DEEP ENSEMBLES APPROACH FOR ENERGY FORECASTING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ONUR ENGİNAR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
FINANCIAL MATHEMATICS

SEPTEMBER 2023

Approval of the thesis:

DEEP ENSEMBLES APPROACH FOR ENERGY FORECASTING

submitted by **ONUR ENGİNAR** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Financial Mathematics Department, Middle East Technical University** by,

Prof. Dr. A. Sevtap Selçuk-Kestel
Dean, Graduate School of **Applied Mathematics**

Prof. Dr. A. Sevtap Selçuk-Kestel
Head of Department, **Financial Mathematics**

Prof. Dr. Ömür Uğur
Supervisor, **Scientific Computing, METU**

Examining Committee Members:

Prof. Dr. A. Sevtap Selçuk-Kestel
Actuarial Sciences, METU

Prof. Dr. Ömür Uğur
Scientific Computing, METU

Prof. Dr. Kazım Barış Atıcı
Business Administration, Hacettepe University

Prof. Dr. Ceylan Yozgatlıgil
Department of Statistics, METU

Prof. Dr. Ümit Aksoy
Department of Mathematics, Atılım University

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: ONUR ENGİNAR

Signature :

ABSTRACT

DEEP ENSEMBLES APPROACH FOR ENERGY FORECASTING

Enginar, Onur
Ph.D., Department of Financial Mathematics
Supervisor : Prof. Dr. Ömür Uğur

September 2023, 104 pages

In this thesis study, we develop a novel deep ensembles based architecture that enables transfer learning to reduce the time requirement of deep ensembles without compromising the model's accuracy. We apply our model to open energy datasets. Moreover, this thesis compares SoTA tabular learning models with deep ensembles and traditional machine learning models and provides a benchmark for the literature. We further develop a feature selection algorithm based on boosted deep ensembles model and compare it with linear feature selection models and tree-based feature selection algorithms.

Keywords: Deep Learning, Energy Forecasting, Tabular Learning, Feature Selection

ÖZ

ENERJİ TAHMİNLEMEDE DERİN TOPLULUKLAR YAKLAŞIMI

Enginar, Onur
Doktora, Finansal Matematik Bölümü
Tez Yöneticisi : Prof. Dr. Ömür Uğur

Eylül 2023, 104 sayfa

Bu tez çalışmasında, model performansından ödün vermeden derin toplulukların (deep ensembles) zaman gereksinimini azaltmak için transfer öğrenmesini kullanarak yenilikçi derin topluluklar (deep ensembles) modeli geliştirilmiştir. Geliştirilen model açık veri setlerine uygulanmıştır. Ayrıca, bu çalışmada, derin topluluklar (deep ensembles) modeli, literatürde tabular öğrenmeye özel geliştirilmiş güncel modeller ile kıyaslanarak sonuçları raporlanmıştır. Son olarak, derin topluluklar (deep ensembles) tabanlı öz-nitelik seçim algoritması geliştirilerek, bu yöntem doğrusal öz-nitelik çıkarma modelleri ve ağaç tabanlı öz-nitelik çıkarma modelleri ile kıyaslanmıştır.

Anahtar Kelimeler: Derin Öğrenme, Enerji Tahminleme, Tabular Öğrenme, Öznitelik Çıkarma

To my lovely family.

ACKNOWLEDGMENTS

I sincerely thank my supervisor, Prof. Ömür Uğur, for his patience, support, and belief in this thesis study. He provided feedback for almost every line of the thesis. With his invaluable help, I am able to finish this thesis.

I also would like to thank my thesis committee members, Prof. A. Sevtap Selçuk-Kestel and Prof. Mehmet Baha Karan, for pushing me hard to finish this study. I deeply appreciate Prof. Karan for his continuous support for my career.

I would also like to thank my father, mother, and brother for their sincere support.

I would also like to thank my friends - a long list of names - for their never-ending support, care, good company, and rave parties.

Last but not least, I would like to express my gratitude to my lovely wife, who makes days brighter.

TABLE OF CONTENTS

ABSTRACT vii

ÖZ ix

ACKNOWLEDGMENTS xiii

TABLE OF CONTENTS xv

LIST OF TABLES xix

LIST OF FIGURES xxi

LIST OF ALGORITHMS xxiii

LIST OF ABBREVIATIONS xxv

CHAPTERS

1 INTRODUCTION 1

2 USE CASES OF MACHINE & DEEP LEARNING MODELS IN ENERGY 3

2.1 Energy Forecasting 3

2.2 Optimization in Energy 6

2.3 Various Other Use Cases 8

3 A REVIEW ON SUPERVISED LEARNING 11

3.1 Notation 11

3.2 Machine Learning Models for Tabular Data 12

3.3 Deep Learning Models 15

3.3.1 Tabular Learning 16

4 BOOSTED DEEP ENSEMBLES: A NOVEL ARCHITECTURE BASED ON DEEP ENSEMBLES 23

4.1 Introduction 23

4.2 Background 25

4.3 Deep Ensembles 28

4.3.1 Model Structure of Deep Ensembles 28

4.3.2 Boosted Deep Ensembles 32

4.3.3 Complexity of Boosted Deep Ensembles 35

4.4 Experiments on Public Datasets 37

4.5 Experiments 40

4.6 Results 41

4.6.1 Hypothesis Testing 44

4.7 Robustness Checks 46

4.8 Ensemble selection 48

4.9 Conclusion and Further Study 51

5 APPLICATION: COMPARISON OF TABULAR LEARNING METHODS FOR SHORT-TERM LOAD FORECASTING IN SPAIN 53

5.1 Introduction 53

5.2 Literature on Short Term Load Forecasting 54

5.2.1 Point Forecasting 55

5.2.2 Probabilistic forecast 57

5.3 Analysis of Energy Load Dataset 58

5.3.1 Dataset 58

5.4 Experimental Methodology 62

5.5 Experiments 65

5.6 Results 66

5.6.1 Pairwise Model Ensembling 68

5.7 Model Ablation Study 69

5.8 Conclusion and Further Study 71

6 A NOVEL FEATURE SELECTION FRAMEWORK WITH DEEP ENSEMBLES 73

6.1 Introduction 73

6.2 Related Work 74

6.2.1 Physical Model 75

6.2.2 Statistical Models 75

6.2.3 Machine Learning Models 75

6.2.4 Hybrid Models 76

6.2.5 Deterministic and Probabilistic Models 76

6.2.6 Dimension Reduction 76

6.2.7 Feature Extraction 77

    - 6.2.8 Feature Importance 77
    - 6.2.9 Feature Selection in Wind Power Forecasting . . . 78
  - 6.3 Model 78
  - 6.4 Dataset 80
  - 6.5 Application 81
  - 6.6 Results of Feature Importance Algorithms 82
  - 6.7 Conclusion and Further Study 85
- 7 CONCLUSION 87

REFERENCES 89

CURRICULUM VITAE 103

LIST OF TABLES

Table 3.1 Summary Table for Data Structures 19

Table 4.1 Variables of Load Dataset 38

Table 4.2 Descriptive Analysis of Load Dataset 38

Table 4.3 Descriptive Analysis of Load Dataset 38

Table 4.4 Variables of Wind Dataset 39

Table 4.5 Descriptive Analysis of Wind Dataset 39

Table 4.6 Variables of Solar Dataset 39

Table 4.7 Descriptive Analysis of Solar Dataset 40

Table 4.8 Forecast Results for Load Forecasting 42

Table 4.9 Forecast Results for Wind Generation Forecasting 42

Table 4.10 Forecast Results for Radiation Forecasting 42

Table 4.11 Results of Hypothesis Tests for Datasets 45

Table 4.12 Change of Hyperparameters 46

Table 4.13 Forecast Results for Load Forecasting with Hyperparameters 46

Table 4.14 Forecast Results for Load Forecasting under Different Model Stuctures 46

Table 4.15 Forecast Results for Load Forecasting under Different Base Model Structures 47

Table 4.16 Results of Model Selection for Load Forecasting 49

Table 5.1 Results for point forecast 66

Table 5.2 Parameters tuned 67

Table 5.3 Parameters tuned 67

Table 5.4 Parameters tuned results for point forecast 67

Table 5.5 Results for ensemble with FTT model 68

Table 5.6 Results for ensemble with TabNet model 68

Table 5.7 Results for ensemble with Xgboost model 68

Table 5.8 Results for ensemble with CNN model 68

Table 5.9 Results for ensemble with Deep Ensembles model 68

Table 5.10 Results for ensemble with TabTransformer model 68

Table 5.11 Results for ensemble with GANDALF model 69

Table 5.12 Results for ensemble with NODE model 69

Table 5.13 Results of Linear Regression and MLP model 70

Table 6.1 Descriptive statistics of selected variables 80

Table 6.2 Descriptive statistics of selected variables 80

Table 6.3 Names of Variables Used in Application 81

Table 6.4 Weighted mape comparison of feature importance and reduction algorithms 83

LIST OF FIGURES

Figure 4.1 Model Structure of Boosted Deep Ensembles 34

Figure 4.2 Results of Deep Ensemble (nll) Model 42

Figure 4.3 Results of Boosted Deep Ensemble Model 43

Figure 4.4 Results of Deep Ensemble (mse) Model 43

Figure 4.5 Results of XGBoost Model 44

Figure 4.6 Kernel Density Comparison of Deep Ensemble (mse) and Boosted Deep Ensemble Model 45

Figure 4.7 Box Plot Comparison of Deep Ensemble (mse) and Boosted Deep Ensemble Model 45

Figure 4.8 Time and Accuracy Comparison of Different Structure of Boosted Deep Ensemble 47

Figure 4.9 Learning Curves of Boosted Deep Ensembles with Different Depths 48

Figure 4.10 Accuracy with Different Number of Ensemble Members 49

Figure 4.11 Distribution of Mape Values of Ensemble Members 50

Figure 4.12 Histograms of Randomly Selected Predictions 51

Figure 5.1 Upward Trend 59

Figure 5.2 Downward Trend 59

Figure 5.3 Weekly Periodicity 60

Figure 5.4 Hourly Periodicity 60

Figure 5.5 Correlation of Spain Energy Demand 61

Figure 5.6 New Year’s Week 61

Figure 5.7 Partial-autocorrelation within a day 62

Figure 5.8 t-SNE Reducted Raw Dataset 70
Figure 5.9 t-SNE Reducted Embeddings 70

Figure 6.1 Input Variable Correlations 83
Figure 6.2 Variable Drop Feature Importances 84
Figure 6.3 Variable Permutation Feature Importances 84

LIST OF ALGORITHMS

Algorithm 1 Basic Bagging Algorithm 13
Algorithm 2 Deep Ensembles Algorithm 29
Algorithm 3 Boosted Deep Ensembles Algorithm 33
Algorithm 4 Calculating feature importance with variable drop 79
Algorithm 5 Calculating feature importance with variable permutation . . . 79

LIST OF ABBREVIATIONS

ML	Machine Learning
NN	Neural Network
CNN	Convolutional Neural Network
BNN	Bayesian Neural Network
RNN	Recurrent Neural Network
DNN	Deep Neural Network
MLP	Multi Layer Perceptron
XGBoost	Extreme Gradient Boosting
GBDT	Gradient Boosting Decision Trees
SoTA	State-of-the-art
t-SNE	t-Distributed Stochastic Neighbor Embedding
MSE	Mean Squared Error
EMS	Energy Management System
NILM	Non-Intrusive Load Monitoring
SQL	Structured Query Language
ERP	Enterprise Resource Planning
OLTP	Online Transactional Processing
CPU	Central Processing Unit
GPU	Graphics Processing Unit
MCMC	Markov Chain Monte Carlo
GANDALF	Gated Adaptive Network for Deep Automated Learning of Features
NODE	Neural Oblivious Decision Ensembles
FT-T	Feature Tokenizer Transformer

CHAPTER 1

INTRODUCTION

Energy forecasting is a long-studied subject with various subdomains, including demand forecasting, generation forecasting, and energy price forecasting. These problems arise mainly for two reasons. First, energy companies aim to minimize uncertainties. Second, system operators and market authorities set rules to regulate energy transmission and sustain energy markets efficiently. Both energy companies and authorities rely on accurate energy forecasts to securely maintain their operations. Probabilistic forecasting is more informative on managing risks since it provides information about how point forecasts might deviate.

Current studies use machine and deep learning models in forecasting tasks. A deep ensemble model [95], which efficiently approximates predictive uncertainty, is a promising approach for energy forecast tasks [39, 101, 102, 93]. Deep ensembles generates results by averaging the collection of neural networks as base learners. In applications, the number of base learners is usually small and is between 5-10. Increasing the number of base learners increases the model accuracy, yet doing so makes the application of deep ensemble model computationally complex. To solve this, in Chapter 4, we propose a novel deep ensembles architecture that sufficiently reduces time requirements by using transfer learning.

Contributions of Chapter 4 are the following: We investigate ensemble methods in machine learning and deep learning literature and their probabilistic applications. We propose a parallelizable transfer learning enabled deep ensemble model that runs significantly faster than plain deep ensembles and generates distributional equivalent prediction with deep ensembles and extensively test it on energy datasets, which are tabular learning tasks and provide statistical results. When trained with a sufficient amount of base learners, the boosted deep ensemble model generates normally distributed predictions when using mse as a loss function where using negative log-likelihood hurts point forecasting accuracy. We see that decorrelated predictions are insufficient to generate a superior model; instead, leaving out poor-performing base learners generates superior forecasts.

As energy forecasting is a subclass of tabular learning, we investigate SoTA tabular learning models [80, 3, 128] which are dominantly based on transformer architecture, and compare those models with XGBoost, a well-known machine learning model, a plain CNN model and deep ensemble model in Chapter 5. We see that the deep ensembles, the CNN model, and Xgboost beats SoTA transformer-based tabular learning models are well suited for load forecasting tasks. Ensembling with XGBoost significantly boosts the performance of each model. We further inspect embedding space with the t-SNE algorithm [152] and perform a model ablation study by which we conclude transformer-based models are not able to represent categorical and numerical features as embeddings and hence result in poor performance.

Due to the transformer architecture, tabular learning models could extract features from the dataset. In order to contribute to explainable AI, we develop a feature selection framework for the deep ensemble model in Chapter 6. We integrate variable drop and variable permutation feature importance algorithms into the deep ensembles and test the accuracy of those algorithms against benchmark models.

The remainder of this thesis study is organized as follows. Chapter 2 introduces various use cases of the machine and deep learning in the energy domain. Chapter 3 defines supervised learning models, tabular learning, and literature on tabular learning and sets notation used therein. In Chapter 4, we develop the boosted deep ensemble model. In Chapter 5, we thoroughly analyze energy load forecasting and benchmark SoTA tabular learning models compared with CNN, XGBoost, and deep ensembles. In Chapter 6, we build and test a feature selection framework based on the deep ensemble model. Finally, we provide a brief discussion and conclusion in Chapter 7.

CHAPTER 2

USE CASES OF MACHINE & DEEP LEARNING MODELS IN ENERGY

Digital transformation of energy has excelled in recent years, and as a result, the energy sector has become a sensor-rich environment. Through IoT devices, an enormous amount of data is being collected in databases, and naturally, machine and deep learning algorithms are applied to those datasets. This section introduces prominent use cases of machine and deep learning in the energy sector.

2.1 Energy Forecasting

Forecasting-based energy use cases such as generation, demand, and price forecasting are to be briefly introduced in the sequel.

Energy Generation Forecasting

Accurate prediction of renewable power generation is required and beneficial for energy market participants such as energy generators, trading companies, dispatchers, and system and market operators. The system operators need accurate predictions to manage the reserve and to maintain the grid operations securely. Accurate and reliable predictions of renewable power generation are essential for electricity transmission and are also essential for competitive renewable energy supply. Moreover, accurate predictions allow utilities favorable trading performance on the electricity markets. In literature, there are mainly three methods, namely physical models, statistical models, and machine learning models and their combinations to perform wind power predictions. Generally, data-driven wind power forecasting is performed via numerical weather prediction (NWP), which is used as input variables to map wind plant generation. NWP is provided from spatially close points to the turbines as zonal and meridional components of winds, and necessary calculations are made from NWP data such as wind speed and direction. Generally, more than one point is selected for each turbine to ensure the wind profile is met for specific locations. However, this might result in redundant input variables for the model.

Electricity Load Forecasting

Energy demand/load forecasting is crucial and sometimes vital for governments and private companies. Government agencies use long-term demand forecasts to plan economic growth efficiently; thus, they can create social benefits. Moreover, banks and financial institutions utilize long-term forecasts for investment valuation. Private companies, such as energy distribution companies and system operators, require long-term and short-term forecasts to minimize their losses and maximize system operation security. Load forecasting is complex due to dependency on the quality of weather forecasts, exhibiting nonlinearity and multi-seasonality and social factors like holidays. Load forecasting deals with the prediction of energy demand in different periods. Depending on the time scale, load forecasting can be categorized into three types: long-term, medium-term, and short-term forecasting. Short-term load forecasting is considered for predicting the system load for one hour to one week, medium-term demand forecasting is for one week to one year, while the long-term load forecasting requires performing longer than one year. Moreover, very short-term load forecasting, 15 minutes to 1 hour, is usually made in smart grids to manage peak demand. In general, long-term load forecasting deals with power system planning required by government agencies; medium-term load forecasting deals with generation expansion planning and bilateral contracting. Short-term forecasting, on the other hand, is vital for maintaining the real-time operation of the power systems.

Electricity Price forecasting

Markets are the physical or non-physical places where sellers and buyers come together to exchange the goods (money, asset, commodity, or physical deliverable) they already have or do not have (i.e., future markets) with money or other goods. In financial markets, buyers and sellers interchange financial assets via mostly online tools; no physical delivery is required at the end of transactions. However, in electricity markets, sellers are required to deliver electricity physically. Also, electricity markets are different from financial markets in how they are operated in a daily setting. In day-ahead markets, market participants buy and sell electricity for the following day, and prices are determined each hour daily. After the day-ahead market closure, the auction in the balancing market starts to eliminate real-time imbalances in supply and demand in distribution. Then, the intraday market bridges the gap between the day-ahead and balancing markets, by a continuous auction in which buyers and sellers can trade one hour before delivery.

The electricity market has become increasingly complex and dynamic in recent years, with many factors affecting prices. These factors include changes in demand and supply, the integration of renewable energy sources, and the increasing use of distributed generation. Accurate forecasting of electricity prices is crucial for a wide range of stakeholders, including electricity generators, distributors, and even consumers. It can help power companies make informed decisions about generation and purchasing, and hence, enable consumers make more informed choices about their energy usage.

As new technologies and renewable energy opportunities emerge in electricity markets, managing market price risk in electricity day-ahead markets has become more challenging since hedging opportunities fall short. Therefore, forecasting electricity prices is the foremost important tool for market price risk management in electricity day-ahead markets.

Machine and deep learning models have recently become considerably popular in electricity price forecasting. Traditional statistical methods, which have been used for electricity price and load forecasting for a long time, are still used as benchmark models for performance comparison. Recently, interest in energy forecasting literature has shifted towards hybrid and ensemble models. These models are a combination of statistical and machine learning models; hence, they can produce more accurate forecasts than the individual one. However, in the related literature, there is no sound consensus on models which produce more accurate forecasts.

2.2 Optimization in Energy

As in many theoretical and practical fields, optimization becomes not only crutial but also handy.

Electricity Management Systems

An energy management system (EMS) is a tool for monitoring, optimizing, and controlling energy use in buildings and industrial facilities. EMS helps to reduce energy consumption, improve energy efficiency, and minimize the environmental impact of energy used in buildings.

In recent years, the increasing availability of sensor data, IoT devices, and the development of deep learning techniques have led to integrating deep learning methods into EMS modeling. Deep learning models could accurately predict energy demand or energy generation, and these predictions are used as inputs to optimize the energy supply in real-time, which helps to reduce energy costs.

Deep reinforcement learning emerges as one of the main methods to improve the ability of EMS to control and optimize energy usage. For example, deep reinforcement learning could be used to learn the optimal way to control building heating and air conditioning (HVAC) systems to optimize energy usage.

Battery Optimization

Battery technology is one of the fastest-growing emerging technologies in energy. Batteries store energy generated by renewable power plants, enabling excess energy whenever needed, including selling in the energy market or for self-use. Moreover, battery technology reduces the share of fossil fuels and makes the grid sustainable. In addition, the integration of battery technology into the grid makes it more efficient, reliable, and stable. Since batteries could be employed when there is grid instability or outages/disruptions, deep reinforcement learning is crucial in optimizing stored energy utilization. An agent could be trained to enable the acquisition of optimal strategies for managing battery charging and discharging in energy systems. By doing so, deep agents could determine the most efficient times to charge and discharge batteries, aiming to minimize energy costs and maximize the utilization of renewable energy. Furthermore, deep reinforcement learning is employed to optimize battery energy use while considering additional constraints, such as grid stability and environmental impact. For example, reinforcement models could also be trained to strike a balance between battery usage and utilization of other energy sources to minimize greenhouse gas emissions.

Energy Trade Optimization

The optimization problem of buying and selling the produced energy, which includes energy types diesel generation, natural gas, and renewable sources like solar and wind power, to minimize costs or maximize profits is called *energy trade optimization*. It also extends to optimizing energy exchanges between diverse geographic regions or participants in the market. Leveraging sophisticated optimization methodologies, energy trade optimization empowers energy producers, consumers, and traders to make well-informed decisions, thereby reducing costs associated with energy transactions. Traditional optimization techniques typically rely on predefined rules and constraints, which may hinder their adaptability to dynamic market conditions. In contrast, deep reinforcement learning facilitates more flexible and adaptive decision-making by enabling deep agents to learn from experiences and make choices based on the most up-to-date information. This empowers the deep agent to make precise and advantageous decisions regarding energy trading. Deep reinforcement learning also presents the potential for integrating multiple objectives and constraints into energy trade optimization. In the energy market, various objectives compete, such as cost minimization, profit maximization, and environmental impact reduction. By employing deep reinforcement learning, multiple objectives and constraints are incorporated simultaneously, making decisions that optimize the balance among these factors. Consequently, this enhances the efficiency and effectiveness of energy trade optimization.

Asset Management

Energy asset management is a strategic process of planning, coordinating, and overseeing the operation and maintenance of energy assets, which include a wide range of equipment, such as power plants, transmission networks, renewable energy sources, and energy storage systems. The main objective of energy asset management is to optimize the value of an organization's energy assets by ensuring their efficient, secure, and economical operation. Machine and deep learning models handle these tasks by predicting an asset's lifespan and cost. Unsupervised learning models, on the other hand, could provide an asset management framework as well.

2.3 Various Other Use Cases

Apart from the cases described above, there are various other energy use cases, and possibly, cannot be classified into a single categorization.

Energy Disaggregation

Energy disaggregation, or non-intrusive load monitoring (NILM), involves breaking down a household's electricity usage into specific appliance-level consumption data. This is especially useful for helping households understand and manage their energy use more efficiently, an essential aspect of sustainable energy policy. While various approaches, including hidden Markov models and non-negative matrix factorization, have been used to address the problems in NILM; a growing literature on supervised modeling with deep neural networks (DNNs) may be particularly effective at identifying the energy usage of different appliances. Moreover, clustering methods efficiently detect resistive subloads such as heaters and boilers.

Predictive Maintainance

Predictive maintenance uses mathematical models and the data generated from power plants to predict equipment failure times or require maintenance, which helps asset owners prevent equipment failures, reduce downtime, and improve a system's overall reliability and performances. Apart from the deterministic mathematical models, deep learning methodology has become one of the main tools for the performance measure and condition of equipment in a renewable power plant. This includes data on equipment temperatures, vibrations, and other performance metrics. Deep models predict equipment failure times or require maintenance, allowing for proactive maintenance and repairs. Moreover, specific neural networks with autoecoders could detect and analyze anomalies within renewable power facilities. To illustrate, these models can be trained to collect patterns within data that signal equipment malfunctions, enabling faster and more precise identification of the problems. This proactive approach can effectively minimize operational disruptions and enhance renewable power plants' overall dependability and efficiency.

Solar Fault Detection

Solar generation fault detection deals with identifying and diagnosing problems of a solar power generation system. This includes detecting problems with solar panels, inverters, and other system components. Fault detection is essential for ensuring a solar generation system's efficiency and reliability. It can also help prevent downtime and reduce the risk of equipment failures. Both supervised and unsupervised learning methods are employed in solar fault detection. Supervised classification models are applied also to anomaly labels in order to detect faults. Even further, clustering, as unsupervised learning models, helps detect faulty clusters as well.

Power Quality

Power quality problems deal with various challenges that could impact the quality of electrical power customers receive. These problems arise from various sources, such as equipment failures, inadequate power supply infrastructure, or natural disasters.

The consequences of power quality problems are diverse: equipment malfunction or failure decreased efficiency of electrical systems, and heightened vulnerability to power outages are some to mention. Commonly encountered power quality issues include voltage fluctuations, power surges, power sags, and harmonics. Anomaly detection models with machine and deep learning have become popular tools for detecting power quality issues.

CHAPTER 3

A REVIEW ON SUPERVISED LEARNING

Based on data availability, supervised learning is one of the most used learning algorithms. It generates a linear or nonlinear mapping between input and output datasets. Throughout time, complex architectures have been proposed by researchers [40]. Deep neural networks (NN) have been proven to achieve state-of-the-art results in a wide range of tasks, including image and video processing and speech recognition [135, 55]. In essence, neural networks are nonlinear systems for approximating some unknown functions over a parameter set. Even though numerous studies on tabular learning models have been proposed, no dominant structure such as Res-Net [67] emerged in image detection. Tabular data is used in many fields, such as medical records of patients, companies' financial records, or producers' manufacturing logs. Results from Kaggle competitions with tabular data suggest tree-based boosting algorithms are shown to be superior against deep learning models [11]. The work in [86] states that tabular data tasks are the last "unconquered castle" for deep neural network models.

This section sets the notation and defines well-known deep learning models and traditional machine learning algorithms used in tabular datasets. Further, we give literature on deep learning for tabular datasets.

3.1 Notation

In this section, we will be considering a regression problem, on pairs (X_i, Y_i) where $i = 0, 1, \ldots, n$ where input $X_i \in \mathbb{R}^d$ and output $Y_i \in \mathbb{R}$ with the function of interest $f(x) = \mathbb{E}[Y|X = x]$. As base learners on pairs (X_i, Y_i), we define $\hat{g}_i(\cdot)$ for $i = 0, 1, \ldots, m$. We consider decision trees for bagging and boosting, and consider shallow neural networks with three or less dense layers having 8 neurons at maximum for deep ensembles.

3.2 Machine Learning Models for Tabular Data

In this section, we introduce briefly the machine learning, tree based boosting and bagging models, used in learning tasks with tabular datasets.

Decision Tree Regression (DT)

Decision tree [50] regression that creates a tree like structure based on decision rules is the model that breaks the input space into subsets so that the weighted average of standard deviation of the subsets is smaller than that before the partition. When partition is completed, algorithm creates a tree composed of decision nodes and leaf nodes. Decision nodes are the nodes that at each node input space is partitioned and the resulting nodes are the leaf nodes where no further partition is required. Algorithm converges when there is no further gain in the standard deviated left or maximum depth of the tree is reached.

Bagging

Bagging (bootstapped aggregation) method reduces the instability of the estimator; that is it reduces the variance of an estimator. Bagging is an ensemble algorithm based on randomly resampling from original dataset and fitting a base procedure, usually decision tree or a simple function, to generate a number of estimators and then averaging. As meta-algorithms, the estimators result in a final estimate. See Algorithm 1. It is a very straight-forward method and usually performs quite well in applications.

Algorithm 1: Basic Bagging Algorithm

```
Input: Bootstrapped datasets $D = \{d_1, d_2, \ldots, d_M\}$, model $g$ to be fit
Output: Prediction of random forest
for $m \leftarrow 1$ to $M$ do
    fit base model $g$ on train set of $d_m$
end
return Average of predictions on test dataset
```

Random Forest Model

Random forest regression [71] is one of the most popular bagging methods, which resamples data with replacement to create bootstrapped series and, for each of these series, it randomly drops some of the input features with uniform probability and fit a decision tree in order to create decorrelated trees. Then, the algorithm averages the results over those bootstrapped series.

Boosting

Unlike bagging algorithm, which could be run in parallel, boosting algorithm, runs sequentially, to minimize a given loss function, iteratively by fitting base learners.

Gradient Boosting Trees (GBDT)

Gradient boosting regression [65] is a boosting model that fits a decision tree to residuals from previous base learners, and works in a sequential order to improve results from base learners.

Consider a GBM with M stages; where for each stage m, we define $g(x)$ base learner which is then fitted to $e_m = y - F_m(x)$, where F_m is the boosting model at step stage m and the goal is minimize the squared loss function via gradient descent algorithm.

More formally, for $\{x_i, y_i\}$ in the training set, we define

$$F(x) = \sum_{i=1}^{M} g_i(x) + g_0,$$

where g_i some base learner g_0 is the constant and ψ_i is the weight to be determined. Starting with a constant model g_0; at each step m, we compute

$$F_m(x) = F_{m-1}(x) + f_m(x)$$

where

$$f_m(x) = \text{argmin}_f \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + f(x_i))$$

Support Vector Regression (SVR)

Support vector regression [30] sets the decision line which minimizes the error terms which are out of the ϵ neighborhood around the decision line. For a given training set of $\{x_i, y_i\}$, we can define SVR as a optimization problem:

$$\text{minimize}\frac{1}{2}||w||^2 + C\sum_{i=1}^{N}(\xi^+ + \xi^-)$$

subject to

$$\begin{gathered} y_i - \phi(wx_i) - b \leq \epsilon + \xi^+ \\ \phi(wx_i) + b - y_i \leq \epsilon + \xi^- \\ \xi^+, \xi^- \geq 0 \quad \text{and} \quad i = 1, \ldots, N, \end{gathered}$$

where ϕ is a hyper-parameter which could be linear, polynomial of radial basis function, maps input space into lager space, C is the cost function, ξ^+ and ξ^- positive and negative error respect to ϵ neighborhood.

K-Nearest Neighborhood Regression (KNN)

KNN [49] regression is a non-parametric model which calculates the average of the k-nearest elements, based on some distance function defined, as regression output. Algorithm is quite straight forward. First, for each element, using a distance function (Euclidean, Manhattan, etc.) distances to other elements are calculated and sorted in ascending order. Then, based on predetermined number k, the average is calculated based on these nearest k elements.

3.3 Deep Learning Models

In this section, we define the fundamental structure of deep learning models, such as multilayer perceptron (feed forward neural network), convolutional neural network (CNN) and recurrent neural network (RNN). We also introduce the tabular learning and related literature.

Feed Forward Neural Networks (FNN)

Feed forward neural network [119] is a differentiable layered structure that maps input space into larger space by passing it through layers with nonlinear activation functions and finds the set of parameters $W_{i,l}$, where index i stands for the input i and l stands for the layer l, by minimizing the squared loss function via gradient descent algorithm.

Consider the data $\{x_i, y_i\}$ in the training set with n elements and a network with a single hidden layer, say h_1, with a single node n^1. We define a simple neural network

$$\hat{y} = h_1(z^1),$$

where $z^1 = \sum_{i=1}^{n} w_{1,i}^1 x_i + w_{1,0}^1$ and h_1 differentiable transformation. Now consider a network with two layer and with two node, we have

$$\hat{y} = h_2(z^2),$$

where $z^2 = \sum_{i=1}^{2} w_{2,i}^j z^j + w_{2,0}^j for j = 1, 2$ and $z^j = \sum_{i=1}^{n} w_{1,i}^j x_i + w_{1,0}^j$ and h_2 differentiable transformation. This layered structure could be generalized into l layers with n^l nodes at each layer l. Using gradient descent algorithm, the squared loss function is then minimized with respect to the weights w_l^n.

Convolutional Neural Network (CNN)

Convolutional neural network [96] is a powerful tool for image processing and classification, video recognition tasks. It is a sub-model, belongs to neural network family. Similar to multilayer perceptron, CNN is a layered structure, and at each layer it convolves input space with filters to extract features from it and thereby generates results.

Recurrent Neural Network (RNN)

Recurrent neural network, [113] is a class of neural networks for processing sequential data, which are used for modeling complex relations within the data sequences and prominently used for prediction tasks by scholars [55]. RNN, as the name suggests, recursively processes the previous values of the input series as a nonlinear transformation from input space to future of the input space:

$$\hat{y}_{t+1} = f(h_t, y_t)$$
$$h_t = g(y_t)$$

where functions f and g are the nonlinear transformations of the RNN model and h_t is the states generated by the model where $t = 1, \ldots, T$. and T is length of recurrent window.

3.3.1 Tabular Learning

Generally, datasets could be categorized as structured (heterogenous, tabular) and unstructured (homogenous). Tabular data is characterized by heterogeneous features with mixed input types collected from various sources integrated into relational databases. For example, consider data instances of purchase interaction of an individual with an e-commerce website. Structured data from this interaction could be as follows: demographic information, information on the frequency of the visits and interaction with the website, and date-related information about their visit. Collected data might have different types, such as categorical and numerical variables. Numerical data might be highly skewed or clustered, or some might need to be more reliable (e.g., answers to survey questions). A structured (heterogeneous) dataset is a data collection organized for easy searching, analyzing, and processing. Structured data has a form of rows and columns where each row represents a single record. These records contain a set of fields (columns) containing a value. This dataset type is used in business intelligence applications such as data mining to generate insights from datasets. Also, they can be used for analyzing recent trends and making predictions; they are also suitable for reporting and visualizations. Some examples of structured data include: customer records, product inventories, financial data, and sensor data.

Pros. Easily used by machine learning (ML) algorithms: due to its specific architecture, structured datasets are easy to query and manipulate; thus, these datasets could be easily used with machine learning models. Easily used by business users: structured data is easy to interpret and understand. A basic understanding of the data allows users to understand data easily. Accessible by more tools: since technologies for collecting unstructured datasets are premature relative to structured datasets, more tools are available for structured datasets.

Cons. Limited use: the flexibility and utility of data with a fixed format are constrained to the purpose for which it was developed. Limited storage options: data storage systems with a rigid schema (such as "data warehouses") are often used to save structured data. As a result, changes in data needs to entail an update of all structured data, resulting in a significant investment of time and money.

Data collections without a predefined data model are called unstructured (homogenous) datasets. These datasets can be found in text, audio, images, and video formats. It is more challenging to work with unstructured data because of its ambiguous format. Also, since it lacks a predefined format, and hence, it is hard to search and analyze. The amount of unstructured data in the world is growing exponentially. It is estimated that 80% of all data created today is unstructured. The increasing use of mobile devices, social media, and the Internet of Things drives this growth. Some examples of unstructured data are: emails, social media posts, images, and audio files.

Pros. Native format: Until it is required, unstructured data is unclear when preserved in its original form. Because of its adaptability, the database's file formats may be expanded, expanding the data set and enabling data scientists to prepare and analyze only the data they are interested. Rapid data accumulation: Data can be acquired quickly and easily because it is unnecessary to characterize it beforehand. Data lake storage: Lowers costs and makes scaling easier by allowing for large storage and pay-as-you-go pricing.

Cons. Requires expertise: due to the undefined/non-formatted nature of unstructured data, preparation and analysis of such data require knowledge in data science. Data analysts benefit from this, but non-expert business users who may not fully understand specialist data concerns or how to use their data are alienated. Needs specialized tools: specialized tools are required to process unstructured data, restricting the range of products available to data managers.

What are the key differences between structured and unstructured data? It is considered more valuable in business applications since it is easier to search, analyze, and generate insights from structured data. For example, companies could use structured datasets to analyze trends and take necessary actions. For example, a database of customer records can be easily searched to find all customers who live in a particular city or have purchased a specific product. Even though it is richer in content, unstructured data takes time to comprehend quickly. For example, a text file of customer reviews would be difficult to find all reviews mentioning a particular product. Structured (quantitative) data provides a "birds-eye view" of consumers, whereas unstructured (qualitative) data provides a more in-depth insight into client behavior and purpose. The main difference is the way that data is organized. Since a structured dataset has a predefined model, it is organized to be used easily. Also, structured data is easy to fetch and manipulate, usually via SQL queries, due to its storage type. Unstructured data is generally stored in a file since files are a more general-purpose way to store data. Files could store any type of data, including structured and unstructured. Sources of structured and unstructured data differ as well. The primary sources of structured data could be GPS data, sensor data, web forms, server logs, ERP data, and OLTP systems. Email messages, word-processing documents, PDFs, and IoT devices are sources for unstructured data. Both datasets are easy to use in ML applications. Structured data is plain to use in ML applications since it has a structure that complies with easy model fitting. On the other hand, unstructured data generally requires more preprocessing and run time to fit. Generally, forecasting or classification is the main purpose of using structured data in business applications. Image detection, audio or video processing, or text generation is the main usage of unstructured datasets. Both have challenges, such as sometimes it is hard to generate insights from structured datasets and it requires more resources to use unstructured datasets in ML modeling.

Table 3.1 below summarizes the key differences between structured and unstructured data:

Table 3.1: Summary Table for Data Structures

Feature	Structured Data	Unstructured Data
Organization	Organized in a way that makes it easy to search and analyze	Not organized in a way that makes it easy to search and analyze
Sources	GPS sensors, online forms, ERP data, web server logs, OLTP systems	email messages, word-processing documents, PDF, IOT devices
Data model	Predefined	No predefined
Storage	Typically stored in a relational database, data warehouses	Often stored in a file or in NoSQL databases
Form	Tabular form, generally numbers and values	sensors readings, text files, audio and video files
Examples	Names, addresses, phone numbers, dates, product SKUs	Text, images, audio, video, social media posts, emails
ML Usage	Easily used with ML algorithms.	Mainly used in image processing and NLP models.
Usage	Usually used in forecasting and classification tasks	Image recognition, text generation, video processing.
Challenges	Can be difficult to extract meaning from	Can be difficult to store and manage

Deep Learning and Tabular Data

Deep learning models are proven successful on common data modalities such as image, audio, text, and video [135, 55]. Tabular data modeling is dominated by machine learning methods, such as extremely gradient-boosted decision trees (XGBoost) [26], which have superior performance over deep learning models. The main problems with tabular datasets are the following: poor signal-to-noise ratio and, therefore, this make developing a high-performance model challenging in considering missing input values, feature redundancy, and noisy measurements.

Recent literature could loosely be divided into two: the first looks for an answer to why deep models underperform against tree-based ensemble models; the other proposes new deep architectures to prove that deep models could perform better as tree-based ensembles. Nevertheless, there are some problems with the research in evaluating the model performance on tabular data stated by [13]. There is no standard benchmark for evaluating tabular data learning methods. However, this gives researchers much freedom to choose the datasets and evaluation metrics that best showcase their methods.

Additionally, most tabular datasets are small compared to other machine learning benchmarks, such as ImageNet [33], which makes it hard to compare different methods, as the results can be unreliable. This, along with other factors like unequal hyperparameter tuning efforts, makes it challenging to replicate machine learning results. The work in [12] provides a comprehensive literature survey on tabular data and states the related open questions and concludes that tree-based models outperform deep learning models on tabular learning tasks and that progress on deep learning models for tabular data is stagnating. Also, in [58] it is stated that neural networks struggle to fit nonsmooth target functions and that redundant features adversely affect MLP-like architectures, while the GBDT model is unaffected by such issues. Furthermore, in [138], various deep learning models are proposed for tabular data with XGBoost model. The results reveal that XGBoost outperforms most deep-learning techniques across all datasets. Another important point is that XGBoost requires far less hyperparameter adjustment to work effectively, which is a big advantage in many real-world circumstances. The attempts with diverse ensembles are noteworthy: When deep neural networks are paired with XGBoost, the best results are obtained. The work in [86] proposes a new method for regularizing neural networks, called “cocktail regularization,” which involves searching for the optimal combination of 13 different regularization techniques for each dataset. They evaluate the proposed method on 40 tabular datasets and show that it can significantly improve the performance of neural networks, even when compared to state-of-the-art specialized neural network architectures and traditional machine learning methods.

In recent literature, a growing number of deep architectures have been proposed for tabular data tasks. Below are some of these.

TabNet [3] selects which attributes to infer from each decision step by employing sequential attention. Using the network’s learning potential for the most crucial characteristic enables greater interpretability and more effective learning.

GrowNet [5] architecture for gradient boosting employs shallow neural networks as "weak learners," in which the output of the previous layer of the current iteration is added to the initial input features at each boosting step. The following weak learner is trained to utilize this enhanced feature set by boosting the most recent residuals.

Tree Ensemble Layers [66] introduce a new layer as "Tree Ensemble Layers," consisting of an ensemble of differentiable decision trees. Because of the soft routing capabilities of these trees, a sample can be routed in either direction in various ways. Soft trees become differentiable as a result, enabling gradient-based learning.

TabTransformer [80] transforms categorical characteristics into dimensional embeddings by utilizing self-attention-based transformers. Transformer blocks, shown to produce reasonably precise accuracy, may then be fed with these embeddings.

Self Normalizing Neural Networks (SNNs) [91] propose a unique activation function that has self-normalizing features and are used in Self Normalizing Neural Networks (SNNs), which are neural networks that self-normalize. These activations tend to zero mean and unit variance even in noise and disturbances.

The Neural Oblivious Decision Ensembles (NODE) [128] provide a novel layer-wise structure comprising differentiable oblivious trees. These trees are decision tables that divide the input into segments and compare each to a learned threshold. Backpropagation is then used to teach NODE from beginning to conclusion.

In [56], authors provide a broad overview of the deep architectures for tabular data and set a baseline for tabular learning by proposing two new structures: The first model inspires the from model developed in [67]. The latter is an adaption of the transformer model proposed by authors, which performs better than most of the Sota models on tabular learning tasks. Proposed models are compared against current architectures on various tasks using the same training and tuning methodologies. They also compare the Sota tabular learning models with XGBoost and fail to find a uniformly superior solution.

CHAPTER 4

BOOSTED DEEP ENSEMBLES: A NOVEL ARCHITECTURE BASED ON DEEP ENSEMBLES

In this chapter, we propose a novel structure that uses transfer learning and reduces the time complexity of deep ensembles. Using various energy datasets, we show proposed model generates predictions which are statistically equivalent that of deep ensemble model. Authors of the original paper state that using *mse* as a cost function underestimates predictive uncertainty. However, using *nll* as a cost function negatively affects the model's accuracy. We further see that if the number of base learners increases, the predictive distribution could be sufficiently approximated without compromising accuracy while using *mse* as a cost function. We provide robustness results by changing the structure of the model and that of base learners as well. As ensembling theory states that using decorrelated ensemble members ensures minimizing total variation, we observe that ensemble members with inaccurate prediction hurt model performance. Simply, leaving those ensemble members with the worst performance out of the ensembling increases accuracy.

4.1 Introduction

The training process of neural networks depends on minimizing/maximizing proper cost function with respect to its parameters using gradient-based algorithms. While the architecture might differ among applications, the stochastic gradient algorithm (SGD) is the common ground within these applications. Stochasticity is induced by random initialization of parameters and mini-batch learning during training. With a noisy dataset and poor initialization of parameters, SGD might still converge to a local minima, but results in poor performance. Furthermore, in practice, the same network structure with different parameter initialization is expected to converge to different local minima since deep networks typically never converge to global minima [79]. However, this could be exploited by using an ensembling strategy to produce better and generalized results [64].

Neural network ensembles have been widely studied and applied in machine learning [92, 126]. Ensemble models are used for increasing the stability of the predictions. However, these methods can also be exploited to generate prediction uncertainty. ML-based decision support systems must deliver uncertainty of the output (predictive intervals) to provide meaningful and integrated information. In many applications, standard deep learning approaches produce overconfident probabilistic forecasts [59]. Since many practical applications of such systems require uncertainty bounds and uncertainty estimation becomes a critical issue. However, it is still an open question and challenging as the works [51] and [95] state.

Bayesian neural networks (BNN) are the primary tool for estimating uncertainty; NNs are unsuitable for this task. Bayesian models are computationally complex and thus require long times in standard computer CPUs, even considering the increasing computational performance of modern GPUs. The benefits of ensemble models are not limited to increasing generalization performance; they allow Bayesian modeling and model calibration. Using variance decomposition or minimizing for negative log likelihood [89]. Moreover, deep ensembles allow for parallel computation and overperform BNNs in uncertainty estimation while requiring significantly less computation power [122, 95] and [60].

The power of ensembling methods is due to diversity by initializing randomly the parameters of each neural network in the ensemble or bootstrapping the dataset. However, authors of [95] state that bootstrapping can even hurt the performance of ensembles; random initializing is in general sufficient for generating diversity among base neural networks. However, we observe that when base networks are initialized randomly, some ensemble members converge to poor local minima that produce inaccurate predictions and, therefore, reduce the ensemble's performance. Thus, it is necessary to leave those with poor performance out from the ensemble using a validation set approach.

Boosting is another mechanism for ensembling; instead of running a number of models, it sequentially fits a base model to residuals of the previous base model until convergence. This idea is similar to pre-built transfer learning in deep learning literature. In the boosting mechanism, it is assumed that each model transfers the knowledge of the previous one to the successor model, and the successor adds to this knowledge by fitting a base model to residuals. Using this insight, we propose transfer learning-enabled deep ensembles called *boosted deep ensembles* that run base neural networks in clusters, and in each cluster, models are sequentially fit to the same training set by transferring the knowledge from previous base neural networks, and hence, reducing epoch number to decrease run time which could easily be executed in parallel. Moreover, by using a validation set, we select the top n best-performing models to calibrate the ensemble to increase predictive accuracy while deteriorating uncertainty performance.

Contributions of this chapter are fourfold: first, we investigate ensemble methods in machine learning and deep learning literature and its probabilistic applications. Then, we propose a parallelizable transfer learning enabled deep ensemble model that runs significantly faster than plain deep ensembles and generates distributionally equivalent prediction deep ensembles. Moreover, when trained with a sufficient amount of base learners, boosted deep ensembles generate normally distributed predictions when using *mse* as a loss function, whereas using negative log likelihood hurts point forecasting accuracy.

Finally, we see that decorrelated predictions are insufficient to generate a superior model; instead, leaving out poorly-performing base learners generates superior forecasts.

4.2 Background

Ensemble methods have been a long-studied topic in machine learning models. In the literature, there are several ways that combine different models, considered as weak or base learners. We focus on bagging and boosting methods, which build our proposed model. Tree-based methods such as random forest [14] and gradient boosting trees [134] are well-known examples of bagging and boosting methods, respectively. Another class of models, deep ensembles, uses network architectures as base learners. These models are shown to be superior over BNNs over quantifying uncertainty [95], in addition to the requirement of the computational cost of deep ensembles, which are significantly lower.

The theoretical foundation of ensemble learning with neural networks has been established back in the 1990s. The studies in [92, 126] provide theoretical and empirical evidence that decorrelation across member models can boost ensemble performance, basically by averaging sets right individual errors from base models. There are two strategies to generate decorrelated base learners: randomly initializing each base neural network or training on bootstrapped series. The main idea behind deep ensembles is that each base network converges to different local minima, which then generate predictors with various characteristics; thus, combining those base models generates better results. In order to see how powerful simple averaging ensembling could be, we define

$$g_{bem}(x) = \frac{1}{M}\sum_{i=1}^{M} g_i(x) = \bar{g}(x)$$

and error $e_i = y - g_i(x)$ for each of the M methods in the ensemble. Then, expected mean squared error $\epsilon_i = \mathbb{E}[e_i^2]$ is used to calculate the average mean squared error, which is,

$$\bar{\epsilon} = \frac{1}{M}\sum_{1}^{M} \mathbb{E}[e_i^2].$$

It is easy to see that mean squared error of g_{bem} is

$$\epsilon_{g_{bem}} = \frac{1}{M}\sum_{i=1}^{M} \mathbb{E}[e_i]^2. \tag{4.1}$$

Rearranging (4.1) as

$$\frac{1}{M}\sum_{i=1}^{M} \mathbb{E}[e_i^2] + \frac{1}{M^2}\sum_{i \neq j} \mathbb{E}[e_i e_j]$$

and assuming that the e_i are mutually independent with zero mean, we have:

$$\frac{1}{M^2}\sum_{i=1}^{M} \mathbb{E}[e_i^2] = \frac{\bar{\epsilon}}{M}; \tag{4.2}$$

and thus,

$$\lim_{M \to +\infty} \epsilon_{g_{bem}} \to 0. \tag{4.3}$$

This result implies that, as number of base model increases, mean squared error decreases. That proves how simple ensembling, in theory, increases stability and the accuracy of the model.

Moreover, using bias-variance trade off, we can prove that expected mean squared error of ensemble bounded by average of expected mean squared error of base learners; this implies that it is always favourable to use ensemble models.

Variation of each base learner $g_i(x)$ from ensemble $\bar{g}(x)$ as follows:

$$a_i(x) = \mathbb{E}[\bar{g}(x) - g_i(x)],$$

and average variation as

$$\bar{a}(x) = \frac{1}{M} \sum_{i=1}^{M} \mathbb{E}[\bar{g}(x) - g_i(x)]^2. \tag{4.4}$$

We further define individual base models expected predictive variation from each base model as

$$\epsilon_i = \mathbb{E}[(y - g_i)^2],$$

and predictive variation of ensemble as

$$\epsilon_{ens} = \mathbb{E}[(y - \bar{g})^2].$$

Adding and substracting y to (4.4), we get

$$\bar{a}(x) = \frac{1}{M} \sum_{i=1}^{M} \mathbb{E}[\bar{g}(x) - y + y - g_i(x)]^2$$

with some algebra, which leads to

$$\bar{a}(x) = \frac{1}{M} \sum_{i=1}^{M} \epsilon_i - \epsilon_{ens}.$$

Moreover opening the summation and setting $\bar{\epsilon} = \frac{1}{M} \sum_{i=1}^{M} \epsilon_i$, we obtain

$$\bar{a} = \bar{\epsilon} - \epsilon_{ens}, \tag{4.5}$$

which directly implies that $\epsilon_{ens} \leq \bar{\epsilon}$, meaning that, total variation of ensemble model is smaller than average of variations of base models. Moreover, we can observe that, as $\bar{a}$ variation of base learners increases predictive variation of ensemble $\bar{\epsilon}$ decreases. Together with (4.3), these strong results state that, decorrelation of base learners is crucial in model ensembling. In literature, bootstrapping and model training with random initialization is the core methods for generating diversity amongst base learners. However, in [53] and [95] it is stated that if necessary randomization is generated through random initialization, bagging is unnecessary. The explanation by [53] states that, the models could only see 63% of unique data in dataset. Therefore, it reduces generalization power of base models and hence reduces that of ensemble. Since, ensembling methods are computationally expensive, the work [79] proposes another ensembling approach by saving parameters of a single model during training process and then combining these snapshots. It uses cyclic learning rate as $\alpha(t) = f(\mod (t-1, [T/M]))$, where f is strictly decreasing function to converge local minima faster. Snapshot model reports that snapshot ensemble methods outperforms models such as Resnet, and DenseNet.

4.3 Deep Ensembles

Deep ensembles, shown to be promising alternative for both regression and classification tasks with tabular datasets [95], are simply inspired by bootstrapping. Randomly initializing M base neural network or training M neural network with bootstrapped series, we get deep ensemble estimate of y simply taking average over the base learners. As we have a list of pairs $\{\hat{y}_i, \hat{\sigma}_i^2\}_{i=1}^M$, the mean estimate $\hat{y}_{DE}$ is simply

$$\hat{y}_{DE} = \frac{1}{M}\sum_{i=1}^{M}\hat{y}_i. \tag{4.6}$$

In [95] authors use negative log likelihood as a cost function:

$$L(y) = \frac{1}{D}\sum_{i=1}^{M}\frac{\log\sigma^2}{2} + \frac{(y_i - \hat{y})^2}{2\sigma^2}. \tag{4.7}$$

4.3.1 Model Structure of Deep Ensembles

The model structure is plain and simple in deep ensembles. Running in parallel the M models and averaging them will generate deep ensembles. See Algorithm 2. In application, each base model is randomly initialized for training on the same training set; then, model averaging is performed on the test set. The critical part is that, as we showed, also by [92, 126], that the number of decorrelated models decreases as the mean squared error goes to zero in theory. The primary strategy for generating decorrelated models is random initialization of model weights. However, we observe that this strategy weakens the model performance of deep ensembles as some elements of the ensemble converging to poor local minima, deteriorating the ensemble estimates. Thus, deep ensembles require model selection on the validation set, for example, by choosing top-performing base models or hill climbing method [19].

Algorithm 2: Deep Ensembles Algorithm

```
Input: Dataset $D$, base neural netowrk $g$ to be fit
Output: Prediction of deep ensembles
for $m \leftarrow 1$ to $M$ do
    fit base model $g_i$ with initialized randomly parameters $\theta$ on train set of $D$
end
return Average of predictions on test dataset
```

Uncertainty Quantification with Neural Networks

Accurately estimating uncertainty is the backbone of a reliable machine learning model as a decision support system. Since neural networks are overconfident in uncertainty estimation, which may lead to catastrophic application problems [68, 131]. Classical Bayesian inference requires complex computations for calculating posterior distribution, which is usually infeasible in the production environment. Deep ensembles are shown to be a good candidate for this problem as well [95]. Uncertainty is usually considered in two categories: reducable and unreducable uncertainty. Below, we discuss the types of uncertainty and estimation uncertainty with ensembles.

Epistemic Uncertainty. Epistemic uncertainty, which could be reduced with additional data, refers to model uncertainty caused by ignorance or lack of knowledge about the data-generating process. Therefore, it could be reduced by observing additional data. It is governed by employing alternative models for the data-generating process.

Aleatoric Uncertainty. Aleatoric uncertainty cannot be reduced by observing additional data, which is the natural randomness of the process itself. Noisy data measurements or lack of technology that could measure the actual values is the reason for this uncertainty. Therefore, it could not be reduced by simply observing additional data.

Bayesian Inference.

As it is known, most machine learning models generate point forecasts as a single prediction using a stochastic gradient descent algorithm. These models do not consider the data (aleatoric) and model (epistemic) uncertainty. However, Bayesian models deal with uncertainty by introducing prior and posterior distribution of model parameters, unlike point forecast models. This procedure is called Bayesian inference.

In Bayesian Inference, we have

$$p(W|Y,X) = \frac{p(Y|X,W)p(W|X)}{p(Y|X)}, \tag{4.8}$$

where W is the weight parameters of neural network, X, Y input and output variables correspondingly. $p(Y|X,W)$ is likelihood of Y given X, W, and $p(Y|X)$ is prior distribution. Marginalizing over W,

$$p(Y|X) = \int_{\Omega} p(Y|W,X)p(W|X)dW. \tag{4.9}$$

We obtain predictive distribution y^* for given x^* as

$$p(y^*|x^*) = \int_{\Omega} p(y^*|W,x^*)p(W|x^*)dW. \tag{4.10}$$

However, in most cases, posterior $p(W|Y,X)$ is analytically intractable, thence one needs, for instance, variational inference such as Markov chain Monte Carlo (MCMC). Even this formalism often requires long computational times and therefore is infeasible in production environment. Thus, ensemble models might be considered a simple alternative of Bayesian inference.

Uncertainty Estimation with Monte Carlo Dropout Method

In [51] it is shown that variational inference for posterior $P(Y|X, W)$ could be approximated with a simple function $q^*(W|X)$ by Monte Carlo dropout method which randomly switches off weights in each layer during both in training and test stages. This procedure could be done by fitting a neural network minimizing cost function L on dataset with length D where L is defined as in (4.7) with dropout layers after each weight layer and performing dropout both during training and test periods. This neural network outputs a list of pairs $(\hat{y}, \hat{\sigma^2})$ for each element in the test set. Monte Carlo inference is performed by forward passing of the neural network M times. After performing, we have $\{\hat{y}_i, \hat{\sigma_i^2}\}_{i=1}^M$, a set with M pairs, each generated by random sampling from posterior $q^*(W|X)$. Using this we have,

$$\mathbb{E}_{p^*}[y^*|X^*] = \int_\Omega y^* p(y^*|W, x^*) p(W|x^*) dW \overset{MC}{\approx} \frac{1}{M} \sum_{i=1}^M \hat{y}_i, \tag{4.11}$$

the right-hand side of which is Monte Carlo dropout estimate:

$$\hat{y}_{MC} = \frac{1}{M} \sum_{i=1}^M \hat{y}_i. \tag{4.12}$$

For the variance estimation of y, we use the *total law of variance*:

$$\text{Var}(y|x) = \text{Var}[\mathbb{E}(y|W, x)] + \mathbb{E}[Var(y|W, x)], \tag{4.13}$$

where first term corresponds to epistemic uncertainty and second term represents aleatoric uncertainty. Using (4.13), Monte Carlo estimate of Var$(y|x)$, denoted by $\hat{\sigma}^2_{y_{MC}}$, is computed as

$$\hat{\sigma}^2_{y_{MC}} = \frac{1}{M} \sum_{i=1}^M (\hat{y}_i - \hat{y}_{MC})^2 + \frac{1}{M} \sum_{i=1}^M \hat{\sigma}_i^2. \tag{4.14}$$

This corresponds to (4.13) as follows

$$\text{Var}[\mathbb{E}(y|W, x)] = \frac{1}{M} \sum_{i=1}^M (\hat{y}_i - \hat{y}_{MC})^2 \tag{4.15}$$

and

$$\mathbb{E}[\text{Var}(y|W, x)] = \frac{1}{M} \sum_{i=1}^M \hat{\sigma}_i^2. \tag{4.16}$$

Uncertainty Estimation with Deep Ensembles

As previously defined, having M base estimates, we have a list of pairs $\{\hat{y}_i, \hat{\sigma}_i^2\}_{i=1}^M$, and following [51], the variance estimate $\hat{\sigma}_{y_{DE}}^2$ of y becomes

$$\hat{\sigma}_{y_{DE}}^2 = \frac{1}{M}\sum_{i=1}^{M}(\hat{y}_i - \hat{y}_{DE})^2 + \frac{1}{M}\sum_{i=1}^{M}\hat{\sigma}_i^2, \tag{4.17}$$

where

$$\hat{y}_{DE} = \frac{1}{M}\sum_{i=1}^{M}\hat{y}_i. \tag{4.18}$$

Hence, as in (4.13) we have

$$\text{Var}[\mathbb{E}(y|W,x)] = \frac{1}{M}\sum_{i=1}^{M}(\hat{y}_i - \hat{y}_{DE})^2 \tag{4.19}$$

and

$$\mathbb{E}[\text{Var}(y|W,x)] = \frac{1}{M}\sum_{i=1}^{M}\hat{\sigma}_i^2. \tag{4.20}$$

4.3.2 Boosted Deep Ensembles

Deep ensembles, or in general model ensembling, produces superior predictions over individual models. The main reason is that having multiple models smoothens the errors of an individual model. However, the main downside of deep ensembles is that fitting M models instead of a single model requires a longer computational time. When M gets significantly large, such as 1000 or more, the required computational time becomes infeasible for the production environment. If base models are shallow networks, parallelization could manage this problem since each M base model runs independently. Nevertheless, depending on the problem, using shallow neural networks as base learners could be problematic in terms of generalization. Therefore, arrangements regarding reducing the required time to train the base models, such as decreasing the number of layers or epochs, might yield problematic consequences.

In order to keep base neural networks relatively large, reducing the number of epochs, which might affect the learning process of neural networks, is the primary option. There is a trade-off between epoch number and computing time. However, by transferring the weights among base neural networks, the model we propose allows us to reduce epoch numbers and thus reduce computational time while preserving the model accuracy of base models. As we mentioned before, relatively complex models, which require longer computational time, make the production process troublesome in terms of system load. Thus, it is crucial to have models that run faster in the production environment. We propose a model based on deep ensembles that runs significantly faster than plain deep ensembles while preserving accuracy, even improving it slightly. The proposed model runs in parallel on the number of clusters for base learners. A group of base neural networks boosts consequent base learner by weight transfer and we drop the epoch number as weight is transferred through base models; this leads to reducing time requirement significantly while preserving accuracy. Optionally, weights from a warm-up model could also be used as initialization for clusters trained on bootstrapped datasets to generate decorrelated predictors. Hence, on a validation set, we choose top performing N base models and calculate the boosted deep ensembles predictor by simply taking the average. Pseudocode for proposed algorithm is shown in Algorithm 3 and an illustrative schema of the structure of the method is presented in Figure 4.1.

Algorithm 3: Boosted Deep Ensembles Algorithm

```
Input: $D = \{d_1, d_2, \ldots, d_C\}$ base neural network $g$ to be fit, initial
        parameters $\theta^*$ from warmup base model, C, N
Output: Prediction of deep ensembles
for $c \leftarrow 1$ to $C$ do
    for $n \leftarrow 1$ to $N$ do
        fit base model $g_{c,n}$ with initialized parameters with $\theta^*_{c,n-1}$ on train set
          of $d_c$
        Reduce number of epochs
    end
end
Sort base models in ascending order based on predictive performance
  criterion on validation set
return Average of top $N$ base models on test dataset
```

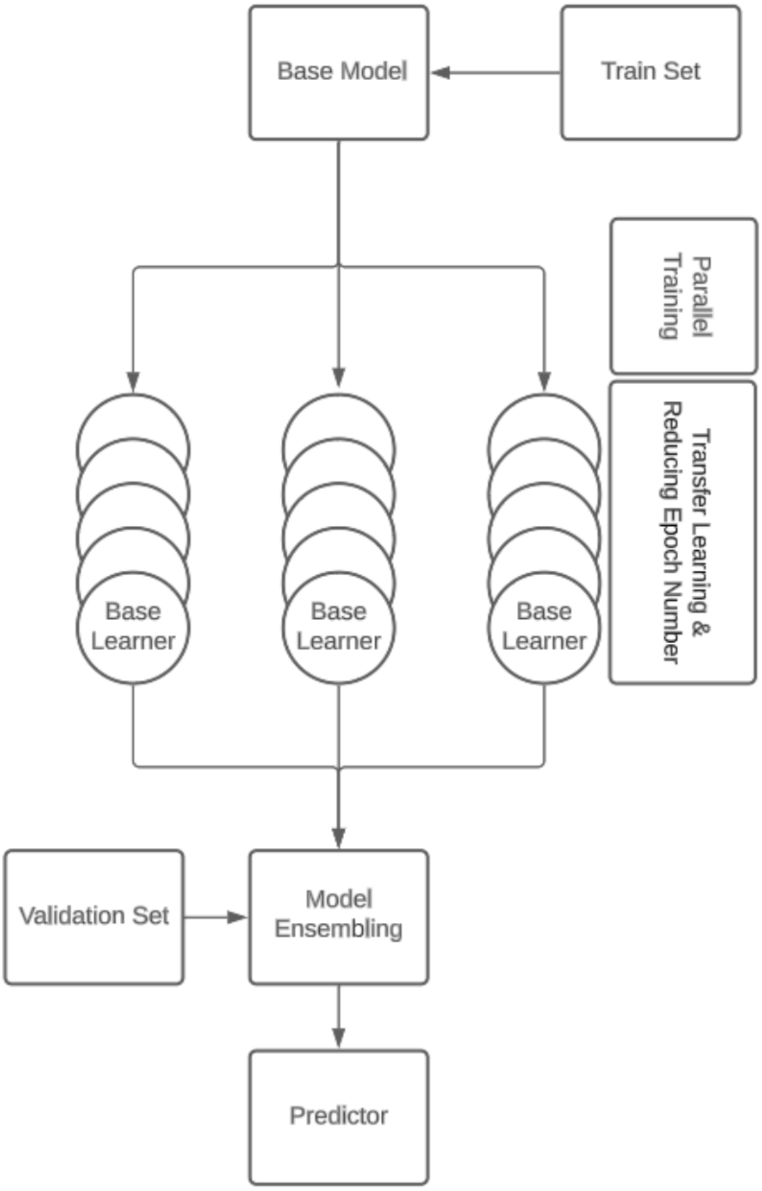

Figure 4.1: Model Structure of Boosted Deep Ensembles

Let us give an example of our proposed model by considering a deep ensembles model with a 100 base learner, each running 200 epochs for training. The corresponding structure, shown in Figure 4.1, of the boosted deep ensemble for this example is as follows: We first run a warm-up base model and save its parameters. We then generate 25 clusters and 25 bootstrapped training datasets. Each cluster has four base models, which run consequently by transferring parameters from the previous model. For training, initializing first base learners of each cluster, optionally with a warm-up base model, and transferring the parameters to consecutive base learners, we divide the number of epochs by the square of the cluster level to reduce the time requirements. For example, after fitting the first layer of the clusters, i.e., the first 25 base models, in the second layer, we divide the number of epochs by $2^2 = 4$. Consequently, we divide the number of epochs by $3^2 = 9$ in the next cluster level, and this process continues until the last cluster level. Finally, we get all predictors, and by evaluating them on a validation set, we pick the top best performing N models to calculate the average in order to generate ensemble results.

4.3.3 Complexity of Boosted Deep Ensembles

In this section, we analyze and compare the computational complexity of deep ensembles and boosted deep ensembles. Since neural networks are composed of multiple matrix operations, we can calculate the complexity of a 3 layer neural network with a number of nodes i, j, k respectively in each layer and having t training size as follows:

Given two matrices M_{ij} and M_{jk}, we have $\mathcal{O}(ijk)$ for matrix mutiplication. For the feed-forward pass, we have matrix multiplication for going from layer i to j:

$$S_{jt} = W_{ij} Z_{it},$$

where W_{ij} is the weight matrix between the input layer and next layer and Z_{it} contains dataset. We have $\mathcal{O}(ijt)$ for this operation. Then applying f elementwise,

$$Z_{jt} = f(S_{jt}),$$

which is $\mathcal{O}(jt)$. In total, for this operation, we have

$$\mathcal{O}(ijt + jt) = \mathcal{O}(ijt).$$

Now, iterating through layer j to k, we have

$$\mathcal{O}(ijt + jkt) = \mathcal{O}(t(ij + jk).$$

For the backpropagation algorithm, starting from layer k to j, we have the derivative of the error signal E_{kt} as

$$E_{kt} = f'(S_t) \odot (Z_{kt} - O_{kt}),$$

where Z_{kt} is the model predictions and O_{kt} is the output data. Note that $\odot$ is an elementwise operation. The complexity of such an operation is $\mathcal{O}(kt) + \mathcal{O}(kt)$. Then computing derivatives of D_{kj},

$$D_{kj} = E_{kt} Z_{tj},$$

requires $\mathcal{O}(ktj)$ operations. For the weight update we have,

$$W_{kt} = W_{kj} - D_{kj},$$

which is $\mathcal{O}(kj)$. Then for this totally we have,

$$\mathcal{O}(kj(t+1)) = \mathcal{O}(kjt).$$

Similarly, going backwards from j to i, with exact same steps we have,

$$\mathcal{O}(ijt).$$

In total, we have

$$\mathcal{O}(t(ij + jk)),$$

which is equal to feed-forward propagation. Thus we have,

$$\mathcal{O}(t(ij + jk)).$$

finally having e epochs at hand requires

$$\mathcal{O}(et(ij + jk)).$$

Therefore, it is easy to calculate that, deep ensembles model with M base neural networks with 3 layers, the complexity is

$$\mathcal{O}(Met(ij + jk)). \tag{4.21}$$

Having M base learner, with c clusters computational complexity of boosted deep ensembles model is

$$\mathcal{O}\left(\frac{M}{c} et(ij + jk)\right). \tag{4.22}$$

Indeed, the computational effort for training a boosted deep ensemble is obtained by summing the number of computation in each clusters, which is

$$\sum_{n=1}^{c} \frac{M}{c} \frac{e}{n^2} t(ij + jk). \tag{4.23}$$

Thus, in order to show that boosted deep ensemble has less computational complexity, we need to show

$$\sum_{n=1}^{c} \frac{M}{c} \frac{e}{n^2} t(ij + jk) \leq Met(ij + jk). \tag{4.24}$$

Since, we have the trivial identity that

$$\sum_{n=1}^{c} \frac{1}{n^2} \leq 1 + \int_1^c \frac{1}{x^2} dx, \tag{4.25}$$

in which the integral can easily be calculated as

$$\int_1^c \frac{1}{x^2} dx = 1 - x^{-1}\Big|_1^c = 2 - \frac{1}{c} \leq 2. \tag{4.26}$$

Therefore, the relation in (4.24) becomes

$$\frac{Met(ij + jk)}{c} \sum_{n=1}^{c} \frac{1}{n^2} \approx \left(2 - \frac{1}{c}\right) \left(\frac{Met(ij + jk)}{c}\right). \tag{4.27}$$

Equivalently, we have that

$$\left(2 - \frac{1}{c}\right) \left(\frac{Met(ij + jk)}{c}\right) \leq 2 \, \frac{Met(ij + jk)}{c} \leq Met(ij + jk), \tag{4.28}$$

for $c \geq 2$. We note that minimum cluster number is 2 by definition of boosted deep ensembles. Thus, we obtain

$$\mathcal{O}\left(\frac{M}{c} et(ij + jk)\right) \leq \mathcal{O}(Met(ij + jk)). \tag{4.29}$$

4.4 Experiments on Public Datasets

We run experiments on three publicly available energy datasets to compare the accuracy and time spent fitting each model defined in the experiment section. Datasets are available in Kaggle datasets [87]. First dataset [81] contains four years of energy demand and weather data for Spain. Energy demand data is retrieved from ENTSO-E [46], a public portal for Transmission Service Operator (TSO) data. As an input variable, weather data is retrieved from the Open Weather API [121] for the five largest cities in Spain. However, to be concise, we run experiments with variables from Valencia. This dataset is used in following papers [4, 130, 133]. Energy demand (Load forecasting) forecasting is an important task for energy market participants. In general, temperature and demand values of previous days are dominant variables for predicting demand in the next days. The Tables below 4.1, 4.2, 4.3 summarize the variables and dataset:

Table 4.1: Variables of Load Dataset

Target variable	Total actual load
Weather variables	Humidity Rain probability Temperature Maximum Temperature Wind Speed
Calendar Variables	Days of week Hours of day
Lagged Variables	24 hour

Table 4.2: Descriptive Analysis of Load Dataset

	Total Load	**Humidity**	**Rain probability**	**Temperature**
count	34946	34946	34946	34946
mean	28695.0174	65.0957	0.0345	290.8064
std	4572.1340	19.6831	0.2640	7.2237
min	18041	8	0	268.8306
25%	24809	51	0	285.15
50%	28900	67	0	290.25
75%	32188	82	0	296.15
max	41015	100	12	311.15

Table 4.3: Descriptive Analysis of Load Dataset

	Maximum Temperature	**Wind Speed**	**Lag 24**	**Lag 1**
count	34946	34946	34946	34946
mean	291.3807	2.6926	28698.8335	28696.5231
std	7.5126	2.5794	4574.9796	4572.5696
min	268.8306	0	18041	18041
25%	285.75	1	24808	24810
50%	291.15	2	28902	28901
75%	297.15	4	32194	32188.75
max	314.82	133	41015	41015

The second dataset is the Texas wind dataset, which consists of hourly wind generation and related weather variables time-series, simulated using the National Renewable Energy Laboratory (NREL) software for a location in Texas, US [129]. This dataset is used in following papers [143], [106] and [111]. Accurate wind forecasting is crucial for energy trading and is highly related to wind speed and direction from the wind farm. The tables below summarize the variables:

Table 4.4: Variables of Wind Dataset

Target variable	Power Generated
Weather variables	Wind Speed Wind Direction Pressure Temperature

Table 4.5: Descriptive Analysis of Wind Dataset

	Power generated	**Wind speed**	**Wind direction**	**Pressure**	**Air temperature**
count	8760	8760	8760	8760	8760
mean	964.4679	7.3761	146.6162	0.9923	22.1472
std	878.5581	3.1382	84.6787	0.0053	4.85529
min	0	0.119	0	0.9745	3.263
25%	215.9515	5.121	99	0.9889	19.463
50%	693.9685	7.3405	135	0.9914	22.763
75%	1562.2875	9.599	169	0.9950	26.063
max	3004.01	19.743	360	1.0145	32.963

We use a radiation forecasting task from Kaggle datasets [41] for the NASA Hackathon, as the third dataset contains five-minute radiation measurements with weather variables. This dataset is used in [45].

Table 4.6: Variables of Solar Dataset

Target Variable	Radiation
Weather Variables	Temperature Pressure Humidity Wind Direction Wind Speed

Table 4.7: Descriptive Analysis of Solar Dataset

	Radiation	**Temperature**	**Pressure**	**Humidity**	**WindDirection**	**Wind Speed**
count	32686	32686	32686	32686	32686	32686
mean	207.1246	51.1032	30.4228	75.0163	143.4898207	6.243869241
std	315.9163	6.2011	0.0546	25.9902	83.1674	3.4904
min	1.11	34	30.19	8	0.09	0
25%	1.23	46	30.4	56	82.2275	3.37
50%	2.66	50	30.43	85	147.7	5.62
75%	354.235	55	30.46	97	179.31	7.87
max	1601.26	71	30.56	103	359.95	40.5

4.5 Experiments

We compare the model performances on the test set with the error metrics used below. For deep ensembles (nll), deep ensembles (mse), boosted deep ensembles, and XGBoost model, we compare both mean accuracy and probabilistic accuracy. Implementation details are given below.

Dataset Preprocessing

We use min-max normalization for both datasets to compare results fairly. Scikit-learn library [125] is employed for this normalization. For each datasets, both input and target variables are normalized using

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}. \tag{4.30}$$

Hyperparameters

Default parameters for XGBoost model are library's defaults with 200 base learners. All neural networks, run for 100 epochs, have 4 layer and 32 nodes in each layer except for final layer. Deep ensembles run with 100 base learners.

Training

We split both datasets as follows: 65% for training, 10% for validation and 25% for testing with 5-fold cross validation using a rolling window.

Metrics

For mean accuracy, we use mean square error (MSE) and mean absolute percentage error (MAPE) for load forecasting and normalized mean absolute error (nMAPE) for the Texas wind and solar datasets. For probabilistic forecasting, following [34, 73, 54] we compare calibration error which corresponds to coverage (CV) in a confidence interval. Additionally, we compare widths (W) of intervals and coverage penalised by width (CPW) since having arbitrarily wide widths, coverage values could be misleading. The metrics we use are:

$$\text{MSE}(y_i, \hat{y_i}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2, \tag{4.31}$$

$$\text{MAPE}(y_i, \hat{y_i}) = \frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \hat{y_i})^2}{y_i}, \tag{4.32}$$

$$\text{nMAPE}(y_i, \hat{y_i}) = \frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \hat{y_i})^2}{c}, \tag{4.33}$$

$$\text{CV}(y, \Gamma(x)) = \text{prob}(y \in \Gamma(x)), \tag{4.34}$$

$$\text{W}(\Gamma(x)) = \frac{1}{n}\sum_{i=1}^{n}|u(x)_i - l(x)_i|, \tag{4.35}$$

$$\text{CPW}(x) = \frac{\text{CV}(x)}{\text{W}(x)}, \tag{4.36}$$

where y is point to be forecasted, $\hat{y}$ is the prediction, c is a constant $\Gamma(x)$ is the confidence interval, x is input variable $u(x)$, upper bound of confidence interval and $l(x)$ is lower bound of confidence interval. We use 95% confidence interval for this experiment.

4.6 Results

We report mean and probabilistic accuracy of predictions of the models and results in this section.

Table 4.8: Forecast Results for Load Forecasting

Load dataset	MAPE	MSE	W	CV	CV/W	time
Deep ensembles (mse)	1.4142	616.3317	1961.5604	0.9598	0.0489	446 sec.
Deep ensembles (nll)	2.7778	1316.1332	8160.5977	0.9847	0.0072	482 sec.
XGBoost	1.5191	671.8364	1569.2807	0.9003	0.0529	61 sec.
Boosted deep ensembles	1.4352	618.1924	1830.8610	0.9485	0.0542	212 sec.

Table 4.9: Forecast Results for Wind Generation Forecasting

Wind dataset	nMAPE	MSE	W	CV	CV/W	time
Deep ensembles (mse)	0.9906	1449.4027	275.9436	0.9967	0.3935	206 sec.
Deep ensembles (nll)	1.9267	10647.0216	850.9101864	0.9995	0.1267	228 sec.
XGBoost	0.9420	1114.5431	418.1629	0.9809	0.3101	38 sec.
Boosted deep ensembles	0.9794	2975.6419	265.9313	0.9990	0.4116	78 sec.

Table 4.10: Forecast Results for Radiation Forecasting

Solar dataset	nMAPE	MSE	W	CV	time
Deep ensembles (mse)	2.3172	360.0010	535.2858	0.96064	392 sec.
Deep ensembles (nll)	4.2782	291.5812	659.9617	0.9877	412 sec.
XGBoost	2.3698	332.7018	462.6163	0.9810	52 sec.
Boosted deep ensembles	2.2468	360.4090	536.0728	0.9592	169 sec.

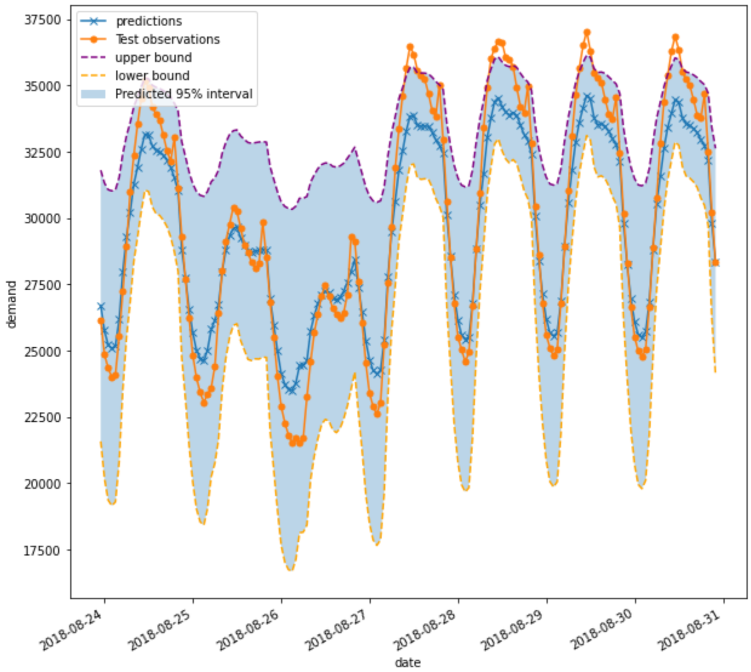

Figure 4.2: Results of Deep Ensemble (nll) Model

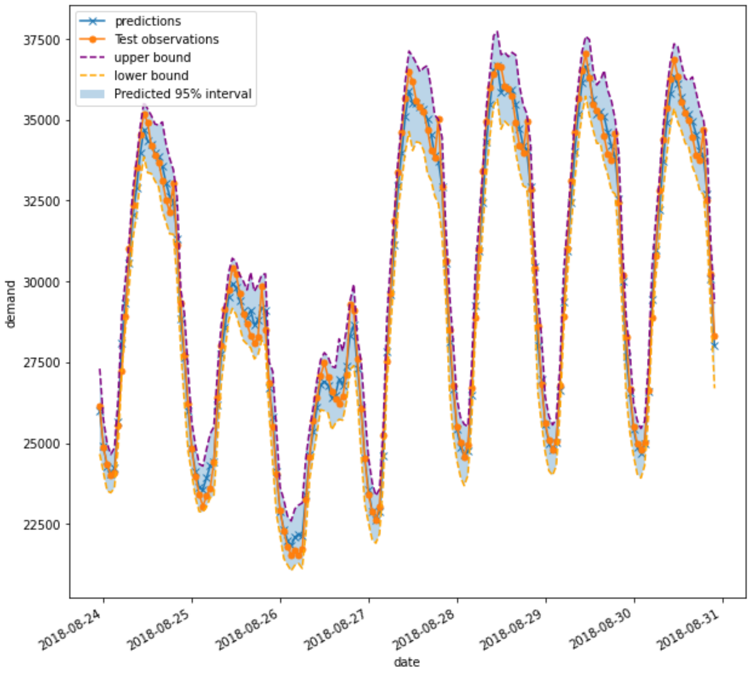

Figure 4.3: Results of Boosted Deep Ensemble Model

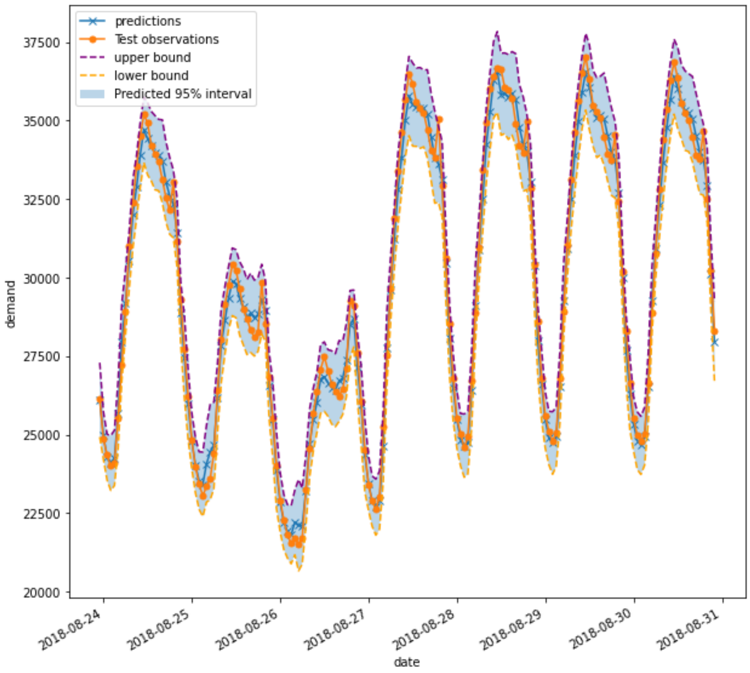

Figure 4.4: Results of Deep Ensemble (mse) Model

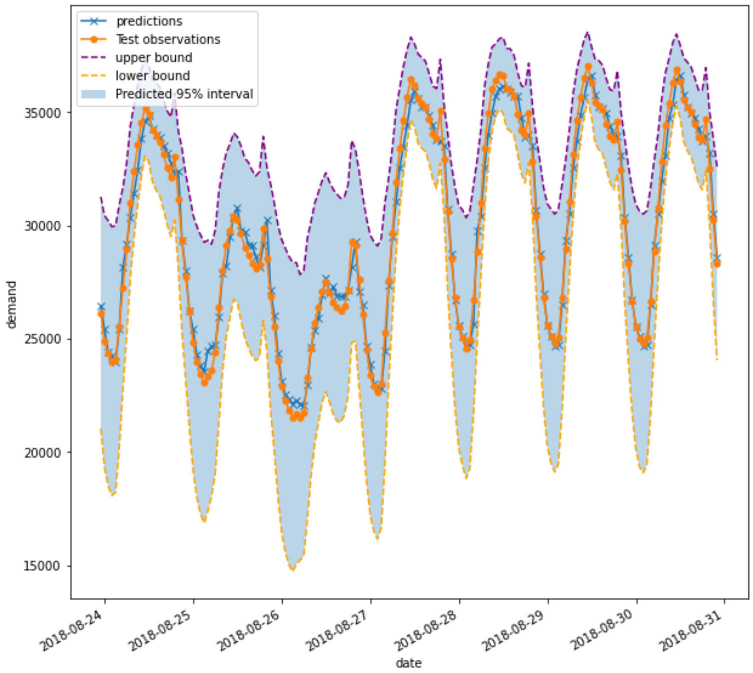

Figure 4.5: Results of XGBoost Model

We can see from Tables 4.8, 4.9, 4.10 that results for point forecasting comparison, both deep ensembles (mse) and boosted deep ensembles models are, promising models for predictive tabular energy data tasks, outperforming XGBoost on load forecasting task. Even though having calibrated results in probabilstic forecasting deep ensembles (nll), generates poor mean forecasting results which shows optimizing negative log likelihood cost function hurts mean prediction ability. Both deep ensembles and boosted deep ensembles generates well calibrated probabilistic forecasts when MSE is consisdered as cost function, which could be seen in Figures 4.3, 4.4, 4.5.

4.6.1 Hypothesis Testing

Further following [138], in addition to using the RMSE and MAPE for evaluating the performance of the models, it is also necessary to assess whether these differences are statistically significant. Diebold-Mariano test (DM Test) [36] for comparing the accuracy of two forecasts, a widely used method for testing statistical significance for comparison of forecast accuracy. In this section, we aim to show that the distribution of the boosted deep ensembles, which dramatically reduces the computational complexity of the plain deep ensembles model, is statistically the same as that of plain deep ensembles. Thus, we compare distributions of boosted deep ensemble and plain deep ensembles.

Table 4.11: Results of Hypothesis Tests for Datasets

	test statistics	p-value
Load dataset	1.3291	0.0968
Wind dataset	1.8459	0.9671
Solar dataset	1.1126	0.867

DM test result in Table 4.11 shows that we fail to reject the null hypothesis for each of the three datasets, as two forecasts have the same accuracy. Thus, we can statistically conclude that the boosted deep ensembles model produces identical forecasts with plain deep ensembles while dramatically reducing the computational effort. We also provide plots for comparison of distribution of the predictions from both models below in Figures 4.6, 4.7.

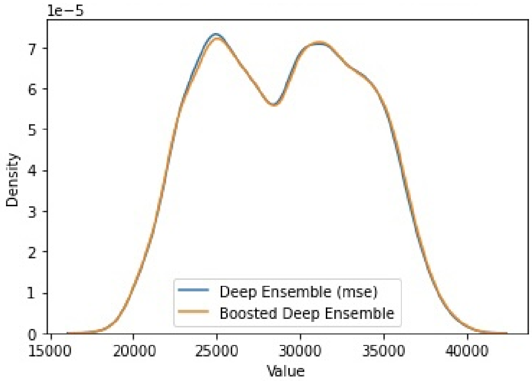

Figure 4.6: Kernel Density Comparison of Deep Ensemble (mse) and Boosted Deep Ensemble Model

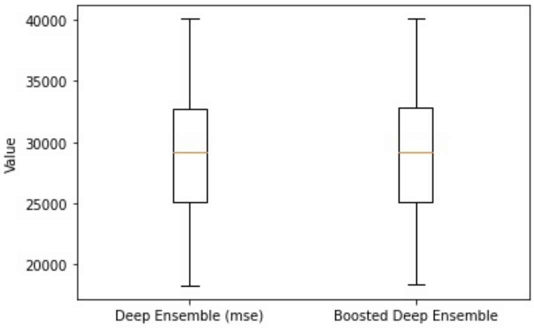

Figure 4.7: Box Plot Comparison of Deep Ensemble (mse) and Boosted Deep Ensemble Model

4.7 Robustness Checks

We also conduct experiments by changing hyperparameters of boosted deep ensembles such as the structure of base models (number of nodes, number of base learners, number of epochs) and structure of boosted deep ensembles such as the number of clusters (depth of clusters). Details are given in the Table 4.12 below:

Table 4.12: Change of Hyperparameters

Load dataset	original version	reduced version
number nodes in layers	32	16
number of layers	4	3
number of epoch	100	50
number of clusters (depth of clusters)	25 (4)	50 (2)

Table 4.13: Forecast Results for Load Forecasting with Hyperparameters

Load dataset	MAPE	MSE	W	CV	CW/W
half node	0.0142	619.4281	1982.6264	0.9693	0.0371
layer drop	0.0142	617.6148	1966.6973	0.9715	0.0360
half epoch	0.01427	619.8332	1910.3554	0.9725	0.0369
cluster structure	0.0141	618.1464	19.18.2803	0.9655	0.0378

We can see that in Table 4.13, in general, boosted deep ensembles is robust against dramatical hyperparameter changes in neural networks as base learner. However, we conduct further experiments for several clusters/depth (structure of boosted deep ensembles) and layer of base learner (structure of base learner). The Table 4.14 shows how the number of clusters/depth affects the accuracy and time required to fit boosted deep ensembles.

Table 4.14: Forecast Results for Load Forecasting under Different Model Stuctures

Load dataset	MAPE	MSE	W	CV	CW/W	time
depth 1	1.4142	616.3317	1961.5604	0.9598	0.0489	647 sec.
depth 2	1.4196	618.1464	1918.2803	0.9655	0.0378	478 sec.
depth 4	1.4352	618.1924	1830.8610	0.9485	0.0542	317 sec.
depth 5	1.4303	620.3303	1980.9989	0.9386	0.0460	296 sec.
depth 10	1.4218	618.8671	1957.5921	0.9602	0.0402	249 sec.
depth 20	1.4172	617.4479	1910.2483	0.9587	0.0411	215 sec.

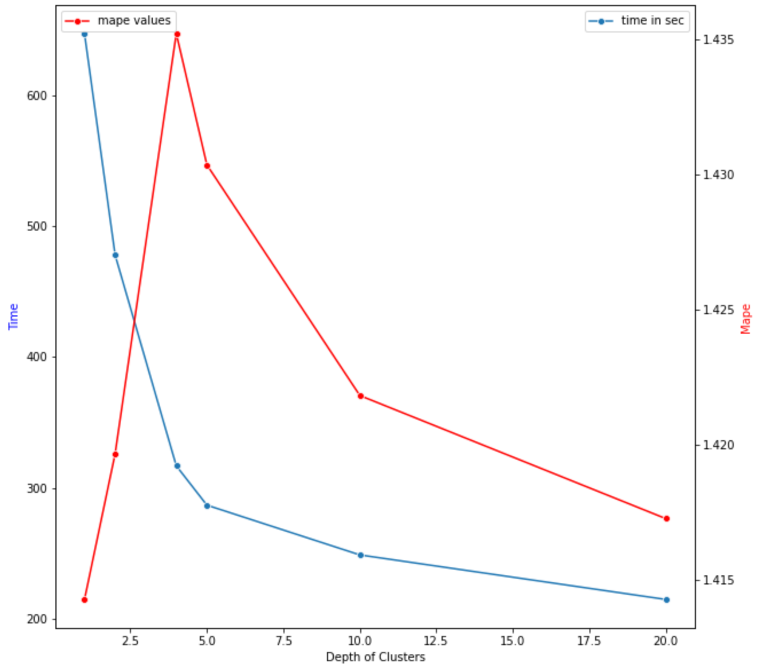

Figure 4.8: Time and Accuracy Comparison of Different Structure of Boosted Deep Ensemble

As we can see from the Table 4.14, as the depth of clusters increases, accuracy converges, mape values drop, and the required amount of time reduces. A simple elbow graph-like approach in Figure 4.8 would suffice to determine the optimal number of depth of clusters which seem to be 10 in this case. In addition to that, we also investigate the effect of the structure of base models on model accuracy, and results are given in the Table 4.15.

Table 4.15: Forecast Results for Load Forecasting under Different Base Model Structures

Load dataset	MAPE	MSE	W	CV	CW/W
5 layers	1.3671	611.0906	1909.4167	0.9335	0.0514
7 layers	1.3216	600.5076	1822.2311	0.9292	0.0542
9 layers	1.3120	595.8788	1929.0489	0.9433	0.0519
11 layers	1.3112	597.2265	1944.8873	0.9413	0.0510
13 layers	1.3214	597.3335	2212.0718	0.9593	0.0451

We see that the base model of the boosted deep ensembles model is a tunable parameter of the model. We stopp searching the space after mape values dropp sufficiently.

Next, we plot learning curves in Figure 4.9. We take the average loss values of ensemble members for each epoch and concatenate learning curves of models at each level. Boosted deep ensembles models with different structures almost converge to similar loss values, which shows us that learning transferring works properly, which we aim to show.

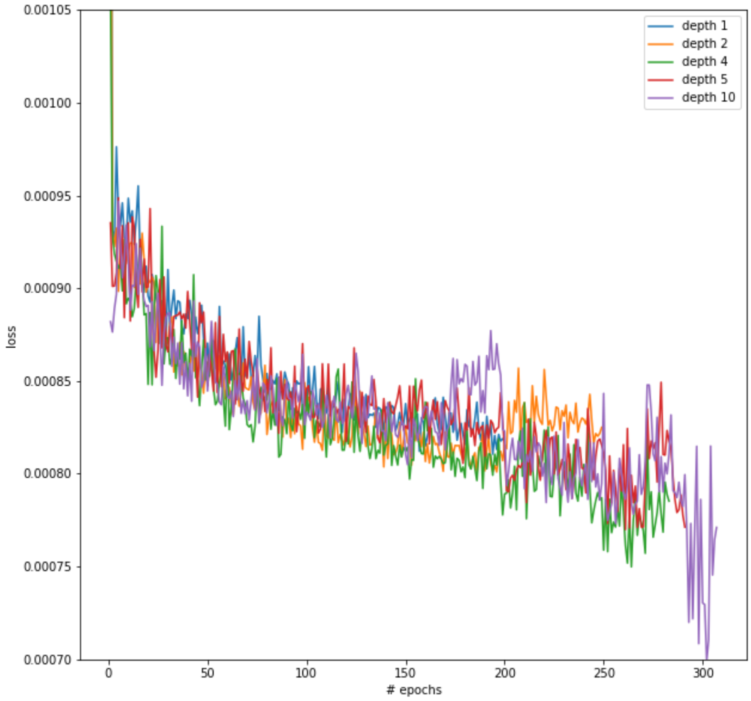

Figure 4.9: Learning Curves of Boosted Deep Ensembles with Different Depths

4.8 Ensemble selection

We also compare whether we can improve results by selecting ensemble members more sophisticatedly. To test that, we perform a simple model selection on the validation set by combining top-performing ensemble members and comparing it with ensembling through decorrelated ensemble members. We do it by mean-variance optimization (MVO), by which we optimally calculate ensemble weights so that mostly decorrelated members are selected as in portfolio optimization.

Optimal weighting problem for forecast combination is formally defined below. Given set of individual forecasts which are predictions from base model $f_1, f_2, \ldots, f_n$, the goal is finding a set of weights $w_1, w_2, \ldots, w_n$ which minimizes the forecast variance V constrainted to weights sum up to 1. More formally,

$$
\begin{aligned}
\min_{w} \quad & w\Sigma w^T \\
\text{s.t.} \quad & \sum_{i}^{n} w_i = 1 \\
& 0 \leq w_i \leq 1, i \in 1, 2, \ldots, n,
\end{aligned}
\tag{4.37}
$$

where Σ is the correlation matrix of forecast errors and $w\Sigma w^T$ is the variance. Results are given in Table 4.16:

Table 4.16: Results of Model Selection for Load Forecasting

Load dataset	MAPE	MSE	W	CV	CW/W
top 10	1.4265	617.8277	1399.4186	0.8488	0.0637
top 20	1.4232	618.0592	1520.6814	0.8862	0.0600
top 30	1.4133	616.5341	1620.7568	0.9079	0.0571
top 40	1.4112	615.7413	1648.6774	0.9159	0.0562
top 50	1.4122	615.9010	1691.4107	0.9242	0.0551
top 60	1.4117	615.9189	1721.2221	0.9293	0.0544
top 70	1.4115	615.8544	1763.7341	0.9369	0.0534
top 80	1.4043	614.9348	1803.6864	0.9442	0.0526
top 90	1.4130	616.7416	1852.4638	0.9484	0.0514
top 100	1.4142	616.3317	1961.5604	0.95984	0.0489
MVO	1.4646	627.7164	1240.5980	0.7852	0.0637

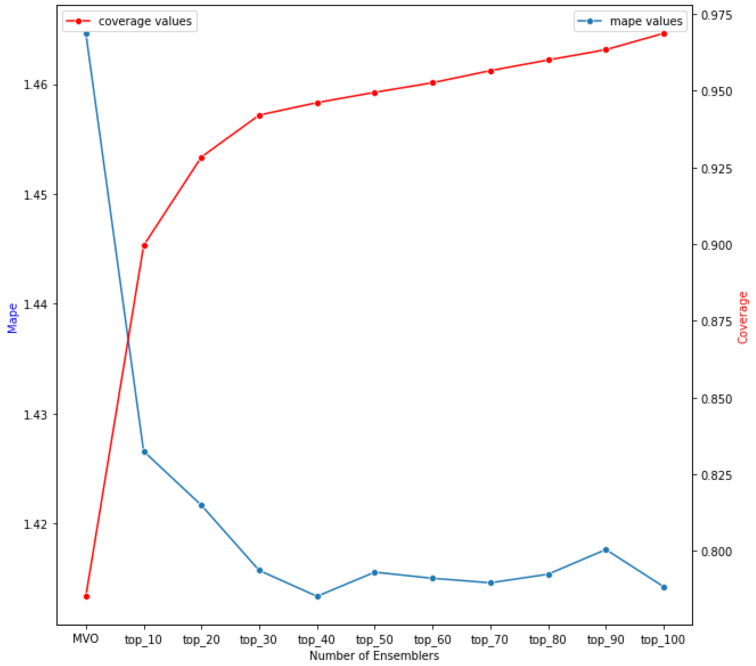

Figure 4.10: Accuracy with Different Number of Ensemble Members

We see that simple averaging outperforms averaging through MVO, which also has calibration issues for probabilistic forecasting. We conclude that averaging decorrelated members is insufficient to generate superior point and probabilistic forecasts. Also, we observe that increasing the number of ensemble members calibrates the forecasts. Moreover, leaving out ensemble members with poor performance increases the point accuracy. Thus, we conclude that more than ensembling through decorrelated members is needed to create superior forecasts. It requires leaving out poor-performance members to generate superior predictions. As shown in the Figure 4.11, ensemble members with mape values as high as $\%2$ need to be excluded from ensembling.

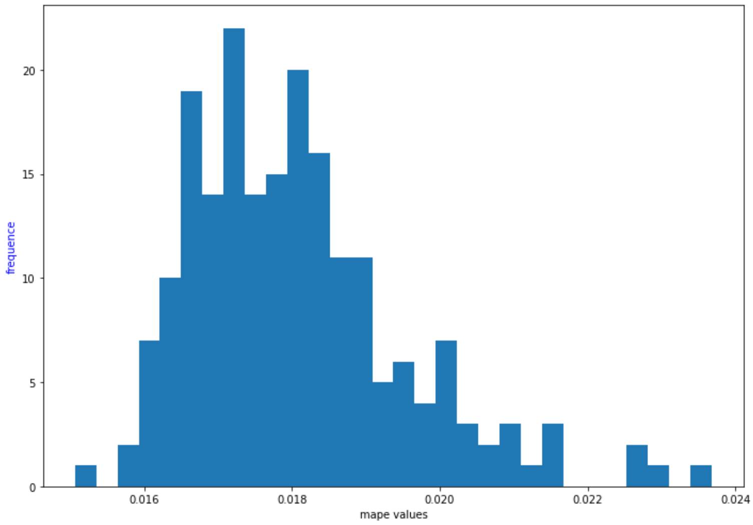

Figure 4.11: Distribution of Mape Values of Ensemble Members

Is cost funtion of NLL necessary?

In [95], authors state that cost function as mean squared error results underestimating predictive uncertainty. Having 200 base learners, we run a boosted deep ensemble model with a mean square error cost function to test whether predictions are normally distributed, and we can use empirical distribution to calculate predictive bounds. We see that 90% of the instance in the test set is normally distributed with having p-values higher than or equal to 0.1 based on KS test statistics. Moreover, considering the inferior performance of the mean forecasting result of deep ensembles (NLL), we see that a relatively more significant number of base learners secures to have normally distributed predictions. There is no need for fitting deep ensembles model with a negative log-likelihood cost function. We observe in Figure 4.12 histograms of randomly selected instances in the test set have normal distribution shapes.

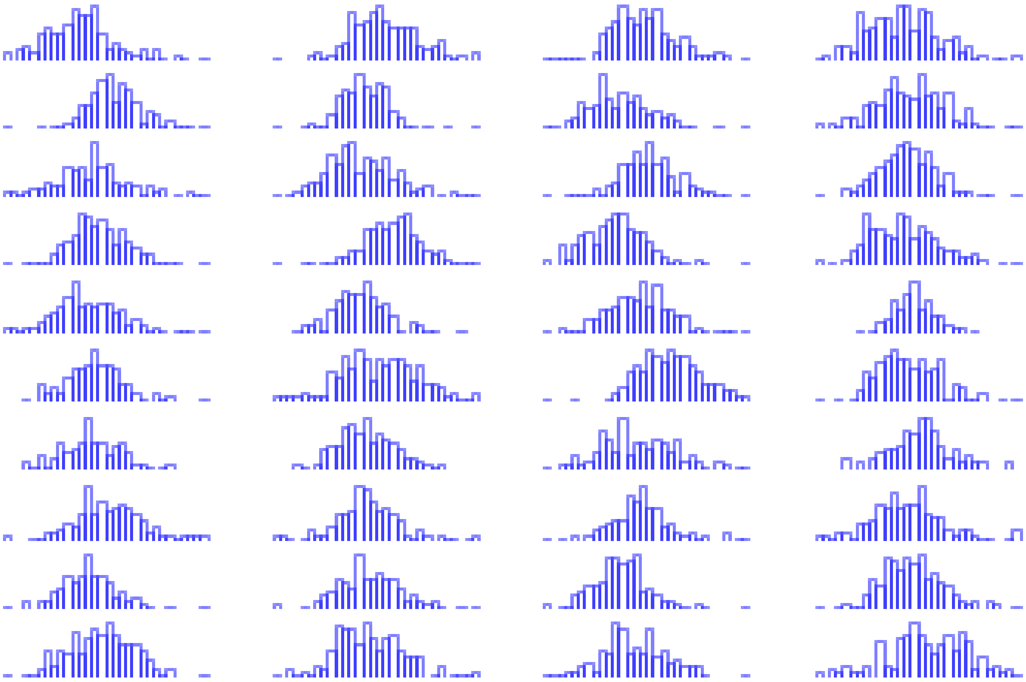

Figure 4.12: Histograms of Randomly Selected Predictions

4.9 Conclusion and Further Study

The deep ensembles model is a well-designed structure for point and probabilistic forecasting. In this chapter, we investigate ensemble learning and propose a novel architecture *boosted deep ensembles*, which benefits from transfer learning to decrease the computational complexity of deep ensembles while preserving model accuracy. We see that boosted deep ensembles are both theoretically and empirically time efficient than deep ensembles and able to generate predictions with statistically equivalent distributions as deep ensembles. We extensively test the model on tabular energy learning tasks to explore deep ensembles thoroughly. We see that the model structure of boosted deep ensembles and the model structure of base models is a tunable hyperparameter of the model. Moreover, we discover that a negative loglikelihood loss function unnecessarily hurts model performance, and the MSE loss function is sufficient to generate well-calibrated forecasts. Finally, we show that more than decorrelated ensembles are needed to generate superior predictions instead of leaving out poor performers.

CHAPTER 5

APPLICATION: COMPARISON OF TABULAR LEARNING METHODS FOR SHORT-TERM LOAD FORECASTING IN SPAIN

Several state-of-the-art (SoTA) tabular learning models have emerged as best performers on tabular data tasks in literature. On the contrary, several works show how traditional ML models, such as XGBoost and the LGBM model, outperform current SoTA models. We benchmark SoTA tabular learning models with deep ensembles, XGBoost, and plain CNN on energy load forecasting tasks. Moreover, we observe that, when pair-wise model ensembling applied, XGBoost model dramatically enhances the accuracy of each model it is ensembled with. We further, visualize the embedding space of the transformer model with t-SNE algorithm, since most SoTA models are based on transformer architecture. Next. we perform ablation study. We observe that transformer architecture is not able to generate embeddings in hidden space propoerly, and thus, it results in poor performance.

5.1 Introduction

Energy demand/load forecasting has received much attention recently and is vital for governments and private companies. Government agencies use long-term demand forecasts to plan economic growth efficiently; thus, they could create social benefits. Moreover, banks and financial institutions utilize long-term forecasts for investment valuation. Private companies related to the business, such as energy distribution companies and system operators, require both long-term and short-term forecasts to minimize losses and maximize system operation security. Load forecasting is complex due to dependency on the quality of weather forecasts, exhibiting nonlinearity and multi-seasonality and social factors like holidays.

Nowadays, tabular learning differs from image, audio, or video processing, where deep learning architectures have excelled and are receiving attention from scholars. Several SoTA models have been proposed recently, most based on transformer architecture. Energy forecasting tasks are a subclass of tabular learning tasks since energy datasets are structured datasets. In general, most of the energy dataset is time-indexed sensor measurements. Statistical models (e.g., ARIMA, GARCH) are still used for load forecasting literature. However, the current load forecasting literature is dominated by deep learning models.

Nevertheless, in literature, there is no consensus on models which produce more accurate forecasts [105]. The deep ensembles model is a well-suited model for predicting load forecasting since it can also produce point and probabilistic forecasts. In this chapter, we compare accuracies of recently proposed SoTA tabular learning models; namely, NODE [128], GANDALF [82], TabNet [3], FT-Transformer [56], TabTransformer [80] with deep ensembles [95] and well-known Xgboost [26] model which is reported as best performer in tabular learning tasks [58] and a plain CNN model. We see that the deep ensemble model, the CNN model, and XGBoost beats SoTA transformer-based tabular learning models are well suited for load forecasting tasks. Ensembling with XGBoost significantly boosts the performance of each model. We further inspect embedding space with the t-SNE algorithm and perform a model ablation study by which we conclude that transformer-based models are not able to represent categorical and numerical features as embeddings and hence result in poor performance.

5.2 Literature on Short Term Load Forecasting

Load Forecasting concerns the prediction of energy demand in different periods. Depending on the time scale, load forecasting can be categorized into three types: long-term, medium-term, and short-term forecasting. Short-term load forecasting predicts system load for one hour to one week, medium-term demand forecasting predicts one week to one year, and long-term load forecasting requires performing longer than one year. Moreover, very short-term load forecasting, 15 minutes to 1 hour, is usually made in smart grids to manage peak demand [74]. In general, long-term forecasting deals with power system planning required by government agencies; medium-term forecasting is needed for generation expansion planning and bilateral contracting. Short-term forecasting is vital for maintaining the real-time operation of the power systems [94]. As stated above, predicting electricity demand for an hour to one week is called short-term load forecasting (STLF), a vital task for energy companies. This procedure allows decision makers to maximize the company's economic benefits and is essential for demand response (DR) as well [32]. Moreover, a fast-growing literature on STLF has been dominated by probabilistic forecasting methods lately. The increasing share of renewable energy sources and sudden price fluctuations made energy markets more volatile. Decision-makers in the energy industry rely on forecasts to manage their market-related risks. Since point forecasts only provide limited information, i.e., one point as forecast, decision makers would instead use the probabilistic forecast to quantify uncertainties [74]. Probabilistic forecasts, such as density, interval, or quantile forecasts, by nature, convey richer information than point forecasts [74]. Therefore, probabilistic forecast has become more valuable for decision makers for evaluating and managing risk [47]. On the other hand, Hong [74] states that probabilistic methods for load forecasting have not received much attention over the past thirty years.

5.2.1 Point Forecasting

In literature, there is a variety of forecasting models have been applied for electricity load forecasting. Econometric models such as AR(I)(MA) [27, 23, 72], artificial neural network and recurrent neural network models [69, 127, 151] and support vector regression [110, 116] are employed in literature. Multiple Regressions is one of the most popular methods and widely adopted in forecasting short-term electricity demand problems by researchers [139, 115, 6, 123, 153, 22, 1]. The main input features for multiple regression models are related variables, observed demand values, and calendar variables [2]. A wide range of methodologies and models for forecasting are given in the literature, including multiple regression.

Time series models are also common in STLF literature since they could be considered a pure time series problem. However, only lagged observations of load values might be insufficient to perform an accurate forecast. Thus, exogenous variables are used in order to increase accuracy. [63, 25, 136, 84, 124]. [114] provides an extensive review on time series models for load forecast such as exponential smoothing, regression models, autoregressive models, ARMA, ARIMA and ARMAX models. Exponential smoothing models are well-known forecasting methods for time series forecasting. [144] introduced the double seasonal Holt-Winters method (DSHW) for short-term demand forecasting.

Together with well-established time series models, exponential models are prevalent in literature [145, 146]. Apart from traditional models, nonparametric models and machine learning models are prominent in short-term load forecasting. Models including SVM [24, 154], fuzzy model [112], grey model [155], and semiparametric additive model [47] is used for short term load forecasting in literature by researchers.

Lately, tree-based models have become one of the most used models in industry and academia for regression tasks. Regarding short-term load forecasting literature, Ben Taieb and Hyndman [9] develop a gradient boosting model for load forecasting in GEFCom 2012. Lloyd [104] also develops a gradient-boosting model for the same competition. Apart from GBM, [42] uses random forest to predict short-term electricity demand.

Machine learning models are extensively investigated in prediction tasks such as financial forecasting, supply chain forecasting, and energy price and demand forecasting. Short-term load forecasting has become one of the main tracks for researchers to improve forecast accuracy with machine learning models. Starting with simple shallow networks, RNN, LSTM, ResNet models are used in STLF [31, 17, 70, 29, 37, 10, 158, 149].

Lately, hybrid models have been prominent in STFL. Combining econometric and machine learning models such as ARIMA-NN [140], or hybrid modeling of different algorithms are common in STLF [44, 43, 118, 157]. Moreover, with the current adaptation of smart grids, load forecasting at the building level has become an essential topic for researchers [78, 28, 120, 7, 52, 109].

5.2.2 Probabilistic forecast

There is a growing literature on probabilistic load forecasting. With increasing popularity of Global energy forecasting competition, probabilistic load forecasting has become one of the mainstream in load forecasting [9, 74, 76, 100, 142, 38, 88]. In literature, there are three ways to generate probabilistic forecast [74]: Input scenarios, Model specification, Residual bootstrapping.

Input scenarios Since STLFs vastly rely on weather forecasts, based on weather forecast distribution, probabilistic forecast can be generated by bootstrapping from weather inputs or by selecting some weather scenarios [38, 132, 156, 147]. Having each scenario assigned equal probability, probabilistic forecast can be generated by calculating specific quantiles.

Model specification In order to have a probabilistic forecast, another approach is designing the model to estimate the distribution F_{th} of y_{th}. In that setting, it comes to estimating underlying conditional estimation of y_{th} given y_t and x_{th}. Conditional density estimation requires some unrealistic assumption and usually outputs poor results when the underlying distribution is not normal. Instead of making those assumptions, one can calculate quantiles for prediction using quantile regression and thereby can generate probabilistic forecasts [62, 57, 9]. The Bayesian approach is another option that calculates the predictive probability of the forecasts to generate probabilistic forecasts [117].

Residual bootstrapping Post-processing point forecast outputs are a method to create probabilistic forecasts. After the forecast, the error term could be modeled based on some probability density function [47]. Also, [77] uses the GARCH model to produce the probabilistic forecast. One could also combine several point forecasts to produce probabilistic forecasts [100].

5.3 Analysis of Energy Load Dataset

In this section we examine dataset, variables and stylized facts about short term load forecasting. We give main variables, check correlations among variables, investigate time series properties such as periodicity, autocorrelation.

In literature, there are four main variables for STFL, these are:

Weather variables consisting of temperature, lagged values of temperature, temperature values from multiple areas, Daily min-max temperature values, polynomial transformation of temperature, humidity, wind, and cloud cover.

Calendar variables consisting of day of the week, month of the year, hour of the day.

Special days consisting of religious and official holidays.

Lagged demand values including demand values from last week, demand values from the same week of the last year, demand values for the same week from last month, and min, max, and mean of lagged demand values.

5.3.1 Dataset

We run experiments Spain load dataset [81] contains four years of energy demand and weather data for Spain. Energy demand data is retrieved from ENTSO-E [46], a public portal for Transmission Service Operator (TSO) data. As an input variable, weather data is retrieved from the Open Weather API [121] for the five largest cities in Spain. However, to be concise, we run experiments with variables from Valencia. This dataset is used in following papers [4, 130] and [133]. Energy demand (load forecasting) forecasting is an important task for energy market participants.

Daily distributed energy data contains multiple periodicities. The first one is seasonal periodicity, with an increase in demand for electricity in summer and winter and a decrease in springs. Upward trend could be seen in Figure 5.1 and downward trend could be seen in Figure 5.2.

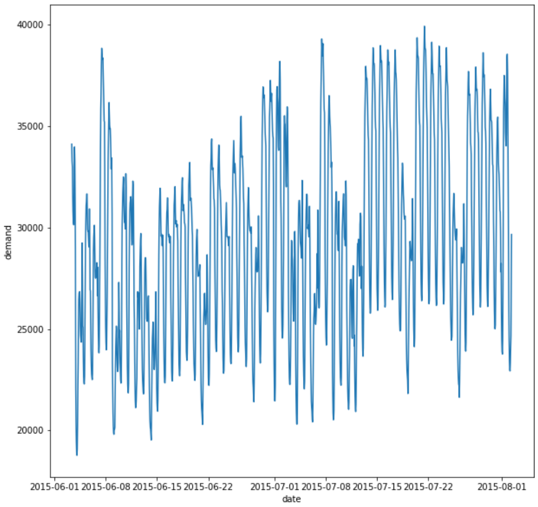

Figure 5.1: Upward Trend

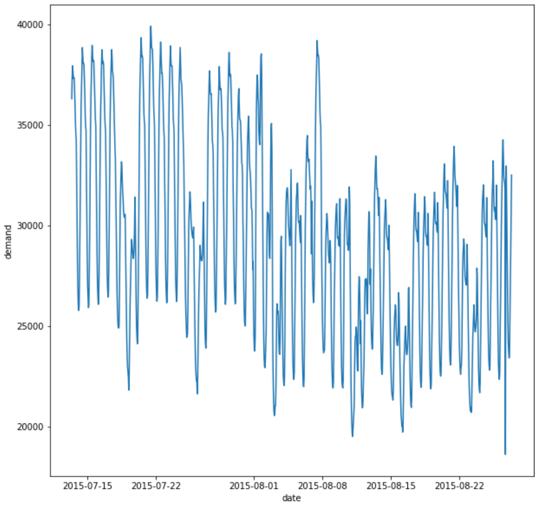

Figure 5.2: Downward Trend

The other periodicity in this series is the weekly periodicity. For each Sunday in each week, there is a significant drop in electricity demand which could be seen in Figure 5.3.

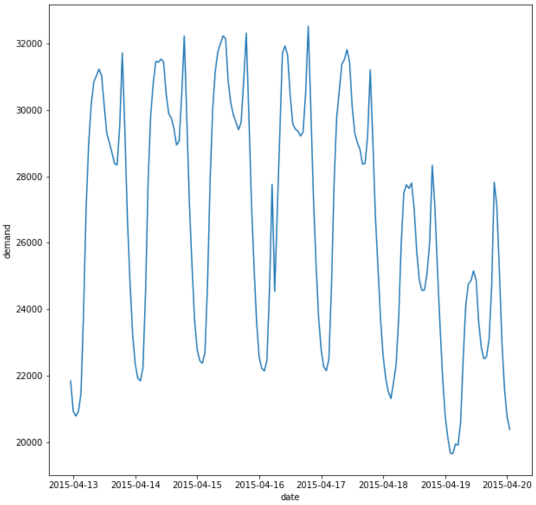

Figure 5.3: Weekly Periodicity

The last periodicity is the one that occurs within a day. For each weekday, electricity demand reaches peak values in midday and just before the evening could be seen in Figure 5.4.

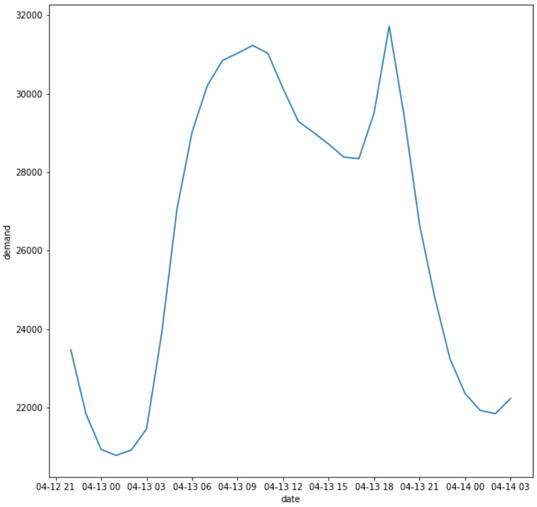

Figure 5.4: Hourly Periodicity

As mentioned above, weather variables are one of the main covariates of the distributed energy. In the summer, demand for electricity reaches the top due to air conditioning needs in the region. Here, we plot the correlation of distributed energy with weather variables in Figure 5.5.

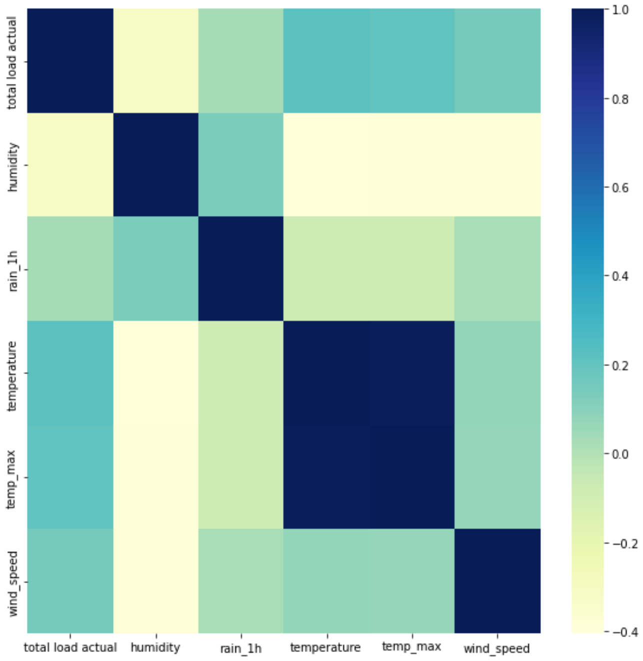

Figure 5.5: Correlation of Spain Energy Demand

Another significant challenge in forecasting short-term load is the deviation in special days. The Figure 5.6 below shows how regime shifts in short-term electricity demand on New Year's eve and the next day.

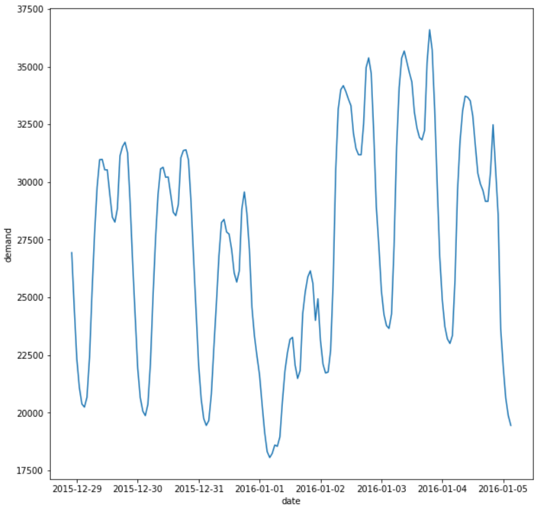

Figure 5.6: New Year's Week

There is a strong correlation with lagged values of distributed energy series. One can observe that there is a significant correlation within a month. Therefore, lagged values of distributed energy series are one of the critical predictors of the series which could be seen in Figure 5.7.

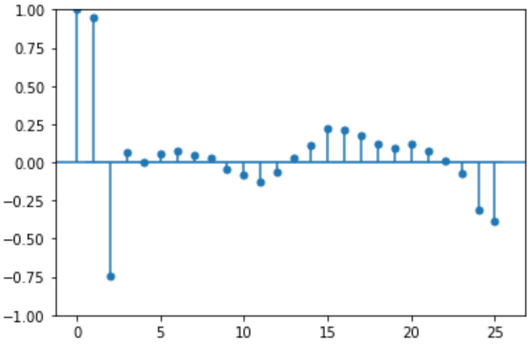

Figure 5.7: Partial-autocorrelation within a day

5.4 Experimental Methodology

We benchmark forecasting accuracies of SoTA tabular learning models with XGBoost, Boosted Deep Ensembles and, plain the CNN model in short term load forecasting. Further, we pairwisely ensemble models. Moreover, we visualize encoder-decoder space of TabNet model using t-SNE algorithm [152]. Then, using model ablation approach, we assess abiliy of representational power of TabNet's transformer architecture and in general transformer architecture.

Machine and Deep Learning Models for Point Forecast

In this section, we define the models used in this study. Namely, XGBoost, Boosted Deep Ensembles, CNN, TabNet, TabTransformer, NODE, GANDALF, and FTT models.

Gradient Boosted Tree Regression (GBM)

Gradient boosting regression is a boosting model that fits a decision tree to residuals from previous base learners, and works in a sequential order to improve results from base learners.

Consider a GBM with M stages; where for each stage m, we define $g(x)$ base learner which is then fitted to $e_m = y - F_m(x)$, where F_m is the boosting model at step stage m and the goal is minimize the squared loss function via gradient descent algorithm.

More formally, for $\{x_i, y_i\}$ in the training set, we define

$$F(x) = \sum_{i=1}^{M} g_i(x) + g_0,$$

where g_i some base learner g_0 is the constant and ψ_i is the weight to be determined. Starting with a constant model g_0; at each step m, we compute

$$F_m(x) = F_{m-1}(x) + f_m(x),$$

where

$$f_m(x) = \text{argmin}_f \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + f(x_i)).$$

Convolutional Neural Network (CNN)

Convolutional neural network is a powerful tool for image processing and classification, video recognition tasks. It is a sub-model, belongs to neural network family. Similar to multilayer perceptron, CNN is a layered structure, and at each layer it convolves input space with filters to extract features from it and thereby generates results.

Tabular Deep Models

We give the latest SoTA (State-of-the-Art) models developed for tabular data tasks. These models are generally based on a transformer-based, encoder-decoder structure where the input set is mapped into lower space by the encoder and then reconstructed by the decoder as embeddings. The main reasoning is that categorical variables might remain redundant and uninformative when one-hot encoded. Thus, these models convert categorical and numerical variables into embeddings via an encoder-decoder structure. Then, they send this information to predictions via complex nonlinear transformations.

Neural Oblivious Decision Ensembles. The NODE model uses oblivious decision trees where each tree is linked to a neural network. This augmentation refines predictions by learning intricate feature interactions and transformations, which enables capturing diverse facets of data features with decision trees. At the same time, neural networks uncover complex patterns beyond the trees' scope. Through training, decision trees leverage gradient boosting, aligning with ensemble residuals, and neural network modules refine their respective tree predictions, yielding heightened precision and accuracy compared to using decision trees or neural networks in isolation.

TabNet. Using four transformer structures, TabNet empowers categorical and continuous features by transforming feature embeddings to generate predictions. It uses sequential attention mechanisms where attention mechanisms learn weights to select relevant features while filtering out noise. This architecture comprises alternating decision points and action layers, orchestrating sequential decisions to process input progressively. The resultant binary decisions, shaped through attention-guided selection, are concatenated to form a mask indicating active features. The subsequent action layers execute operations on this masked input, contributing to the model's prediction. Notably, TabNet's interpretability stems from its ability to unveil feature importance through attention scores.

TabTransformer. TabTransformer, a deep tabular learning model, is used for supervised and semi-supervised learning. Transformers based on self-attention serve as the foundation of the TabTransformer model. The transformer layers convert categorical feature embeddings into robust contextual embeddings to increase prediction accuracy. The design comprises a multi-layer perceptron (MLP), a stack of transformer layers, and a column embedding layer. Concatenated continuous features and the contextual embeddings (produced by the transformer layer) are fed into an MLP.

FT Transformer. A straightforward transformation of the transformer architecture for the tabular domain is the FT-Transformer (Feature Tokenizer + Transformer). All features—categorical and numerical—are converted to tokens by the Feature Tokenizer component, which then applies a stack of transformer layers to the tokens to perform operations at the feature level of a single object for each transformer layer.

GANDALF. Gated Adaptive Network for Deep Automated Learning of Features (GANDALF) is a novel method with high-performance, interpre , parameter- and computationally efficient tabular deep learning architecture. GANDALF uses the Gated Feature Learning Unit (GFLU), a novel tabular processing unit with a gating mechanism and built-in feature selection, as a feature representation learning unit.

5.5 Experiments

We compare the model performances on the test set with the error metrics used below. We compare the XGBoost, Boosted Deep Ensembles, and CNN models with recent SoTA tabular learning models, namely TabNet, TabTransformer, NODE, GANDALF, and FTT models.

Dataset Preprocessing

We use min-max normalization for both datasets to compare results fairly. Scikit-learn library [125] is employed for this normalization. For each datasets, both input and target variables are normalized using

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}. \tag{5.1}$$

Hyperparameters

Default parameters for XGBoost model are library's defaults with 200 base learners. All neural networks, run for 100 epochs, have 4 layer and 32 nodes in each layer except for final layer. Deep ensembles run with 100 base learners.

Training

We split both datasets as follows: 65% for training, 10% for validation and 25% for testing with 5-fold cross validation using a rolling window.

Metrics

For mean accuracy, we use mean square error (MSE) and mean absolute percentage error (MAPE) and BIAS values.

$$\text{MSE}(y_i, \hat{y}_i) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

$$\text{MAPE}(y_i, \hat{y}_i) = \frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \hat{y}_i)^2}{y_i}.$$

$$\text{BIAS}(y_i, \hat{y}_i) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i),$$

where y is point to be forecasted, $\hat{y}$ is the prediction.

5.6 Results

In this section, we provide point forecasting results for tabular models. In order to evaluate point forecast results, we use mean absolute percentage error (MAPE), root mean squared error (RMSE), and BIAS values.

Table 5.1: Results for point forecast

	MAPE	BIAS	MSE	Time
XGBoost	1.5191	-47.7980	671.8364	61 sec.
TabTransformer	2.2980	119.8849	866.9491	316 sec.
TabNet	2.7213	-199.1947	1041.8563	245 sec.
NODE	3.2525	-61.7109	1252.9572	418 sec
CNN	1.7743	203.0704	739.4485	114 sec.
FTT	2.1143	-204.7617	803.9376	466 sec.
Boosted Deep Ensemble	1.4352	7.1619	618.1924	212 sec.
GANDALF	2.1216	-1.6584	851.5260	398 sec.

Results in Table 5.1 state that Deep Ensembles, Xgboost, and CNN outperform SoTA tabular learning models. Tabular learning models' performance is generally higher than MAPE of 3.5% except the TabNet model. |

Parameter Tuning

To investigate further the accuracy of models, we tune the parameters of the models with grid search. The tuned parameters of each model are given in the Tables 5.2, 5.3 below:

Table 5.2: Parameters tuned

	TabNet	XGBoost	Boosted Deep Ensembles
Learning Rate	[1e-02,1e-03, 1e-04]	[1e-02,1e-03, 1e-04]	[1e-02,1e-03, 1e-04]
Dropout	[0.1, 0.2,0.3]	-	[0.1, 0.2,0.3]
Tree Depth	-	[3,4,5]	-
# Estimators	-	[50,100,200]	[50,100,200]
Batch Size	[32,64,128]	-	[32,64,128]

Table 5.3: Parameters tuned

	TabTransformer	GANDALF	CNN	NODE
Learning Rate	[1e-02, 1e-04]	[1e-02, 1e-04]	[1e-02, 1e-04]	[1e-02, 1e-04]
Dropout	[0.1, 0.2,0.3]	[0.1, 0.2,0.3]	[0.1, 0.2,0.3]	[0.1, 0.2,0.3]
GLFU Stages	-	$\{2, 3, \ldots 10\}$	-	-
Tree Depth	-	[3,4,5]	-	[3,4,5]
Batch Size	[32,64,128]	[32,64,128]	[32,64,128]	[32,64,128]

Table 5.4: Parameters tuned results for point forecast

	MAPE	BIAS	MSE
XGBoost	1.3976	-36.2315	550.2232
TabTransformer	2.0166	51.7392	652.5389
TabNet	2.3103	-102.5271	835.3379
NODE	3.1012	-58.1046	805.0328
CNN	1.7047	179.6105	513.8560
FTT	2.0011	-106.6344	625.6109
Boosted Deep Ensemble	1.3656	-13.3102	620.9831
GANDALF	1.9816	73.2392	675.6321

In Table 5.4 we see that there are performance improvements in every model, which is expected, yet still, tabular SoTA models are not able to outperform a basic CNN, Xgboost, and deep ensembles model.

5.6.1 Pairwise Model Ensembling

Since each SoTA tabular learning model has a different structure, we would like to combine each model pairwisely and investigate weather the model ensembling approach could achieve better than individual results. See Tables 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, 5.12.

Table 5.5: Results for ensemble with FTT model

	Xgboost	TabTransformer	TabNet	NODE	CNN	GANDALF	Boosted Deep Ensemble
mape	1.6408	1.9464	2.2093	2.3908	1.7609	1.9868	1.5873
bias	104.7799	42.4384	201.9782	133.2363	0.8457	103.2101	102.2978
rmse	684.4035	800.0301	858.0254	932.4225	730.2794	799.4323	666.1030

Table 5.6: Results for ensemble with TabNet model

	XGBoost	TabTransformer	NODE	CNN	FTT	GANDALF	Boosted Deep Ensemble
mape	1.9178	2.2045	2.7993	1.9680	2.2093	2.1780	1.8643
bias	101.9963	39.6549	130.4528	-1.9378	201.9782	100.4266	99.5143
rmse	768.5335	887.9737	1066.4400	805.9115	858.0254	866.4673	746.9299

Table 5.7: Results for ensemble with Xgboost model

	TabTransformer	TabNet	NODE	CNN	FTT	GANDALF	Boosted Deep Ensemble
mape	1.6566	1.9178	2.1352	1.5454	1.6408	1.6746	1.4061
bias	-57.5434	101.9963	33.2544	-99.1362	104.7799	3.2283	2.3160
rmse	706.1316	768.5335	850.1765	664.6759	684.4035	699.1025	613.3829

Table 5.8: Results for ensemble with CNN model

	XGBoost	TabTransformer	TabNet	NODE	FTT	GANDALF	Boosted Deep Ensemble
mape	1.5454	1.8572	1.9680	2.2276	1.7609	1.8378	1.5067
bias	-99.1362	-161.4776	-1.9378	-70.6798	0.8457	-100.7059	-101.6182
rmse	664.6759	780.1898	805.9115	903.5937	730.2794	760.0651	646.4890

Table 5.9: Results for ensemble with Deep Ensembles model

	XGBoost	TabTransformer	TabNet	NODE	CNN	FTT	GANDALF
mape	1.4061	1.6125	1.8643	2.0485	1.5067	1.5873	1.6305
bias	2.3160	-60.0255	99.5143	30.7724	-101.6182	102.2978	0.7462
rmse	613.3829	686.9629	746.9299	819.5848	646.4890	666.1030	680.0367

Table 5.10: Results for ensemble with TabTransformer model

	XGBoost	TabNet	NODE	CNN	FTT	GANDALF	Boosted Deep Ensemble
mape	1.6566	2.2045	2.4896	1.8572	1.9464	2.0097	1.6125
bias	-57.5434	39.6549	-29.0870	-161.4776	42.4384	-59.1132	-60.0255
rmse	706.1316	887.9737	995.9942	780.1898	800.0301	827.8218	686.9629

Table 5.11: Results for ensemble with GANDALF model

	XGBoost	TabTransformer	TabNet	NODE	CNN	FTT	Boosted Deep Ensemble
mape	1.6746	2.0097	2.1780	2.4031	1.8378	1.9868	1.6305
bias	3.2283	-59.1132	100.4266	31.6847	-100.7059	103.2101	0.7462
rmse	699.1025	827.8218	866.4673	949.7642	760.0651	799.4323	680.0367

Table 5.12: Results for ensemble with NODE model

	XGBoost	TabTransformer	TabNet	CNN	FTT	GANDALF	Boosted Deep Ensemble
mape	2.1352	2.4896	2.7993	2.2276	2.3908	2.4031	2.0485
bias	33.2544	-29.0870	130.4528	-70.6798	133.2363	31.6847	30.7724
rmse	850.1765	995.9942	1066.4400	903.5937	932.4225	949.7642	819.5848

We observe that by ensembling with the XGBoost model dramatically enhances the performance of SoTA tabular data models. Lowering each MAPE value to below 3%. Xgboost model is the best contributor to ensembles with other models, followed by the deep ensembles model.

5.7 Model Ablation Study

In order to inspect why transformer-based tabular models have poor performance, we first visualize embeddings and raw dataset with t-SNE algorithm [152] and then perform a model ablation in which we ablate the remaining part of the TabNet model after extracting embeddings. Then, feed both embeddings and raw data into two straightforward models, linear regression and single layer MLP, and fit models and compare results to assess how embeddings space represents the raw data set. The t-SNE algorithm is a dimensionality reduction algorithm for identifying significant patterns in high-dimension datasets. The capacity to retain local structure is the key benefit of t-SNE. This, in general, suggests that points near each other in the high-dimensional data set will likely be close to each other in the lower dimensions. The t-SNE approximates the probability distribution of neighbors around each point. Using t-distribution, t-SNE approximates the Gaussian distribution of the original dataset into 2-dimensional space. This generates a 2-dimensional mapping that minimizes the distance between these two distributions over the dimensions. Compared to a Gaussian, the larger tails of a t-distribution distribute the points more uniformly in the 2-dimensional space. As a powerful tool to visualize data [141, 8], we both map raw data and embeddings from the TabNet model to 2-dimensional space using the

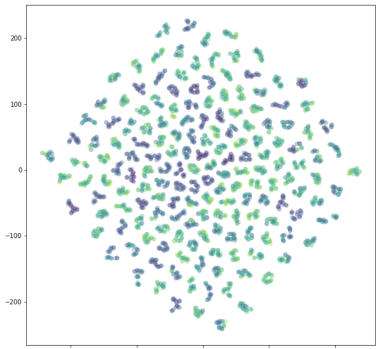

Figure 5.8: t-SNE Reducted Raw Dataset

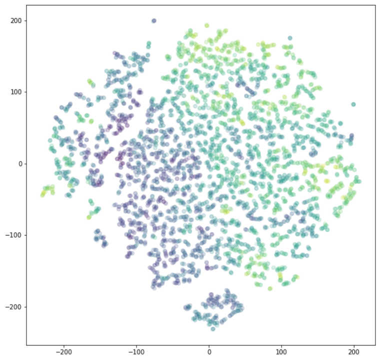

Figure 5.9: t-SNE Reducted Embeddings

t-SNE algorithm to observe patterns in lower dimensions.

In Figure 5.8, we see that the raw data set with one hot encoding is concisely clustered separately where high (darker colors) and lower demand (lighter colors) values are near each other. On the other hand, in Figure 5.9, there is no clear separation of data points, which indicates that embedding space is not informative enough. In order to test, we perform a model ablation study in which we get embedding from the TabNet model, leave the rest of its architecture, and replace it with simple models. After getting embeddings from TabNet, we use two plain models to assess whether categorical embeddings work correctly for load forecasting problems. To do that, we fit a linear regression with the raw dataset having one hot encoding and fit with fetched embeddings. To consider nonlinearities, we also used a single-layer MLP model and fit it in the same manner as in the linear regression model.

Table 5.13: Results of Linear Regression and MLP model

	MAPE	BIAS	RMSE
LinReg with Embedding	3.4955	-26.6617	1320.9116
MLP with Embedding	3.3470	-11.4624	1284.4998
LinReg with Raw Data	2.5764	84.2797	772.8648
MLP with Raw Data	2.2442	24.7444	692.7155

By examining accuracy metrics of models, we see that in Table 5.13, raw data with one hot encoding works better than transforming variables into embeddings. Considering this result together with results from Table 5.1 and the visuals generated in Figures 5.8 and 5.9, we conclude that models using base architecture as transformers do not work properly for energy load forecasting tasks. Feature embeddings are not adequately generated; raw data with one hot encoding works better.

5.8 Conclusion and Further Study

In this chapter of the thesis study, we examine the literature for short-term load forecasting, a tabular learning task, and provide stylized facts of the energy load dataset. We benchmark SoTA tabular learning models against the boosted deep ensembles, plain CNN, and XGBoost models. Results reveal that the deep ensembles model is well suited for load forecasting. Together with CNN and XGBoost, deep ensembles outperform SoTA tabular learning models. We then examine embedding space in two-dimensional space using the t-SNE algorithm and perform a model ablation study, revealing that transformer-based models are inefficient for load forecasting tasks compared to the CNN and the XGBoost, the boosted deep ensembles.

CHAPTER 6

A NOVEL FEATURE SELECTION FRAMEWORK WITH DEEP ENSEMBLES

Tabular models inherit feature importance results from transformer architecture. As we show that deep ensembles and boosted deep ensembles perform successfully in tabular tasks, it can be extended to a feature selection framework since it is based on ensembling where variable permutation feature importance or variable drop feature importance could be integrated. In this chapter, we integrate both variable drop and variable permutation feature selection algorithms into deep ensembles. To measure the performance of proposed algorithms, we compare weighted mape values of each model, namely random forest, gradient boosting, and deep ensembles trained on complete dataset D, with models trained on dimension-reducted datasets based on both feature importance and feature reduction algorithms.

6.1 Introduction

Accurate wind power generation prediction is required and useful for energy market participants such as energy generators, trading companies, dispatchers, and system and market operators. The system operators need accurate predictions to schedule the reserve capacity and to maintain the grid operations safely. Accurate and reliable predictions of renewable power generation are essential for electricity transmission and are also essential for competitive renewable energy supply [90]. Moreover, accurate predictions allow utilities to have favorable trading performance on the electricity markets [20].

In literature, physical, statistical, and machine learning-based models and their combinations are used to perform wind power predictions. Generally, data-driven wind power forecasting is performed via numerical weather prediction (NWP), which is used as input variables to map wind plant generation. NWP is provided from spatially closest points to the turbines as zonal and meridional components of winds, and necessary calculations are made from NWP data such as wind speed and wind direction. Generally, more than one point is selected for each turbine to ensure the wind profile is met for specific locations. However, this might result in redundant input variables for the model. In order to overcome redundant variables and reduce computational complexity, feature selection emerges as a critical method for this problem.

Feature selection remains an active research area, though a significant amount of work has been done due to a significant increase in ML research; it has many benefits: (i) reducing the complexity of computation for prediction; (ii) removing information redundancy (cost savings); (iii) avoiding the issue of overfitting; and (iv) easing interpretation [148]. Though wind power forecasting and feature selection are well-studied topics, research on feature selection for wind power forecasting is limited.

In this chapter, we first investigate and then integrate two feature importance algorithms into a deep ensemble model and test the accuracy of feature importance algorithms by comparing them with base models trained on a complete dataset.

This chapter contributes to the following: We integrate variable drop and variable permutation feature importance algorithms into the deep ensemble and test the accuracy of those algorithms against benchmark models.

6.2 Related Work

Wind forecasting is generally considered as very short-term (few seconds to 30 min), short-term (30 min to 6 h), medium-term (6–24 h), and long-term (1–7 days) [159]. Short-term predictions would yield higher accuracy due to the deterioration of NWP in longer terms.

Wind forecasting has been a long-studied topic in literature, and there are several very detailed reviews in wind power forecasting to benefit from. In [85], an overview of existing research on wind speed and power forecasting is provided. State-of-the-art wind speed and power forecasting approaches and forecasting accuracy based on variable factors and forecast performance improvements are presented in detail.

Wind forecasting literature is dominated by four approaches: physical methods, statistical methods, and ML-based methods. In addition to this, their combinations are considered as hybrid methods.

6.2.1 Physical Model

Using equation below, wind power genetaration could be estimated from the given variables:

$$P = \sigma A v^3 C_p, \tag{6.1}$$

where P is turbines rated power in (W), σ stands for air density in kg/m^3. A is turbines swept area that could be calculated via $A = \pi r^2$), r is turbins blade length. v is wind speed in $\mathrm{m/s}$, C_p is the power coefficient and turbine efficiency, and is lower than 0.45, and its maximum value is called Betz limit equals to 0.59 [18].

6.2.2 Statistical Models

In [35] and [21], detailed literature reviews are provided for statistical models for wind power prediction. Autoregressive models such as AR, MA, ARMA, ARIMA, ARMAX are the most used models for this task.

6.2.3 Machine Learning Models

The popularity of using machine learning models is also presented in wind power forecasting. A literature review [107] and [108] provide mainly used models for wind power prediction tasks, including MLP, ANN, DNN, CNN, RNN, SVR, CART, RF, GBM.

6.2.4 Hybrid Models

Hybrid modeling is a more sophisticated use of several models by combining statistical and machine learning methods. Recently, hybrid modeling has significantly increased in wind power prediction. For related review papers, we refer to [107] and [35].

6.2.5 Deterministic and Probabilistic Models

Probabilistic forecasting has recently become popular in energy forecasting due to the richer information provided to users. Moreover, competitions in Kaggle and GEF [75] require predictions with their prediction intervals [160] provides recent developments in probabilistic wind power forecasting. On the other hand, we refer to [150] for deterministic wind energy forecasting.

6.2.6 Dimension Reduction

Advances in computational power, big data, and complex models make model training times longer. Therefore, data size reduction is crucial, especially when run times are narrower. Dataset size reduction can be performed in two ways: feature set reduction or sample set reduction [83].

Generally, more input variables carry more discriminating power, but in practice, excessive variables are prone to cause many problems, such as prolonged training periods and the curse of dimensionality [61].

Dimension reduction could be performed in feature extraction (FE) and feature selection (FS). Feature extraction maps input space to a lower dimensional space through linear or nonlinear mappings such as LDA or NLDR. The main drawback is that new feature space has no meaningful interpretation [16].

Generally, more input variables carry more discriminating power, but in practice, excessive variables are prone to cause many problems, such as prolonged training periods and the curse of dimensionality [61].

6.2.7 Feature Extraction

Feature extraction maps input space to a lower dimensional space through linear or nonlinear mappings such as LDA or NLDR. The main drawback is that new feature space has no meaningful interpretation [16].

Feature selection is filtering relevant input variables for modeling. It could be grouped into three classes: Filter methods, Wrapper methods, Embedded and hybrid methods.

Filter methods. Uses relevant metrics such as correlation, fisher score, mutual information, redundant features filtered through a threshold.

Wrapped methods. Uses a greedy algorithm, the features that yields best results based on some accuracy metrics are chosen.

Embedded methods. Embedded methods perform feature selection and training of the algorithm in parallel. In other words, the feature selection process is integral to the classification/regressor model.

6.2.8 Feature Importance

Feature importance algorithms are generally greedy algorithms that require running a base model repetitively, which is expensive yet the only solution for model interpretability for now. Other than linear models, which are interpretable, feature importance algorithms are built on tree-based models such as random forest or gradient-boosting tree models. Feature importance algorithms evaluate the features by generating a subset of features given for a problem and then compare the relative increase or decrease in accuracy with respect to the feature set. Thus, it is a natural extension for ensemble models. [98] proposes a feature support metric that averages the accuracy of single base models whose training set includes specific variables, and then it sorts each support value as feature importance. [15] proposes the permutation importance algorithm, based on a random forest model trained on a complete train dataset and shuffling the values of features in a test dataset to measure how disturbing the relation of input and output variable affects accuracy.

6.2.9 Feature Selection in Wind Power Forecasting

Feature selection in wind power forecasting is concentrated on embedded algorithms. In [99], a proposed method combines wavelet decomposition, neural networks, feature selection, and regression for the generation forecasting of a wind farm is proposed. In order to choose a compact set of input features, a feature selection technique based on mutual information is developed for the forecasting model. In [97], ridgelet transform is considered as a feature selection algorithm, and it is combined with a neural network-based prediction algorithm. The work in [103] stacks wavelet transform with PCA and SVM model and uses particle swarm optimization for hyper-parameter optimization. Also, [137] introduces a novel approach combining feature selection with recurrent neural networks (RNN). Furthermore, [48] presents a deep feature selection framework to set input features to neural networks, which produces both deterministic and probabilistic forecasts.

6.3 Model

This study embeds two feature importance algorithms with the deep ensembles model, which uses shallow neural networks as base learners. The deep ensembles model generally does not require a subset selection procedure as for random forest models. However, to implement variable importance, we univariately drop the variable from a given feature set to measure the importance of each variable. Firstly, we drop each feature at a time per base learn training to measure the impact of specific variables on model accuracy. For this, we build a deep ensemble model with size M where M is the number of input variables and drop each feature, resulting in a base learner with $M-1$ input variable. Pseudocode of this algorithm could be found in Algorithm 4.

Algorithm 4: Calculating feature importance with variable drop

```
Input: Dataset $D$ with dimension $M$, base neural network $g$ to be fit, array of
       weighted mape values of base deep ensemble model trained on
       complete dataset $D$
Output: Feature importance of variables
for $m \leftarrow 1$ to $M$ do
    fit base model $g_m$ on dataset $D - m$
    $wmape_{g_m} \leftarrow$ calculate weighted mape values
    $d_m \leftarrow wmape_{g_m}$-$wmape_{DeepEnsemble}$
    return Average of $d_m$
end
```

Secondly, we integrate the permutation feature importance algorithm into the deep ensembles model. The algorithm is straightforward for this importance procedure. After training deep ensembles model on training dataset D, values of each input variable are shuffled so that the relationship between the input variable and output variable is distorted in the model testing dataset phase. Calculating weighted MAPE values for each base learner, which uses permutated input variables, generates permutated feature importance values. Algorithm 5 is the slightly changed version of Algorithm 4 for calculating permutation importance:

Algorithm 5: Calculating feature importance with variable permutation

```
Input: Dataset $D$ with dimension $M$, base neural network $g$ to be fit, array of
       weighted mape values of base deep ensemble model trained on
       complete dataset $D$
Output: Feature importance of variables
for $m \leftarrow 1$ to $M$ do
    get model predictions of base model $g_m$ on dataset $D_{m_{permutated}}$
    $wmape_{g_m} \leftarrow$ calculate weighted mape values
    $d_m \leftarrow wmape_{g_m}$-$wmape_{DeepEnsemble}$
    return Average of $d_m$
end
```

6.4 Dataset

We run experiments on an anonymous wind power plant in Turkey. The dataset for this application is open at https://seffaflik.epias.com.tr/, and NWP values as input variables are gathered from the free API of https://www.meteomatics.com/. Wind observations in the input dataset have two components, u and v. The zonal component (or x-coordinate) is denoted as u and has the west–east direction or vice versa. Meridional component (or y-coordinate) follows a flow from north to south, or vice versa, and is denoted as v. Input variables for the experiments collected from 17 points in the turbine site. As a result, we have 34 measurement points from the turbine site. The training set for this experiment covers 2021-04-01/2021-10-31, and the test set is chosen to be 2021-11-01/2021-11-30. In Tables 6.1 and 6.2 we give a descriptive table for selected points in the wind field to save space:

Table 6.1: Descriptive statistics of selected variables

	wind generated	**windUComponent867**	**windUComponent2385**	**windUComponent2381**
count	12270	12270	12270	12270
mean	14593.5420	0.6927	1.0036	2.9002
std	12188.1779	4.4274	3.8221	5.0098
min	0	-19.8537	-16.6407	-15.3224
0.2500	3655	-1.6332	-1.2810	-1.2181
0.5000	12000	2.1831	2.1533	4.6990
0.7500	24000	3.5115	3.4526	6.7279
max	47000	12.2044	12.5481	15.5567

Table 6.2: Descriptive statistics of selected variables

values	windUComponent868	windUComponent2389	windUComponent2383	windUComponent2390
count	12270	12270	12270	12270
mean	0.6029	3.7481	1.5956	2.589226673
std	5.3603	5.2665	4.3240	5.553144729
min	-23.8693	-14.5255	-16.6661	-16.697449
25%	-1.9780	-0.7244	-1.3372	-1.37944875
50%	2.5419	5.5003	3.1143	4.2080
75%	4.0001	7.9950	4.5772	7.0551
max	14.0369	14.7241	14.2109	15.9120

From wind components we calculate, wind magnitude and direction as:

$$W = \sqrt{u^2 + v^2},$$
$$\theta = \arctan\left(\frac{u}{v}\right),$$

where W is the wind magnitude in m/s., u is the east-west wind component in m/s., v is the north-south wind component in m/s.

To convert it to degrees and adjust for meteorological conventions (clockwise from true north):

$$\theta_{\text{degrees}} = (90 - \frac{\theta}{\pi} \cdot 180) \mod 360,$$

where

θ is the wind direction in radians and θ_{degrees} is the wind direction in degrees. Table 6.3 below is variables used in this application.

Table 6.3: Names of Variables Used in Application

target variable	wind_generated				
input variables	wind_mag_867	wind_mag_2383	wind_mag_2383	wind_dir_2386	wind_dir_2391
	wind_mag_2385	wind_mag_2390	wind_mag_2390	wind_dir_869	wind_dir_2384
	wind_mag_2381	wind_mag_2380	wind_mag_2380	wind_dir_2387	wind_dir_2392
	wind_mag_2386	wind_mag_2391	wind_mag_2391	wind_dir_2382	wind_dir_866
	wind_mag_869	wind_mag_2384	wind_mag_2384	wind_dir_2388	wind_dir_2380
	wind_mag_2387	wind_mag_2392	wind_mag_2392	wind_dir_868	wind_dir_2381
	wind_mag_2382	wind_mag_866	wind_mag_866	wind_dir_2389	wind_dir_2381
	wind_mag_2388	wind_dir_867	wind_dir_867	wind_dir_2383	wind_mag_2389
	wind_mag_868	wind_dir_2385	wind_dir_2385	wind_dir_2390	

6.5 Application

Models used in this study to investigate feature importance algorithms are defined in the sequel.

Principal Component Analysis (PCA). Principal component analysis (PCA) is a statistical model that reduces the dimensionality of a dataset. It reconstructs new features that are linear combinations of the original features, such that these new features are uncorrelated and capture the most variance in the data. We first transform the dataset into 4 dimensional principal components in the application. Then we run each of three base models on this dimensionally reducted dataset: random forest, gradient boosted tree, and deep ensembles.

Singular Value Decomposition (SVD). Singular value decomposition is a method to solve linear systems. It calculates the pseudoinverse of a matrix. SVD decompose a matrix A into USV^T where U and V are orthogonal matrices with orthonormal eigenvectors chosen from AA^T and A^TA respectively. S is a diagonal matrix with r elements equal to the root of the positive eigenvalues of AA^T or A^TA. In machine learning usage, SVD is used for dimension reduction. In this regard, we map the dataset into four-dimensional space using the first 4 eigenvalues. Then, we input this into three baseline models.

Random Forest Model. Random forest regression is one of the most popular bagging methods. It re-samples data with replacement to create a bootstrapped series. For each of these series, it randomly drops some of the input features with uniform probability and fits a decision tree in order to create decorrelated trees. Then, the algorithm averages the results over a bootstrapped series. It has an embedded feature importance algorithm.

Gradient Boosting Trees. Gradient boosting regression is a boosting model that fits a decision tree to residuals from previous base learners and works sequentially to improve results from base learners. As an ensemble model, the gradient boosting tree model provides a feature importance algorithm.

6.6 Results of Feature Importance Algorithms

To measure the proposed algorithms' performance, we compare each model's weighted mape values, namely random forest, gradient boosting, and deep ensembles trained on complete dataset D, with models trained on dimension-reducted datasets based on feature importance and feature reduction algorithms.

For this, we report the weighted MAPE values of the models in Table 6.4.

Table 6.4: Weighted mape comparison of feature importance and reduction algorithms

	Complete Dataset	Variable Drop	Variable Permutation	PCA	SVD
Boosted Deep Ensemble	0.0685	0.0691	0.0659	0.0712	0.0701
Random Forest	0.0881	0.0819	0.0692	0.0919	0.1014
Gradient Boosting	0.1299	0.1225	0.1362	0.1814	0.1972

As we can see from the Table 6.4, variable drop feature importance has a slight adverse effect on the model accuracy of the boosted deep ensemble model. On the other hand, the variable permutation method significantly increased the accuracy of the boosted deep ensemble model and vice versa for the gradient-boosting tree model. Both variable drop and variable permutation methods have increased the accuracy of the random forest model. Models trained on PCA and SVD reducted datasets experienced a slight decrease for each model. The main reason for this might be that the information lost in the data reduction process might have harmed the performance of models.

Moreover, we plot correlations in Figure 6.1 , feature drop in Figure 6.2 , and permutation importance results in Figure 6.3, from the boosted deep ensembles model, enlightening nonlinear relationships.

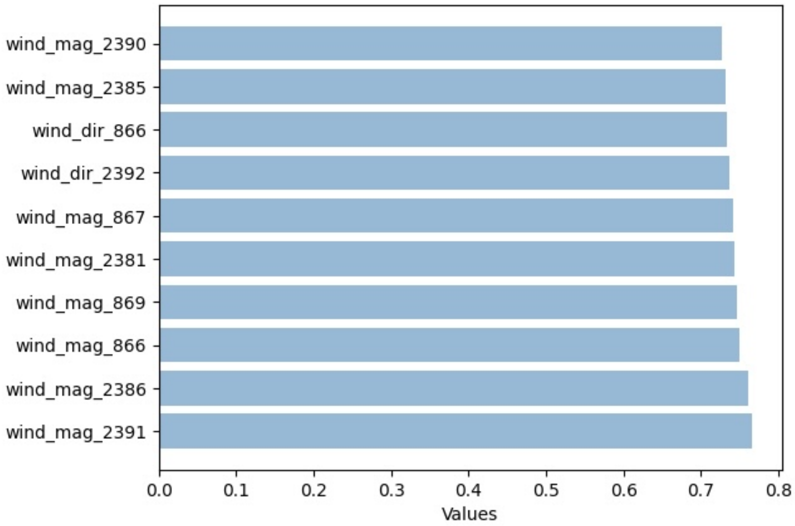

Figure 6.1: Input Variable Correlations

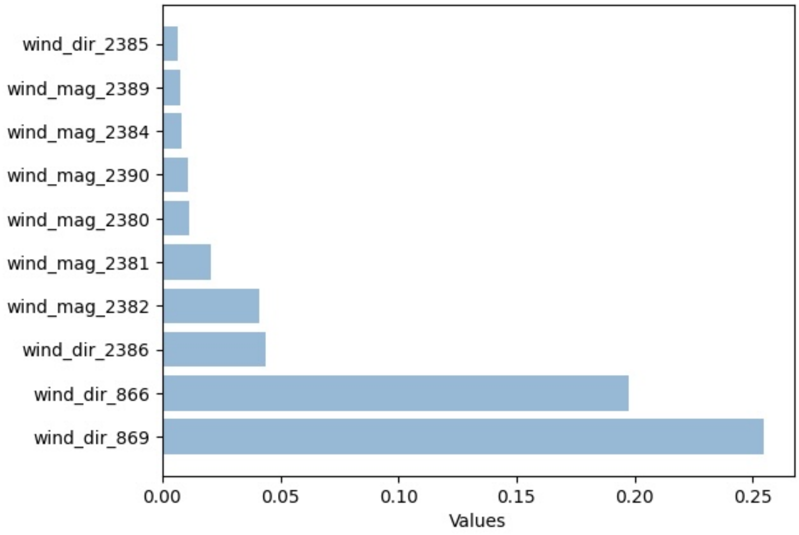

Figure 6.2: Variable Drop Feature Importances

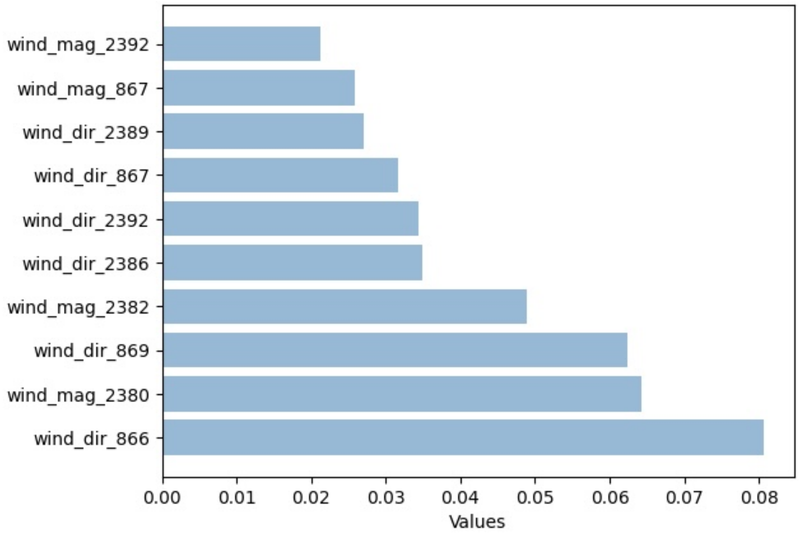

Figure 6.3: Variable Permutation Feature Importances

6.7 Conclusion and Further Study

In this chapter of the thesis study, we investigate and integrate feature importance algorithms to boosted deep ensembles models. First, we deeply investigate feature selection methodologies and its applications in the wind forecasting literature. Then, we integrated both variable drop and variable permutation feature importance algorithms into boosted deep ensemble model. We see that both variable drop and variable permutation works properly with boosted deep ensemble model. Moreover, we plot importance results. In order to have a more robust result, several other feature selection models and various error metrics could be included.

CHAPTER 7

CONCLUSION

This thesis study focuses on energy forecasting in the energy domain, a subclass of tabular learning tasks, with deep ensembles model. Energy forecasting is a crucial action for the energy sector, enabling market participants to manage risks and maintain operations properly. Contemporary research in energy forecasting predominantly employs machine learning and deep learning models.

We investigate and improve the deep ensembles model by proposing the boosted deep ensembles model, which leverages transfer learning to significantly reduce the required computation time, which we show theoretically and empirically. The proposed model generates prediction distributions equivalent to those of deep ensembles, which is statistically tested and is extensively evaluated using open-source energy-related datasets. We also note that when adequately trained, our boosted deep ensembles yield predictions conforming to a normal distribution when employing the mean squared error as the loss function. Conversely, the use of negative log-likelihood detrimentally impacts point forecasting accuracy. Furthermore, we find that superior forecasts are achieved by generating decorrelated predictions and excluding underperforming base learners.

We conduct an in-depth investigation as energy forecasting can be categorized as a subset of tabular learning. We explore state-of-the-art tabular learning models predominantly built upon transformer architectures and compare these models against XGBoost, CNN, and boosted deep ensemble models. Notably, the boosted deep ensembles model, the CNN model, and the XGBoost outperform transformer-based tabular learning models in load forecasting tasks. We also observe that notable performance gains are achieved through ensembling with XGBoost. We also delve into the embedding space using the t-SNE algorithm and perform a model ablation study. Our findings show the limitations of transformer-based models in effectively representing categorical and numerical features as embeddings, thereby leading to inferior performance.

A natural extension of transformer architecture-based tabular learning models is feature extraction from datasets. In order to contribute to explainable AI, we develop a feature selection framework for the boosted deep ensembles model. We integrate variable drop and variable permutation feature importance algorithms into the deep ensemble framework and rigorously assess the accuracy of these algorithms in comparison to benchmark models.

In this study, we improve deep ensembles models using transfer learning. Next, we benchmark SoTA tabular learning models and traditional machine learning models with deep ensembles, where we further investigate why transformer-based models are underperforming. Lastly, we contribute to explainable AI by developing a framework for feature selection with deep ensembles. We embed variable drop and variable premutation feature importance algorithm into boosted deep ensembles. For future study, a more comprehensive comparison of feature selection methodologies could be done by including more feature selection models and various error metrics to have more robust results.

REFERENCES

[1] H. Alfares and M. Nazeeruddin, Regression-based methodology for daily peak load forecasting, in *Proceedings of the 2nd International Conference on Operations and Quantitative Management*, volume 3, pp. 468–471, 1999.

[2] H. K. Alfares and M. Nazeeruddin, Electric load forecasting: literature survey and classification of methods, International journal of systems science, 33(1), pp. 23–34, 2002.

[3] S. Ö. Arik and T. Pfister, Tabnet: Attentive interpretable tabular learning, in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6679–6687, 2021.

[4] N. Bacanin, C. Stoean, M. Zivkovic, M. Rakic, R. Strulak-Wójcikiewicz, and R. Stoean, On the benefits of using metaheuristics in the hyperparameter tuning of deep learning models for energy load forecasting, Energies, 16(3), p. 1434, 2023.

[5] S. Badirli, X. Liu, Z. Xing, A. Bhowmik, K. Doan, and S. S. Keerthi, Gradient boosting neural networks: Grownet, arXiv preprint arXiv:2002.07971, 2020.

[6] E. Barakat, M. Qayyum, M. Hamed, and S. Al Rashed, Short-term peak demand forecasting in fast developing utility with inherit dynamic load characteristics. i. application of classical time-series methods. ii. improved modelling of system dynamic load characteristics, IEEE Transactions on Power Systems, 5(3), pp. 813–824, 1990.

[7] M. Beccali, M. Cellura, V. L. Brano, and A. Marvuglia, Short-term prediction of household electricity consumption: Assessing weather sensitivity in a mediterranean area, Renewable and Sustainable Energy Reviews, 12(8), pp. 2040–2065, 2008.

[8] A. C. Belkina, C. O. Ciccolella, R. Anno, R. Halpert, J. Spidlen, and J. E. Snyder-Cappione, Automated optimized parameters for t-distributed stochastic neighbor embedding improve visualization and analysis of large datasets, Nature communications, 10(1), p. 5415, 2019.

[9] S. Ben Taieb, R. Huser, R. Hyndman, and M. G. Genton, Probabilistic time series forecasting with boosted additive models: an application to smart meter data, Technical report, Monash University, Department of Econometrics and Business Statistics, 2015.

[10] F. M. Bianchi, E. De Santis, A. Rizzi, and A. Sadeghian, Short-term electric load forecasting using echo state networks and pca decomposition, IEEE Access, 3, pp. 1931–1943, 2015.

[11] C. S. Bojer and J. P. Meldgaard, Kaggle forecasting competitions: An overlooked learning opportunity, International Journal of Forecasting, 37(2), pp. 587–603, 2021.

[12] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, Deep neural networks and tabular data: A survey, IEEE Transactions on Neural Networks and Learning Systems, 2022.

[13] X. Bouthillier, P. Delaunay, M. Bronzi, A. Trofimov, B. Nichyporuk, J. Szeto, N. Mohammadi Sepahvand, E. Raff, K. Madan, V. Voleti, et al., Accounting for variance in machine learning benchmarks, Proceedings of Machine Learning and Systems, 3, pp. 747–769, 2021.

[14] L. Breiman, Bagging predictors, Machine learning, 24(2), pp. 123–140, 1996.

[15] L. Breiman, Random forests, Machine learning, 45(1), pp. 5–32, 2001.

[16] P. Bugata and P. Drotár, Weighted nearest neighbors feature selection, Knowledge-Based Systems, 163, pp. 749–761, 2019.

[17] M. Buhari, S. S. Adamu, et al., Short-term load forecasting using artificial neural network, in *International Multi Conference of Engineers and Computer Scientists Hongkong*, pp. 14–16, 2012.

[18] C. Carrillo, A. Obando Montaño, J. Cidrás, and E. Díaz-Dorado, Review of power curve modelling for wind turbines, Renewable and Sustainable Energy Reviews, 21, pp. 572–581, 2013.

[19] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, Ensemble selection from libraries of models, in *Proceedings of the twenty-first international conference on Machine learning*, p. 18, 2004.

[20] D. R. Chandra, M. S. Kumari, and M. Sydulu, A detailed literature review on wind forecasting, in *2013 International Conference on Power, Energy and Control (ICPEC)*, pp. 630–634, 2013.

[21] W.-Y. Chang et al., A literature review of wind forecasting methods, Journal of Power and Energy Engineering, 2(04), p. 161, 2014.

[22] N. Charlton and C. Singleton, A refined parametric model for short term load forecasting, International Journal of Forecasting, 30(2), pp. 364–368, 2014.

[23] S. G. Chavez, J. X. Bernat, and H. L. Coalla, Forecasting of energy production and consumption in asturias (northern spain), Energy, 24(3), pp. 183–198, 1999.

[24] B.-J. Chen, M.-W. Chang, et al., Load forecasting using support vector machines: A study on eunite competition 2001, IEEE transactions on power systems, 19(4), pp. 1821–1830, 2004.

[25] J.-F. Chen, W.-M. Wang, and C.-M. Huang, Analysis of an adaptive time-series autoregressive moving-average (arma) model for short-term load forecasting, Electric Power Systems Research, 34(3), pp. 187–196, 1995.

[26] T. Chen and C. Guestrin, Xgboost: A scalable tree boosting system, in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[27] M. Cho, J. Hwang, and C. Chen, Customer short term load forecasting by using arima transfer function model, in *Proceedings 1995 International Conference on Energy Management and Power Delivery EMPD'95*, volume 1, pp. 317–322, 1995.

[28] P. Chujai, N. Kerdprasop, and K. Kerdprasop, Time series analysis of household electric consumption with arima and arma models, in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, pp. 295–300, 2013.

[29] J. Connor, A robust neural network filter for electricity demand prediction, Journal of Forecasting, 15(6), pp. 437–458, 1996.

[30] C. Cortes and V. Vapnik, Support-vector networks, Machine learning, 20, pp. 273–297, 1995.

[31] R. R. de Aquino, A. A. Ferreira, M. M. Lira, O. N. Neto, P. S. Amorim, C. F. Diniz, and T. M. da Silveira, Short-term load forecasting for electrical regional of a distribution utility considering temperature, in *The 2011 International Joint Conference on Neural Networks*, pp. 2000–2004, 2011.

[32] M. De Felice and Xin Yao, Short-term load forecasting with neural network ensembles: A comparative study [application notes], IEEE Computational Intelligence Magazine, 6(3), pp. 47–56, 2011.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, 2009.

[34] N. Dewolf, B. D. Baets, and W. Waegeman, Valid prediction intervals for regression problems, Artificial Intelligence Review, 56(1), pp. 577–613, 2023.

[35] H. S. Dhiman and D. Deb, A review of wind speed and wind power forecasting techniques, arXiv preprint arXiv:2009.02279, 2020.

[36] F. X. Diebold and R. S. Mariano, Comparing predictive accuracy, Journal of Business & economic statistics, 20(1), pp. 134–144, 2002.

[37] T. Dillon, S. Sestito, and S. Leung, Short term load forecasting using an adaptive neural network, International Journal of Electrical Power & Energy Systems, 13(4), pp. 186–192, 1991.

[38] I. Dimoulkas, P. Mazidi, and L. Herre, Neural networks for gefcom2017 probabilistic load forecasting, International Journal of Forecasting, 35(4), pp. 1409–1423, 2019.

[39] O. Doelle, N. Klinkenberg, A. Amthor, and C. Ament, Probabilistic intraday pv power forecast using ensembles of deep gaussian mixture density networks, Energies, 16(2), p. 646, 2023.

[40] S. Dong, P. Wang, and K. Abbas, A survey on deep learning and its applications, Computer Science Review, 40, p. 100379, 2021.

[41] Dronio, nasa-hackathlon-solar-dataset, 2023, https://www.kaggle.com/datasets/dronio/SolarEnergy, Accessed = 2023-09-18.

[42] G. Dudek, Short-term load forecasting using random forests, in *Intelligent Systems' 2014*, pp. 821–828, Springer, 2015.

[43] G. Dudek, Multilayer perceptron for short-term load forecasting: from global to local approach, Neural Computing and Applications, pp. 1–13, 2019.

[44] A. El Desouky and M. El Kateb, Hybrid adaptive techniques for electric-load forecast using ann and arima, IEE Proceedings-Generation, Transmission and Distribution, 147(4), pp. 213–217, 2000.

[45] E.-S. M. El-Kenawy, S. Mirjalili, S. S. Ghoneim, M. M. Eid, M. El-Said, Z. S. Khan, and A. Ibrahim, Advanced ensemble model for solar radiation forecasting using sine cosine algorithm and newton's laws, IEEE Access, 9, pp. 115750–115765, 2021.

[46] ENTSO, Entso-e dataset, 2023, https://transparency.entsoe.eu/dashboard/show, Accessed = 2023-07-10.

[47] S. Fan and R. J. Hyndman, Short-term load forecasting based on a semi-parametric additive model, IEEE Transactions on Power Systems, 27(1), pp. 134–141, 2011.

[48] C. Feng, M. Cui, B.-M. Hodge, and J. Zhang, A data-driven multi-model methodology with deep feature selection for short-term wind forecasting, Applied Energy, 190, pp. 1245–1257, 2017.

[49] E. Fix and L. Hodges, Discriminatory analysis. nonparametric discrimination: Consistency properties, International Statistical Review/Revue Internationale de Statistique, 57(3), pp. 238–247, 1989.

[50] Y. Freund and L. Mason, The alternating decision tree learning algorithm, in *icml*, volume 99, pp. 124–133, 1999.

[51] Y. Gal and Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in *international conference on machine learning*, pp. 1050–1059, 2016.

[52] A. Garulli, S. Paoletti, and A. Vicino, Models and techniques for electric load forecasting in the presence of demand response, IEEE Transactions on Control Systems Technology, 23(3), pp. 1087–1097, 2014.

[53] P. Geurts, D. Ernst, and L. Wehenkel, Extremely randomized trees, Machine learning, 63(1), pp. 3–42, 2006.

[54] T. Gneiting and M. Katzfuss, Probabilistic forecasting, Annual Review of Statistics and Its Application, 1(1), pp. 125–151, 2014.

[55] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016, `http://www.deeplearningbook.org`.

[56] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, Revisiting deep learning models for tabular data, Advances in Neural Information Processing Systems, 34, pp. 18932–18943, 2021.

[57] Y. Goude, R. Nedellec, and N. Kong, Local short and middle term electricity load forecasting with semi-parametric additive models, IEEE transactions on smart grid, 5(1), pp. 440–446, 2013.

[58] L. Grinsztajn, E. Oyallon, and G. Varoquaux, Why do tree-based models still outperform deep learning on typical tabular data?, Advances in Neural Information Processing Systems, 35, pp. 507–520, 2022.

[59] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, On calibration of modern neural networks, arXiv preprint arXiv:1706.04599, 2017.

[60] F. K. Gustafsson, M. Danelljan, and T. B. Schön, Evaluating scalable bayesian deep learning methods for robust computer vision, CoRR, abs/1906.01620, 2019.

[61] I. Guyon and A. Elisseeff, An introduction to variable and feature selection, Journal of machine learning research, 3(Mar), pp. 1157–1182, 2003.

[62] S. Haben and G. Giasemidis, A hybrid model of kernel density estimation and quantile regression for gefcom2014 probabilistic load forecasting, International Journal of Forecasting, 32(3), pp. 1017–1022, 2016.

[63] M. T. Hagan and S. M. Behr, The time series approach to short term load forecasting, IEEE transactions on power systems, 2(3), pp. 785–791, 1987.

[64] L. K. Hansen and P. Salamon, Neural network ensembles, IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(10), pp. 993–1001, 1990.

[65] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, R. Tibshirani, and J. Friedman, Boosting and additive trees, The elements of statistical learning: data mining, inference, and prediction, pp. 337–387, 2009.

[66] H. Hazimeh, N. Ponomareva, P. Mol, Z. Tan, and R. Mazumder, The tree ensemble layer: Differentiability meets conditional computation, in *International Conference on Machine Learning*, pp. 4138–4148, 2020.

[67] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[68] D. Hendrycks and K. Gimpel, A baseline for detecting misclassified and out-of-distribution examples in neural networks, arXiv preprint arXiv:1610.02136, 2016.

[69] L. Hernández, C. Baladrón, J. M. Aguiar, B. Carro, A. Sánchez-Esguevillas, and J. Lloret, Artificial neural networks for short-term load forecasting in microgrids environment, Energy, 75, pp. 252–264, 2014.

[70] H. S. Hippert, C. E. Pedreira, and R. C. Souza, Neural networks for short-term load forecasting: A review and evaluation, IEEE Transactions on power systems, 16(1), pp. 44–55, 2001.

[71] T. K. Ho, The random subspace method for constructing decision forests, IEEE transactions on pattern analysis and machine intelligence, 20(8), pp. 832–844, 1998.

[72] A. Hoffman, Peak demand control in commercial buildings with target peak adjustment based on load forecasting, in *Proceedings of the 1998 IEEE International Conference on Control Applications (Cat. No. 98CH36104)*, volume 2, pp. 1292–1296, 1998.

[73] L. Hoffmann and C. Elster, Deep ensembles from a bayesian perspective, arXiv preprint arXiv:2105.13283, 2021.

[74] T. Hong and S. Fan, Probabilistic electric load forecasting: A tutorial review, International Journal of Forecasting, 32(3), pp. 914–938, 2016.

[75] T. Hong, P. Pinson, S. Fan, H. Zareipour, A. Troccoli, and R. J. Hyndman, Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond, International Journal of Forecasting, 32(3), pp. 896–913, 2016, ISSN 0169-2070.

[76] T. Hong, J. Xie, and J. Black, Global energy forecasting competition 2017: Hierarchical probabilistic load forecasting, International Journal of Forecasting, 35(4), pp. 1389–1399, 2019.

[77] C.-L. Hor, S. J. Watson, and S. Majithia, Daily load forecasting and maximum demand estimation using arima and garch, in *2006 International Conference on Probabilistic Methods Applied to Power Systems*, pp. 1–6, 2006.

[78] Y.-H. Hsiao, Household electricity demand forecast based on context information and user daily schedule analysis from meter data, IEEE Transactions on Industrial Informatics, 11(1), pp. 33–43, 2014.

[79] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, Snapshot ensembles: Train 1, get m for free, arXiv preprint arXiv:1704.00109, 2017.

[80] X. Huang, A. Khetan, M. Cvitkovic, and Z. Karnin, Tabtransformer: Tabular data modeling using contextual embeddings, arXiv preprint arXiv:2012.06678, 2020.

[81] N. Jhana, Hourly energy demand generation and weather, 2019, http://www.kaggle.com, Accessed = 2023-07-10.

[82] M. Joseph and H. Raj, Gandalf: Gated adaptive network for deep automated learning of features, 2023, https://arxiv.org/abs/2207.08548.

[83] A. Jović, K. Brkić, and N. Bogunović, A review of feature selection methods with applications, in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1200–1205, 2015.

[84] G. Juberias, R. Yunta, J. G. Moreno, and C. Mendivil, A new arima model for hourly load forecasting, in *1999 IEEE Transmission and Distribution Conference (Cat. No. 99CH36333)*, volume 1, pp. 314–319, IEEE, 1999.

[85] J. Jung and R. P. Broadwater, Current status and future advances for wind speed and power forecasting, Renewable and Sustainable Energy Reviews, 31, pp. 762–777, 2014, ISSN 1364-0321.

[86] A. Kadra, M. Lindauer, F. Hutter, and J. Grabocka, Well-tuned simple nets excel on tabular datasets, Advances in neural information processing systems, 34, pp. 23928–23941, 2021.

[87] Kaggle, Kaggle dataset, 2023, http://www.kaggle.com, Accessed = 2023-07-10.

[88] I. Kanda and J. Q. Veguillas, Data preprocessing and quantile regression for probabilistic load forecasting in the gefcom2017 final match, International Journal of Forecasting, 2019.

[89] A. Kendall and Y. Gal, What uncertainties do we need in bayesian deep learning for computer vision?, in *Advances in neural information processing systems*, pp. 5574–5584, 2017.

[90] M. Khalid and A. V. Savkin, A method for short-term wind power prediction with multiple observation points, IEEE Transactions on Power Systems, 27(2), pp. 579–586, 2012.

[91] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, Self-normalizing neural networks, Advances in neural information processing systems, 30, 2017.

[92] A. Krogh and J. Vedelsby, Neural network ensembles, cross validation, and active learning, in *Advances in neural information processing systems*, pp. 231–238, 1995.

[93] V. Kuleshov, N. Fenner, and S. Ermon, Accurate uncertainties for deep learning using calibrated regression, in *International conference on machine learning*, pp. 2796–2804, PMLR, 2018.

[94] E. Kyriakides and M. Polycarpou, Short term electric load forecasting: A tutorial, in *Trends in Neural Computation*, pp. 391–418, Springer, 2007.

[95] B. Lakshminarayanan, A. Pritzel, and C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, in *Advances in neural information processing systems*, pp. 6402–6413, 2017.

[96] Y. LeCun, Y. Bengio, et al., Convolutional networks for images, speech, and time series, The handbook of brain theory and neural networks, 3361(10), p. 1995, 1995.

[97] H. Leng, X. Li, J. Zhu, H. Tang, Z. Zhang, and N. Ghadimi, A new wind power prediction method based on ridgelet transforms, hybrid feature selection and closed-loop forecasting, Advanced Engineering Informatics, 36, pp. 20–30, 2018, ISSN 1474-0346.

[98] S. Li, E. J. Harner, and D. A. Adjeroh, Random knn feature selection-a fast and stable alternative to random forests, BMC bioinformatics, 12(1), pp. 1–11, 2011.

[99] S. Li, P. Wang, and L. Goel, Wind power forecasting using neural network ensembles with feature selection, IEEE Transactions on sustainable energy, 6(4), pp. 1447–1456, 2015.

[100] B. Liu, J. Nowotarski, T. Hong, and R. Weron, Probabilistic load forecasting via quantile regression averaging on sister forecasts, IEEE Transactions on Smart Grid, 8(2), pp. 730–737, 2015.

[101] G. Liu, H. Qin, Q. Shen, H. Lyv, Y. Qu, J. Fu, Y. Liu, and J. Zhou, Probabilistic spatiotemporal solar irradiation forecasting using deep ensembles convolutional shared weight long short-term memory network, Applied Energy, 300, p. 117379, 2021.

[102] G. Liu, Y. Wang, H. Qin, K. Shen, S. Liu, Q. Shen, Y. Qu, and J. Zhou, Probabilistic spatiotemporal forecasting of wind speed based on multi-network deep ensembles method, Renewable Energy, 209, pp. 231–247, 2023.

[103] Z. Liu, M. Hajiali, A. Torabi, B. Ahmadi, and R. Simoes, Novel forecasting model based on improved wavelet transform, informative feature selection, and hybrid support vector machine on wind power forecasting, Journal of Ambient Intelligence and Humanized Computing, 9(6), pp. 1919–1931, 2018.

[104] J. R. Lloyd, Gefcom2012 hierarchical load forecasting: Gradient boosting machines and gaussian processes, International Journal of Forecasting, 30(2), pp. 369–374, 2014.

[105] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, The m4 competition: Results, findings, conclusion and way forward, International Journal of Forecasting, 34(4), pp. 802–808, 2018.

[106] S. M. Malakouti, Estimating the output power and wind speed with ml methods: A case study in texas, Case Studies in Chemical and Environmental Engineering, 7, p. 100324, 2023.

[107] J. Maldonado-Correa, J. Solano, and M. Rojas-Moncayo, Wind power forecasting: A systematic literature review, Wind Engineering, 45(2), pp. 413–426, 2021.

[108] J. Manero, J. Béjar, and U. Cortés, Wind energy forecasting with neural networks: a literature review, Computación y Sistemas, 22(4), pp. 1085–1098, 2018.

[109] A. Marvuglia and A. Messineo, Using recurrent artificial neural networks to forecast household electricity consumption, Energy Procedia, 14, pp. 45–55, 2012.

[110] J. Massana, C. Pous, L. Burgas, J. Melendez, and J. Colomer, Short-term load forecasting in a non-residential building contrasting models and attributes, Energy and Buildings, 92, pp. 322–330, 2015.

[111] A. Massaro and G. Starace, Advanced and complex energy systems monitoring and control: A review on available technologies and their application criteria, Sensors, 22(13), p. 4929, 2022.

[112] P. Mastorocostas, J. Theocharis, and A. Bakirtzis, Fuzzy modeling for short term load forecasting using the orthogonal least squares method, IEEE Transactions on Power Systems, 14(1), pp. 29–36, 1999.

[113] L. Medsker and L. C. Jain, *Recurrent neural networks: design and applications*, CRC press, 1999.

[114] A. Misiorek, S. Trueck, and R. Weron, Point and interval forecasting of spot electricity prices: Linear vs. non-linear time series models, Studies in Nonlinear Dynamics & Econometrics, 10(3), 2006.

[115] I. Moghram and S. Rahman, Analysis and evaluation of five short-term load forecasting techniques, IEEE Transactions on power systems, 4(4), pp. 1484–1491, 1989.

[116] M. Mohandes, Support vector machines for short-term electrical load forecasting, International Journal of Energy Research, 26(4), pp. 335–345, 2002.

[117] H. Mori and M. Ohmi, Probabilistic short-term load forecasting with gaussian processes, in *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, pp. 6–pp, 2005.

[118] H. Mori and A. Takahashi, Hybrid intelligent method of relevant vector machine and regression tree for probabilistic load forecasting, in *2011 2nd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies*, pp. 1–8, 2011.

[119] F. Murtagh, Multilayer perceptrons for classification and regression, Neurocomputing, 2(5-6), pp. 183–197, 1991.

[120] G. R. Newsham and B. J. Birt, Building-level occupancy data to improve arima-based electricity use forecasts, in *Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building*, pp. 13–18, 2010.

[121] open weather, weather dataset, 2023, https://openweathermap.org/api, Accessed = 2023-07-10.

[122] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift, in *Advances in Neural Information Processing Systems*, pp. 13991–14002, 2019.

[123] A. D. Papalexopoulos and T. C. Hesterberg, A regression-based approach to short-term system load forecasting, IEEE Transactions on Power Systems, 5(4), pp. 1535–1547, 1990.

[124] S. S. Pappas, L. Ekonomou, P. Karampelas, D. Karamousantas, S. Katsikas, G. Chatzarakis, and P. Skafidas, Electricity demand load forecasting of the hellenic power system using an arma model, Electric Power Systems Research, 80(3), pp. 256–264, 2010.

[125] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, the Journal of machine Learning research, 12, pp. 2825–2830, 2011.

[126] M. P. Perrone and L. N. Cooper, When networks disagree: Ensemble methods for hybrid neural networks, in *How We Learn; How We Remember: Toward An Understanding Of Brain And Neural Systems: Selected Papers of Leon N Cooper*, pp. 342–358, World Scientific, 1995.

[127] R. Platon, V. R. Dehkordi, and J. Martel, Hourly prediction of a building's electricity consumption using case-based reasoning, artificial neural networks and principal component analysis, Energy and Buildings, 92, pp. 10–18, 2015.

[128] S. Popov, S. Morozov, and A. Babenko, Neural oblivious decision ensembles for deep learning on tabular data, arXiv preprint arXiv:1909.06312, 2019.

[129] pravdomirdobrev, texas-wind-turbine-dataset-simulated, 2023, https://www.kaggle.com/datasets/pravdomirdobrev/texas-wind-turbine-dataset-simulated.

[130] B. Qin, X. Huang, X. Wang, and L. Guo, Ultra-short-term wind power prediction based on double decomposition and lssvm, Transactions of the Institute of Measurement and Control, p. 01423312231153258, 2023.

[131] R. Rahaman and A. H. Thiery, Uncertainty quantification and deep ensembles, arXiv preprint arXiv:2007.08792, 2020.

[132] D. K. Ranaweera, G. G. Karady, and R. G. Farmer, Effect of probabilistic inputs on neural network-based electric load forecasting, IEEE Transactions on Neural networks, 7(6), pp. 1528–1532, 1996.

[133] S. Ryu and Y. Yu, Quantile-mixer: A novel deep learning approach for probabilistic short-term load forecasting, IEEE Transactions on Smart Grid, 2023.

[134] R. E. Schapire, The strength of weak learnability, Machine learning, 5(2), pp. 197–227, 1990.

[135] J. Schmidhuber, Deep learning in neural networks: An overview, Neural networks, 61, pp. 85–117, 2015.

[136] H. L. Shang, Functional time series approach for forecasting very short-term electricity demand, Journal of Applied Statistics, 40(1), pp. 152–168, 2013.

[137] H. Shao, X. Deng, and Y. Jiang, A novel deep learning approach for short-term wind power forecasting based on infinite feature selection and recurrent neural network, Journal of Renewable and Sustainable Energy, 10(4), p. 043303, 2018.

[138] R. Shwartz-Ziv and A. Armon, Tabular data: Deep learning is not all you need, Information Fusion, 81, pp. 84–90, 2022.

[139] A. K. Singh, S. Khatoon, M. Muazzam, D. Chaturvedi, et al., Load forecasting techniques and methodologies: A review, in *2012 2nd International Conference on Power, Control and Embedded Systems*, pp. 1–10, 2012.

[140] R. Singhal, N. K. Choudhary, and N. Singh, Short-term load forecasting using hybrid arima and artificial neural network model, in *Advances in VLSI, Communication, and Signal Processing*, pp. 935–947, Springer, 2020.

[141] D. Smilkov, N. Thorat, C. Nicholson, E. Reif, F. B. Viégas, and M. Wattenberg, Embedding projector: Interactive visualization and interpretation of embeddings, arXiv preprint arXiv:1611.05469, 2016.

[142] S. Smyl and N. G. Hua, Machine learning methods for gefcom2017 probabilistic load forecasting, International Journal of Forecasting, 35(4), pp. 1424–1431, 2019.

[143] M. Sunder, R. Abishek, M. Maiti, K. Bingi, P. A. M. Devan, and M. Assaad, Forecasting of wind turbines generated power with missing input variables, in *2022 International Conference on Future Trends in Smart Communities (ICFTSC)*, pp. 98–103, 2022.

[144] J. W. Taylor, Short-term electricity demand forecasting using double seasonal exponential smoothing, Journal of the Operational Research Society, 54(8), pp. 799–805, 2003.

[145] J. W. Taylor, Triple seasonal methods for short-term electricity demand forecasting, European Journal of Operational Research, 204(1), pp. 139–152, 2010.

[146] J. W. Taylor, Short-term load forecasting with exponentially weighted methods, IEEE Transactions on Power Systems, 27(1), pp. 458–464, 2011.

[147] J. W. Taylor and R. Buizza, Neural network load forecasting with weather ensemble predictions, IEEE Transactions on Power systems, 17(3), pp. 626–632, 2002.

[148] S. Theodoridis and K. Koutroumbas, *Pattern recognition*, Elsevier, 2006.

[149] C. Tian, J. Ma, C. Zhang, and P. Zhan, A deep neural network model for short-term load forecast based on long short-term memory network and convolutional neural network, Energies, 11(12), p. 3493, Dec 2018, ISSN 1996-1073.

[150] Z. Tian, A state-of-the-art review on wind power deterministic prediction, Wind Engineering, 45(5), pp. 1374–1392, 2021.

[151] S. R. Twanabasu and B. A. Bremdal, Load forecasting in a smart grid oriented building, in *22nd International Conference and Exhibition on Electricity Distribution (CIRED 2013)*, pp. 1–4, 2013.

[152] L. Van der Maaten and G. Hinton, Visualizing data using t-sne., Journal of machine learning research, 9(11), 2008.

[153] S. Varadan and E. B. Makram, Harmonic load identification and determination of load composition using a least squares method, Electric power systems research, 37(3), pp. 203–208, 1996.

[154] P. Vrablecová, A. B. Ezzeddine, V. Rozinajová, S. Šárik, and A. K. Sangaiah, Smart grid load forecasting using online support vector regression, Computers & Electrical Engineering, 65, pp. 102–117, 2018.

[155] X.-P. Wang and M. Meng, Forecasting electricity demand using grey-markov model, in *2008 International Conference on Machine Learning and Cybernetics*, volume 3, pp. 1244–1248, IEEE, 2008.

[156] J. Xie and T. Hong, Gefcom2014 probabilistic electric load forecasting: An integrated solution with forecast combination and residual simulation, International Journal of Forecasting, 32(3), pp. 1012–1016, 2016.

[157] T. Xiong, Y. Bao, and Z. Hu, Interval forecasting of electricity demand: A novel bivariate emd-based support vector regression modeling framework, International Journal of Electrical Power & Energy Systems, 63, pp. 353–362, 2014.

[158] M. A. Yahya, S. P. Hadi, and L. M. Putranto, Short-term electric load forecasting using recurrent neural network (study case of load forecasting in central java and special region of yogyakarta), in *2018 4th International Conference on Science and Technology (ICST)*, pp. 1–6, 2018.

[159] J. Zack, Overview of wind energy generation forecasting. true wind solutions. llc and aws scientific, Inc.(December 17, 2003), 2003.

[160] Y. Zhang, J. Wang, and X. Wang, Review on probabilistic forecasting of wind power generation, Renewable and Sustainable Energy Reviews, 32, pp. 255–270, 2014.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Enginar, Onur
Nationality: Turkish (TC)

EDUCATION

Degree	Institution	Year of Graduation
M.S.	Hacettepe University	2018
B.S.	Hacettepe University	2013
High School	Kurtuluş High School	2008

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2016-Present	Hacettepe University	Research Assistant
2020-Present	Presify AI	Co-Founder

PUBLICATIONS

International Conference Publications

1. Enginar, O., Atici, K. B. (2022). Optimal forecast error as an unbiased estimator of abnormal return: A proposition. Journal of Forecasting, 41(1), 158-166.
2. Büyükkara, G., Enginar, O., Temiz, H. (2020). Volatility Spillovers Between Oil and Stock Market Returns in G7 Countries: A VAR-DCC-GARCH Model.

Regulations in the Energy Industry: Financial, Economic and Legal Implications, 169-186.

3. Enginar, O., Karan, M. B., Büyükkara, G. (2018). Performances of emerging stock exchanges during the Fed's tapering announcements. Global Approaches in Financial Economics, Banking, and Finance, 415-443.
4. Enginar, O., Karan, M. B., (2018). Using Neural Network in Event Study . Multinational Finance Society, Budapest, Hungary.