<u>ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL</u>

A NOVEL ARTIFICIAL INTELLIGENCE BASED ENERGY MANAGEMENT SYSTEM FOR MICROGRIDS

Ph.D. THESIS

Necati AKSOY

Department of Electrical Engineering

Electrical Engineering Programme

JUNE 2023

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

A NOVEL ARTIFICIAL INTELLIGENCE BASED ENERGY MANAGEMENT SYSTEM FOR MICROGRIDS

Ph.D. THESIS

Necati AKSOY
(504182007)

Department of Electrical Engineering

Electrical Engineering Programme

Thesis Advisor: Prof. Dr. V. M. İstemihan GENÇ

JUNE 2023

ISTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

MİKRO ŞEBEKELER İÇİN YAPAY ZEKA TEMELLİ YENİ BİR ENERJİ YÖNETİM SİSTEMİ

DOKTORA TEZİ

Necati AKSOY
(504182007)

Elektrik Mühendisliği Anabilim Dalı

Elektrik Mühendisliği Programı

Tez Danışmanı: Prof. Dr. V. M. İstemihan GENÇ

HAZİRAN 2023

Necati AKSOY, a Ph.D. student of İTU Graduate School student ID 504182007, successfully defended the thesis/dissertation entitled “A NOVEL ARTIFICIAL INTELLIGENCE BASED ENERGY MANAGEMENT SYSTEM FOR MICROGRIDS”, which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor : **Prof. Dr. V. M. İstemihan GENÇ** ....
İstanbul Technical University

Jury Members : **Assoc. Prof. Dr. Ramazan ÇAĞLAR** ....
Istanbul Technical University

Assoc. Prof. Dr. Ercan İZGİ ....
Yıldız Technical University

Prof. Dr. Ahmet CANSIZ ....
Istanbul Technical University

Asst. Prof. Dr. Fatih KÜÇÜKTEZCAN ....
Bahçeşehir University

Date of Submission : 18 May 2023
Date of Defense : 19 June 2023

To my lovely wife and family,

FOREWORD

First of all, I would like to express my deep gratitude to my advisor, Prof. Dr. Veysel Murat İstemihan GENÇ, who provided me with the opportunity to study with him throughout the entire doctoral process. His vast knowledge and exemplary qualities as a scientist have been instrumental in guiding and enlightening. I have learned valuable knowledge on science and research from this esteemed professor.

Moreover, I would like to extend my profound gratitude to the jury members, Assoc. Prof. Dr. Ramazan ÇAĞLAR and Assoc. Prof. Dr. Ercan İZGİ, for their sincere and supportive attitudes and invaluable suggestions during the doctoral process.

I would like to express my thanks to Istanbul Technical University and the valuable people in Departement of Electrical Engineering, who provided the opportunity for me to undertake this doctoral study. The department's deep-rooted structure and science-focused atmosphere have been very encouraging in completing this work.

In addition, I am grateful to Dr. Mehmet Murat İSPİRLİ, whom I met during the doctoral process and whose friendship and supportive personality I cherish. Similarly, I thank my elder sister, Dr. Gökçen TURAN, for her advice, suggestions, and support.

Finally, I extend my heartfelt gratitude to my beloved wife, Elif ÖZTÜRK AKSOY, who has always been my supporter, my light, my energy, and my life. I am deeply grateful for her presence. Additionally, I thank my mother, Ayşe AKSOY, my father, Süleyman AKSOY, and my brother, Mehmet Ali AKSOY, for always being by my side and encouraging me.

May 2023

Necati AKSOY
(Lecturer)

TABLE OF CONTENTS

Page

ABBREVIATIONS

AI	: Artificial Intelligence
AIEMS	: Artificial Intelligence-based Energy Management System
ANN	: Artificial Neural Network
Bi-LSTM	: Bidirectional Long Short-Term Memory
CNN	: Convolutional Neural Network
DERs	: Distributed Energy Resources
DDPG	: Deep Deterministic Policy Gradient
DL	: Deep Learning
DNN	: Deep Neural Network
DRL	: Deep Reinforcement Learning
DQN	: Deep Q-Network
DDQN	: Double Deep Q-Network
EFB	: Exclusive Feature Bundling
EL	: Ensemble Learning
EMS	: Energy Management System
ESS	: Energy Storage System
GBM	: Gradient Boosting Machine
GPI	: Generalized Policy Iteration
GRU	: Gated Recurrent Units
GOSS	: Gradient-based one-side sampling
GUI	: Graphical User Interface
LSTM	: Long Short-Term Memory
MAE	: Mean Absolute Error
MDP	: Markov Decision Process
ML	: Machine Learning
MSE	: Mean Squared Error
PPO	: Proximal Policy Optimization
QL	: Q Learning
RF	: Random Forrest
RL	: Reinforcement Learning

RMSE	**:** Root Mean Squared Error
RF	**:** Recurrent Neural Network
SARSA	**:** State-action-reward-state-action
SGD	**:** Stochastic Gradient Decent
SVM	**:** Support Vector Machine
SVR	**:** Support Vector Regressor
TD	**:** Temporal-Difference

SYMBOLS

$\mathbf{C_t}$	: Long-term memory line
$\mathbf{h_t}$	: Value line between the cells
$\mathbf{G_t}$	: Return at time t
$\mathbf{P_t}$	: Power generation value at time t
$\mathbf{Pr_t}$	: Price value at time t
$\mathbf{R^2}$	: R-squared value
t	: Time
$\mathbf{L_t}$	: Load demand level at time t
w	: Weights of neural networks
$\mathbf{SoC_t}$	: State of charge at time t.
$\boldsymbol{\Theta_t}$	: Parameters of Q network and target Q network
α	: Learning rate.
ε	: Action selection parameter
γ	: Discount factor
π	: Policy function

LIST OF TABLES

Page

LIST OF FIGURES

Page

A NOVEL ARTIFICIAL INTELLIGENCE BASED ENERGY MANAGEMENT SYSTEM FOR MICROGRIDS

SUMMARY

In many countries, including our own, large amounts of electrical power are generated where the energy source is located, while it is consumed in areas with large industries and populations. This distance between energy generation and consumption leads to the transmission of energy, which results in the waste of energy as heat and increases energy costs. Microgrids have emerged as a solution to energy use by applying the principle of energy generation and consumption at the same place. Microgrids are small-scale electrical grids that can use distributed energy resources in conjunction with conventional grids. They can combine solar panels or plants, wind turbines, energy storage systems, generators, and the utility grid. This reduces energy loss during transmission, improves energy efficiency, and allows energy to be used efficiently. In addition, microgrids that operate in small settlements such as university campuses, military facilities, towns, or neighborhoods can work in “island mode” without a connection to the utility grid when needed. Many microgrids are currently operated using classical control methods and operate in certain size that has only been determined using optimization methods. This limits the efficiency that can be achieved during the operation of the microgrid and makes it difficult to follow new trends in energy storage technologies. The crux and significance of this thesis revolves around the notion that contemporary energy storage technologies can be utilized efficiently within the system, and that the existing artificial intelligence technology can serve as the foundation of the microgrid energy management system. The energy management system designed in this structure reduces energy waste, lowers costs, improves efficiency, and improves grid stability, while also producing effective solutions for energy demand by controlling the use of various sources together. Moreover, this energy management system contributes to reducing carbon emissions while allowing for the easy adaptation of new technologies. In light of all these advantages, this thesis presents an artificial intelligence-based energy management system design for microgrids.

To further explain the concept of artificial intelligence, it encompasses machine learning algorithms as a subset, while machine learning includes deep learning algorithms and concepts. In this thesis, microgrid applications of various sizes and properties are examined, and a microgrid simulation model was created at commonly used sizes. This simulation model assumed a microgrid applied to a university campus, with a solar power plant and wind turbines serving as renewable energy sources. The energy management system being designed predicts the power that these sources will generate, using the up-to-date prediction algorithms within artificial intelligence. When designing, the focus is initially on predicting the power that solar and wind turbines will generate, using five years of meteorological data collected at five-minute intervals. The meteorological dataset, consisting of nine different data types, has undergone a series of data pre-processing. Missing data is filled in accordance with the characteristics of the dataset, and outliers are removed. The characteristics of this

dataset were analyzed with different graphs and their suitability for training was examined. The labeled data consisting of the generation values at the same region and at the same time/minute intervals were added to the meteorological data set that was deemed suitable for training.

Seven prediction models were developed using four prevalent machine learning methods and three novel algorithms based on the gradient boosting machine to predict the power generated by the solar power plant. These prediction models were trained separately using the training dataset made suitable for training. The results obtained from these seven prediction models were presented in both graphical and tabular formats. In addition to comparing which algorithm gave how successful results for this study, the computation costs were also compared. The designed energy management system must also predict the power generated from wind turbines. In this regard, prediction models were created using three different machine learning algorithms, and the results were obtained. These prediction models were compared using various performance metrics. This study conducted within this thesis, which achieved successful results, offers new approaches and unique results to the literature on the prediction of the power generation of renewable energy sources.

An artificial intelligence-based energy management system should provide not only energy efficiency but also low energy costs and profitability for the user. The widespread use of dynamic electricity pricing should also be considered, which is determined based on the relationship between countrywide generation and consumption level. In this thesis, it is assumed that the microgrid simulation model developed is located in a country where dynamic pricing is applied. A five-year dataset was created from actual dynamic pricing data obtained from open-source platforms and analyzed. The dataset was examined, preprocessed, and made ready for the training of prediction models. Four deep learning algorithms with memory cell structures were selected for this study. Using these algorithms and the training dataset, price prediction models were developed, and the results were obtained. The learning performances, error values, and accuracies of the models were presented comparatively. These innovative prediction models were integrated into the designed energy management system.

Knowing the power demand from a microgrid makes operational decisions more appropriate and robust. The load demand at which time of the day is an important parameter. Knowing the load demand in advance affects decisions regarding resource utilization. Considering this fact, the energy management system designed should also be able to predict load demand. To this end, load demand prediction models were developed using four deep learning methods with memory cell structures similar to price prediction. Actual load values obtained from open sources were scaled according to the simulation model of the microgrid created. Deep learning models were trained using the five-year load dataset, and the results were obtained. The results were presented comparatively using many performance metrics. As a result of this study, successful prediction models were developed and integrated into the designed energy management system.

An artificial intelligence-based energy management system uses many prediction models described above. The theoretical and mathematical foundations of all machine learning and deep learning methods used are provided in the second chapter of this thesis. The energy management system described requires an additional controller to manage the microgrid in addition to human management. In this context, this thesis proposes another artificial intelligence-based controller.

Data-driven control methods that have replaced classical control methods are popular topics nowadays. This thesis focuses on machine learning-based control methods of this type. In this context, reinforcement learning, which is one of the three main branches of machine learning, is investigated and its foundations are given. Reinforcement learning is the general name for methods based on the principle of controlling the system without the need for a mathematical model of the system. It is possible to separate this concept into methods based on table creation and methods using deep neural networks. In this thesis, controller agents using both types of methods are created. The agent, which will learn to control the system in reinforcement learning, needs to optimize itself. This optimization process is done through trial and error. For the agent to be able to take the best version through these trials, the system it will control, which is a microgrid environment model in this thesis, must have specific characteristics. Five different control agents were designed specifically for the energy management system, three of which were temporal-difference-based and two were deep reinforcement learning-based. Three environment models designed specifically for the microgrid are proposed in this thesis to enable these agents to train themselves. These environment models with unique reward strategies present a new approach to the literature. These environment models that use renewable energy sources, load demand, and dynamic prices for the training of agents have shown quite successful results in terms of energy management. The trained reinforcement learning agents have learned to manage the microgrid and offer considerable profitability to the user.

The energy management system whose design steps are explained in this thesis uses many different artificial intelligence algorithms. These artificial intelligence models created, trained, and successful results achieved have been consolidated under a single graphical interface in this thesis. A unique graphical interface has been designed, and all prediction models and control agents have been integrated into this design. This interface design, which consists of seven pages in total, offers many variables and control actions related to the microgrid to the user. The user can see the powers that will be generated for the future, load demand, and the price. In addition, the user can apply many control actions to the microgrid through this interface. The user, who can also see many real-time parameters, can analyze the performance of prediction models and control agents through relevant pages.

In conclusion, this thesis proposes an artificial intelligence-based energy management system that contains many current and innovative algorithms for microgrids and uses them uniquely. Artificial intelligence-based prediction models determine the decisions that an artificial intelligence-based control agent will make. This agent, which learns to select the correct control actions for the microgrid, presents the determined control action to the user through the designed interface. Additionally, the originally designed energy management system interface allows the user to see many parameters related to the microgrid in advance. This thesis proposes an energy management system that contributes to the literature with its original approach and can be used in real-world applications.

MİKRO ŞEBEKELER İÇİN YAPAY ZEKA TEMELLİ YENİ BİR ENERJİ YÖNETİM SİSTEMİ

ÖZET

Tüm enerji kaynakları içerisinde elektrik enerjisi, kullanımı en yaygın enerji türü olarak karşımıza çıkar. Bu bağlamda, elektrik enerjisinin verimli kullanımı günümüz toplumunda önemli bir tartışma ve araştırma konusudur. Enerji tüketimi artmaya devam ettikçe, enerji verimli teknolojilere ve uygulamalara olan ihtiyaç giderek daha vazgeçilmez hale gelmektedir. Enerji verimliliğinin en önemli faydalarından biri, karbon emisyonlarını azaltma potansiyelidir. Enerjiyi daha verimli kullanmak, karbon ayak izimizi azaltır ve iklim değişikliğinin olumsuz etkilerini hafifletmeye yardımcı olur. Çevresel faydalarına ek olarak, enerji verimliliği aynı zamanda maliyet tasarrufuna da yardımcı olabilir. Ayrıca enerji verimliliği, fiyat dalgalanmalarını azaltabilir ve fosil kaynaklara bağımlılığı dolayısıyla bu kaynakların neden olduğu arz kesintilerini azaltarak enerji güvenliğini de artırabilir. Enerji sektörü ve ilgili endüstrilerde yeni iş fırsatları yaratarak, enerji verimliliği ekonomik büyümeyi artırmaya ve yaşam kalitesini iyileştirmeye yardımcı olabilir. Sonuç olarak, elektrik enerjisinin verimli kullanımı günümüz toplumunda çok önemli bir odak alanıdır. Bu alanda öne çıkan en temel çözümler halihazırdaki enerji iletim dağıtım şebekelerini yeni nesil teknolojiler ile yenilemek ve geliştirmektir.

Şehirler veya ülkeler boyutunda akıllı şebekeler içerdiği bilişim ve iletişim teknolojileri ile yenilikçi çözüm olarak karşımıza çıkarken, küçük bir yerleşim yeri veya kampüs boyutunda mikro şebekeler enerji verimliliğine çözüm olarak sunulur. Bu iki şebeke türüne ek olarak bir hane boyutunda yenilebilir enerji kaynakları ve dağıtım şebekesini birleştiren nano şebekeler öne çıkmaktadır. Bu tez çalışması bu çözümlerden mikro şebekelere odaklanır.

Birçok ülkede ve ülkemizde elektrik enerjisi üretim yöntemine bağlı olarak kaynağın bulunduğu yerlerde büyük miktarlarda üretilirken büyük sanayi ve nüfusa sahip yerlerde tüketilir. Enerjinin üretim ve tüketimi arasındaki bu mesafe büyük iletim şebekelerine olan ihtiyacı ortaya çıkarır. Bu uzun iletim ve dağıtım hatları enerjinin ısı olarak israfına yol açar ve enerji maliyetini artırır. Mikro şebekeler enerjinin üretim ve tüketimin aynı yerde olması prensibi ile bu soruna çözüm olarak ortaya çıkmıştır. Mikro şebekeler dağıtık enerji kaynaklarını konvansiyonel şebeke ile birlikte kullanabilen küçük ölçekli elektrik şebekeleridir. Mikro şebekeler güneş panellerini veya santralini, rüzgâr türbinlerini, enerji depolama sistemlerini, jeneratörleri ve ana şebekeyi birlikte kullanabilir. Bu özellikleri sayesinde enerjinin taşınmasındaki kayıpları azaltıp enerjinin verimli kullanılmasına olanak sağlar. Ek olarak, üniversite kampüsleri, askeri tesisler, kasabalar veya mahalleler gibi küçük yerleşim yerlerinde çalışan mikro şebekeler ihtiyaç duyulduğunda ana şebeke bağlantısı olmadan 'ada modunda' çalışabilir. Bu özelliği acil durumlarda avantaj sağlarken ayrıca ana şebekedeki büyük kesintilerden etkilenmemesini de beraberinde getirir. Ayrıca mikro şebekeler, arz ve talebi dengelemek ve enerji piyasalarına katılmak için elektrik alıp satarak daha ana şebekeyle uyumlu çalışabilir.

Mikro şebekeler, artırılmış enerji verimliliği, azaltılmış sera gazı emisyonları, düşük enerji maliyetleri, iyileştirilmiş şebeke güvenilirliği ve esnekliği ek olarak artırılmış enerji güvenliği dahil olmak üzere çeşitli avantajlar sunar. Sürdürülebilir ve merkezi olmayan enerji sistemlerini teşvik etmenin bir yolu olarak giderek daha popüler hale gelmektedir. Günümüzde birçok mikro şebeke, klasik kontrol yöntemleriyle sürdürülmekte ve yalnızca matematiksel analiz yöntemleriyle boyutlandırılmış şekilde çalışmaktadır. Bu durum hem mikro şebekenin işletilmesi sırasında elde edilebilecek verimliliği kısıtlamakta hem de enerji depolama teknolojilerindeki yeni trendleri takip etmeyi zorlaştırmaktadır. Mevcut enerji depolama teknolojilerinin sistemde etkin bir şekilde kullanılabileceği ve günümüz yapay zekâ teknolojisinin mikro şebeke enerji yönetim sisteminin temelini oluşturabileceği düşüncesi tezin özünü ve önemini oluşturmaktadır.

Yapay zekâ tabanlı bir enerji yönetim sistemi, mikro şebekelerin işletimini ve yönetimini optimize etmede çok önemli bir rol oynayabilir. Bu tarz enerji yönetim sisteminin mikro şebekeler için önemli olmasının bazı temel nedenleri şunlardır: Gelişmiş enerji verimliliği: yapay zekâ temelli enerji yönetim sistemi, doğru tahminler yapmak ve mikro şebeke bileşenlerinin enerji kullanımını optimize etmek için hava durumu tahminleri, enerji talebi ve enerji depolama seviyeleri gibi çeşitli kaynaklardan gerçek zamanlı verileri analiz edebilir. Bu, enerji israfını azaltmaya, verimliliği artırmaya ve maliyetleri düşürmeye yardımcı olabilir. Geliştirilmiş şebeke kararlılığı: bu tarz enerji yönetim sistemi, mikro şebeke bileşenlerinin güç çıkışını gerçek zamanlı olarak izleyip kontrol ederek bunların güvenli ve istikrarlı sınırlar içinde çalışmasını sağlayabilir. Bu özellik, aşırı yüklenmeyi önlemeye ve elektrik kesintisi veya voltaj düşmesi riskini azaltmaya yardımcı olabilir. Artırılmış esneklik: hedeflenen enerji yönetim sistemi, farklı bileşenlerin güç çıkışını gerçek zamanlı olarak ayarlayarak değişen enerji taleplerine ve tedarik modellerine uyum sağlayabilir. Bu, arz ve talebin dengelenmesine yardımcı olabilir ve yedek jeneratörlere veya enerji depolama sistemlerine olan ihtiyacı azaltabilir. Daha iyi karar verme: bu enerji yönetim sistemi, enerji kullanım modelleri hakkında bilgi sağlayabilir, enerji tasarrufu fırsatlarını belirleyebilir ve optimum bakım programları önerebilir. Bu durum, mikro şebeke operatörlerinin enerji yönetimi ve bakımı hakkında bilinçli kararlar almasına yardımcı olabilir. Sürdürülebilir enerji yönetimi: yapay zekâ temelli enerji yönetim sistemi, güç çıkışını ve depolamayı optimize ederek mikro şebekelerin güneş ve rüzgâr enerjisi gibi yenilenebilir enerji kaynaklarını entegre etmesine yardımcı olabilir. Böylelikle, karbon gazı emisyonlarının azaltılmasına ve sürdürülebilir enerji yönetiminin desteklenmesine yardımcı olabilir. Tüm bu avantajlar göz önünde bulundurularak bu tez çalışmasında mikro şebekeler için yapay zekâ temelli bir enerji yönetim sistemi tasarımı sunulmuştur.

Yapay zekâ kavramını detaylandıracak olursak, yapay zekâ üst başlığı alt başlık olan makine öğrenme algoritmalarını kapsarken, makine öğrenmesi başlığı da derin öğrenme algoritmalarını ve kavramlarını barındırmaktadır. Birçok mikro şebeke bünyesinde yenilebilir enerji kaynakları bulundurur. Bu tez çalışmasında, çeşitli boyutlarda ve özelliklerde mikro şebeke uygulamaları incelenmiş ve yaygın kullanım boyutlarında bir mikro şebeke benzetim modeli oluşturulmuştur. Bir üniversite kampüsü için uygulandığı varsayılan bu mikro şebekenin simülasyon modelinde yenilebilir enerji kaynakları olarak güneş santrali ve rüzgâr türbinleri bulunur. Tasarımı hedeflenen enerji yönetim sistemi bu kaynakların üreteceği güçleri tahmin eder. Bu bağlamda enerji yönetim sistemi yapay zekâ bünyesindeki en güncel tahmin algoritmalarını kullanır. Tasarım yapılırken ilk olarak güneş ve rüzgâr türbinlerinin üretecekleri gücün tahminine odaklanılır. Burada ortalama beş dakika zaman aralıklarında toplanmış beş yıllık gerçek meteorolojik verilerden oluşan veri kümesi kullanılır. Dokuz farklı veri tipinden oluşan meteorolojik veri kümesi bir dizi veri ön işlemeden geçirilmiştir. Eksik verileriler veri kümesinin özelliklerine uygun şekilde doldurulmuş ve aykırı değerdeki veriler ayıklanmıştır. Bu veri kümesindeki özellikler farklı grafikler ile analiz edilmiş ve eğitim için uygunluğu irdelenmiştir. Eğitim için uygun bulunan meteorolojik veri kümesine aynı bölge aynı saat/dakika aralıklarında üretim değerlerine sahip olduğumuz veriler etiket olarak eklenmiştir.

Güneş santralinin üreteceği gücü tahmin etmek için sıkça kullanıma sahip dört makine öğrenmesi metodu ile gradyan artırım makinesi temelli üç yenilikçi algoritma kullanılarak toplam yedi tahmin modeli geliştirilmiştir. Bu tahmin modeller eğitim için uygun hale getirilen eğitim veri kümesi ayrı ayrı eğitilmiştir. Birebir aynı eğitim kümesi ile eğitilen bu yedi tahmin modelinden elde edilen sonuçlar hem grafikler hem de tablolar halinde sunulmuştur. Hangi algoritmanın bu çalışma için ne kadar başarılı sonuç verdiği karşılaştırmalı şekilde sunulurken ek olarak hesaplama maliyeti de karşılaştırılmıştır. Tasarlanan enerji yönetim sistemi ayrıca rüzgâr türbinlerinden elde edilecek gücü de tahmin etmelidir. Bu bağlamda, üç farklı makine öğrenmesi algoritması ile tahmin modelleri oluşturulmuş ve sonuçlar elde edilmiştir. Birçok performans metriği kullanılarak bu tahmin modelleri karşılaştırılmıştır. Bu tez bünyesinde yapılan ve başarılı sonuçlar elde edilen yenilenebilir enerji kaynaklarının üretim gücü tahmini çalışması literatüre yeni yaklaşımlar ve benzersiz sonuçlar sunar.

Yapay zekâ temelli bir enerji yönetim sistemi kullanıcınsa enerji verimliliğinin yanında düşük enerji maliyeti ve karlılık da sunmalıdır. Günümüzde yaygınlaşan dinamik elektrik enerjisi fiyatlandırması göz önünde bulundurulmalıdır. Dinamik fiyatlandırma ülke çapındaki üretim ve tüketim arasında ilişki gözetilerek yapılır. Bu tez çalışmasında oluşturulan mikro şebeke benzetim modelinin dinamik fiyatlandırmanın uygulandığı bir ülkede yer aldığı varsayılır. Açık kaynaklı platformlardan elde edilen gerçek dinamik fiyat verilerinden beş yıllık bir veri kümesi oluşturulmuş ve analizi gerçekleştirilmiştir. Veri kümesi irdelenmiş, ön işlemden geçirilmiş ve tahmin modellerinin eğitimine hazır hale getirilmiştir. Bu çalışma için hafızalı hücre yapılarına sahip dört derin öğrenme algoritması seçilmiştir. Bu algoritmalar ve eğitim veri kümesi kullanılarak fiyat tahmin modelleri gerçekleştirilmiş ve sonuçlar elde edilmiştir. Modellerin öğrenme performansları, hata değerleri ve doğrulukları karşılaştırmalı şekilde sunulmuştur. Bu yenilikçi tahmin modelleri tasarlanan enerji yönetim sistemine entegre edilmiştir.

Bir mikro şebekeden talep edilen gücün bilinmesi o mikro şebekeye dair verilecek operasyonel kararları daha güçlü kılar. Günün hangi zamanında ne gibi bir yük talebinde bulunulacağı önemli bir parametredir. Yük talebinin önceden bilinmesi kaynakların kullanımı ile ilgili kararları etkiler. Bu gerçek ışığında tasarlanan enerji yönetim sisteminin yük talebini de tahmin edebilmesi gerekir. Bu kapsamda fiyat tahminine benzer şekilde hafızalı hücre yapılarına sahip dört derin öğrenme metodu kullanılarak yük talebi tahmin modelleri geliştirilmiştir. Açık kaynaklardan elde edilen gerçek yük değerleri oluşturulan mikro şebekenin simülasyon modeline göre ölçeklenmiştir. Beş yıllık yük verileri ile oluşturulan eğitim kümesi kullanılarak derin öğrenme modelleri eğitilmiş ve sonuçlar elde edilmiştir. Birçok performans metriği kullanılarak sonuçlar karşılaştırılmalı şekilde sunulmuştur. Bu çalışma sonucunda oldukça başarılı sonuçlar ortaya koyan tahmin modelleri tasarlanan enerji yönetim sistemi ne entegre edilmiştir.

Yapay zekâ temelli enerji yönetim sistemi yukarıda anlatılan birçok tahmin modelini kullanır. Uygulanan tüm makine öğrenmesi ve derin öğrenme metotlarına ait teorik ve matematiksel temeller tezin ikinci bölümünde verilir. Tasarımı anlatılan enerji yönetim sistemi mikro şebekeyi kontrol etmek için insan kaynaklı yönetime ek bir kontrolcüye ihtiyaç duyar. Bu kapsamda bu tez yine yapay zekâ temelli bir kontrolcü önermektedir.

Günümüzde klasik kontrol yöntemlerinin yerini alan veri güdümlü kontrol yöntemleri popüler araştırma konuları arasındadır. Bu tez çalışmasında bu tarz makine öğrenmesi temelli kontrol yöntemlerine odaklanılır. Bu kapsamda makine öğrenmesinin üç temel dalından biri olan pekiştirmeli öğrenme konusu araştırılmış ve temelleri verilmiştir. Pekiştirmeli öğrenme temelde sistemin matematiksel modeline ihtiyaç duymadan sistemi kontrol etme prensibine dayanan yöntemlere verilen genel isimdir. Bu kavramı tablo oluşturmaya dayanan yöntemler ile derin sinir ağları kullanan yöntemler olarak ayırmak mümkündür. Bu tez çalışmasında her iki çeşit yöntemleri kullanan kontrolcü ajanlar oluşturulmuştur. Pekiştirmeli öğrenmede sistemi kontrol etmeyi öğrenecek olan ajanın kendini optimize etmesi gerekir. Bu iyileştirme işlemini ise deneme yanılma yoluyla yapar. Ajanın bu deneme yanılmaları yapıp en iyi halini alabilmesi için kontrol edeceği sistemin ki bu tez çalışmasında bu bir mikro şebeke çevre modeline sahip olması gerekir. Tasarlanan enerji yönetim sitemine özgü olarak üçü zamansal-fark temelli ikisi derin pekiştirmeli öğrenme temelli olmak üzere beş farklı kontrol ajanı tasarlanmıştır. Bu ajanların kendini eğitebilmesi için mikro şebekeye özgün olarak tasarlanmış üç çevre modeli bu tez çalışmasında önerilir. Benzersiz ödüllendirme stratejilerine sahip olan bu çevre modelleri literatüre yeni bir yaklaşım sunar. Hem yenilebilir enerji kaynaklarını hem yük talebini hem de dinamik fiyatı ajanların eğitimi için kullanan bu çevre modelleri enerji yönetimi açısından oldukça başarılı sonuçlar ortaya koymuştur. Eğitilen pekiştirmeli öğrenme ajanları mikro şebekeyi yönetmeyi öğrenmiş kullanıcıya hatırı sayılır yüzdelerde karlılık sunmuştur.

Tasarım adımları anlatılan enerji yönetim sistemi birçok farklı yapay zekâ algoritmasını kullanır. Oluşturulan, eğitilen ve başarılı sonuçlar elde edilen bu yapay zekâ modelleri bu tez çalışmasında grafik arayüz ile tek bir çatı altında toplanmıştır. Özgün bir grafik arayüz tasarlanmış tüm tahmin modelleri ile kontrol ajanları bu tasarıma entegre edilmiştir. Toplamda yedi sayfadan oluşan bu arayüz tasarımı mikro şebekeye dair birçok parametreyi ve kontrol aksiyonunu kullanıcıya sunar. Kullanıcı geleceğe dair üretilecek güçleri, yük talebini ve fiyatın ne olacağını görebilir. Ayrıca, birçok kontrol aksiyonunu bu arayüz aracılığıyla mikro şebekeye uygulayabilir. Gerçek zamanlı birçok parametreyi de görebilen kullanıcı ayrıca tahmin modellerinin ve kontrol ajanlarının performanslarını ilgili sayfalar aracılığıyla analiz edebilir.

Sonuç olarak, bu tez çeşitli güncel ve yenilikçi algoritmaları bünyesinde barındıran ve özgün olarak kullanan yapay zekâ temelli enerji yönetim sistemini mikro şebekeler için önerir. Yapay zekâ temelli tahmin modelleri yine yapay zekâ temelli bir kontrol ajanının vereceği kararları belirler. Mikro şebeke için doğru kontrol aksiyonları seçmeyi öğrenen bu ajan belirlediği kontrol aksiyonunu tasarlanan arayüz aracılığı ile kullanıcıya sunar. Ayrıca, özgün tasarlanan enerji yönetim sistemi arayüzü kullanıcıya mikro şebekeye dair birçok parametreyi önceden görebilme imkânı tanır. Bu tez çalışması özgün çevre tasarımları, ajan önerileri, tahmin modelleri ve yenilikçi tasarıma sahip arayüzü ile literatüre katkıda bulunur.

1. INTRODUCTION

1.1 Thesis Statement

The importance of energy for humanity increases every day, and efficient and cost-effective use of energy is one of the most researched scientific topics. Such usage helps reduce the carbon footprint and contributes to leaving a less disrupted nature for future generations. Efficient use of electric energy, which is the most used type of energy in daily life, is crucial in this context. While electric energy is generated at a single point with high production capacity energy plants, it is transmitted to areas where residential areas and industrial facilities are located together. This transmission increases the cost of energy due to energy loss in lines and creates a more fragile situation against physical or cyber-attacks. The inefficiency in energy usage can be overcome with grid systems where production and consumption are located in the same place. While smart grids are a large-scale solution for energy efficiency, microgrids as the medium scale and nano-grids at the household level are becoming more prominent.

A microgrid is a small-scale local electrical grid system that can employ distributed energy resources (DERs) such as solar power plants, wind turbines, energy storage systems, generators and the energy provided by the utility grid. Some static microgrids are sized appropriately only at the installation stage, while others can be dynamically managed by software and communication tools. Such networks can use control solutions and optimization techniques. Microgrids can be used in various locations, including university campuses, military bases, industrial parks, and small towns and districts. Moreover, microgrids can operate in 'island mode' through DERs within it. Thanks to this operational advantage, it can meet the energy needs without utility grid connection in case of blackouts and emergencies. In addition, microgrids can transfer or sell the excess energy they generate to the utility grid via grid connection. It can reduce the energy cost with the profit being making in the energy trade.

While microgrids promise low cost and high efficiency in terms of energy, they enable to reduce greenhouse gas emissions. In addition, these systems increase the flexibility and reliability of the grid, while at the same time offering improvements in cybersecurity. They are becoming increasingly popular as a way to promote sustainable and decentralized energy systems. Today, many microgrids are sustained by classical control methods and operate only in a dimension that is sized by purely mathematical analysis methods. This situation both limits the efficiency that can be achieved during the operation of the microgrid and makes it difficult to follow the new trends in energy storage technologies. The idea that current energy storage technologies can be effectively used in the system and that today's artificial intelligence technology can form the basis of microgrid energy management system constitutes the essence and importance of the thesis. An artificial intelligence-based energy management system (AIEMS) can play a crucial role in optimizing the operation and management of microgrids. Here are some key reasons why AIEMS is important for microgrids:

- AIEMS can utilize real-time data from meteorological data sources, load demand, state of charge (SoC) of energy storage units and price data to provide high-accuracy predictions and optimize the energy usage. Therefore, AIEMS can reduce energy waste and improve productivity and leads to lower energy cost.

- Grid stability could be enhanced via AIEMS. It can dynamically monitor and manage to power output of microgrid components. AIEMS ensures to operate within safe and stable constraints. Moreover, AIEMS can offer preventing overloading and decrease the risk of blackouts and brownouts.

- Adaptation to fluctuations in energy demands and supply patterns can be easily handled by AIEMS. Since, AIEMS can adjust power level of different components in real-time. So, AIEMS can provide balance supply and demand and also reduce the need for backup generators.

- Better decision-making: AIEMS can provide insights into energy usage patterns, identify opportunities for energy savings, and suggest optimal maintenance schedules. This can help microgrid operators make informed decisions about energy management and maintenance.

- AIEMS can provide a better decision-maker with its energy usage information, the ability to identify earnings opportunities and the level of power to be obtained from DERs. Thanks to this decision-maker, operational decisions of the microgrid can be made more conveniently.

- AIEMS can improve energy sustainability by effectively using renewable energy sources such as solar power plants and wind turbines. In addition, it provides sustainable energy management by meeting that power need in the most optimal way. Sustainable energy management ultimately helps reduce carbon emissions.

The objective of this thesis is to create a dynamic energy management system for a microgrid using artificial intelligence (AI) techniques. The proposed system is expected to offer the advantages outlined above.

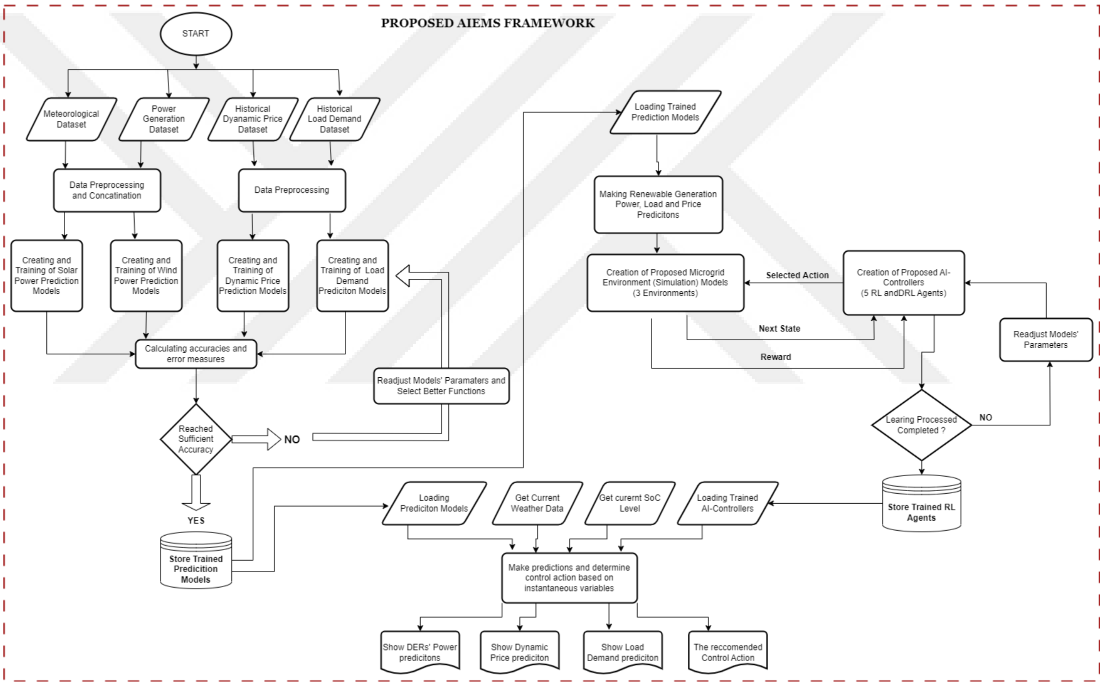


Figure 1.1: A generalized flowchart of the proposed AIEMS

The microgrid's energy flow variables are predicted using machine learning and deep learning methods. Additionally, an AI agent based on data-driven control is developed. All these results are integrated under the AIEMS framework.

1.2 Literature Review and Main Contribution

As research into energy continues to expand, efforts are being made to develop solutions that minimize energy loss, effectively manage distributed generation resources, and reduce carbon emissions in energy production, through the use of smart grids, microgrids, and nanogrids. Scientific issues related to the installation, sizing, and integration of energy storage units into the system, as well as the need for robust energy management, are prominent concerns in microgrid research [1]. These studies examine and evaluate the types of energy management systems that should be utilized and the technological infrastructure that is required [2]. [3] have emphasized the critical role that big data plays in energy management in microgrids. Another study has shown that the ability to predict the future under the control of a microgrid is essential to strengthen its control [4]. These and other similar studies underscore the importance of developing an energy management system for microgrids that is capable of processing big data and predicting the future. Therefore, it has become imperative to integrate AI techniques into the energy management system. Moreover, the use of AI methods on the main components of a microgrid has become increasingly important. For instance, [5] have examined the application of machine learning techniques on energy storage devices and systems. [6] have conducted important research on the application of deep learning methods for microgrid components. On the other hand, energy storage systems (ESSs) are regarded as a critical component of microgrids, facilitating the effective incorporation and regulation of renewable energy sources, such as solar and wind power. Therefore, the energy management of microgrids is the subject of constant research [7]. This thesis presents an advanced machine learning (ML)-based control approach. The specifics of ML-based control techniques are elaborated upon in great detail in [8] as data-driven control. The potential of data-driven-based energy management systems for microgrids is expounded upon in [9]. In tandem with this domain, where system dynamics are sought to be learned from data and ML models [10], the subject of reinforcement learning (RL), which involves learning to regulate the system through a model environment, is gaining traction. The use of RL in energy systems, in particular, is becoming increasingly popular. The intended basic decision maker of AIEMS aimed in this thesis is an RL-based artificial intelligence agent. If we look at similar studies in the literature. The study conducted by [11] investigates the use of reinforcement learning (RL) for optimal power control in a grid-connected microgrid. RL algorithms can be divided into basic and deep categories. In the works of [12] and [13], basic RL algorithms such as Q-Learning, SARSA, and Expected SARSA are utilized in energy management for microgrids with various reward strategies. The study of [14] proposes a scheduling strategy for energy storage in microgrids using the Q Learning algorithm, which significantly reduces the solution time. On the other hand, [15] employs deep reinforcement learning (DRL) for energy management in a microgrid, while [16] proposes DRL for energy management in multi-microgrids connected to the distribution system. In [16], the profitability of three microgrids is demonstrated through the implementation of the dueling Deep Q Network (DQN) and Deep Deterministic Policy Gradient (DDPG) algorithms. The dynamic operations of energy storage systems play a crucial role in microgrid management. To this end, [17] proposes an artificial intelligence (AI)-based energy management system interface, which offers control advice to the user using a dueling DQN-based DRL. Similarly, in [18], the Soft Actor-Critic (SAC) algorithm, another DRL method, is preferred for dynamic operations, and a solution to the coordination problem is proposed. The study of [19] performs frequency regulation of multi-microgrids with shared energy storage using DQN, DDPG, and twin delayed DDPG (TD3) algorithms of DRL. Energy storage systems (ESSs) can enhance the overall efficiency of microgrids by allowing load shifting and peak shaving, reducing the need for expensive peak generation capacity.

The effective operation of the energy management system (EMS) for homes, similar to microgrids, offers numerous benefits to the user. EMS using RL for homes is another frequently studied subject. DRL is preferred as the decision-maker in the EMS of homes in the works of [20], [21], and [22]. The DRL agent in these studies is trained with the dynamics of different types of loads in a house and provides users with benefits such as profitability and loss reduction. Furthermore, [23] improves the performance of the DRL agent with multistep prediction. Conversely, for large-scale smart grids, RL is also preferred to overcome different control problems. For instance, in [24], the control of ESS within a smart grid is carried out with DRL to increase profitability on a large scale. Similarly, [25] establishes the pricing strategy of renewable energy sources within a grid using DRL. The control of charging stations, an essential component of smart grids, with RL has been proposed in the works of [26], [27], and [28], which has been shown to be profitable. In other studies that consider smart grids, [29] utilizes DRL with prediction models for proper lithium-ion battery degradation, while [30] uses it for scheduling battery capacity.

Energy storage systems (ESS) are utilized to store the excess energy generated during peak production times, which can later be utilized during times of low production or high demand. This approach is critical in addressing the intermittency issues associated with renewable energy sources and contributes to enhancing their reliability. Moreover, ESS can function as a backup power supply during outages, and they enable microgrids to operate independently of the main grid, which, in turn, increases their overall resilience and reliability. Given these factors, RL-based studies have been conducted for both grid-connected and standalone ESS. For instance, in [31], a range of RL-based data-driven techniques were employed to manage the energy storage of a microgrid, while [32] proposed deep reinforcement learning for energy storage control of a wind turbine power plant. Similarly, specific studies have been conducted for electric vehicles, such as those presented in [33], [34], [35], and [36], wherein DRL-based energy management approaches were developed for vehicles that utilize hybrid energy storage units incorporating different types of storage systems. Across these studies, DRL was found to provide benefits such as profitability, accurate energy management, and peak shaving for electric vehicles.

After analyzing studies that discuss the use of RL as the main controller in energy systems, it can be concluded that this method will be the primary decision-maker in the targeted energy management of a microgrid. However, this thesis does not rely solely on the use of RL and presents a new approach. The main decision-maker AI agent is strengthened by predictions of power obtained from DERs such as solar and wind, predictions of the load to be demanded from the microgrid, and predictions of dynamic prices.

Power prediction studies from renewable energy sources hold a significant place in the literature. The article by Kim [37] presents a two-stage estimation study focused on predicting solar power generation. The first stage involves estimating irradiance data from numerical data, while the second stage involves predicting solar power generation using the estimated irradiance. Moreover, it should be noted that while some studies employ multiple algorithms in conjunction with each other [38], others opt to use modified and enhanced versions of a single algorithm in combination [39]. Furthermore, apart from the utilization of machine learning techniques, deep learning approaches have also been proposed in certain studies for the prediction of solar power generation [40], [41]. Specifically, in the works of [42], [43], and [44], artificial neural network (ANN) models are frequently employed for the purpose of prediction [42], whereas in some instances, LSTM models with memory cells or their derivatives are preferred [43], [44].

One of the key factors that determines the methods used in these studies is whether the intended prediction is for a short-term or long-term timeframe. In addition, successful results have been achieved in day-ahead forecasting [45]. The selection of appropriate data for training predictive models also significantly affects the performance outcomes of these models. Some studies incorporate geographic structural features for training, while others use satellite photos, which are popular training data in this context [46]. Convolutional Neural Networks (CNN) are a powerful deep learning method that stands out in this regard. Both LSTM and CNN are frequently used with deterministic, heuristic, or meta-heuristic optimization methods to enhance learning [47]. However, these approaches often require high computational power and long training times. On the other hand, meteorological data have a significant impact on the level of solar energy to be produced, particularly as radiation is directly related to solar power generation. Some studies have therefore focused on estimating solar radiation rather than power, and have achieved very successful results [48], [49].

Upon examining various prediction studies of solar power plants, two main challenges arise. Firstly, selecting and preprocessing a training dataset that accurately represents the problem at hand poses a difficulty. Secondly, despite achieving high accuracy, the long training time required to train models remains problematic. In order to address these challenges, researchers have investigated the use of meteorological data with multiple features in the training set, rather than relying solely on radiation data or satellite photos. Furthermore, innovative regression algorithms, such as those based on gradient boosting machine (GBM), have been explored to reduce training time. The XGBoost algorithm has been shown to be particularly promising for short-term energy forecasts [50], as well as for load prediction [51], [52]. Similarly, the combination of LightGBM and CNN has shown great potential for power generation forecasting of wind turbines [53], [54]. CatBoost, a relatively new GBM-based algorithm, has also demonstrated significant success in long-term load prediction [55].

The prediction of power generation from wind turbines, which are widely used in microgrids, poses a challenging problem. Studies have been conducted on power prediction for wind power plants, which constitute the second most important type of renewable energy source. Some of these studies aim to develop models for predicting the power output of a single turbine [56], while others focus on the power generation of a wind farm with a large-scale power plant [57]. Deep learning techniques, such as CNN and LSTM, supported by effective machine learning methods, have been proposed as solutions for wind power forecasting, which traditionally rely on DNN models [58]. In some studies, such as [59], the prediction of wind speed, the most fundamental variable that affects wind power generation, is performed using deep learning techniques instead of directly forecasting wind power.

On the other hand, dynamic price prediction is another essential component of microgrids or smart grids as it can provide an accurate forecast of electricity prices, which allows for efficient energy management and optimal decision-making. In a traditional power grid, electricity prices are usually fixed, but in a microgrid or smart grid, electricity prices can fluctuate based on the demand and supply of electricity. These fluctuations can occur due to various factors such as peak demand, renewable energy availability, and grid congestion. By accurately predicting these price fluctuations, microgrid operators can adjust their energy consumption to minimize their energy costs. [59] presents a comprehensive review on dynamic price and load demand prediction. [60] proposes the use of Kalman Filters and Echo State Network methods together for dynamic price forecasting and applies it for energy price forecasting of Italy. In the area of price forecasting, machine learning techniques have been extensively employed [61], [62], [63]. In [64], prediction studies are carried out using machine learning methods with the dynamic price dataset of Germany, France, Belgium, and Switzerland. The comparison of Support Vector Machine and other methods is made based on the output of error calculation functions. Additionally, in the [65] study, the study with the electricity price dataset of the three states of the United States, Pennsylvania, New Jersey and Maryland, suggests the use of extreme kernel learning machine (KELM) and auto regressive moving average (ARMA) techniques supported by wavelet transform. The success of the models that closely follow the change in price has been demonstrated. Despite the success of kernel-based ML techniques, it can be deduced that the use of deep neural networks will give better results in this prediction study, which is tightly dependent on time [66], [67], [68]. In [69], models were built using the same dataset with both machine learning techniques and deep learning techniques. [69] and [70] show that deep learning models are superior to ML-based techniques for this problem. The dynamic change in the electricity price is closely tied to the life routine of societies, that is, at what hour and at what minute of the day. This fact leads to the conclusion that deep neural network structures with memory cell structures would be more successful for price prediction. [71], [72] and [73] studies show that recurrent neural networks (RNN) and Long Short-Term Memory (LSTM) structures show exceptional results in dynamic electricity price prediction. Moreover, a hybrid electricity price prediction model, which is a combination of LSTM and Bayesian optimization and hyperband (BOHB), is proposed in [74]. In terms of error values and accuracy, better results were obtained when compared to using LSTM alone.

Further, knowing the amount of load to be demanded from a microgrid is very important. Accurate load forecasting helps to ensure that enough power is generated and available to meet the needs of customers, reducing the risk of outages or other disruptions. In addition, load forecasting enables grid operators to better manage the integration of renewable energy sources, which can be variable and unpredictable. By predicting the expected load, operators can adjust the production of renewable energy to match the expected demand, ensuring that the energy is used efficiently and effectively. Moreover, accurate load forecasting can also help to promote energy efficiency and conservation by providing consumers with real-time pricing signals based on the current state of the grid. This can encourage consumers to adjust their energy usage during times of high demand or low supply, reducing strain on the grid and promoting sustainable energy practices. Additionally, the datasets containing dynamic pricing and load demands exhibit undeniable similarities. This fact consequently leads to the inference that the deep learning techniques proposed for price prediction are also likely to yield favorable outcomes for load forecasting. In the [75] study, a random forest-enhanced gated recurrent units (GRU)-based load estimation model is proposed, while [76] presents an LSTM-based load estimation model. In another study, a hybrid model is created by combining LSTM with convolutional neural network (CNN) for accurate load prediction [77].

In conclusion, the AIEMS described in this thesis utilizes several machine learning and deep learning methods in combination. It has been analyzed in terms of various microgrid dimensions and a microgrid simulation model with average dimensions has been prepared. An appropriate AIEMS design is implemented based on this model.

The main contributions of the proposed and designed the novel AIEMS can be listed as follows:

- In this thesis, novel ML models are proposed for the power generation prediction of a solar power plant, which is essential for the efficient management of a microgrid, and are compared with conventional ML techniques, demonstrating their superiority.
- Similar to solar power prediction study, a new strategy and novel ML models are proposed for the power prediction of wind turbines.
- Profitability is of great importance in microgrid energy management. Therefore, novel DL-based prediction models are developed and proposed for dynamic pricing, which is widely used worldwide.
- DL-based prediction models are developed for predicting the level of load demand at different minutes and hours, and the use of these models, which are different from the conventional ones, is proposed for load prediction.
- AI-based controllers are designed within the scope of this thesis, and a simulation model of the microgrid is required for these controllers to optimize themselves. Within this scope, three different microgrid environment models have been developed and proposed.
- Five different AI-controllers, three of which are vanilla RL and two of which are DRL, have been designed within the scope of this thesis. These RL agents trained provide highly successful results for accurate energy management. This RL-based control methodology is proposed.
- A GUI that includes all methods has been designed within the scope of this thesis. All design steps are provided. The user interface of the proposed AIEMS offers unique contributions both in terms of data monitoring and analysis and microgrid control.

This proposed AIEMS offers unprecedented and novel prediction models for renewable energy sources. Additionally, the load prediction and energy price prediction for the microgrid purchasing energy from the utility grid, are more innovative compared to previous studies in the literature. Furthermore, distinctive AI-based decision agents are designed for the proposed AIEMS in this thesis. These proposed deep reinforcement learning agents are more advanced and innovative than previous studies in the literature. The AIEMS presented for microgrids is original, innovative, and exceptional in many aspects.

1.3 Organization

This thesis, which aims to cover all the design steps of the targeted energy management system, consists of six chapters. In the following Chapter 2, the mathematical foundations and specifications of the prediction algorithms used in the energy management system are described. The first part of Chapter 2 focuses on well-known and classically defined machine learning methods, followed by the presentation of the mathematical foundations of gradient boosting machine-based algorithms proposed for the prediction of power obtained from DERs of microgrids in this thesis. Lastly, the fundamentals of deep learning methods used are explained in Chapter 2.

Chapter 3 is dedicated to the AI-based control in the energy management system of the microgrid. Here, the infrastructure of machine learning-based control is explained, including the fundamentals of RL and a description of what deep reinforcement learning is. The foundations of the algorithms used by the RL agents tested as controllers in this thesis are also presented in this section.

In Chapter 4, first, the proposed formulation of the energy management system for a microgrid is presented. The dynamics of the simulation model of the microgrid are explained, along with the proposed functions of the designed environmental model. Chapter 4 also includes the results of the created prediction models, and the outputs of the trained RL agents' performance are presented.

In Chapter 5, the completed and successful prediction models and RL agents are consolidated under a single umbrella. This section presents the designed interface of the energy management system. All the features of this interface are described, and the advantages it provides to the user are demonstrated.

Finally, Chapter 6 presents the conclusion of the thesis. The contributions and results of this study are summarized, and future research goals in this field are discussed.

2. THEORETICAL AND MATHEMATICAL FOUNDATIONS OF PREDICTIVE MODELS

2.1 Introduction

The level of power to be obtained from the energy sources of a microgrid, together with the level of the load to be demanded at that moment, provides considerable benefits. Moreover, it is highly important to know the current price in terms of the energy cost of this microgrid, which it is assumed that the microgrid is operating in a country with dynamic pricing. An AIEMS targeted in this thesis is able to make these predictions with the prediction models included in it. In this chapter, the fundamentals of predictions models used by the designed energy management system are explained.

First, the well-known prediction techniques of machine learning are briefly described. With these techniques, the power values to be obtained from renewable energy sources have been predicted. In addition, these classical methods are used to compare the gradient boosting machine (GBM)-based techniques proposed in this thesis. Secondly, GBM-based techniques used for estimating the power to be obtained from solar panels and wind turbines are explained. Finally, deep learning methods, which are the basis of prediction models developed for load demand and dynamic price forecasting, are explained.

2.2 Prevalent Prediction Methods in Machine Learning

The designed AIEMS can predict the power to be generated from renewable energy sources. These predictions are initially made using methods that are relatively classic and well-known in machine learning. Here, four prominent prediction models used in the thesis are briefly described.

2.2.1 Polynomial regression

There may be a linear relationship between some types of data in a dataset and the result that will emerge from that data. In this case, linear regression models could yield good results. However, if the second, third or more degrees of the data, not the first order, are effective on the result, then a polynomial regression model is needed so that an accurate prediction result can be obtained. When it comes to elucidate polynomial regression, polynomial regression, which is a special type of linear regression, or rather multi linear regression, is preferred when the effect of the data on the result is not linear. First, the equation of the linear regression model given in equation 2.1.

$$\mathbf{Y} \approx \beta_0 + \beta_1 \mathbf{X} + \epsilon \qquad (2.1)$$

Where, **Y** is label vector and **X** is input vector. This model, which is used to predict outputs with an input data, has been enriched to establish a linear connection between multiple inputs and outputs. The equation called multi linear regression given in equation 2.2.

$$\mathbf{Y} \approx \beta_0 + \beta_1 \mathbf{X_1} + \beta_2 \mathbf{X_2} + \cdots + \beta_n \mathbf{X_n} + \epsilon \qquad (2.2)$$

This regression equation, which produces a suitable estimation result for many problems, may be insufficient in cases where the relationship between input data and output is not linear. In this case, the polynomial regression equation comes into play (equation 2.3.)

$$\mathbf{Y} \approx \beta_0 + \beta_1 \mathbf{X_1} + \beta_2 \mathbf{X_2^2} + \beta_3 X_3^3 + \cdots + \beta_n \mathbf{X_n^n} + \epsilon \qquad (2.3)$$

This type of regression can give appropriate results in creating an appropriate estimation model for the problem. The prediction model specific to the theis using meteorological data is created using polynomial regression and 4 degrees are used.

2.2.2 Support vector machines

Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for classification or regression problems. However, it is mostly used in classification problems. The regression type of this algorithm, which is preferred in this study to create a prediction model, is called Support Vector Regressor (SVR). SVR is a supervised learning algorithm that is used to predict discrete values. SVR uses the same principle as the SVMs. The fundamental idea behind SVR is to find the best fit line (kernel function). In SVR, the best fit line is the hyperplane that has the maximum number of points. This algorithm is fundamentally based on the selection and arrangement of a kernel function according to the problem. Thus, a kernel function is selected (linear or not) and arranged in such a way as to best separate the data if the purpose is to classify. Otherwise, if the goal is to making prediction, this time the kernel function that will fit the data best is selected and edited. The most used (major) kernel functions in SVM: Gaussian kernel, Gaussian Kernel Radial Basis Function (RBF) which gives sufficiently good results in many problems, polynomial kernel and linear kernel.

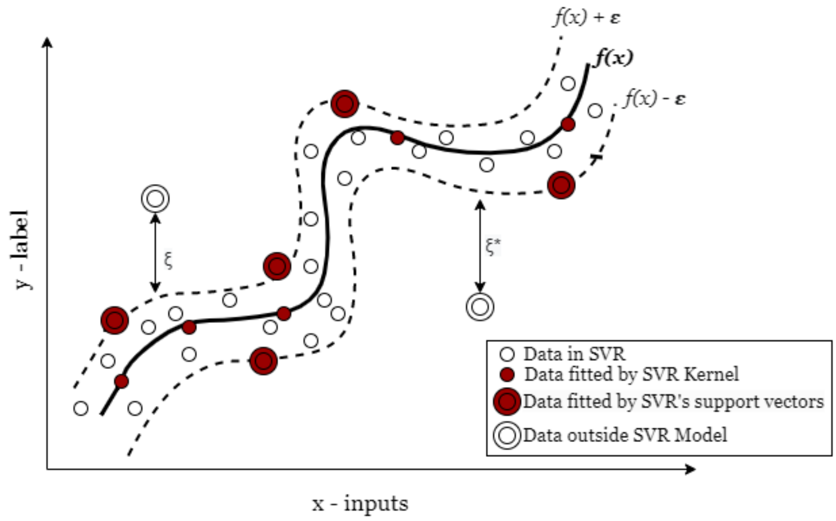


Figure 2.1 : Support Vector Regression.

The aim here is to get as many data points as possible between the support vectors with the help of the kernel function to be selected, and to minimize the error between the data outside the support vectors and also, in the support vectors. This can be seen in Figure 2.1. In addition to the short explanation in this section, how SVM is used in estimation and classification problems is explained in detail in the review articles. It can be seen quite clearly in these studies that it is a very powerful method [78].

2.2.3 Decision tree

Decision Tree (DT) is a machine learning algorithm based on the logic of dividing a dataset of variables into certain regions. Although it is generally used for classification problems, it can also be preferred as a regression tree for regression problems and good results can be obtained. A tree structure is created from the sections created from the dataset. It is used that regression trees as the prediction algorithm and the goal are to generate non-overlapping regions using the learning set variables.

$$X_1, X_2, X_3, \dots X_p \longrightarrow R_1, R_2, R_3, \dots R_j$$

Where: *Xs* represent features of the dataset and *Rs* are regions that the DT algorithms creates. Here, our estimation is the average of the output values (y_j) of the training data in the R_j region, for example, the *j* region. So how do we decide on these regions

$(R_1, R_2, R_3, \ldots R_j)$? At this point, our model uses the Residual Squares Error function as in linear connection.

$$\sum_{j=1}^{J} \sum_{j \in 1} \left(y_i - \widehat{y_{R_j}}\right)^2 \tag{2.4}$$

Where, $\widehat{y_{R_j}}$ is average of outputs in R_j region. In decision trees, regions are separated according to the recursive binary splitting method. The formula is given in equation 2.5.

$$R_1(\mathrm{j}, \mathrm{s}) = \{\mathrm{X}|\mathrm{X_j} < \mathrm{s}\} \qquad R_2(\mathrm{j}, \mathrm{s}) = \{\mathrm{X}|\mathrm{X_j} \geq \mathrm{s}\} \tag{2.5}$$

Where, the *j* and *s* values that minimize the following expression are being tried to find.

$$\sum_{i: x_i \in R_1(j,s)} \left(y_i - \widehat{y_{R_1}}\right)^2 + \sum_{i: x_i \in R_2(j,s)} \left(y_i - \widehat{y_{R_2}}\right)^2 \tag{2.6}$$

After dividing into regions, if there are very few points left in any region, it will be stopped. If the values shown by the upward arrow in Figure 2.2 on right, are considered as the average values of the separated regions, they show our estimation values. If we divide our data into too many regions in our learning set, the model that we are trying to obtain eventually will over-learn (be overfitted) because it will memorize each data and its output value.

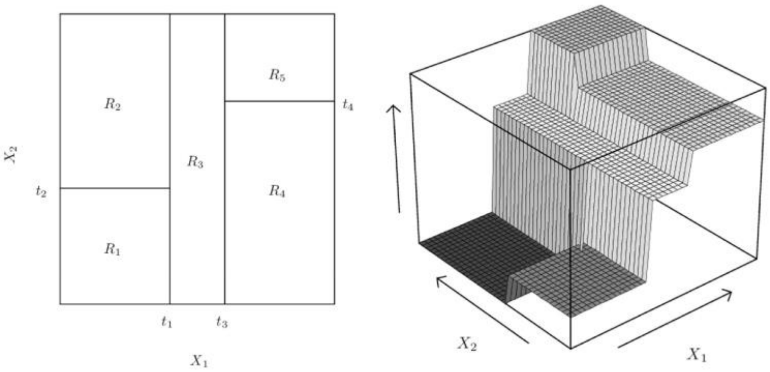


Figure 2.2 : Identifying regions in DT, X_1 and X_2 features of dataset. On right, surfaces corresponding to the tree being created.

In order to prevent over-learning, tree pruning comes to play, which is basically based on adjusting the size of our tree.

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} \left(y_i - \hat{y}_{Rj}\right)^2 + \alpha\, |T| \tag{2.7}$$

Where, T ⊂ T0: subtree, |T|: Number of terminal nodes in T tree, R_m: The region corresponding to the terminal node m and α: hyperparameter. By examining the formula above, it can be said that in order to reduce the output of the formula, we will either reduce the number of nodes in the tree or reduce our squared error function. The value that determines the size of the tree is done by k-fold cross validation, in this way the most appropriate value is determined, and the best prediction result is found.

Last of all, DT algorithm creates a multidimensional tree structure on the dataset and produces predictions. This working principle has made this algorithm one of the indispensable methods for both classification and prediction problems. The adequacy, efficiency, and use of this method in the fields of classification and forecasting has been demonstrated [79].

2.2.4 Random forrest

The random forests algorithm is basically based on the decision trees method and is an Ensemble Learning (EL) method. Bagging in EL can be described when constructing multiple decision trees in the same learning set and reflecting the average of their outputs as the result. The Random Forest method aims to reduce the similarity between the randomly selected sub-datasets from the dataset and the trees built by establishing smaller trees. This structure is demonstrated in Figure 4. These decision trees that make up the random forest algorithm can be created on several different bases. Several decision tree algorithms have been proposed, differing in their feature selection methods, such as ID3 [80], C4.5 [81], and CART [82]. In the previous section, it is explained on which mathematical basis the trees of the CART algorithm are created. However, these trees can be created in different ways. For example, among these trees, ID3 and C4.5 choose entropy as an information measure to calculate and evaluate the quality of a split node based on a particular attribute. Entropy is used in these algorithms to determine from which feature the tree will start to form. It does this by calculating the information gain. How it is calculated is formulated in equation 2.8.

$$info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i) \qquad (2.8)$$

In equation 2.8, information value of a feature in dataset is formulated, where pi is the probability value of being chosen. Then, for each data in the feature is included in calculation that is shown below (see equation 2.9).

$$info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} . I(D_j) \tag{2.9}$$

After calculating the general and specific info value, the difference between these values gives us the information gain value.

$$Gain(A) = info(D) - info_A(D) \tag{2.10}$$

Calculated separately for each feature, the feature with the largest gain value is used at the beginning of the tree. Then the branching continues by making these calculations again [83]. After all trees in the random forest have been created and the prediction values have been obtained, the average of the results from more than one tree with reduced correlation returns to us as output data. First big advantages of RF, this algorithm can be used in both classification and regression problems and gives good results. The second advantage is that overfitting or over-learning is a critical problem that adversely affects the results, but for the RF algorithm, if there are enough trees in the forest, the probability of the overfitting problem will decrease.

In the use of this algorithm, the number of random trees can be entered as a hyperparameter. Appropriate selection of this parameter changes the quality of learning. This forest structure needs to be designed specifically for the problem. Before this model can be trained, some parameters need to be chosen specific to the problem. These are: number of trees, maximum number of features to use in trees, minimum number of leaves to be used to split a node. The number of trees in the forest to be established and the level of randomness in the random mixing of the data are very critical for the correct solution of the problem. Further, the randomness parameter, which generates random subsets from the main dataset, should be fine-tuned so that the training can be completed accurately and quickly. This algorithm, which is one of the two basic elements of the ensemble learning concept, shows very good prediction results even though it works slowly with its parallel working structure.

2.3 Gradient Boosting Machine Based Prediction Methods

EL algorithms have become prominent in many artificial intelligence-based prediction-themed academic publications, competitions, and prediction projects that aim for high accuracy. This concept is based on creating multiple models for a prediction problem instead of relying on a single model and combining their results for the final output. EL methods are divided into two categories: bagging and boosting. In the bagging concept, multiple models are trained simultaneously and in parallel using the same dataset. Different tree structures are used in the training process, and the different prediction results are reduced to a single result through a voting method for use. In the bagging method, the models work in parallel. The most well-known algorithm in bagging is RF algorithm. The boosting method has a sequential working structure. After the first prediction model is created, a new prediction model is obtained based on the results obtained until the desired model number is reached. As a result of this sequential method, the improved prediction result in each new model is presented as the final output. Gradient boosting, one of the two main branches of the EL area, is an increasingly popular field. Boosting is a method for transforming weak learners into strong learners. In this boosting method that works by creating tree structures using DT algorithm, each new tree progresses by applying a modified version of the original dataset. Unlike the RF algorithm, GBM creates relatively weaker trees instead of multiple large tree structures. Each new tree takes the output of the previous tree as input and produces its own output. All of these weak but mutually supportive tree structures ultimately produce a strong result. The basic and sequential working structure of GBM-based regression is shown in Figure 2.3.

Special algorithms based on GBM have been developed over time by many researchers and large institutes/technology companies. The first pioneer of these is the AdaBoost algorithm. AdaBoost is considered the first boosting algorithm and has won the Gödel Prize, one of the most important awards in the computer world. In this algorithm, the training set is first trained with a weak learner. After training, the incorrectly predicted examples are important for this algorithm. In the next training, the incorrectly learned training data in the first prediction is given more priority, i.e., the weights are increased, and it is retrained.

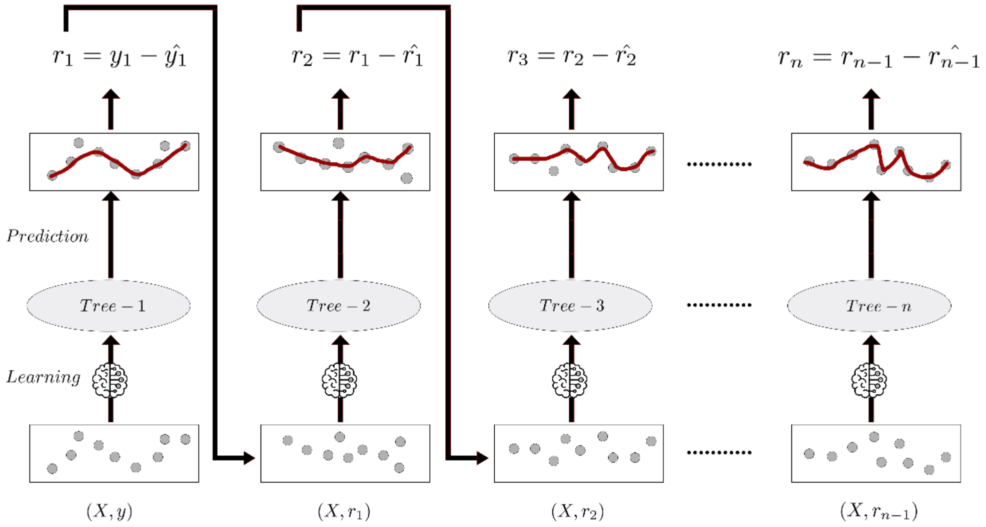


Figure 2.3 : Fundamental overview of GBM-based prediction algoritms.

Following the AdaBoost method, the XGBoost, LightGBM, and CatBoost algorithms were developed. In this thesis, prediction models were developed using the new methods XGBoost, LightGBM, and CatBoost.

2.3.1 XGBoost

XGBoost is an algorithm derived from the term "Extreme Gradient Boosting". It is based on Friedman's article "Greedy Function Approximation: A Gradient Boosting Machine" [84]. It has been proposed as a more advanced version of the AdaBoost algorithm. XGBoost is used for supervised learning problems with input data x_i (with multiple features) and training data to predict a target variable (label). XGBoost and other GBM-based algorithms work by constructing a tree structure similar to the DT algorithm. Prior to the invention of boosting methods, several DT algorithms with different feature selection methods such as ID3 [80], C4.5 [81], and CART [82] were proposed in the literature. Firstly, the DT integration, which is the model selection of XGBoost, should be addressed. The tree structures that are combined and applied in this boosting algorithm are based on CART. CART is slightly different from other decision trees in that each leaf is associated with an actual score, which provides richer interpretations beyond classification. This also provides a principled, unified approach to optimization. Generally, a single tree is not powerful enough to be used in practice. In fact, the model used is an ensemble of multiple trees, whose predictions are summed up as explained, known as the EL model. Therefore, the prediction scores of each tree are aggregated to obtain the eventual score. This can be formulated as follows in equation 2.11.

$$\widehat{y_i} = \sum_{k=1}^{K} f_k(x_i)\,, f_k \in \mathcal{F} \tag{2.11}$$

In the formula, **K** represents the number of trees. Also, $\mathcal{F}$ is the set of all possible CARTs. Also, f_k is a function in the practical space $\mathcal{F}$. The objective function to be optimized is defined in equation 2.12.

$$obj(\theta) = \sum_{i}^{n} l(y_i, \widehat{y_i}) + \sum_{k=1}^{K} \omega(f_k) \tag{2.12}$$

where $\omega(f_k)$ represents the sophistication of the tree f_k. It is elaborated in detail. Boosting methods are built upon the use of multiple trees and bear a close resemblance to RF. The primary difference between these two methods lies in their tree structures: while boosting methods rely on sequential trees, RF utilizes a parallel tree structure. Despite this difference, RF and boosted trees are effectively the same models, with variation arising from differences in their respective training methodologies. Therefore, the creation of a predictive service for tree ensembles requires only one model that can function effectively for both RF and GBM trees. When working with tree boosting, the objective function shifts as a result of the incorporation of multiple trees [85].

$$obj = \sum_{i=1}^{n} l\left(y_i, \widehat{y_i^{(t)}}\right) + \sum_{i=1}^{t} \omega(f_i) \tag{2.13}$$

While establishing the trees used by the methods, the parameters that make these trees suitable for the problem had better be resolved. These can be differentiated that what it needed to learn are those functions f_i, each represents the structure of the tree and the leaf scores. Learning tree structure is more difficult than common optimization problem where can be simply taken the gradient. It is intransigent to learn all the trees at once. Instead, it is utilized an alternative strategy: correct what is has been learned and add one new tree at a time. It can be defined the prediction value at step as $\widehat{y_i^{(t)}}$. Accordingly, equation 2.14 is obtained and given.

$$\widehat{y_i^{(0)}} = 0$$
$$\widehat{y_i^{(1)}} = f_1(x_i) = \widehat{y_i^{(0)}} + f_1(x_i)$$

$$\widehat{y_i^{(2)}} = f_1(x_i) + f_2(x_i) = \widehat{y_i^{(1)}} + f_2(x_i)$$

…

$$\widehat{y_i^{(t)}} = \sum_{k=1}^{t} f_k(x_i) = \widehat{y_i^{(t-1)}} + f_t(x_i) \tag{2.14}$$

In order to expedite the learning process, it is essential to ascertain the selection of pertinent trees that will be retained and those that will be deemed unnecessary at each step or stage. The elimination of redundant trees from the structure can significantly augment the pace of the learning process. It is judicious to add the tree that optimizes the objective, as it serves as a practical approach in achieving an optimal model.

$$\begin{aligned} obj^{(t)} &= \sum_{i=1}^{n} l\left(y_i, \widehat{y_i^{(t)}}\right) + \sum_{i=1}^{t} \omega(f_i) \\ &= \sum_{i=1}^{n} l\left(y_i, \widehat{y_i^{(t-1)}} + f_t(x_i)\right) + \omega(f_t) + \text{const} \end{aligned} \tag{2.15}$$

If it is selected employing mean squared error (MSE) as the loss function, the objective turns into the form that in equation 2.16.

$$\begin{aligned} obj^{(t)} &= \sum_{i=1}^{n} \left(y_i - \left(\widehat{y_i^{(t-1)}} + f_t(x_i)\right)\right)^2 + \sum_{i=1}^{t} \omega(f_i) \\ &= \sum_{i=1}^{n} \left[2\left(\widehat{y_i^{(t-1)}} - y_i\right) f_t(x_i) + f_t(x_i)^2\right] + \omega(f_t) + \text{const} \end{aligned} \tag{2.16}$$

The form of MSE is basic with a first-order term called the residual and a quadratic term. For other losses of interest (i.e. logistic loss), it is not so simple to obtain such a useful form. Hence, in the common circumstance, it is performed the Taylor expansion of the loss function up to the second order:

$$obj^{(t)} = \sum_{i=1}^{n} \left[l\left(y_i, \widehat{y_i^{(t-1)}}\right) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \omega(f_t) + \text{const} \tag{2.17}$$

where the g_i and h_i are defined as

$$g_i = \partial_{\widehat{y_i^{(t-1)}}}\, l\left(y_i, \widehat{y_i^{(t-1)}}\right) \tag{2.18}$$

$$h_i = \partial^2_{\widehat{y_i^{(t-1)}}}\, l\left(y_i, \widehat{y_i^{(t-1)}}\right) \tag{2.19}$$

Accordingly, if it is eliminated all the constants, the specific objective at step transforms into the form as equation 2.20.

$$\sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + \omega(f_t) \tag{2.20}$$

Now, this is the optimization target for the upcoming tree. Primer significant benefit of this representation is that the result of the objective function only relies on the g_i and h_i. Thus, this is how XGBoost provides traditional loss functions. It can be optimized every loss function, including logistic regression, and pairwise ranking etc., totally employing the identical solver that utilizes g_i and h_i as input. *Model complexity:* The regularization term is a primary element in the model's performance metrics and represents complexity. It is demanded to determine the complexity of the tree $\omega(f)$. To do so, firstly it should be refined the expression of the tree $f(x)$ as in equation 2.21.

$$f_t(x) = w_{q(x)}, w \in R^T, \quad q: R^d \rightarrow \{1,2,\cdots,T\} \tag{2.21}$$

where, *w* is the vector of scores on leaves, *q* is a function assigning each data point to the corresponding leaf, and *T* is the number of leaves. In XGBoost, expression the complexity is given in equation 2.22.

$$\omega(f) = \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2 \tag{2.22}$$

In practice, there exist various approaches to defining complexity, but the following method has proven effective. Many tree learning packages often neglect or treat less rigorously the issue of regularization. This is because traditional approaches to tree learning have primarily focused on improving impurity while leaving complexity control to heuristics. However, formal definition of complexity can lead to a better understanding of the learning process and facilitate the creation of models that perform well in various situations. To ensure efficient learning, appropriate scoring procedures should be used in this framework. Once the tree model has been re-formulated, the objective value with the *t*-th tree can be expressed as follows:

$$\begin{aligned} obj^{(t)} &\approx \sum_{i=1}^{n}\left[g_i w_{q(x_i)} + \frac{1}{2}h_i w_{q(x_i)}^2\right] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right)w_j + \frac{1}{2}\left(\sum_{i\in I_j} h_i + \lambda\right)w_j^2\right] + \gamma T \end{aligned} \tag{2.23}$$

Where, $I_j = \{i|q(x_i) = j\}$ is the set of indices of data points appointed to the *j*-th leaf. As it could be seen that in the second line it has been differentiated the index of the aggregation because all the data points on the same leaf get the equal score. Accordingly, it could be compactly expressed the calculation by defining $G_j = \sum_{i\in I_j} g_i$ and $H_j = \sum_{i\in I_j} h_i$. Then, objective function can be represented in equation 2.24.

$$obj = \sum_{j=1}^{T}\left[G_j w_j + \frac{1}{2}\left(H_j + \lambda\right)w_j^2\right] + \gamma T \tag{2.24}$$

In this equation, $\mathrm{w_j}$ are independent with respect to each other. The form of $\mathrm{G_j w_j} + \frac{1}{2}\left(\mathrm{H_j} + \lambda\right)\mathrm{w_j^2}$ is quadratic and the best $\mathrm{w_j}$ for a given structure $q(x)$ and the best objective reduction is given in equation 2.25 and 2.26.

$$w_j^* = -\frac{G_j}{H_j + \lambda} \tag{2.25}$$

$$obj^* = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T \tag{2.26}$$

The final equation illustrates the measure of acceptability and adequacy of a given tree structure *q(x)*. Essentially, the algorithm assigns the statistics of the leaves belonging to a particular tree structure, sums them, and applies the formula to determine the suitability of the tree. This score is similar to the impurity measure in decision trees, but it also accounts for the complexity of the model. The evaluation and scoring of trees allow for a logical ranking, and the tree with the best score is selected. However, choosing the optimal tree in practice can be challenging, and therefore, trees are optimized at each level by attempting to split a leaf into two and using the score obtained from the formula in equation 2.27.

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right] - \gamma \tag{2.27}$$

The formula defined and given in this section can be deconstructed into four constituents. First, the score on the new left leaf. Second, the score on the new right leaf. Third, the score on the original leaf and fourth, the regularization on the additional leaf. It is noteworthy that if the gain is smaller than a certain value, γ, it would be better not to add the branch. This pruning technique is commonly employed in tree-based learning models. Additional details on the GBM-based method can be found in [85].

By considering the fundamentals of supervised learning, the reasons for the effectiveness of these techniques can be directly revealed. XGBoost is an example of a tool that is based on the formal principles introduced in this subsection, and it has been developed with deep consideration of both systems optimization and principles in machine learning. The same mathematical methodology is employed in LightGBM and CatBoost, which will be described in the next subsections of this chapter. In these subsections, the prominent features of these algorithms will be explained rather than their mathematical basis.

2.3.2 LightGBM

LightGBM is a boosting algorithm introduced as part of the Microsoft DMTK project in 2017. It boasts several advantages over other boosting algorithms, including high processing speed, large data processing capacity, reduced RAM usage, high prediction accuracy, and support for parallel and GPU learning. According to the article "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," LightGBM is 20 times faster than other models [86]. This histogram-based algorithm reduces computational costs by discretizing continuous variables. The training time for decision trees is directly proportional to the number of divisions, which this method reduces. Therefore, both the training time and resource usage are minimized. Decision trees can be learned using two strategies: level-wise (depth-wise) or leaf-wise. The level-oriented strategy maintains the balance of the tree as it grows, while the leaf-focused strategy continually divides from leaves to reduce loss. This feature distinguishes LightGBM from other boosting algorithms and enables it to learn faster with less error rate. However, the leaf-oriented growth strategy may lead to over-learning in low-data scenarios, making the algorithm more suitable for big data. The number of leaves and tree depth can be optimized to prevent over-learning. LightGBM utilizes two techniques different from other algorithms, Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB), which deal with data samples and variables, respectively. GOSS reduces the number of data samples while maintaining the accuracy of decision trees by omitting data that may be considered less important. EFB reduces the number of variables without harming the accuracy rate, which increases the efficiency of model training. It has two processing steps, creating packages, and merging variables into the same package, to combine sparse features into denser ones, thus reducing complexity and memory consumption.

LightGBM produces successful results in both classification and regression studies, thanks to its improvements over GBM-based XGBoost. Furthermore, models created with this structure can be trained much faster than those of other algorithms.

2.3.3 CatBoost

CatBoost is a machine learning algorithm, based on GBM, that was developed by the Yandex company and introduced as an alternative to XGBoost and LightGBM in April 2017 with the article "CatBoost: unbiased boosting with categorical features" [87]. It is open-source and the algorithm's name is derived from the words "Category" and "Boosting". Catboost has various features that distinguish it from other algorithms, including its ability to handle numerical, categorical, and text data, its GPU support, its high learning speed, and its visualization options. Additionally, Catboost has an important position in the data preparation phase, as it can deal with empty data and encoding categorical data without the need for a coding process. This unique coding method significantly improves the learning speed and the quality of the results. Furthermore, Catboost builds symmetric trees to achieve a high prediction rate without establishing very deep trees, thus overcoming the over-fitting problem.

In the present thesis, the CatBoost algorithm is employed to predict the power output of a solar power plant, as a component of a larger energy management system. Although CatBoost is primarily designed for categorical datasets, it is used in this study to analyze a numerical meteorological dataset, with satisfactory results. One advantage of the algorithm is its ability to detect over-fitting and halt learning before reaching the pre-specified tree limit. This feature is set from the initial parameters. The use of a GPU is also important in machine learning and deep learning projects, as it can significantly reduce the learning time. CatBoost can be trained with a GPU, provided that an NVIDIA Driver version 418 or higher graphics card is available. During the learning phase, CatBoost takes snapshots, which enables the learning process to resume from where it left off if an unexpected problem occurs. This specific feature can be enabled or disabled by adjusting the parameters. This boosting method is particularly beneficial when dealing with large volumes of data. Once the model is built, CatBoost offers three different performance measurement metrics: PredictionValuesChange, LossFunctionChange, and InternalFeatureImportance. PredictionValuesChange indicates how much the prediction changes on average if the variable value changes for each independent variable, while LossFunctionChange represents the difference between the lost value of the model with and without a particular property for each argument. The model without the variable is the model that will be trained if that variable is removed from the dataset. In addition to these features, CatBoost has model parameters that can be adjusted specifically for regression or classification problems.

In conclusion, the mathematical foundations and principles of three GBM-based algorithms, which are preferred for creating prediction models, are elucidated in the subsection. In the following subsection, the fundamentals of deep learning-based prediction models used in this thesis are briefly explained.

2.4 Deep Learning Based Prediction Methods

Deep learning is a subset of machine learning where neural networks and human brain-inspired algorithms learn from data. The layer structure established in deep learning can be grouped as multilayer perceptron as known as ANN, Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). Depending on the nature of the relationship between the training data and the label, the appropriate layer type is selected. While the CNN structure gives much better results in problems where the input is a training set of images, RNN-based structures are more effective in problems where the input is tightly dependent on time. In this study, the data aimed to create a prediction model is very time-dependent, so deep learning models with RNN-based memory cell structure are selected. The neural network structures having memory cells, which form the infrastructure of the models, are expressed respectively.

2.4.1 Recurrent neural networks

In some cases, the correlation between input and output data may not be readily apparent. Rather, it may be based on past and future links, thus necessitating a prediction model with memory. Deep learning methods, as described by LeCun [88], meet this requirement. RNNs, one such deep learning method, enhance the predictive power of the model through the use of memory cells. This method is briefly outlined in this section.

In contrast to ANN, RNN derive results not only from current inputs, but also from previous inputs. As illustrated in Figure 2.4, in addition to the input data at time *t*, the hidden layer at time *t-1* also serves as input to the hidden layer at time *t*. The decision made for the input at time *t-1* affects the decision made at time *t*. In other words, these networks generate output by combining current and past information.

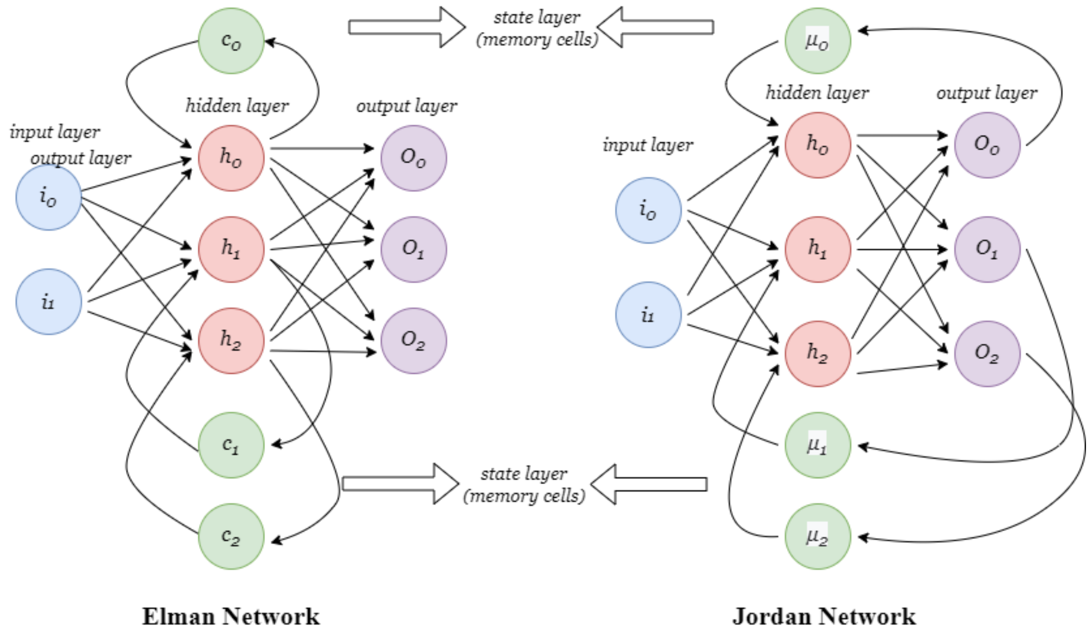


Figure 2.4 : Elman network vs Jordan network in RNN.

Although the first examples of this recurrent structure are introduced by John Hopfield in 1982 [89], the Elman network proposed by Elman in 1990 [90] and the Jordan network introduced by Michael L. Jordan can be said to be the most widely used RNN structures [91]. These two networks are given in Figure 2.4. While the output of each node in the hidden layer in the Elman network is an additional input to itself, the outputs of the result nodes in the Jordan network provide inputs to the nodes in the hidden layer in addition to the input layer. RNN has two main problems to be solved.

- *Vanishing gradients*: In a long sequential array, the gradients are multiplied by a (transposed) weight matrix at each time step. If there are small values in the weight matrix, the norms of the gradients get smaller and smaller, exponentially.
- *Exploding gradients*: If we have a large weight matrix and the non-linearity in the iterative layer is not saturated, the gradients explode. The weights will diverge at each update step. We may need to use a very small learning rate for gradient descent to work.

One of the reasons to use RNNs is the advantage of remembering past information. However, without applying some tricks, RNNs may not be able to remember information from long ago. Using the outputs as inputs at the next time t has given this structure a memory, which is very advantageous in time series prediction problems. How powerful this deep learning framework RNN is explained in detail in [92].

2.4.2 Long short-term memory

Based on RNN structures, LSTM was developed by Hochreiter et al. (1997) to solve the vanishing gradient problem [93]. LSTM, which is later modified and popularized with the contribution of many people, now has a wide range of uses. LSTM preserves the error value from different times and layers in backpropagation. By providing a more constant error value, it ensures that the learning steps of iterative networks can continue. It does this by opening a new channel between cause and effect. The difference of the repeating module in the LSTM structure is that instead of a single neural network layer, there are 4 layers that are connected in a special way. These layers are also called gates. It is a structure that receives information outside of the normal flow. This information can be stored, written to the cell, and read. The LSTM neuron's features are given in Figure 2.5.

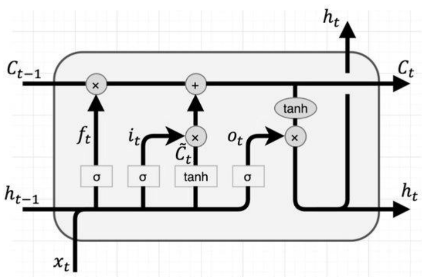


Figure 2.5 : LSTM cell structure.

When it is looked at this figure, c(t-1) represents the long-term memory line from the past, and h(t-1) is the output value of the previous cell. Here, the current input is shown as x(t) and it is decided whether this input will be forgotten or stored in long-term memory with a forget gate and a sigmoid function. When it is examined at the cell outputs, the *c(t)* value sends the current long-term memory output, and *h(t)* sends the current output value to the next cell. As a result, a bond is formed between the past and the present. Equations showing the relations within this cell structure are given below.

$$f_t = \sigma_g\left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{2.28}$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{2.29}$$
$$o_t = \sigma_o(W_o x_t + U_o h_{t-1} + b_o) \tag{2.30}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{2.31}$$
$$h_t = o_t \odot \tanh(c_t) \tag{2.32}$$

Where $\odot$ denotes the Hadamard product, $x_t \in R^a$ is one of the inputs of the LSTM unit, $f_t \in R^h$ is the activation vector of the forget gate, $i_t \in R^h$ is the activation vector of the input/update gate, $o_t \in R^h$ displays the hidden state vector (as well as the output of that unit) and $c_t \in R^h$ represents the state vector of the cell. An LSTM unit uses the cell state c_t to transmit information within the unit. It regulates whether the information is preserved in the cell state using structures called gates. The forget gate looks at the current input and the previous hidden state f_t decides how much information we want to keep from the previous cell state c_{t-1} and expresses it with a value between 0 and 1 with the coefficient c_{t-1}. Moreover, $tanh(W_c x_t + U_c h_{t-1} + b_c)$ calculates a candidate to update cell state, and as with forget gate, input gate $\mathbf{i_t}$ determines how many updates to apply. Finally, output h_t is calculated based on cell state c_t, but then passed through *tanh* and filtered by output port o_t.

The complexity of the problem determines the number of cells to be used in the LSTM structure to be constructed. While complex problems require a large number of cells, fewer LSTM cells are sufficient for easy-to-learn problems. In this thesis, this method has been used for both dynamic price prediction problem and predicting the demand for electricity load from the microgrid. The LSTM structure is constructed according to the sophisticated structure of the problems.

2.4.3 Bidirectional LSTM

Bi-LSTM, which is an abbreviation of the name Bi-directional long short-term memory, is basically a network structure that combines two RNN structures [94]. The difference in this deep learning method, which is the enhanced version of the bidirectional RNN structure with LSTM cells, is to look and evaluate both the front and the back of the data in this network structure.

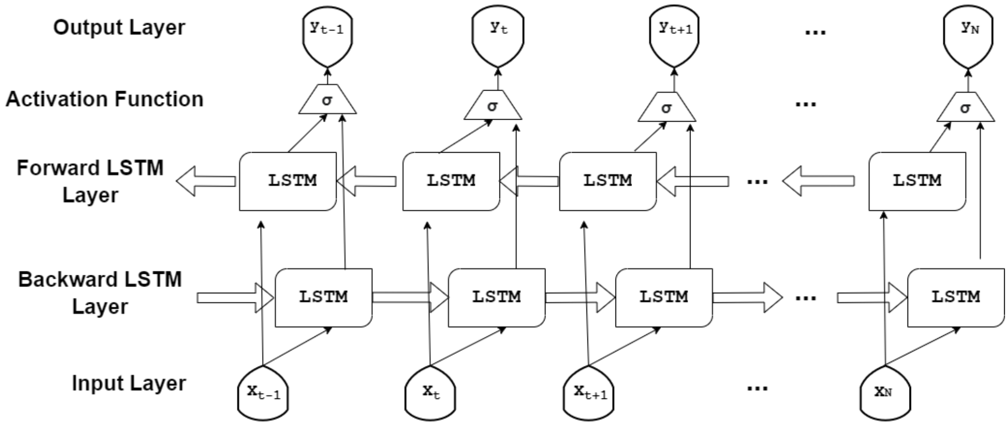


Figure 2.6 : Overview of Bi-LSTM.

This type of deep learning structure works well if the output we want to predict in a prediction problem depends not only on past inputs but also on future data. In Figure 2.6., the Bi-LSTM structure is depicted. As can be seen in the figure, the input data is first inserted into the forward and backward LSTM layers. In this way, the results from both past and future data are given as input to the activation function together. Activation layer outputs form the final outputs. In areas such as natural language processing, the Bi-LSTM structure gives very good results with this bidirectional evaluation feature. The production amounts of renewable energy sources throughout the country have annual similarities. Considering this situation, this deep learning structure has been used for both solar power prediction, wind power prediction and forecasting the energy production of renewable energy sources.

2.4.4 Gated recurrent units

Numerous ANN variations have been developed to address the issue of Vanishing-Exploding gradients frequently encountered in basic RNNs. One of the most prominent variations is the LSTM, while another lesser known but equally effective variation is the Gated Recurrent Unit (GRU) network. GRU was proposed by Cho et al. [95], who compared the performance of RNN, LSTM, and their approach GRU. GRU is actually a simplified version of LSTM, which was developed earlier. Figure 2.7 depicts a GRU cell, which as demonstrated, only consists of three gates and does not maintain an internal cell state, unlike LSTM in Figure 2.5.

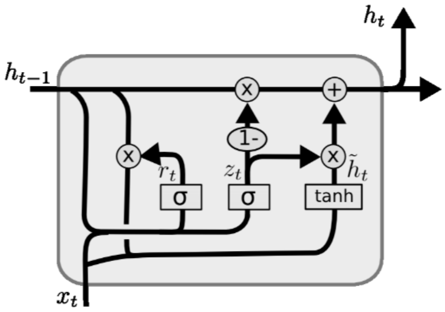


Figure 2.7 : Cell structure of GRU.

The correlation equations containing the variables in this cell structure are provided below.

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \tag{2.33}$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \tag{2.34}$$
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{2.35}$$

Where, x_t is the input vector, h_t is the output vector, r_t is the reset gate vector, σ is the Sigmoid activation function, ϕ is the hyperbolic tanh activation function, and W, U, b are the learning parameters.

To be specific, z_t is a gate vector that determines how much of the past information should be transferred to the future. Applies the sigmoid function to the sum of the two linear layers, biasing the input x_t and the past h_{t-1} state. As a result of applying the sigmoid, z_t contains coefficients between 0 and 1. The final output state is a linear combination of the expression $\phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$ obtained with h_t , h_{t-1} and z_t. If the coefficient is 1, the output of the current volume is a copy of the previous state and is ignored (by default behavior). If it is less than 1, some of the new information from the input is taken into account. The reset gate r_t determines how much of the past information should be forgotten. In the new memory content, if the coefficient $\phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$, r_t is zero, the information from the past is not used. Also, if z_t is 0, the whole system is completely reset since h_t will only look at the input.

This structure, which is based on RNN and provides solutions to the basic problems that arise in time related prediction or classification problems. GRU is one of the preferred DL methods in this thesis, where it is utilized to predict dynamic price and load demand in less than one hour frequency. The performance outputs of the four methods involving the GRU are presented comparatively in chapter four.

3. REINFORCEMENT LEARNING FUNDAMENTALS

3.1 Introduction

The designed AIEMS requires a sophisticated controller to cope with controlling the microgrid in addition to the human-based management. In this regard, this thesis proposes a controller based on artificial intelligence. Nowadays, data-driven control methods that replace classical control methods are among the popular topics. This thesis focuses on such machine learning based control methods. In this context, reinforcement learning, one of the three main branches of machine learning, is investigated and its foundations are given. Reinforcement learning is the inclusive name given to methods based on the principle of controlling the system without the need for a mathematical model of the system. This chapter describes the theoretical and mathematical principles of the AI-based control agents designed in this thesis.

3.2 Methodology

The concept of Reinforcement Learning (RL), which is one of the three basic building blocks of machine learning, offers us a different framework for the control of systems from the classical control methods. Thanks to this concept, suitable control solutions are sought for problems that could have discrete or continuous states. In today's world, both RL algorithms and the problems and solutions in which these algorithms are applied are in a continuous development. This concept has come to the fore as a structure that has been studied academically intensively in the last couple of years and that up-to-date academic results are obtained day by day.

3.2.1 Markov decision process

Markov decision process (MDP) is a discrete-time stochastic control process. It provides a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker. MDPs are useful for studying optimization problems solved via dynamic programming and Monte Carlo. Accordingly, MDPs, more specifically Finite MDPs, form the basis of RL. MDPs are based on a framework of determining an action according to the outcome they receive (as a reward, can be positive or negative) by interacting with an environmental model. This solution framework is expressed in Figure 3.1.

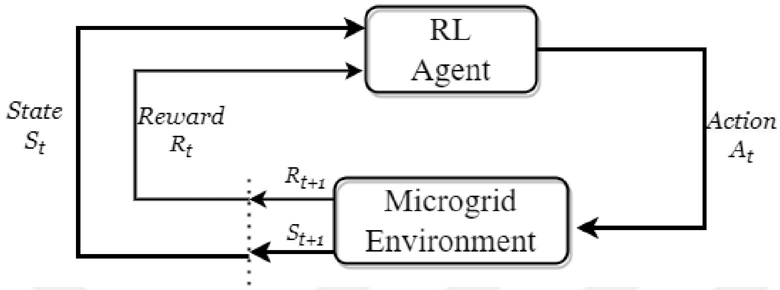


Figure 3.1 : Markov Decision Process Framework for RL.

As can be seen in Figure 3.1, RL framework consists of an agent and environment as well as action A_t, current state of state space S_t, and reward R_t.

- *Agent:* It is the general name given to the algorithm used, the agent interacts with the environment and learns and develops according to the results.
- *Environment:* It is the name given to system model in which the agent interacts, this environment sends reward (or punishment) and next state information according to the actions chosen and implemented by the agent.
- *State:* RL algorithms work by analyzing and learning discrete and continuous states. States are the information obtained by the agent as the result of each action, denoted by s. State space is the set containing all states. And these states can be continuous or discrete. The state space is denoted by S_t and current state $s \in S_t$.
- *Action:* The name of the action performed by the agent in the environment. Actions are denoted by a and are part of an action space (set). In RL, actions are usually determined as discrete set. Current action is denoted as $a \in A_t$ and next action as a'.
- *Observation:* It is the information obtained from the environment as a result of the action, which can be generalized as the next state $s'$and reward r info.

- *Reward:* Although the concept of reward has a positive meaning in many languages, reward can be both positive and negative in RL. The reward is the information obtained by the agent as the result of the action chosen and applied. Current reward is denoted by $r \in R_t$as the result of action performed.
- *Return:* This concept refers to the sum of the rewards that the agent collects after performing a series of actions and interacting with it. This series can be several actions performs and results, as well as the rewards it collects during an episode. It is denoted by G_t.
- *Policy:* This concept is the name of the structure that the agent uses to determine the action to be taken. This is a principle function, which is denoted by π if stochastic and μ if deterministic.
- *Episode:* It is the time (or series) taken by the agent to reach the target as the result of its actions.

Trajectory: A data array in which the 'state, action, next state, reward, ending information' that the agent receives from the environment during one or specified number of episodes is kept sequentially. Some algorithms need this structure, while others do not. it is denoted by τ.

The aim of RL is to improve the policy used when choosing actions and to choose the most appropriate actions according to the current situation. At this point, the methods of updating the policy function are divided into two. These:

- On-Policy Algorithms: There is only one policy function which is used to choose an action when learning stage.
- Off-Policy Algorithms: Two different policy functions are used. One for choosing random action when learning process and second is updating due to the learning stage.

In addition, RL algorithms are divided into model-based (model-based) and non-model-based (model-free) algorithms according to whether the environment model is known or not. The model should be well prepared to give good results for the algorithm to be used.

In Finite MDPs, the agent interacts with the environment at different time steps. At each time step t, the agent receives the state s_t information from the environment and according to the policy, it selects an action a_t from the action space A_t and switches to a new state s_{t+1}. The policy function π maps from state s to action a and specifies which action should be taken. The state transition takes place with the probability of $\mathrm{P}(\mathrm{s}_{t+1}, \mathrm{r}|\mathrm{s}_t, \mathrm{a}_t)$ according to the environment or model dynamics, and as a result of the transition, a reward r is received from the environment. The reward r is determined by the reward function, which is defined as $\mathrm{R}(\mathrm{s}, \mathrm{a})$. The reward function could be single function or piecewise function. In episodic and continuous problems, this process continues until the terminal state is reached. Collected total reward namely return is expressed in equation (3.1).

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{3.1}$$

where, γ is a discount rate ($\gamma \in [0,1]$) which makes agent forward rewards and act considering prospected total reward. γ is a hyper-parameter to be choose depending on learning results. In episodic problems, RL agent tries to optimize state values denoted by $\mathrm{V(s)}$ or action-state values denoted by $\mathrm{Q(s, a)}$ using maximized return $\mathrm{G_t}$. And, when state values or action-state values reach optimum values, agent uses policy function to pick a proper action in every state that can be faced using optimum $\mathrm{V}_*(\mathrm{s})$

or $\mathrm{Q}_*(\mathrm{s}, \mathrm{a})$. These optimum functions refer as in equation (3.2).

$$V_*(s) = max_\pi V_\pi(s) \ and \ Q_*(s, a) = max_\pi Q_\pi(s, a) \tag{3.2}$$

The determination of the state value and action value functions is based on Bellman equations and are expressed in equations (3.3) and (3.4).

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{s', r} p(s', r \mid s, a)[r + \gamma v_\pi(s')] \tag{3.3}$$

$$Q_\pi(s, a) = \sum_{s', r} p(s', r \mid s, a) \left[r + \gamma \sum_{a'} \pi(a' \mid s') q_\pi(s', a') \right] \tag{3.4}$$

Equation (3.3) and equation (3.4) say that state value or action-state value are written as the probability taking each action following the policy multiplied by the return, we expect to get taking that action. And, that return can be expressed as the probability of reaching each possible successor state multiplied by the reward obtained upon reaching that state, plus the discounted value of that successor state. There is a recursive relationship between the value of one state and the values of other states. This is very useful for developing algorithms capable of solving control tasks. Solving of a control task consist of maximizing the expected return G_t. The value of a state is precisely that expected return, so solving a task involves maximizing the value of state or maximizing every q-value ($q(s,a)$). The optimal value of a state is the expected return following the optimal policy:

$$v_*(s) = E_{\pi_*}[\,G_t \mid S_t = s\,] \tag{3.5}$$
$$q_*(s,a) = E_{\pi_*}[\,G_t \mid S_t = s, A_t = a\,] \tag{3.6}$$

To maximize those returns, we have to find the optimal policy, which is the policy that takes the optimal actions in all states. On the other hand, the optimal policy is precisely defined as the policy that in each state chooses the action that leads to the maximum expected return. The optimal π_* policy is the one that chooses actions that maximizes $\mathrm{v}(\mathrm{s})$ or $\mathrm{q}(\mathrm{s},\mathrm{a})$ (see equations (3.7) and (3.8))

$$\pi_*(s) = argmax_a \sum_{s',r} p(\,s', r \mid s, a\,)[r + \gamma v_*(s)] \tag{3.7}$$
$$\pi_*(s) = argmax_a q_*(s,a) \tag{3.8}$$

Using the value of the states will consider the states to which the action leads and the return that we expect to obtain in that successor state. Using the q-values, the action will be selected whose q-value is highest. That can be seen in equations (3.7) and (3.8). But it seems to a problem: to find the optimal policy π_*, the optimal values v_* or q_* must be known and to find the optimal values, the optimal policy must be learned. It seems like a chicken and egg dilemma. To obtain these two elements that one depends on the other, it should be looked back Bellman equations which could be seen on equations (3.3) and (3.4). In equation (3.3), the probability of reaching each successor state by taking the optimal action multiplied by the reward achieved by reaching that state, plus the discounted optimal value of that state. In (3.4), the optimal q-value for an action in a state is the weighted sum of the returns obtained upon reaching each ot the possible successor states weighted by the probability of reaching that successor state. Return in that equation defined as the reward achieved upon reaching the successor state plus the maximum q-value among the actions for that next state. In next sections, the solutions of this recursive equations are given under the methods of Dynamic Programming and Monte Carlo, and also Temporal-Difference Learning.

3.2.2 Dynamic programming and monte carlo

Dynamic programming is the family of methods that find the solution to a problem by breaking it down into smaller, easier-to-solve problems. In RL, the problem that the agent is trying to solve finding the optimal policy. Applying dynamic programming to a problem, the problem has to have two properties. The first one is an optimal substructure which is that in finding the solution to each of its subproblems and joining those individual solutions, the optimal solution to the original problem will have found. Finding the optimal policy for all states can divide into finding the optimal policy for every state individually. If we find the optimal policy for each of the states, the optimal policy for the entire problem will be obtained. The optimal policy chooses actions based on state or q-values. Therefore, to obtain the optimal policy, the optimal values need to be achieved. If we find the optimal value for each state independently, then the optimal value function will be attained for the overall problem. The second property is to contain overlapping subproblems. The solution to the subproblems are mutually dependent on each other. Bellman equations are expressed the value of a state in terms of the values of other states or in the case of the q-function, the value of an action in a state as a function of the value of other actions in other states. This satisfies the second required property. With dynamic programming, we will keep a value table with and entry for each state. And in that value table, we are going to maintain an estimate of the value of each state. That estimates at the beginning don't have to be accurate, but the agent is going to improve them in an iterative manner. To do that, Bellman equations are going to be converted into an update rule. And with it, the agent is going to sweep the state space and update the estimated value of each state according to this update rule. Each time of updating the estimated value of a state, the agent will have better estimates for the related values. And therefore, new estimate will be more accurate than the old one. In dynamic programming, to perform the update, the agent needs to know in advance the state transition probabilities. One big limitation of dynamic programming is that the agent needs a perfect model of the environment. Thus, the agent need access to these state transition probabilities. However, in a large number of the tasks, that model won't be available, and the agent should use other algorithms to solve them. Dynamic programming solves problems using expected values, not trial and error, so it takes into account every possible outcome of taking an action and uses it to update the estimated values. To start with the family, value iteration is one of the methods in dynamic programming, which is based on updating state values using the Bellman update rule. Second important method in dynamic programming is policy iteration. This method serves as inspiration to design the vast majority of RL. The algorithm finds the optimal policy and its value function by alternating two processes. Here, the policy and the value function with arbitrary values are initialized and then, it will start by evaluating the value functions. Then, depending on evaluated values, the agent will improve the policy. Then, the algorithm keeps evaluation and improvement until both reach the optimal form. These two processes are in competition with each other, but they also collaborate pushing each other to the optimal solution.

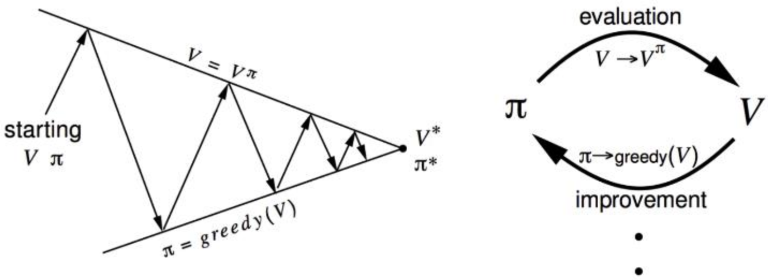


Figure 3.2 : Generalized policy iteration (GPI).

GPI on Figure 3.2 is the general template that most RL algorithms follow. Unlike dynamic programming methods, Monte Carlo methods or temporal difference methods do not require a full environment model and improve themselves by interacting with the environment and learning by trial and error. However, even these methods use generalized policy iteration.

Monte Carlo methods are the family of algorithms that learn based on experience. The family of methods learn the optimal state values or q-values based on samples of experience collected by the agent while interacting with the environment. At the beginning of the learning process, the agent follows an arbitrary policy. Then the agent tries to perform the task using that policy until the end of the episode generating trajectory. At the end of the episode, the return from every state visited are computed.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[\, G_t - Q(S_t, A_t)\,] \tag{3.9}$$

Update rule of Monte Carlo method could be seen in equation (3.9). Every time that the agent observes a new return for a specific state, it updates the estimated value for that state (q-values) as the average of all the returns that the agent has collected and that start in that state. By the law of large numbers, the more returns the agent observe for the state or action, the more it approaches it expected value. So, more experience makes more accurate the q-value estimates. Therefore, Monte Carlo methods have some advantages over dynamic programming. First, the estimate of one state does not depend on the rest. Hence, complexity of estimating the value of a state doesn't depend on the number of states in the task. In the case of dynamic programming to estimate the value of a state, the algorithm bootstraps the value of other states. That is used one estimate to produce another estimate. Therefore, complexity of the algorithms grows exponentially with the number of states. Another advantage is that the agent can focus its efforts on correctly estimating the value of the states that lead to the goal. Dynamic programming sweeps through the state space and updates every single state whether they are important or not. Monte Carlo methods use the optimal policy which is based on maximum q-values. Since, the algorithm crates Q value table that keeps all the q-values of the state space. Then, using GPI, it improves the policy with evaluating q-values in that table.

$$\pi_0 \longrightarrow Q_{\pi_0} \longrightarrow \pi_1 \longrightarrow Q_{\pi_1} \longrightarrow \cdots \longrightarrow Q_{\pi_*} \longrightarrow \pi_*$$

The experience that the agent collects depends on the actions that it takes, and those actions depend on the policy that the agent is using at that time. If action a is optimal but has a bad estimate $Q(s, a)$, the agent will never select it. The only way to avoid this is to explore all actions every once in a while and update their estimate $Q(s, a)$. To maintaining exploration, the policy should have a probability of choosing every action greater that 0. This ensures that from time to time, the agent takes an action which isn't considered optimal to improve its understanding the task. So, with probability ϵ, the agent selects a random action and $1 - \epsilon$, it selects the action with the highest $Q(s, a)$. This process is called $\epsilon - greedy$ strategy.

3.2.3 Temporal-difference learning

Temporal-Difference (TD) learning refers a family of methods that learn the optimal $v_*(s)$ or $q_*(s, a)$ values based on experience. These methods are the basis of many advanced algorithms. TD learning is a combination features of Monte Carlo methods and features of dynamic programming. As in Monte Carlo methods, the agent faces the environment and generating a trajectory with the states visited, the actions taken and the rewards obtained at the end of the episode and at the end, it uses that experience to update q-value estimates and the policy. Further, as in Monte Carlo methods, the agent does not have a model of the environment dynamics. Since it does not have a model, it uses q-values to guide the policy because the q-values implicitly estimate environment dynamics. In that way, the policy will simply choose the action with highest q-value. On the other hand, as in dynamic programming, the agent uses bootstrapping. It relies on estimates of q-values to produce new estimates that makes more accurate.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t \; + \; \alpha[\; R\_\{t+1\} \; + \; \gamma\, Q(S_{t+1}, A_{t+1}) \; - \; Q(S_t, A_t))] \qquad (3.10)$$

As can be seen on equation (3.10) (Update rule for State–action–reward–state–action (SARSA) algorithm which is one of TD learning methods), TD learning uses bootstrapping to update. TD learning also follows GPI process. While Monte Carlo methods wait until the return G_t is available before updating $Q(s, a)$, TD methods perform the cycle of policy evaluation and improvement every time that an action is taken during the episode without waiting until the end. In this way, learning process is constant and uniform. That is a big advantage because the learning that the agent does at the beginning of the episodes influences the policy during the rest of the episode, improving its decision making. In TD learning, q-value estimates for each S_t, A_t are keeping a table called q-table. This table is being updated with every outcome of action performing. α on the update rule determines learning speed. It is a hyper parameter that should be chosen due to the task's specifications. TD methods are divided into two parts: on-policy and off-policy algorithms. In TD, whereas SARSA uses on-policy learning, Q-Learning (QL) algorithm uses off-policy learning which means there are two different policies that one for optimizing policy and one for exploration.

$$Q(S, A) \leftarrow Q(S, A) + \alpha[R_{t+1} + \gamma max_a Q(S', a) - Q(S, A))] \tag{3.11}$$

Update rule of QL on equation (3.12). QL directly learns optimal policy, and it doesn't need next action that the policy chooses. So, it can be said that QL is a more aggressive agent, while SARSA is more conservative. Besides SARSA and QL, Expected SARSA is also one of TD methods, which has different update rule from others.

3.2.4 *n*-Step bootstrapping

A family of algorithms that lie between Monte Carlo and TD methods is called n-Step TD methods. These algorithms learn based on experience and use a technique known as n-Step Bootstrapping. In SARSA, every time that the agent updates a q-value estimate, it pushes the current estimate in the direction of the target $R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$ (update target) in an amount to proportional α. And this is a bootstrapping. This target is the reward obtained after taking the action at time 't' plus the estimated q-value of the next state and the action chosen in that next state. And q-value is the expectation of future rewards from taking an action. The advantage of using an estimate to update another estimate is that the agent doesn't need to wait until the end of the episode to obtain the remaining rewards because the agent uses an estimate to replace them. In SARSA, the agent is performing one step bootstrapping because it uses one actual reward, and it estimates the rest. So, the agent is applying its estimate one step in the future [10]. But the agent could have also taken another action obtained another actual reward, interacting with the environment and estimated the rest by replacing them with the q-value estimate of the state and action chosen two steps into the future. That is called 2-step bootstrapping $\mathrm{R_{t+1}} + \gamma\mathrm{R_{t+2}} + \gamma^2\mathrm{Q}(\mathrm{S_{t+2}}, \mathrm{A_{t+2}})$. Or even three rewards and estimate, 3-step bootstrapping: $\mathrm{R_{t+1}} + \gamma\mathrm{R_{t+2}} + \gamma^2\mathrm{R_{t+3}} + \gamma^3\mathrm{Q}(\mathrm{S_{t+3}}, \mathrm{A_{t+3}})$. Or n-step in the future as called as n-step bootstrapping:

$$R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n Q(S_{t+n}, A_{t+n}) \tag{3.12}$$

All the expression are valid estimates of the return of the episode, the difference is how many actual rewards they include and how many the agent estimates using the q-values.

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n Q(S_{t+n}, A_{t+n}) \tag{3.13}$$

Now, the agent can use n-step return to update its estimates. The update rule:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[G_{t:t+n} - Q(S_t, A_t))] \tag{3.14}$$

Using these methods, the agent must wait 'n' steps into the future to update the q-value estimate of the present state. In this method, value of n is a hyper parameter and arbitrarily chosen. When making an estimate of a value, there are two problems that can negatively affect these estimates and trying to mitigate one tends to make the other worse. The first problem that can arise when the agent tries to estimate a quantity is bias. This problem occurs when our estimates are systematically away from actual quantity. The second problem is the variance of the estimate. This problem occurs when the estimates are very different from one to another, although on average they are aiming at the right quantity. And these problems are not mutually exclusive. They could happen at the same time. One step bootstrapping method such as SARSA has high bias. So, it could be said that the higher n value lowers the bias. On the other hand, the problem with variance arises for other reasons. As in Monte Carlo methods, using whole return to update estimated values could cause higher variance. For that reason, the larger n makes greater the variance. Determining value of n is vital important for exchanging bias to variance. In practice, intermediate values of *n* achieve better results than extreme values.

3.3 Deep Reinforcement Learning

Deep reinforcement learning (DRL) is a type of machine learning in which an agent learns to make decisions in a complex environment by trial-and-error learning, using feedback signals from the environment. It involves a combination of two powerful techniques: deep learning and reinforcement learning. SARSA, Q-Learning and n-Step Bootstrapping methods, which are temporal-difference methods, create a table with all possible outputs and all control actions of the state to be controlled. These methods work well in control problems where possible states are limited, and control actions are countable (discrete) if high computational power is available. However, in control problems with continuous states (infinite possible states) and/or continuous control actions (infinite possible control actions), these TD methods are insufficient. If the state-action values are obtained as an approximate result instead of obtaining the actual value from the Q table, which takes its final form at the end of thousands of episodes, the problems in which this table cannot be created can be overcome. DRL is the general name given to methods based on approximately calculating state-action values $Q(s_t, a_t)$ or directly approximating the outputs of the policy function $\pi(s_t)$. Because these approximate calculations are made using deep neural networks, it is called deep RL. In DRL, the inputs are states, and the outputs are the Q values or the outputs of the policy function. The neural network (NN) structure, which makes approximate calculations in DRL, improves, and optimizes itself as a result of each episode. In this regard, the loss function appears as a special version of the MSE function. The cost function is calculated using equation 3.15.

$$L(\theta) = \frac{1}{|\mathrm{K}|} \sum_{\mathrm{i}=1}^{\mathrm{K}} \left[\mathrm{R_i} + \gamma \hat{\mathrm{q}}(\mathrm{S_i}', \mathrm{A_i}' | \, \theta_{\mathrm{targ}}) - \hat{\mathrm{q}}(\mathrm{S_i}, \mathrm{A_i} | \, \theta) \right]^2 \tag{3.15}$$

Where, target is $y_i = \mathrm{R_i} + \gamma \hat{\mathrm{q}}(\mathrm{S_i}', \mathrm{A_i}' | \, \theta_{\mathrm{targ}})$ and estimate is $\hat{y}_i = \hat{\mathrm{q}}(\mathrm{S_i}, \mathrm{A_i} | \, \theta)$ in MSE. And K in the equation is batch number from experience. NN optimize itself using experiences from past episodes. Moreover, θ represents parameters of NN in DRL. The ability to calculate the loss function allows the calculation of gradients. Calculation of gradients is given in equation 3.16.

$$\nabla L(\theta) = \left[\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \dots, \frac{\partial L}{\partial w_n} \right] \tag{3.16}$$

Once the gradients are calculated and the gradient vector is created, the NN parameters are reduced with this vector and learning rate α. This reduction is called *Stochastic Gradient Descent* (SDG) and is shown in equation 3.17.

$$\theta \leftarrow \theta - \alpha \nabla L(\theta) \tag{3.17}$$

Update rule is rather like tabular methods. However, instead of modifying the Q table in TD methods, the parameters of NN are being modified using SDG. Therefore, SDG is crucial for learning how to choose appropriate control actions. On the other hand, the accuracy of the approximations used in DRL is possible with a well-trained NN. While NN is at its optimum, it must overcome the problems of memorization and over-learning (over-fitting). These two undesirable situations are overcome by using a memory structure called *'replay memory*' or *'experience replay*'. Experience replay is a technique used in RL to improve the efficiency and stability of the learning process. The basic idea is to store past experiences (or "samples") of the agent's interaction with the environment in a replay buffer, and then use these experiences to train the agent's policy and/or value function. The replay buffer is a fixed-size queue or circular buffer that stores transitions *(s, a, r, s')* observed by the agent during its interaction with the environment. Each transition corresponds to a state-action pair *(s, a)* taken by the agent in the environment, along with the resulting reward r and the next state s'. During training, the agent randomly samples a batch of transitions from the replay buffer and uses them to update its policy and/or value function. By randomly sampling transitions from the buffer instead of using them in the order they were observed, the agent can break the correlations between successive samples and reduce the variance of its updates. This has several benefits for the RL agent:

- Experience replay allows the agent to learn from a diverse set of experiences, which can help it avoid getting stuck in local optima and learn more robust policies.
- By reusing past experiences, the agent can learn more efficiently and effectively from its experience, since it can learn from each experience multiple times.
- Using a replay buffer can help stabilize the learning process by reducing the variance of the updates and preventing the agent from overfitting to recent experiences.

While all DRL methods use this memory structure, DRL methods differ depending on the update rule they use or what they approximate (state-action pair (*Q(s,a)*), direct policy function (*π(s)*), or both).

In DRL, the basic idea is to use deep neural networks to represent the agent's policy and/or value function, allowing the agent to learn more complex and abstract representations of the environment and make better decisions.

The key difference between traditional RL and DRL is that DRL uses deep neural networks to approximate the agent's policy and/or value function, instead of explicitly storing or computing them. This allows the agent to learn more complex and abstract representations of the environment, and make better decisions based on these representations. There are several popular DRL algorithms, including:

- Deep Q-Networks (DQN): This is a popular DRL algorithm that uses a deep neural network to approximate the agent's Q-function, which represents the expected cumulative reward for each action in each state. DQN uses a replay buffer to store past experiences and reduce the variance of the updates [96].

- Policy Gradient Methods: These are a class of DRL algorithms that directly optimize the agent's policy using gradient-based optimization techniques. Examples include REINFORCE, Actor-Critic, and Trust Region Policy Optimization (TRPO).

- Deep Deterministic Policy Gradient (DDPG): This is a DRL algorithm that combines Q-learning and policy gradient methods to learn a deterministic policy that maps states to actions. DDPG uses a replay buffer and target networks to stabilize the learning process [97].

- Proximal Policy Optimization (PPO): This is a recent DRL algorithm that uses a trust region optimization approach to update the agent's policy. PPO is known for its simplicity and robustness and has been used in a wide range of applications [98].

In this thesis, DQN-based DRL algorithms are created and trained specific to the energy management system.

3.4 Double Deep Q Networks

Double Deep Q Networks (DDQN) is an extension of the DQN algorithm, which was proposed by Hasselt et al. in 2015 [99]. The main idea behind DDQN is to address the problem of overestimation of Q-values in the original DQN algorithm.

In the DQN algorithm, the Q-values are estimated using a neural network that is trained using a variant of the Bellman equation. However, when the neural network is used to estimate the Q-values, it can sometimes overestimate the values due to the max operator used in the update equation. This can lead to unstable training and suboptimal policies.

To address this issue, DDQN introduces a second neural network that is used to select the actions in the update equation. Specifically, the second network is used to evaluate the Q-values of the actions that the primary network selects, and the maximum Q-value from the second network is used to update the primary network. The update equation for the DDQN algorithm is as follows (see equation 3.18):

$$y_t^{DDQN} = R_{t+1} + \gamma Q(S_{t+1}, argmax_a Q(S_{t+1}, a\,; \theta_t\,), \theta_t^-) \tag{3.18}$$

where θ_t is parameters of the primary neural network, θ_t^- is parameters of the secondary neural network (also called the target network), S_t is the current state, a is the selected action, S_{t+1} is the next state, R_{t+1} is the reward, γ is the discount factor, and $argmax_a Q(S_{t+1}, a)$ is the action that maximizes the Q-value according to the primary network.

In the DDQN algorithm, which has a different y label calculation method, the calculation method of the loss function differs depending on this approach. The loss function of DDQN is given in equation 3.19.

$$L(\theta) = \frac{1}{|K|} \sum_{i=1}^{K} \left[R_{t+1} + \gamma Q(S_{t+1}, argmax_a Q(S_{t+1}, a\,; \theta_t), \theta_t^-) - \hat{Q}(S_t, A_t | \,\theta_t)\,\right]^2 \tag{3.19}$$

In the DDQN algorithm, the weights of the target network are updated periodically using the weights of the primary network. This helps to stabilize the training and prevent overestimation of the Q-values. An overview of the DDQN algorithm is given in Figure 3.3.

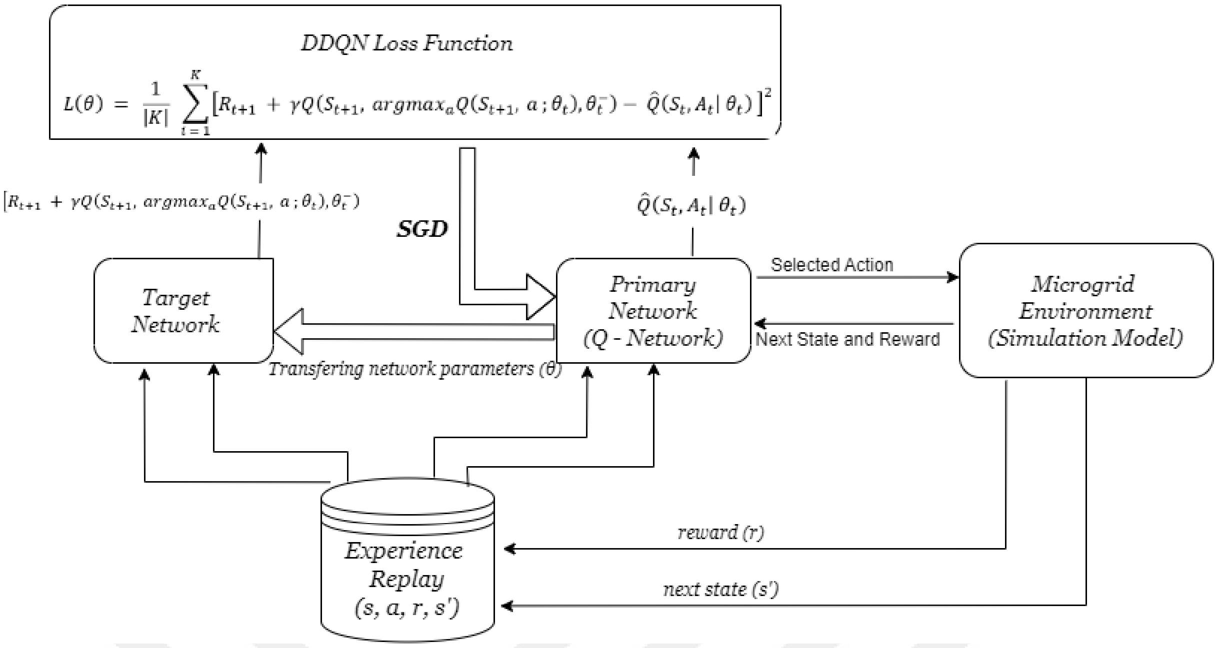


Figure 3.3 : Overview of DDQN algorithm (SGD: Stochastic Gradient Descent).

To summarize, DDQN is an extension of the DQN algorithm that addresses the problem of overestimation of Q-values by introducing a secondary neural network to select the actions in the update equation. By using the maximum Q-value from the secondary network, DDQN can improve the stability and performance of the DQN algorithm.

4. PROPOSED METHOD AND SIMULATION RESULTS

4.1 Introduction

The most vital component of the targeted energy management system is the decision-making AI agent and accurate predictions of the outputs of the microgrid that this agent uses to make its decisions correctly. The fundamentals of the ML and DL based prediction models created for this thesis are presented in the second chapter. Furthermore, in the previous section, the theoretical and mathematical foundations of the AI-controller agents employed in the energy management system were elucidated. These models and agents are gathered under AIEMS framework.

The proposed AIEMS methodology consists of three main sections (see Figure 4.1). In the first section, prediction models required for AIEMS are developed and trained. Once sufficient accuracy results are obtained, the models are stored. In the subsequent second section, three simulation models with different dynamics for the microgrid are created. These environments utilize the trained prediction models from the previous section. In this section, AI-based controllers are also designed and trained with these simulation models. The AI controllers that have successfully completed the learning process are then used in the third section, which is the interface. The GUI developed in the third section utilizes all the prediction models and presents the results obtained from these models along with real-time variables specific to the microgrid to the user. The flowchart of this entire process is provided in Figure 4.1.

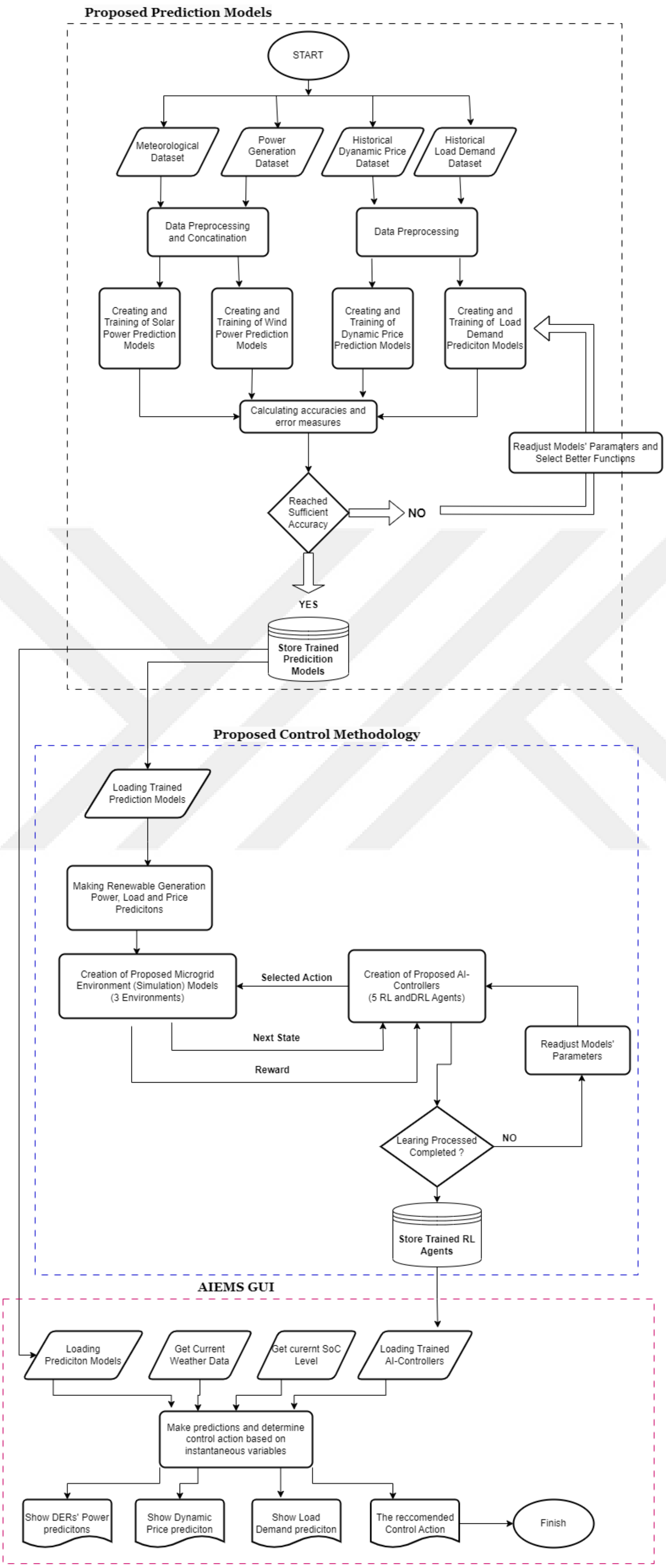


Figure 4.1 : The flowchart of proposed AIEMS

All methods are brought together in the designed graphical user interface and the energy management system of the microgrid is created for the user. In this section, the features, constituents, and offerings of the designed GUI are given.

4.2 Microgrid Environment Design

4.2.1 Features of microgrid simulation model

The utilization of the AI-based control agent that the targeted energy management system will use to optimize itself is possible with a well-prepared simulation model. As described in section three, the two fundamental components of RL, 'agent' and 'environment', emerge as factors that require careful design. The 'environment' mentioned here is the simulation model of the microgrid. The sizes of many microgrids used have been investigated, and the sizes of suitable microgrid components for a university campus or similar facility are determined on average. With the determined sizes, a simulation model is created, which is used both in the training of RL agents and in the interface of the energy management system. It could be claimed that one of the primary benefits of AIEMS is its profitability, particularly in countries where energy prices are subject to dynamic fluctuations. The hypothetical microgrid presented in this thesis is assumed to be situated in Germany, where the average energy price is currently 0.3279 €/kWh. Dynamic pricing is determined through a mathematical calculation of day-ahead prices data on the ENTSO-e platform relative to the average energy price in Germany. Although not all microgrids possess such a feature, the modeled microgrid in this study includes an energy storage unit with a capacity of 130 kWh, irrespective of the battery type used. This energy storage unit is a crucial tool in ensuring the sustained supply of energy demanded from the microgrid. Additionally, the simulated microgrid comprises renewable energy sources that generate 90 kW AC active power, comprising 20 kW wind and 70 kW solar power plants. The microgrid can purchase or sell energy by connecting to the utility grid as necessary. A back-up diesel generator with 7 kW AC active power is also included. Moreover, various loads require around 500 MWh of energy from the microgrid annually. The size and components of the simulated microgrid are depicted in Figure 4.2.

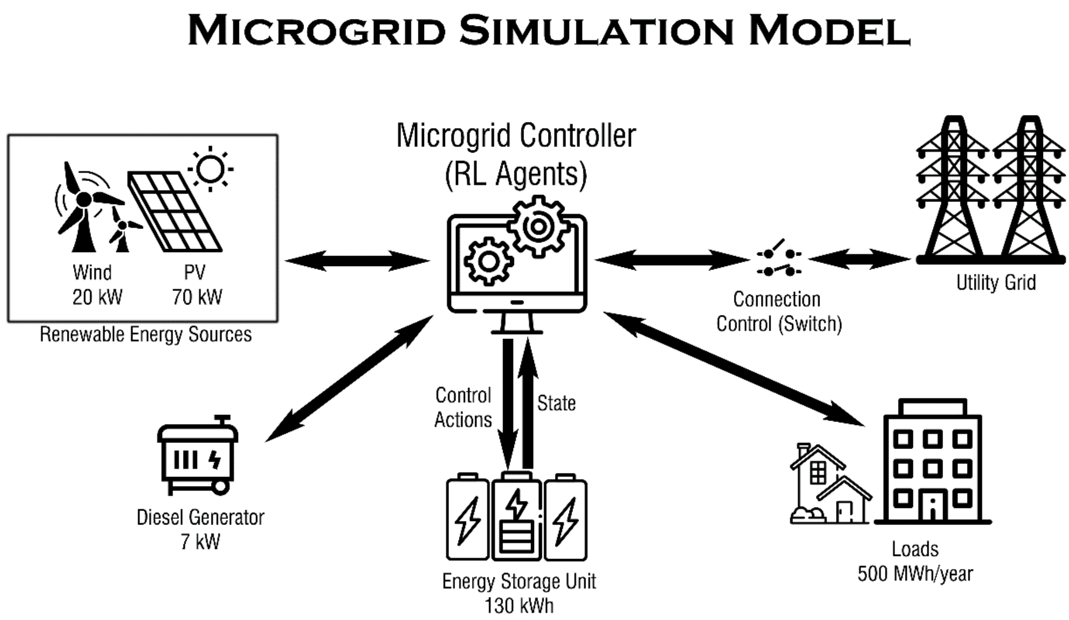


Figure 4.2 : Overview of the microgrid's simulation model

In the figure presented, the main controller RL agent is positioned in the center, and it is trained on the production and consumption values of the components in the figure, as well as dynamic price data. As can be seen, the agent controls the energy storage component with actions determined based on the performance of the other components. In this thesis, the microgrid simulation model, consisting of components with the sizes described above, which were determined for the training of the designed RL agents, has been diversified with three different action sets. Three microgrid environmental models are designed with three, five, and seven control actions and an associated unique reward strategy. However, seven control actions from these three different environments are proposed in this thesis. As explained in the section where the models are compared, it has been determined that seven actions are superior to other designs. Based on this, the simulation environment model of the energy storage system has been designed to utilize three, five and seven different control actions for the AI agent to optimize itself. Furthermore, a reset function has been created to enable the agent to start over in each episode, a step function that performs one of these seven actions, and a clean function that removes all parameters of the environment when desired. The features of the proposed action set are listed below.

- *Action 0* (a_0): The action chosen when charging or discharging does not benefit the microgrid.

- *Action 1* (a_1): First-level charging: the preferred action when the price is high, on the contrary, the load demand is low, and the renewable resources are overproduction.
- *Action 2* (a_2): This control action chosen for second level of power charging when state of charge (SoC), price, load demand and power generation are appropriate.
- *Action 3* (a_3): Third level charging action. This action is chosen when the profitability is the highest.
- *Action 4* (a_4): The first level discharge action selected when the load demand is high and the SoC value is at the appropriate level.
- *Action 5 (a_5)*: Second discharge action, depending on the SoC level, when there is not enough generation from renewable energy sources, but the load demand is high.
- *Action 6 (a_6)*: It is the action that allows the agent to fully use the energy storage unit in cases where the energy need is high and/or the price is high.

Therefore, the agent has the option to choose between doing nothing, three levels of charging, and three levels of discharging. In addition, in the 5-action environment model, there are two levels of charging and discharging, as well as a doing nothing action. Furthermore, the 3-action environment model only includes charging, discharging, and doing nothing actions.

At the conclusion of the training process, the agent must equip itself with an innovative and problem-specific reward strategy to adequately prepare for any situation. By training itself using the proposed reward and penalty strategy embedded in the step function of the environment model, the agent can learn to select the most suitable control action in each state.

4.2.2 Proposed reward strategy

The most innovative contribution revealed by this study is that it proposes a new and unique rewarding strategy for microgrids. The agent receives a reward (not always positive) as a result of the implementation of each chosen action in the environmental model. In an episode, a reward is received as a result of each action until it reaches the end of the episode. And the sum of these rewards is called the return. The agent learns to choose more appropriate actions based on the results of this reward and feedback. These rewards are determined according to the action chosen at each time t and reward function $\mathrm{R}(s,\mathrm{a})$. However, a penalty value is also determined at the same time t, and this penalty value changes the reward value to positive or negative. When an unfit action selected, the penalty coefficient makes the reward value negative. Conversely, in the same state at another time, the agent chooses the more appropriate action with obtaining better reward. These rewards are determined through a reward function created specifically for this study. Created reward function is given in equation 4.1.

$$R(s_t,a_t)=\begin{cases} -\dfrac{SoC_{goal}-SoC_{t+1}}{100}C(s_t,a_t) & ,if\, a_t=a_0 \\ -\dfrac{P_t-P_{min}}{P_{max}-P_{min}}\dfrac{SoC_{goal}-SoC_{t+1}}{C(s_t,a_t).2.SoC_t} & ,if \quad a_t=a_1 \\ -\dfrac{P_t-P_{min}}{P_{max}-P_{min}}\dfrac{SoC_{goal}-SoC_{t+1}}{C(s_t,a_t).SoC_t} & ,if \quad a_t=a_2 \\ -\dfrac{P_t-P_{min}}{P_{max}-P_{min}}\dfrac{SoC_{goal}-SoC_{t+1}}{C(s_t,a_t).0.5.SoC_t} & ,if \quad a_t=a_3 \\ \dfrac{P_t-P_{min}}{P_{max}-P_{min}}-\dfrac{SoC_{goal}-SoC_{t+1}}{C(s_t,a_t).2.SoC_t} & ,if \quad a_t=a_4 \\ \dfrac{P_t-P_{min}}{P_{max}-P_{min}}-\dfrac{SoC_{goal}-SoC_{t+1}}{C(s_t,a_t).SoC_t} & ,if \quad a_t=a_5 \\ \dfrac{P_t-P_{min}}{P_{max}-P_{min}}-\dfrac{SoC_{goal}-SoC_{t+1}}{C(s_t,a_t).0.5.SoC_t} & ,if \quad a_t=a_6 \end{cases} \tag{4.1}$$

Where:

- a_t is the action chosen at time interval t
- $R(s_t,a_t)$ is the proposed reward function which produces proper reward value for specific state s_t and action chosen at time interval a_t
- SoC_{goal} is the goal or terminal state of charge level of the ESS.
- SoC_t is current value of state of charge and SoC_{t+1} is the next value of *SoC*.
- P_t is power generation level by renewable energy sources at time *t*.
- P_{max} and P_{min} are the highest and lowest power level coming from RES within the microgrid.
- $C(s_t,a_t)$ is the proposed function that determines penalty coefficients for actions respectively.

In this reward function, while the rewards are determined according to the production of renewable energy sources and the *SoC* value, they are decreased or increased by the designed penalty function. Furthermore, in this study, unlike the related works, the penalty coefficients are determined by a specially created function instead of fixed values. The created penalty function is given in equation 4.2.

$$C(s_t, a_t) = \begin{cases} -\dfrac{Pr_t - Pr_{min}}{Pr_{max} - Pr_{min}} & , if \quad a_t = a_0 \\ -\dfrac{Pr_t - Pr_{min}}{Pr_{max} - Pr_{min}} \dfrac{L_t - L_{min}}{L_{max} - L_{min}} & , if \quad a_t = a_1, a_2, a_3 \\ \dfrac{Pr_t - Pr_{min}}{Pr_{max} - Pr_{min}} \dfrac{L_t - L_{min}}{L_{max} - L_{min}} & , if \quad a_t = a_4, a_5, a_6 \end{cases} \tag{4.2}$$

Where:

- $C(s_t, a_t)$ gives the appropriate penalty value for specific state s_t and action chosen a_t and it considers dynamic price and load demands to do so.
- Pr_t is the current price value at time t.
- P_{max} and P_{min} are the highest and lowest price value up to time *t*.
- L_t is the current load demand level from the microgrid at time *t*
- L_{max} and L_{min} are the highest and lowest load demand from the microgrid up to time *t*.

This novel function allows current price and load demand to have a say in the choice of action. Thus, the level of load demand or whether the price is high or cheap changes the penalty coefficients. While the reward value is determined by using the reward function according to the state and action pair at the time t, the penalty coefficient is also determined by using the penalty function according to the same state and action. Whereas the reward function uses the power generation value of renewable energy sources and SoC as the main reward determinant, this reward value is decreased or increased using the penalty function using the current price and load demand data.

In addition to the proposed 7-action environment model and the reward and punishment functions created specifically for this environment, relatively less advanced 3-action, and 5-action microgrid environments are designed. These environments are used to train TD-based RL algorithms. Both environmental models use the same reward function. The second reward function specially designed for these environments is given in equation 4.3.

$$R_2(s_t, a_t) = \begin{cases} -\dfrac{SoC_{goal} - SoC_{t+1}}{100} C_2(s_t, a_0) & ,if \quad a_t = a_0 \\ -\dfrac{Pr_t - Pr_{min}}{Pr_{max} - Pr_{min}} \dfrac{SoC_{goal} - SoC_{t+1}}{100} C_2(s_t, a_{1,2}) & ,if \ a_t = a_{1,2} \\ \dfrac{Pr_t - Pr_{min}}{Pr_{max} - Pr_{min}} \dfrac{SoC_{goal} - SoC_{t+1}}{100} C_2(s_t, a_{3,4}) & ,if \ a_t = a_{3,4} \end{cases} \quad (4.3)$$

Where:

- $C_2(s_t, a_t)$ gives the appropriate penalty value for specific state s_t and action chosen a_t and it considers load demands to do so.
- SoC_{goal} is terminal state of charge value. It is usually determined as %98.
- SoC_{t+1} is next state of charge value after executing the control action.
- Pr_t is the current price value at time t.
- Pr_{max} and Pr_{min} are the highest and lowest price value up to time *t*.

Both 3-action and 5-action environments use this second reward function. However, the 3-action environment model only has actions to do nothing (a_0), charge(a_1) and discharge(a*4*). And the reward function used by this environment uses fixed penalty coefficients instead of specifying the penalty coefficients with $C_2(s_t, a_t)$. On the other hand, the 5-action environment uses this reward function while also using the penalty function, which is specific to this environment. This second penalty function is given in equation 4.4.

$$C_2(s_t, a_t) = \begin{cases} -\dfrac{L_t - L_{min}}{L_{max} - L_{min}} & ,if \quad a_t = a_0 \\ -\dfrac{L_t - L_{min}}{L_{max} - L_{min}} 0.2 SoC_t & ,if \quad a_t = a_1 \\ -\dfrac{L_t - L_{min}}{L_{max} - L_{min}} 0.1 SoC_t & ,if \quad a_t = a_2 \\ \dfrac{L_t - L_{min}}{L_{max} - L_{min}} 0.1 SoC_t & ,if \quad a_t = a_3 \\ \dfrac{L_t - L_{min}}{L_{max} - L_{min}} 0.2 SoC_t & ,if \quad a_t = a_4 \end{cases} \quad (4.4)$$

Where:

- SoC_t is current state of charge value before executing the control action.
- L_t is the current load demand at time t.

- L_{max} and L_{min} are the highest and lowest load demand up to time t.

In this thesis, two reward functions and two penalty functions are designed and proposed. With these functions, three TD-based RL agents and two DRL agents are trained. In 3-action environment design, only price data is used in the optimization of agents, while in 5-action environment design, both dynamic price and power demanded from the microgrid are used in the training of agents. In 7-action environmental design, in addition to dynamic price and load demand, the power level generated from renewable energy sources operating in the microgrid is also used for agents to learn energy management.

4.3 Dataset Analysis

The present thesis introduces an energy management system that encompasses four distinct categories of prediction models. These models are designed to forecast the active power output of both solar and wind power plants within a microgrid, the load demanded from the microgrid, and dynamic pricing. In order to create accurate and effective prediction models, four distinct datasets were sourced from open-source platforms and subjected to specific preprocessing procedures. This section provides an overview of the datasets used in solar power predictions, power prediction of wind turbines, as well as load and price predictions.

In solar power prediction studies, the selection and preparation of the dataset are crucial to the performance of the model. The choice of ML or DL methods highly depends on the nature of the dataset. For example, a CNN-based model can produce good results with a dataset consisting of satellite photos, while a model using LSTM requires training with time-series data for efficient learning of memory cells. The GBM-based models selected for this study possess a specific advantage in that they are capable of learning from datasets with multiple features and rows. This is primarily because these algorithms construct trees as they learn, and datasets with more columns allow for stronger trees, thereby mitigating the issue of over-learning. In light of these considerations, the training dataset is sourced from EDP Open Data [100], which offers open-source data for researchers. This platform provides a variety of datasets for different research purposes. To predict solar power generation from weather data in this study, the meteorological data collected by the weather station in the city of Faro, Portugal, is chosen. This dataset spans from January 1st, 2014, and includes data collected at intervals shorter than five minutes. It comprises nine columns indicating eight meteorological parameters, with the first column representing the date and time. The meteorological features of the dataset are ambient temperature, global radiation, diffuse radiation, ultraviolet radiation, wind speed, wind direction, precipitation, and atmospheric pressure. The dataset contains a total of 524,890 entries. Upon inspection of the dataset, outliers are observed in the ambient temperature, wind velocity, and atmospheric pressure columns. These outliers, which are assumed to result from measurement errors, are appropriately filled with values. The distribution plots of these data, following the removal of outliers, are shown in Figure 4.3.

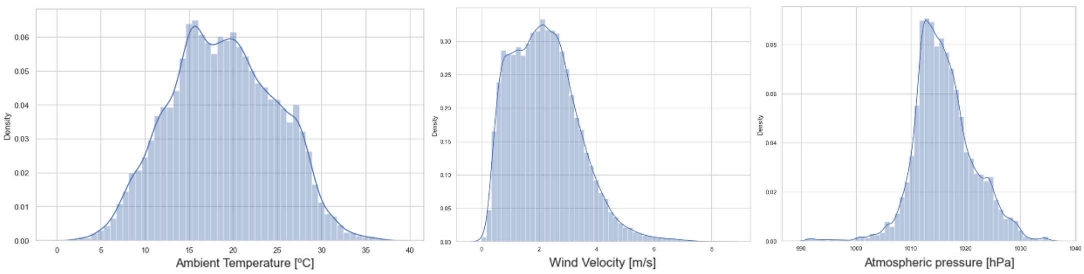


Figure 4.3 : Distributions of ambient temperature, wind velocity and atmospheric pressure

After reorganizing the time intervals between the rows in the dataset, the resulting total number of data was reduced to 43747 entities. Tables 4.1 and 4.2 provide the distribution characteristics of the final form of the dataset containing 47347 data. In tables, ‘std’ stands for standard deviation. In order to predict solar power generation, a dataset consisting of the generation data of two solar power plants located in the same city and position was utilized. This generation dataset was obtained from the same open-source dataset platform [100]. In its raw form, the dataset contains several columns and 202646 entries, including current, voltage, power data, and panel temperatures of the solar power plants. Unlike the meteorological dataset, this dataset starts from January 20, 2014. The dataset underwent several preprocessing steps, beginning with removing outliers, followed by arranging dates and minute intervals in the same manner as the meteorological data. Subsequently, only the required power generation data was retrieved and combined. Finally, the meteorological and power generation data with the same number of inputs were combined into a single dataset.

Table 4.1 : Distribution characteristics of meteorological dataset characteristics I.

	Ambient Temperature [ºC]	Global Radiation [W/m^2]	Diffuse Radiation [W/m^2]	Ultraviolet [W/m^2]
Mean	18.866	229.69	63.36	13.94
Std	5.96	314.13	91.74	18.89
Min	1.75	0.756	1.606	0.500
25%	14.59	1.57	1.72	0.509
50%	18.71	8.453	8.07	1.08
75%	23.28	444.01	96.67	25.25
Max	37.46	1165.49	668.77	72.02

Table 4.2 : Distribution characteristics of meteorological dataset characteristics II.

	Wind Velocity [m/s]	Wind Direction [º]	Precipitation [mm]	Atmospheric pressure [hPa]
Mean	2.199	206.48	0.000697	1016.04
Std	1.161	93.22	0.010302	5.3844
Min	0.000	0.000	0.00000	990.82
25%	1.286	118.83	0.00000	1012.56
50%	2.109	227.33	0.00000	1015.47
75%	2.917	294.67	0.00000	1019.00
Max	8.472	353.93	0.647500	1035.90

The dataset comprises three distinct categories of radiation data recorded over a duration of five years. These data are depicted in Figure 4.4, which illustrates the visual representation of seven randomly selected days (140 hours). Given the absence of radiation at night and its peak occurrence at noon, it can be inferred that the radiation data have a pattern similar to that of power generation levels.

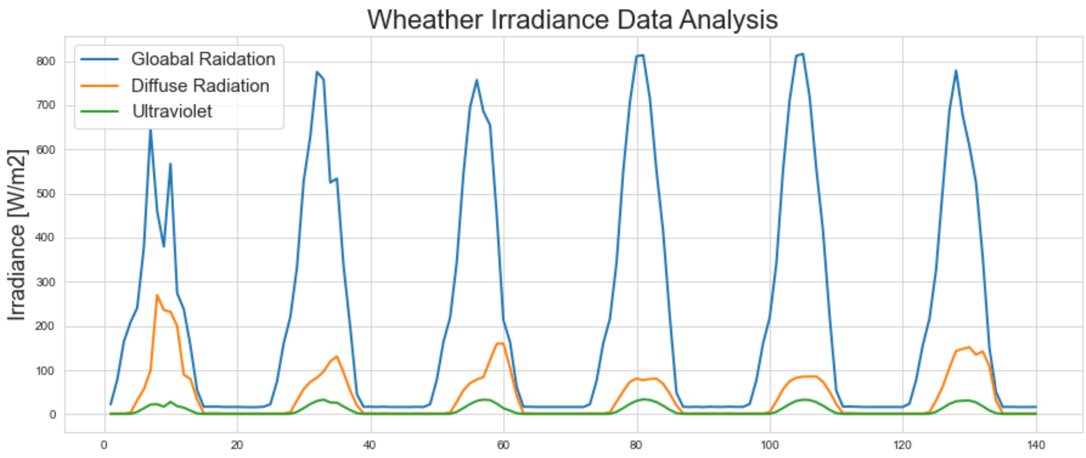


Figure 4.4 : Variation of radiation features in the dataset.

It is reasonable to posit that solar energy generation is intricately connected to radiation and its corresponding parameters. Nonetheless, it is essential to consider other categories of data as well. For instance, the efficiency of panels used in the production of solar energy is significantly impacted by their temperature, which is determined by the ambient temperature. Figure 4.5 provides a visual representation of the variation in ambient temperature over the same seven randomly selected days.

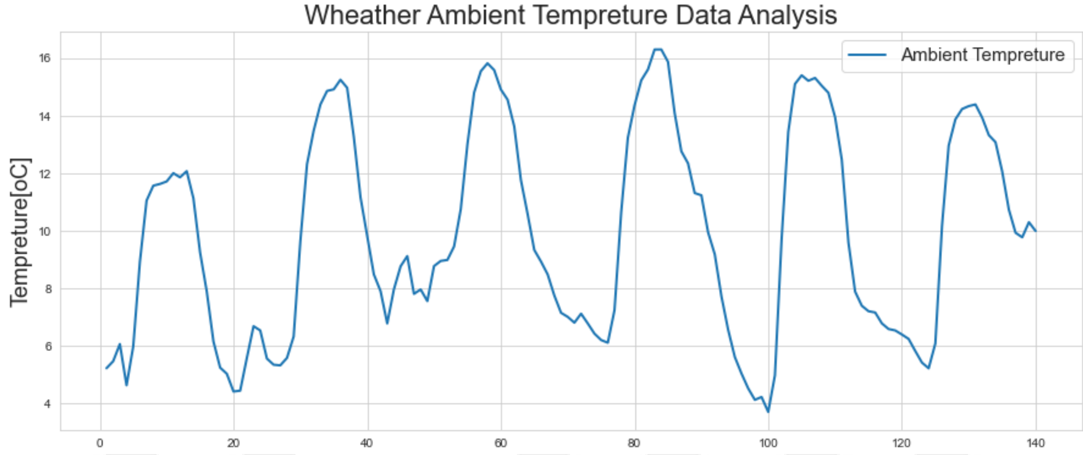


Figure 4.5 : Dynamics of ambient temperature variation along with randomly picked seven days from dataset.

Solar and wind power generation are influenced by various factors besides radiation. One such factor is wind speed, which contributes to power generation by helping to cool the solar panels. Wind speed also affects wind direction, which in turn can impact power generation. Although wind speed is not as crucial to power generation as radiation, it is still an important factor to consider. Figure 4.6 displays the wind speed data for the same seven randomly selected days as the other graphs.

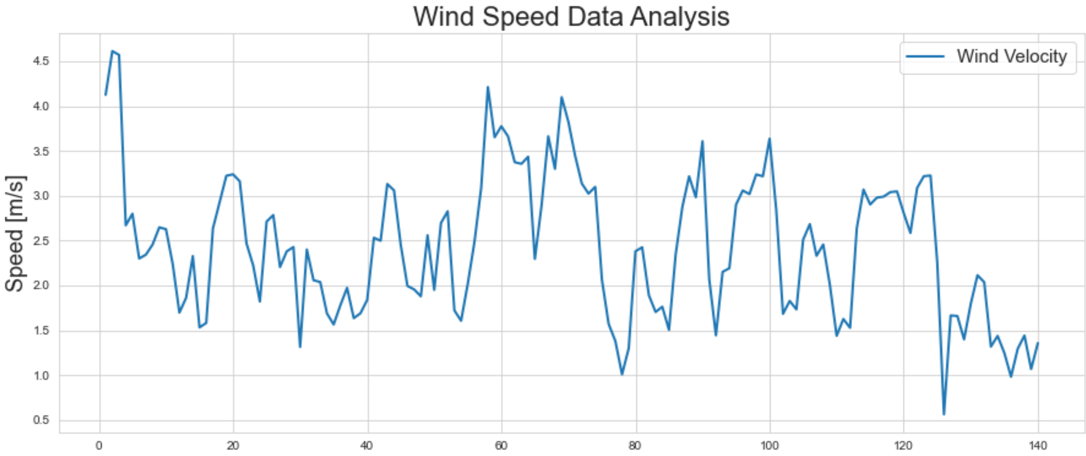


Figure 4.6 : Dynamics of wind speed during hundred and forty hours.

Another factor that can affect power generation is cloudy or rainy weather. Such weather can cause a decrease in power generation, and thus it is necessary to take into account the precipitation data. Figure 4.7 provides a visualization of the precipitation data for the year 2017.

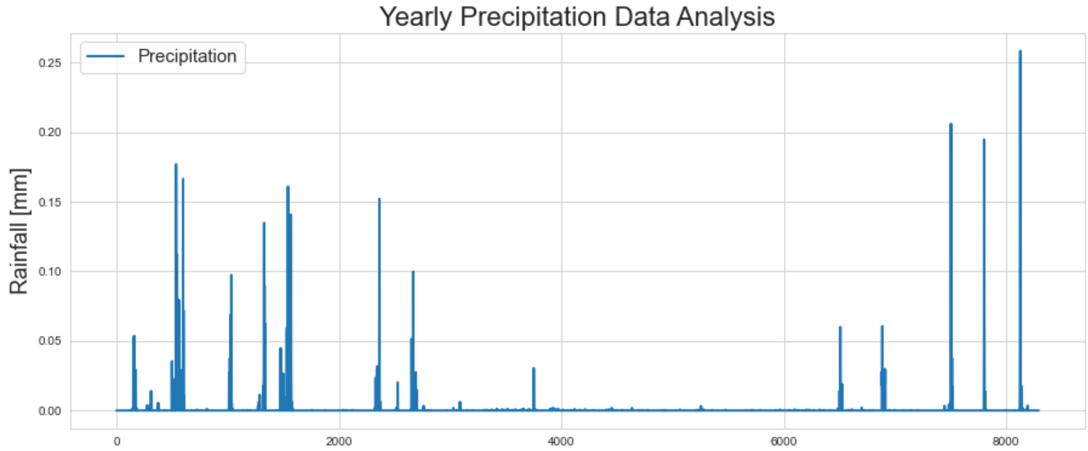


Figure 4.7 : Overview of annual precipitation.

The graphical representation of the meteorological data in Figure 4.7 shows that the summer months are characterized by dry weather, indicating clear skies. Consequently, it can be inferred that energy production levels increase during summer and decrease during autumn, which is the period with the highest amount of rainfall based on the visualization.

Having established the effectiveness of this dataset in power generation analysis, it is deemed suitable for model training after appropriate rearrangement and processing. Specifically, the meteorological data and power generation data are concatenated while ensuring synchronicity, and the index and period are rearranged accordingly.

In addition to the meteorological dataset used for predicting renewable energy sources, a training dataset is required for price and load demand forecasting. In this context, open-source platforms that collect and provide load and price data for European Union countries to researchers are utilized to obtain these datasets [101]. These platforms offer various types of load and price data indexed in time series format, collected at hourly, half-hourly, and 15-minute intervals. The designed energy management system predicts dynamic pricing and load demand. A dataset containing five years of dynamic pricing data from Germany is utilized for dynamic pricing. Additionally, the AIEMS microgrid predicts how much load demand will be required using deep learning models. The load data used by these prediction models is also available on the same platform at the same intervals. For this purpose, the load data to be used is rescaled according to the simulation model of the microgrid. The dynamic pricing and load data sets undergo a series of preprocessing steps to prepare them for training DL models.

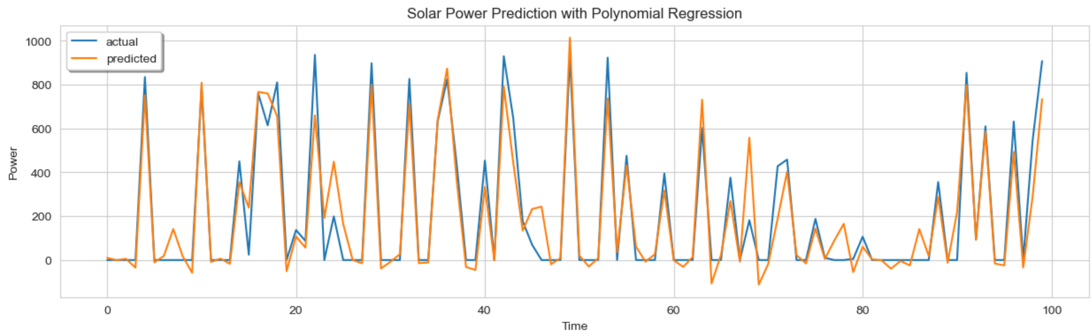
4.4 Prediction Results of Renewable Energy Sources

The simulation model of the microgrid has wind turbines with 20 kW AC active power and a solar power plant with 70 kW AC active power. Knowing the power to be generated from these renewable energy sources is crucial for the dynamic operation of the microgrid. The designed AIEMS has prediction models for the power generation of these sources. While the main decision maker RL agent uses these estimation results to optimize itself, the microgrid manager person uses it to make long-term plans and future decisions. Well-known and new ML methods are used for prediction. ML-based prediction models are trained with the meteorological dataset described in the data analysis section. In this section, results are presented first for the power generation prediction of the solar power plant, followed by the power generation prediction of the wind turbines.

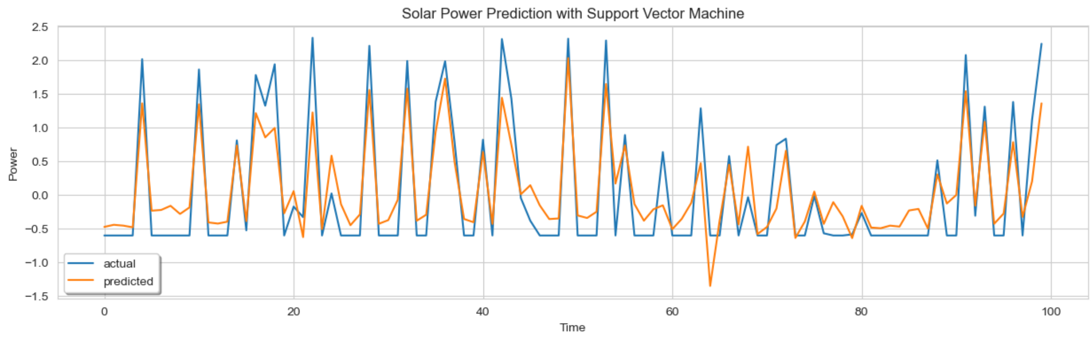
4.4.1 Power prediction of solar power plant

The model training is carried out using a preprocessed dataset containing meteorological data and training four prevalent ML-based prediction models along with three novel GBM-based prediction models proposed for use in the thesis. Initially, prediction models are created using prominent methods such as polynomial regression (Poly. Reg.), SVM, DT, and RF algorithms. Figure 4.8. shows the comparison of these models with each other for randomly selected hundred hours within a year.

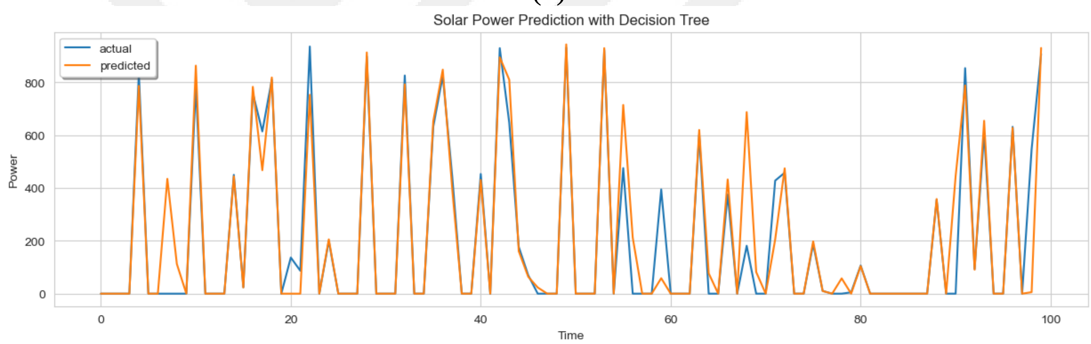
Prediction models created using four commonly used ML techniques with mathematical backgrounds given in Chapter 2 are compared for the same 100-hour period. When examining the graphs, it can be seen that the SVR model has the worst prediction performance. In the graph of this model (Figure 4.8b), the prediction results are generally lower than the actual values.



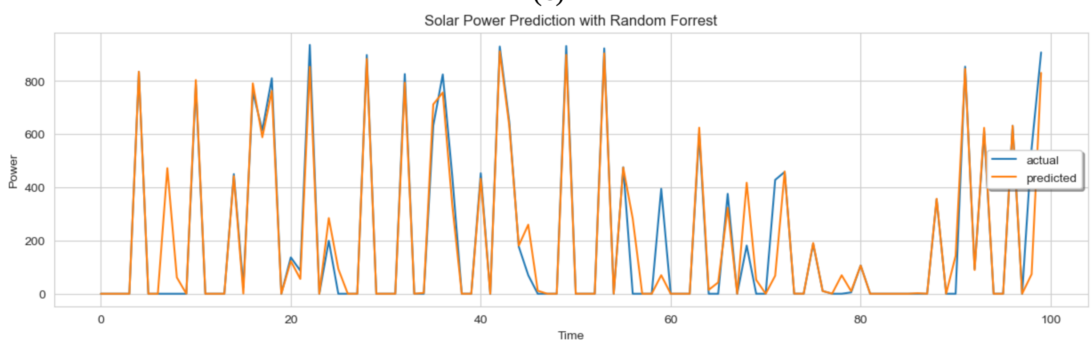
(a)



(b)



(c)



(d)

Figure 4.8 : Comparison of four prevalent prediction models using actual power versus predicted power.

Additionally, around the 63rd hour of the comparison graph of the SVR model, this model predicts very little and negative compared to the actual value. This defect reduces confidence in this model. On the other hand, the second worst performance was exhibited by the polynomial regression prediction model. When examining the graph of this model (Figure 4.8a), it can be seen that sometimes (at hours 11, 19, 39, 62, 65, etc.), this model predicts negative power generated, falling behind the actual value. While the overall success of the polynomial regression prediction model seems high, negative power prediction makes this model problematic. When the graph of the decision tree model (Figure 4.8c) and the random forest graph (Figure 4.8d) are examined, it can be seen that the two models exhibit reliable and similar prediction performance.

In ML, error functions are used to evaluate the performance of a model by measuring the difference between its predicted output and the actual output. The objective of ML models is to minimize the error function, which indicates the degree of error in the model's predictions. A lower error function means the model's predictions are closer to the actual values, which indicates that the model is performing well. Different types of error functions are used for different types of machine learning problems, such as regression or classification. For regression problems like solar power prediction, popular error functions include mean squared error (MSE), mean absolute error (MAE), and root mean squared error (RMSE).

In other respect, R-squared (R^2) is a statistical metric used to measure how well a regression model fits the data. It represents the proportion of variance in the dependent variable that is explained by the independent variables in the model. The R^2 value ranges from 0 to 1, where 0 means the model explains none of the variance in the dependent variable, and 1 means the model explains all the variance. An R^2 value closer to 1 indicates a better fit of the model to the data. Besides, accuracy is a metric used to measure how well a prediction model correctly predicts the class labels of the test data. It represents the percentage of the number of correctly predicted samples to the total number of samples in the test set. The accuracy value ranges from %0 to %100, where %0 means the model makes no correct predictions, and %100 means the model makes all correct predictions. A higher accuracy value indicates better performance of the model. The prediction results of the four models are shown in Table 4.3 with error functions outputs, R^2 and accuracy values.

Table 4.3 : Results of Poly. Reg., SVR, Decision Tree and Random Forrest prediction models.

	MSE	RMSE	MAE	R^2	Accuracy [%]
Poly. Reg	10387.83	101.92	65.53	0.89826	89.08%
SVR	18078.70	134.45	76.05	0.81921	81.92%
DT	12790.32	113.09	42.15	0.87474	87.47%
RF	7135.75	84.47	36.10	0.93011	93.01%

When examining the table, it can be said that the most unsuccessful model in terms of both error values, R^2 and accuracy is SVR. The reason for this is the inability of the kernel functions used by SVR to imitate the relationship between the features that make up the input columns of the dataset and the power values that are its output. On the other hand, it can be seen that the polynomial regression model outperforms the DT model in terms of R^2 and accuracy. However, this model has a significant flaw in its performance, such as predicting negative power, as seen in the comparison graphs. Finally, when comparing the DT and RF prediction models, which produce similar results in the comparison graphs, it can be seen that the RF model, which has an ensemble learning method, demonstrates high R^2 and accuracy. The main reason for this can be attributed to the use of parallel DT models together in the RF model.

The results of the solar power plant prediction problem with four well-known ML methods suggest that the RF prediction model, due to its ensemble learning technique, will be useful. This fact leads to the idea that other ensemble learning techniques may also yield good results for the same prediction problems. Therefore, GBM-based prediction methods have been researched, organized, and adapted to the problem. Prediction models have been created using XGBoost, LightGBM, and CatBoost methods whose mathematical fundamentals are given in Chapter 2. These prediction models also use the same meteorological dataset as the training dataset. Python programming language and open-source libraries are utilized to create the models for each method. XGBoost, LightGBM, and CatBoost are employed as the ML methods, and their respective open-source software libraries are used in the process. The XGBoost library utilized was developed by the people of Distributed Machine Learning Community (DMLC) [85], while Microsoft provides the LightGBM library [102], and Yandex ML team developed and provide the CatBoost library [103].

During the implementation of the XGBoost model, the objective function was selected as objective='reg:linear'. The number of trees was set to n_estimators=10, and the parameter seed=123 was employed to avoid data confusion and prevent memorization for each tree structure. The training time for the model was less than 9 seconds on a computer with a 16 GB RAM and a 2.50 Hz 8-core processor, which is notably shorter than that of non-boosting-based machine learning algorithms. Figure 4.9 presents a comparison between the estimated and actual values for hundred hours.

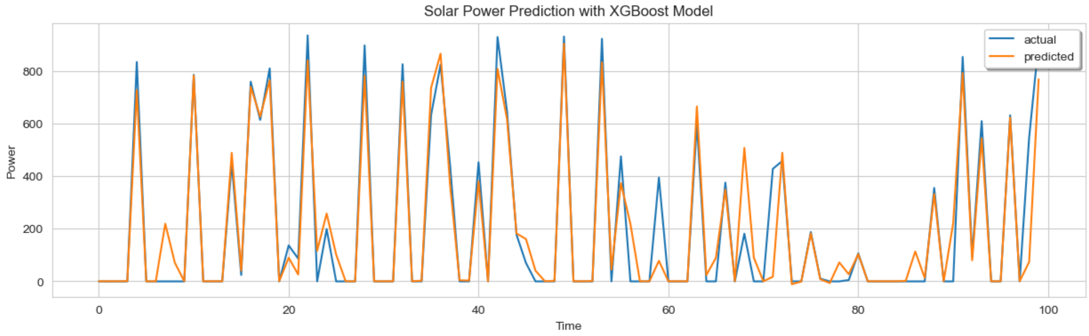


Figure 4.9 : Performance of XGBoost prediction model using actual versus predicted for randomly selected hundred hours.

The XGBoost prediction model has been observed to learn with a high degree of accuracy, specifically 93.56%. The LightGBM model, on the other hand, has identified the 'regression' objective function as a critical parameter for effective estimation studies involving numerical data sets. In addition, a number of leaves equal to 10 has been selected for this model, while the learning rate has been determined to be 0.05. Finally, the power values for the same one hundred hours utilized in the LightGBM results have been plotted and compared with the predicted values, as shown in Figure 4.10.

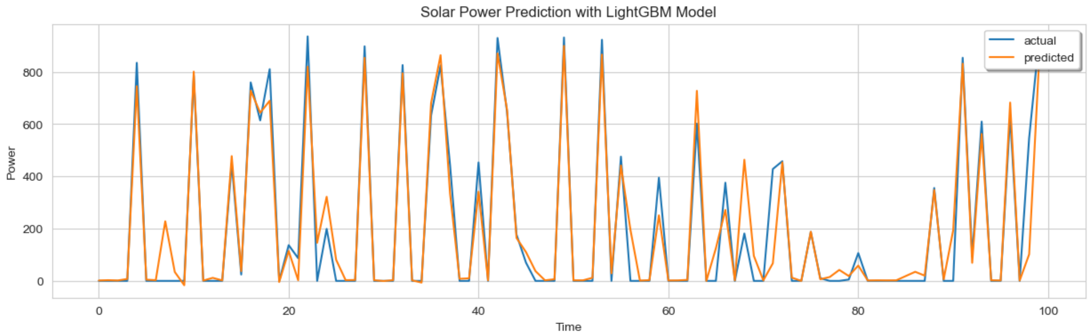


Figure 4.10 : Performance of LightGBM model using actual versus predicted for same one hundred hours.

The LightGBM model has exhibited faster learning times, completing in only around 5 seconds, while also producing more accurate predictions compared to the XGBoost model. Specifically, this forecast model has achieved an accuracy rate of 94.77%.

In the implementation of the CatBoost algorithm, certain critical parameters have been selected, including an iteration parameter of 20, a learning rate of 0.95, and a parameter to determine the depth of the symmetrical tree structure set at 7. The resulting predictions have been plotted in Figure 4.11, utilizing the same 100-hour data set as used in the previous visualization of the two other models' results.

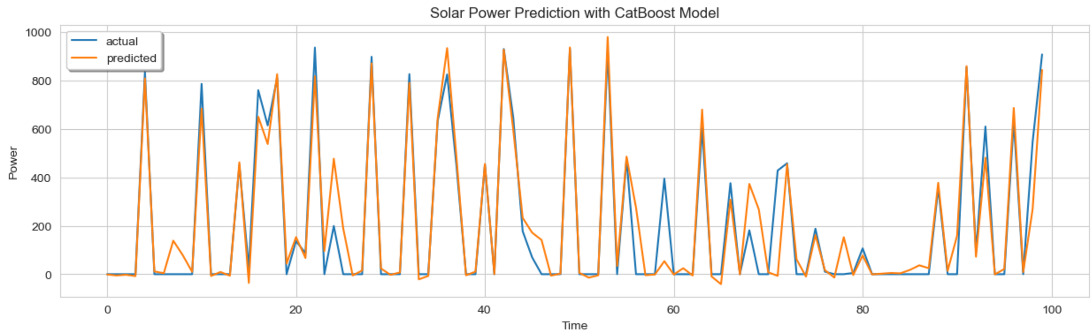


Figure 4.11 : Performance of CatBoost model using actual versus predicted for same one hundred hours.

As depicted in Figure 4.11, the developed forecasting model has achieved a commendable performance with an accuracy of 92.79% for the entire test dataset. Nonetheless, some instances may exhibit predictions that are slightly below zero or exhibit a small deviation from the actual values, such as the case around the 64th hour.

To enable a comparison of the developed GBM-based models, three distinct types of error functions have been selected as performance metrics as well as R2 and accuracy, namely the MSE, the RMSE, and the MAE.

Table 4.4 presents the outputs of three distinct error metrics for the three developed prediction models. The results indicate that the LightGBM model has achieved the highest level of accuracy while also demonstrating the most favorable error measurement outcomes. The XGBoost model, while displaying a high level of accuracy, also exhibits relatively small error metrics. Although the performance outputs of the CatBoost model are comparatively inferior to those of the other models, the results for all three models, including CatBoost, are notably impressive.

Table 4.4 : Results of XGBoost, LightGBM and CatBoost predicition models.

	MSE	RMSE	MAE	R^2	Accuracy [%]
XGBoost	6689.60	81.70	35.82	0.93559	93.56%
LightGBM	5899.44	76.81	32.31	0.94774	94.77%
CatBoost	7590.46	87.12	38.29	0.92791	92.79%

To conduct a comprehensive comparison, the computational metrics associated with the developed models have been analyzed. Specifically, the memory requirements during training, the CPU utilization, and the duration of training have been calculated and are presented in Table 4.5 for each model.

Table 4.5 : Comparison of computational metrics of prediction models.

	Memory Usage[MB]	CPU Usage[%]	Training Time[a]	Training Time[b]
Poly. Reg	10.93	6.42	1.35s	6.98s
SVR	26.23	14.75	3.21s	16.23s
DT	15.25	8.3	1.95s	7.76s
RF	29.75	15.22	2.66s	9.12s
XGBoost	21.617	10.4	1.04s	6.23s
LightGBM	23.745	13.2	938ms	4.98s
CatBoost	22.001	7.8	1.12s	8.42s

Table 4.5 presents two distinct measurements of the training time, labeled as (a) and (b). The first measurement (a) signifies the duration of training when utilizing the one-year dataset, while the second measurement (b) corresponds to the training time when employing the five-year dataset. The results demonstrate that the newly developed GBM-based prediction models surpass the classical ML prediction algorithms with respect to both learning metrics and computational metrics. Among the GBM-based methods, the LightGBM model clearly stands out as superior to the other two models in both measurements. Notably, as the size of the training dataset increases, the LightGBM model exhibits significantly faster learning capabilities than the other two models, despite requiring more memory and greater CPU usage. In contrast, while the XGBoost model demonstrates the lowest memory usage with a moderate CPU requirement, the CatBoost prediction model incurs a higher CPU utilization compared to the other two models.

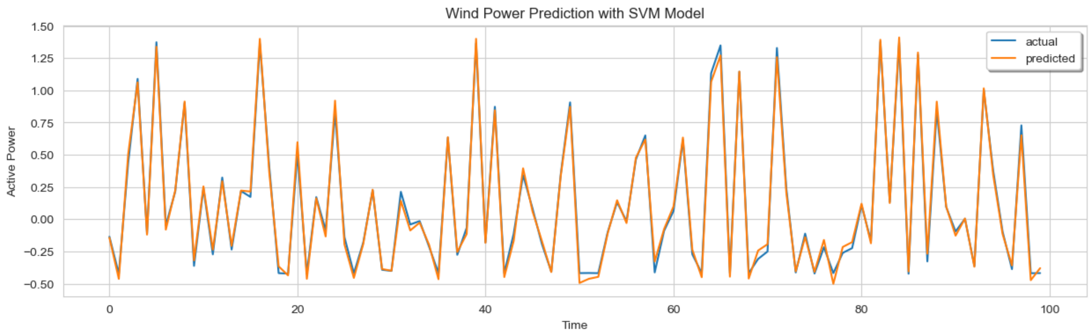
4.4.2 Power prediction of wind turbines

Although not present in every microgrid, wind turbines are significant distributed sources of production. Therefore, in this thesis study, it is assumed that the microgrid being simulated has wind turbines with a 20 kW AC active power. The integration of these turbines into the designed AIEMS and the efficient use of energy production are analyzed. In addition to solar panels, the estimated energy generated by these turbines is crucial for the designed AIEMS. RL agents trained for this AIEMS utilize these production outputs. Furthermore, knowing the power generated and to be generated by wind turbines is crucial for operational decisions, future plans, and decisions regarding the microgrid.

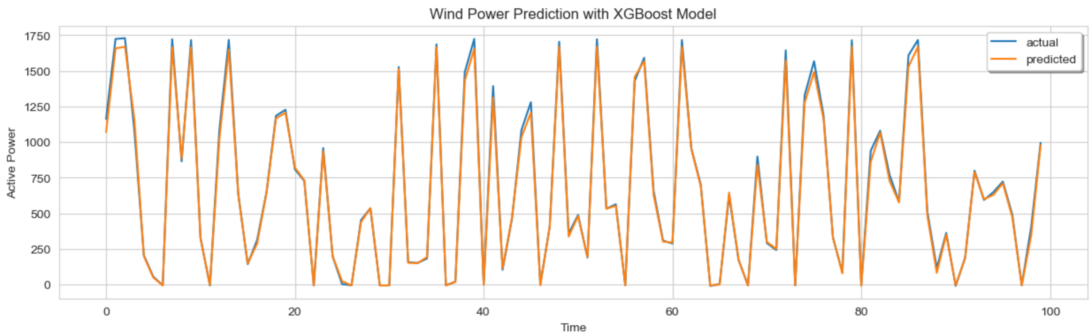
As it can be understood, the energy production of wind turbines is fundamentally dependent on wind speed and wind direction. The meteorological dataset described in the data analysis section has these two types of data for five years. Figure 4.4 shows how wind speed changed over 140 hours. It can be seen that the change in wind speed is not dependent on a particular pattern. Moreover, variations in wind direction are highly dynamic and random. This situation highlights the need to consider these characteristics when choosing the ML method for prediction. In addition to these two data types, mechanical data such as blade pitch angle, motor rotation speed, and some temperature data obtained from a turbine have been included in the training dataset of the prediction model. The active power output is determined as the label.

Considering the characteristics of the dataset to be used for training, it can be anticipated that an SVM-based prediction model will yield good results. Of course, selecting a kernel function that is suitable for the dynamics of the dataset is important for a successful SVR model. Therefore, when creating an SVR model for this problem, the 'Gaussian radial basis function (RBF)', 'Laplace (RBF)', and 'Sigmoid' kernel functions are separately tested, and it is observed that RBF provides the best result. The SVR model is constructed with these parameters. The variation between the actual value and the predicted value for randomly selected hundred hours from the test dataset can be seen in Figure 4.12a. In addition to this method, models are also created using XGBoost and LightGBM methods, which have shown the best performance among the prediction models created in the solar power prediction study. The performance of these models is shown for the same hundred hours in Figure 4.12b and Figure 4.12c.

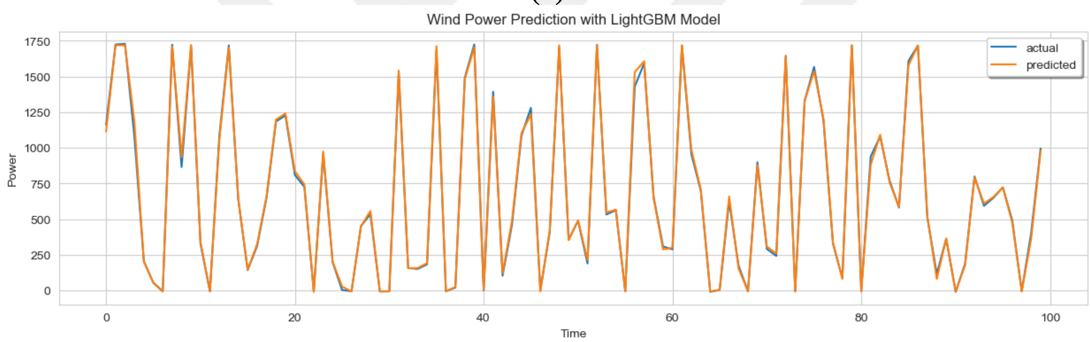
Just like in the solar power prediction described in the previous section, the best result can be seen from the LightGBM model. The predicted values that follow the actual values demonstrate good performance even at these sharp change points.



(a)



(b)



(c)

Figure 4.12 : Comparison of wind power prediction models using actual power versus predicted power.

Input and output values were separately scaled using 'Standard Scaler' and 'Robust Scaler' before training the SVR model. It is observed that the SVR model also predicts the actual values quite well. Although the XGBoost model sometimes produces predictions lower than the actual value, it still demonstrates sufficient performance. The performance metrics of the models are presented in Table 4.6.

Table 4.6 : Performance metrics of SVR, XGBoost and LightGBM predicition models.

	MSE	RMSE	MAE	R^2	Accuracy [%[
SVR	1943.02	44.08	33.66	0.9948	99.48%
XGBoost	1302.16	36.08	22.96	0.9961	99.63%
LightGBM	692.65	26.32	14.77	0.9980	99.80%

Based on both the error values and the R-square (R^2) result, it can be seen that the best performance belongs to LightGBM. The table and graphs confirm each other. XGBoost gives the second best result, while less effective results belong to the SVR model. In terms of accuracy, all three models achieve results above ninety-nine percent, indicating that they have sufficient success. The RL agent in AIEMS uses wind predictions and creates them using the LightGBM model, which yields the most successful results.

4.5 Load Demand and Dynamic Price Prediction Results

One of the parameters that significantly affects the dynamic energy management of a microgrid is the level of demand for power from the microgrid. Both dynamic operations and operational decisions for the future differ depending on whether the load demand is known or not. Furthermore, it is assumed that the simulated microgrid is in a position where dynamic pricing is applied. Accordingly, it is revealed that the price is another variable that is important to know. Knowing the energy price to be applied at that moment directly affects the energy management of the microgrid and indirectly positively impacts its profitability. In this section, solutions for these two problems have been developed using the methods described in chapter 2, and the results have been presented in the subheadings.

4.5.1 Prediction of load demand

An AI-based energy management system relies on data and machine learning algorithms to make decisions in real-time. By analyzing historical data and current trends, the system can accurately forecast future load demand and adjust the energy supply accordingly. This helps to ensure that the microgrid operates efficiently, reduces energy waste, and minimizes costs.

It is assumed that the microgrid under simulation has an annual load demand of approximately 500 Wh. The load profile is generated from load data on open-source platforms [101]. The five-year actual load data is scaled to fit the simulation model. After careful preprocessing of the data and preparation of the dataset for model training, prediction models are trained. The deep learning models whose working principles and infrastructure are provided in Chapter 2 are used in this study. RNN, LSTM, GRU, and Bi-LSTM methods are preferred for load demand prediction problems. The fact that the load data is tightly dependent on time at yearly, monthly, daily, hourly, and even minute intervals is the main reason for choosing these techniques. The models are trained using the same dataset, the same number of neurons, and the same loss and optimization function. The number of epochs for model training is set to 100, and the batch size is set to 32. Figure 4.13 shows how the four models reduced their loss functions during both testing and training.

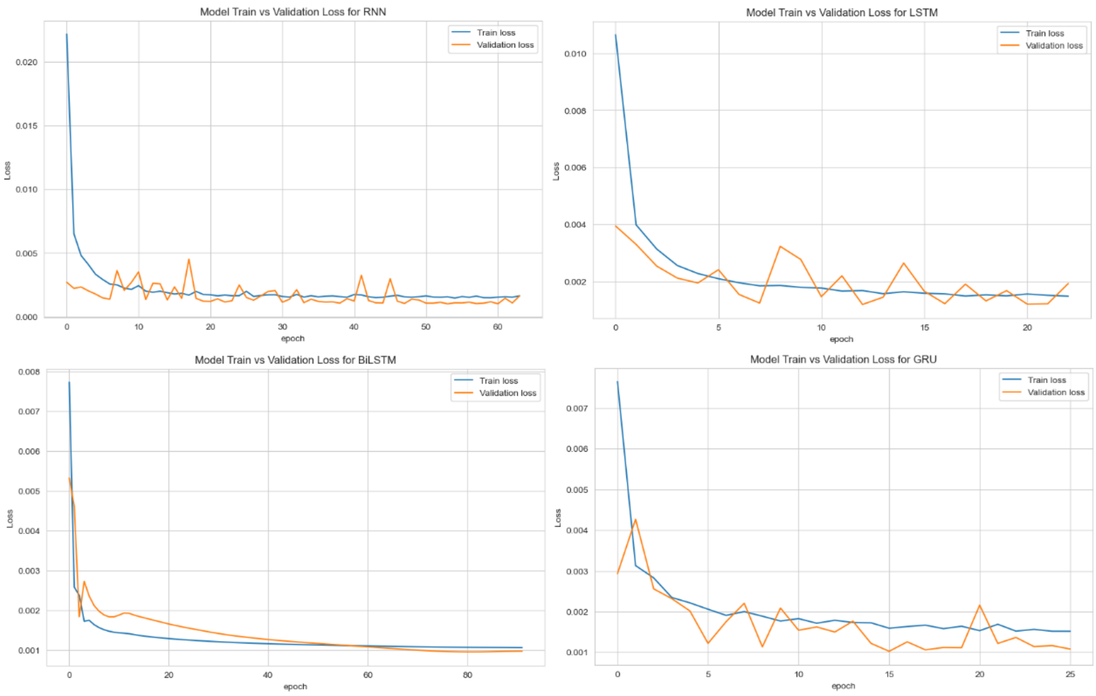
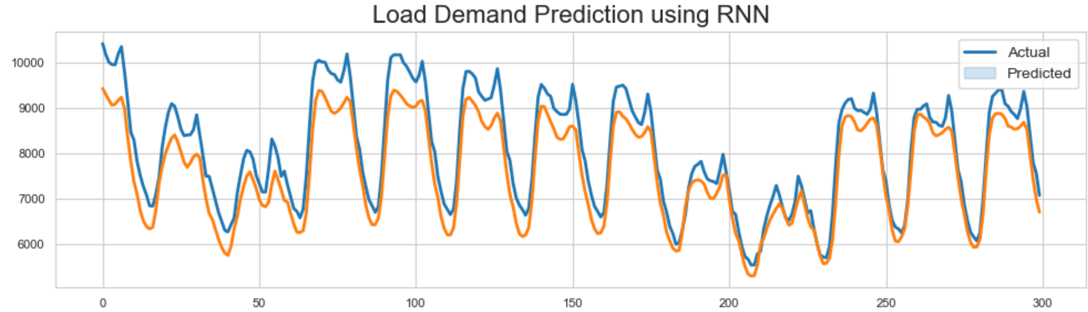
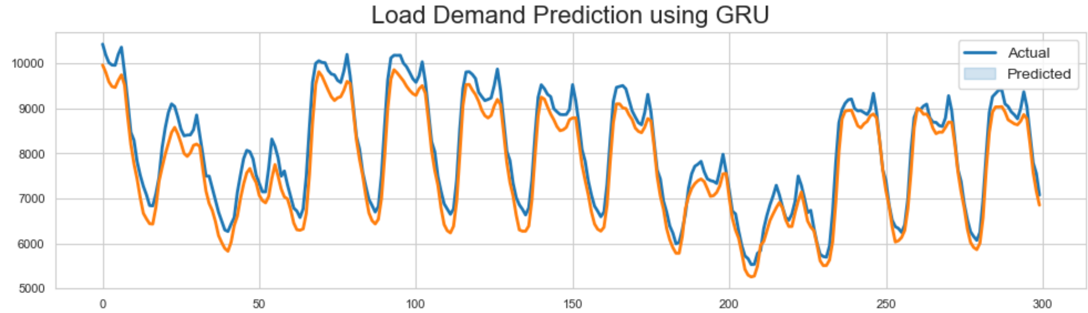


Figure 4.13 : Validation and training loss change for four deep learning models in load demand prediction.

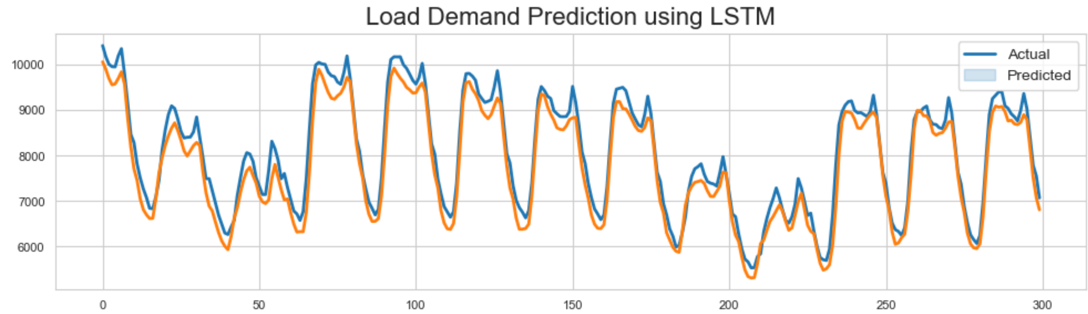
Upon analyzing the results, it can be observed that the Bi-LSTM model exhibits the best learning performance. It has significantly reduced the loss value in a more stable manner. The changes in the loss value during the learning process are represented in blue in all four graphs. Upon examination of these changes, it is seen that the LSTM model shows the second-best learning performance. While all four models exhibit successful learning performance, the RNN model is the least successful among them. To visualize how close the predicted values are to the actual values, the real and predicted values are displayed together. In Figure 4.14, the actual and predicted values for three hundred randomly selected hours from the test data are shown together.



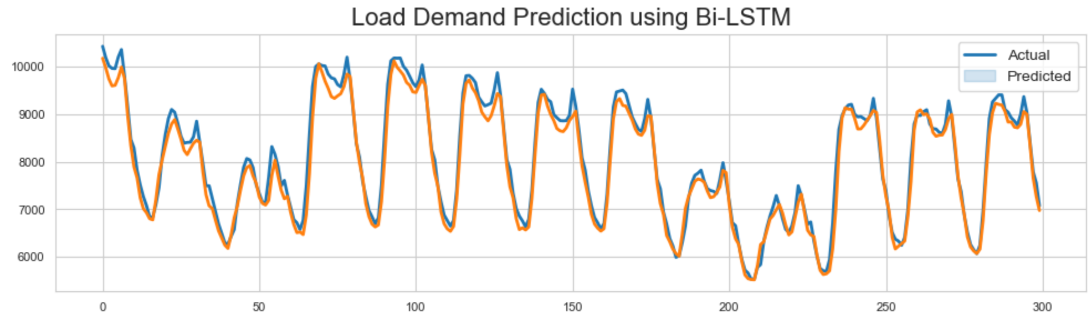
(a)



(b)



(c)



(d)

Figure 4.14 : Actual versus predicted values for four DL models in load demand prediction.

It can be observed in the graphs that all four prediction models generate predictions that are very close to the actual load values. The prediction values that best follow the actual values, confirming the learning performance, belong to the Bi-LSTM graph. Meanwhile, the LSTM and GRU models show similar performance, while the RNN model produces slightly less accurate predictions than the other models. The R-square (R^2) and error measurements of the models are presented in Table 4.7.

Table 4.7 : Comparison the performances of RNN, LSTM, GRU and Bi-LSTM prediction models.

	MSE	RMSE	MAE	R^2 (Accuracy)	Training Loss
RNN	121879	349.11	252.19	0.9385 (93.85%)	≈ 0.0004
GRU	66128	257.15	195.76	0.9666 (96.66%)	≈ 0.0003
LSTM	58141	241.12	187.39	0.9706 (97.06%)	≈ 0.00025
Bi-LSTM	31628	177.84	128.95	0.9804 (98.40%)	≈ 0.0001

Upon examining the table, it can be seen that the best performance in terms of both error values and the value at which the loss function converges belongs to the Bi-LSTM model. Additionally, this model provides an accuracy of over 98%. While the LSTM and GRU models provide a similar performance of approximately 97%, they are also similar in terms of error values. Although the RNN model shows an accuracy of around 94%, the error values indicate significant differences between the actual and predicted values at certain times.

The method that exhibits the best performance among these prediction models, the Bi-LSTM model, is preferred in the AIEMS interface. Additionally, this model is used in training the DRL agents. In terms of energy management, predicting load demand with high accuracy enables both the improvement of user operational decisions and the AI-based decision-maker of AIEMS to optimize itself more appropriately.

4.5.2 Dynamic price prediction

The AIEMS design assumes that the simulation model of the targeted microgrid is located in a location where dynamic pricing is applied. Germany, where dynamic pricing is applied, is chosen as the operating location for the microgrid. The dynamic electricity prices applied here are obtained from open-source platforms [101]. A five-year price change is used to create a training dataset. This dataset is preprocessed, with missing data filled in appropriately and outliers removed. Deep learning models, whose foundations are explained in chapter 2, were trained with the price dataset prepared for training the models.

As can be understood, energy consumption and production differ between night and day. In addition, hours and even minutes of a day are important in terms of energy consumption. Dynamic prices are determined according to the amount of energy consumption and production. This situation reveals the fact that electricity prices are related to energy consumption and that energy consumption is also time-dependent. Therefore, RNN, LSTM, GRU and Bi-LSTM models are preferred due to their memory neuron structures. All models are trained with the same dataset. In addition, the structures, loss functions, and optimizer of the models are selected the same. Although the same methods were selected for load demand prediction study, the DNN structures established in this study have two-layer memory cells. The epoch number is chosen as 100 and batch size as 32. The learning performances of the models are shown in Figure 4.15.

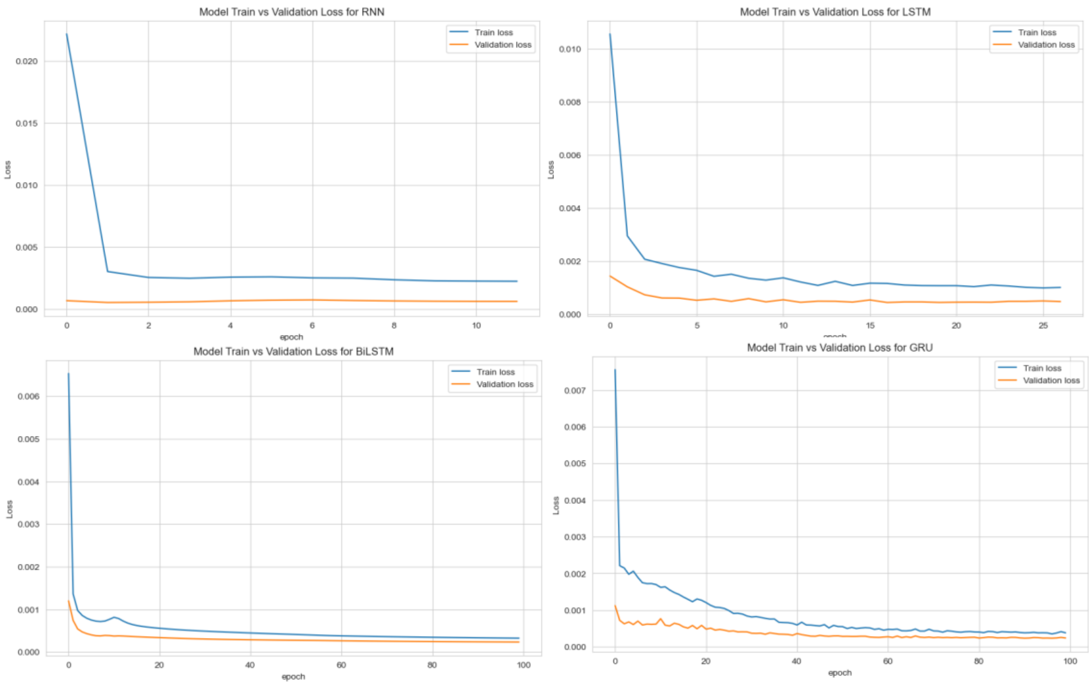
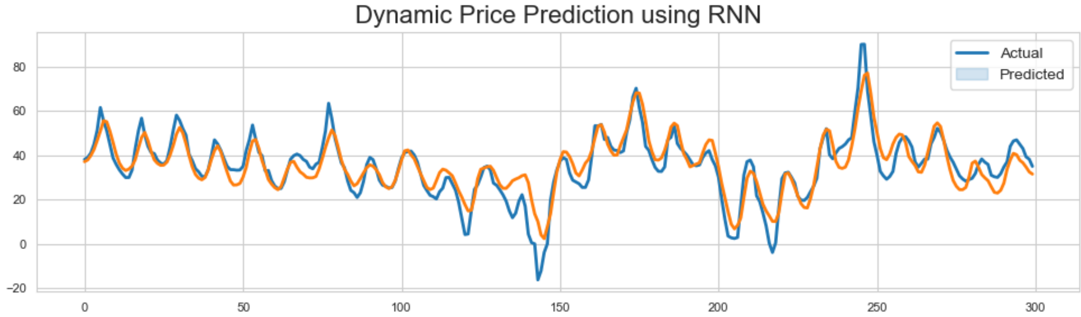
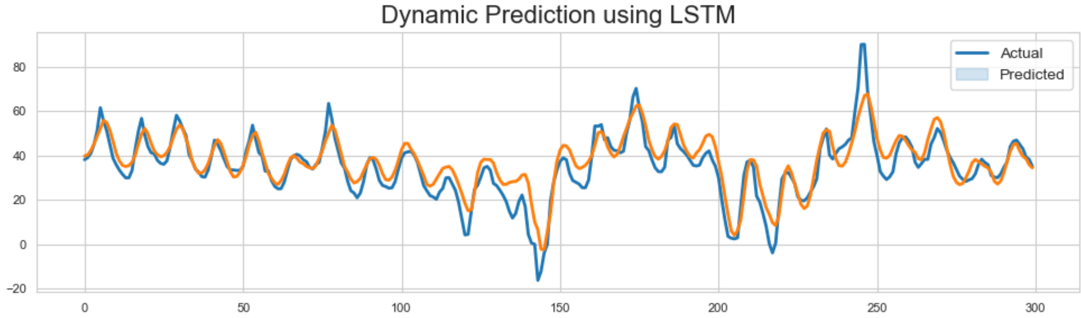


Figure 4.15 : Validation and training loss change for four DL models in dynamic price prediction.

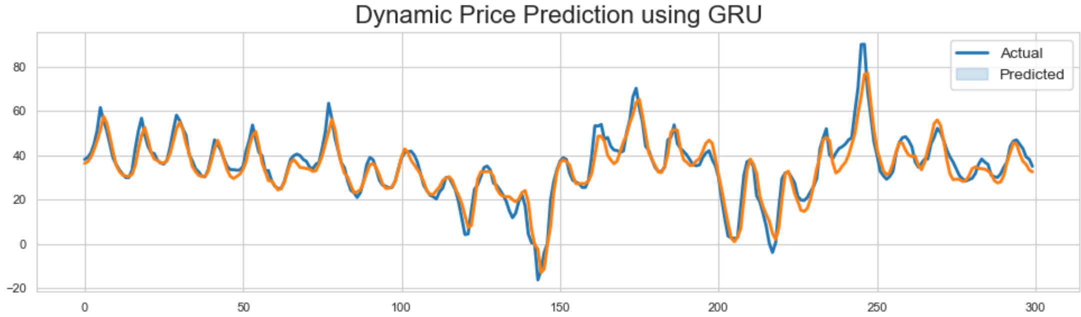
Upon examining the graphs, it can be seen that Bi-LSTM and GRU models have achieved quite good results in terms of the loss variation shown in blue during the training. Although the LSTM model is partly good, the RNN model exhibits a very poor learning performance. After gaining an idea about the models' learning success, the extent to which they make predictions close to the actual values is shown in the graphs in Figure 4.16.



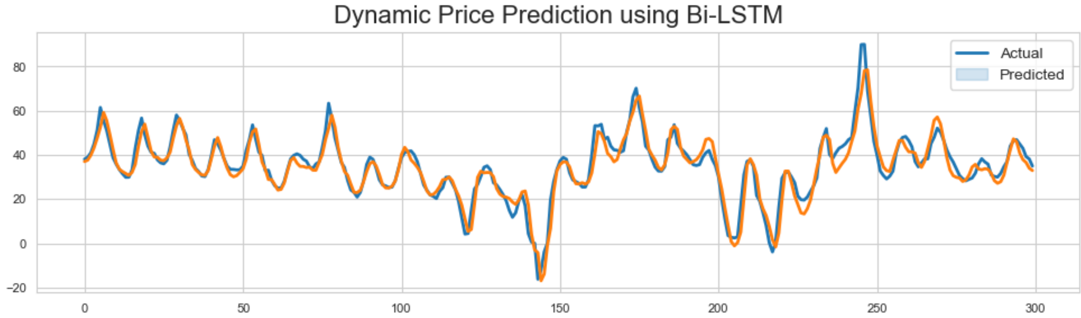
(a)



(b)



(c)



(d)

Figure 4.16 : Actual versus predicted in dynamic price prediction of four DL models.

In the graphs, the actual and predicted values for three hundred randomly selected hours from the test dataset are plotted together. When the graphs are analyzed, it is observed that the Bi-LSTM and GRU models produce similar and highly successful prediction results. On the other hand, RNN and LSTM models exhibit lower prices compared to the actual values. This situation can mislead RL agents trained with these prices. For a more accurate comparison, the accuracies, R-square, and error values of the models are presented together in Table 4.8.

Table 4.8 : Performance results of RNN, LSTM, GRU and Bi-LSTM price prediction models.

	MSE	RMSE	MAE	R^2 (Accuracy)	Training Loss
RNN	120.86	10.99	8.20	0.5658 (56.58%)	≈ 0.0004
GRU	27.36	5.23	3.34	0.9086 (90.86%)	≈ 0.00009
LSTM	102.63	10.13	7.78	0.6312 (63.12%)	≈ 0.0003
Bi-LSTM	23.76	4.87	3.01	0.9214 (92.14%)	≈ 0.00005

As both the error values and the R-square output indicate, the Bi-LSTM model is considered the most successful prediction model with an over 92% accuracy, while the GRU model demonstrates similar success with about 91% accuracy. However, the RNN and LSTM models yielded approximately 56% and 63% accuracy values, respectively. In terms of error values, these two models also produced poor results. The error values, which show the magnitude of the difference between actual and predicted values, demonstrate that these two models are not suitable for AIEMS.

The Bi-LSTM model, with its highly satisfactory results, is used in both the AIEMS interface and the designed environment models used in RL models. Accurate price predictions are crucial both for RL agents to learn to make good decisions and for showing the user what the price will be, which is important for the profitability of the microgrid.

4.6 Performance Outputs of RL Agents

An AI-based energy management system should be designed with an AI-based decision maker. The designed AIEMS requires a fundamental decision maker. In this thesis, the proposed AIEMS implements the decisions of the microgrid user and suggests control actions to the user with an AI-based RL agent. RL agents were identified as the primary controllers of the microgrid in the simulation model, using both Temporal-Difference (TD) and Deep Reinforcement Learning (DRL) methods. These agents are trained with environment models specifically designed for this thesis. Action sets with three, five, and seven actions are proposed, along with reward strategies for each set. The design of the proposed environment models is explained in section 4.1. This section first provides a comparative performance analysis of TD-based RL agents, followed by an assessment of the performance of DRL methods.

4.6.1 TD methods-based agents

The most crucial stage of the designed AIEMS is the design of a core decision maker using the ML and DL models, the results of which were given in the previous sections. For this purpose, RL was selected and the proposed decision maker designs were developed. Five algorithms are used to create the RL agent, which is the basic controller of the designed AIEMS. The results of the five RL agents created, trained, and evaluated for performance are presented in two categories. Firstly, the performance outputs of SARSA, Q Learning and *n*-Step Bootstrapping agents created with TD-based methods are presented in this section. It is examined how well these three methods, whose update rules are unique to them, manage the AIEMS being targeted in the thesis.

Three RL agents were generated through the application of SARSA, Q Learning and the *n*-Step Bootstrapping algorithms. The value of '*n*' that determines the number of times the update rule will be applied with was set at 8. This value is a hyperparameter and may vary to produce optimal results in different problems. Besides '*n*', other hyperparameters such as α, ε, γ and the total number of episodes were kept constant for fair performance comparison. The value of α was set to 0.1, γ was set to 0.95 and ε was set to 0.2, based on the best learning performances of the RL agents in the microgrid environment model. Three RL agents are trained in environments with three and five action sets. These methods, also called tabular methods, create tables containing the Q values of action-state pairs. The seven-action environment model forces the agents to create a huge Q table, which both prolongs the training time and adversely affects the learning performance of the agents. Therefore, microgrid environment models with three and five action sets are used for simulation. The number of episodes, which expresses the number of simulations, affects the learning performance of the agents. In order to see this change, agents were trained separately with the iteration numbers of 10000, 50000 and 75000. Agents are trained with these numbers and results are obtained. Figure 4.17, which contains 1600 episodes, shows the fluctuation in reward collection by RL agents over the number of iterations that result in the best learning performance. In the figure, it can be concluded that the three agents learned quickly. Also, it can be seen that the *n*-Step Bootstrapping agent, seen in darker green, learns faster than other agents

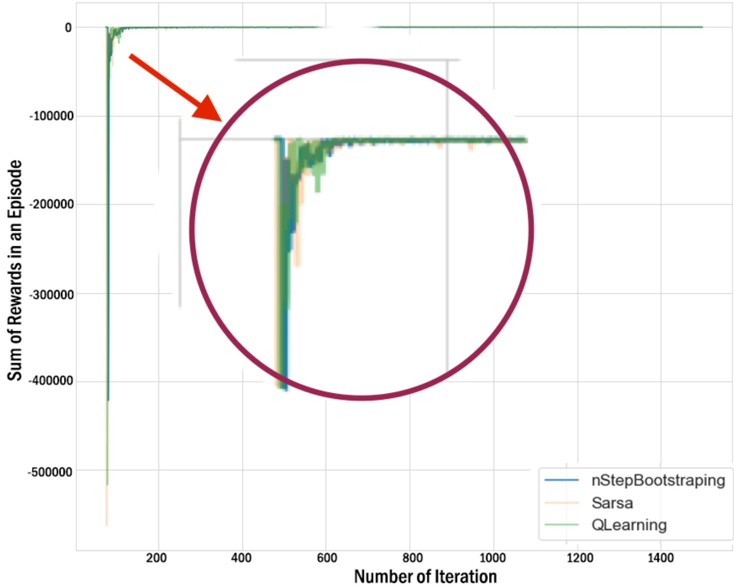
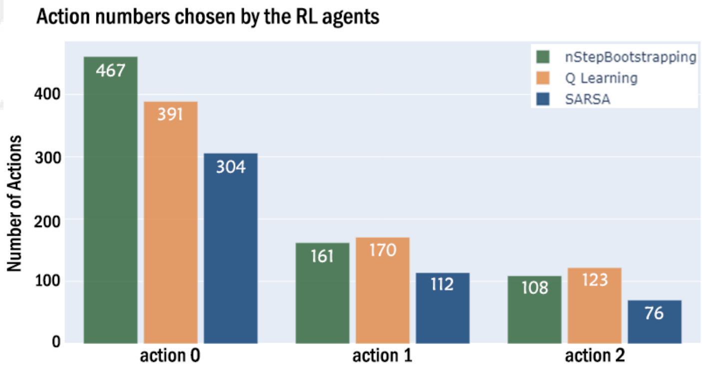
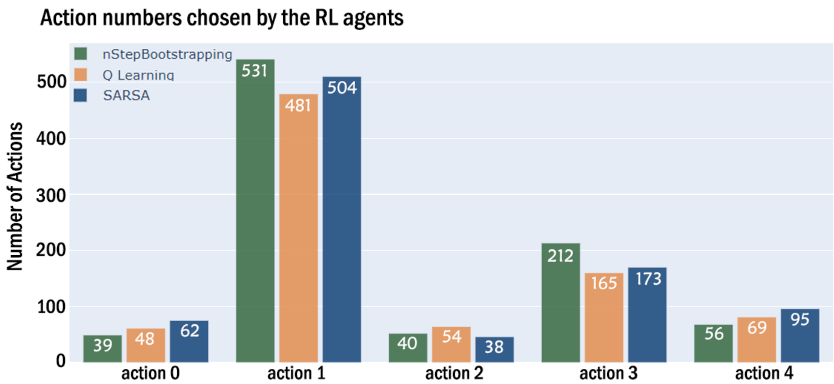


Figure 4.17 : Agents' performances in terms of reward collection.

Moreover, SARSA agent has worst learning speed among them. Agents were trained in 3-action and 5-action environments in the same episode count. Information on how many of which actions the agents have chosen is given in Figure 4.18.



(a)



(b)

Figure 4.18 : a) Selected action number in 3-action environment. b) Selected action number in 5-action environment

As depicted in Figure 4.18a, agents trained with the 3-action environment model which has actions of *action0: doing nothing*, *action1: charge* and *action1: discharge* tend to choose the action of doing nothing. This training leads to agents not providing profitability to the microgrid. In addition, in the 3-action environment model, it is seen that the Q Learning agent chooses the most charge and discharge action. This environment model makes the Q Learning agent more active than others. On the other hand, the actions chosen by the agents trained with the 5-action environment model that has *action0: doing nothing*, *action1,2: two level charge* and *action3,4: two level discharge* are shown in Figure 4.18b. Agents trained with this environmental model produce more dynamic outputs. Among them, the *n*-Step Bootstrapping agent appears to be more operational than the others in terms of both charge and discharge. These two graphs also show that both environmental designs are suitable for the training of agents. In order to see the results of the agents in more detail, the outputs of each agent in two different environmental models and in three different iterations are presented in tables. Table 4.9 shows the results for the SARSA agent, while Table 4.10 shows the results for the Q Learning agent. Also, the performance results of the *n*-Step Bootstrapping agent are given in Table 4.11. The three tables were created using the same evaluation criteria in order to make an accurate comparison.

Table 4.9 : Performance metrics of SARSA agent.

	3-Actions Env.			5-Actions Env.		
	10k	50k	75k	10k	50k	75k
Average returns	-358.2	-130.8	-58.26	-206.72	-20.38	-3.38
Training time[min]	1.09	2.36	4.66	2.14	5.44	6.54
Charging cost	91.22€	54.22€	52.93€	74.26€	48.09€	44.27€
Saving [%]	-83.22	-10.76	-4.87	-64.32	-0.98	5.81

Table 4.10 : Performance metrics of Q Learning agent.

	3-Actions Env.			5-Actions Env.		
	10k	50k	75k	10k	50k	75k
Average returns	-283.7	-114.4	-52.41	-149.50	-3.83	8.44
Training time[min]	1.13	3.22	5.19	4.23	5.56	7.34
Charging cost	89.22€	53.05€	51.45€	70.69€	45.93€	36.86€
Saving [%]	-80.15	-9.34	-4.47	-54.22	3.56	22.61

Table 4.11 : Performance metrics of *n*-Step Bootstrapping agent.

	3-Actions Env.			5-Actions Env.		
	10k	50k	75k	10k	50k	75k
Average returns	-223.3	-66.9	-23.92	-130.22	3.77	10.34
Training time[min]	1.49	4.42	7.13	6.31	8.39	11.02
Charging cost	87.82€	48.22€	44.28€	65.06€	42.19€	32.06€
Saving [%]	-79.12	-1.52	4.56	-64.32	11.42	32.69

Before analyzing the tables, it should be noted that if the RL agent was not used and the microgrid's energy storage unit was charged at the average electricity price of Germany, this charging process would cost around 47.63€. When the tables are examined, firstly, in terms of environment models, it is seen that the 3-action environment offers less profitability than the 5-action environment even in all iteration numbers. It can be understood that this is due to the fact that agents often choose to do nothing (see Figure 4.17a). Among the agents trained with 3-action environment, only the *n*-Step Bootstrapping agent offers around 5% profitability when trained with 75k iterations. On the other hand, all agents trained with the 5-action environment offer profitability to the microgrid. If the agents trained with this environment are examined in terms of iteration numbers. It is deduced that the number of 10k iterations is not sufficient to learn the energy management of the agents. At this number of iterations, agents cost much more energy, the average rewards they collect are very low, and they offer negative profitability to the microgrid. At 50k and 75k iteration numbers, agents produce similar results. It follows from this that agents won't offer more profitability with division numbers over 75k. When the 5-action environment in which the agents perform best and the training results with 75k iterations are examined, it is seen that the n-Step Bootstrapping agent is superior to the others. In this number of iterations, SARSA agent provides 5.81% profitability to microgrid, Q Learning agent 22.61% and *n*-Step Bootstrapping agent 32.69% profitability to microgrid. It can be concluded from the reward collecting graphs, action selection graphs and tables that agents can learn energy management. Seeing how the agents change the SoC of the energy storage unit provides different insights into the performance of the trained agents. Figure 4.19 demonstrates how three agents trained with the best parameters increased their energy storage from 0% SoC at empty to 100% SoC at fully charged.

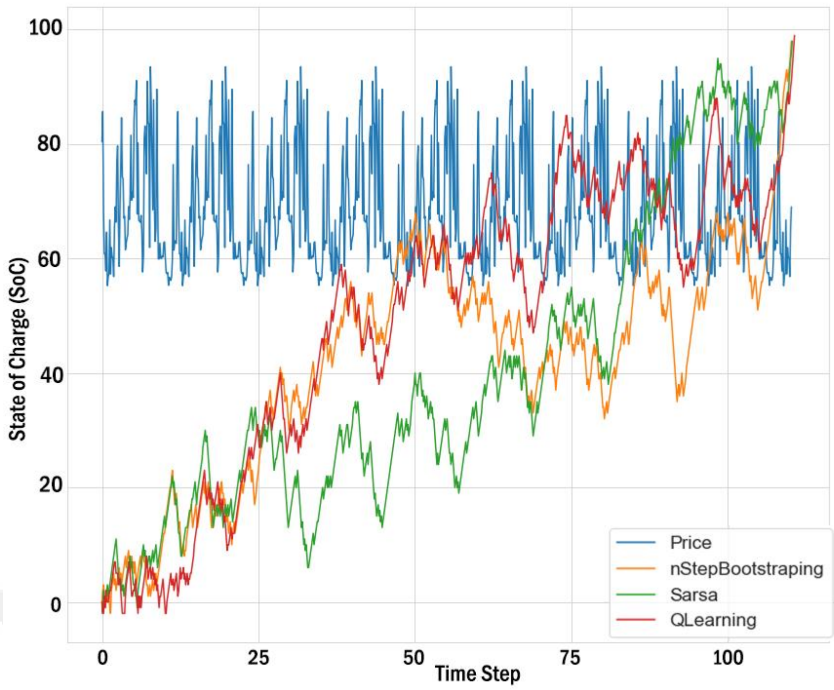


Figure 4.19 : SoC boost performance of agents versus price.

Analyzing the graph, when the energy storage unit is empty, unlike SARSA, Q learning and *n*-Step Bootstrapping agents choose to wait when the price is high. It can be seen from the SoC change that the *n*-Step Bootstrapping agent, indicated in orange, chooses more discharge action. This provides profitability to the microgrid. This situation confirms the tables. In addition, it is seen that the Q Learning agent shown in red discharges when the price is high and shows profitability. SARSA agent showed less accurate actions than other agents.

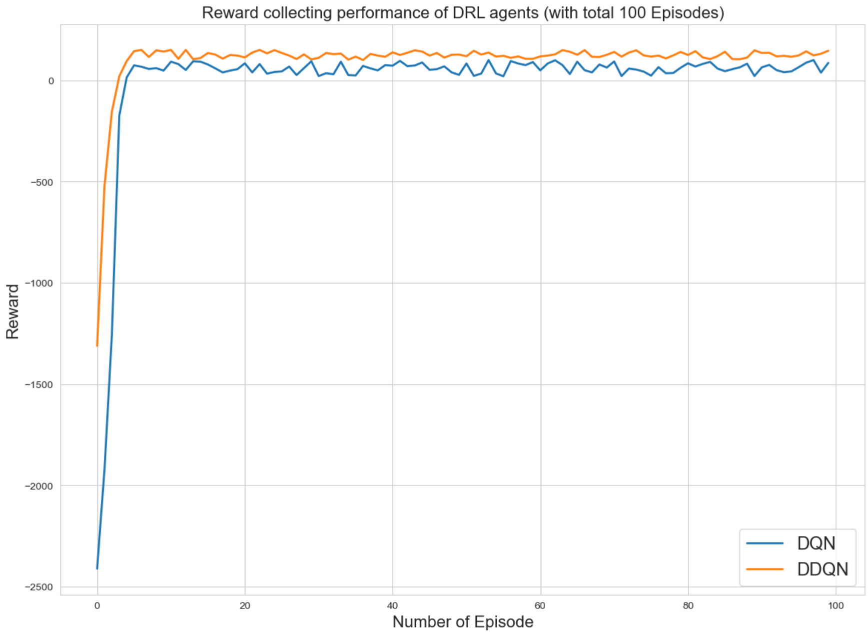
In this section, the performances of the TD-based methods created for the designed AIEMS are presented. These methods are trained with a 3-action environment model using price data only and a 5-action environment model using both price and load demand. It is concluded that TD-based agents trained with the 5-action environment model better learn the energy management system. Among these methods, it is seen that the *n*-Step Bootstrapping method produces better results by taking into account more rewards in the update rule in each episode. It can be said that the agents learned to control the energy storage unit productively. However, it should be noted that these three methods create a Q table using all possible states and actions and determine their actions accordingly. This is possible by knowing approximately all possible states of the microgrid. Moreover, it is unclear which action the agents will choose when a state outside the table occurs. This is the biggest disadvantage of tabular methods. In order to prevent this undesirable situation, DRL-based methods are used.

4.6.2 DRL agents

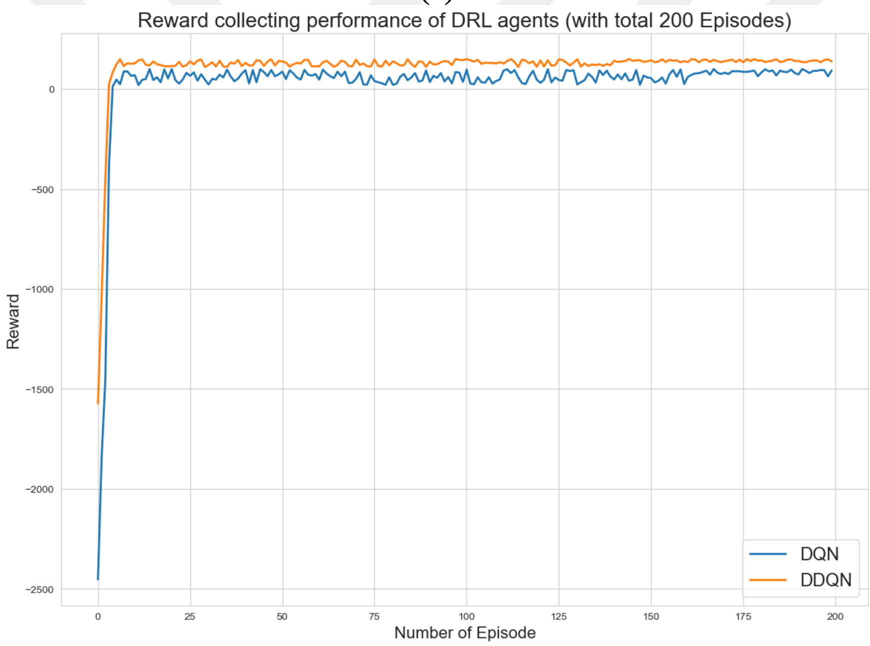
It is nearly impossible to know all possible situations a system will encounter. However, TD-based methods are trained for all possible states. It makes it unclear what decisions such agents would make for a different state they would encounter. The decision agent to be created for the designed AIEMS is expected to be ready for any situation. To meet this expectation, DRL methods that do not need to create a Q table are selected. For this purpose, the use of the DQN algorithm and the use of the DDQN algorithm are determined. Two DRL agents are created using the two algorithms. These agents are trained with the 7-action environment model designed and proposed specifically for this thesis. This environmental model creates its rewards and penalty coefficients for each action separately. In addition, it uses dynamic price, load demand and power generation values of renewable energy sources, which are the most important variables for a microgrid when creating these rewards and penalties. This environment is unsolvable for TD-based algorithms and can only train a DRL-based agent. Therefore, the 7-action environment is not used in the training of TD agents, the results of which were given in the previous section.

First, the DQN agent is created. This agent has experience replay memory with a capacity of 1000000 *(s, a, r, s')*. As is known, this agent approximates the *Q(s, a)* values using DNN. The DNN structure used for this purpose has an input layer of 128 neurons, hidden layers consisting of 64, 48 and 16 neurons, respectively, and an output layer of 7 neurons. With a 32-capacity mini-batch, the DNN structure optimizes itself. It uses 'AdamW' algorithm as optimizer. It also generates the cost values with the 'mean squared error' loss function. Second, the DDQN agent is created. This algorithm uses two different DNN structures called Q network and Target Q network. These DNN structures are set up with parameters similar to those in the DQN agent. Two agents are trained separately with 100 and 200 episodes.

The total rewards that agents collect at the end of each episode, which is the most important indicator of learning performance, are visualized in Figure 4.20.



(a)



(b)

Figure 4.20 : Reward colleting outputs of DQN and DDQN Agents a) for 100 episodes b) for 200 episodes.

These two agents are trained in exactly the same environment. When the graphics are examined, it is seen that the DQN agent collects rather bad rewards in the first episodes compared to the DDQN agent. It is also clear that the rewards that the DQN agent collects later in learning fluctuate more. In both 100- and 200-episode trainings, it is seen that the DDQN agent not only learns faster, but also collects more rewards. When Figure 4.19b is examined, it is seen that two agents collect more stable rewards towards the last episodes, while the DDQN agent is more stable from the 132nd episode.

Agents who have completed their training are tested over twenty episodes. The numbers of the actions they chose during this test are shown in Figure 4.21.

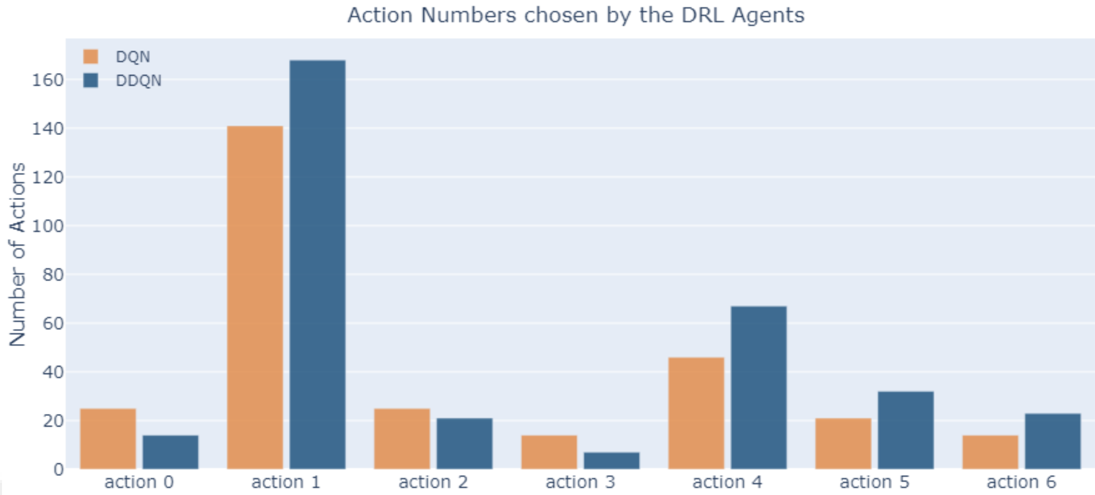


Figure 4.21 : Action numbers selected by DQN and DDQN agents.

The DDQN agent chose more charging action when the price was cheap, while it chose more discharge action when the price was expensive or the need was high (see Figure 4.21) Considering these action numbers, it can be said that the DDQN agent is more operational and active. The learning and action selection performances of DDQN and DQN agents lead to the conclusion that these agents have learned the energy management of the microgrid. To reinforce this conclusion, metrics showing average reward, training time, energy cost, and percentage of earnings are given in Table 4.12.

Table 4.12 : Performance metrics of DRL agents.

	DQN Agent		DDQN Agent	
Number of Episodes:	100	200	100	200
Average returns	0.21	32.41	101.26	114.57
Training time[hours]	4.10	5.56	6.24	7.47
Charging cost	31.22€	29.05€	26.78€	25.27€
Saving [%]	34.45	38.65	44.77	46.94

Both the average reward collected at the end of 100 episodes and the average reward collected after 200 episodes can be considered quite successful. DDQN outperforms DQN in terms of average rewards it collects. The training period takes 4 hours and 10 minutes for 100 episodes and 5 hours and 56 minutes for 200 episodes for the DQN algorithm. In terms of the time to complete his training, the DQN agent is superior to the DDQN agent. These algorithms based on deep learning require much more training time than TD based algorithms. In terms of energy cost, two agendas give successful results. While the DQN agent trained with 200 episodes provides 38.65% profitability, the DDQN agent trained with the same episode number provides 46.94% profitability. Compared to TD based RL agents, DRL agents are much more successful in terms of profitability. Increasing the number of training episodes for the two DRL agents from 100 to 200 did not make any obvious difference in profitability (see Table 4.12.). This shows the result that both agents completed between 100 and 200 training. Consequently, in terms of learning performance, the DDQN agent is superior to the DQN agent in many parameters. This success is caused by the DDQN agent using two DNN structures.

In this thesis, five agents, three TD-based RL and two DRL agents, were created and trained as the main decision maker of AIEMS. Agents were trained with three different environmental models designed as microgrid simulation models. Each agent has been trained multiple times in different iterations and number of segments. Optimized with the best hyperparameters. TD based algorithms are trained with demand load values and dynamic price data from a microgrid. On the other hand, DRL agents are trained with both load demand, dynamic price and the power generated by solar and wind farms. Thus, DRL agents make a decision for every possible situation using the power generation level of renewable energy sources, load demand and dynamic price data.

In the next chapter, it is given how all ML and DL prediction models and decision-making RL agents are combined under single roof. The DDQN agent with the best results was chosen as the main decision maker. In addition, five RL agents have been embedded to the designed GUI. The performance of all agents can be viewed on the energy storage page of the GUI.

5. DESIGN OF THE GRAPHICAL USER INTERFACE

5.1 Introduction

In this thesis, many AI solutions have been proposed that can be used in the dynamic energy management of a microgrid. The most vital constituent of the targeted energy management system is the decision-making AI agent and accurate predictions of the outputs of the microgrid that this agent uses to make its decisions correctly. The successful results of the ML and DL based prediction models created for this thesis are presented in the previous chapter. In addition, learning performances of decision-making RL agents created using different methods are evaluated in the same chapter and successful results are presented. These models and agents, which are now sure to work correctly, are gathered under one roof. All methods are brought together in the designed graphical user interface and the energy management system of the microgrid is created for the user. In this section, the features, pars, and offerings of the designed graphical user interface (GUI) are given.

5.2 Features and Functionalities of the GUI

The designed GUI of the energy management system consists of 7 pages (tabs). The features of these tabs and what they offer to the user are explained in sub-titles. The intended primary goal of AIEMS is to perform dynamic operations of a microgrid. To achieve this, all necessary features required by AIEMS have been integrated into the energy management system using different techniques and compared to one another. However, it is necessary to provide individual solutions for all AI models presented to the microgrid user. In this regard, a GUI specific to this thesis work is designed. This interface includes all prediction models as well as RL agents that use the results of these predictions. This computer interface, consisting of a total of seven pages, is prepared using the latest GUI libraries and works with the same performance on both

Windows and MacOS operating systems. In this section, the contents, features, and offerings of all pages (sometimes called ‘tabs’) are described under subheadings.

5.2.1 Main page

The dynamic operations of the microgrid are performed jointly by the human manager (i.e., the microgrid user) and the AI-based controller. Therefore, in the interface design, the information, and authorizations that the human user will need are taken into consideration. While the main page of AIEMS's GUI provides the user with many parameters and data related to the microgrid, it also allows the user to perform control actions related to the microgrid. The general layout of this page is shown in Figure 5.1. As can be seen, the main page design consists of three sections (layouts). These three parts, where different microgrid data can be viewed and different authorizations can be obtained, are explained in the following sections under headings.

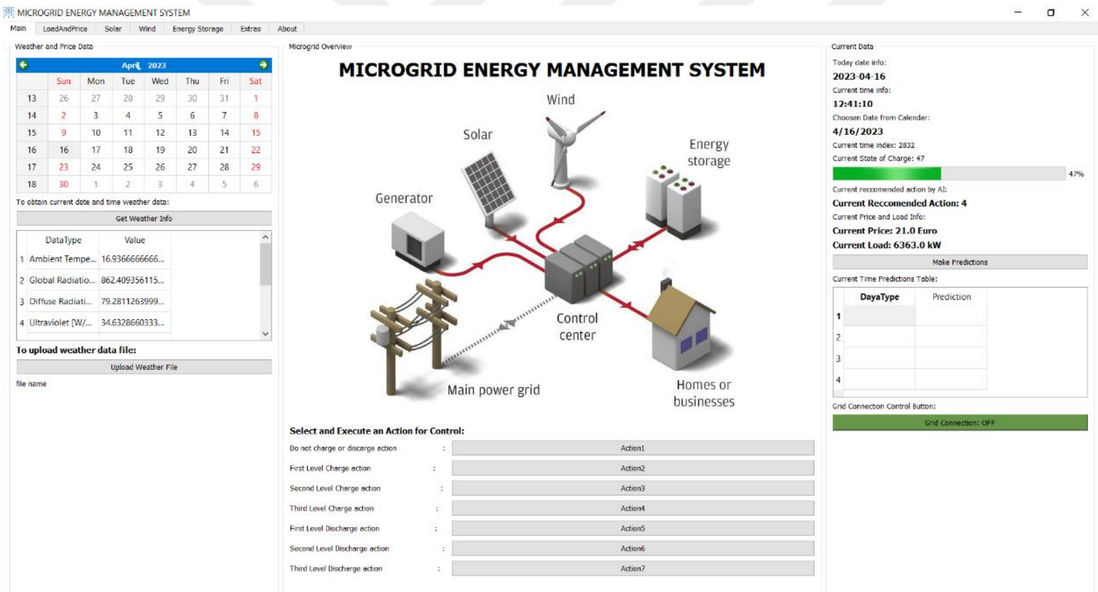


Figure 5.1 : General look of the main page.

5.2.1.1 Meteorological data part

This layout, located on the left side of the page, is named *'weather data'*. The calendar placed here offers different conveniences to the user. First of all, not only does it show that day, but the '*Get Weather Info'* button positioned just below the calendar, the user gets the meteorological data of the day the user is in. Secondly, if the wants to know what the weather will be like in the future, he chooses a day from the future and presses the same button again. Prediction models in the designed energy management system present future weather forecasts to the user. This part helps the user to make decisions about the microgrid, as well as obtains the meteorological data needed by the power prediction models of renewable energy sources.

5.2.1.2 Action selection part

The designed energy management system offers some control actions to the user. The user can manage the energy storage unit with the control actions provided by EMS. In the middle of the page, seven control actions are positioned to manage the energy storage unit. These are charge and discharge actions at different levels, and the action that controls whether the energy storage system does not charge and discharge when needed. Here, the action executed by the user is colored green.

5.2.1.3 Making prediction and action recommendation part

This part of the main page, where information about the energy resources of the microgrid is obtained, is positioned on the far right of the page. In this part, the user can see several data about the microgrid. In addition to current day and time information, it can be seen the SoC value of the ESS within the microgrid. Moreover, current load and price information can be seen. The information of the future day selected from the calendar and information about that day can be seen. The 'Make Predictions' button, also located in this section, makes predictions for the selected day in the calendar (current day or a future day) using the ML and DL models, which have been shown to give good results in the chapter four. These predictions and the details of the part are given in Figure 5.2.

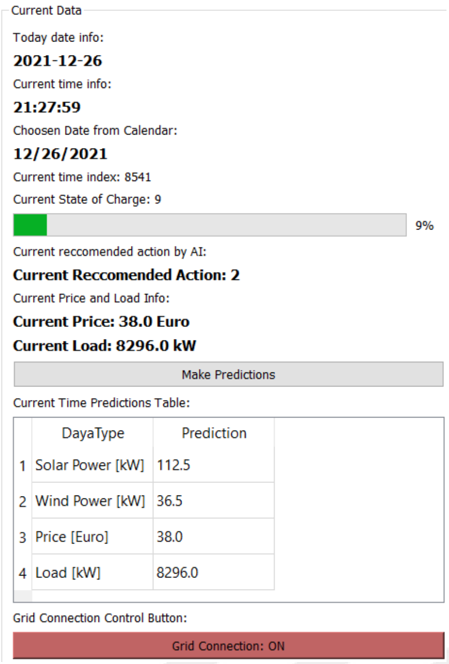


Figure 5.2 : Making Predictions and action reccomendation part.

As it is known, EMS uses an AI-based decision-making agent. This agent, which has been trained correctly, learned correctly and has been shown to make successful decisions, generates an action by using the current information about the microgrid. It uses the forecasts and SoC value of ESS when making this decision. The action decision made by the agent is also presented to the user in this part. Lastly, the user can disconnect the microgrid from the utility grid by using the *'Grid Connection: On/Off'* button located in this section.

5.2.2 Load and price page

Acquiring pricing information is crucial for the profitability of microgrids, while managing the amount of load demanded from the microgrid is vital for energy management. To this end, a dedicated page is provided to enable users to access and analyze pricing and load demand data. This page allows the user to review price data from the previous year, in conjunction with the forecast results generated by the Bi-LSTM-based prediction model for that year. This provides an opportunity to assess the performance of the prediction models, and to evaluate its behavior in response to sharp fluctuations. The page is divided into two parts, with the left-hand layout dedicated to price data analysis, and the right-hand layout allocated to load demand analysis. The page layout is illustrated in Figure 5.3.

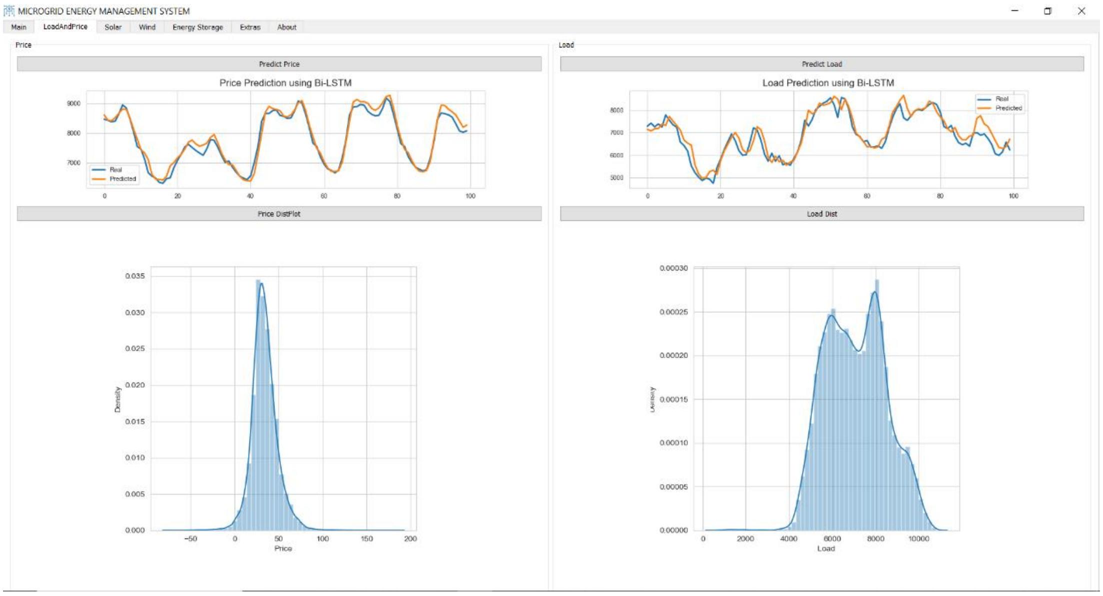


Figure 5.3 : Load and Price Page

The "*Predict Price*" button is located in the top-left corner of the page, enabling the user to generate predictions and plots that compare actual price data for the same hundred days. The "*Price DistPlot*" button generates a distribution plot of prices for the year, providing users with the ability to examine price changes. Similarly, on the right-hand side of the page, users can generate load demand predictions and compare them to actual load demand data from the same hundred days of the previous year. The "*Load DistPlot*" button displays changes in load demand over the year. Crucially, when the prediction buttons are activated, the GUI does not display pre-existing graphics, but instead generates forecasts using the appropriate forecast models each time it is pressed, allowing the user to evaluate the current performance of the prediction models. This functionality enables users to take appropriate measures to manage their microgrid effectively.

The best performing dynamic price and load demand prediction models, whose designs, performances, and comparison results were explained in the previous section, are utilized on this page. The prediction models using the Bi-LSTM method, which outperformed other DL models in both prediction studies, are assigned as the predictor of the designed GUI.

5.2.3 Solar power page

During the design of the AIEMS, the simulation model of the microgrid that was created includes a solar power plant. Similar to the proposed simulation model for many microgrids, it includes solar panels or a solar power plant of varying sizes. Presented page is dedicated to the prediction models for the power generated by the solar power plant. The solar page presents the performance of the solar power prediction models that were previously presented in the previous chapter to the user. The solar page consists of four sections, and its overall appearance is shown in Figure 5.4. The left part of the page enables the analysis of one of the meteorological data by selecting the prediction model and executing the prediction and analysis of the desired meteorological data. This section includes both widely used ML models and new GBM-based prediction models. The user selects the model whose performance they want to analyze and clicks on the "*Prediction Results*" button. Then, AIEMS runs the selected model and obtains the prediction results. The real and predicted results for randomly selected 100 hours from the test dataset using the trained model are visualized and displayed in a plot created in the upper middle part of the page.

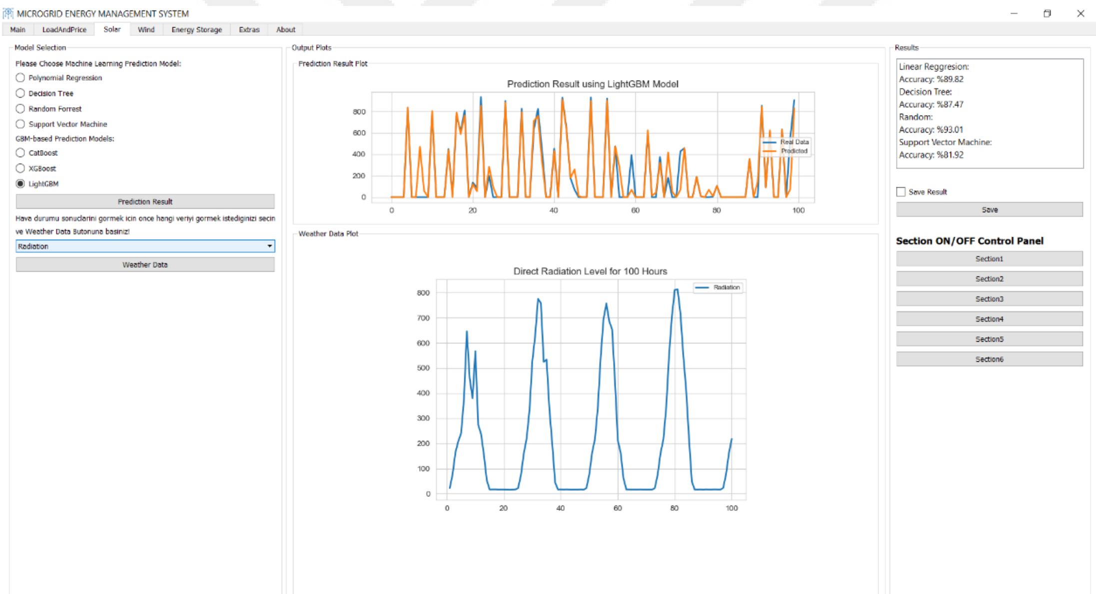


Figure 5.4 : Overview of the *Solar* tab

Additionally, the left layout of the page provides the user with the opportunity to analyze nine different meteorological features. The user selects the desired meteorological data and clicks on the "*Weather Data*" button. The change in the selected data is then presented to the user in the middle bottom part of the page. In this way, the microgrid manager gains insight into both the prediction models and the analysis of meteorological data. Finally, the accuracy and error values of the prediction models are presented to the user in the right section of the page. If desired, the user can save these results as a .txt file format to their computer using the "*Save*" button.

Furthermore, in this right section, the user can enable or disable the solar panels that are divided into sections.

5.2.4 Wind power page

The simulation model of the microgrid includes wind turbines in addition to the solar power plant. The analysis of the power to be obtained from these turbines facilitates operational decisions for the user. This page, which includes the analysis of wind turbines, is added to the AIEMS interface under the name 'Wind'. This wind page consists of four layouts, and the overall appearance of the page is given in Figure 5.5. First, the user selects the method of the prediction model from the left side of the page. These methods are used for estimating the power to be obtained from wind turbines. The SVM, XGBoost, and LightGBM models, which provide the best results for this prediction study, are added to the interface on this page. Firstly, the user selects the desired prediction model from here and presses the *'Predict'* button. Then, the predicted values produced by the prediction model and the actual values are visualized together in the layout that has been added from top to bottom of the page. The proximity between the actual and predicted values can be analyzed for randomly selected hundred-hour test data.

Figure 5.5 : Overview of the *Wind* tab

In addition, the error outputs, which are performance metrics of all models such as 'MSE', 'RMSE', and 'MAE', are also presented to the user on the right side of the page. Moreover, accuracy results are also displayed. The user can easily see which prediction model provides better results and analyze the models. Furthermore, with the 'Plot Heatmap Graph' button located on the left side of the page, the user can visualize the mechanical data of the turbine, wind data, and output of active and reactive power in the form of a heatmap. The heatmap created by AIEMS is presented to the user in the upper middle part of this page. The user can analyze from here which data will affect the production rate to what extent. Finally, with the turbine buttons located on the right side of the page, the user can activate or deactivate the turbines within the microgrid.

5.2.5 Energy storage page

One of the most important ways to ensure profitability, energy quality, and continuity in a microgrid is to use an energy storage system. In this context, it is assumed that the simulation model of the microgrid in which the proposed solutions in this thesis are tested has a large-capacity next-generation energy storage unit. Additionally, the AI-based Energy Management System proposes additional operational control actions to the human manager's decisions. These proposals are made using an AI-based controller. The page designed for this GUI is dedicated to the performance analysis of these controllers under the name "*Energy Storage*". The appearance of the page consisting of three sections is given in Figure 5.6. Within the scope of this thesis, five RL-based control agents have been designed and trained. The performance of these agents is presented to the user on this page.

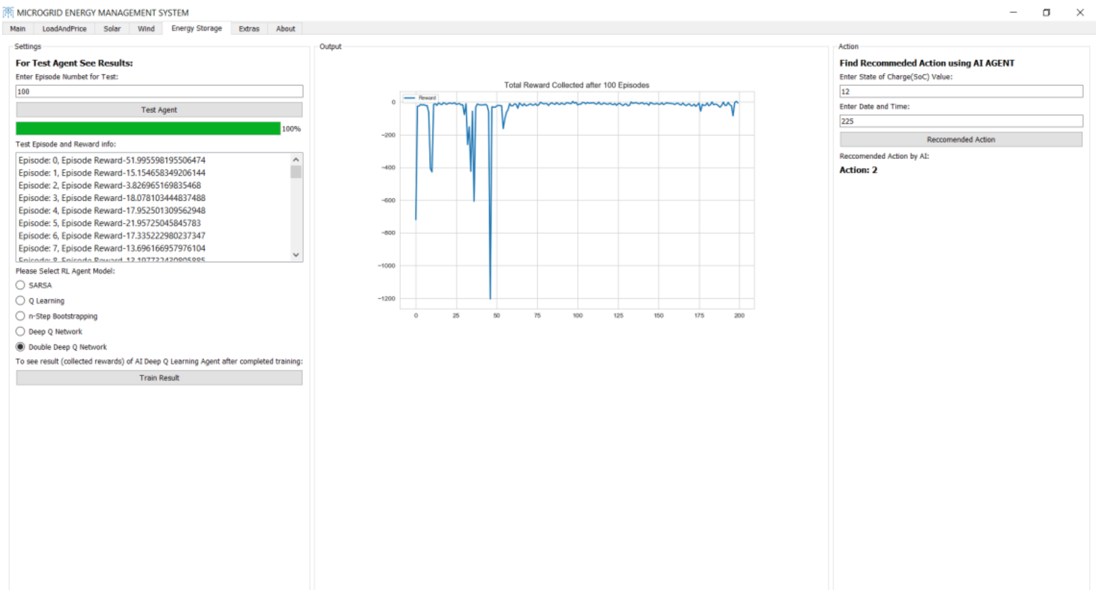


Figure 5.6 : Energy Storage page

Firstly, the user selects the RL agent whose performance he/she wants to analyze from the left side of the page, enters the number of sections he/she wants to see the performance of, and clicks the "*Test Agent*" button. This button tests the pre-trained RL agents as many times as the number of sections entered and lists the reward collected by the relevant agent in each section. After the testing process is completed, the middle part of the page shows the user the reward collection performance of the agent in a graph. Hence, the user can have an idea about how well decision-makers produce good decisions. Moreover, the user can see the performance of the selected agent during the training process by clicking the "*Train Result*" button on the left side of the page. This button presents the user with a graph of the training process of the selected agent. Finally, in the rightmost section of the page, the user can view the control action recommendation of the selected agent. In this section, the user first selects a day and time, then sets a hypothetical SoC value. After that, the user clicks the "Recommended Action" button and can see the action selected by the agent for that time and SoC value. Here, the agent uses the prediction modules explained in previous sections and decides on the control action by taking the entered SoC value into consideration. Ultimately, the user can review and analyze the RL agents used by the AIEMS on this page.

5.2.6 Extra and about pages

Various energy sources, different types of energy storage systems, and new components can be added to the microgrid. Considering these additions, a page named '*Extras*' is included in the GUI of the proposed AIEMS. This page is prepared in four layout forms. The main purpose of adding this page is to design AIEMS to be open to innovations and improvements. New AI methods that will benefit the energy management of the microgrid can be added to this page. In addition, different data obtained from the microgrid can be analyzed on this page. This page design makes it suitable for updating. The last page is created under the name *'About'*. This page provides information to the user of the interface about this thesis and the university where the work was done. It briefly explains the models, methods, and capabilities of this proposed AIEMS.

6. CONCLUSIONS AND RECOMMENDATIONS

The awareness regarding the significance of efficient utilization of energy resources and the prevention of their reckless depletion is increasingly growing. The reduction of carbon footprint holds crucial importance for preserving the environment for future generations. In terms of electrical energy, while it is produced in large capacities at a specific point using nuclear or thermal power plants, it is not consumed at the same point to the same extent. Instead, electrical energy is transferred over long distances using high power lines, which results in significant energy waste. To mitigate this issue, efforts are made to prevent wastage through grid systems where production and consumption occur within the same region. Smart grids are gaining popularity on a city or country level, while microgrids are preferred for smaller communities. Additionally, nanogrids that operate at a household level hold the promise of more efficient energy usage.

The thesis is built on the concept of microgrids, which aim to utilize distributed generation resources, generators, the main electrical grid, and energy storage systems together. Microgrids focus on optimizing the usage of all these resources, promising energy efficiency. Microgrids are suitable for places such as university campuses, military bases, small towns, or districts, providing efficient energy use based on the needs and possibilities of these areas. Microgrids are tailored according to the available resources and requirements of the area. For example, while some microgrids may not have a connection to the main electrical grid, others may have one. Similarly, while some microgrids heavily utilize energy storage units, others may not even have one. During the installation phase, microgrids are sized according to the available resources and requirements of the area, and optimization algorithms are often utilized during this process.

In this thesis, the focus is on the real-time management of a microgrid rather than its sizing through deterministic or stochastic optimization techniques. This thesis proposes an artificial intelligence-based energy management system that effectively utilizes the most up-to-date artificial intelligence techniques for the most efficient and cost-effective use of energy. An energy management system that can be used for many microgrids is designed. For this design, a simulation model of a microgrid is created. Many microgrids are examined to create a simulation model that is suitable for a university campus. This simulation model includes a solar power plant, wind turbines, energy storage units, and grid connection. The load demand is determined according to the needs of a medium-sized university campus. Additionally, it is assumed that this microgrid is located in an area where dynamic pricing is applied.

The AIEMS system incorporates prediction models that utilize machine learning and deep learning techniques. The mathematical foundations of these prediction models are explained in Chapter 2, while the quantitative results are presented in Chapter 4, and the qualitative results are given under the next heading. Furthermore, data-driven control is a popular topic in addition to classical control methods today. This machine learning-based control paradigm provides data-based control solutions without the need for a mathematical model of the system. Reinforcement learning, which is one of the three main branches of machine learning, offers control methods that work in this way. In this thesis, a deep reinforcement learning-based controller is proposed within the AIEMS system. The principles of the developed reinforcement learning agents are presented in Chapter 3, while the performance outputs of the agents are given in Chapter 4. The qualitative results of the design of an RL-based controller are presented in section 6.2.

6.1 Predictions for AIEMS

The aim of the energy management system being designed is to incorporate renewable energy sources into a microgrid. The simulation model of this microgrid includes a 70kW AC solar power plant and wind turbines with an active power of 20kW. The designed AIEMS predicts how much energy the solar power plant will produce using meteorological data and provides this information to both the user and the deep reinforcement learning agents. To this end, a total of seven models are developed using four commonly used ML methods and three new GBM-based prediction methods for solar power prediction. The results of these models show that GBM-based techniques outperform classical ML techniques, with the LightGBM and XGBoost algorithms producing the best results. For wind turbine energy generation, these two successful methods are also used along with SVM, and again these two models produce highly successful results.

Furthermore, AIEMS also needs to know the instantaneous changing load demand and the energy price to be applied at that time for a microgrid. In this context, deep learning-based models are developed for both dynamic price and microgrid load demand prediction. Considering that both of these data change over time, DL techniques with memory cell structures are used. Eight prediction models are developed for two problems using RNN, GRUs, LSTM, and Bi-LSTM methods. Bi-LSTM technique outperforms the others for both load demand prediction and dynamic price prediction. The designed AIEMS provides the results of these prediction models to the user through the interface and provides this information to the DRL agents for optimizing themselves.

6.2 AI-controller of AIEMS for Microgrids

The energy management system of a microgrid requires a decision maker, which can be either a human responsible for the microgrid's operations or an automated system. In this thesis, an AI-based controller is proposed in addition to the user. This controller is built using a data-driven control paradigm. Five RL agents are designed using RL techniques, which use environment models specifically designed for the microgrid's simulation model. RL agents are designed using tabular methods such as SARSA, Q Learning, and *n*-Step Bootstrapping algorithms, as well as DRL-based DQN and DDQN agents designed for AIEMS. Additionally, this thesis proposes two unique reward strategies specifically designed for AIEMS. The control agents are trained over thousands of episodes, allowing them to optimize themselves. It has been shown that all agents have learned the microgrid's energy management. Among TD-based methods, the *n*-Step Bootstrapping agent uses energy most efficiently and profitably, while among DRL algorithms, the DDQN algorithm outperforms all others in both efficiency and profitability. The DDQN agent, which yields the best results, is used to determine actions on the AIEMS graphical user interface's homepage, while the other agents' performances are presented to the user on the energy storage page of the interface.

6.3 Future Works and Recommendations

All predictive models and AI-based controllers created are gathered in a user interface specifically designed for AIEMS. This GUI, consisting of seven pages, presents the user with various data related to the microgrid. Additionally, the interface provides the user with the ability to obtain meteorological data and presents control actions that can be applied based on all these parameters. The designed AIEMS predicts future power generation, load demand, and dynamic pricing at a desired time and presents it to the user. The performances of the AI models are presented to the operator in detail on relevant pages.

This designed AIEMS can be updated with AI prediction techniques that will enter the literature in the future. In addition, the RL controller can be upgraded with newly created data-driven control techniques. AI controllers can be optimized to a greater extent with a detailed knowledge of the load profile. It is evident that electric vehicles will replace conventional vehicles in the future. In this context, the designed AIEMS can be improved and updated by considering the needs and effects of electric vehicles that will be part of the microgrid.

REFERENCES

[1] **Cagnano, A., De Tuglie E. & Manceralla, P.** (2020). Microgrids: Overview and guidelines for practical implementations and operation, *Applied Energy, 258,* 114039.

[2] **Vera, Y. E. G., Dufo-Lopez, R. & Bernal-Agustin, J.L.** (2019). Energy Management in Microgrids with Renewable Energy Sources: A Literature Review, *Applied Sciences, 9*, 38–54.

[3] **Marino, C. A. & Maruffuzaman, M.** (2020). A microgrid energy management system based on chance-constrained stochastic optimization and big data analytics, *Computers & Industrial Engineering, 143*, 106392.

[4] **Roslan, M. F., Hannan, M. A. & Ker, P.J. et al.** (2022). Microgrid control methods toward achieving sustainable energy management: A bibliometric analysis for future directions, *Journal of Cleaner Production, 348*, 131340.

[5] **Gao, T. & Lu, W.** (2021). Machine learning toward advanced energy storage devices and systems, *iScience, 101-1*, 101936.

[6] **Aslam, S., Herodotou, H. & Mohsin, S.M. et al.** (2021). A survey on deep learning methods for power load and renewable energy forecasting in smart microgrids, *Renewable & Sustainable Energy, 144*, 110992.

[7] **Arunkumar, A. P., Kuppusammy, S. & Muthusammy, S. et al.** (2022). An extensive review on energy management system for microgrids, *Energy Sources, 44:2*, 4203–4228.

[8] **Brunton, S. L. & Kutz, J. N.** (2019). *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*, Cambridge *University Press, Cambridge, UK.*

[9] **Bao, G., & Xu, R.** (2023). A Data-Driven Energy Management Strategy Based on Deep Reinforcement Learning for Microgrid Systems, *Cognitive Computation, 119* 10106-3.

[10] **Sutton, R. S., & Barto, A. G.** (2018). *Reinforcement Learning: An Introduction, The MIT Press, Cambridge, MA, US.*

[11] **Arwa, E. O., & Folly, K. A.** (2020). Reinforcement Learning Techniques for Optimal Power Control in Grid-Connected Microgrids: A Comprehensive Review, *IEEE Access, 8,* 208992-209007.

[12] **Kofinas, P., Vouros, G. & Dounis, A. I.** (2018). Energy management in solar microgrid via reinforcement learning using fuzzy reward, *Advances in Building Energy Research, 12:1*, 97–115.

[13] **Foruzan, E., Soh, L. & Asgarpoor, S.** (2018). Energy Reinforcement Learning Approach for Optimal Distributed Energy Management in a Microgrid, *IEEE Transactions on Power Systems, 33:5*, 5749–5758.

[14] **Zhou, K., Zhou, K. & Yang, S.** (2022). Reinforcement learning-based scheduling strategy for energy storage in microgrid, *Journal of Energy Storage*, ***51***, 104379.

[15] **Alabdullah, M. H. & Abido, M.** (2022). Microgrid energy management using deep Q-network reinforcement learning, *Alexandria Engineering Journal*, *61*, 9069-9078.

[16] **Guo, C., Wang, X., Zheng, Y. & Zhang, F.** (2021). Optimal energy management of multi-microgrids connected to distribution system based on deep reinforcement learning, *International Journal of Electrical Power and Energy Systems*, *131*, 107048.

[17] **Aksoy, N. & Genc, I.** (2023). Design of Graphical User Interface for Artificial Intelligence-Based Energy Management System for Microgrids, *Electrica*, ***23***, 202-211.

[18] **Hu, C., Cai, Z. & Zhang, Y., et al.** (2022). A soft actor-critic deep reinforcement learning method for multi-timescale coordinated operation of microgrids, *Protection and Control of Modern Power Systems, 7:29*, 2-10.

[19] **He, X., Ge, S. & Liu, H., et al.** (2023). Frequency regulation of multi-microgrid with shared energy storage based on deep reinforcement learning, *Electric Power Systems Research, 214*, 108962.

[20] **Yu, L., Xie, W. & Zou, Y., et al.** (2020). Deep Reinforcement Learning for Smart Home Energy Management, *IEEE Internet of Things Journal, 7:4*, 2751-2762.

[21] **Liu, Y., Zhang, D. & Gooi, H. B.** (2020). Optimization Strategy Based on Deep Reinforcement Learning for Home Energy Management, *CSEE Journal of Power And Energy Systems*, ***6:3***, 572-582.

[22] **Si, C., Tao, Y., Qui, J. & Lai, S., et al.** (2021). Deep reinforcement learning based home energy management system with devices operational dependencies, *International Journal of Machine Learning and Cybernetics, 12*, 1687-1703.

[23] **Kodama, N., Harada, T. & Miyazaki, K.** (2021). Home Energy Management Algorithm Based on Deep Reinforcement Learning Using Multistep Prediction, *IEEE Access, 9*, 153108-153115.

[24] **Han, G., Lee, S. & Lee, J., et al.** (2021). Deep-learning- and reinforcement-learning-based profitable strategy of a grid-level energy storage system for the smart grid, *Journal of Energy Storage*, *41*, 102868.

[25] **Chuang, Y., & Chiu, W.** (2022)., Deep Reinforcement Learning Based Pricing Strategy of Aggregators Considering Renewable Energy, *IEEE Transactions on Emerging Topics in Computational Intelligence, 6:3*, 499-508.

[26] **Ozcelik, M., Kesici, M., Aksoy, N. & Genc, I.** (2022). Reinforcement learning-based demand-sidemanagement by smart charging of electric vehicles, *Electrical Engineering*, *104*, 3933-3942.

[27] **Hussain, A., Bui, V. & Kim, H.** (2022). Deep reinforcement learning-based operation of fast charging stations coupled with energy storage system, *Electric Power System Research, 210*, 108087.

[28] **Park, K., & Moon, I.** (2022). Multi-agent deep reinforcement learning approach for EV charging scheduling in a smart grid, *Applied Energy, 328*, 120111.

[29] **Cao, J., Harrold, D. & Fan, Z., et al.** (2020). Deep Reinforcement Learning-Based Energy Storage Arbitrage with Accurate Lithium-Ion Battery Degradation Model, *IEEE Transactions on Smart Grid, 11:5*, 4513-4521.

[30] **Huang, B., & Wang, J.** (2021). Deep-Reinforcement-Learning-Based Capacity Scheduling for PV-Battery Storage System, *IEEE Transactions on Smart Grid, 12:3*, 2272-2283.

[31] **Harrold, D.J.B., Cao, J. & Fan, S.** (2022). Data-driven battery operation for energy arbitrage using rainbow deep reinforcement learning, *Energy, 238*, 121958.

[32] **Liu, F., Liu, Q. & Tao, Q., et al.** (2023). Deep reinforcement learning based energy storage management strategy considering prediction intervals of wind power, *Journal of Electrical Power and Energy Systems, 145*, 108608.

[33] **Li, W., Cui, H. & Nemeth, T., et al.** (2021). Deep reinforcement learning-based energy management of hybrid battery systems in electric vehicles, *Journal of Energy Storage, 36*, 102355.

[34] **Wang, J., Zhou, J. & Zhao, W.** (2022). Deep reinforcement learning based energy management strategy for fuel cell/battery/supercapacitor powered electric vehicle, *Green Energy and Intelligent Transportation, 1*, 100028.

[35] **Zheng, C., Li, W. & Li, W., et al.** (2022). A Deep Reinforcement Learning-Based Energy Management Strategy for Fuel Cell Hybrid Buses, *International Journal of Precision Engineering and Manufacturing-Green Technology, 9*, 885–897.

[36] **Gorostiza, F. S., & Gonzalez-Longatt, F.** (2020). Deep Reinforcement Learning-Based Controller for SOC Management of Multi-Electrical Energy Storage System, *IEEE Transactions on Smart Grid, 11:6*, 5039-5050.

[37] **Kim, H., & Lee, D.** (2021). Probabilistic Solar Power Forecasting Based on Bivariate Conditional Solar Irradiation Distributions, *IEEE Transactions on Sustainable Energy, 12:4*, 2031-2041 .

[38] **Al-Dahidi, S., Ayadi, O. & Alrbai, M., et al.** (2019). Ensemble Approach of Optimized Artificial Neural Networks for Solar Photovoltaic Power Prediction, *IEEE Access, 7*, 81741-81758.

[39] **Wen, S., Zhang, C. & Lan, H., et al.** (2021). A Hybrid Ensemble Model for Interval Prediction of Solar Power Output in Ship Onboard Power Systems, *IEEE Transactions on Sustainable Energy, 12:1*, 14-24.

[40] **Mishra, M., Dash, P. B. & Nayak, J., et al.** (2020). Deep learning and wavelet transform integrated approach for short-term solar PV power prediction, *Measurement, 166*, 108250.

[41] **Gao, Y., Miyata, S. & Akashi, Y.** (2022). Interpretable deep learning models for hourly solar radiation prediction based on graph neural network and attention, *Applied Energy, 321*, 119288.

[42] **Izgi, E., Oztopal, A., Kaymak, M. K. & Sahin, A. D.** (2012). Short–mid-term solar power prediction by using artificial neural networks, *Solar Energy, 86*, 725-733.

[43] **Chai, M., Xia, F. & Hao, S., et al.** (2019). PV Power Prediction Based on LSTM With Adaptive Hyperparameter Adjustment, *IEEE Access, 7*, 115473-115486.

[44] **Tang, N., Mao, S., Wang, Y. & Nelms, R. M.** (2018). Solar Power Generation Forecasting With a LASSO-Based Approach, *IEEE Internet of Things Journal, 5:2*, 1090-1099.

[45] **Zhang, T., Taheri, R., Zareipour, H. & Wood, D.** (2015). Day-Ahead Power Output Forecasting for Small-Scale Solar Photovoltaic Electricity Generators, *IEEE Transactions on Smart Grid, 6:5*, 2253-2262 .

[46] **Cheng, L., Zang, H. & Wei, Z., et al.** (2022). Short-term Solar Power Prediction Learning Directly from Satellite Images With Regions of Interest, *IEEE Transactions on Sustainable Energy, 13:1*, 629-639.

[47] **Ghimire, S., Nguyen-Huy, T. & Deo, R. C., et al.** (2022). Efficient daily solar radiation prediction with deep learning 4-phase convolutional neural network, dual stage stacked regression and support vector machine CNN-REGST hybrid model, *Sustainable Materials and Technologies, 32*, e00429.

[48] **Bae, K. Y., Jang, H. S. & Sung, D. K.** (2017). Hourly Solar Irradiance Prediction Based on Support Vector Machine and Its Error Analysis, *IEEE Transactions on Power Systems, 32:2*, 935-945 .

[49] **Singla, P., Duhan, M. & Saroha, S.** (2022). An ensemble method to forecast 24-h ahead solar irradiance using wavelet decomposition and BiLSTM deep learning network, *Earth Science Informatics, 15*, 291-306.

[50] **Ma, Z., Chang, H. & Sun, Z., et al.** (2020, March). Very Short-Term Renewable Energy Power Prediction Using XGBoost Optimized by TPE Algorithm, In *2020 4th International Conference on HVDC (HVDC), China*, 1236-1241.

[51] **Deng, Z., Ye, A. & Zhong, J., et al.** (2022). Bagging–XGBoost algorithm based extreme weather identification and short-term load forecasting model, *Energy Reports, 8*, 8662-8674.

[52] **Yao, X., Fu, X. & Zong, C.** (2022). Short-Term Load Forecasting Method Based on Feature Preference Strategy and LightGBM-XGboost, *IEEE Access, 10*, 75257-75268.

[53] **Ju, Y., Sun, G. & Chen, Q., et al.** (2019). A Model Combining Convolutional Neural Network and LightGBM Algorithm for Ultra-Short-Term Wind Power Forecasting, *IEEE Access, 7*, 28309-28318.

[54] **Ren, J., Yu, Z. & Gao, G., et al.** (2021, December). A CNN-LSTM-LightGBM based short-term wind power prediction method based on attention mechanism, In *2021 The 2nd International Conference on Power Engineering (ICPE 2021), China*, 437-443.

[55] **Xiang, W., Xu, P. & Fang, J., et al.** (2022). Multi-dimensional data-based medium- and long-term power-load forecasting using double-layer CatBoost, *Energy Reports, 8*, 8511-8522.

[56] **Sun, L., Liu, T. & Xie, Y., et al.** (2021). Real-time power prediction approach for turbine using deep learning techniques, *Energy*, *233*, 121130.

[57] **Almutairi, A. & Alrumayh, O.** (2022). An intelligent deep learning based prediction model for wind power generation, *Computers and Elecrical Engineering, 101*, 108000.

[58] **Deepa, S. N. & Banerjee, A.** (2022). Intelligent Neural Learning Models for Multi-step Wind Speed Forecasting in Renewable Energy Applications, *Journal of Control, Automation and Electrical Systems, 33*, 881-900.

[59] **Khan, A. R., Mahmood, A. & Khan, Z. A., et al.** (2016). Load forecasting, dynamic pricing and DSM in smart grid: A review, *Renewable and Sustainable Energy Reviews, 54*, 1311-1322.

[60] **Crisostomi, E., Gallicchio, C. & Micheli, A., et al.** (2015). Prediction of the Italian electricity price for smart grid applications, *Neurocomputing, 170*, 286-295.

[61] **Shah, I., Bibi, H. & Ali, S., et al.** (2020). Forecasting One-Day-Ahead Electricity Prices for Italian Electricity Market Using Parametric and Nonparametric Approaches, *IEEE Access, 8*, 123104-123113.

[62] **Alkawaz, A. N., Abdellatif, A. & Kanesan, J., et al.** (2022). Day-Ahead Electricity Price Forecasting Based on Hybrid Regression Model, *IEEE Access, 10*, 108021-108033.

[63] **Bisoi, R., Dash, P. K. & Das, P. P.** (2020). Short-term electricity price forecasting and classification in smart grids using optimized multikernel extreme learning machine , *Neural Computing and Applications, 32*, 1457-1480.

[64] **Tschora, L., Pierre, E., Plantevit, M. & Robardet, C.** (2022). Electricity price forecasting on the day-ahead market using machine learning, *Applied Energy, 313*, 118752.

[65] **Yang, Z., Ce, L. & Lian, L.** (2017). Electricity price forecasting by a hybrid model, combining wavelet transform, ARMA and kernel-based extreme learning machine methods, *Applied Energy, 190*, 291-305.

[66] **Pourdaryaei, A., Mokhlis, H. & Illias, H. A., et al.** (2019). Short-Term Electricity Price Forecasting via Hybrid Backtracking Search Algorithm and ANFIS Approach, *IEEE Access, 7*, 77674-77691.

[67] **Pany, P. K. & Ghoshal, S. P.** (2015). Dynamic electricity price forecasting using local linear wavelet neural network, *Neural Computing and Applications, 26*, 2039-2047.

[68] **Dash, S. K., Bisoi, R. & Dash, P. K.** (2016). A hybrid functional link dynamic neural network and evolutionary unscented Kalman filter for short-term electricity price forecasting, *Neural Computing and Applications, 27*, 2123-2140.

[69] **Ranrab, Li** (2023). Forecasting energy spot prices: A multiscale clustering recognition approach, *Resources Policy, 81*, 103320.

[70] **Lago, J., Ridder, F. D. & Schutter, B. D.** (2018). Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms, *Applied Energy, 221*, 386-405.

[71] **Zhou, S., Zhou, L. & Mao, M., et al.** (2019). An Optimized Heterogeneous Structure LSTM Network for Electricity Price Forecasting, *IEEE Access, 7*, 108161-108173.

[72] **Zhang, R., Li, G. & Ma, Z.** (2020). A Deep Learning Based Hybrid Framework for Day-Ahead Electricity Price Forecasting, *IEEE Access, 8*, 143423-143436.

[73] **Yang, G., Du, S., Duan, Q. & Su, J.** (2022). Short-term Price Forecasting Method in Electricity Spot Markets Based on Attention-LSTM-mTCN, *Journal of Electrical Engineering & Technology, 17,* 1009 - 1018.

[74] **Xiong, X. & Qing, G.** (2023). A hybrid day-ahead electricity price forecasting framework based on time series, *Energy, 264*, 126099.

[75] **Veeramsetty, V., Reddy, K. R. & Santhosh, M., et al.** (2022). Short-term electric power load forecasting using random forest and gated recurrent unit, *Electrical Engineering, 104,* 307-329

[76] **Mamerzadeh, G. & Keynia, F.** (2021). Short-term electricity load and price forecasting by a new optimal LSTM-NN based prediction algorithm, *Electric Power System Research, 192*, 106995.

[77] **Guo, X., Zhao, Q., Zheng, D., Ning, Y. & Gao, Y.** (2020). Reinforcement learning-based scheduling strategy for energy storage in microgrid, *Energy Reports*, *6*, 1046-1053.

[78] **Cervantes, J., & Garcia-Lamont, F., et al.** (2020). A comprehensive survey on support vector machine classification: Applications, challenges and trends, *Neurocomputing, 408,* 189-215.

[79] **James, G., Written, D., Hastie, T. & Tibshirani, R.** (2021). *An Introduction to Statical Learning. Springer, New York .*

[80] **Quinlan, J. R.** (1986). *Introduction to decision trees, Springer, New York.*

[81] **Quinlan, J. R.** (1993). *C4.5 Programs for Machine Learning, Elsevier Press, Amsterdam, London, New York.*

[82] **Breiman, L., Freidman, J. H, Olshen, R. A & Stone, J. H.** (1984). *Classification and Regression Trees, Chapman Hall/CRS, New York*.

[83] **Gao, K., Wang, Y. & Ma, L.** (2022). Belief Entropy Tree and Random Forest: Learning from Data with Continuous Attributes and Evidential Labels, *Entropy,* 24:5**,** 605.

[84] **Friedman, J. H.** (2001). Greedy Function Approximation: A Gradient Boosting Machine, *The Annals of Statistics*, *29:5*, 1189-1232.

[85] **Url-1** <*https://xgboost.readthedocs.io/* >, date retrieved 19.04.2023.

[86] **Ke, G., Meng, Q. & Finley, Y., et al** (2017, December). LightGBM: A Highly Efficient Gradient Boosting Decision Tree, *in NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing,* 3149–3157.

[87] **Prokhorenkova, L., Gusev, G. & Vorobev, A., et al.** (2018, December). CatBoost: unbiased boosting with categorical features, *in NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing,* 6639–6649.

[88] **LeCun, Y., Bengio, Y. & Hinton, G.** (2015). Deep Learning, *Nature, 521, 436–444*.

[89] **Hopfield, J. J.** (1982). Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences, 79:8*, 2554-2558.

[90] **Elman, J. L.** (1990). Finding structure in time, *Cognitive Science, 14:2*, 179-211.

[91] **Jordan, M. I. & Rumelhart, D. E.** (1992). Forward Models: Supervised Learning with a Distal Teacher, *Cognitive Science, 16:3*, 307-354.

[92] **Lipton, Z. C., Berkowitz, J. & Elkan, C.** (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning, *arXiv,* 1506.00019.

[93] **Hochreiter, S. & Schmidhuber, J.** (1997). Long Short-Term Memory, *Neural Computation, 9:8,* 1735-1780.

[94] **Schuster, M. & Paliwal, K. K.** (1997). Bidirectional recurrent neural networks, *IEEE Transactions on Signal Processing, 45:11*, 2673-2681 .

[95] **Cho, K., Merrienboer, B. V., Bahdanau, D. & Bengio, Y.** (2014). On the Properties of Neural Machine Translation: Encoder-Decoder Approaches, *CoRR, 09*, 12-59.

[96] **Mnih, V., Kavukcuoglu, K. & Silver, D., et al.** (2015). Human-level control through deep reinforcement learning, *Nature, 518,* 529-533.

[97] **Lillicrap, T. P., Hunt, J. J. & Pritzel, A., et al.** (2015). Continuous control with deep reinforcement learning, *arXiv, 9*, 1509.02971.

[98] **Schulman, J., Wolski, F. & Dahiriwall, P., et al.** (2017). Proximal Policy Optimization Algorithms, *arXiv, 7*, 1707.06347.

[99] **Hasselt, H. V., Guez, A. & Silver, D.** (2015). Deep Reinforcement Learning with Double Q-learning, *arXiv, 9*, 1509.06461.

[100] **Url-2** <*https://opendata.edp.com/*>, date retrieved 29.06.2019.

[101] **Url-3** <*https://data.open-power-system-data.org/time_series/*>, date retrieved 19.04.2022.

[102] **Url-4** <*https://lightgbm.readthedocs.io/*>, date retrieved 12.05.2022.

[103] **Url-5** <*https://catboost.ai/en/docs/* >, date retrieved 13.05.2022.

CURRICULUM VITAE

Name Surname : Necati AKSOY

EDUCATION :

- **B.Sc.** : 2011, Kırıkkale University, Faculty of Engineering and Architecture, Department of Electrical and Electronics Engineering
- **M.Sc.** :2015, Temple University, Faculty of Engineering, Department of Electrical and Computer Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2020-Present Lecturer in department of Electronics and Automation at Yozgat Bozok University
- 2016-2020 Research assistant in department of Electrical-Electronics at Kilis 7 Aralik University
- 2011-2015 Graduate student supported by TÜBİTAK and the Ministry of National Education scholarship for abroad graduate education.

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Aksoy, N.** and Genc, V. M. I. 2023. Energy Storage Management for Microgrids Using *n*-Step Bootstrapping Gestion du stockage de l'énergie pour les micro-réseaux à l'aide d'un Bootstrapping en *n* étapes, *IEEE Canadian Journal of Electrical and Computer Engineering*, Vol 46. No 2. 21-31.
- **Aksoy, N.** and Genc, V. M. I. 2023. Predictive models development using gradient boosting based methods for solar power plants, *Journal of Computational Science*, 67(2023), 101958.
- **Aksoy, N.** and Genc, V. M. I. 2023. Design of Graphical User Interface for Artificial Intelligence-Based Energy Management System for Microgrids, *Electrica*, 23(2), 202-211.
- **Aksoy, N.** and Genc, V. M. I. (June 2022): Development and Analysis of Prediction Models for an AI-Based Energy Management System for Microgrids. In International Graduate Research Symposium – IGRS'22. June 1-3, 2022 Istanbul, Turkey.

OTHER PUBLICATIONS, PRESENTATIONS AND PATENTS:

- Ozcelik, M., Kesici, M., **Aksoy, N.** and Genc, V.M.I. 2022. Reinforcement learning-based demand-sidemanagement by smart charging of electric vehicles, *Electrical Engineering*, 104, 3933-3942.
- Gurkan, C., Mohammadpourfard M., **Aksoy, N.** and Genc, V.M.I**.** (May 2023). Deep Learning - Based FDIA Detection with High Penetration Rate of DERs. In 2nd International Graduate Research Symposium – IGRS'23. May 16-18, 2023, Istanbul, Turkey.