictions on the testing set
        y_G_pred_RBF = gpr_model.predict(X_G_test)

        # Calculate RMSE
        rmse = np.sqrt(mean_squared_error(y_G_test, y_G_pred))

        # Check if this model has the lowest RMSE so far
        if rmse < best_rmse_RBF:
            best_rmse_RBF = rmse
            best_model_RBF = gpr_model
            best_nu_RBF = nu
            best_length_scale_RBF = length_scale
            best_y_pred_RBF = y_G_pred_RBF

# Output the best nu value, length scale, and the corresponding minimum RMSE
print("Best nu value:", best_nu_RBF)
print("Best length scale:", best_length_scale_RBF)
print("Minimum RMSE:", best_rmse_RBF)
r_sq_G_RBF = r2_score(y_G_test, best_y_pred_RBF)
print(r_sq_G_RBF)

# %%
def Generate_Contours(n, gprMdl1, scaler):
    power = np.linspace(100, 200, n)
    speed = np.linspace(200, 1100, n)
    hatch = np.linspace(0.04, 0.175, n)
    z_1 = 0.50
    z_2 = 0.75

    hatchC = 0.1

    Porosity_H = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            Porosity_H[i, j] = gprMdl1.predict(scaler.transform([[power[i], speed[j],
hatchC]]))

    plt.figure()
    contour = plt.contourf(speed, power, Porosity_H, levels=20, cmap='viridis')
    plt.contour(speed, power, Porosity_H, levels=[z_1, z_2], colors='k', linewidths=2)
    plt.colorbar(contour, label='Porosity (%)')
    plt.xlim([200,1100])
    plt.ylim([100,200])
    plt.xlabel('Speed (mm/s)')
```

```
    plt.ylabel('Power (W)')
    plt.title('Porosity at constant hatch space 100 µm', fontsize=20)
    plt.grid(True)

    speedC = 600

    Porosity_S = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            Porosity_S[i, j] = gprMdl1.predict(scaler.transform([[power[i], speedC,
hatch[j]]]))

    plt.figure()
    contour = plt.contourf(power, hatch, Porosity_S, levels=20, cmap='viridis')
    plt.contour(power, hatch, Porosity_S, levels=[z_1, z_2], colors='k', linewidths=2)
    plt.colorbar(contour, label='Porosity (%)')
    plt.xlim([100,200])
    plt.ylim([0.04,0.175])
    plt.xlabel('Power (W)')
    plt.ylabel('Hatch Space (µm)')
    plt.title('Porosity at constant speed 600 mm/s', fontsize=20)
    plt.grid(True)

    powerC = 175

    Porosity_P = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            Porosity_P[i, j] = gprMdl1.predict(scaler.transform([[powerC, speed[i],
hatch[j]]]))

    plt.figure()
    contour = plt.contourf(speed, hatch, Porosity_P, levels=20, cmap='viridis')
    plt.contour(speed, hatch, Porosity_P, levels=[z_1, z_2], colors='k', linewidths=2)
    plt.colorbar(contour, label='Porosity (%)')
    plt.xlim([200,1100])
    plt.ylim([0.04,0.175])
    plt.xlabel('Speed (mm/s)')
    plt.ylabel('Hatch Space (µm)')
    plt.title('Porosity at constant power 175 W', fontsize=20)
    plt.grid(True)


    plt.show()


# %%
kernel1 = Matern(length_scale=best_length_scale, nu=best_nu)
gprMdl1 = GaussianProcessRegressor(kernel=kernel1, optimizer=None, normalize_y =
False).fit(Data_x_standardized, Data_y)
print('score', gprMdl1.score(Data_x_standardized, Data_y))

import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
n=300
Generate_Contours(n, gprMdl1, scaler)

# %%
def plots2D_Power(n, gprMdl, scaler, X, Y):

    power = np.linspace(100, 200, n)
    S_C = 600
    H_C = 0.1

    # Plot for P vs Porosity_P
    Porosity_P = np.zeros(n)
    Sigma_P = np.zeros(n)
```

```
    for i in range(n):
        Porosity_P[i], Sigma_P[i] = gprMdl.predict(scaler.transform([[power[i], S_C, H_C]]),
return_std=True)
    plt.figure(figsize=(8, 6))
    plt.plot(power, Porosity_P, 'k', label='Predicted Mean')
    plt.scatter(X[20:29, 0], Y[20:29], c="r", label = 'Training Data')
    plt.fill_between(power, Porosity_P - 1.96 * np.sqrt(Sigma_P), Porosity_P + 1.96 *
                np.sqrt(Sigma_P), alpha=0.2, color='k', label='95% Confidence Interval')
    plt.title('Power vs Porosity', fontsize=20)
    plt.xlabel('Power (W)')
    plt.ylabel('Porosity (%)')
    plt.legend()
    plt.grid()

plots2D_Power(n, gprMdl1, scaler, Data_x, Data_y)

# %%
def plots2D_Speed(n, gprMdl, scaler, X, Y):

    speed = np.linspace(200, 1100, n)
    P_C = 175
    H_C = 0.1

    # Plot for S vs Porosity_S
    Porosity_S = np.zeros(n)
    Sigma_S = np.zeros(n)
    for i in range(n):
        Porosity_S[i], Sigma_S[i] = gprMdl.predict(scaler.transform([[P_C, speed[i], H_C]]),
return_std=True)
    plt.figure(figsize=(8, 6))
    plt.plot(speed, Porosity_S, 'k', label='Predicted Mean')
    plt.scatter(X[:10, 1], Y[:10], c="r", label = 'Training Data')
    plt.fill_between(speed, Porosity_S - 1.96 * np.sqrt(Sigma_S), Porosity_S + 1.96 *
                np.sqrt(Sigma_S), alpha=0.2, color='k', label='95% Confidence Interval')
    plt.title('Speed vs Porosity', fontsize=20)
    plt.xlabel('Speed(mm/s)')
    plt.ylabel('Porosity(%)')
    plt.legend()
    plt.grid()

plots2D_Speed(n, gprMdl1, scaler, Data_x, Data_y)

# %%
def plots2D_Hatch(n, gprMdl, scaler, X, Y):

    hatch = np.linspace(0.04, 0.175, n)
    P_C = 175
    S_C = 600

    # Plot for H vs Porosity_H
    Porosity_H = np.zeros(n)
    Sigma_H = np.zeros(n)
    for i in range(n):
        Porosity_H[i], Sigma_H[i] = gprMdl.predict(scaler.transform([[P_C, S_C, hatch[i]]]),
return_std=True)
    plt.figure(figsize=(8, 6))
    plt.plot(hatch, Porosity_H, 'k', label='Predicted Mean')
    plt.scatter(X[10:19, 2], Y[10:19], c="r", label = 'Training Data')
    plt.fill_between(hatch, Porosity_H - 1.96 * np.sqrt(Sigma_H), Porosity_H + 1.96 *
                np.sqrt(Sigma_H), alpha=0.2, color='k', label='95% Confidence Interval')
    plt.title('Hatch Spacing vs Porosity', fontsize=20)
    plt.xlabel('Hatch Spacing(mm)')
    plt.ylabel('Porosity(%)')
    plt.legend()
    plt.grid()

plots2D_Hatch(n, gprMdl1, scaler, Data_x, Data_y)
```

```
# %%
import itertools

# Define the range of nu values and length scale parameters to try
nu_values = np.linspace(0.1, 3.0, 12)
length_scale_values = np.linspace(0.1, 3.0, 12)

# Generate combinations of length scale parameters
length_scale_combinations = list(itertools.product(length_scale_values, repeat=3))

# Initialize variables to store the best model and minimum RMSE
best_rmse = float('inf')
best_model = None
best_nu = None
best_length_scale = None
best_y_pred = None

# Standardize the input data
scaler = StandardScaler()
Data_x_standardized = scaler.fit_transform(Data_x)

# Split the data into training and testing sets
X_G_train, X_G_test, y_G_train, y_G_test = train_test_split(Data_x_standardized, Data_y,
test_size=0.25, random_state=42)
y_G_pred = np.zeros_like(y_G_test)

# Loop through each nu value and length scale parameter combination
for nu in nu_values:
    for length_scale in length_scale_combinations:
        # Define the Matern kernel with the current nu value and length scale parameters
        kernel = Matern(length_scale=length_scale, nu=nu)

        # Fit the Gaussian Process Regression model
        gpr_model = GaussianProcessRegressor(kernel=kernel, optimizer=None,
normalize_y=False).fit(X_G_train, y_G_train)

        # Make predictions on the testing set
        y_G_pred = gpr_model.predict(X_G_test)

        # Calculate RMSE
        rmse = np.sqrt(mean_squared_error(y_G_test, y_G_pred))

        # Check if this model has the lowest RMSE so far
        if rmse < best_rmse:
            best_rmse = rmse
            best_model = gpr_model
            best_nu = nu
            best_length_scale = length_scale
            best_y_pred = y_G_pred

# Output the best nu value, length scale, and the corresponding minimum RMSE
print("Best nu value:", best_nu)
print("Best length scale:", best_length_scale)
print("Minimum RMSE:", best_rmse)
r_sq_G = r2_score(y_G_test, best_y_pred)
print(r_sq_G)



# %%
from sklearn.gaussian_process.kernels import RBF, ConstantKernel as C

y_G_pred = np.zeros_like(y_G_test)
lbound = 1e-2
rbound = 1e1
n_restarts = 5000
```

```
n_features = 3 # Actually determined elsewhere in the code

# Define the initial Matern kernel with the given parameters
initial_length_scale = (0.6272727272727272, 1.9454545454545453, 3.0)
initial_nu = 3.0
# Bounds for length_scale
length_scale_bounds = (lbound, rbound)
# Constant kernel multiplied by RBF
kernel = C(1.0, (1e-2, 1e2)) * Matern(length_scale=initial_length_scale,
length_scale_bounds=length_scale_bounds, nu=initial_nu)

gp_new = GaussianProcessRegressor(kernel=kernel, n_restarts_optimizer=n_restarts)
gp_new.fit(X_G_train, y_G_train)
y_G_pred = gp_new.predict(X_G_test)
rmse = np.sqrt(mean_squared_error(y_G_test, y_G_pred))
print("Minimum RMSE:", rmse)
```