This study presents the development and evaluation of five distinct Machine Learning models applied to a structured dataset, with the objective of identifying the most accurate and computationally efficient approach. The goal is the best approach for forecasting the Power Demand of the building. The dataset underwent rigorous preprocessing, including feature engineering and feature selection. These steps were essential to enhance model performance and reduce dimensionality. Table 1 presents the final dataset obtained after the complete data preprocessing pipeline, which included data cleaning, feature engineering, and feature selection.

Table 1 - Dataset Summary

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Subject** | **Total Features** | **Name** | **Type** | **Unit** | **Description** | **Start date** | **End date** | **Resolution** | **Source** |
| Power Demand | 1 | Target variable | Float | kW | Power demand of the building | 1/1/22 | 12/31/23 | hourly | Sensors in the building |
| Weather (outdoor) | 1 | temperature\_2m | Float | °C | temperature at 2m | 1/1/22 | currently | hourly | Open-Meteo |
| Feature Engineering | 6 | month | Integer | 1 to 12 | Indicates the month of the record | 1/1/22 | currently | hourly | Feature Engineering |
| hour | Integer | 0 to 23 | Indicates the hour of the record | 1/1/22 | currently | hourly | Feature Engineering |
| day\_of\_week | Integer | 0 to 6 | Indicates the day of the week with integer numbers | 1/1/22 | currently | hourly | Feature Engineering |
| hour\_sin | Float | - | cyclic variables to pass the information of periodicity to the models | 1/1/22 | currently | hourly | Feature Engineering |
| hour\_cos | Float | - | cyclic variables to pass the information of periodicity to the models | 1/1/22 | currently | hourly | Feature Engineering |
| day\_sin | Float | - | cyclic variables to pass the information of periodicity to the models | 1/1/22 | currently | hourly | Feature Engineering |
| 48 | Lag Features | Float | kWh | used to capture the temporal dependencies and patterns in time series data for the last 48hrs | 1/1/22 | currently | hourly | Feature Engineering |
| 6 | Hist Features | Float | kWh | used to capture the temporal dependencies and patterns in time series data (same time for 3,4,5,6,7, and 14 days ago) | 1/14/22 | currently | hourly | Feature Engineering |
| 2 | is\_closed\_holiday | Integer | binary | Binary characteristic (0 or 1) indicating whether it is a holiday that the mall is closed | 1/1/22 | currently | hourly | Feature Engineering |
| is\_open\_holiday | Integer | binary | Binary characteristic (0 or 1) indicating whether it is a holiday that the mall is open | 1/1/22 | currently | hourly | Feature Engineering |

Among the evaluated models, the following were implemented: Linear Regression, Random Forest, Support Vector Regression (SVR), Artificial Neural Networks (ANN), and Extreme Gradient Boosting (XGBoost) [1]. Hyperparameter tuning was conducted using the Optuna optimization framework [2] in Python [3], which allowed for an efficient and automated search of the optimal parameter configurations. The use of Optuna provided significant advantages, including faster convergence, support for pruning unpromising trials, and seamless integration with scikit-learn pipelines [4].

Figure 1 illustrates the top 20 most important features identified by the XGBoost model, highlighting the variables that contributed most significantly to predictive performance. This insight is crucial for understanding the underlying data structure and for guiding future data collection efforts.

A bar graph with blue bars

AI-generated content may be incorrect.

Figure 1 - Top 20 features

Table 2 presents a comprehensive comparison of model performance using standard error metrics: R², RMSE, MSE, MAE, and MAPE. Additionally, the table includes training and inference times for each model, providing a holistic view of both predictive accuracy and computational efficiency.

Table 2 - Performance and error metrics summary.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **R2** | **RMSE** | **MSE** | **MAE** | **MAPE** | **Total Training time [seconds]** | **Total Prediction time [seconds]** |
| XGBoost | 0.9811 | 82.77 | 6850.90 | 46.91 | 6.19% | 4.53 | 0.018 |
| Linear Regression | 0.9556 | 126.81 | 16081.27 | 79.58 | 10.37% | 0.03 | 0.001 |
| Random Forest | 0.9774 | 90.55 | 8199.67 | 45.48 | 5.68% | 40.66 | 0.018 |
| ANN | 0.9676 | 108.41 | 11753.46 | 66.10 | 9.36% | 80.26 | 0.009 |
| SVR | 0.9536 | 129.66 | 16810.93 | 75.77 | 8.89% | 10.35 | 0.732 |

To further assess model behavior, Figure 2 through Figure 6 display scatter plots of prediction errors for each model. These visualizations are instrumental in identifying patterns of underestimation or overestimation, heteroscedasticity, and potential outliers, thereby offering a deeper understanding of model reliability across the prediction range.

A graph of a graph of power

AI-generated content may be incorrect.

Figure 2 - Scatter plot

A graph of a graph of power

AI-generated content may be incorrect.

Figure 3 - Scatter plot

A graph of a power diagram

AI-generated content may be incorrect.

Figure 4 - Scatter plot

A graph of a power supply line

AI-generated content may be incorrect.

Figure 5 - Scatter plot

A graph of a graph of power

AI-generated content may be incorrect.

Figure 6 - Scatter plot

Moreover, Figure 7 presents boxplots of the RMSE dispersion for all models. Boxplots are particularly valuable for comparing the dispersion of model errors, providing a robust visual summary of model stability and consistency.

A graph of blue squares and lines

AI-generated content may be incorrect.

Figure 7 - Boxplot of RMSE comparison

In conclusion, the XGBoost model emerged as the most effective solution, achieving the best performance across all error metrics while maintaining low computational cost during both training and inference. These results underscore the suitability of XGBoost for high-performance predictive modeling in structured data scenarios.

1. Chen, T., Guestrin, C.: XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794. Association for Computing Machinery, San Francisco, California, USA (2016)

2. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A Next-generation Hyperparameter Optimization Framework. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2623–2631. Association for Computing Machinery, Anchorage, AK, USA (2019)

3. Van Rossum, G., Drake, F.L.: Python reference manual. Centrum voor Wiskunde en Informatica Amsterdam (1995)

4. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V.: Scikit-learn: Machine learning in Python. the Journal of machine Learning research 12,2825-2830 (2011)