Development of a Lexicographic Multi-Objective Optimization-Based Heterogeneous Ensemble Regression Analysis: A Case of Cost-Benefit Analysis of a Risk Control Measure in a Steel Plant

BACHELOR TERM PROJECT REPORT

By

AJEET KUMAR SHUKLA
(20MF3IM02)

Under the supervision of

Prof. J. MAITI
Department of Industrial and Systems Engineering, IIT Kharagpur

Department of Industrial and Systems Engineering
Indian Institute of Technology Kharagpur
West Bengal, India
28th November,2023

DECLARATION

I certify that

a. The work contained in this report has been done by me under the guidance of my supervisor.

b. The work has not been submitted to any other Institute for any degree or diploma.

c. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

d. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

Date:28/11/2023	Name: Ajeet Kumar Shukla
Place: Kharagpur	Roll number: 20MF3IM02

DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR
KHARAGPUR-721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Development of a Lexicographic Multi-Objective Optimization-Based Heterogeneous Ensemble Regression Analysis: A Case of Cost-Benefit Analysis of a Risk Control Measure in a Steel Plant**" submitted by **Ajeet Kumar Shukla (20MF3IM02)** to Indian Institute of Technology, Kharagpur towards partial fulfillment of requirements for the award of the degree of **Dual Bachelor of Technology (Hons)** in **Manufacturing Science and Engineering** is a record of bona fide work carried out by him under my supervision and guidance during **Autumn Semester 2023.**

Date:28/11/2023
Place: Kharagpur

Prof J Maiti
Department of Industrial and Systems Engineering
Indian Institute of Technology, Kharagpur
Kharagpur, India

CONTENTS

1. Introduction

Cost-benefit analysis constitutes a crucial process for industries aiming to optimize operations, ensuring economic viability, minimizing environmental impact, and prioritizing worker safety. Within this framework, the development of a Lexicographic Multi-Objective Optimization-Based Heterogeneous Ensemble Regression Analysis emerges as a compelling avenue, particularly within the context of the steel plant industry. This introduction aims to provide comprehensive insights into the significance of the topic, its background, identified gaps in recent literature, and the unique contributions of this research endeavor.

In the relentless pursuit of economic viability and operational efficiency, industries, notably steel plants, grapple with intricate challenges. The steel manufacturing process, characterized by heavy machinery, extreme temperatures, and numerous variables, inherently carries inherent safety and environmental risks. Addressing these challenges necessitates a pivotal role for cost-benefit analysis. Implementing safety measures, optimizing operational processes, and investing in environmental initiatives mandate a thorough understanding of their economic implications. The proposed development of a Lexicographic Multi-Objective Optimization-Based Heterogeneous Ensemble Regression Analysis offers a novel and sophisticated approach to facilitate cost-benefit analysis in the unique context of steel plants. Leveraging this methodology empowers industries to make data-driven decisions that adeptly balance the triple bottom line: profits, safety, and sustainability.

As a cornerstone of modern infrastructure and manufacturing, the steel plant industry perpetually seeks innovative methods to enhance its operations. Over the years, cost-benefit analysis has proven indispensable for decision-makers in the sector. Traditional approaches involve calculating benefit-cost ratios, which, while valuable, often fall short in effectively comparing diverse designs and strategies. In this context, the proposed Lexicographic Multi-Objective Optimization-Based Heterogeneous Ensemble Regression Analysis stands out as a contemporary solution to address the limitations of conventional methods. By integrating multi-objective optimization, ensemble regression techniques, and a lexicographic approach, this research aims to provide a more comprehensive understanding of cost-benefit trade-offs within the intricate landscape of steel plant operations.

Despite the overarching significance of cost-benefit analysis in industrial decision-making, recent literature reveals certain gaps that necessitate focused attention. Existing methodologies tend to oversimplify the complexities inherent in real-world industrial systems. While traditional cost-benefit analysis provides a valuable framework, it may struggle to accommodate the intricacies specific to the steel plant industry and its multifaceted objectives. Consequently, there arises a compelling need for a more nuanced and data-driven approach aligning with the industry's specific requirements. This research endeavors to bridge these gaps by introducing a novel framework that intricately accounts for the dynamics inherent in a steel plant's economic, safety, and environmental objectives.

This research not only addresses critical gaps in the existing literature but also makes significant contributions to the fields of industrial decision-making, safety enhancement, and sustainability within steel manufacturing. Through the development of a Lexicographic Multi-Objective Optimization-Based Heterogeneous Ensemble Regression Analysis, this study pioneers an innovative methodology poised to serve as a blueprint for industries seeking to adeptly balance cost-effectiveness, safety, and environmental impact. The outcomes of this research promise to provide a robust model for real-time cost-benefit analysis in steel plants, enabling more informed and data-driven decisions. Furthermore, it offers a fresh and impactful perspective on navigating the complexities inherent in modern industrial processes, influencing the adoption of safer and more sustainable practices in steel manufacturing.

2. Literature Review

The literature review presents a comprehensive exploration of diverse research endeavors aimed at addressing complex challenges in various industries, with a primary focus on optimizing operations in the steel plant industry. The first section underscores the significance of lexicographic multi-objective optimization-based heterogeneous ensemble regression analysis as a tool for conducting cost-benefit analysis in the steel plant industry. Drawing insights from a range of studies, including those on fire safety engineering, resilience enhancement in adverse weather conditions, data center security, and risk-based cost-benefit analysis for offshore Resource Centers, the review highlights the versatility of the proposed methodology. The authors offer a nuanced understanding of the interconnected objectives in the steel plant industry and allied fields, providing valuable insights for decision-makers seeking a balanced approach to economic viability, environmental sustainability, and worker safety.

The subsequent section delves into the UK External Vehicle Speed Control (EVSC) project, offering a detailed examination of its comprehensive analysis on accident savings and cost-benefit considerations related to Intelligent Speed Adaptation (ISA) implementation. The project's predictions, methodology, and recommended implementation path underscore the potential of ISA systems to significantly reduce injury and fatal accidents, providing a valuable roadmap for road safety improvement. The authors (O.M.J. Carsten et al., 2005) contribute to the literature by shedding light on the impactful integration of advanced technological interventions in addressing road safety concerns.

Moving forward, the literature review delves into a study assessing the costs and benefits of a community injury prevention program in Motala, Sweden. Through a quasi-experimental design and meticulous data recording, the authors (K Lindqvist et al., 2001) demonstrate the cost-effectiveness of the program, emphasizing the economic advantages and societal benefits associated with proactive safety interventions. This research enriches the literature on community-based injury prevention strategies, offering valuable insights for decision-makers in public health.

The final sections explore innovative approaches in optimization, introducing novel methodologies for addressing prioritized objectives in evolutionary algorithms (Leonardo Lai et al. 2020) and developing a multi-objective mathematical model for optimizing the grinding process (Soheyl Khalilpourazari et al., 2017). These studies contribute to the literature by providing flexible and effective strategies for tackling complex decision-making scenarios, enhancing the efficiency of optimization processes in various domains. The literature review, collectively, showcases the diverse applications of advanced methodologies in addressing multifaceted challenges across industries.

3. Preliminaries

Machine learning is a dynamic and rapidly evolving field within artificial intelligence, where algorithms and models are designed to enable computer systems to learn from data, improve their performance, and make predictions or decisions without explicit programming. By harnessing the power of data, machine learning applications have permeated diverse domains, from healthcare and finance to natural language processing and image recognition, offering insights, automation, and solutions to complex problems. As it continues to advance, machine learning holds the promise of transforming industries, enhancing decision-making processes, and opening new frontiers in technology and data-driven innovation. Some prominent machine-learning techniques include:

Supervised Learning: Supervised learning represents a foundational paradigm within the realm of machine learning, characterized by its ability to train algorithms through labeled data for predictive modeling. In this approach, an algorithm harnesses a dataset replete with input-output pairs to discern underlying patterns and relationships. The crux of supervised learning lies in its proficiency to learn a mapping function that enables the algorithm to predict outputs for new, unseen data. This process entails exposing the algorithm to a training dataset wherein each input is associated with a corresponding output, effectively providing a roadmap for the algorithm to understand and generalize from the provided examples.

The effectiveness of supervised learning hinges on the quality and diversity of the labeled dataset. A well-constructed training set facilitates the algorithm in discerning intricate patterns, capturing the nuances of the underlying data distribution. This process involves iteratively adjusting the algorithm's parameters to minimize the disparity between its predictions and the actual outcomes in the training data. The iterative nature of this learning process allows the algorithm to continuously refine its predictive capabilities, enhancing its adaptability to diverse and complex datasets.

Supervised learning is versatile and finds applications in various domains, from image and speech recognition to natural language processing and predictive analytics. Classification and regression are two primary types of supervised learning tasks. In classification, the algorithm learns to assign inputs to predefined categories or classes, while regression involves predicting numerical values based on input variables. The choice between classification and regression depends on the nature of the problem and the desired output.

In conclusion, supervised learning serves as a cornerstone in machine learning, leveraging labeled data to empower algorithms with predictive capabilities. Its adaptability to diverse domains and applications underscores its significance in modern data-driven problem-solving. As machine learning continues to evolve, supervised learning remains an essential tool for creating intelligent systems capable of making informed predictions in a wide array of scenarios.

Unsupervised Learning: Unsupervised learning stands as a fundamental machine learning approach distinguished by its unique ability to analyze and extract patterns from unlabeled datasets, in stark contrast to supervised learning, which heavily relies on labeled output for training. In the realm of unsupervised learning, algorithms embark on the exploration of unannotated data, making it particularly well-suited for scenarios where obtaining labeled data is impractical or cost-prohibitive. The primary objective of unsupervised learning is to unveil intrinsic structures, relationships, or hidden patterns within the data, contributing to a more profound understanding of the underlying information.

This approach is particularly valuable for tasks such as clustering and dimensionality reduction. Clustering involves grouping similar data points together based on inherent similarities, leading to the discovery of natural divisions within the data. On the other hand, dimensionality reduction aims to streamline complex datasets by capturing essential features and reducing the number of variables. These unsupervised learning techniques empower data scientists and researchers to gain insights into the underlying structure of the data, paving the way for enhanced decision-making and problem-solving.

Unsupervised learning has proven its efficacy in diverse domains, including but not limited to image and speech processing, anomaly detection, and recommendation systems. Its versatility lies in its ability to autonomously uncover patterns within data without the need for labeled examples. This flexibility makes unsupervised learning a valuable tool in scenarios where the true nature of the data is not fully understood or when exploring novel datasets where labeled information is scarce.

In essence, unsupervised learning stands as a powerful and exploratory tool in the machine learning arsenal, offering valuable insights into the inherent structures and relationships within datasets. Its application in various fields underscores its role in uncovering hidden knowledge and facilitating a deeper comprehension of complex datasets. As machine learning continues to advance, unsupervised learning remains integral to extracting meaningful information from the vast sea of unlabeled data.

Deep Learning: Deep learning emerges as a specialized subfield within the expansive domain of machine learning, leveraging the potency of artificial neural networks characterized by multiple layers, commonly referred to as deep neural networks. Unlike traditional machine learning methods, which may rely on shallow architectures, deep learning excels in handling intricate and high-dimensional data, making it particularly well-suited for applications requiring nuanced pattern recognition and extraction. The hallmark of deep learning lies in its ability to autonomously learn hierarchical representations from vast datasets, empowering it to discern complex features and relationships.

Central to the success of deep learning is the architecture of deep neural networks, which consist of layers of interconnected nodes or neurons. Each layer progressively refines and abstracts information, allowing the network to capture hierarchical representations of the input data. The learning process involves adjusting the weights and biases of these connections through iterative exposure to labeled data, a technique known as backpropagation. This iterative learning process enables deep neural networks to adapt and optimize their parameters, ultimately enhancing their predictive capabilities.

Deep learning finds particular prominence in tasks demanding advanced pattern recognition, such as image and speech recognition, natural language processing, and complex data analysis. In image recognition, for example, deep learning models can automatically identify and classify objects within images, surpassing traditional methods in accuracy and efficiency. Similarly, in speech recognition, deep learning excels at converting spoken language into text, facilitating the development of virtual assistants and voice-controlled systems. Its application extends to language processing, where deep learning models can comprehend and generate human-like text, contributing to advancements in machine translation and chatbot functionalities.

In summary, deep learning represents a revolutionary paradigm in machine learning, leveraging the capabilities of deep neural networks to autonomously learn intricate patterns and representations from large datasets. Its applications in image and speech recognition, language processing, and complex pattern analysis underscore its versatility and transformative potential across diverse domains. As technological advancements continue, deep learning remains at the forefront of cutting-edge research and practical implementations, shaping the future landscape of artificial intelligence.

Ensemble Learning: Ensemble learning emerges as a potent machine learning technique dedicated to enhancing predictive accuracy by leveraging the collective intelligence of multiple models. In the realm of regression, where the goal is to predict continuous numeric values, ensemble methods play a pivotal role in refining the precision and reliability of predictive models. The fundamental concept underlying ensemble learning involves the amalgamation of predictions from diverse individual models, thus harnessing the strengths of each to mitigate weaknesses and improve overall performance.

In the context of regression, ensemble methods prove particularly valuable for addressing the intricacies of predicting continuous outcomes. Techniques such as bagging (Bootstrap Aggregating) and boosting offer distinctive approaches to ensemble learning. Bagging involves training multiple instances of the same model on different subsets of the dataset, subsequently aggregating their predictions to achieve a more stable and accurate overall prediction. On the other hand, boosting iteratively builds a sequence of models, with each subsequent model focusing on correcting the errors of its predecessor, resulting in a highly adaptive and robust predictive model.

Ensemble learning in regression extends beyond these classic approaches to incorporate sophisticated methodologies like Random Forests, Gradient Boosting Machines (GBM), and

Stacking. Random Forests, for instance, employ an ensemble of decision trees to collectively contribute to the final prediction, offering a versatile and powerful regression technique. Gradient Boosting Machines iteratively refine predictions by emphasizing areas where previous models fall short, progressively enhancing predictive accuracy.

The significance of ensemble learning in regression is evident in its capacity to mitigate overfitting, improve generalization to unseen data, and enhance model robustness. This is particularly crucial in scenarios where the complexity of the underlying data necessitates a nuanced and adaptive modeling approach. The versatility of ensemble methods positions them as indispensable tools in regression tasks, contributing to the advancement of predictive modeling and the broader field of machine learning.

3.1 Regression: Regression, a foundational technique in both machine learning and statistics, serves as a pivotal tool for modeling and quantifying the relationship between a dependent variable, often termed the target, and one or more independent variables, known as features or predictors. The fundamental objective of regression is to derive a mathematical model that most accurately characterizes this relationship, facilitating the prediction of the target variable based on the values observed in the independent variables. This modeling process enables a comprehensive understanding of the interplay between variables and empowers predictive analytics in various domains.

The predictive utility of regression is harnessed in a myriad of applications, with its versatility extending to tasks such as predicting prices, estimating values, and uncovering intricate relationships between variables. In scenarios where there exists a numerical outcome of interest, regression provides a quantitative framework to discern patterns and make informed predictions. For instance, in the realm of financial markets, regression models can be employed to predict stock prices based on various economic indicators. Similarly, in the field of healthcare, regression aids in forecasting patient outcomes by considering a multitude of contributing factors.

Linear regression, a fundamental variant of this technique, assumes a linear relationship between the dependent and independent variables. However, regression encompasses a spectrum of methods, including polynomial regression, multiple regression, and nonlinear regression, each tailored to address the nuances of specific datasets and relationships. The modeling process involves optimizing parameters to minimize the difference between the predicted values generated by the model and the actual observed values in the dataset, a technique often known as fitting the model to the data.

In essence, regression stands as an indispensable tool in the data scientist's arsenal, offering a systematic and quantitative approach to understanding and predicting relationships within complex datasets. Its widespread application across diverse domains underscores its importance as a foundational building block in the broader landscape of machine learning and statistical analysis.

Since the work has been done on four regression models, we will concentrate on these:

1) Support Vector Regression (SVR)
2) Artificial Neural Network (ANN)
3) K-Nearest Neighbors (KNN)
4) Random Forest (RF)

4.Objectives

The primary objective of this innovative research is to introduce and empirically validate a novel approach in the realm of machine learning by developing an ensemble-based regression model using lexicographic multi-objective optimization. This cutting-edge methodology harnesses the capabilities of four distinct base models—Random Forest, Support Vector Machines, Artificial Neural Network, and K-Nearest Neighbors—as the foundational building blocks of the ensemble. The central focus of this approach is to elevate prediction performance, with a specific emphasis on the reduction of Root Mean Square Error (RMSE) and Mean Absolute Error (MAE).

The crux of this research lies in the incorporation of lexicographic multi-objective optimization, a sophisticated technique aimed at simultaneously optimizing both RMSE and MAE. The uniqueness of this approach lies in the application of lexicographic ordering to prioritize these objectives, enabling a nuanced and comprehensive optimization process. By doing so, the research seeks to derive optimal weights for the individual base models within the ensemble, thus pushing the boundaries of ensemble learning and showcasing superior predictive capabilities.

The overarching goal of this research is not only to contribute to the theoretical advancements in machine learning methodologies but also to demonstrate the practical utility of the proposed ensemble-based regression model. To validate the efficacy of the approach, the research employs a benchmark cost-benefit analysis dataset, specifically tailored to the context of a steel plant. Through rigorous empirical validation, the study aims to showcase the superior predictive performance and versatility of the proposed ensemble model across various domains, providing a significant leap forward in the landscape of machine learning applications.

5. Methodology/Model

5.1 Training and Test set

The methodology adopted for this research involves a crucial step in ensuring robust model evaluation and generalization. The dataset has been judiciously divided into two distinct subsets: a training set and a test set, adhering to the widely accepted split ratio of 80:20. This strategic partitioning of the data serves to strike a balance between training the models on a sufficiently large and diverse set of examples while reserving a portion for unbiased evaluation and performance assessment.

In detail, the training set, encompassing 80% of the dataset, serves as the foundation for model development and learning. The diverse patterns and relationships within this subset provide the basis for the models to understand and adapt to the underlying data distribution. Through an iterative training process, the models adjust their parameters to minimize errors and capture the nuances present in the training set.

Following the rigorous training phase, the models are subjected to a critical evaluation on the test set, constituting the remaining 20% of the dataset. This partitioning ensures that the models are assessed on data they have not encountered during the training process, serving as a litmus test for their generalization capabilities. The test set, being an independent subset, allows for a robust examination of how well the models can extend their learned patterns to new, unseen instances, thereby providing a realistic measure of their predictive performance.

The adoption of this 80:20 split ratio for training and testing is a widely recognized best practice in machine learning, striking a balance between training efficiency and model generalization. This meticulous approach ensures that the final performance evaluation of the ensemble-based regression model is both rigorous and unbiased, laying the groundwork for credible and reliable conclusions regarding its predictive capabilities across diverse domains.

In this study, we opted for Support Vector Regression (SVR), Random Forest (RF), Artificial Neural Networks (ANN), K-Nearest Neighbors (KNN):

5.2.1 Support Vector Regression (SVR)

Support Vector Regression (SVR) stands as a sophisticated regression technique that draws inspiration from the principles of support vector machines. Unlike traditional linear regression methods, SVR is adept at predicting continuous numerical values by identifying a hyperplane that optimally fits the data, all while incorporating a margin of error, often referred to as a 'tube.' This innovative approach makes SVR especially powerful for handling non-linear relationships within datasets, offering a robust and flexible solution for diverse regression tasks.

The fundamental premise of SVR lies in its ability to create a hyperplane that not only fits the data points as closely as possible but also accommodates a designated margin of error. This margin, represented by the 'tube,' allows for a degree of flexibility in the model, acknowledging that complete precision may not always be achievable or desirable. SVR excels in scenarios where traditional linear regression methods may fall short, particularly when faced with complex, non-linear relationships between variables.

One key strength of SVR is its adaptability to a wide range of kernel functions, which serve as transformation mechanisms for the input data. These kernel functions enable SVR to map the data into higher-dimensional spaces, uncovering intricate patterns and relationships that might be imperceptible in the original feature space. This capacity to handle non-linearity sets SVR apart as a valuable tool for regression tasks characterized by intricate and dynamic data interactions.

In practical applications, SVR finds utility in diverse domains, from finance and economics to engineering and environmental modeling. Its ability to capture non-linear dependencies empowers it to effectively model complex systems and predict continuous outcomes with enhanced accuracy. As a result, SVR has become a go-to choice for researchers and practitioners seeking a versatile and robust regression technique that transcends the limitations of traditional linear models.

In conclusion, Support Vector Regression represents a powerful approach to regression tasks, leveraging the principles of support vector machines to handle non-linear relationships in data. Its adaptability to various kernel functions and suitability for diverse domains make it a valuable tool for predictive modeling where capturing intricate patterns is paramount.

The results for the SVR model are shown below and Table-A shows the SVR Predicted Values:

Mean Square Error: 1387.91

Root Mean Squared Error: 37.25

R-squared: 0.96

Mean Absolute Error: 23.81

Mean Absolute Percentage Error (MAPE): 1.38%

Symmetric Mean absolute percentage error (sMAPE): 1.37%

Table-A: SVR Predictions

TEST SET	SVR PREDICTIONS
1995	1987.80150016
1824	1755.26006243
2002	1958.94680308
1956	1939.91856272
1650	1644.51761083
1563	1656.65355052
1678	1649.04240637
1541	1545.28653441
1753	1745.03548444
1527	1522.71775618
1543	1541.99214238
1578	1573.00532408

5.2.2 Artificial Neural Networks (ANN):

Artificial Neural Networks (ANN) represent a cutting-edge class of regression models, drawing inspiration from the intricate architecture of the human brain's neural network. The fundamental building blocks of ANNs are layers of interconnected nodes, commonly referred to as neurons. These neurons collaborate to process data, facilitating the learning of complex patterns and relationships within the input information.

The structural analogy with the human brain is a defining feature of ANNs, mimicking the interconnectedness and parallel processing capabilities of biological neural networks. ANNs are characterized by an input layer, one or more hidden layers, and an output layer. Each layer contains neurons that are interconnected through weighted connections, and these weights are adjusted during the training process to optimize the model's predictive capabilities.

One of the remarkable strengths of ANNs lies in their ability to capture intricate relationships within data. Through the process of learning from examples, ANNs can discern and internalize complex patterns, enabling them to generalize well to unseen data. This adaptability makes ANNs particularly effective in regression tasks, where the goal is to predict continuous numerical values based on input features.

ANNs find significant application in the domain of deep learning, a subfield of machine learning characterized by the use of deep neural networks with multiple layers. The depth of these networks allows them to automatically extract hierarchical features from the data, making them well-suited for tasks that involve nuanced and intricate relationships. In regression applications within deep learning, ANNs have demonstrated exceptional performance, outperforming traditional regression models, especially when faced with large and complex datasets.

In conclusion, Artificial Neural Networks stand as versatile regression models, leveraging the structural inspiration from the human brain to process data and learn intricate patterns. Their adaptability and effectiveness in capturing complex relationships make them a cornerstone in deep learning applications, particularly in the realm of regression tasks where uncovering intricate patterns is paramount.

The results of this model are shown below and Table-B shows the ANN Predicted Values:

Mean Square Error: 18165.37

Root Mean Squared Error: 134.78

R-squared: 0.42

Mean Absolute Error: 123.10

Mean Absolute Percentage Error (MAPE): 7.28%

Symmetric Mean absolute percentage error (sMAPE): 7.14%

Table B: ANN Predictions

TEST SET	ANN PREDICTIONS
1995	1831.84
1824	1769.07
2002	1823.83
1956	1818.03
1650	1735.50
1563	1733.69
1678	1733.10
1541	1702.71
1753	1760.79
1527	1696.3
1543	1702.28
1578	1711.61

5.2.3 K-Nearest Neighbors (KNN):

K-Nearest Neighbors (KNN) stands out as a straightforward yet powerful regression model that derives its strength from simplicity and adaptability. The core principle of KNN involves predicting a target value by averaging the values of its k-nearest neighbors within the training dataset. This intuitive approach makes KNN particularly effective for tasks characterized by localized patterns and varying data distributions, positioning it as a valuable tool in the realm of regression analysis.

The simplicity of the KNN algorithm lies in its ease of implementation and understanding. The model does not assume any underlying mathematical structure in the data, making it particularly versatile and suitable for scenarios where the relationship between input features and the target variable may not adhere to a predefined form. Instead, KNN relies on proximity and similarity measures to make predictions, adapting to the local characteristics of the dataset.

A notable feature of KNN is its adaptability to varying data distributions. The model's performance is not contingent on specific assumptions about the shape or nature of the data, allowing it to handle situations where other regression models might struggle. This adaptability is especially advantageous in real-world scenarios where the complexity of the underlying patterns may not be fully understood or where the data exhibits localized variations.

KNN's effectiveness in regression tasks extends to its ability to handle both linear and non-linear relationships. By considering the k-nearest neighbors, the model inherently captures local trends and variations, making it well-suited for datasets with diverse and intricate structures. Moreover, KNN's performance tends to improve with larger datasets, as a greater number of neighbors contribute to more robust predictions.

In summary, K-Nearest Neighbors emerges as a powerful yet straightforward regression model, offering a pragmatic solution for tasks involving localized patterns and diverse data distributions. Its adaptability and simplicity make it a valuable tool in regression analysis, particularly in scenarios where other models might struggle to capture the nuances of the data.

The results of this model are shown below and Table-C shows the KNN Predicted Values:

Mean Squared Error: 5054.70

Root Mean Squared Error: 71.10

R-Squared: 0.84

Mean Absolute Error (MAE): 52.05

Mean Absolute Percentage Error (MAPE): 2.89%

Symmetric Mean Absolute Percentage Error: 2.89%

Table C: KNN Predictions

TEST SET	KNN PREDICTIONS
1995	1877.9
1824	1865.9
2002	1877.2
1956	1871.9
1650	1724.2
1563	1569.4
1678	1680.2
1541	1556.4
1753	1692.7
1527	1598.3
1543	1561.5
1578	1602.2

5.2.4 Random Forest (RF):

Random Forest (RF) stands out as a robust ensemble-based regression model that harnesses the collective power of multiple decision trees. Through the aggregation of predictions from individual trees, RF not only enhances accuracy but also ensures the robustness of regression results. Renowned for its capacity to handle intricate relationships within data, RF emerges as a versatile and effective choice for a diverse array of regression tasks.

At the core of the Random Forest algorithm is the concept of decision trees, which individually contribute predictions based on the features of the input data. The ensemble nature of RF involves creating numerous decision trees, each trained on a subset of the dataset or with different feature subsets. The final prediction is then determined by aggregating the outputs of these trees, commonly through averaging in the case of regression tasks. This ensemble approach mitigates overfitting, reduces variance, and enhances the overall generalization capability of the model.

One of the distinctive strengths of RF lies in its adeptness at handling complex relationships in data. The combination of diverse decision trees captures nuanced patterns and interactions within the dataset, making RF particularly effective when the underlying structure is multifaceted and intricate. This capability is crucial for regression tasks where the predictive relationship between features and the target variable may exhibit non-linearity or interactions that are not easily captured by simpler models.

Furthermore, RF demonstrates notable scalability and efficiency, especially when dealing with large datasets. The parallel training of individual trees and the aggregative nature of predictions make RF well-suited for scenarios where the volume of data is substantial. This scalability, coupled with its ability to manage high-dimensional feature spaces, positions RF as a go-to choice for regression tasks in diverse domains.

In conclusion, Random Forest emerges as a powerful ensemble-based regression model that excels in handling complex data relationships and large datasets. Its ensemble nature, drawing strength from multiple decision trees, contributes to enhanced accuracy and robustness, making RF a versatile and effective tool for a broad spectrum of regression analyses.

The results of the RF model are shown below and Table-D shows the RF Predicted Values:
Mean Square Error: 12695.06
Root Mean Squared Error: 112.67
R-squared: 0.59
Mean Absolute Error: 55.00
Mean Absolute Percentage Error (MAPE): 2.86%
Symmetric Mean absolute percentage error (sMAPE): 3.04%

Table D: RF Predictions

TEST SET	RF PREDICTIONS
1995	1957.3
1824	1749.7
2002	1833.0
1956	1613.4
1650	1658.4
1563	1570.4
1678	1681.2
1541	1542.9
1753	1768.8
1527	1531.5
1543	1536.1
1578	1571.7

5.3 Hyperparameter Tuning

Hyperparameter optimization plays a pivotal role in enhancing the predictive performance of machine learning models, particularly in scenarios where the expected R-squared value is not readily achieved. Grid Search, a popular technique for hyperparameter tuning, is employed to systematically explore various configurations and identify the set of hyperparameters that yields the highest R-squared value. This iterative process involves defining a range of values for key model settings, testing different combinations, and selecting the optimal configuration based on performance evaluation, specifically targeting the R-squared metric.

The overarching objective of hyperparameter tuning is to fine-tune machine learning models to maximize their performance on new and unseen data. This iterative optimization process is crucial for achieving the highest possible accuracy, robustness, and generalization capabilities of the models. It requires selecting a specific machine learning model, determining a range of values for its hyperparameters, and systematically evaluating different combinations to identify the most effective configuration.

The workflow of the undertaken work is visually depicted in Figure 1, illustrating the step-by-step process involved in optimizing the machine learning models. The initial phase involves the careful splitting of the dataset into a training set and a test set, a fundamental step to ensure unbiased model evaluation. Subsequently, four distinct machine learning models—Support Vector Regression (SVR), Random Forest (RF), K-Nearest Neighbors (KNN), and Artificial Neural Network (ANN)—are trained on the designated training dataset. Following the training phase, the efficiency and predictive performance of each model are rigorously assessed using the test set.
Then lexicographic optimization is done for the further evaluation.

This comprehensive approach allows for a thorough evaluation of the effectiveness of each machine learning model in capturing the underlying patterns within the data. By systematically tuning hyperparameters through techniques like Grid Search, the aim is to optimize the models for superior performance, ultimately enhancing their ability to make accurate predictions on new and unseen data. The visual representation in Figure 1 encapsulates the methodological rigor applied in the study, providing a clear overview of the workflow and highlighting the importance of hyperparameter optimization in achieving optimal model performance.

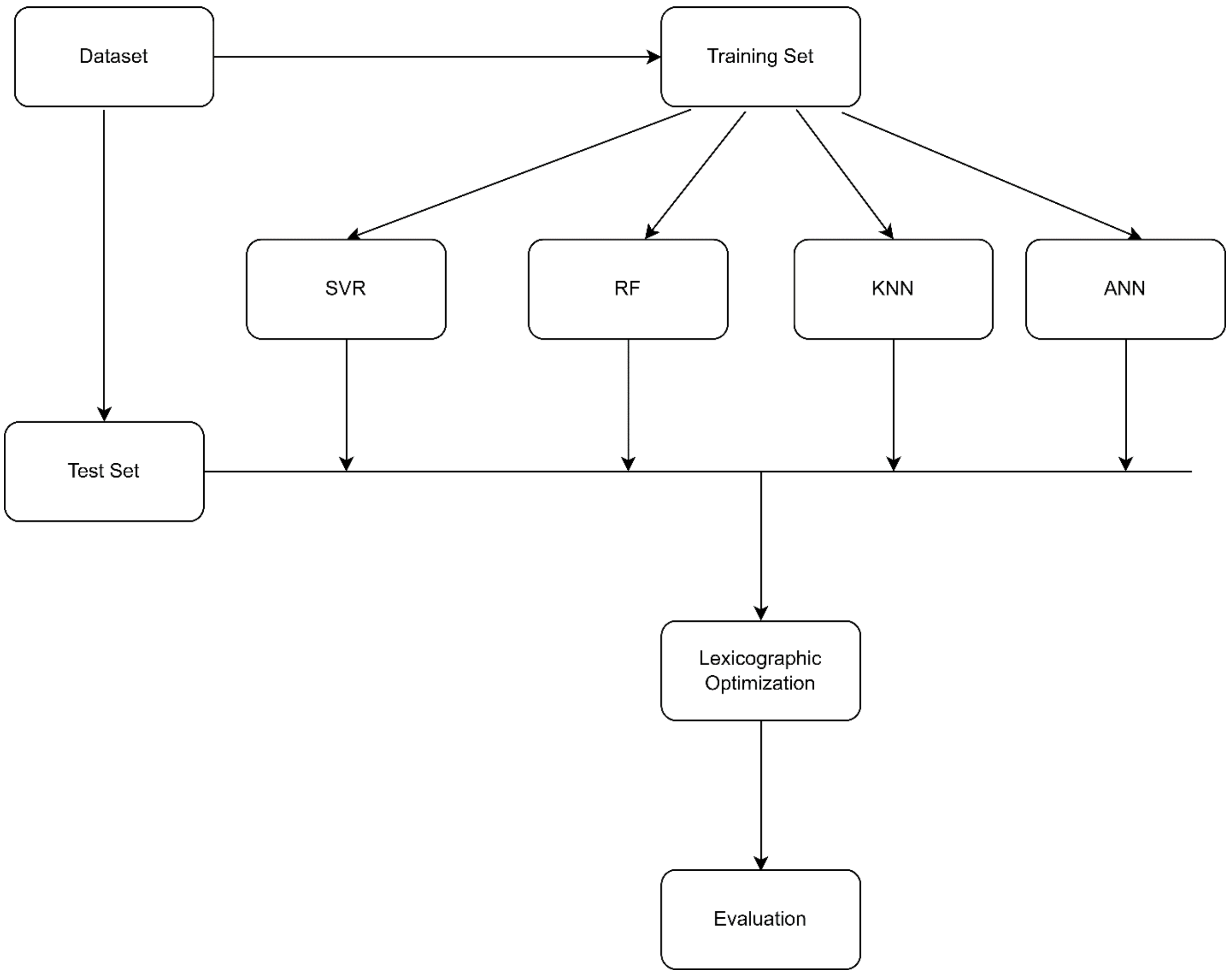


Figure 1. Process workflow of work done till now

5.4. Experimental Results

We used *scikit-learn* to implement our models in Python on Google Colab. This popular and widely used machine learning library provides APIs to implement the most common algorithms in an efficient way**.** The results we obtained from different models are shown in Table 1.Also the final optimized weights of different models is shown in Table 2.

Table 1. Results obtained from different models

Model Name	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)	R-squared	Mean Absolute Error (MAE)
SVR	1387.91	37.25	0.96	23.81
RF	12695.06	112.67	0.59	55.00
KNN	4462.68	66.80	0.86	53.37
ANN	18165.37	134.78	0.42	123.10

Table 2. Optimized weights of different models

Models	**Optimized Weights**
SVR	7.82705131e-01
RF	-6.47639904e-10
KNN	2.17294870e-01
ANN	-1.62399527e-10

The calculated values of MAE and RMSE after a simple average ensemble are: 58.705 and 61.4275, respectively

Final RMSE and MAE values after lexicographic bi-objective optimization:

MAE: 25.237098593955086
RMSE:32.886844986486956

6.Case Study

6.1 Data Description and Variables

In the steel plants of a reputed steel manufacturer, it is observed that there is a high possibility of accidents occurring. One such safety issue is in steel melting shops where the electric arc furnace can cause hot metal spillage and dropping of sparks on the hands or skin of operators which can cause permanent injury. This happens due to the carelessness of the workers and not following safe operating practices.

To reduce the severity of such accidents, the administration of the industry has conceptualized a strategy, which requires the tilting of the furnace on time in the right manner. For this strategy, the direct cost includes the training cost, the operation cost, and the maintenance cost. The indirect cost mostly includes the loss of production due to the implementation of a safety strategy. All the costs and benefits were computed for a period of 60 weeks, based on which an ensemble model is required to be developed.

Our dataset was sourced from a report on accident investigations conducted at a facility specializing in steel processing. From the dataset, we selected the Direct Cost and Indirect Cost as our independent features, with Benefit serving as our dependent variable, often referred to as the "y" variable. The direct cost can be expressed as:

Direct cost = $C_1 + C_R$ --------------(1)

where,

C_1 - Initial cost of a safety strategy; C_R -Recurring cost of a safety strategy

The indirect cost of implementation of a safety strategy is defined as:

Indirect Cost=C_P+C_A+ C_{AS} ---------------(2)

where,

C_P-Cost of loss of production due to safety strategy; C_A-Accident Cost of Safety Strategy; C_{AS}-Annual Cost of Safety Strategy, which includes insurance and legal costs

Benefit = Cost of saving due to implementation of safety strategy

Table AA below shows the sample dataset.

Table AA: Sample Dataset

Sl.No.	Direct Cost	Indirect Cost	Benefit
1	1498	3153	1995
2	1376	3106	1809
3	1168	3091	1808
4	859	3054	1674
5	752	3042	1564

6.2 Application of Machine Learning Models

In contemporary data analysis, the application of machine learning models has become indispensable, and this holds particularly true for regression tasks. This section highlights four prominent machine learning models employed for regression analysis, each offering distinct advantages in addressing complex problems: Random Forest Regression, Artificial Neural Network (ANN) Regression, Support Vector Regression (SVR), and K-Nearest Neighbors (KNN) Regression.

Random Forest Regression

Random Forest Regression is a powerful ensemble learning approach that harnesses the collective strength of multiple decision trees to make accurate predictions. What sets this model apart is its versatility, allowing it to effectively handle both categorical and numerical data. This adaptability makes Random Forest Regression well-suited for tackling intricate regression tasks where the relationship between variables may involve a mix of discrete and continuous features.

The fundamental principle behind Random Forest Regression lies in the aggregation of predictions from diverse decision trees. Instead of relying on a single tree, the model constructs an ensemble of trees, each trained on a different subset of the dataset or with different feature subsets. This ensemble approach mitigates overfitting, reduces variance, and enhances the model's ability to generalize to unseen data.

One notable advantage of Random Forest Regression is its robustness in dealing with noisy or complex datasets. The combination of multiple decision trees, each capturing different aspects of the data, contributes to the model's ability to handle variations and outliers effectively. This robustness is particularly valuable in real-world scenarios where datasets may exhibit inherent complexities or contain irregularities.

Moreover, Random Forest Regression excels in providing accurate predictions across a range of scenarios. The model's ability to capture intricate patterns within the data, along with its adaptability to different types of features, contributes to its success in regression tasks. This makes it a reliable choice for applications where the relationship between input variables and the target variable is not easily characterized by a simple linear or nonlinear model.

In conclusion, Random Forest Regression emerges as a versatile and robust ensemble learning model suitable for complex regression tasks. Its capacity to handle both categorical and numerical data, coupled with its ability to deliver accurate results in the presence of noise or complexity, positions Random Forest Regression as a valuable tool in the machine learning toolkit for regression analysis.

Artificial Neural Network (ANN) Regression

Artificial Neural Network (ANN) Regression leverages the potency of deep learning to unravel intricate relationships and patterns embedded within data. Constructed with layers of interconnected neurons, ANNs undergo a dynamic adjustment of connections during training, allowing them to approximate complex functions. The adaptability ingrained in ANNs positions them as an ideal choice for regression problems, particularly when traditional methodologies encounter limitations.

The distinctive feature of ANNs lies in their ability to model nonlinear relationships and dependencies within the data. Unlike conventional regression techniques that may struggle with intricate patterns, ANNs excel in capturing the nuanced structures present in real-world datasets. This makes them invaluable for tasks where the underlying relationships are complex and not easily delineated by conventional linear or nonlinear models.

The training process of ANNs involves learning from the data, iteratively refining the connections between neurons to improve predictive accuracy. This iterative learning enables ANNs to adapt to the specific characteristics of the dataset, making them highly effective in scenarios where the relationships are dynamic or context-dependent.

In essence, ANN Regression stands as a powerful tool for addressing regression challenges, providing a sophisticated framework capable of uncovering hidden patterns in data. Its deep learning architecture, coupled with adaptability, positions ANN Regression as a valuable asset in the realm of regression analysis, offering a flexible and effective approach for capturing the complexities inherent in diverse datasets.

Support Vector Regression (SVR)

Support Vector Regression (SVR) stands out as a specialized regression technique crafted to discern the optimal hyperplane that seamlessly fits the data, all while allowing for a specified margin of error. Its unique design makes SVR particularly valuable when confronted with datasets featuring outliers or intricate nonlinear relationships. The primary objective of SVR is to minimize margin violations while adeptly modeling the underlying data, rendering it a robust and reliable choice for regression tasks that demand precision and resilience.

The fundamental principle guiding SVR involves identifying a hyperplane that best represents the data's structure while acknowledging a margin of error. This margin allows for a certain degree of flexibility in accommodating variations and outliers within the dataset, enhancing the model's ability to generalize to unseen data effectively. SVR's focus on minimizing margin violations ensures that the selected hyperplane optimally captures the intrinsic patterns within the data, even in the presence of complex relationships.

One of SVR's key strengths lies in its versatility across various regression scenarios, particularly those where traditional linear models may fall short. By incorporating a kernel function, SVR can efficiently handle nonlinear relationships, allowing it to adapt to the intricacies of diverse datasets. This adaptability positions SVR as a powerful tool for regression tasks in fields such as finance, where nonlinear relationships often play a crucial role.

In summary, Support Vector Regression emerges as a specialized and robust technique for regression analysis, excelling in scenarios marked by outliers or intricate nonlinear relationships. Its ability to identify an optimal hyperplane, minimize margin violations, and adapt to diverse data structures makes SVR a valuable asset in the machine learning toolkit, particularly for tasks that demand precision and resilience in capturing the underlying patterns within the data.

K-Nearest Neighbors (KNN) Regression

K-Nearest Neighbors (KNN) Regression provides a simple yet effective methodology for regression analysis, estimating the target value for a given instance by leveraging the average or weighted target values of its k nearest neighbors within the training dataset. Notably, KNN Regression distinguishes itself as a non-parametric approach, meaning it makes minimal assumptions about the underlying data distribution. This non-parametric nature is particularly advantageous when dealing with datasets characterized by noise or intricate patterns, allowing KNN Regression to perform admirably across various regression scenarios.

The key mechanism employed by KNN Regression involves identifying the closest neighbors to a given data point and utilizing their target values to predict the target value for that point. The choice of k, representing the number of neighbors considered, influences the model's sensitivity to local variations. This adaptability makes KNN Regression suitable for situations where the relationships between variables may not adhere to a predefined structure.

The non-parametric nature of KNN Regression makes it a dependable option for scenarios where other regression models might struggle, especially when faced with complex or unpredictable data patterns. The model's ability to adapt to the local characteristics of the data contributes to its robustness in capturing nuanced relationships.

In conclusion, the choice of a regression model should be informed by the specific characteristics of the data and the unique requirements of the problem under consideration. Random Forest Regression, ANN Regression, SVR, and KNN Regression each offer distinct advantages within the realm of machine learning, catering to a diverse array of regression tasks across various domains. These models collectively serve as valuable tools for researchers and analysts seeking to tackle the multifaceted challenges presented by regression analysis.

6.3 Application of Ensemble

In the context of cost-benefit analysis within the steel industry, the utilization of ensemble techniques has emerged as a transformative approach. This research introduces a groundbreaking initiative by presenting a Lexicographic Multi-Objective Optimization-Based Heterogeneous Ensemble Regression Analysis, specifically crafted to address the intricate nuances of cost-benefit analysis within a steel plant. Through the integration of diverse regression models within an ensemble framework, our methodology offers a comprehensive perspective on the multifaceted factors influencing the financial landscape of steel manufacturing. This innovative ensemble approach surpasses the limitations of individual regression models, providing a more precise and robust foundation for critical decision-making processes, risk assessment, and strategic planning. The synergistic application of Lexicographic Multi-Objective Optimization and ensemble techniques equips us to navigate the intricate trade-offs inherent in cost-benefit analysis, generating insights crucial for enhancing efficiency and sustainability in the steel production sector.

Furthermore, in our analysis, we have computed the mean and standard deviation values for Indirect Cost, Direct Cost, and Benefit data, as detailed in Table 3. These statistical measures contribute to a more comprehensive understanding of the financial landscape within the steel plant, offering valuable insights into the variability and central tendency of key cost and benefit parameters. The calculated mean values provide a measure of the central tendency, while the standard deviation values offer insights into the dispersion or variability around these central values. These statistical metrics serve as essential tools for decision-makers, enabling a more informed and nuanced interpretation of the financial aspects involved in the steel manufacturing process.

Table 3. Mean and Standard Deviation

	Mean	Standard Deviation
Indirect Cost	3086.217	48.60833
Direct Cost	1044.567	313.5459
Benefit	1734.967	165.1277

7. Comparative analysis

The predicted value on the test data set, model-wise, is shown in Table 4, basically, the table is a comparison between the benefit (Actual Predicted Value) and Simple Average Ensemble Predicted Value also there are the prediction values from the four models. Also, a bar graph is shown in Figure 2 where there is a pictorial representation of the comparison between Benefit and Simple Average Ensemble.

Table 4. Benefit (Actual Predicted Value) vs Simple Average Ensemble

Benefit (Actual Predicted Value) y	SVR Predicted Value	RF Predicted Value	KNN Predicted Value	ANN Predicted Value	Simple Average Ensemble Predicted Value ŷ	Z=|y- ŷ|	Z^2
1995	1987.80	1957.3	1877.9	1831.84	1918.21	76.79	5896.7
1824	1755.26	1749.7	1865.9	1769.07	1782.73	41.27	1703.2
2002	1958.95	1833.0	1877.2	1823.83	1873.24	128.76	16579.13
1956	1939.92	1613.4	1871.9	1818.03	1810.81	145.19	21080.13
1650	1644.52	1658.4	1724.2	1735.50	1690.65	40.65	1652.42
1563	1656.65	1570.4	1569.4	1733.69	1632.53	69.53	4834.42
1678	1649.04	1681.2	1680.2	1733.10	1685.88	7.88	62.1
1541	1545.28	1542.9	1556.4	1702.71	1586.82	45.82	2099.47
1753	1745.03	1768.8	1692.7	1760.79	1741.83	11.17	124.77
1527	1522.72	1531.5	1598.3	1696.3	1587.20	60.2	3624.04
1543	1541.99	1536.1	1561.5	1702.28	1585.46	42.46	1802.85
1578	1573.00	1571.7	1602.2	1711.61	1614.62	40.62	1649.98

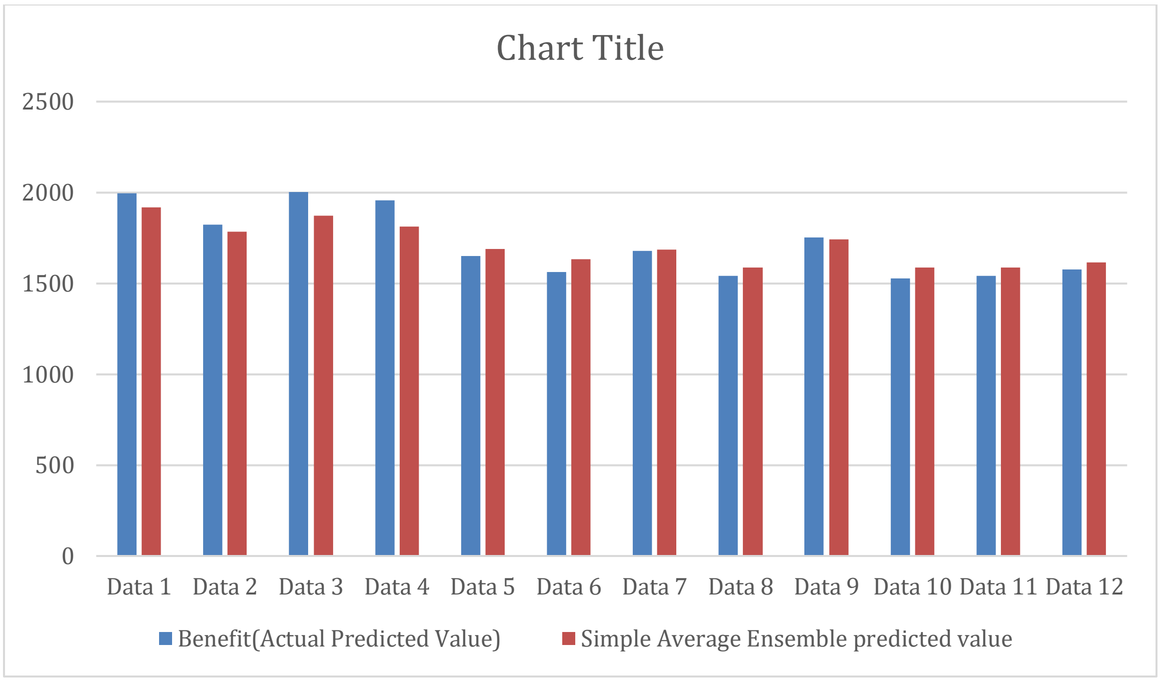


Figure 2. Comparison between Benefit and Simple Average Ensemble

8. Discussion and Conclusion

The key innovation in this study lies in the implementation of *lexicographic* multi-objective optimization. This technique prioritizes the minimization of both *RMSE* and *MAE* through lexicographic ordering, ultimately deriving optimal weights for the base models. The use of multi-objective optimization adds depth to the ensemble, allowing for a more nuanced and balanced approach to prediction accuracy.

In conclusion, this research presents a significant advancement in ensemble learning, which we will implement in predictive modeling. The validation of the proposed ensemble model on a real-world cost-benefit analysis dataset from a steel plant will demonstrate its practical utility.

By embracing the power of *lexicographic* multi-objective optimization and the diversity of four base models, this study opens new doors for improved predictive capabilities across various domains. This approach is not only applicable to cost-benefit analysis but offers potential benefits in diverse sectors where precise and reliable predictions are of utmost importance. As such, this work contributes to the ongoing evolution of ensemble learning and promises to impact data-driven decision-making positively.

References

Carsten, O. M., & Tate, F. N. (2005). Intelligent speed adaptation: accident savings and cost–benefit analysis. *Accident Analysis & Prevention*, *37*(3), 407-416.

Grisci, B., Kuhn, G., Colombelli, F., Matter, V., Lima, L., Heinen, K., ... & Ramos, G. D. O. (2022). Perspectives on risk prioritization of data center vulnerabilities using rank aggregation and multi-objective optimization. *arXiv preprint arXiv:2202.07466*.

Khalilpourazari, S., & Khalilpourazary, S. (2017). A lexicographic weighted Tchebycheff approach for multi-constrained multi-objective optimization of the surface grinding process. *Engineering Optimization*, *49*(5), 878-895.

Lai, L., Fiaschi, L., & Cococcioni, M. (2020). Solving mixed Pareto-Lexicographic multi-objective optimization problems: The case of priority chains. *Swarm and Evolutionary Computation*, *55*, 100687.

Lai, L., Fiaschi, L., Cococcioni, M., & Deb, K. (2021). Solving mixed pareto-lexicographic multiobjective optimization problems: the case of priority levels. *IEEE Transactions on Evolutionary Computation*, *25*(5), 971-985.

Lindqvist, K., & Lindholm, L. (2001). A cost-benefit analysis of the community-based injury prevention programme in Motala, Sweden—a WHO Safe Community. *Public health*, *115*(5), 317-322.

Ikpe, E., Hammond, F., Proverbs, D., & Oloke, D. (2011). Model predicting cost benefit analysis (CBA) of accident prevention on construction projects. *International Journal of Safety and Security Engineering*, *1*(3), 298-311.

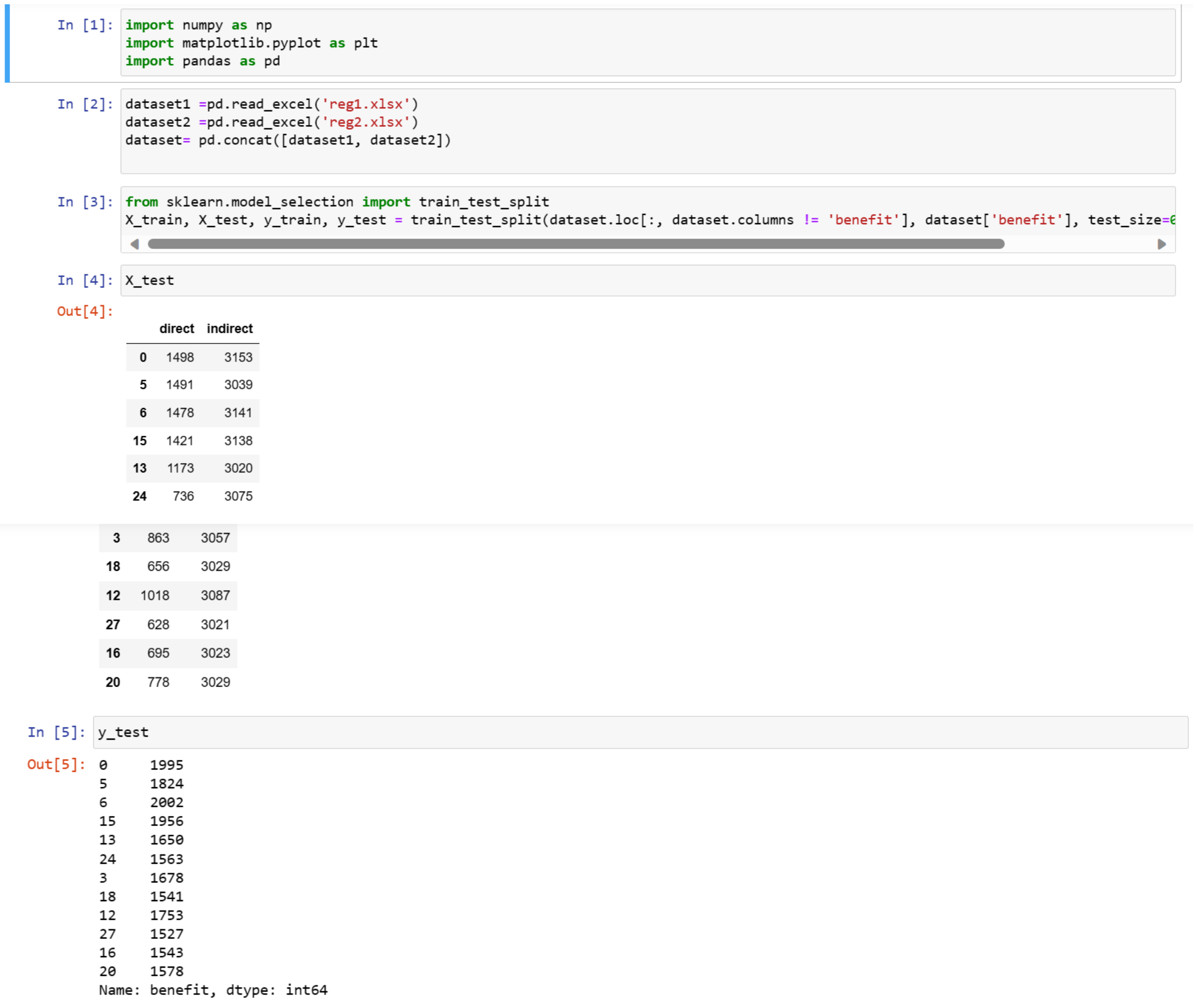
Pagliara, F., & Zingone, M. (2023). Providing resilience due to adverse weather events: A cost-benefit analysis for the case of the Milan Malpensa airport in Italy. *Journal of Air Transport Management*, *113*, 102484.

Rahman, M. S., Colbourne, B., & Khan, F. (2021). Risk-based cost benefit analysis of offshore resource centre to support remote offshore operations in harsh environment. *Reliability Engineering & System Safety*, *207*, 107340.

Van Coile, R., Lucherini, A., Chaudhary, R. K., Ni, S., Unobe, D., & Gernay, T. (2023). Cost-benefit analysis in fire safety engineering: State-of-the-art and reference methodology. *Safety Science*, *168*, 106326.

Appendix

The code used in this project is given below.



```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
```

```
In [2]: dataset1 =pd.read_excel('reg1.xlsx')
        dataset2 =pd.read_excel('reg2.xlsx')
        dataset= pd.concat([dataset1, dataset2])
```

```
In [3]: from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(dataset.loc[:, dataset.columns != 'benefit'], dataset['benefit'], test_size=
```

```
In [4]: X_test
```

Out[4]:

	direct	indirect
0	1498	3153
5	1491	3039
6	1478	3141
15	1421	3138
13	1173	3020
24	736	3075
3	863	3057
18	656	3029
12	1018	3087
27	628	3021
16	695	3023
20	778	3029

```
In [5]: y_test
```

```
Out[5]: 0      1995
        5      1824
        6      2002
        15     1956
        13     1650
        24     1563
        3      1678
        18     1541
        12     1753
        27     1527
        16     1543
        20     1578
        Name: benefit, dtype: int64
```

```
In [6]: from sklearn.svm import SVR
        svr=SVR()
        svr.fit(X_train, y_train)

Out[6]: ▾ SVR
        SVR()
```

```
In [7]: import numpy as np
        import pandas as pd
        from sklearn.svm import SVR
        from sklearn.model_selection import GridSearchCV, train_test_split

        param_grid = {
            'kernel': ['linear'],
            'C': [0.1],
            'epsilon': [0.01],
            'gamma': ['scale', 'auto', 0.1, 1]
        }

        grid_search = GridSearchCV(estimator=svr, param_grid=param_grid, cv=5, scoring='neg_mean_squared_error', n_jobs=-1)

        grid_search.fit(X_train, y_train)

        best_params = grid_search.best_params_
        best_estimator = grid_search.best_estimator_
```

```
In [8]: svr_predictions = best_estimator.predict(X_test)
```

```
In [9]: svr_predictions

Out[9]: array([1987.80150016, 1755.26006243, 1958.94680308, 1939.91856272,
               1644.51761083, 1656.65355052, 1649.04240637, 1545.28653441,
               1745.03548444, 1522.71775618, 1541.99214238, 1573.00532408])
```

```
In [10]: from sklearn.metrics import mean_squared_error, r2_score
         from sklearn.metrics import mean_absolute_error

         # Calculate MAE
         mae = mean_absolute_error(y_test, svr_predictions)
         mape = np.mean(np.abs((y_test - svr_predictions) / y_test)) * 100

         # Print the result

         # Print or use the MAE value
         mse = mean_squared_error(y_test, svr_predictions)
         rmse = np.sqrt(mse)
         def symmetric_mean_absolute_percentage_error(y_test, svr_predictions):
             # Avoiding division by zero
             y_test, svr_pred = np.array(y_test), np.array(svr_predictions)
             mask = y_test != 0
             y_test, svr_predictions = y_test[mask], svr_predictions[mask]

             # Calculate sMAPE
             smape = 2 * np.mean(np.abs(svr_predictions - y_test) / (np.abs(svr_predictions) + np.abs(y_test))) * 100

             return smape
         smape_value = symmetric_mean_absolute_percentage_error(y_test, svr_predictions)
         r2 = r2_score(y_test, svr_predictions)
         print(f"Mean Squared Error: {mse:.2f}")
         print(f"Root Mean Squared Error: {rmse:.2f}")
         print(f"R-squared: {r2:.2f}")
         print(f"Mean Absolute Error (MAE): {mae:.2f}")
         print(f"Mean Absolute Percentage Error (MAPE): {mape:.2f}%")
         print(f"Symmetric Mean absolute percentage error(sMAPE): {smape_value:.2f}%")

         Mean Squared Error: 1387.91
         Root Mean Squared Error: 37.25
         R-squared: 0.96
         Mean Absolute Error (MAE): 23.81
         Mean Absolute Percentage Error (MAPE): 1.38%
         Symmetric Mean absolute percentage error(sMAPE): 1.37%
```

```
In [11]: from sklearn.ensemble import RandomForestRegressor
         regressor=RandomForestRegressor(n_estimators=10, random_state=42)
         regressor.fit(X_train, y_train)
         rf_predictions = regressor.predict(X_test)
```

```
In [12]: rf_predictions
```

```
Out[12]: array([1975.3, 1749.7, 1833. , 1613.4, 1658.4, 1570.4, 1681.2, 1542.9,
                1768.8, 1531.5, 1536.1, 1571.7])
```

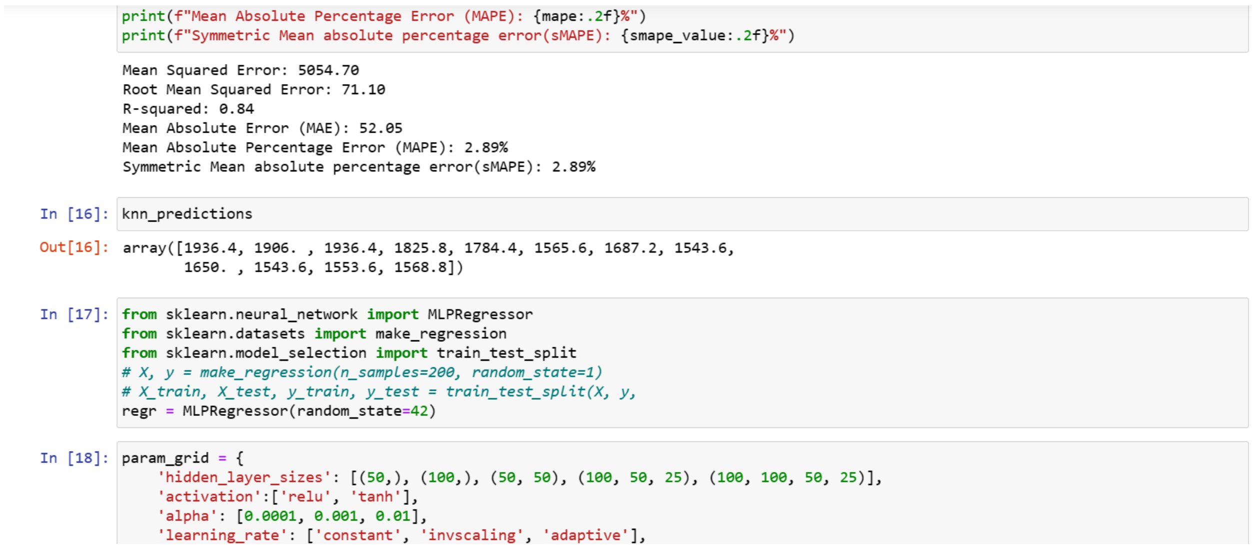
```
In [13]: from sklearn.metrics import mean_squared_error, r2_score
         from sklearn.metrics import mean_absolute_error

         # Calculate MAE
         mae = mean_absolute_error(y_test,rf_predictions)

         # Print or use the MAE value
         mse = mean_squared_error(y_test,rf_predictions)
         rmse = np.sqrt(mse)
         mape = np.mean(np.abs((y_test - rf_predictions) / y_test)) * 100

         def symmetric_mean_absolute_percentage_error(y_test, rf_predictions):
             # Avoiding division by zero
             y_test, y_pred = np.array(y_test), np.array(rf_predictions)
             mask = y_test != 0
             y_test, rf_predictions = y_test[mask], rf_predictions[mask]

             # Calculate sMAPE
             smape = 2 * np.mean(np.abs(rf_predictions - y_test) / (np.abs(rf_predictions) + np.abs(y_test))) * 100

             return smape

         # Calculate sMAPE
         smape_value = symmetric_mean_absolute_percentage_error(y_test, rf_predictions)

         r2 = r2_score(y_test,rf_predictions)
         print(f"Mean Squared Error: {mse:.2f}")
         print(f"Root Mean Squared Error: {rmse:.2f}")
         print(f"R-squared: {r2:.2f}")
         print(f"Mean Absolute Error (MAE): {mae:.2f}")
         print(f"Mean Absolute Percentage Error (MAPE): {mape:.2f}%")
         print(f"Symmetric Mean absolute percentage error(sMAPE): {smape_value:.2f}%")
```

```
Mean Squared Error: 12695.06
Root Mean Squared Error: 112.67
R-squared: 0.59
Mean Absolute Error (MAE): 55.00
Mean Absolute Percentage Error (MAPE): 2.86%
Symmetric Mean absolute percentage error(sMAPE): 3.04%
```

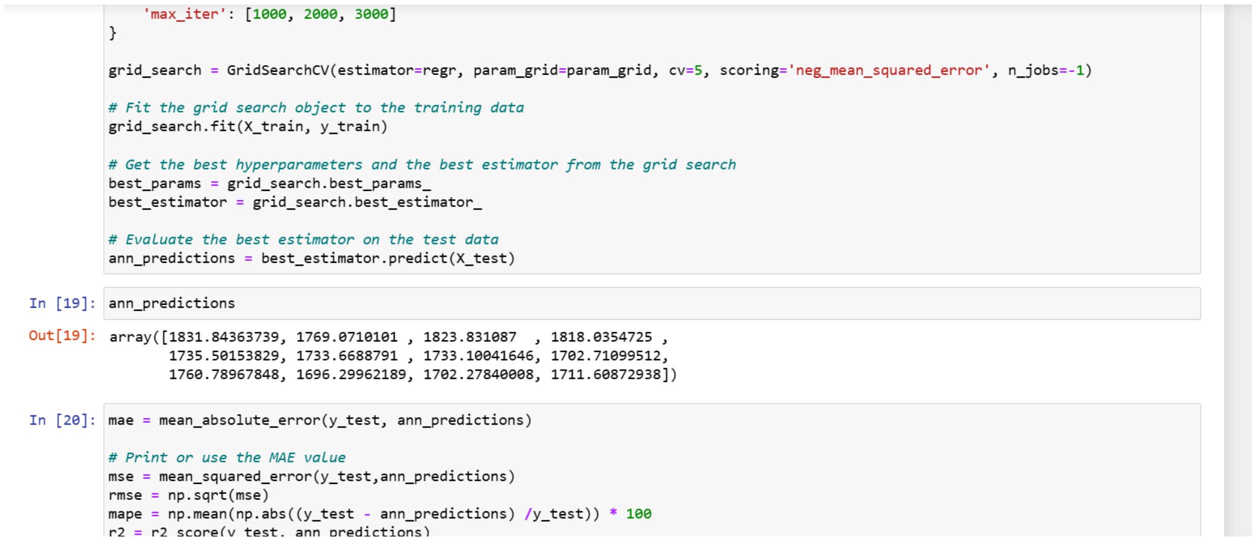
```
In [14]: from sklearn.neighbors import KNeighborsRegressor
         knr = KNeighborsRegressor(n_neighbors=5, algorithm = 'auto')
         knr.fit(X_train, y_train)
```

```
Out[14]: ▾ KNeighborsRegressor
         KNeighborsRegressor()
```

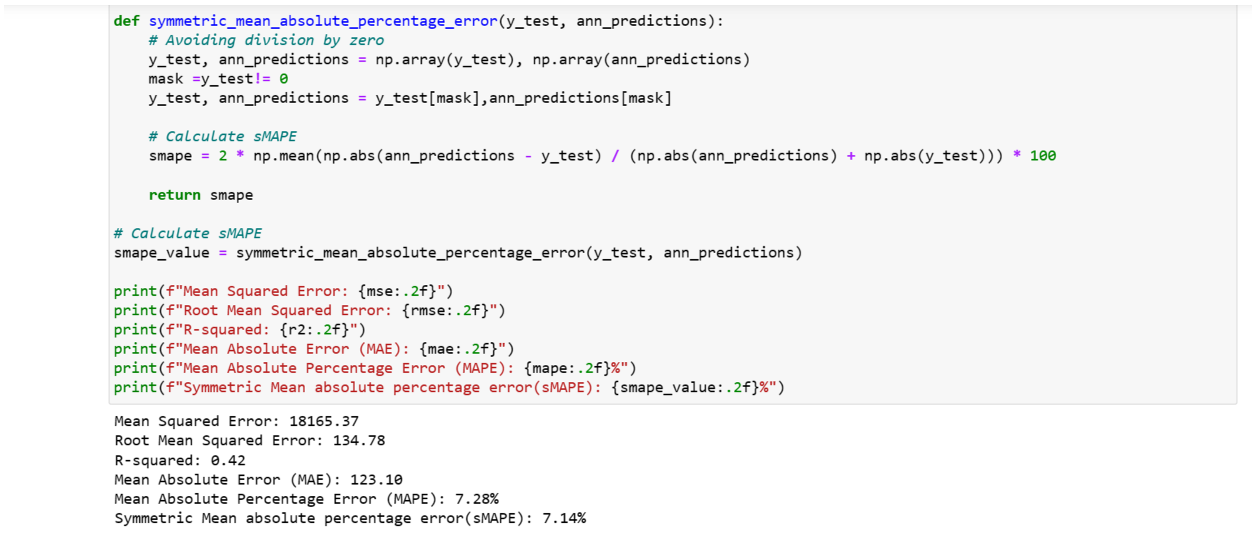
```
In [15]: knn_predictions = knr.predict(X_test)
         mae = mean_absolute_error(y_test, knn_predictions)

         # Print or use the MAE value
         mse = mean_squared_error(y_test, knn_predictions)
         rmse = np.sqrt(mse)
         mape = np.mean(np.abs((y_test -knn_predictions) / y_test)) * 100

         def symmetric_mean_absolute_percentage_error(y_test, knn_predictions):
             # Avoiding division by zero
             y_test, y_pred = np.array(y_test), np.array(knn_predictions)
             mask = y_test != 0
             y_test, knn_predictions = y_test[mask], knn_predictions[mask]

             # Calculate sMAPE
             smape = 2 * np.mean(np.abs(knn_predictions -y_test) / (np.abs(knn_predictions) + np.abs(y_test))) * 100

             return smape

         # Calculate sMAPE
         smape_value = symmetric_mean_absolute_percentage_error(y_test, knn_predictions)

         r2 = r2_score(y_test,knn_predictions)
         print(f"Mean Squared Error: {mse:.2f}")
         print(f"Root Mean Squared Error: {rmse:.2f}")
         print(f"R-squared: {r2:.2f}")
         print(f"Mean Absolute Error (MAE): {mae:.2f}")
```



```
        print(f"Mean Absolute Percentage Error (MAPE): {mape:.2f}%")
        print(f"Symmetric Mean absolute percentage error(sMAPE): {smape_value:.2f}%")

        Mean Squared Error: 5054.70
        Root Mean Squared Error: 71.10
        R-squared: 0.84
        Mean Absolute Error (MAE): 52.05
        Mean Absolute Percentage Error (MAPE): 2.89%
        Symmetric Mean absolute percentage error(sMAPE): 2.89%

In [16]: knn_predictions

Out[16]: array([1936.4, 1906. , 1936.4, 1825.8, 1784.4, 1565.6, 1687.2, 1543.6,
                1650. , 1543.6, 1553.6, 1568.8])

In [17]: from sklearn.neural_network import MLPRegressor
         from sklearn.datasets import make_regression
         from sklearn.model_selection import train_test_split
         # X, y = make_regression(n_samples=200, random_state=1)
         # X_train, X_test, y_train, y_test = train_test_split(X, y,
         regr = MLPRegressor(random_state=42)

In [18]: param_grid = {
             'hidden_layer_sizes': [(50,), (100,), (50, 50), (100, 50, 25), (100, 100, 50, 25)],
             'activation':['relu', 'tanh'],
             'alpha': [0.0001, 0.001, 0.01],
             'learning_rate': ['constant', 'invscaling', 'adaptive'],
```



```
             'max_iter': [1000, 2000, 3000]
         }

         grid_search = GridSearchCV(estimator=regr, param_grid=param_grid, cv=5, scoring='neg_mean_squared_error', n_jobs=-1)

         # Fit the grid search object to the training data
         grid_search.fit(X_train, y_train)

         # Get the best hyperparameters and the best estimator from the grid search
         best_params = grid_search.best_params_
         best_estimator = grid_search.best_estimator_

         # Evaluate the best estimator on the test data
         ann_predictions = best_estimator.predict(X_test)

In [19]: ann_predictions

Out[19]: array([1831.84363739, 1769.0710101 , 1823.831087  , 1818.0354725 ,
                1735.50153829, 1733.6688791 , 1733.10041646, 1702.71099512,
                1760.78967848, 1696.29962189, 1702.27840008, 1711.60872938])

In [20]: mae = mean_absolute_error(y_test, ann_predictions)

         # Print or use the MAE value
         mse = mean_squared_error(y_test,ann_predictions)
         rmse = np.sqrt(mse)
         mape = np.mean(np.abs((y_test - ann_predictions) /y_test)) * 100
         r2 = r2_score(y_test, ann_predictions)
```



```
         def symmetric_mean_absolute_percentage_error(y_test, ann_predictions):
             # Avoiding division by zero
             y_test, ann_predictions = np.array(y_test), np.array(ann_predictions)
             mask =y_test!= 0
             y_test, ann_predictions = y_test[mask],ann_predictions[mask]

             # Calculate sMAPE
             smape = 2 * np.mean(np.abs(ann_predictions - y_test) / (np.abs(ann_predictions) + np.abs(y_test))) * 100

             return smape

         # Calculate sMAPE
         smape_value = symmetric_mean_absolute_percentage_error(y_test, ann_predictions)

         print(f"Mean Squared Error: {mse:.2f}")
         print(f"Root Mean Squared Error: {rmse:.2f}")
         print(f"R-squared: {r2:.2f}")
         print(f"Mean Absolute Error (MAE): {mae:.2f}")
         print(f"Mean Absolute Percentage Error (MAPE): {mape:.2f}%")
         print(f"Symmetric Mean absolute percentage error(sMAPE): {smape_value:.2f}%")

         Mean Squared Error: 18165.37
         Root Mean Squared Error: 134.78
         R-squared: 0.42
         Mean Absolute Error (MAE): 123.10
         Mean Absolute Percentage Error (MAPE): 7.28%
         Symmetric Mean absolute percentage error(sMAPE): 7.14%
```

In [21]:
```python
import numpy as np
from scipy.optimize import minimize
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.neighbors import KNeighborsRegressor
# Combine the predictions into a list
model_predictions = [svr_predictions, rf_predictions, knn_predictions, ann_predictions]

# Concatenate the predictions from all models for the test set
def combined_predictions(weights, *args):
    model_predictions = args
    return np.dot(weights, model_predictions)

# Objective function to minimize (RMSE)
def objective_function(weights, *args):
    y_test, model_predictions = args
    combined_pred = combined_predictions(weights, *model_predictions)
    rmse = np.sqrt(mean_squared_error(y_test, combined_pred))
    return rmse

# Constraints for weights
constraints = ({'type': 'eq', 'fun': lambda w: 1 - sum(w)})

# Initial guess for weights
initial_weights = np.ones(len(model_predictions)) / len(model_predictions)
```
```python
# Minimize the objective function
result = minimize(objective_function, initial_weights, args=(y_test, model_predictions), constraints=constraints, bounds=[(0, 1)]

# Extract the optimized weights
optimized_weights = result.x

# Combined prediction using optimized weights
final_combined_pred = combined_predictions(optimized_weights, *model_predictions)

# Evaluate the final RMSE
print(final_combined_pred)
final_rmse = np.sqrt(mean_squared_error(y_test, final_combined_pred))
final_mae = np.mean(np.abs(y_test - final_combined_pred))
print(f'Optimized Weights: {optimized_weights}')
print(f'Final RMSE: {final_rmse}')
print(f'Final MAE: {final_mae}')
```
```
[1976.62998909 1788.02161375 1954.04652081 1915.11623625 1674.91940179
 1636.86413299 1657.33551027 1544.91998603 1724.38064044 1527.25626612
 1544.51497364 1572.09134641]
Optimized Weights: [7.82661769e-01 1.62285101e-11 2.17338230e-01 3.41233208e-11]
Final RMSE: 32.88684486699511
Final MAE: 25.237741962950793
```

In [22]:
```python
import numpy as np
from scipy.optimize import minimize
from sklearn.metrics import mean_absolute_error, mean_squared_error
```
```python
# Objective function to minimize (MAE)
def objective_function(weights, *args):
    y_test, model_predictions = args
    combined_pred = np.dot(weights, model_predictions)
    mae = np.mean(np.abs(y_test - combined_pred))
    return mae

# Additional constraint
def additional_constraint(weights, y_test, model_predictions):
    combined_pred = np.dot(weights, model_predictions)
    rmse = np.sqrt(mean_squared_error(y_test, combined_pred))
    return 32.88684486699511-rmse

# Constraints for weights
constraints = [{'type': 'eq', 'fun': lambda w: 1 - sum(w)},
               {'type': 'ineq', 'fun': lambda w: w},
               {'type': 'ineq', 'fun': lambda w: additional_constraint(w, y_test, model_predictions)}]

# Initial guess for weights
initial_weights = np.ones(len(model_predictions)) / len(model_predictions)

# Minimize the objective function
result = minimize(objective_function, initial_weights, args=(y_test, model_predictions), constraints=constraints)

# Extract the optimized weights
optimized_weights = result.x

# Combined prediction using optimized weights
```

```python
final_combined_pred = np.dot(optimized_weights, model_predictions)

# Evaluate the final MAE and RMSE
final_mae = np.mean(np.abs(y_test - final_combined_pred))
final_rmse = np.sqrt(mean_squared_error(y_test, final_combined_pred))

# Print the results
print(f'Optimized Weights: {optimized_weights}')
print(f'Final MAE: {final_mae}')
print(f'Final RMSE: {final_rmse}')
```

```
Optimized Weights: [ 7.82705131e-01 -6.47639904e-10  2.17294870e-01 -1.62399527e-10]
Final MAE: 25.237098593955086
Final RMSE: 32.886844986486956
```