Stage 4 ML Project

Thomas DANIEL Théo CHICHERY Jean MARCHEGAY Julien DE VOS

December 26, 2024
DIA 3

Contents

1. Introduction

In this document you will find a short summary of the work we did as a team during this semester. Our work was divided into three stages. The first one was dedicated to data analysis and simple models implementations. The second one explored the use of more advanced models and data pre-processing techniques and the last one made use of more complex models such as neural networks.

2. Stage 1

In stage 1 we mainly preprocessed the data and made the first predictions.

2.1. Implementations

2.1.1 Data quality

First we looked at the quality of our data by looking at the following points:

- **Data statistic and graphical distributions**: First, we look at the statistics of our data and their distribution with graphs to give us an initial idea of our data.
- **Missing values**: We look at the number of missing values in our dataset. We see that there are no missing values, so we can move on to the next step.
- **Outliers**: We check our data for outliers, using a boxplot. We notice high values for WS10M, but this can be explained by the fact that the wind can reach very high speeds at certain times. This shows that there are no obvious outliers in our data. We therefore decide not to modify our data with regard to our outliers.
- **Correlation matrix**: Finally, we create a correlation matrix to see how closely variables are related to each other. We see, for example, that GD(i) Gb(i) HSun and P are strongly correlated, which is logical since P represents the output of solar panels, which are therefore closely linked to the sun and its position.

2.1.2 Predictions

For these first predictions, we have chosen to use two simple models: linear regression and KNN.

We decide to predict residual load directly, and to avoid overfitting, we decide to use a cross validation of 5.

For the hyperparameters, we looked at the number of optimal neighbors for the KNN that gave us the best results. We then tested from 20 to 1000 several possibilities and found that the best parameter was 100.

Finally, we use two different metrics:

- **R2**: This metric is the easiest to understand and allows you to effectively compare two models.

- **RMSE**: The rmse is the metric used for the kaggle results and gives us a good idea of the quality of our model.

2.2. Results

We obtain the following results :

Model	**5-Folds RMSE**	**R2 Score**
Linear Regression	-64.71	0.748
KNN (k = 100)	-62.95	0.765

Table 1: Summary of Model Results and Performance

Overall,the KNN model with k=100 is the best choice due to its superior balance between high predictive accuracy (R^2) and low error (Neg RMSE).

3. Stage 2

During Stage 2, we focused on building upon the findings and limitations identified in the previous methods to enhance our approach. The key contributions of this stage include data processing improvements, advanced modeling techniques, and detailed evaluations.

3.1. Data Processing

We refined the dataset by exploring univariate and multivariate approaches:

- **Univariate Analysis:** Each variable was analyzed independently to uncover individual trends and patterns.
- **Multivariate Analysis:** Multiple variables were considered simultaneously to capture interactions and dependencies, leveraging features such as temporal and weather variables.

3.2. Model Implementations

To achieve more robust predictions, the following advanced modeling techniques were explored:

- **Tree-Based Models:** Algorithms like XGBoost and LightGBM were employed for their ability to model non-linear relationships and handle feature interactions effectively.
- **Deep Learning Techniques:** Neural network architectures, including LSTM models, were implemented to capture temporal dependencies inherent in the data. Specific efforts included sequence generation for time-series data and the use of dropout layers to mitigate overfitting.

3.3. Iterative Model Refinement with the Repeat Model Function

A critical innovation introduced in Stage 2 was the implementation of the `repeat_model` function, designed to iteratively refine predictions by identifying and removing outlier data points based on the model's performance. This approach enhances the overall dataset quality and improves model accuracy.
The `repeat_model` function operates as follows:

1. **Initialization:** The function accepts parameters such as the model, number of iterations, input features, target variable, dataset, and an error multiplier.
2. **Prediction and Error Calculation:** For each iteration, the model predicts the target variable based on the input features. The prediction error is computed as the squared difference between the predicted and actual values.
3. **Outlier Removal:** Rows with errors exceeding a threshold (determined by the average root mean squared error multiplied by the error multiplier) are removed from the dataset.
4. **Iteration:** The refined dataset is used in subsequent iterations, progressively reducing the influence of outliers on the model's learning process.

This iterative process ensures that the model focuses on high-quality data, avoiding biases introduced by extreme outliers or noise. The function is particularly valuable in time-series and regression tasks, where small deviations can significantly impact performance.

Benefits and Results The `repeat_model` function demonstrated the following benefits:

- **Enhanced Data Quality:** By iteratively removing high-error rows, the dataset became more representative of typical patterns.
- **Improved Model Performance:** Models trained using the refined dataset exhibited lower RMSE values, reflecting more accurate predictions.
- **Flexibility Across Models:** The function was tested with various models, including XGBoost, Random Forest, and Linear Regression, showcasing its adaptability.

This function represents a significant methodological enhancement in Stage 2, addressing the challenge of outlier data and improving the robustness of predictive models.

3.4. Evaluation and Insights

Key results from this stage demonstrated significant improvements:

- The Load-P approach with XGBoost/LightGBM achieved a top-3 leaderboard score on Kaggle, highlighting its robustness.
- Metrics such as RMSE and R^2 scores were used to quantify model performance, with clear advantages observed in multivariate methods.

Through the implementation of advanced models and refined data preparation strategies, this stage provided substantial groundwork for optimizing predictive accuracy and addressing prior limitations.

4. Stage 3

In Stage 3, we focused on optimizing existing algorithms while introducing new methods to improve residual load prediction. The primary advancements included implementing Temporal Convolutional Networks (TCN) and enhancing Convolutional Neural Networks (CNN) for both univariate and multivariate data processing.

4.1. Implementations

4.1.1 TCN

Temporal Convolutional Networks (TCNs) were introduced as a novel deep learning approach to analyze time-series data. The TCN model offers several advantages:

- **Memory Efficiency:** TCNs require less memory than LSTM models since each layer uses a single kernel, reducing resource demands.
- **Performance:** Parallelization enables TCNs to process data efficiently, achieving good results in capturing temporal dependencies.

Before using the TCN model, data sequences were generated to prepare inputs. The sequence creation process involved splitting the data into fixed-size windows, which helped the model learn temporal patterns effectively. For TCN layers, dilated convolutions were applied, enabling the model to capture long-term dependencies in the data.

4.1.2 CNN

Convolutional Neural Networks (CNNs) were adapted for residual load forecasting. Their structure allowed efficient extraction of complex patterns from multivariate data. Key benefits include:

- **Capturing Complex Patterns:** CNNs efficiently modeled non-linear interactions and temporal dependencies using convolutional layers.
- **Regularization Techniques:** Dropout layers and batch normalization improved the model's generalization and reduced overfitting risks.
- **Simplified Preprocessing:** Unlike TCNs, CNNs do not require explicit sequence creation, simplifying the implementation process.

CNN models were also tested with the `repeat_model` function from Stage 2, demonstrating substantial improvements in prediction accuracy, particularly for the variable P.

4.2. Limitations

TCN:

- **Domain Transfer Challenges:** TCNs struggle with adapting from short to long data histories, which may affect performance on datasets with varied temporal characteristics.

CNN:

- **Data Requirements:** CNNs perform better with larger datasets due to their complexity.
- **Long-Term Dependencies:** While effective for short-term temporal patterns, CNNs are less suited for capturing long-term dependencies compared to RNNs or LSTMs.

4.3. Results Obtained

The performance of models implemented in Stage 3 highlighted the impact of advanced methodologies and preprocessing techniques:

- **CNN with `repeat_model`:** This configuration achieved significant improvements, obtaining an RMSE of 23.04 for P and 21.4 for load predictions, ranking among the top 4 on the Kaggle leaderboard with a score of 48.
- **CNN without `repeat_model`:** Without preprocessing enhancements, the RMSE was higher at 40.02 for P and 22.31 for load predictions, demonstrating the importance of data refinement.
- **TCN:** While promising in theory, the TCN model achieved an RMSE of 42.2 and a Kaggle score of 62.7, suggesting challenges in model adaptation and computational resource limitations.
- **Comparison with Stage 2 Models:** Despite the advanced capabilities of TCN and CNN, the XGBoost/LightGBM approach from Stage 2 remained the best performer with an RMSE of 17.01 for P and 23.77 for load predictions.

These results underscore the importance of balancing model complexity with data preprocessing and hyperparameter tuning to achieve optimal performance.

5. Potential Improvements

While the models implemented in Stage 3 showcased notable advancements, there are several avenues that could be explored to further improve the results. These include:

5.1. Model Refinements

- **Hyperparameter Optimization:** A more exhaustive search for optimal hyperparameters using techniques like Bayesian optimization or random search could improve model performance and prevent overfitting.

- **Enhanced Neural Network Architectures:** Incorporating attention mechanisms, transformers, or hybrid models combining CNNs and LSTMs could capture both temporal and spatial dependencies more effectively.
- **Advanced Regularization Techniques:** Techniques such as L2 regularization, weight decay, or dropout with varying rates could help mitigate overfitting in deep learning models.

5.2. Data Preprocessing

- **Feature Engineering:** Introducing additional features, such as derived weather indices, lag features, or rolling averages, could provide more predictive power.
- **Handling Missing Data:** Employing advanced imputation methods, such as matrix factorization or model-based approaches, could improve data quality.
- **Sequence Length Optimization:** Experimenting with different sequence lengths for TCNs and LSTMs might better capture relevant temporal patterns without overloading the model.

5.3. Training and Validation Strategies

- **Cross-Validation with Time-Series Splits:** Implementing time-series-specific cross-validation, such as rolling or expanding window validation, could lead to more robust model evaluation.
- **Balanced Train-Test Split:** Ensuring balanced representation of different temporal patterns (e.g., seasonal or diurnal variations) in training and testing datasets could enhance model generalization.
- **Augmentation Techniques:** Synthetic data generation through time-series augmentation methods, such as jittering, cropping, or adding noise, could increase the effective training dataset size.

5.4. Alternative Models and Approaches

- **Probabilistic Models:** Approaches such as Gaussian Processes or Bayesian Neural Networks could quantify uncertainty in predictions.
- **Hybrid Models:** Combining tree-based models (e.g., XGBoost) with neural networks could leverage their respective strengths for structured and unstructured data.
- **AutoML Frameworks:** Using automated machine learning platforms could identify better-performing models and configurations with minimal manual intervention.

5.5. Data Quality and Acquisition

- **High-Resolution Data:** Acquiring data at higher temporal or spatial resolutions could provide finer-grained insights for model training.

- **Inclusion of External Variables:** Incorporating additional external data, such as real-time weather forecasts, regional grid loads, or solar irradiance predictions, could enhance model inputs.
- **Noise Reduction:** Applying signal processing techniques, such as wavelet transforms or Fourier filtering, could reduce noise and enhance signal clarity.

These improvements offer multiple pathways to build upon the existing work, balancing computational efficiency, model complexity, and data quality to achieve superior results.

6. Final Conclusion

This project showcased a wide range of models, from traditional machine learning methods to advanced deep learning techniques, highlighting their strengths and challenges in the context of residual load prediction. Models like XGBoost and LightGBM stood out for their robustness, ease of use, and ability to handle non-linear relationships. These models consistently delivered reliable results with minimal hyperparameter tuning, making them effective and efficient tools, especially when paired with well-preprocessed and engineered data.

Deep learning models brought a more nuanced approach to the table. Convolutional Neural Networks (CNNs) demonstrated their ability to capture patterns across multiple features, offering a simple yet powerful way to work with time-series data without requiring explicit sequence creation. On the other hand, Temporal Convolutional Networks (TCNs) excelled in modeling long-term dependencies through dilated convolutions, but their reliance on sequence preparation introduced additional complexity. While TCNs showed promise in capturing temporal patterns, their computational demands and sensitivity to hyperparameters made them less practical compared to CNNs in this project.

One of the most valuable insights gained was the importance of data preprocessing and feature engineering. Improvements in these areas, such as better handling of missing values, feature selection, and sequence generation, had a significant impact on model performance. Ultimately, while deep learning models offered advanced capabilities, simpler approaches like XGBoost emerged as reliable choices, particularly when computational efficiency and interpretability were key. Future work could focus on enhancing data quality, exploring more robust temporal models, and fine-tuning hyperparameters to unlock the full potential of these methods.

Accurate residual load predictions are essential for maintaining a stable and reliable energy supply as renewable energy systems, like solar panels, become more prevalent. The residual load, the difference between energy consumption and self-generated energy, depends heavily on weather conditions, making traditional forecasting methods insufficient. Precise forecasts enable energy suppliers to balance supply and demand effectively, ensuring sufficient energy during deficits and managing excess energy fed back into the grid. This supports grid stability, cost optimization, and the successful integration of renewables into the energy system, aligning with the goals of the energy transition.