ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Ένα Συνεργατικό Σχήμα Μηχανικής Μάθησης για Εξαγωγή Γνώσης στο Ευφυές Ηλεκτρικό Δίκτυο

ΧΑΡΙΟΠΟΛΙΤΗΣ ΑΝΔΡΕΑΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

ΚΟΛΟΜΒΑΤΣΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ
Αναπληρωτής Καθηγητής

Λαμία 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Ένα Συνεργατικό Σχήμα Μηχανικής Μάθησης για Εξαγωγή Γνώσης στο Ευφυές Ηλεκτρικό Δίκτυο

ΧΑΡΙΟΠΟΛΙΤΗΣ ΑΝΔΡΕΑΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

ΚΟΛΟΜΒΑΤΣΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ
Αναπληρωτής Καθηγητής

Λαμία 2025

UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

A Collaborative Machine Learning Scheme for Knowledge Extraction in the Smart Electric Grid

CHARIOPOLITIS ANDREAS

FINAL THESIS

ADVISOR

KOLOMVATSOS KONSTANTINOS
Associate Professor

Lamia 2025

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία: 11/06/2025

Ο – ~~Η~~ Δηλών

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση
του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων
σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΕΗΔ = Ευφυές Ηλεκτρικό Δίκτυο
ΣΣΜΜ = Συνεργατικό Σχήμα Μηχανικής Μάθησης
OSGP = Open Smart Grid Protocol
AKA = Also Known As
ML = Machine Learning
SM = Smart Meters
Reroute = Αναδρομολόγηση

ΠΕΡΙΛΗΨΗ

Η πτυχιακή εργασία "Ένα Συνεργατικό Σχήμα Μηχανικής Μάθησης για Εξαγωγή Γνώσης στο Ευφυές Ηλεκτρικό Δίκτυο" επικεντρώνεται στην ανάπτυξη ενός καινοτόμου πλαισίου μηχανικής μάθησης για την βελτιστοποίηση της λειτουργίας του έξυπνου ηλεκτρικού δικτύου. Ενώ τα έξυπνα δίκτυα προσφέρουν πληθώρα πλεονεκτημάτων στη διαχείριση της ενέργειας, η ένταξη συνεργατικών μοντέλων μηχανικής μάθησης αναδεικνύεται ως κρίσιμο βήμα για την αποδοτική ανάλυση και πρόβλεψη των ενεργειακών συστημάτων. Η εργασία παρουσιάζει τη σχεδίαση, την υλοποίηση και τις δομές του συνεργατικού σχήματος, επικεντρώνεται στα αποτελέσματα και τις προοπτικές της εφαρμογής της στον τομέα της ενέργειας που αφορούν μια έξυπνη πόλη. Μέσα από επιτυχημένες προσομοιώσεις σε σενάρια του ευφυούς ηλεκτρικού δικτύου, η έρευνα αναδεικνύει τη παροχή του σχήματος στη βελτίωση της αποδοτικότητας (efficiency) και της αξιοπιστίας (reliability) του συστήματος. Συνολικά, η πτυχιακή εργασία δείχνει τη σημασία της εφαρμογής της μηχανικής μάθησης σε συνεργασία με την εξελισσόμενη τεχνολογία για την αναβάθμιση των έξυπνων ηλεκτρικών δικτύων και των έξυπνων μηχανήματων όπως energy meters, AC passthrough meters , neighbor electricity meters κτλπ. aka Smart Meters για την καλύτερη αποδοτικότητα και τον πιο αποτελεσματικό έλεγχο της κατανάλωσης του ρεύματος στα πλαίσια μιας έξυπνης πόλης.

ABSTRACT

The thesis "A Collaborative Machine Learning Scheme for Knowledge Extraction in the Smart Electric Grid" focuses on the development of an innovative machine learning framework for optimizing the operation of the smart electric grid. While smart grids offer numerous advantages in energy management, the integration of collaborative machine learning models is emerging as a critical step for the efficient analysis and forecasting of energy systems. The paper presents the design, implementation and structures of the cooperative scheme, focuses on the results and perspectives of its application in the field of energy concerning a smart city. Through successful simulations in smart grid scenarios, the research highlights the scheme's contribution to improving system efficiency and reliability. Overall, the thesis shows the importance of applying machine learning in collaboration with evolving technology to upgrade smart electrical networks and smart machines such as energy meters, AC passthrough meters, neighbor electricity meters, etc. aka Smart Meters for the best efficiency and the most effective control of electricity consumption in the context of a smart city.

ΠΕΡΙΕΧΟΜΕΝΑ

ΛΙΣΤΑ ΕΙΚΟΝΩΝ ΚΑΙ ΓΡΑΦΗΜΑΤΩΝ

ΚΕΦΑΛΑΙΟ 1 Εισαγωγή

Στην εποχή της ραγδαίας τεχνολογικής εξέλιξης, η ενέργεια, και ειδικότερα η ηλεκτρική ενέργεια, αποτελούν κρίσιμους πυλώνες για την οικονομική ανάπτυξη και την καθημερινή ζωή.

Το Έξυπνο Ηλεκτρικό Δίκτυο (ΕΗΔ) αναδύεται ως ένα καινούριο σύστημα, ενσωματώνοντας προηγμένες τεχνολογίες όπως η Τεχνητή Νοημοσύνη (ΤΝ) και η Μηχανική Μάθηση (ΜΜ). Στο πλαίσιο αυτό, η παρούσα πτυχιακή εργασία, εστιάζει σε ένα Συνεργατικό Σχήμα Μηχανικής Μάθησης για την Εξαγωγή Γνώσης στο ΕΗΔ. Η κεντρική πρόκληση που επιχειρείται να αντιμετωπιστεί είναι η ανάγκη για την πλήρη και αποτελεσματική εξαγωγή γνώσης από τα δεδομένα του ΕΗΔ, προκειμένου να επιτευχθεί η βέλτιστη λειτουργία, η ασφάλεια και αποδοτικότητα του. Το Συνεργατικό Σχήμα Μηχανικής Μάθησης σε σχέση με μια νέα προσέγγιση που αφορά το machine learning ενώνει τις δυνατότητες πολλαπλών μοντέλων μηχανικής μάθησης και εκμεταλλεύεται την συνεργατική τους δυναμική.

Η πτυχιακή θα μας παρουσιάσει τον σχεδιασμό και την υλοποίηση του Συνεργατικού αυτού Σχήματος, εξετάζοντας την αποτελεσματικότητά του. Βάσει των αποτελεσμάτων που θα προκύψουν, θα αναδειχθούν οι προοπτικές εφαρμογής αυτής της μεθόδου σε ένα ευρύ φάσμα περιβαλλοντικών συνθηκών, ενώ επιπλέον, θα διατυπωθούν προτάσεις για μελλοντικές επεκτάσεις και βελτιώσεις του Σχήματος, ανοίγοντας τον δρόμο για περαιτέρω έρευνα και ανάπτυξη στον τομέα των ευφυών ηλεκτρικών δικτύων.

1.1 Εισαγωγή στο Ευφυές Ηλεκτρικό Δίκτυο

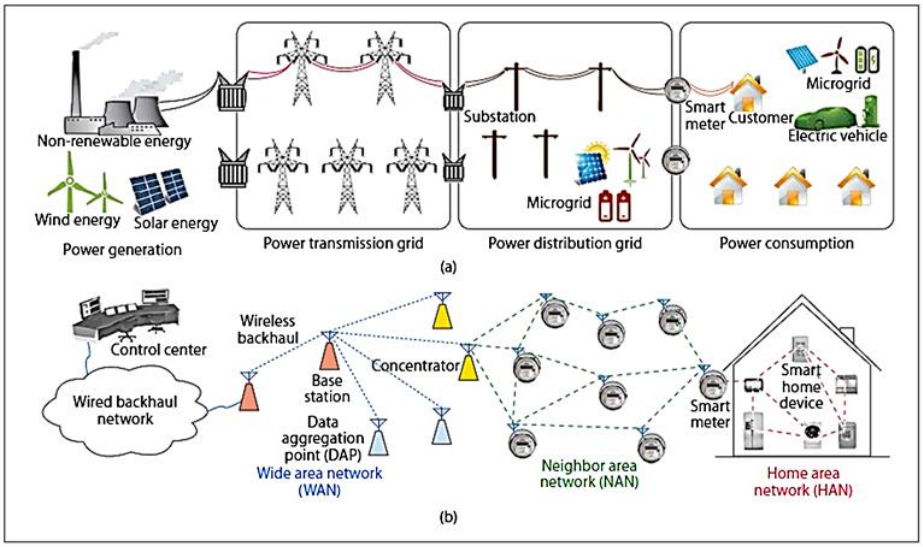
Για περισσότερο από έναν αιώνα, το ηλεκτρικό δίκτυο έχει λειτουργήσει με βάση μια κεντροποιημένη αρχιτεκτονική, όπου η παραγωγή ενέργειας πραγματοποιείται σε μεγάλους σταθμούς και η ροή της είναι μονόδρομη, κατευθυνόμενη από αυτούς τους σταθμούς προς τους τελικούς καταναλωτές. Αυτό το παραδοσιακό μοντέλο έχει εξυπηρετήσει αποτελεσματικά τις ανάγκες του παρελθόντος.

Ωστόσο, η σύγχρονη εποχή φέρνει στο προσκήνιο νέες προκλήσεις, οι οποίες καθιστούν επιτακτική την αναθεώρηση και την εξέλιξη του υφιστάμενου πλαισίου. Πιο συγκεκριμένα, η συνεχής αύξηση της ζήτησης ενέργειας, η επιτακτική ανάγκη για την επίτευξη περιβαλλοντικής βιωσιμότητας και η ολοένα και μεγαλύτερη ενσωμάτωση των Ανανεώσιμων Πηγών Ενέργειας (ΑΠΕ) στο ενεργειακό μείγμα, έχουν αναδείξει την ανεπάρκεια της παραδοσιακής προσέγγισης.

Υπό αυτές τις συνθήκες, γεννήθηκε η έννοια του Ευφυούς Ηλεκτρικού Δικτύου (Smart Grid).

1.1.α Σκοπός του Ευφυούς Ηλεκτρικού Δικτύου

Ένα ΕΗΔ, ενσωματώνει τεχνολογίες πληροφορικής και επικοινωνιών σε κάθε πτυχή της παραγωγής, διανομής και κατανάλωσης ηλεκτρικής ενέργειας. Στόχος του είναι η ελαχιστοποίηση των περιβαλλοντικών επιπτώσεων, η ενίσχυση των αγορών, η βελτίωση της αξιοπιστίας και των παρεχόμενων υπηρεσιών, καθώς και η μείωση του κόστους και η αύξηση της αποδοτικότητας. Αυτό επιτυγχάνεται με την εφαρμογή αισθητήρων, συστημάτων επικοινωνίας, υπολογιστικών δυνατοτήτων και λειτουργιών ελέγχου, με τελικό σκοπό την ενίσχυση της συνολικής λειτουργικότητας του συστήματος παροχής ηλεκτρικής ενέργειας. [1]

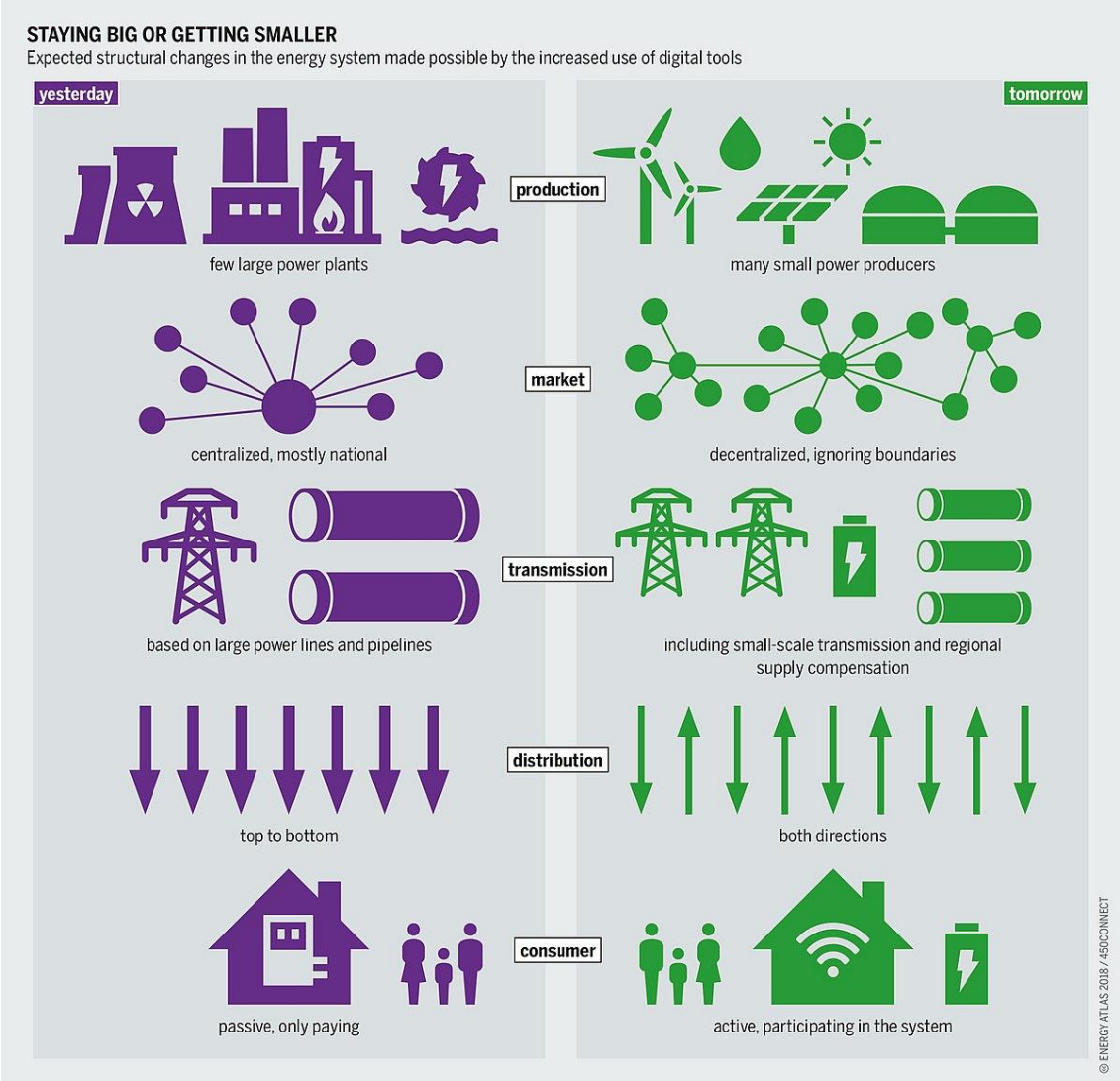


Εικόνα 1 Αναπαράσταση ενός ΕΗΔ. **[2]**

Σε σύγκριση με τους παραδοσιακούς τρόπους τα δίκτυα διαφέρουν στα παρακάτω:

ΠΑΡΑΔΟΣΙΑΚΟ ΔΙΚΤΥΟ	**ΕΥΦΥΕΣ ΗΛΕΚΤΙΚΟ ΔΙΚΤΥΟ**
Βασισμένο σε ελάχιστα εργοστάσια ηλεκτροπαραγωγής εκ των οποίων τα περισσότερα είναι επιβλαβές για το περιβάλλον	Βασισμένο σε πολλά και μικρά συστήματα παραγωγής ενέργειας καθαρά για το περιβάλλον
Συγκεντρωτική μορφή Παραγωγής	Αποκεντρωμένη μορφή Παραγωγής
Βασιζόμενο σε μεγάλα ηλεκτρικά καλώδια και αγωγούς*	*++ της μεταφοράς μικρής κλίμακας και της αντιστάθμισης περιφερειακής προμήθειας
Επικεντρωμένο στην μεταφορά από τα εργοστάσια ηλεκτροπαραγωγής προς τον καταναλωτή	Αμφίρροπη μεταφορά ενέργειας από τις αποθήκες ενέργειας και τον καταναλωτή
Ο καταναλωτής μπορεί μόνο να καταναλώσει ενέργεια με αποτέλεσμα να πληρώνει ότι χρησιμοποιεί	Ο καταναλωτής μπορεί να καταναλώνει ενέργεια αλλά και να παράγει πχ με φωτοβολταϊκά και να πληρώνει την διαφορά καθιστώντας τον ενεργό στο έξυπνο δίκτυο

Η οπτικοποίηση του πίνακα βρίσκεται στην παρακάτω εικόνα:



Εικόνα 2 Σύγκριση συμβατικού Ηλεκτρικού Δικτύου (αριστερά) με το ΕΗΔ (δεξιά). **[3]**

1.1.β Πλεονεκτήματα του Ευφυούς Ηλεκτρικού Δικτύου

Ως αποτέλεσμα του ΕΗΔ έχουμε τα εξής πλεονεκτήματα σε σχέση με τους παραδοσιακούς τρόπους. Κάποια από αυτά είναι:

- **Αξιοπιστία και Ανθεκτικότητα:** Με τη χρήση προηγμένων αισθητήρων και συστημάτων επικοινωνίας, το δίκτυο αποκτά την ικανότητα αυτόματης ανίχνευσης και αποκατάστασης σφαλμάτων (self-healing) [4] [5]. Αυτό σημαίνει ότι μπορεί να εντοπίσει και να απομονώσει γρήγορα βλάβες, επαναφέροντας την παροχή ενέργειας σε σύντομο χρονικό διάστημα. Επιπλέον, η συνεχής παρακολούθηση της κατάστασης του εξοπλισμού συμβάλλει στην προληπτική συντήρηση, αποτρέποντας βλάβες πριν αυτές συμβούν [6].
- **Ευελιξία και Προσαρμοστικότητα:** Το ευφυές δίκτυο αναλαμβάνει τον διαμοιρασμό και την σύνδεση τόσο των κεντρικών σταθμών παραγωγής ενέργειας όσο και των μικρών πηγών ενέργειας που απευθύνονται στους καταναλωτές και μη αλλά και των αποθηκών ενέργειας και των υπολοίπων πηγών παραγωγής. Μέσα σε αυτές τις πηγές παραγωγής είναι και οι ΑΠΕ, οι οποίες εξ ορισμού χαρακτηρίζονται από μια μεταβλητότητα. Με τη βοήθεια έξυπνων συστημάτων πρόβλεψης και ελέγχου, το ΕΗΔ, διαχειρίζεται αυτές τις διακυμάνσεις,

επιτρέποντας την αποτελεσματική αξιοποίηση μεγαλύτερων ποσοτήτων ανανεώσιμης ενέργειας [5].

- **Αποδοτικότητα:** Μέσω της παρακολούθησης σε πραγματικό χρόνο της κατανάλωσης και των ροών ενέργειας, οι διαχειριστές δικτύου μπορούν να εντοπίσουν και να διορθώσουν σημεία απωλειών, βελτιστοποιώντας έτσι τη διανομή και μειώνοντας τις ενεργειακές απώλειες. Επιπλέον, προγραμμάτα διαχείρισης αιχμής ζήτησης (peak demand management), μειώνουν την ανάγκη για λειτουργία ακριβών μονάδων παραγωγής κατά τις ώρες υψηλής ζήτησης, οδηγώντας σε πιο ορθολογική χρήση των πόρων [5].
- **Οφέλη προς τον καταναλωτή:** Μέσω των έξυπνων μετρητών, αποκτούν λεπτομερή πρόσβαση στα δεδομένα κατανάλωσής τους, ενδυναμώνοντάς τους να λαμβάνουν τεκμηριωμένες αποφάσεις για τη χρήση της ενέργειας [5]. Επιπλέον, η αυξημένη αξιοπιστία του δικτύου μεταφράζεται σε λιγότερες διακοπές και συνολικά καλύτερη ποιότητα υπηρεσιών.
- **Φιλικότητα προς το περιβάλλον:** Η αυξημένη ενσωμάτωση και αποτελεσματική διαχείριση των ΑΠΕ μειώνει την εξάρτηση από τα ορυκτά καύσιμα, οδηγώντας σε σημαντική μείωση των εκπομπών αερίων του θερμοκηπίου με αποτέλεσμα τα ΕΗΔ, να συμβάλλουν στην περιβαλλοντική βιωσιμότητα [4].

1.2 Η Σημασία της Εξαγωγής Γνώσης στο ΕΗΔ

Η διασύνδεση των συστατικών του Ευφυούς Ηλεκτρικού Δικτύου οδηγούν στην παραγωγή τεράστιων όγκων δεδομένων σε πραγματικό χρόνο. Αυτά τα δεδομένα προέρχονται από ποικίλες πηγές, όπως έξυπνοι μετρητές, αισθητήρες στο δίκτυο μεταφοράς και διανομής, όπως θα δούμε στο παρακάτω κεφάλαιο. Η αποτελεσματική αξιοποίηση αυτών των «μεγάλων δεδομένων» (Big Data) είναι καθοριστική για την πλήρη αξιοποίηση των δυνατοτήτων του ΕΗΔ.

Η εξαγωγή γνώσης από αυτά τα δεδομένα, ένας από τους βασικούς πυλώνες της παρούσας πτυχιακής, αναφέρεται στη διαδικασία ανακάλυψης τάσεων, συσχετίσεων και πληροφοριών που δεν είναι άμεσα εμφανείς. Αυτή η γνώση είναι απαραίτητη για τη λήψη τεκμηριωμένων αποφάσεων καθώς και τη βελτιστοποίηση της λειτουργίας του δικτύου. Πιο συγκεκριμένα:

- **Βελτιωμένη Πρόβλεψη Φορτίου:** Η ακριβής πρόβλεψη της ζήτησης ενέργειας σε διάφορες χρονικές κλίμακες επιτρέπει στο αποτελεσματικότερο προγραμματισμό της παραγωγής ενέργειας, και αποτέλεσμα την μείωση του κόστους. [7]
- **Εντοπισμός και Διάγνωση Σφαλμάτων:** Η ανάλυση των δεδομένων μπορεί να αποκαλύψει ανωμαλίες στη λειτουργία του δικτύου, επιτρέποντας την ταχεία ανίχνευση βλαβών, την πρόληψη προβλημάτων και τη μείωση του χρόνου αποκατάστασης. [7]
- **Βελτιστοποίηση Λειτουργίας και Συντήρησης:** Η γνώση για την κατάσταση του εξοπλισμού και την απόδοση του δικτύου επιτρέπει την προληπτική συντήρηση, τη βελτιστοποίηση της ροής ενέργειας και τη μείωση των λειτουργικών απωλειών. [7]
- **Ενεργή Διαχείριση Ζήτησης:** Η κατανόηση των τάσεων κατανάλωσης των χρηστών επιτρέπει την ανάπτυξη πιο αποτελεσματικών προγραμμάτων διαχείρισης ζήτησης. [7]
- **Ασφάλεια Δικτύου:** Η ανάλυση των δεδομένων ροής επικοινωνίας μπορεί να βοηθήσει στην ανίχνευση κυβερνοεπιθέσεων και κακόβουλων δραστηριοτήτων, θωρακίζοντας το δίκτυο. [8]

1.3 Ο Ρόλος της Μηχανικής Μάθησης

Η πολυπλοκότητα, ο όγκος και η ταχύτητα παραγωγής των δεδομένων στο Ευφυές Ηλεκτρικό Δίκτυο καθιστούν τις παραδοσιακές μεθόδους ανάλυσης ανεπαρκείς για την αποτελεσματική εξαγωγή γνώσης. Σε αυτό το σημείο, ένα χρήσιμο εργαλείο είναι η Μηχανική Μάθηση (Machine Learning). Η Μηχανική Μάθηση, είναι ένα πεδίο της Τεχνητής Νοημοσύνης (ΤΝ), το οποίο εστιάζει στην ανάπτυξη αλγορίθμων που επιτρέπουν σε ένα σύστημα να μαθαίνει αυτόματα από δεδομένα, να αναγνωρίζει σύνθετα πρότυπα και να λαμβάνει αποφάσεις ή να πραγματοποιεί προβλέψεις, χωρίς να χρειάζεται να προγραμματιστεί για κάθε ξεχωριστό σενάριο [9].

Οι αλγόριθμοι Μηχανικής Μάθησης είναι ιδιαίτερα βοηθητικοί για την επεξεργασία των δεδομένων του Smart Grid - δηλαδή, δεδομένων που είναι υψηλής διάστασης, χρονικά εξαρτώμενα και συχνά ατελή ή θορυβώδη. Μέσω της εφαρμογής τους, μπορούν να αποκαλύψουν τάσεις και εξαρτήσεις που θα ήταν αδύνατο να εντοπιστούν με χειροκίνητες αναλύσεις ή απλές στατιστικές μεθόδους.

1.4 Σκοπός και Δομή της Εργασίας

Ο σκοπός της παρούσας πτυχιακής εργασίας είναι να εξερευνήσει τον τρόπο με τον οποίο η Μηχανική Μάθηση μπορεί να ενσωματωθεί σε ένα Συνεργατικό Σχήμα στον τομέα του ΈΗΔ.

Η δομή της εργασίας υποδεικνύει τη συστηματική ανάλυση του θέματος, αρχίζοντας από το θεωρητικό υπόβαθρο και φτάνοντας στην υλοποίηση του συνεργατικού σχήματος και την αξιολόγηση των αποτελεσμάτων.

1.4.α Σκοπός της Εργασίας

Σκοπός της εργασίας είναι η διερεύνηση και η ανάπτυξη ενός Συνεργατικού Σχήματος Μηχανικής Μάθησης, προσαρμοσμένου στις ανάγκες και τις ιδιαιτερότητες ενός Ευφυούς Ηλεκτρικού Δικτύου. Επιδιώκεται η αξιοποίηση των δυνατοτήτων της κατανεμημένης επεξεργασίας δεδομένων, με στόχο την αποτελεσματική εξαγωγή γνώσης από τους τεράστιους όγκους πληροφοριών που παράγονται από το Δίκτυο αντιμετωπίζοντας παράλληλα τις προκλήσεις που θα έρθουν προς το μέρος μας.

Πιο συγκεκριμένα στοχεύουμε:

- Να εμβαθύνουμε στις έννοιες του ΕΗΔ, της Μηχανικής Μάθησης καθώς και του Συνεργατικού Σχήματος.
- Να αναπτύξουμε την αρχιτεκτονική ενός Συνεργατικού Σχήματος που θα επιτρέπει την εκπαίδευση μοντέλων σε κατανεμημένο περιβάλλον και αξιοποιώντας την υπολογιστική ισχύ των τοπικών κόμβων.
- Να δημιουργήσουμε μια προσομοίωση του παραπάνω σχήματος και να σχεδιάσουμε ένα πίνακα ελέγχου μέσα από τον οποίο θα μπορούμε να οπτικοποιήσουμε τα αποτελέσματά μας.
- Να εξετάσουμε τον ρόλο και τις δυνατότητες της Μηχανικής Μάθησης για την ανάλυση δεδομένων καθώς και τις εφαρμογές της στην πρόβλεψη φορτίου μιας πόλης με πλήθος ταχυδρομικών κωδικών, καθώς και στην βελτιστοποίηση των μοντέλων μας.
- Να αξιολογήσουμε την αποτελεσματικότητα και την αποδοτικότητα του προτεινόμενου σχήματος, τόσο ως προς την ποιότητα της εξαγόμενης γνώσης όσο και ως προς την αντιμετώπιση των προκλήσεων.

1.4.β Δομή της Εργασίας

Η παρούσα εργασία υποδιαιρείται σε επιμέρους κεφάλαια με σκοπό τη συστηματική ανάλυση και παρουσίαση των βασικών στοιχείων. Παρακάτω παρέχεται μια επισκόπηση της δομής της εργασίας:

Κεφάλαιο 2: Θεωρητικό Υπόβαθρο

- Παρουσιάζει αναλυτικά τις βασικές αρχές λειτουργίας του ΕΗΔ, τα συστατικά του και τις τεχνολογίες που το υποστηρίζουν.
- Εμβαθύνει στις θεμελιώδεις έννοιες της Μηχανικής Μάθησης και των εφαρμογών τους σε ενεργειακά συστήματα.
- Παρουσιάζει συναφείς εργασίες και προσεγγίσεις από τη διεθνή βιβλιογραφία, αναδεικνύοντας τα κενά που επιχειρεί να καλύψει η παρούσα έρευνα.

Κεφάλαιο 3: Εφαρμογή Συνεργατικού Σχήματος Μηχανικής Μάθησης

- Αναλύει λεπτομερώς τη μεθοδολογία που ακολουθήθηκε για την ανάπτυξη του Συνεργατικού Σχήματος Μηχανικής Μάθησης.
- Περιγράφει την διαδικασία δημιουργίας δεδομένων καθώς και του συστήματος ασφαλείας.
- Παρουσιάζει την αρχιτεκτονική του προτεινόμενου σχήματος, εξηγώντας τη λειτουργία των επιμέρους συστατικών του (Edge Nodes, Master Nodes, Dashboard κλπ.).
- Περιγράφει τους αλγορίθμους Μηχανικής Μάθησης που χρησιμοποιήθηκαν καθώς και τις τροποποιήσεις που χρειάστηκε να πραγματοποιηθούν.

Κεφάλαιο 4: Αξιολόγηση Αποτελεσμάτων

- Παρουσιάζει τα προβλήματα που αντιμετωπίστηκαν κατά την διάρκεια της εκπόνησης των πειραμάτων.
- Παραθέτει τις μετρικές αξιολόγησης της απόδοσης του σχήματος.
- Παρουσιάζει και ερμηνεύει τα αποτελέσματα.

Κεφάλαιο 5: Συμπεράσματα και Μελλοντικές Επεκτάσεις

- Συνοψίζει τα κύρια συμπεράσματα που προέκυψαν από την έρευνα και τα πειραματικά αποτελέσματα.
- Αξιολογεί την επίτευξη των στόχων που τέθηκαν στην εισαγωγή.
- Προτείνει κατευθύνσεις για μελλοντική έρευνα και επέκταση της παρούσας εργασίας.

ΚΕΦΑΛΑΙΟ 2 Θεωρητικό Υπόβαθρο

Το παρόν κεφάλαιο έχει ως στόχο να παρέχει ένα ολοκληρωμένο θεωρητικό υπόβαθρο, απαραίτητο για την κατανόηση των μεθόδων και συστημάτων που θα χρησιμοποιηθούν κατά την διάρκεια της παρούσας εργασίας. Θα αναλυθούν οι βασικές έννοιες που σχετίζονται με το Ευφυές Ηλεκτρικό Δίκτυο, τη Μηχανική Μάθηση και τις κατανεμημένες προσεγγίσεις μάθησης. Επιπλέον, θα γίνει μια ανασκόπηση της υφιστάμενης βιβλιογραφίας, αναδεικνύοντας τις τρέχουσες προσεγγίσεις και δυνητικά, τα ερευνητικά κενά.

2.1 Το Ευφυές Ηλεκτρικό Δίκτυο

Όπως αναφέραμε και στο 1ο Κεφάλαιο, το ΕΗΔ αντιπροσωπεύει τη μετεξέλιξη του παραδοσιακού ηλεκτρικού δικτύου, ενσωματώνοντας σύγχρονες τεχνολογίες πληροφορικής και επικοινωνιών, με σκοπό τη βελτίωση της αποδοτικότητας, της αξιοπιστίας, της ασφάλειας και της βιωσιμότητας της παροχής ενέργειας. Σε αντίθεση με το παλιό μοντέλο που χαρακτηριζόταν από μονοκατευθυντική ροή ενέργειας (από τον παραγωγό στον καταναλωτή) και περιορισμένη ορατότητα, το ΕΗΔ λειτουργεί ως ένα αμφίδρομο, διαδραστικό σύστημα.

2.1.α Αρχιτεκτονική του ΕΗΔ

Η αρχιτεκτονική του ΕΗΔ μπορεί να κατηγοριοποιηθεί σε τρία βασικά επίπεδα, τα οποία λειτουργούν συνεργατικά για τη βελτιστοποίηση της συνολικής απόδοσης του συστήματος. Συνοπτικά, τα τρία επίπεδα είναι:

- **Επίπεδο Συστημάτων Ισχύος:** Το επίπεδο αυτό φέρει την ευθύνη για την παραγωγή και διανομή της ηλεκτρικής ενέργειας στους τελικούς καταναλωτές, διατηρώντας λειτουργικά χαρακτηριστικά ανάλογα με αυτά ενός συμβατικού δικτύου ηλεκτροδότησης [10].
- **Επίπεδο Επικοινωνιών:** Το εν λόγω επίπεδο εξασφαλίζει τη διασύνδεση μεταξύ όλων των επιμέρους συνιστωσών του συστήματος. Η λειτουργία του περιλαμβάνει τη συλλογή δεδομένων από αισθητήρες και διεπαφές τελικών χρηστών, την οποία ακολουθεί η διαβίβαση των δεδομένων αυτών σε κέντρα επεξεργασίας και αντιστρόφως [10].
- **Επίπεδο Εφαρμογών:** Σε αυτό το επίπεδο, τα συλλεγόμενα δεδομένα υποβάλλονται σε επεξεργασία με σκοπό την έκδοση μηνυμάτων παρακολούθησης και ελέγχου. Επιπρόσθετα, τα δεδομένα αξιοποιούνται για την υποστήριξη ποικίλων εφαρμογών, όπως η διαχείριση της ζήτησης, η αυτόματη ανάγνωση μετρητών και η ανίχνευση περιπτώσεων απάτης ή κακής χρήσης [10].

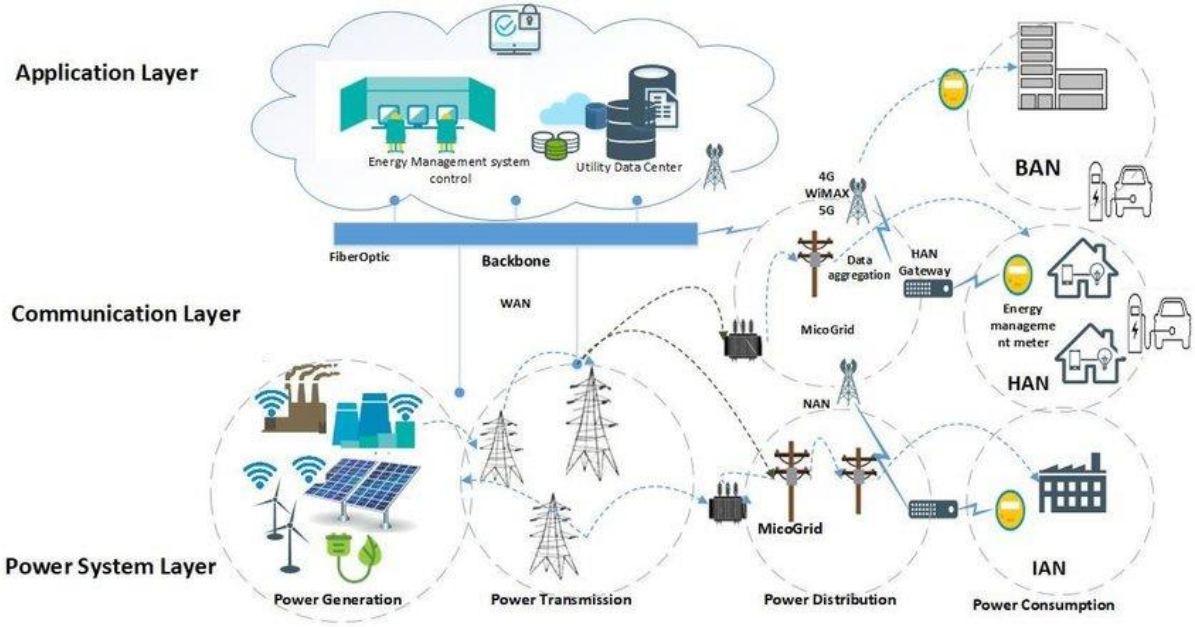
Το Επίπεδο Συστημάτων Ισχύος είναι το ίδιο όπως και σε ένα παραδοσιακό Ηλεκτρικό Δίκτυο και αποτελείται από τα παρακάτω:

- **Παραγωγή (Generation):** Περιλαμβάνει τόσο τις συμβατικές μονάδες παραγωγής όσο και την ολοένα αυξανόμενη ενσωμάτωση των Ανανεώσιμων Πηγών Ενέργειας (ΑΠΕ), όπως φωτοβολταϊκά και αιολικά πάρκα, γεωθερμία και άλλα. Το ΕΗΔ επιτρέπει την πιο ευέλικτη διαχείριση της διαλείπουσας φύσης των ΑΠΕ και την ενσωμάτωση μικρότερων, κατανεμημένων μονάδων παραγωγής (Distributed Generation), οι οποίες μπορεί να βρίσκονται ακόμη και στις εγκαταστάσεις των καταναλωτών (π.χ., φωτοβολταϊκά στέγης) [11].
- **Μεταφορά (Transmission):** Αφορά το δίκτυο υψηλής τάσης που μεταφέρει ηλεκτρική ενέργεια από τους μεγάλους σταθμούς παραγωγής σε κέντρα ζήτησης. Στο ΕΗΔ, αυτό το επίπεδο περιλαμβάνει προηγμένα συστήματα παρακολούθησης και ελέγχου (όπως Wide Area Measurement Systems - WAMS) [12] για την

ανίχνευση ανωμαλιών, τη βελτιστοποίηση της ροής ισχύος και τη διαχείριση της σταθερότητας του δικτύου [11].

- **Διανομή (Distribution):** Είναι το τμήμα του δικτύου που μεταφέρει την ενέργεια από τους υποσταθμούς υψηλής/μέσης τάσης στους τελικούς χρήστες. Εδώ, το ΕΗΔ εισάγει σημαντικές καινοτομίες, όπως αυτοματοποίηση δικτύου διανομής (Distribution Automation - DA) [8], συστήματα αυτο-ίασης (self-healing) και τη δυνατότητα διαχείρισης αμφίδρομων ροών ενέργειας λόγω της κατανεμημένης παραγωγής [11].
- **Κατανάλωση (Consumption):** Αναφέρεται στους οικιακούς, εμπορικούς και βιομηχανικούς καταναλωτές. Το ΕΗΔ ενδυναμώνει τους καταναλωτές μέσω έξυπνων μετρητών (smart meters) και συσκευών που επιτρέπουν την παρακολούθηση της κατανάλωσης σε πραγματικό χρόνο και τη συμμετοχή σε προγράμματα διαχείρισης ζήτησης (Demand-Side Managment - DSM) [13]. Αυτό μπορεί να οδηγήσει σε πιο αποδοτική χρήση ενέργειας και μείωση του κόστους.
- **Αγορά Ενέργειας (Market):** Αν και δεν είναι ένα φυσικό επίπεδο του δικτύου, η αγορά ενέργειας επηρεάζεται και επηρεάζει όλα τα παραπάνω επίπεδα. Το ΕΗΔ υποστηρίζει πιο δυναμικές και διαφανείς αγορές, επιτρέποντας την ενσωμάτωση μικρών παραγωγών και την εφαρμογή ευέλικτων τιμολογιακών πολιτικών [11].

Στην παρακάτω εικόνα μπορούμε να δούμε πιο αναλυτικά την αρχιτεκτονική των ΕΗΔ:



Εικόνα 3 Η Αρχιτεκτονική ενός ΕΗΔ **[10]**

2.1.β Βασικές Τεχνολογίες Υποστήριξης

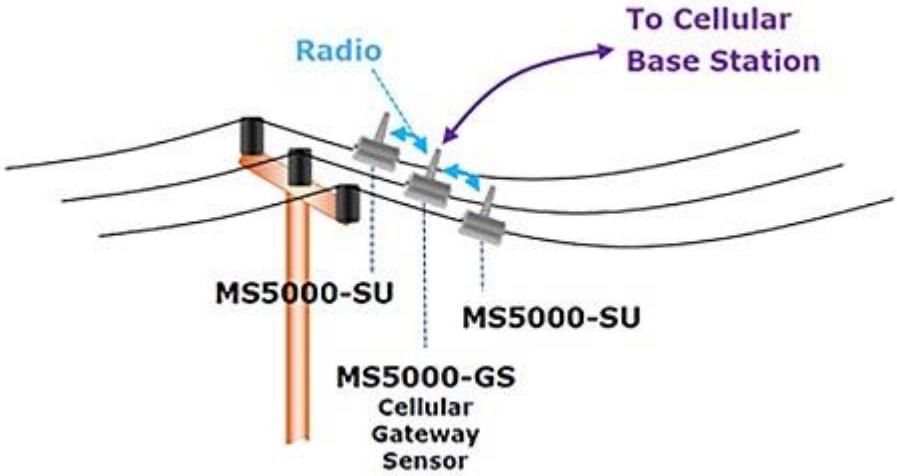
Η λειτουργία του Smart Grid βασίζεται σε μια σειρά αλληλοσυνδεόμενων τεχνολογιών [14]:

- **Προηγμένες Υποδομές Μέτρησης (Advanced Metering Infrastructure - AMI):** Αποτελούν τον πυρήνα της αμφίδρομης επικοινωνίας. Συγκροτούνται από ένα σύνολο φυσικών και ψηφιακών στοχείων στα οποία περιλαμβάνονται αισθητήρες, συστήματα παρακολούθησης, έξυπνοι μετρητές, λογισμικό καθώς και συστήματα διαχείρισης δεδομένων. Αυτά τα συστήματα φέρουν την ευθύνη για τη συλλογή, ανάλυση και αποθήκευση των δεδομένων μετρήσεων που αποστέλλονται από τους αισθητήρες, τα συστήματα παρακολούθησης και τους

έξυπνους μετρητές στους τελικούς χρήστες, προς τις εταιρείες παροχής ενέργειας, με σκοπό την τιμολόγηση, τη διαχείριση του δικτύου και την πρόβλεψη [15]. Μέσω αυτής της πληροφόρησης, οι καταναλωτές αποκτούν τη δυνατότητα να παρακολουθούν την κατανάλωση ενέργειας, γεγονός που τους επιτρέπει να τη διαχειρίζονται αποδοτικότερα, συμβάλλοντας έτσι στη μείωση των εκπομπών διοξειδίου του άνθρακα. Επιπλέον, μέσω της διαχείρισης του φορτίου αιχμής με τη συμμετοχή των καταναλωτών, η εταιρεία παροχής ενέργειας δύναται να προσφέρει ηλεκτρική ενέργεια σε χαμηλότερες και σταθερότερες τιμές για το σύνολο των χρηστών [8].

Εικόνα 4 Ένας έξυπνος μετρητής **[16]**

- **Τεχνολογίες Επικοινωνίας (Communication Technologies):** Για την ανταλλαγή τόσο μεγάλου όγκου δεδομένων απαιτούνται αξιόπιστα και ασφαλή δίκτυα. Χρησιμοποιούνται ευρυζωνικές τεχνολογίες (π.χ., οπτικές ίνες, Wi-Fi), τεχνολογίες κινητής τηλεφωνίας (π.χ., 4G/5G) [15] και Power Line Communication (PLC) [8], που χρησιμοποιεί τις υπάρχουσες γραμμές ηλεκτρικής ενέργειας για τη μεταφορά δεδομένων.
- **Αισθητήρες και Ενεργοποιητές (Sensors and Actuators):** Οι αισθητήρες είναι εγκατεστημένοι σε κρίσιμα σημεία όπως μετασχηματιστές, γραμμές μεταφοράς και υποσταθμούς, συλλέγουν δεδομένα σε πραγματικό χρόνο για την τάση, το ρεύμα, τη θερμοκρασία και άλλες κρίσιμες παραμέτρους. Τα δεδομένα αυτά είναι ζωτικής σημασίας για την παρακολούθηση της κατάστασης του δικτύου και την αποτελεσματική διαχείριση της ισχύος. [15] Οι ενεργοποιητές (όπως οι έξυπνοι διακόπτες) λειτουργούν συμπληρωματικά, εκτελώντας εντολές που βασίζονται στις πληροφορίες των αισθητήρων, με στόχο την ανακατεύθυνση της ροής ενέργειας ή την άμεση απομόνωση σφαλμάτων, εξασφαλίζοντας έτσι την αξιοπιστία και τη δυναμική απόκριση του δικτύου [17].



Εικόνα 5 Παράδειγμα Αισθητήρων ΕΗΔ **[18]**

- **Συστήματα Ελέγχου και Διαχείρισης (Control and Management Systems):** Τα συστήματα ελέγχου και διαχείρισης αποτελούν τον πυρήνα των έξυπνων δικτύων, εξασφαλίζοντας την αποτελεσματική και αξιόπιστη λειτουργία τους. Ενώ στα συμβατικά δίκτυα τα SCADA (Supervisory Control and Data Acquisition) συστήματα περιορίζονταν κυρίως στα δίκτυα μεταφοράς, στα έξυπνα δίκτυα οι δυνατότητές τους έχουν επεκταθεί δραματικά. Αυτή η εξέλιξη οφείλεται στην ταχεία αμφίδρομη επικοινωνία και στην εκτεταμένη χρήση αισθητήρων σε όλο το δίκτυο.
  Πλέον, τα SCADA, σε συνδυασμό με τα Distribution Management Systems (DMS), επεξεργάζονται δεδομένα σε πραγματικό χρόνο για την επίβλεψη, τον έλεγχο και τη βελτιστοποίηση ολόκληρης της λειτουργίας του δικτύου [8]. Η συνεχής παρακολούθηση, μέσω έξυπνων αισθητήρων και των Προηγμένων Υποδομών Μετρήσεων (AMI), είναι κρίσιμη για τη διατήρηση της ποιότητας ισχύος [15].
  Αυτά τα προηγμένα συστήματα επιτρέπουν την ταχεία αναγνώριση, διαχείριση και αποκατάσταση βλαβών, καθώς μπορούν να παρέχουν ειδοποιήσεις για διακοπές ακόμα και πριν γίνουν αντιληπτές από τον καταναλωτή. Το αποτέλεσμα είναι καλύτερη διαχείριση, ακριβέστερη βελτιστοποίηση πόρων, ταχύτερη αναγνώριση βλαβών και βελτιωμένη αξιοπιστία. Ωστόσο, η εξάρτηση από δίκτυα επικοινωνιών και τεχνολογίες διαδικτύου εγείρει σημαντικές ανησυχίες σχετικά με τις απειλές κυβερνοασφάλειας [8].
- **Τεχνολογίες Πληροφορικής (Information Technology - IT):** Στην σύγχρονη πραγματικότητα, αποτελούν αναπόσπαστο προαπαιτούμενο για τη λειτουργία και εξέλιξη του Ευφυούς Ηλεκτρικού Δικτύου (ΕΗΔ). Η ενσωμάτωσή τους είναι απαραίτητη για τη διαχείριση του τεράστιου όγκου δεδομένων που παράγεται, καθώς και για την υποστήριξη προηγμένων λειτουργιών.
  - **Συστήματα Διαχείρισης Δεδομένων:** Αυτά περιλαμβάνουν την εφαρμογή προηγμένων μεθόδων ανάλυσης μεγάλων δεδομένων (Big Data analytics). Μέσω αυτών, το δίκτυο είναι σε θέση να συλλέγει, να αποθηκεύει και να επεξεργάζεται τεράστιες ποσότητες πληροφοριών σε πραγματικό χρόνο, οι οποίες προέρχονται από διάφορες πηγές, όπως έξυπνους μετρητές, αισθητήρες και συστήματα ελέγχου [7].
  - **Πλατφόρμες Cloud Computing:** Η χρήση υπολογιστικού νέφους (cloud computing) προσφέρει την απαραίτητη υπολογιστική ισχύ και ευελιξία για την φιλοξενία και εκτέλεση πολύπλοκων εφαρμογών και αλγορίθμων [10]. Επιτρέπει την αποδοτική διαχείριση πόρων και την

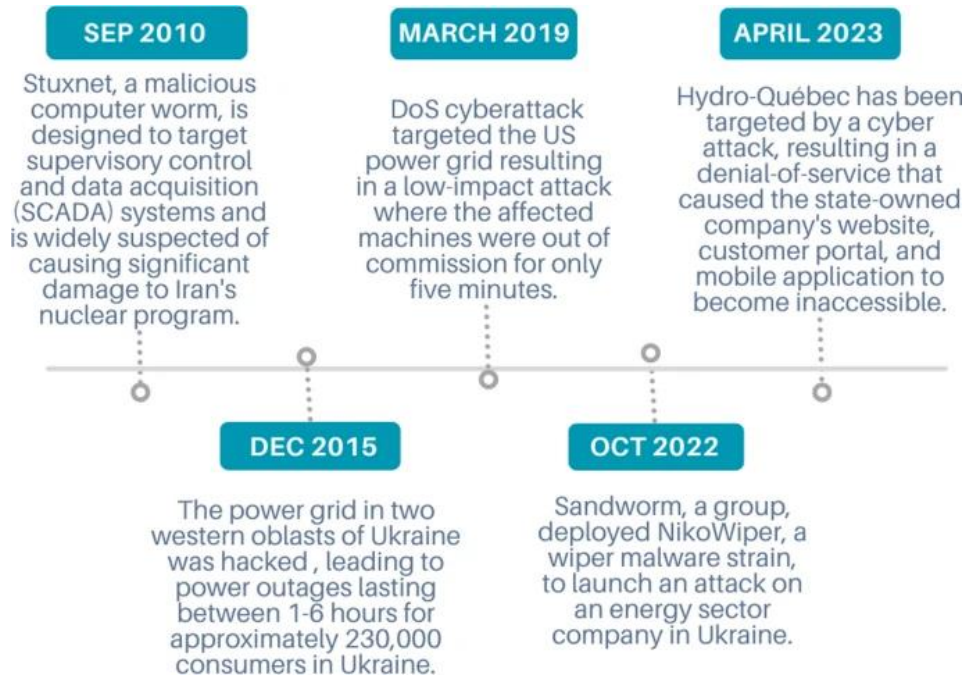
απρόσκοπτη πρόσβαση σε δεδομένα και υπηρεσίες, ανεξάρτητα από τη φυσική τους τοποθεσία.

- **Τεχνολογίες Τεχνητής Νοημοσύνης και Μηχανικής Μάθησης:** Η Τεχνητή Νοημοσύνη και η Μηχανική Μάθηση παρέχουν τα αναγκαία εργαλεία για την εξαγωγή γνώσης από τα συλλεγόμενα δεδομένα, την αναγνώριση προτύπων, την πρόβλεψη συμπεριφορών (π.χ., πρόβλεψη ζήτησης ή παραγωγής από ΑΠΕ), καθώς και τη λήψη έξυπνων αποφάσεων σε πραγματικό χρόνο.
  Συγκεκριμένα, η ενσωμάτωση των αλγορίθμων Μηχανικής Μάθησης στο ΕΗΔ διευκολύνει την απρόσκοπτη ενσωμάτωση ανανεώσιμων πηγών ενέργειας και συμβάλλει στην αύξηση της λειτουργικής αποδοτικότητας, καθιστώντας το δίκτυο πιο εύρωστο, προσαρμόσιμο και οικολογικά φιλικό. Αυτές οι τεχνολογίες επιτρέπουν την ανάλυση των λειτουργιών του έξυπνου δικτύου σε διάφορα επίπεδα, όπως η πρόβλεψη της ενεργειακής ζήτησης, ο εντοπισμός ανωμαλιών και η μείωση των απειλών στον κυβερνοχώρο [19].

2.1.γ Προκλήσεις και Ευκαιρίες του Smart Grid

Η μετάβαση σε ένα Smart Grid, αν και προσφέρει τεράστιες ευκαιρίες, αντιμετωπίζει και σημαντικές προκλήσεις:

- **Ασφάλεια και Κυβερνοασφάλεια (Security and Cybersecurity):** Η διασύνδεση και η εξάρτηση από τεχνολογίες πληροφορικής και επικοινωνιών καθιστούν το ΕΗΔ ευάλωτο σε κυβερνοεπιθέσεις. Μια επιτυχημένη επίθεση θα μπορούσε να οδηγήσει σε εκτεταμένες διακοπές ρεύματος ή χειραγώγηση των δεδομένων. Η διασφάλιση της ανθεκτικότητας έναντι τέτοιων απειλών είναι μια διαρκής πρόκληση. Στο παρακάτω διάγραμμα μπορούμε να δούμε μερικές επιθέσεις σε δίκτυα ηλεκτρισμού καθώς και τα αποτελέσματα αυτών, σε μια εποχή που τα ΕΗΔ δεν ήταν τόσο διαδεδομένα.



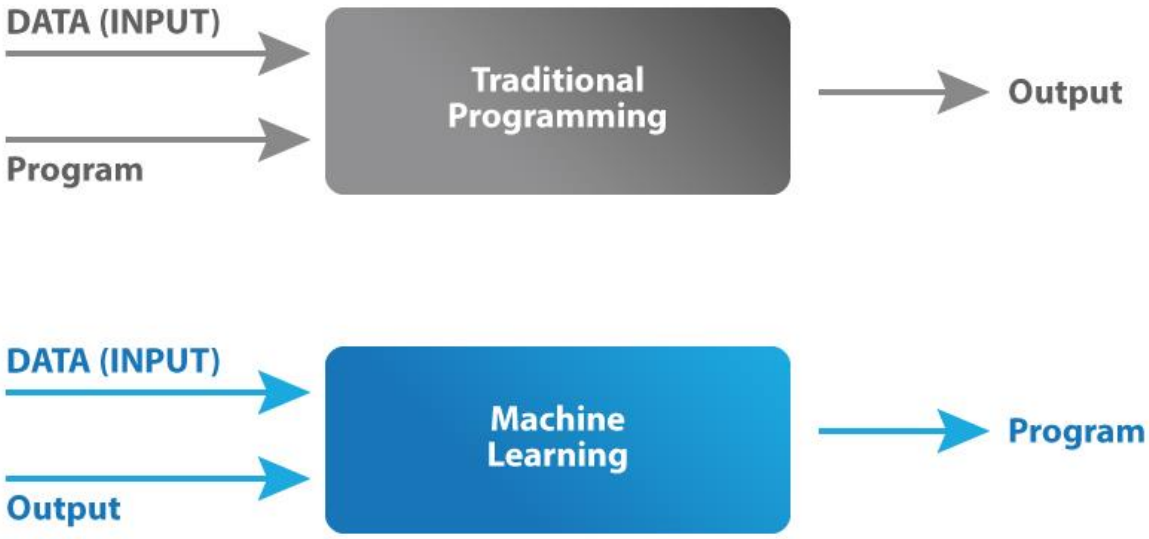
Εικόνα 6 Ιστορικό επιθέσεων στον κυβερνοχώρο σε βιομηχανικές και ενεργειακές εγκαταστάσεις **[20]**

- **Ιδιωτικότητα Δεδομένων (Data Privacy):** Οι έξυπνοι μετρητές συλλέγουν λεπτομερή δεδομένα κατανάλωσης, τα οποία μπορούν να αποκαλύψουν ευαίσθητες πληροφορίες για τις συνήθειες των καταναλωτών (π.χ., πότε είναι σπίτι, ποιες συσκευές χρησιμοποιούν). Η προστασία της ιδιωτικότητας αυτών των δεδομένων αποτελεί σοβαρό ζήτημα και απαιτεί ισχυρά ρυθμιστικά πλαίσια και τεχνικές λύσεις [21].
- **Διαλειτουργικότητα (Interoperability):** Το ΕΗΔ αποτελείται από ένα πλήθος διαφορετικών συσκευών και συστημάτων από διαφορετικούς κατασκευαστές. Η διασφάλιση της απρόσκοπτης επικοινωνίας και συνεργασίας μεταξύ τους είναι μια τεχνική πρόκληση που απαιτεί την ανάπτυξη κοινών προτύπων [22].
- **Κόστος Υλοποίησης:** Η αναβάθμιση της υπάρχουσας υποδομής σε Smart Grid απαιτεί σημαντικές επενδύσεις σε τεχνολογία, εξοπλισμό και εκπαίδευση προσωπικού [23].
- **Διαχείριση Δεδομένων (Big Data Management):** Ο τεράστιος όγκος, η ταχύτητα και η ποικιλία των δεδομένων που παράγονται θέτουν προκλήσεις στην αποθήκευση, επεξεργασία, ανάλυση και εξαγωγή γνώσης σε πραγματικό χρόνο [7] [24].

Παρ' όλες τις προκλήσεις, οι ευκαιρίες που προσφέρει το Smart Grid είναι τεράστιες, όπως αναφέραμε και στο 1ο κεφάλαιο, όπως βελτιωμένη ενεργειακή απόδοση, ευκολότερη ενσωμάτωση ΑΠΕ, αυξημένη αξιοπιστία, καλύτερη ευελιξία του συστήματος και ένα βιώσιμο ενεργειακό μέλλον.

2.2 Βασικές Αρχές Μηχανικής Μάθησης

Η Μηχανική Μάθηση (Machine Learning - ML) αποτελεί έναν κλάδο της τεχνητής νοημοσύνης και επικεντρώνεται στην ανάπτυξη αλγορίθμων και μοντέλων που επιτρέπουν στο σύστημα να μαθαίνει από τα δεδομένα που του παρέχονται. Αντί να προγραμματίζεται εξαρχής με συγκεκριμένους κανόνες, το ML επιτρέπει στο σύστημα να προσαρμόζεται και να βελτιώνεται με την εμπειρία που αποκτάει από τα διαφορά runs που εκτελούνται με την πάροδο του χρόνου [9].



Εικόνα 7 Διάγραμμα ροής ενός παραδοσιακού τρόπου προγραμματισμού (πανω) έναντι ενός προγράμματος Μηχανικής Μάθησης **[25]**

2.2.α Βασικές Έννοιες Μηχανικής Μάθησης

Η θεμελιώδης αρχή της Μηχανικής Μάθησης έγκειται στην εκμετάλλευση του **συνόλου δεδομένων (dataset)**, το οποίο συνιστά μια συλλογή παραδειγμάτων ή παρατηρήσεων, απαραίτητη τόσο για την εκπαίδευση όσο και για την επικύρωση των μοντέλων. Ειδικότερα, ένα ολοκληρωμένο σύνολο δεδομένων διαρθρώνεται σε τρεις κρίσιμες υποκατηγορίες: το **σύνολο εκπαίδευσης (training set)**, το οποίο χρησιμοποιείται για τη ρύθμιση των εσωτερικών παραμέτρων του αλγορίθμου· το **σύνολο επικύρωσης (validation set)**, που επιτρέπει τη βελτιστοποίηση των *υπερπαραμέτρων* του μοντέλου και την έγκαιρη αναγνώριση φαινομένων *υπερπροσαρμογής (overfitting)* κατά τη φάση ανάπτυξης· και το **σύνολο δοκιμής (test set)**, το οποίο διατηρείται ανέπαφο μέχρι την ολοκλήρωση της ανάπτυξης, προσφέροντας μια αντικειμενική εκτίμηση της γενικευτικής ικανότητας του μοντέλου σε δεδομένα που δεν έχει συναντήσει ποτέ στο παρελθόν [26].

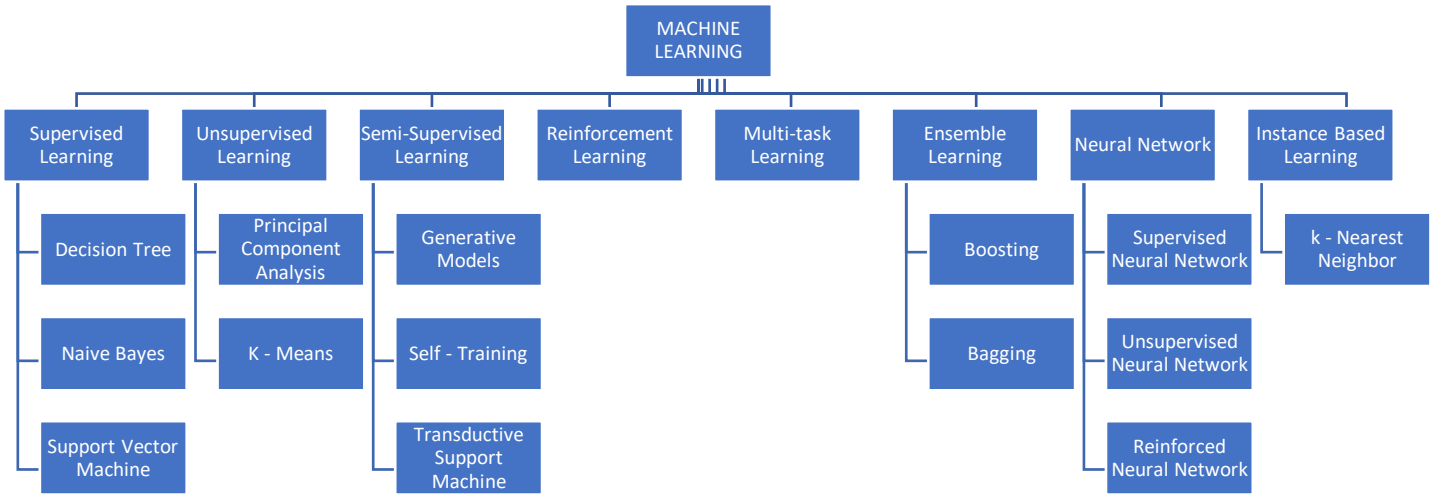
Η διαδικασία της εκπαίδευσης περιλαμβάνει την επαναληπτική προσαρμογή των βαρών του μοντέλου, με στόχο την ελαχιστοποίηση μιας **συνάρτησης κόστους (loss function)** σε σχέση με το σύνολο εκπαίδευσης, επιτρέποντας στο μοντέλο να αποκτήσει αναπαραστάσεις των υποκείμενων σχέσεων των δεδομένων. Η επικύρωση είναι ζωτικής σημασίας για τη διασφάλιση ότι το μοντέλο δεν έχει απλώς απομνημονεύσει τα δεδομένα εκπαίδευσης, αλλά έχει αναπτύξει την ικανότητα να γενικεύει τις προβλέψεις του σε νέα, άγνωστα δεδομένα. Τέλος, η **αξιολόγηση** στο σύνολο δοκιμής παρέχει την τελική, αμερόληπτη ένδειξη της αποτελεσματικότητας και της στιβαρότητας του μοντέλου έναντι πραγματικών προκλήσεων [27].

2.2.β Κατηγορίες Μηχανικής Μάθησης

Η μηχανική μάθηση διακρίνεται σε τρεις βασικές κατηγορίες, οι οποίες καθορίζονται από τον τρόπο εκπαίδευσης των αλγορίθμων και τη φύση των δεδομένων: την επιβλεπόμενη, τη μη επιβλεπόμενη και την ενισχυτική μάθηση.

- **Επιβλεπόμενη Μάθηση (Supervised Learning):** το μοντέλο εκπαιδεύεται σε δεδομένα που περιλαμβάνουν τόσο **εισόδους** όσο και τις αντίστοιχες **ετικέτες/εξόδους (labeled data)** [28]. Στόχος είναι η εκμάθηση μιας συνάρτησης που χαρτογραφεί τις εισόδους στις εξόδους, επιτρέποντας την πρόβλεψη για νέα, αόρατα δεδομένα.
    - **Παλινδρόμηση (Regression):** Πρόβλεψη συνεχών τιμών (π.χ., καταναλώσεις νοικοκυριών).
    - **Ταξινόμηση (Classification):** Πρόβλεψη διακριτών κατηγοριών (π.χ., διάγνωση ασθένειας: ναι/όχι).
- **Μη Επιβλεπόμενη Μάθηση (Unsupervised learning):** αναλύει δεδομένα χωρίς προκαθορισμένες ετικέτες (unlabeled data), ανακαλύπτοντας κρυμμένες δομές, πρότυπα ή σχέσεις. Το μοντέλο αναζητά εγγενείς ομοιότητες ή διαφορές στα δεδομένα.
    - **Ομαδοποίηση (Clustering):** Ομαδοποίηση παρόμοιων σημείων δεδομένων σε ομάδες (π.χ., ταξινόμηση φρούτων ανά μέγεθος και χρώμα).
    - **Ανάλυση Κανόνων Συσχέτισης (Association Rule Learning):** έχει ως πρωταρχικό στόχο την ανακάλυψη ισχυρών σχέσεων ή «κανόνων συσχέτισης» μεταξύ μεταβλητών σε μεγάλα σύνολα δεδομένων.
    - **Μείωση Διαστατικότητας (Dimensionality Reduction):** Μείωση του πλήθους των χαρακτηριστικών διατηρώντας την ουσιώδη πληροφορία.
- **Ενισχυτική Μάθηση (Reinforcement Learning):** επικεντρώνεται στον τρόπο με τον οποίο ένας **«πράκτορας» (agent)** μαθαίνει να λαμβάνει διαδοχικές αποφάσεις μέσα σε ένα **περιβάλλον (environment)**, με στόχο τη μεγιστοποίηση μιας **συνάρτησης ανταμοιβής** [29]. Ο πράκτορας δεν λαμβάνει εκ των προτέρων δεδομένα με σωστές απαντήσεις, αλλά μαθαίνει μέσω της αλληλεπίδρασης, της

δοκιμής και του λάθους, λαμβάνοντας θετικές ή αρνητικές **«ανταμοιβές»** για τις ενέργειές του.



Εικόνα 8 Απεικόνιση Κατηγοριών και υποκατηγοριών της Μηχανικής Μάθησης σε μορφή Δέντρου

2.2.γ Στάδια Μηχανικής Μάθησης

Η ανάπτυξη ενός μοντέλου μηχανικής μάθησης ακολουθεί μια δομημένη πορεία, η οποία περιλαμβάνει τα εξής βασικά στάδια:

1. **Προσδιορισμός Προβλήματος και Συλλογή Δεδομένων:** Το αρχικό στάδιο συνίσταται στον ακριβή προσδιορισμό του προβλήματος που καλείται να επιλύσει το σύστημα ΜΜ. Αυτό περιλαμβάνει τη σαφή διατύπωση των στόχων και τον καθορισμό των αναμενόμενων αποτελεσμάτων. Ακολουθεί η συστηματική συλλογή δεδομένων από ποικίλες πηγές, διασφαλίζοντας την επάρκεια, την ποικιλομορφία και την αντιπροσωπευτικότητά τους σε σχέση με το υπό μελέτη φαινόμενο. Η ποιότητα των δεδομένων σε αυτό το στάδιο επηρεάζει άμεσα την επιτυχία των επόμενων βημάτων.
2. **Προεπεξεργασία και Μηχανική Χαρακτηριστικών:** Τα ακατέργαστα δεδομένα σπάνια είναι άμεσα κατάλληλα για την εκπαίδευση μοντέλων. Το στάδιο της προεπεξεργασίας περιλαμβάνει τεχνικές για τον καθαρισμό των δεδομένων (π.χ., χειρισμός ελλειπουσών τιμών, αφαίρεση θορύβου, εντοπισμός και αντιμετώπιση ακραίων τιμών), τη μετατροπή τους σε κατάλληλη μορφή (π.χ., κωδικοποίηση κατηγορικών μεταβλητών, κανονικοποίηση αριθμητικών δεδομένων) και τη **μηχανική χαρακτηριστικών (feature engineering)** κατά την οποία

δημιουργούνται νέα, πιο ενημερωτικά χαρακτηριστικά από τα υπάρχοντα, με σκοπό τη βελτίωση της απόδοσης του μοντέλου.

3. **Επιλογή και Εκπαίδευση Μοντέλου:** Μετά την προετοιμασία των δεδομένων, επιλέγεται το κατάλληλο μοντέλο μηχανικής μάθησης, λαμβάνοντας υπόψη τον τύπο του προβλήματος (εποπτευόμενη, μη εποπτευόμενη μάθηση κ.λπ.) και τα χαρακτηριστικά των δεδομένων. Το επιλεγμένο μοντέλο στη συνέχεια εκπαιδεύεται σε ένα υποσύνολο των δεδομένων (training set), προσαρμόζοντας τις εσωτερικές του παραμέτρους (βάρη) ώστε να ελαχιστοποιεί μια συνάρτηση κόστους (loss function) και να μαθαίνει τα υποκείμενα μοτίβα.
4. **Αξιολόγηση και Βελτιστοποίηση Μοντέλου:** Η αξιολόγηση του μοντέλου είναι ένα κρίσιμο στάδιο που διασφαλίζει την ικανότητά του να γενικεύει σε νέα, αόρατα δεδομένα. Αυτό επιτυγχάνεται χρησιμοποιώντας ένα ανεξάρτητο σύνολο δεδομένων (test set). Χρησιμοποιούνται διάφορες μετρικές απόδοσης (π.χ., ακρίβεια, ανάκληση, F1-score για ταξινόμηση, R-squared, Mean Squared Error για παλινδρόμηση) ανάλογα με τον στόχο. Σε αυτό το στάδιο συχνά πραγματοποιείται βελτιστοποίηση των υπερπαραμέτρων του μοντέλου, μεθόδους όπως η διασταυρούμενη επικύρωση (cross-validation) για την αποφυγή υπερ-προσαρμογής (overfitting).
5. **Ανάπτυξη και Παρακολούθηση:** Το τελικό στάδιο περιλαμβάνει την ανάπτυξη (deployment) του βελτιστοποιημένου μοντέλου σε ένα λειτουργικό περιβάλλον, όπου μπορεί να χρησιμοποιηθεί για την παραγωγή προβλέψεων ή αποφάσεων σε πραγματικό χρόνο. Μετά την ανάπτυξη, είναι απαραίτητη η συνεχής παρακολούθηση της απόδοσης του μοντέλου. Αυτό περιλαμβάνει την παρακολούθηση για φαινόμενα όπως το "data drift" (αλλαγή στην κατανομή των δεδομένων εισόδου) ή το "model drift" (μείωση της ακρίβειας του μοντέλου με την πάροδο του χρόνου), τα οποία ενδέχεται να απαιτήσουν επανεκπαίδευση ή αναθεώρηση του μοντέλου.

2.3 Συνεργατικά Σχήματα Μηχανικής Μάθησης

Η Συνεργατική Μηχανική Μάθηση, συχνά αναφερόμενη και ως Ομοσπονδιακή Μάθηση (Federated Learning - FL) αποτελεί μια εξέλιξη της Μηχανικής Μάθησης και εστιάζει στη συνεργασία και τον συντονισμό πολλαπλών παραγόντων για την επίλυση προβλημάτων, χωρίς να απαιτείται η συγκέντρωση των δεδομένων τους σε μία κεντρική τοποθεσία, σε αντίθεση με την παραδοσιακή μηχανική μάθηση που ένα μοντέλο εκπαιδεύεται σε ένα σύνολο δεδομένων [30]. Η συνεργατική προσέγγιση επιτρέπει την αλληλεπίδραση και την ανταλλαγή πληροφοριών μεταξύ διαφορετικών παραγόντων στο πλαίσιο ΕΗΔ, η συνεργατική μάθηση μπορεί να επιτρέψει σε διαφορετικές συσκευές και συστήματα να συνεργαστούν για τη βελτιστοποίηση της ενεργειακής απόδοσης, διατηρώντας παράλληλα την ιδιωτικότητα των επιμέρους δεδομένων.

2.3.α Οι αρχές της Συνεργατικής Μάθησης

Οι βασικές αρχές της συνεργατικής μηχανικής μάθησης περιλαμβάνουν:

- **Αποκέντρωση Δεδομένων (Data Decentralization):** Στη συνεργατική μηχανική μάθηση τα ακατέργαστα δεδομένα παραμένουν στην πηγή τους. Αντί να συγκεντρώνονται σε έναν κεντρικό χώρο αποθήκευσης, τα δεδομένα παραμένουν τοπικά στις συσκευές, τους οργανισμούς ή τους διακομιστές που τα παράγουν. Αυτό επιλύει σημαντικές προκλήσεις ιδιωτικότητας και ασφάλειας, καθώς μειώνεται δραστικά ο κίνδυνος μαζικών παραβιάσεων δεδομένων και διευκολύνεται η συμμόρφωση με αυστηρούς κανονισμούς προστασίας προσωπικών δεδομένων [31].
- **Τοπική Εκπαίδευση (Local Training):** Κάθε συμμετέχων στο συνεργατικό σχήμα εκπαιδεύει ένα μοντέλο μηχανικής μάθησης χρησιμοποιώντας αποκλειστικά

τα δικά του τοπικά δεδομένα. Αυτή η διαδικασία επιτρέπει την αξιοποίηση των δεδομένων χωρίς την ανάγκη μεταφοράς τους, διασφαλίζοντας ότι η ευαίσθητη πληροφορία δεν εκτίθεται. Η τοπική εκπαίδευση μπορεί να προσαρμοστεί στις ιδιαιτερότητες των τοπικών δεδομένων, ενώ παράλληλα συμβάλλει στη συνολική βελτίωση του κοινού μοντέλου [32].

- **Ανταλλαγή Ενημερώσεων Μοντέλου (Model Update Exchange):** Αντί για την ανταλλαγή δεδομένων, οι συμμετέχοντες ανταλλάσσουν μόνο συγκεντρωτικές πληροφορίες για τις ενημερώσεις του μοντέλου που προκύπτουν από την τοπική τους εκπαίδευση. Αυτές οι ενημερώσεις, συνήθως με τη μορφή βαρών ή παραμέτρων του νευρωνικού δικτύου, είναι αφαιρετικές και δεν αποκαλύπτουν άμεσα τα υποκείμενα δεδομένα. Ένας κεντρικός διακομιστής/ κόμβος (ή σε αποκεντρωμένα συστήματα, άλλοι πελάτες) συλλέγει αυτές τις ενημερώσεις [33].
- **Συγκέντρωση και Άθροιση (Aggregation and Averaging):** Οι ενημερώσεις μοντέλου που λαμβάνονται από τους διάφορους πελάτες συγκεντρώνονται και συνδυάζονται για να δημιουργήσουν ένα βελτιωμένο, καθολικό μοντέλο. Ο πιο κοινός τρόπος είναι η σταθμισμένη μέση τιμή των ενημερώσεων, λαμβάνοντας υπόψη παράγοντες όπως το μέγεθος του συνόλου δεδομένων κάθε πελάτη ή την ποιότητα της τοπικής εκπαίδευσης. Αυτή η διαδικασία διασφαλίζει ότι η συλλογική γνώση από όλους τους συμμετέχοντες ενσωματώνεται στο κοινό μοντέλο [31].
- **Επαναληπτική Βελτίωση (Iterative Refinement):** Η συνεργατική μηχανική μάθηση λειτουργεί σε επαναληπτικούς κύκλους. Μετά τη συγκέντρωση και συνάθροιση των ενημερώσεων, το βελτιωμένο καθολικό μοντέλο αποστέλλεται πίσω στους πελάτες. Κάθε πελάτης επικαιροποιεί το τοπικό του μοντέλο με την τελευταία έκδοση του καθολικού μοντέλου και συνεχίζει την τοπική εκπαίδευση. Αυτή η συνεχής ανατροφοδότηση επιτρέπει στο μοντέλο να συγκλίνει και να βελτιώνεται με την πάροδο του χρόνου, αξιοποιώντας την εμπειρία όλων των συμμετεχόντων [32].

2.4 Σχετική Εργασία

Η παρούσα ενότητα εξετάζει τη σχετική βιβλιογραφία, ομαδοποιώντας τις υπάρχουσες μελέτες σε βασικούς τομείς εφαρμογής της FL στα έξυπνα δίκτυα.

2.4.α Απόκριση Ζήτησης (Demand Response)

Η απόκριση ζήτησης (DR) αποτελεί ακρογωνιαίο λίθο για τη διαχείριση της ισορροπίας προσφοράς-ζήτησης στα έξυπνα δίκτυα, ενθαρρύνοντας τους καταναλωτές να μεταβάλλουν την κατανάλωσή τους σε περιόδους αιχμής ή υψηλών τιμών. Η εφαρμογή της FL στον τομέα αυτό είναι ιδιαίτερα επωφελής, καθώς τα δεδομένα κατανάλωσης των χρηστών είναι ευαίσθητα. Έρευνες έχουν επικεντρωθεί σε μεθόδους βελτιστοποίησης της DR που βασίζονται στην FL, με έμφαση στην προστασία της ιδιωτικότητας των χρηστών [34]. Πιο συγκεκριμένα, προτείνονται πλαίσια FL που επιτρέπουν την εκπαίδευση μοντέλων DR χωρίς να εκτίθενται τα ατομικά πρότυπα κατανάλωσης. Επιπρόσθετα, έχουν αναπτυχθεί και διερευνηθεί μοντέλα FL για την απόκριση ζήτησης σε συστήματα cloud-edge μέσω μηχανισμών δημοπρασιών, προάγοντας τη βιώσιμη χρήση της τεχνητής νοημοσύνης σε κατανεμημένα περιβάλλοντα [35]. Αυτές οι προσεγγίσεις αναδεικνύουν την ικανότητα της FL να συνδυάζει την αποτελεσματικότητα της DR με τις επιταγές της προστασίας δεδομένων.

2.4.β Βέλτιστη Ροή Ισχύος (Optimal Power Flow)

Η επίλυση του προβλήματος της βέλτιστης ροής ισχύος (OPF) είναι θεμελιώδης για την αποδοτική και ασφαλή λειτουργία των ηλεκτρικών δικτύων. Παραδοσιακά, η επίλυση του OPF απαιτεί συγκεντρωτική πρόσβαση σε δεδομένα από ολόκληρο το σύστημα, κάτι που

μπορεί να είναι προβληματικό σε κατανεμημένα και ιδιωτικά δίκτυα. Η FL προσφέρει μια εναλλακτική λύση, επιτρέποντας την κατανεμημένη επίλυση του OPF. Έρευνες έχουν επιδείξει τη δυνατότητα της FL για την εξεύρεση κατανεμημένων λύσεων OPF σε έξυπνα δίκτυα [36], καθιστώντας δυνατή τη συνεργατική βελτιστοποίηση χωρίς την κοινοποίηση ευαίσθητων τοπικών δεδομένων. Παράλληλα, έχουν αναπτυχθεί και μοντέλα FL που υποστηρίζουν την κατανεμημένη βελτιστοποίηση ενέργειας, συμβάλλοντας στην αποδοτικότερη διαχείριση των ενεργειακών πόρων σε ευρύτερα συστήματα [37]. Αυτές οι μελέτες υπογραμμίζουν την ικανότητα της FL να διαχειρίζεται πολύπλοκα προβλήματα βελτιστοποίησης σε κατανεμημένα περιβάλλοντα.

2.4.γ Ανανεώσιμες Πηγές Ενέργειας (Renewable Energy)

Οι ανανεώσιμες πηγές ενέργειας (ΑΠΕ) αποτελούν κεντρικό πυλώνα της ενεργειακής μετάβασης, όμως η ενσωμάτωσή τους στα δίκτυα παρουσιάζει προκλήσεις λόγω της μεταβλητότητας και της διακοπτόμενης φύσης τους. Η FL μπορεί να διαδραματίσει καθοριστικό ρόλο στην αποτελεσματική διαχείριση και ενσωμάτωση των ΑΠΕ. Μια περιεκτική ανασκόπηση έχει αναδείξει τις δυνατότητες, τις προκλήσεις και τις μελλοντικές κατευθύνσεις της FL σε εφαρμογές ανανεώσιμων πηγών ενέργειας [38], καλύπτοντας τομείς όπως η πρόβλεψη, ο έλεγχος και η βελτιστοποίηση. Επιπλέον, έχουν προταθεί λύσεις FL για την προστασία της ιδιωτικότητας στην κατανάλωση ανανεώσιμης ενέργειας για τοπικές κοινότητες [39], διασφαλίζοντας ότι τα δεδομένα κατανάλωσης παραμένουν ιδιωτικά, ενώ παράλληλα συμβάλλουν στην αποτελεσματική διαχείριση της ενέργειας.

2.4.δ Προστασία της Ιδιωτικότητας (Privacy)

Η προστασία της ιδιωτικότητας είναι μια εγγενής και κρίσιμη πτυχή της FL, ειδικά σε εφαρμογές έξυπνων δικτύων όπου διαχειρίζονται ευαίσθητα δεδομένα καταναλωτών και υποδομών. Πέρα από την ενσωματωμένη προστασία που προσφέρει η FL, η τεχνολογία μπορεί επίσης να χρησιμοποιηθεί για την ενίσχυση της κυβερνοασφάλειας. Έχει προταθεί ένα συνδυαστικό μοντέλο Res2-UNeXt με FL για την ανίχνευση και ταξινόμηση κυβερνοεπιθέσεων σε συστήματα έξυπνων δικτύων πολλαπλών περιοχών [40], επιδεικνύοντας την ικανότητα της FL να συνεισφέρει στην ασφάλεια, διατηρώντας παράλληλα την ιδιωτικότητα των δεδομένων που χρησιμοποιούνται για την εκπαίδευση των μοντέλων ανίχνευσης.

2.4.ε Ανίχνευση Ανωμαλιών (Anomaly Detection)

Η ανίχνευση ανωμαλιών είναι ζωτικής σημασίας για την έγκαιρη αναγνώριση δυσλειτουργιών, σφαλμάτων ή ακόμα και κακόβουλων επιθέσεων στα έξυπνα δίκτυα. Η εφαρμογή της FL σε αυτόν τον τομέα επιτρέπει την αποτελεσματική ανίχνευση ανωμαλιών σε κατανεμημένα δεδομένα, όπως αυτά από έξυπνους μετρητές, διατηρώντας παράλληλα την ιδιωτικότητα. Εργασίες έχουν παρουσιάσει μεθόδους ανίχνευσης ανωμαλιών που διατηρούν την ιδιωτικότητα σε δεδομένα έξυπνων μετρητών μέσω FL [41]. Επιπλέον, έχουν διερευνηθεί υβριδικά μοντέλα, όπως τα Generative Adversarial Networks με νευρο-ασαφή αλγορίθμους, για την ανίχνευση σφαλμάτων στα έξυπνα δίκτυα [42]. Ένα πλαίσιο FL έχει επίσης προταθεί ως μέθοδος ανίχνευσης ανωμαλιών για έξυπνα ηλεκτρικά δίκτυα [43], επιβεβαιώνοντας την αποτελεσματικότητα της κατανεμημένης προσέγγισης στην αναγνώριση μη φυσιολογικής συμπεριφοράς.

2.4.στ Πρόβλεψη Φορτίου (Load Forecasting)

Η ακριβής πρόβλεψη φορτίου είναι απαραίτητη για τον αποτελεσματικό προγραμματισμό και τη λειτουργία των έξυπνων δικτύων. Η FL προσφέρει μια λύση για την πρόβλεψη φορτίου, ειδικά σε επίπεδο κατοικίας ή περιφερειακό, όπου τα δεδομένα μπορεί να είναι ιδιωτικά. Έχουν αναπτυχθεί μοντέλα βραχυπρόθεσμης πρόβλεψης

οικιακού φορτίου βασισμένα σε FL [44], τα οποία επιτρέπουν την εκπαίδευση μοντέλων χρησιμοποιώντας δεδομένα από πολλά νοικοκυριά χωρίς την άμεση κοινοποίησή τους. Επιπλέον, η FL σε συνδυασμό με νευρωνικά δίκτυα LSTM έχει χρησιμοποιηθεί για την περιφερειακή πρόβλεψη φορτίου [45], βελτιώνοντας την ακρίβεια των προβλέψεων σε μεγαλύτερες γεωγραφικές περιοχές. Αξίζει να σημειωθεί ότι η προστασία της ιδιωτικότητας στην κατανάλωση ανανεώσιμης ενέργειας για τοπικές κοινότητες μέσω FL, όπως αναφέρθηκε προηγουμένως [39], έχει επίσης άμεση συνάφεια με την πρόβλεψη φορτίου, καθώς τα δεδομένα κατανάλωσης αποτελούν βασική εισαγωγή για τέτοια μοντέλα.

Αφήσαμε την πρόβλεψη φορτίου ως τελευταίο παράδειγμα μιας και αντικατοπτρίζει περισσότερο την παρούσα πτυχιακή. Περισσότερα θα δούμε στο παρακάτω κεφάλαιο.

ΚΕΦΑΛΑΙΟ 3 Εφαρμογή Συνεργατικού Σχήματος Μηχανικής Μάθησης στο ΈΗΔ

Στο παρόν κεφάλαιο, θα παρουσιαστεί η διαδικασία του σχεδιασμού και της υλοποίησης ενός συνεργατικού σχήματος μηχανικής μάθησης, ειδικά προσαρμοσμένου για εφαρμογές στο Έξυπνο Ηλεκτρικό Δίκτυο (ΈΗΔ). Αρχικά, θα αναλυθούν οι θεμελιώδεις αρχές που διέπουν τον σχεδιασμό τέτοιων συστημάτων, εστιάζοντας στην αρχιτεκτονική, την επιλογή των κατάλληλων αλγορίθμων μηχανικής μάθηση, την επικοινωνίας των επιμέρους στοιχείων του συστήματος, καθώς και την συλλογή δεδομένων. Στη συνέχεια, θα περιγραφεί λεπτομερώς η διαδικασία ανάπτυξης του προτεινόμενου σχήματος, συμπεριλαμβανομένων των τεχνολογιών και των εργαλείων που χρησιμοποιήθηκαν για την υλοποίησή του, ενώ, τέλος θα σχολιαστούν τα πειράματα που πραγματοποιήσαμε και οι τρόποι με τους οποίους βελτιστοποιήσαμε το σύστημά μας.

3.1 Σχεδιασμός του Συνεργατικού Σχήματος

Για τον σχεδιασμό του ΣΣΜΜ, χρειάστηκε να αναζητήσουμε σχετική εργασία στην διεθνή βιβλιογραφία, μέσω της οποίας, λαμβάνοντας υπόψιν τους περιορισμούς που γνωρίζαμε ότι θα αντιμετωπίσαμε, καταλήξαμε στο σύστημα που θα αναλύσουμε παρακάτω.

3.1.α Στόχος

Το βασικότερο βήμα για τον σχεδιασμό ήταν ο ορθός και ακριβής ορισμός του στόχου που θέλουμε να επιτεύξει το Συνεργατικό μας Σχήμα. Αυτός είναι ο εξής:

Θέλουμε να αναπτύξουμε ένα Συνεργατικό Σχήμα Μηχανικής Μάθησης για χρήση στο Ευφυές Ηλεκτρικό Δίκτυο το οποίο θα προσομοιώνει μία μικρή πόλη 1000 νοικοκυριών, χωρισμένα σε 4 Ταχυδρομικούς Κώδικες (ΤΚ), για ένα χρονικό διάστημα 25 ετών. Κατά την διαδικασία της προσομοίωσης, το σύστημα θα πρέπει να κάνει προβλέψεις για την μέση ημερήσια κατανάλωση ηλεκτρικής ενέργειας των διαφορετικών ΤΚ, με την χρήση συνεργατικής μάθησης. Αναπόσπαστο κομμάτι του στόχου μας είναι και η δημιουργία ενός πίνακα ελέγχου που θα μας οπτικοποιεί τα αποτελέσματα και θα μας δίνει μετρικές για την ακρίβεια του συστήματος μας.

3.1.β Αρχιτεκτονική Συνεργατικού Συστήματος

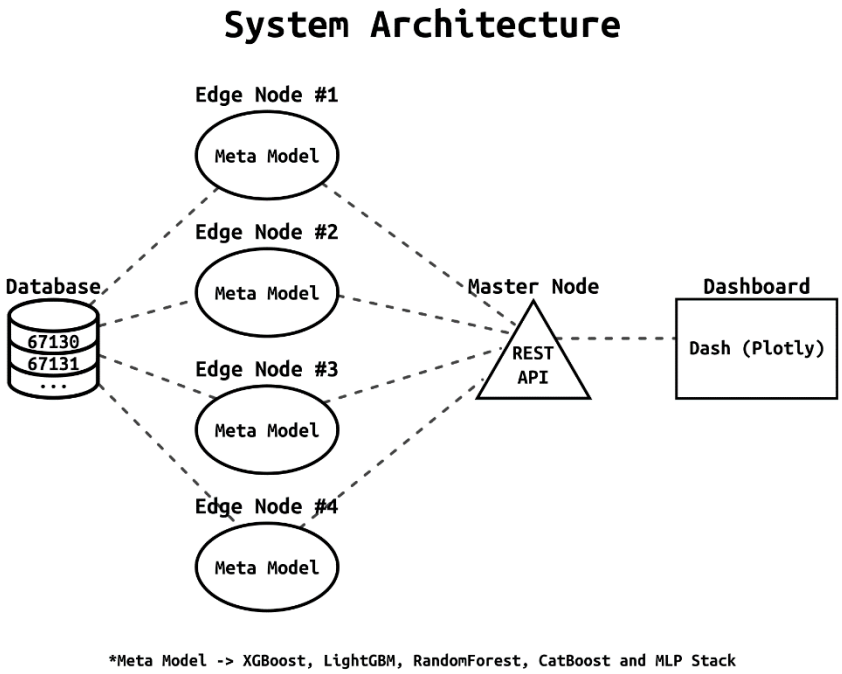
Αφού ορίσαμε τον στόχο μας, προχωρήσαμε στην δημιουργία της αρχιτεκτονικής του ΣΣΜΜ. Αυτή αποτελείται από:

- **Βάση Δεδομένων (Database):** Κρατάει τις τιμές ημερήσιας κατανάλωσης των νοικοκυριών, σε ξεχωριστούς πίνακες για κάθε ΤΚ. Σε ένα πραγματικό σενάριο, και όχι σε προσομοίωση όπως αυτή που πραγματοποιεί το σύστημά μας, ο κάθε Τοπικός Κόμβος που βρίσκεται σε κάθε έναν από τους Ταχυδρομικούς Κώδικες, θα είχε την δικιά του βάση δεδομένων, το οποίο εξυπηρετεί και στην αποκεντροποίηση των δεδομένων, όπως αναφέρθηκε και στο παραπάνω κεφάλαιο. Στην προσομοίωση μας, αυτό επιτυγχάνεται με την δημιουργία μια κεντρικής βάσης της οποίας οι πίνακες αντιστοιχούν στους 4 διαφορετικούς ΤΚ της πόλης.
- **Τοπικοί Κόμβοι (Edge Nodes):** Πρόκειται για την προσομοίωση ενός υπολογιστικού συστήματος που αποθηκεύει και επεξεργάζεται την κατανάλωση του δικτύου. Αντιστοιχεί ένας για κάθε Ταχυδρομικό Κώδικα. Στόχος τους είναι, να διαβάζουν τα στοιχεία από την βάση δεδομένων και με την χρήση μοντέλων μηχανικής μάθησης να μας παρέχουν προβλέψεις για την μελλοντική κατανάλωση. Λόγω της περιορισμένης υπολογιστικής ισχύος, επικοινωνούν μόνο

με τον κεντρικό διακομιστή, ενώ όπως είδαμε παραπάνω, στην πραγματικότητα επικοινωνούν και μεταξύ τους.

- **Κεντρικός Κόμβος (Master Node):** Δέχεται όλες τις ενημερώσεις των μοντέλων που παράγονται από τους Τοπικούς Κόμβους του δικτύου μας.
- **Πίνακας Ελέγχου (Dashboard):** Οπτικοποιεί τα αποτελέσματα της προσομοίωσης σε αληθινό χρόνο, ενώ μας δίνει και απαραίτητες μετρικές για την αξιολόγηση του μοντέλου πρόβλεψης.

Πιο αναλυτικά μπορούμε να δούμε την αρχιτεκτονική του συστήματος στο παρακάτω διάγραμμα που δημιουργήσαμε για την πιο εύκολη κατανόηση του Συστήματος.



Εικόνα 9 Η Αρχιτεκτονική του Συνεργατικού Συστήματος Μηχανικής Μάθησης

3.1.γ Μηχανισμός Επικοινωνίας

Ο τρόπος με τον οποίο τα υποσυστήματα του ΣΣΜΜ λειτουργούν είναι με την χρήση ενός RESTful API (Application Programming Interface) που δημιουργήσαμε. Η **Representational State Transfer (REST)** συνιστά ένα αρχιτεκτονικό στυλ λογισμικού που καθορίζει ένα σύνολο περιορισμών για τη δημιουργία κατανεμημένων συστημάτων. Η κεντρική ιδέα της REST περιστρέφεται γύρω από τους πόρους (resources). Κάθε πόρος αναπαρίσταται από δεδομένα και διαθέτει ένα μοναδικό αναγνωριστικό **(Uniform Resource Identifier - URI)**, όπως ένα URL [46]. Η αλληλεπίδραση με αυτούς τους πόρους γίνεται μέσω ενός ενιαίου συνόλου από stateless λειτουργίες, οι οποίες αντιστοιχούν συνήθως στις μεθόδους του πρωτοκόλλου HTTP:

- **GET:** Για ανάκτηση της αναπαράστασης ενός πόρου.
- **POST:** Για τη δημιουργία ενός νέου πόρου ή την υποβολή δεδομένων.
- **PUT:** Για την πλήρη ενημέρωση (αντικατάσταση) ενός υπάρχοντος πόρου.
- **DELETE:** Για τη διαγραφή ενός πόρου.

Ένα από τα βασικά χαρακτηριστικά της REST είναι η μη διατήρηση κατάστασης (statelessness). Αυτό σημαίνει ότι κάθε αίτημα από τον client προς τον server περιέχει όλες τις απαραίτητες πληροφορίες για την επεξεργασία του, χωρίς ο server να χρειάζεται να διατηρεί καμία πληροφορία από προηγούμενα αιτήματα του ίδιου client. Αυτή η αρχή ενισχύει την επεκτασιμότητα, καθώς ο server δεν δεσμεύει πόρους για να θυμάται την

κατάσταση κάθε client, επιτρέποντας την αποτελεσματική διαχείριση μεγάλου αριθμού ταυτόχρονων αιτημάτων. Προφανώς σε ένα ρεαλιστικό σενάριο, ο κεντρικός μας κόμβος θα πρέπει να κρατάει αυτά τα δεδομένα. Λόγω περιορισμένης υπολογιστικής ισχύς, προτιμήσαμε να χρησιμοποιήσουμε τους πόρους μας στην εκπαίδευση του μοντέλου.

Το REST API, εκτελείται στο Master Node. Έτσι επιτυγχάνεται η αμφίδρομη επικοινωνία με τα Edge Nodes καθώς και η επικοινωνία με το Dashboard ώστε να έχουμε την οπτικοποίηση των αποτελεσμάτων μας.

3.1.δ Συλλογή Δεδομένων

Προκειμένου να αναπτύξουμε ένα αξιόπιστο μοντέλο πρόβλεψης της κατανάλωσης ενέργειας, έπρεπε να συλλέξουμε δεδομένα που περιλαμβάνουν την κατανάλωση ενέργειας των νοικοκυριών μέσα στα χρόνια. Ωστόσο, η συλλογή αυτών των δεδομένων είναι ιδιαίτερα δύσκολη, καθώς οι πάροχοι ενέργειας δεν παρέχουν τόσο λεπτομερείς πληροφορίες σχετικά με τις καταναλώσεις.

Αντιμετωπίζοντας αυτή την πρόκληση, αποφασίσαμε να αναπτύξουμε τα δικά μας δεδομένα τα οποία βασίσαμε σε γενικές πληροφορίες από καταξιωμένες πηγές. Τέτοιες πηγές [47] μας δίνουν την μέση κατανάλωση ενός νοικοκυριού στην Ελλάδα για το έτος 2011-2012 γύρω στα 10kW ημερησίως, ενώ η θερμική κατανάλωση είναι περίπου στα 25kWh. Προφανώς μέσα στα χρόνια από την δημοσίευση της παρούσας πηγής, το προσκήνιο έχει αλλάξει. Με την άνοδο του φυσικού αερίου και την μείωση της χρήσης πετρελαίου για θέρμανση, καθώς και με τις επιδοτήσεις για φωτοβολταϊκά και νέες και αποδοτικότερες συσκευές, δεν μπορούμε να ξέρουμε αν αυτά τα στοιχεία είναι ακόμα σχετικά. Άλλες έρευνες μας δείχνουν μια μικρή άνοδο στην κατανάλωση, λόγω χρήσης ηλεκτρικών μέσων θέρμανσης και ψύξης [48].

Χρησιμοποιώντας αυτές τις αξιόπιστες, αν και γενικευμένες πηγές δεδομένων, καταλήξαμε σε μερικούς κανόνες κάτω από τους οποίους θα δημιουργήσουμε το σύνολο δεδομένων που θα χρησιμοποιήσουμε για την εκπαίδευση του μοντέλου μας. Παρόλο που τα δεδομένα μας είναι σε έναν βαθμό πλασματικά, βασίζονται σε όσες περισσότερες έμπιστε πηγές μπορούσαμε να βρούμε και προσομοιώνουν αρκετά ρεαλιστικά την κατανάλωση ενέργειας μιας μικρής πόλης.

3.1.ε Επιλογή Μοντέλων Μάθησης

Τα μοντέλα που επιλέξαμε για την εκπόνηση του πειραματικού μέρους της εργασίας χωρίζονται σε 3 μέρη:

1. **Βασικά Μοντέλα:**
    a. *XGBoost (Extreme Gradient Boosting):* Είναι ένα εξαιρετικά αποδοτικό και ευέλικτο μοντέλο βασισμένο σε δέντρα αποφάσεων, το οποίο χρησιμοποιεί την τεχνική του boosting [49]. Είναι γνωστό για την υψηλή του ακρίβεια, την ταχύτητα εκπαίδευσης και την ικανότητά του να χειρίζεται μη γραμμικές σχέσεις και αλληλεπιδράσεις μεταξύ μεταβλητών. Η βελτιστοποίηση των υπερπαραμέτρων υποδηλώνει μια προσπάθεια λεπτομερούς ρύθμισης για βέλτιστη απόδοση.
    b. *LightGBM: (Light Gradient Boosting Machine):* Παρόμοιο με το XGBoost, είναι ένα άλλο πλαίσιο gradient boosting βασισμένο σε δέντρα [50]. Ξεχωρίζει για την ταχύτητά του και την χαμηλή χρήση μνήμης, καθιστώντας το κατάλληλο για μεγάλα σύνολα δεδομένων και ενδεχομένως για περιορισμένους πόρους σε ένα edge node περιβάλλον. *RandomForestRegressor:* Οι τυχαίοι δασμοί είναι ένα μοντέλο συνόλου (ensemble) που βασίζεται σε πολλαπλά δέντρα αποφάσεων χρησιμοποιώντας τη μέθοδο του bagging [51]. Είναι γνωστό για την ισχυρότητά του έναντι της υπερεκπαίδευσης (overfitting), την ικανότητά

του να χειρίζεται γραμμικές και μη γραμμικές σχέσεις, και την ερμηνευσιμότητά του σε κάποιο βαθμό.

c. *CatBoostRegressor:* Το CatBoost είναι ένα μοντέλο gradient boosting που αναπτύχθηκε από τη Yandex [52]. Ιδιαίτερο χαρακτηριστικό του είναι η ενσωματωμένη διαχείριση κατηγορικών μεταβλητών και η ανθεκτικότητά του σε υπερεκπαίδευση μέσω μιας τεχνικής που ονομάζεται "ordered boosting"

d. *GradientBoostingRegressor:* Αυτός ο αλγόριθμος είναι η γενική βάση των XGBoost, LightGBM και CatBoost. Η συμπερίληψή του ως αυτόνομο μοντέλο παρέχει μια πιο "γενική" εφαρμογή της ιδέας του boosting, λειτουργώντας ως ένα συμπληρωματικό baseline.

e. *MLPRegressor: (Multi-layer Perceptron Regressor):* Το MLP είναι ένα τεχνητό νευρωνικό δίκτυο (Artificial Neural Network - ANN) με τουλάχιστον μία κρυφή στρώση [53]. Είναι ικανό να μοντελοποιεί εξαιρετικά μη γραμμικές σχέσεις και να ανακαλύπτει περίπλοκα πρότυπα στα δεδομένα, κάτι που είναι συχνό σε δεδομένα κατανάλωσης ενέργειας.

2. **Μετα-Μοντέλο:**
   - *XGBoost (ως Meta-Model):* Το μετα-μοντέλο μαθαίνει πώς να συνδυάζει τις προβλέψεις των βασικών μοντέλων για να παράγει μια τελική, βελτιστοποιημένη πρόβλεψη [54]. Η ικανότητά του να χειρίζεται τη μη γραμμικότητα επιτρέπει την εκμάθηση σύνθετων αλληλεπιδράσεων μεταξύ των προβλέψεων των βασικών μοντέλων.
3. **Εξαγωγή Χαρακτηριστικών:**
   a. *Lag Features (prev_day_power, prev_7_day_power, prev_30_day_power):* Είναι κρίσιμα για χρονοσειρές, καθώς η κατανάλωση ενέργειας είναι ισχυρά αυτοσυσχετιζόμενη [55], δηλαδή εξαρτάται σε μεγάλο βαθμό από τον εαυτό της. Προφανώς υπάρχουν και εξωτερικοί παράγοντες, αλλά σε έναν μεγάλο βαθμό αυτοί δεν επηρεάζουν σε τόσο μεγάλο βαθμό.

   b. *Temporal Features (year, month, day, season):* Επιτρέπουν στα μοντέλα να συλλάβουν την εποχικότητα και τις τάσεις στην κατανάλωση ενέργειας (π.χ., αύξηση λόγω θέρμανσης/κλιματισμού).

Στην διεθνή βιβλιογραφία είναι συχνή η χρήση Recurrent Neural Networks (RNNs) και πιο συγκεκριμένα Long Short-Term Memory (LSTMs) καθώς και παραδοσιακά μοντέλα χρονοσειρών όπως [ARIMA/SARIMA (AutoRegressive Integrated Moving Average / Seasonal ARIMA)]. Η χρήση των πρώτων κρίθηκε απαγορευτική λόγω του υπολογιστικού κόστους που δυστυχώς δεν είχαμε τον εξοπλισμό να υποστηρίξουμε, ενώ οι χρονοσειρές δεν είναι θα ήταν τόσο αποδοτικές [44].

3.2 Υλοποίηση και Τεχνικές Λεπτομέρειες

Παρακάτω θα εξηγήσουμε τεχνικά το πως αναπτύξαμε την υλοποίηση μας, τι εργαλεία χρησιμοποιήσαμε και τις πειραματικές διαδικασίες.

3.2.α Περιβάλλον Υλοποίησης

Η παρούσα πτυχιακή εργασία αναπτύχθηκε σε Linux με την χρήση python και βιβλιοθηκών μέσα σε περιβάλλον Anaconda.

Η Python, ως γλώσσα προγραμματισμού επιλέχθηκε, λόγω του πλήθους των βιβλιοθηκών που έχει, ειδικότερα για μηχανική μάθηση. Πιο συγκεκριμένα επιλέχθηκε η έκδοση *3.12.9* η οποία κατά την εκπόνηση της παρούσας πτυχιακής εργασίας ήταν η τελευταία σταθερή έκδοση.

Για να αποφύγουμε κινδύνους σχετικούς με την ακεραιότητα του συστήματος και άλλα προβλήματα σχετικά με εκδόσεις βιβλιοθηκών χρησιμοποιήσαμε το Anaconda. Το Anaconda πρόκειται για ένα απομονωμένο περιβάλλον εργασίας Python.

Οι βιβλιοθήκες που χρησιμοποιήσαμε είναι οι εξής:

- argparse: Παρέχει έναν στιβαρό μηχανισμό για την ανάλυση ορισμάτων γραμμής εντολών, επιτρέποντας την ευέλικτη παραμετροποίηση scripts και εφαρμογών.
- catboost: Αποτελεί μια εξελιγμένη βιβλιοθήκη για Gradient Boosting σε δέντρα αποφάσεων, διακρινόμενη για την αποτελεσματική διαχείριση κατηγορικών χαρακτηριστικών χωρίς προεπεξεργασία.
- dash: Προσφέρει ένα ισχυρό πλαίσιο για την ανάπτυξη διαδραστικών εφαρμογών web και dashboards, μετατρέποντας Python κώδικα σε δυναμικές οπτικοποιήσεις δεδομένων.
- dash_bootstrap_components: Συμπληρώνει το Dash με μια εκτεταμένη συλλογή στοιχείων UI βασισμένων στο Bootstrap, επιταχύνοντας τον σχεδιασμό αισθητικά ευχάριστων διεπαφών.
- flask: Ως micro-framework για την ανάπτυξη web εφαρμογών, είναι ιδανικό για την υλοποίηση RESTful APIs και την εξυπηρέτηση backend για μοντέλα μηχανικής μάθησης.
- lightgbm: Πρόκειται για μια εξαιρετικά γρήγορη και αποδοτική βιβλιοθήκη για Gradient Boosting, κατάλληλη για την επεξεργασία μεγάλων συνόλων δεδομένων με υψηλή ακρίβεια.
- numpy: Αποτελεί τη θεμελιώδη βιβλιοθήκη για επιστημονικούς υπολογισμούς, παρέχοντας δομές δεδομένων (arrays) και συναρτήσεις για υψηλής απόδοσης αριθμητικές πράξεις.
- os: Προσφέρει λειτουργίες για την αλληλεπίδραση με το λειτουργικό σύστημα, διευκολύνοντας τη διαχείριση αρχείων, καταλόγων και τη ροή εκτέλεσης διεργασιών.
- pandas: Η καθιερωμένη βιβλιοθήκη για χειρισμό και ανάλυση δομημένων δεδομένων, μέσω των ευέλικτων δομών DataFrame και Series.
- plotly.graph_objects: Επιτρέπει τη δημιουργία προσαρμοσμένων και υψηλής ποιότητας γραφημάτων, προσφέροντας λεπτομερή έλεγχο στην οπτικοποίηση δεδομένων.
- requests: Απλοποιεί την αποστολή αιτημάτων HTTP, καθιστώντας την αλληλεπίδραση με διαδικτυακές πηγές και APIs εύκολη και αποδοτική.
- sklearn.ensemble: Περιλαμβάνει μια συλλογή αλγορίθμων συνόλων (ensemble methods), όπως τα Random Forests, για βελτιωμένη προβλεπτική ικανότητα και στιβαρότητα.
- sklearn.metrics: Παρέχει ένα ολοκληρωμένο σύνολο μετρικών αξιολόγησης για την ποσοτική εκτίμηση της απόδοσης μοντέλων μηχανικής μάθησης.
- sklearn.model_selection: Προσφέρει βασικά εργαλεία για τη διαδικασία επιλογής μοντέλου και επικύρωσης, όπως η διασταυρούμενη επικύρωση, για την αξιόπιστη αξιολόγηση.
- sklearn.neural_network: Περιέχει υλοποιήσεις τεχνητών νευρωνικών δικτύων για ταξινόμηση και παλινδρόμηση, ενσωματωμένες στο πλαίσιο του scikit-learn.
- socket: Επιτρέπει την επικοινωνία μέσω δικτύου με χρήση sockets, παρέχοντας μια χαμηλού επιπέδου διεπαφή για τη δημιουργία εφαρμογών δικτύου.
- sqlite3: Η ενσωματωμένη βιβλιοθήκη για αλληλεπίδραση με βάσεις δεδομένων SQLite, ιδανική για ελαφριά και αυτόνομη αποθήκευση δεδομένων.
- time: Παρέχει λειτουργίες για χειρισμό του χρόνου και χρονικές καθυστερήσεις, απαραίτητες για τον έλεγχο της ροής εκτέλεσης σε προγράμματα.

- xgboost: Μια εξαιρετικά δημοφιλής και αποδοτική βιβλιοθήκη για Extreme Gradient Boosting, ευρέως χρησιμοποιούμενη σε προβλήματα μηχανικής μάθησης λόγω της ταχύτητας και ακρίβειάς της.

3.2.γ Δημιουργία του Dataset

Όπως αναφέραμε και παραπάνω, δημιουργήσαμε το δικό μας σύνολο δεδομένων, με βάση δεδομένα που βρήκαμε από στατιστικά στοιχεία. Έπειτα τα οπτικοποιήσαμε με κώδικες που αναπτύξαμε (βρίσκονται στο Παράρτημα) για να δούμε τις τάσεις και να μπορέσουμε να βελτιώσουμε την διακύμανση, ώστε το τελικό μας σύνολο δεδομένων να είναι όσο το δυνατόν πιο ρεαλιστικό.

Το πρώτο αρχείο λοιπών που δημιουργήσαμε ήταν το *dataset.py*:

```
import pandas as pd
import numpy as np

# Step 1: Create date range
dates = pd.date_range(start="2000-01-01", end="2024-12-31")

# Step 2: Define house IDs and zip codes
house_ids = [f"{i+1}" for i in range(1000)]
zip_codes = [67130, 67131, 67132, 67133]  # Example ZIP codes

# Step 3: Generate dataset with only power consumption
data = []
for house in house_ids:
    zip_code = np.random.choice(zip_codes)  # Assign a random ZIP
code
    for date in dates:
        row = {
            "date": date,
            "house_id": house,
            "zip_code": zip_code,
            "power_consumption": np.random.uniform(1, 20)  # Random
power consumption (kWh)
        }
        data.append(row)

# Step 4: Convert to DataFrame
df = pd.DataFrame(data)

# Step 5: Sort the dataset by 'date' column
df = df.sort_values(by="date")

# Step 6: Save to CSV
df.to_csv("dataset.csv", index=False)
```

Ουσιαστικά αυτό μας δημιουργεί ένα αρχείο . csv, μέσα στο οποίο έχουμε 4 στήλες (*date, house_id, zip_code, power consumption*). Αρχικά ορίζουμε ένα εύρος ημερομηνιών από 01/01/2000 έως 31/12/2024. Το ίδιο κάνουμε και για τα σπίτια και τους ΤΚ. Έπειτα, μέσα σε μια λούπα (επανάληψη) για κάθε ένα από τα 1000 σπίτια ξεκινάμε και αναθέτουμε τυχαία έναν ΤΚ και για κάθε μια ημερομηνία, σε επανάληψη, αναθέτουμε ένα σπίτι, με τον ΤΚ του και προσθέτουμε την κατανάλωση του. Έπειτα μετατρέπουμε το αποτελέσμα αυτης της εξίσωσης σε DataFrame ώστε να μπορούμε να το επεξεργαστούμε. Το ταξινομούμε με βάση την ημερομηνία με αύξοντα αριθμό και τέλος το εξάγουμε. Με αυτόν τον τρόπο αναπτύσσουμε όλους τους κώδικες, η διαφορά βρίσκεται στις τιμές των καταναλώσεων.

Στο πρωταρχικό μας αρχείο βάλαμε τυχαίες τιμές κατανάλωσης από το 1 μέχρι το 20 (kWh). Όπως φαίνεται από την εντολή *"power_consumption": np.random.uniform(1, 20).*

Στην παρούσα περίπτωση δεν χρειάζεται καν να οπτικοποιήσουμε τα δεδομένα για να καταλάβουμε πως κάτι τέτοιο δεν ανταποκρίνεται στην πραγματικότητα.

Επομένως, στην επόμενη μας προσπάθεια, κοιτάξαμε να λάβουμε υπόψιν τις καταναλώσεις ανά εποχή στο αρχείο *dataset_seasonal.py:*

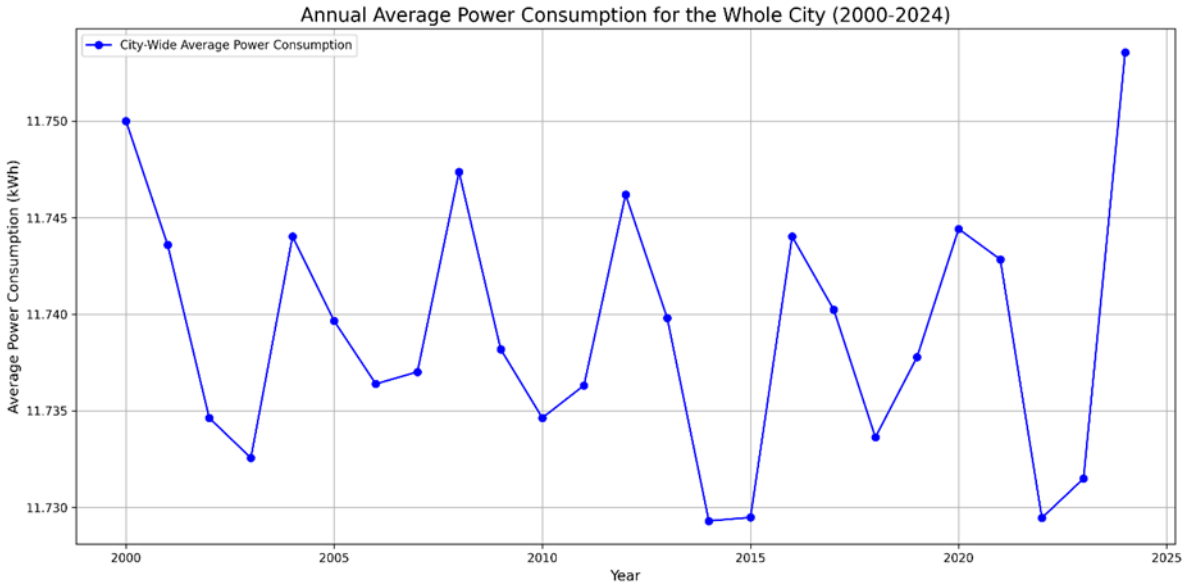
```
import pandas as pd
import numpy as np

# Step 1: Create date range
dates = pd.date_range(start="2000-01-01", end="2024-12-31")

# Step 2: Define house IDs and zip codes
house_ids = [f"{i+1}" for i in range(1000)]
zip_codes = [67130, 67131, 67132, 67133]  # Example ZIP codes

# Seasonal power consumption function based on the month
def seasonal_power_consumption(date):
    month = date.month
    if month in [12, 1, 2]:  # Winter months
        return np.random.uniform(12, 20)  # Higher power consumption
(e.g., heating)
    elif month in [6, 7, 8]:  # Summer months
        return np.random.uniform(10, 18)  # Higher power consumption
(e.g., cooling)
    else:  # Spring and Fall
        return np.random.uniform(5, 12)  # Lower power consumption

# Step 3: Generate dataset with seasonal power consumption
data = []
for house in house_ids:
    zip_code = np.random.choice(zip_codes)  # Assign a random ZIP
code
    for date in dates:
        row = {
            "date": date,
            "house id": house,
            "zip_code": zip_code,
            "power_consumption": seasonal_power_consumption(date)  #
Apply seasonal power consumption
        }
        data.append(row)

# Step 4: Convert to DataFrame
df = pd.DataFrame(data)

# Step 5: Sort the dataset by 'date' column
df = df.sort_values(by="date")

# Step 6: Save to CSV
df.to_csv("dataset_with_seasonal_consumption.csv", index=False)
```

Πλέον κατά την ανάθεση τιμών κατανάλωσης, παίρνουμε υπόψιν ότι οι καταναλώσεις αλλάζουν ανά εποχή. Έτσι, το καλοκαίρι και τον χειμώνα, λόγω ανάγκης για θέρμανση και

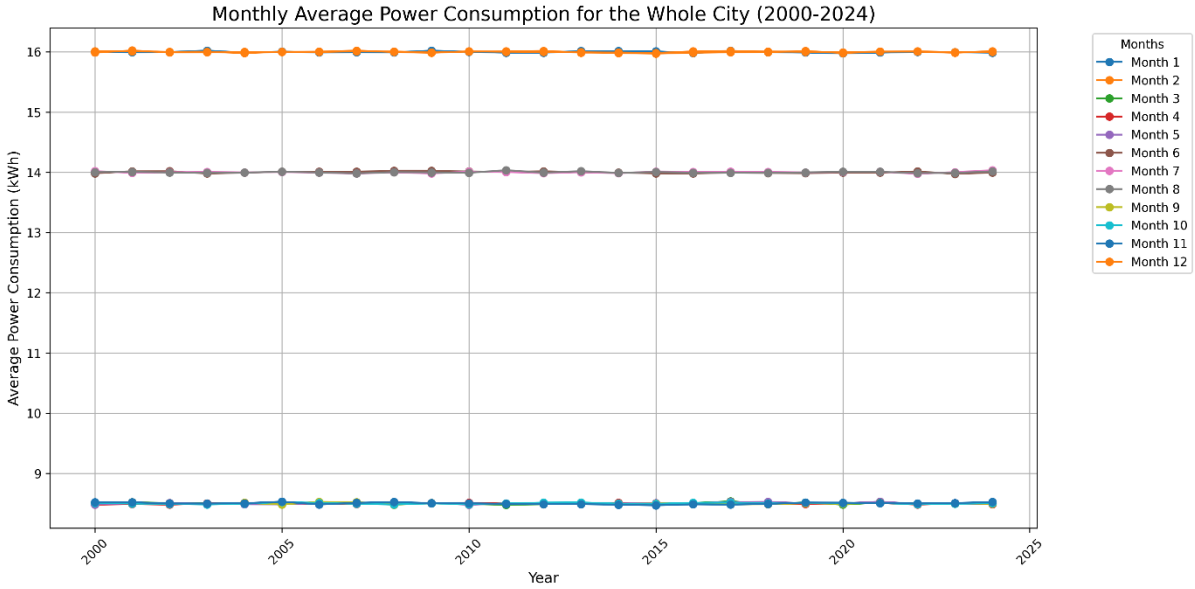
ψύξη, οι καταναλώσεις είναι μεγαλύτερες, επομένως το εύρος τιμών που θα πάρει η κατανάλωση του νοικοκυριού αλλάζει μέσα στον χρόνο.

Τρέχοντας τα παρακάτω παίρνουμε και τα γραφήματα.

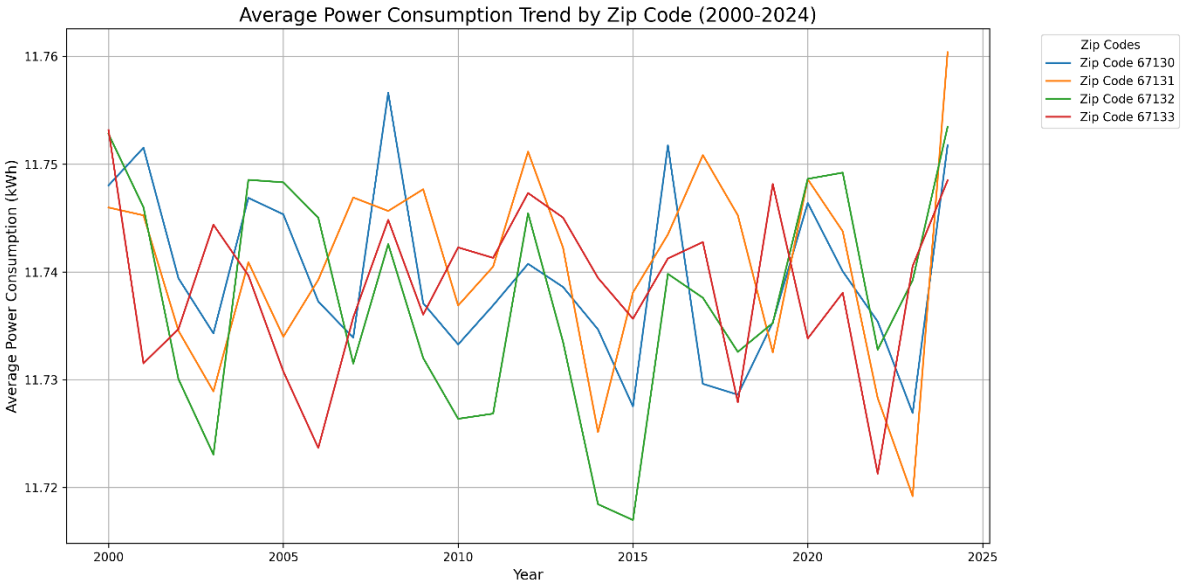
```
cd trends
python trends_viz_city_monthly.py --dataset
datasets/dataset_with_seasonal_consumption.csv
python trends_viz_city_yearly.py --dataset
datasets/dataset_with_seasonal_consumption.csv
python trends_viz_zip_yearly.py --dataset
datasets/dataset_with_seasonal_consumption.csv
```



Εικόνα 10 Μέση Ετήσια Κατανάλωση Ολόκληρης Πόλης (2000 - 2024) για το dataset_with_seasonal_consumption.csv



Εικόνα 11 Μέση Μηνιαία Κατανάλωση Ολόκληρης Πόλης (2000 - 2024) για το dataset_with_seasonal_consumption.csv



Εικόνα 12 Μέση Ετήσια Κατανάλωση Κάθε ΤΚ (2000 - 2024) για το dataset_with_seasonal_consumption.csv

Η *Εικόνα 11*, όπως είπαμε, είναι το πιο σημαντικό γιατί μας δίνει την μέση κατανάλωσης της πόλης ανά μήνα η οποία όπως βλέπουμε είναι σταθερή επομένως το μοντέλο μας θα επιβεβαιώνει κάτι που είναι πολύ σίγουρο.

Στην επόμενη εκδοχή, στην προσπάθεια μας να δώσουμε πιο ρεαλιστικά στοιχεία συμπεριλάβαμε και την πιθανότητα διακοπών. Κατά την οποία ένα νοικοκυριό έχει μικρότερες καταναλώσεις. *dataset_seasonal_vacation.py*:

```
import pandas as pd
import numpy as np

# Step 1: Create date range
dates = pd.date_range(start="2000-01-01", end="2024-12-31")

# Step 2: Define house IDs and zip codes
house_ids = [f"{i+1}" for i in range(1000)]
zip_codes = [67130, 67131, 67132, 67133]  # Example ZIP codes

# Seasonal power consumption function based on the month
def seasonal_power_consumption(date):
    month = date.month
    if month in [12, 1, 2]:  # Winter months
        return np.random.uniform(12, 20)  # Higher power consumption
(e.g., heating)
    elif month in [6, 7, 8]:  # Summer months
        return np.random.uniform(10, 18)  # Higher power consumption
(e.g., cooling)
    else:  # Spring and Fall
        return np.random.uniform(5, 12)  # Lower power consumption

# Function to randomly decrease power consumption for a fixed
vacation period
def random_vacation_reduction(power_consumption, date,
vacation_info, season_vacation_chance):
    """
```

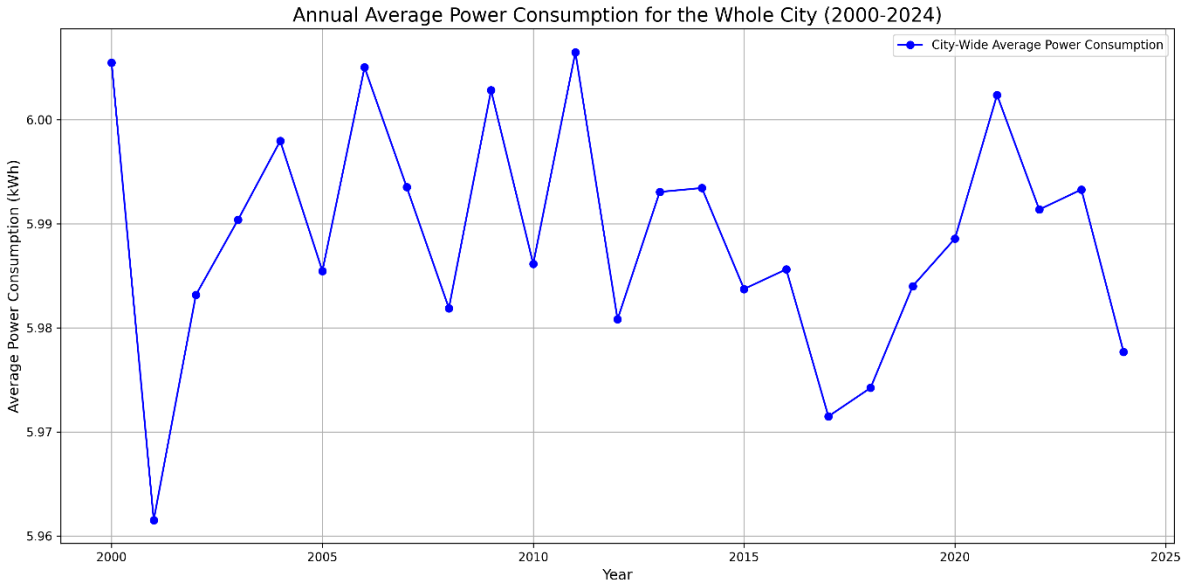
```
    Reduce power consumption if the household is on vacation for
consecutive days.

    - power_consumption: the initial power consumption.
    - date: the current date.
    - vacation_info: dictionary tracking vacation status and
duration.
    - season_vacation_chance: chance of vacation occurring in the
season.

    Returns adjusted power consumption.
    """
    house = vacation_info["house_id"]

    # If currently on vacation, continue reducing power for the
duration
    if vacation_info["on_vacation"] and (date -
vacation_info["start_date"]).days <
vacation_info["vacation_duration"]:
        return power_consumption * np.random.uniform(0.1, 0.4)  #
Reduce by 60-90%

    # If not on vacation, decide whether to start a vacation
    if np.random.rand() < season_vacation_chance:
        vacation_info["on_vacation"] = True
        vacation_info["start_date"] = date
        vacation_info["vacation_duration"] = np.random.randint(3,
15)  # Vacation lasts 3-14 days
        return power_consumption * np.random.uniform(0.1, 0.4)  #
Reduce by 60-90%

    # Not on vacation
    vacation_info["on_vacation"] = False
    return power_consumption


# Step 3: Generate dataset with seasonal power consumption and
controlled vacation durations
data = []
vacation_info = {house: {"house_id": house, "on_vacation": False,
"start_date": None, "vacation_duration": 0} for house in house_ids}

for house in house_ids:
    zip_code = np.random.choice(zip_codes)  # Assign a random ZIP
code
    for date in dates:
        # Get the seasonal power consumption
        power_consumption = seasonal_power_consumption(date)

        # Vacation reduction chance based on season
        if date.month in [12, 1, 2, 6, 7, 8]:  # Winter and Summer
months
            season_vacation_chance = 0.5  # 50% chance for reduced
power usage
        else:  # Other months
            season_vacation_chance = 0.03  # 3% chance for reduced
```

```
power usage

        # Apply random vacation reduction considering vacation
duration
        power_consumption =
random_vacation_reduction(power_consumption, date,
vacation_info[house], season_vacation_chance)

        # Record the row
        row = {
            "date": date,
            "house_id": house,
            "zip_code": zip_code,
            "power_consumption": power_consumption  # Adjusted for
season and vacation
        }
        data.append(row)

# Step 4: Convert to DataFrame
df = pd.DataFrame(data)

# Step 5: Sort the dataset by 'date' column
df = df.sort_values(by="date")

# Step 6: Save to CSV
df.to_csv("dataset_with_seasonal_and_vacation_consumption.csv",
index=False)
```

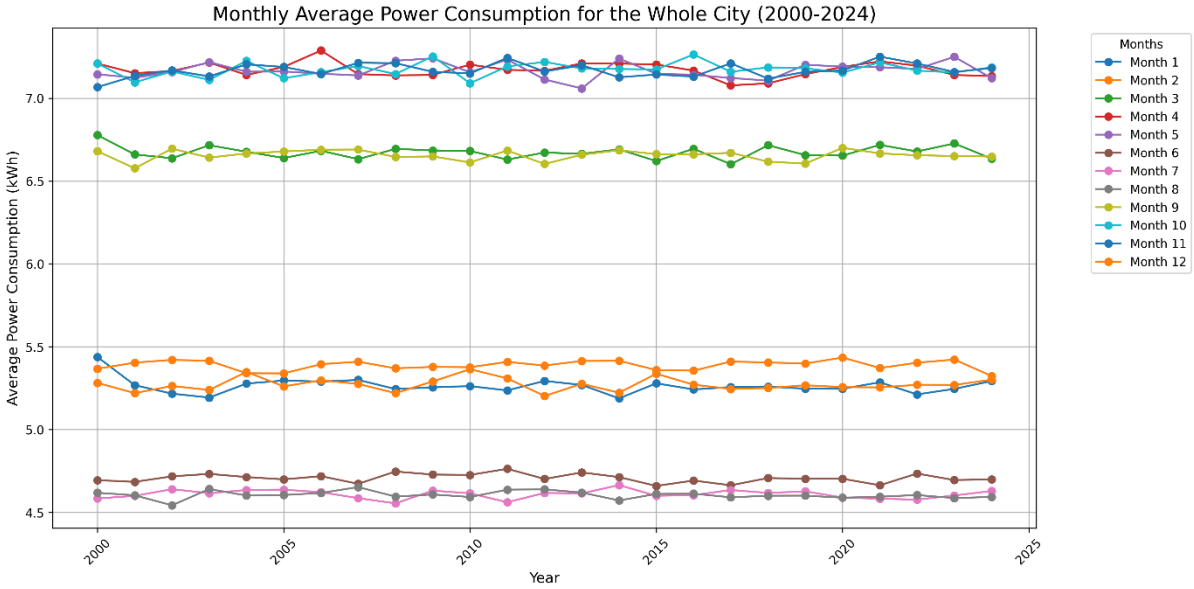
Για να έχουμε πιο σωστά δεδομένα υπολογίσαμε διακοπές. Πιο συγκεκριμένα τους μήνες που ο κόσμος πηγαίνει διακοπές, δηλαδή τους χειμερινούς και καλοκαιρινούς μήνες, να μειώνει την κατανάλωση κατά 60-90%. Η πιθανότητα διακοπών αυτούς τους μήνες είναι 50%, και μπορούν να κρατάνε από 3 ως 14 μέρες. Τους υπόλοιπους μήνες η πιθανότητα να είναι κάποιο νοικοκυριό διακοπές είναι 1%.

```
python trends_viz_city_monthly.py --dataset
../datasets/dataset_with_seasonal_and_vacation_consumption.csv
python trends_viz_city_yearly.py --dataset
../datasets/dataset_with_seasonal_and_vacation_consumption.csv
python trends_viz_zip_yearly.py --dataset
../datasets/dataset_with_seasonal_and_vacation_consumption.csv
```

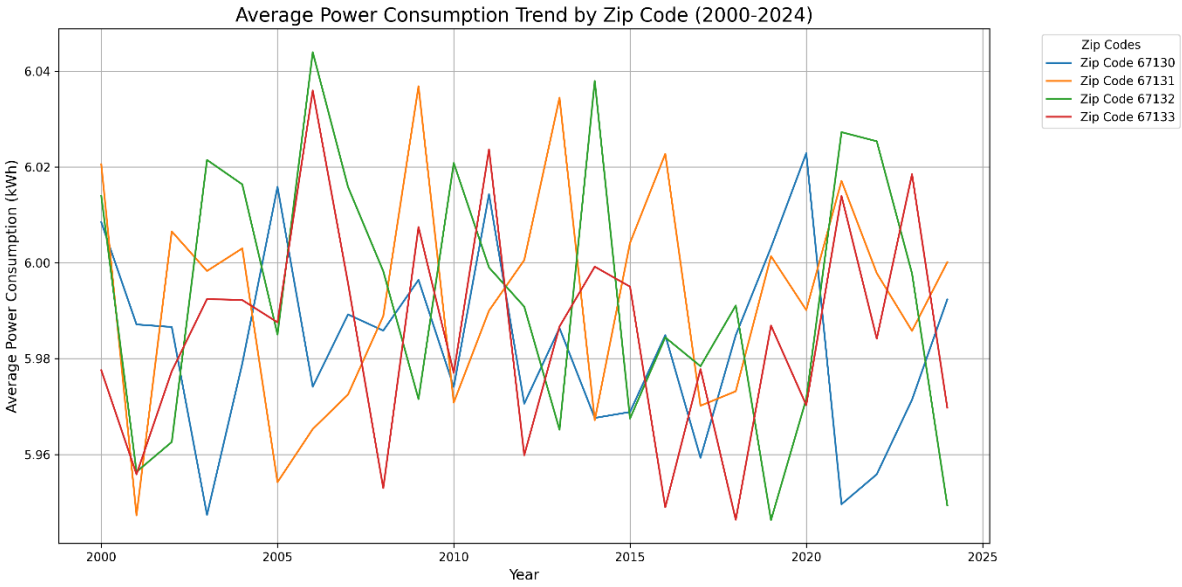
Τώρα τα γραφήματα είναι:



Εικόνα 13 Μέση Ετήσια Κατανάλωση Ολόκληρης Πόλης (2000 - 2024) για το dataset_with_seasonal_and_vacation_consumption.csv



Εικόνα 14 Μέση Μηνιαία Κατανάλωση Ολόκληρης Πόλης (2000 - 2024) για το dataset_with_seasonal_and_vacation_consumption.csv



Εικόνα 15 Μέση Ετήσια Κατανάλωση Κάθε ΤΚ (2000 - 2024) για το dataset_with_seasonal_and_vacation_consumption.csv

Στο προηγούμενο σενάριο, προσθέσαμε και ακατοίκητα σπίτια με πιθανότητα 1% τα οποία καταναλώνουν την ελάχιστη απαιτούμενη ενέργεια, γύρω στα 500W με 2kW. Επιπλέον προσομοιώσαμε διακοπές ρεύματος που κρατάνε από 1 μέχρι 6 ώρες και επηρεάζουν όλο τον ΤΚ. Η πιθανότητα να συμβεί διακοπή ρεύματος είναι 0.2%. *dataset_seasonal_vacations_vacant_outage.py*:

```
import pandas as pd
import numpy as np

# Step 1: Create date range
dates = pd.date_range(start="2000-01-01", end="2024-12-31")

# Step 2: Define house IDs and zip codes
house_ids = [f"{i+1}" for i in range(1000)]
zip_codes = [67130, 67131, 67132, 67133]  # Example ZIP codes

# Randomly select 2% of houses to be vacant (not used)
vacant_houses = set(np.random.choice(house_ids,
size=int(len(house_ids) * 0.02), replace=False))

# Function to generate seasonal power consumption
def seasonal_power_consumption(date):
    month = date.month
    if month in [12, 1, 2]:  # Winter months
        return np.random.uniform(12, 20)  # Higher power consumption
(e.g., heating)
    elif month in [6, 7, 8]:  # Summer months
        return np.random.uniform(10, 18)  # Higher power consumption
(e.g., cooling)
    else:  # Spring and Fall
        return np.random.uniform(5, 12)  # Lower power consumption

# Function to apply vacation reductions
def random_vacation_reduction(power_consumption, date,
vacation_info, season_vacation_chance):
```

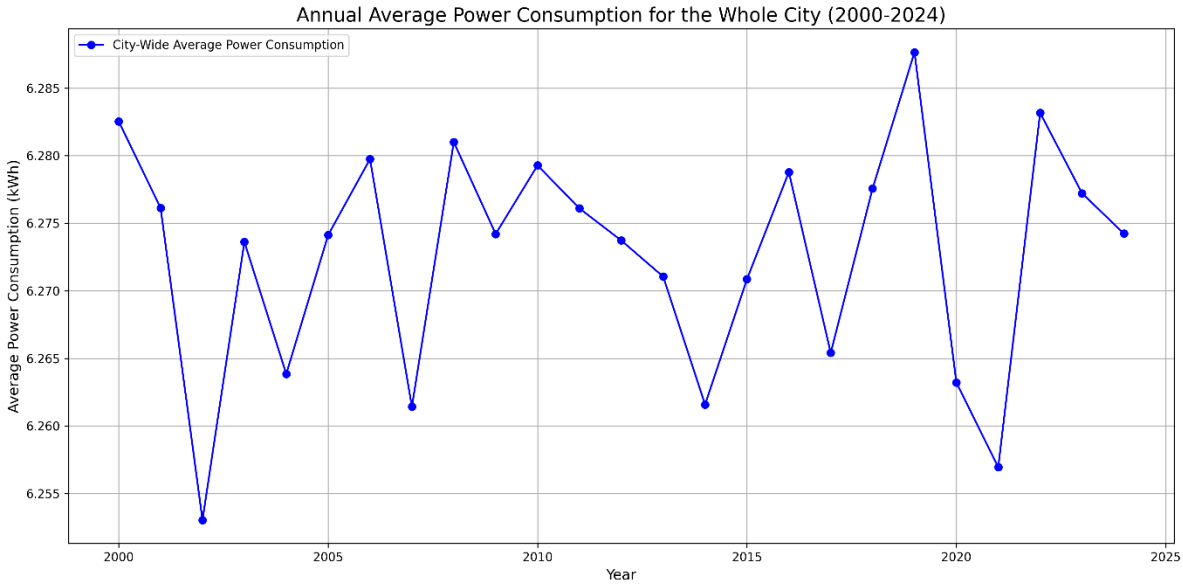
```
    """Applies vacation power reductions, simulating consecutive
days of absence."""
    # If on vacation, continue reducing power for the duration
    if vacation_info["on_vacation"] and (date -
vacation_info["start_date"]).days <
vacation_info["vacation_duration"]:
        return power_consumption * np.random.uniform(0.1, 0.4)  #
Reduce by 60-90%

    # If not on vacation, decide whether to start a vacation
    if np.random.rand() < season_vacation_chance:
        vacation_info["on_vacation"] = True
        vacation_info["start_date"] = date
        vacation_info["vacation_duration"] = np.random.randint(3,
15)  # 3-14 days
        return power_consumption * np.random.uniform(0.1, 0.4)  #
Reduce by 60-90%

    vacation_info["on_vacation"] = False
    return power_consumption

# Function to generate energy breaks (power outages for 1-6 hours)
def generate_energy_breaks(zip_codes, dates, probability=0.002):
    """Generates short power outages (1-6 hours) affecting entire
ZIP codes."""
    outages = {}
    for date in dates:
        if np.random.rand() < probability:  # Small chance of an
outage occurring
            affected_zip = np.random.choice(zip_codes)
            outage_duration = np.random.randint(1, 7)  # Outage
lasts 1-6 hours
            outages[date] = {"zip_code": affected_zip, "hours":
outage_duration}
    return outages

# Step 3: Generate power outages
energy_breaks = generate_energy_breaks(zip_codes, dates)

# Step 4: Generate dataset with seasonal power consumption,
vacations, outages, and vacant houses
data = []
vacation_info = {house: {"house_id": house, "on_vacation": False,
"start_date": None, "vacation_duration": 0} for house in house_ids}

for house in house_ids:
    zip_code = np.random.choice(zip_codes)  # Assign a random ZIP
code
    for date in dates:
        # If the house is vacant, set a very low power consumption
        if house in vacant_houses:
            power_consumption = np.random.uniform(0.5, 2)  # Minimal
standby power
        else:
            # Get the seasonal power consumption
            power_consumption = seasonal_power_consumption(date)
```

```
            # Vacation reduction chance based on season
            if date.month in [12, 1, 2, 6, 7, 8]:  # Winter and
Summer months
                season_vacation_chance = 0.5  # 50% chance per day
            else:
                season_vacation_chance = 0.01  # 1% chance per day

            # Apply vacation reductions
            power_consumption =
random_vacation_reduction(power_consumption, date,
vacation_info[house], season_vacation_chance)

        # Apply energy breaks (power outages for 1-6 hours)
        if date in energy_breaks and zip_code ==
energy_breaks[date]["zip_code"]:
            outage_hours = energy_breaks[date]["hours"]
            power_consumption *= (1 - outage_hours / 24)  # Reduce
power proportionally

        # Record the row
        row = {
            "date": date,
            "house_id": house,
            "zip_code": zip_code,
            "power_consumption": power_consumption  # Adjusted for
season, vacation, outage, and vacancy
        }
        data.append(row)

# Step 5: Convert to DataFrame
df = pd.DataFrame(data)

# Step 6: Sort the dataset by 'date' column
df = df.sort_values(by="date")

# Step 7: Save to CSV
df.to_csv("dataset_with_seasonal_vacation_outages_and_vacant_houses.
csv", index=False)
```

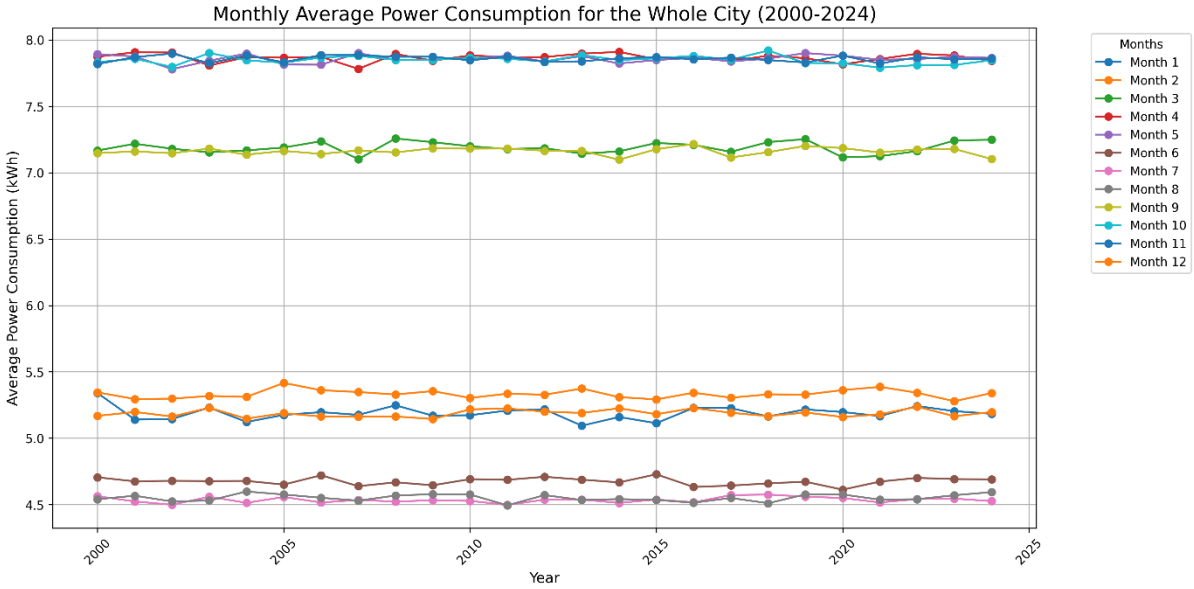
Τρέχουμε το παρακάτω:

```
python trends_viz_city_monthly.py --dataset
../datasets/dataset_with_seasonal_vacation_outages_and_vacant_houses
.csv
python trends_viz_city_yearly.py --dataset
../datasets/dataset_with_seasonal_vacation_outages_and_vacant_houses
.csv
python trends_viz_zip_yearly.py --dataset
../datasets/dataset_with_seasonal_vacation_outages_and_vacant_houses
.csv
```

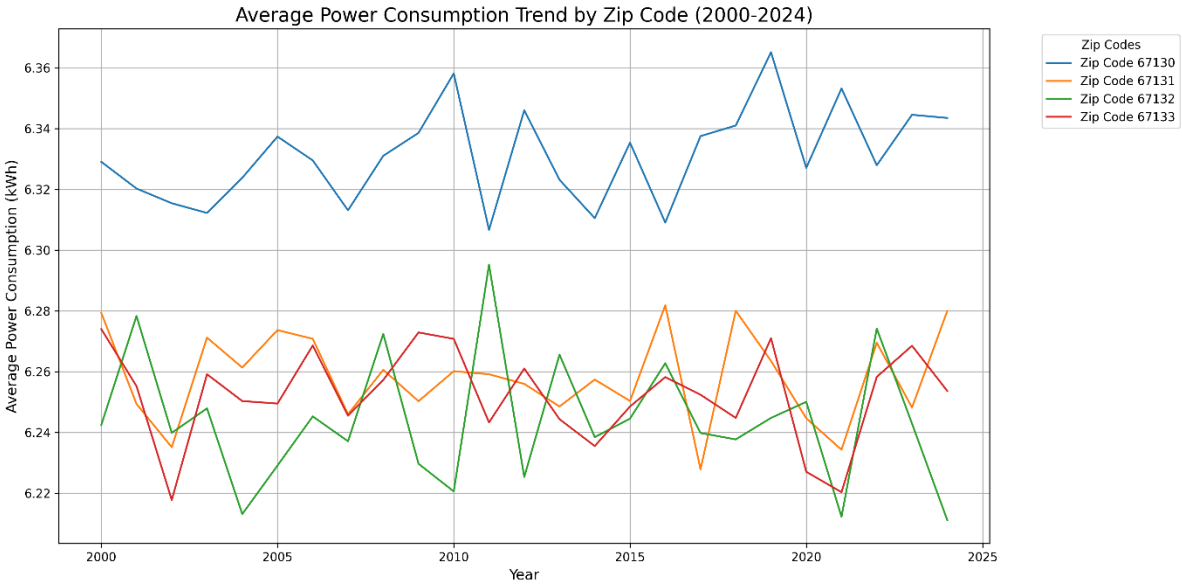
Και παίρνουμε τα γραφήματα:



Εικόνα 16 Μέση Ετήσια Κατανάλωση Ολόκληρης Πόλης (2000 - 2024) για το dataset_with_seasonal_vacation_outages_and_vacant_houses.csv



Εικόνα 17 Μέση Μηνιαία Κατανάλωση Ολόκληρης Πόλης (2000 - 2024) για το dataset_with_seasonal_vacation_outages_and_vacant_houses.csv



Εικόνα 18 Μέση Ετήσια Κατανάλωση Κάθε ΤΚ (2000 - 2024) για το dataset_with_seasonal_vacation_outages_and_vacant_houses.csv

Βλέπουμε ότι η διακύμανση στα γραφήματα μας, και ειδικότερα στο γράφημα *«Μέση Μηνιαία Κατανάλωση Ολόκληρης Πόλης (2000 - 2024)»* αυξάνεται. Θα θεωρήσουμε ότι σε αυτό το σημείο το αποτέλεσμα είναι ικανοποιητικό.

3.2.γ Ασφάλεια Δεδομένων

Και αφού καταλήξαμε σε ένα τρόπο δημιουργίας δεδομένων, θα τον αναπαράγουμε ξανά, απλά αυτή την φορά θα εισάγουμε την μεταβλητή της ασφάλειας. Οι καταναλώσεις εισάγονται σε μία βάση δεδομένων, οργανωμένη σε τέσσερις διακριτούς πίνακες. Κάθε πίνακας αντιστοιχεί σε έναν συγκεκριμένο ταχυδρομικό κώδικα, αντιπροσωπεύοντας ουσιαστικά ένα διαφορετικό γεωγραφικό τμήμα του δικτύου.

Η δομή αυτή επιτρέπει σε κάθε edge node του συνεργατικού σχήματος να επιλέξει και να επεξεργαστεί δεδομένα από έναν συγκεκριμένο πίνακα/ ταχυδρομικό κώδικα, προσομοιώνοντας έτσι ένα κατανεμημένο σενάριο όπου κάθε κόμβος είναι υπεύθυνος για τη διαχείριση δεδομένων από την τοπική του περιοχή. Αυτό όπως αναφέραμε και στο Κεφάλαιο 2 στοχεύει στην αποκέντρωση των δεδομένων.

Μπορεί να φαίνεται παράδοξο, ότι μιλάμε για αποκέντρωση δεδομένων, ενώ αυτά βρίσκονται σε μια βάση δεδομένων, αλλά αυτό είναι απαραίτητο λόγω της φύσης του συστήματος μας. Στην πραγματικότητα, ο κάθε ένας πίνακας θα βρισκόταν αποθηκευμένος στην μνήμη του καθενός edge node. Επειδή όμως η προσομοίωσή μας πραγματοποιείται σε έναν υπολογιστή, εκεί που κάθε node θα ήταν ένας υπολογιστής, ο τρόπος με τον οποίο επιτυγχάνουμε την αποκέντρωση είναι χωρίζοντας το dataset.

Όλα αυτά επιτυγχάνονται με τον παρακάτω κώδικα:

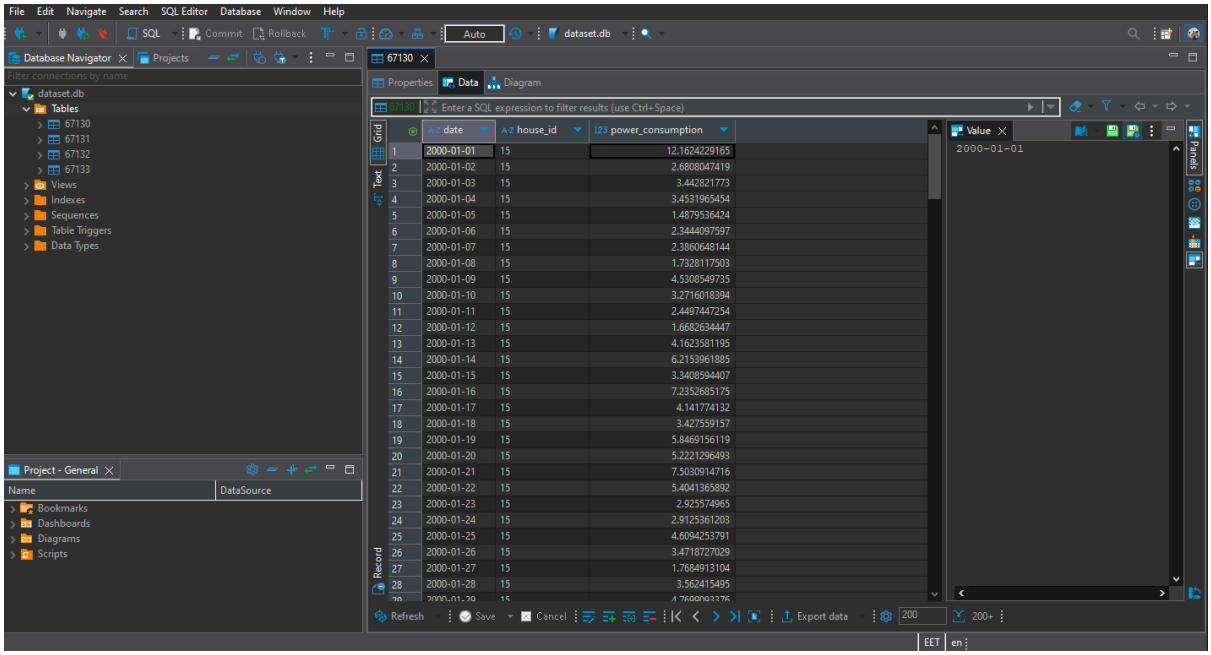
```
import sqlite3
import pandas as pd
import numpy as np

# Create date range
dates = pd.date_range(start="2000-01-01", end="2024-12-31")

# Define house IDs and zip codes
house_ids = [f"{i+1}" for i in range(1000)]
zip_codes = [67130, 67131, 67132, 67133]  # Example ZIP codes
```

```
# Randomly select 2% of houses to be vacant (not used)
vacant_houses = set(np.random.choice(house_ids,
size=int(len(house_ids) * 0.02), replace=False))

# Function to generate seasonal power consumption
def seasonal_power_consumption(date):
    month = date.month
    if month in [12, 1, 2]:  # Winter months
        return np.random.uniform(12, 20)  # Higher power consumption
(e.g., heating)
    elif month in [6, 7, 8]:  # Summer months
        return np.random.uniform(10, 18)  # Higher power consumption
(e.g., cooling)
    else:  # Spring and Fall
        return np.random.uniform(5, 12)  # Lower power consumption

# Function to apply vacation reductions
def random_vacation_reduction(power_consumption, date,
vacation_info, season_vacation_chance):
    """Applies vacation power reductions, simulating consecutive
days of absence."""
    # If on vacation, continue reducing power for the duration
    if vacation_info["on_vacation"] and (date -
vacation_info["start_date"]).days <
vacation_info["vacation_duration"]:
        return power_consumption * np.random.uniform(0.1, 0.4)  #
Reduce by 60-90%

    # If not on vacation, decide whether to start a vacation
    if np.random.rand() < season_vacation_chance:
        vacation_info["on_vacation"] = True
        vacation_info["start_date"] = date
        vacation_info["vacation_duration"] = np.random.randint(3,
15)  # 3-14 days
        return power_consumption * np.random.uniform(0.1, 0.4)  #
Reduce by 60-90%

    vacation_info["on_vacation"] = False
    return power_consumption

# Function to generate energy breaks (power outages for 1-6 hours)
def generate_energy_breaks(zip_codes, dates, probability=0.002):
    """Generates short power outages (1-6 hours) affecting entire
ZIP codes."""
    outages = {}
    for date in dates:
        if np.random.rand() < probability:  # Small chance of an
outage occurring
            affected_zip = np.random.choice(zip_codes)
            outage_duration = np.random.randint(1, 7)  # Outage
lasts 1-6 hours
            outages[date] = {"zip_code": affected_zip, "hours":
outage_duration}
    return outages
```

```
# Generate power outages
energy_breaks = generate_energy_breaks(zip_codes, dates)

# Generate dataset with seasonal power consumption, vacations,
outages, and vacant houses
vacation_info = {house: {"house_id": house, "on_vacation": False,
"start_date": None, "vacation_duration": 0} for house in house_ids}

# Connect to SQLite database
conn = sqlite3.connect('dataset.db')
cursor = conn.cursor()

# Loop over each ZIP code and create a table for each ZIP code
for zip_code in zip_codes:
    cursor.execute(f'''
        CREATE TABLE IF NOT EXISTS "{zip_code}" (
            date TEXT,
            house_id TEXT,
            power_consumption REAL
        )
    ''')

for house in house_ids:
    house_zip_code = np.random.choice(zip_codes)

    for date in dates:
        if house in vacant_houses:
            power_consumption = np.random.uniform(0.5, 2)  # Minimal
standby power
        else:
            # Get the seasonal power consumption
            power_consumption = seasonal_power_consumption(date)

            # Vacation reduction chance based on season
            if date.month in [12, 1, 2, 6, 7, 8]:  # Winter and
Summer months
                season_vacation_chance = 0.5  # 50% chance per day
            else:
                season_vacation_chance = 0.01  # 1% chance per day

            # Apply vacation reductions
            power_consumption =
random_vacation_reduction(power_consumption, date,
vacation_info[house], season_vacation_chance)

        # Apply energy breaks (power outages for 1-6 hours)
        if date in energy_breaks and house_zip_code ==
energy_breaks[date]["zip_code"]:
            outage_hours = energy_breaks[date]["hours"]
            power_consumption *= (1 - outage_hours / 24)  # Reduce
power proportionally

        cursor.execute(f'''
            INSERT INTO "{house_zip_code}" (date, house_id,
power_consumption)
            VALUES (?, ?, ?)
```

```
        ''', (date.strftime('%Y-%m-%d'), house, power_consumption))

# Commit the changes and close the connection
conn.commit()
conn.close()

print("Data successfully saved to the SQLite database
'dataset.db'.")
```

Με την εισαγωγή του dataset.db σε ένα πρόγραμμα περιήγησης βάσης δεδομένων, μπορούμε να δούμε την δομή της.



Εικόνα 19 Το περιβάλλον του DBeaver και ο πίνακας 67130

3.2.δ Edge Nodes

Η παρούσα ενότητα καθώς και οι δύο παρακάτω περιλαμβάνουν μόνο την τελευταία εκδοχή της υλοποίησης μας. Οι προηγούμενες εκδόσεις, καθώς και ο σχολιασμός τους βρίσκονται στο Παράρτημα.

Κάθε κόμβος, που αντιστοιχεί σε έναν συγκεκριμένο ταχυδρομικό κώδικα, ανακτά ιστορικά δεδομένα από μια τοπική βάση SQLite και τα προεπεξεργάζεται, δημιουργώντας χαρακτηριστικά όπως η κατανάλωση προηγούμενων ημερών και η εποχή. Για την πρόβλεψη, χρησιμοποιεί μια μέθοδο stacking ensemble, όπου εκπαιδεύει πολλαπλά μοντέλα βάσης (όπως XGBoost, LightGBM, Random Forest, CatBoost και ένα νευρωνικό δίκτυο) και ένα μετα-μοντέλο που συνδυάζει τις προβλέψεις τους για να βελτιώσει την ακρίβεια.

Ο κόμβος λειτουργεί σε συνεχή προσομοίωση, προβλέποντας την κατανάλωση ενέργειας για κάθε ημέρα και στέλνοντας αυτές τις προβλέψεις, μαζί με πραγματικά δεδομένα και μετρικές απόδοσης (MAE, RMSE, R^2), σε έναν κεντρικό κόμβο (master node) μέσω HTTP αιτημάτων. Αυτή η επικοινωνία επιτρέπει στον κεντρικό κόμβο να παρακολουθεί την απόδοση και να ελέγχει τον ρυθμό της προσομοίωσης, ενώ ο ακραίος κόμβος μπορεί να λάβει σήματα επαναφοράς για επανεκκίνηση.

```
import sqlite3
import requests
import pandas as pd
import numpy as np
import xgboost as xgb
import lightgbm as lgb
from catboost import CatBoostRegressor
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
import time
import argparse
import socket
import os

historical_predictions = []
historical_actuals = []

parser = argparse.ArgumentParser(description="Edge Node for Power
Consumption Forecasting")
parser.add_argument("--zip", type=int, required=True, help="ZIP code
for this edge node")
args = parser.parse_args()
ZIP_CODE = args.zip

NODE_ID = f"{socket.gethostname()}_{os.getpid()}"

# Connect to the database
conn = sqlite3.connect('datasets/dataset.db')
cursor = conn.cursor()

cursor.execute('SELECT * FROM "{}"'.format(ZIP_CODE))
data = cursor.fetchall()

# Convert to DataFrame for easier processing
df = pd.DataFrame(data, columns=["date", "house_id",
"power_consumption"])
df['date'] = pd.to_datetime(df['date'])

# Lag Features (1-day, 7-day, and 30-day power consumption)
df['prev_day_power'] = df['power_consumption'].shift(1)
df['prev_7_day_power'] = df['power_consumption'].shift(7)
df['prev_30_day_power'] = df['power_consumption'].shift(30)

# Season Variable
def get_season(month):
    if month in [12, 1, 2]: return 0  # Winter
    elif month in [3, 4, 5]: return 1  # Spring
    elif month in [6, 7, 8]: return 2  # Summer
    else: return 3  # Fall

df['season'] = df['date'].dt.month.apply(get_season)

# Prepare training data
```

```
train_data = df[df['date'].dt.year < 2024].dropna()  # Drop NaN
caused by shifting
X_train = train_data[['date']].copy()
X_train['year'] = X_train['date'].dt.year
X_train['month'] = X_train['date'].dt.month
X_train['day'] = X_train['date'].dt.day
X_train['prev_day_power'] = train_data['prev_day_power']
X_train['prev_7_day_power'] = train_data['prev_7_day_power']
X_train['prev_30_day_power'] = train_data['prev_30_day_power']
X_train['season'] = train_data['season']
y_train = train_data['power_consumption']

#Base models Parameters (XGBoost, LightGBM, RandomForest, CatBoost,
and MLP)
xgb_model = xgb.XGBRegressor(
    objective='reg:squarederror',
    tree_method='hist',
    device='cuda',
    learning_rate=0.09979359710905994,
    max_depth=8,
    min_child_weight=9,
    subsample=0.7225909453820122,
    colsample_bytree=0.9568932771634039,
    gamma=0.4520419990324307,
    reg_lambda=2.1150201222243434,
    reg_alpha=0.5763612495790174
)
xgb_model.fit(X_train.drop(columns=['date']), y_train)

lgb_model = lgb.LGBMRegressor(
    n_estimators=500,
    learning_rate=0.1,
    max_depth=8
)
lgb_model.fit(X_train.drop(columns=['date']), y_train)

rf_model = RandomForestRegressor(n_estimators=500, max_depth=8)
rf_model.fit(X_train.drop(columns=['date']), y_train)

gbr_model = GradientBoostingRegressor(n_estimators=500,
learning_rate=0.1, max_depth=8)
gbr_model.fit(X_train.drop(columns=['date']), y_train)

cat_model = CatBoostRegressor(iterations=500, depth=6,
learning_rate=0.1, verbose=0)
cat_model.fit(X_train.drop(columns=['date']), y_train)

mlp_model = MLPRegressor(hidden_layer_sizes=(128, 64), max_iter=500,
random_state=42)
mlp_model.fit(X_train.drop(columns=['date']), y_train)

# Meta model
meta_model = xgb.XGBRegressor(objective='reg:squarederror',
learning_rate=0.05, max_depth=6, n_estimators=300)

# Base model predictions on training set
```

```
xgb_preds = xgb_model.predict(X_train.drop(columns=['date']))
lgb_preds = lgb_model.predict(X_train.drop(columns=['date']))
rf_preds = rf_model.predict(X_train.drop(columns=['date']))
gbr_preds = gbr_model.predict(X_train.drop(columns=['date']))
cat_preds = cat_model.predict(X_train.drop(columns=['date']))
mlp_preds = mlp_model.predict(X_train.drop(columns=['date']))

# Stack base model predictions as features for meta-model
X_meta = np.column_stack((xgb_preds, lgb_preds, rf_preds, gbr_preds,
cat_preds, mlp_preds))

# Train meta model on stacked predictions
meta_model.fit(X_meta, y_train)

# Function to send real data and metrics to master node
def send_real_data_and_metrics(sim_date, prediction):
    """Send real data and ML metrics to master node."""
    actual_value = df[df['date'] ==
sim_date]['power_consumption'].values
    actual_value = float(actual_value[0]) if len(actual_value) > 0
else None

    if actual_value is not None:
        historical_predictions.append(prediction)
        historical_actuals.append(actual_value)

        requests.post("http://127.0.0.1:5001/submit_real_data",
json={
            "date": sim_date.strftime('%Y-%m-%d'),
            "zip_code": ZIP_CODE,
            "actual": actual_value
        })

        if len(historical_predictions) > 10:
            mae = mean_absolute_error(historical_actuals,
historical_predictions)
            rmse = np.sqrt(mean_squared_error(historical_actuals,
historical_predictions))
            r2 = r2_score(historical_actuals,
historical_predictions)

            requests.post("http://127.0.0.1:5001/submit_ml_metrics",
json={
                "zip_code": ZIP_CODE,
                "mae": mae,
                "rmse": rmse,
                "r2": r2
            })

# Fetch simulation speed
def fetch_speed():
    try:
        response =
requests.get("http://127.0.0.1:5001/get_speed").json()
        return float(response.get("speed", 1.0))
    except requests.exceptions.RequestException:
```

```
        return 1.0

def run_simulation():
    sim_date = pd.Timestamp("2000-01-01")
    last_reset_status = False

    while True:
        status = requests.get("http://127.0.0.1:5001/status").json()
        if not status.get("running", False):
            time.sleep(1)
            continue

        reset_response =
requests.post("http://127.0.0.1:5001/get_reset_signal",
json={"node_id": NODE_ID}).json()
        if reset_response.get("reset", False):
            print(f"Node {NODE_ID} detected reset! Restarting
simulation.")
            sim_date = pd.Timestamp("2000-01-01")
            last_reset_status = True
            continue

        speed_factor = fetch_speed()

        # Prepare the input data for prediction
        X_pred = pd.DataFrame([[sim_date.year, sim_date.month,
sim_date.day]], columns=['year', 'month', 'day'])

        # Get lag features for the prediction
        prev_day_power = df[df['date'] == sim_date -
pd.Timedelta(days=1)]['power_consumption'].values
        prev_7_day_power = df[df['date'] == sim_date -
pd.Timedelta(days=7)]['power_consumption'].values
        prev_30_day_power = df[df['date'] == sim_date -
pd.Timedelta(days=30)]['power_consumption'].values
        prev_day_power = float(prev_day_power[0]) if
len(prev_day_power) > 0 else 0
        prev_7_day_power = float(prev_7_day_power[0]) if
len(prev_7_day_power) > 0 else 0
        prev_30_day_power = float(prev_30_day_power[0]) if
len(prev_30_day_power) > 0 else 0

        # Get season for current prediction date
        season = get_season(sim_date.month)

        # Create final prediction input
        X_pred = pd.DataFrame([[sim_date.year, sim_date.month,
sim_date.day, prev_day_power, prev_7_day_power, prev_30_day_power,
season]],
                              columns=['year', 'month', 'day',
'prev_day_power', 'prev_7_day_power', 'prev_30_day_power',
'season'])

        # Get base model predictions
        xgb_pred = xgb_model.predict(X_pred)[0]
        lgb_pred = lgb_model.predict(X_pred)[0]
```

```
        rf_pred = rf_model.predict(X_pred)[0]
        gbr_pred = gbr_model.predict(X_pred)[0]
        cat_pred = cat_model.predict(X_pred)[0]
        mlp_pred = mlp_model.predict(X_pred)[0]

        # Stack base model predictions
        X_meta_pred = np.column_stack((xgb_pred, lgb_pred, rf_pred,
gbr_pred, cat_pred, mlp_pred))

        # Get final prediction from meta-model
        final_prediction = meta_model.predict(X_meta_pred)[0]

        send_real_data_and_metrics(sim_date, final_prediction)

        response =
requests.post("http://127.0.0.1:5001/submit_prediction", json={
            "date": sim_date.strftime('%Y-%m-%d'),
            "zip_code": ZIP_CODE,
            "prediction": float(final_prediction)
        })

        print(f"Node {NODE_ID} submitted prediction for
{sim_date.strftime('%Y-%m-%d')} at speed {speed_factor}, response:
{response.status_code}")

        sim_date += pd.Timedelta(days=1)

        sleep_time = max(0.01, 1.0 / speed_factor)
        time.sleep(sleep_time)

run_simulation()
```

3.2.ε Master Node

Ο κεντρικός κόμβος χρησιμοποιεί το Flask για τη δημιουργία ενός RESTful API. Ο κύριος ρόλος του είναι να λειτουργεί ως κεντρικός ελεγκτής και συλλέκτης δεδομένων για τους τοπικούς κόμβους. Παρέχει endpoints για την υποβολή προβλέψεων, πραγματικών δεδομένων κατανάλωσης και μετρικών αξιολόγησης (MAE, RMSE, R^2) από τους ακραίους κόμβους, διατηρώντας αυτές τις πληροφορίες σε καθολικές μεταβλητές.

Πέρα από τη συλλογή δεδομένων, ο κεντρικός κόμβος διαχειρίζεται την κατάσταση της προσομοίωσης, επιτρέποντας την έναρξη, παύση ή επαναφορά της, καθώς και τον καθορισμό της ταχύτητας προσομοίωσης. Διαθέτει επίσης μηχανισμό για την αποστολή σήματος επαναφοράς στους τοπικούς κόμβους και την επιβεβαίωση ότι όλοι οι αναμενόμενοι κόμβοι έχουν λάβει το σήμα.

```
from flask import Flask, request, jsonify

app = Flask(__name__)

global_predictions = []
real_data = []
ml_metrics = {}
simulation_date = None
simulation_running = False
```

```
reset_signal = False
nodes_detected_reset = set()
speed_factor = 1
EXPECTED_NODES = 4

@app.route('/submit_prediction', methods=['POST'])
def submit_prediction():
    global simulation_date
    data = request.json
    simulation_date = data["date"]
    global_predictions.append(data)
    return jsonify({"status": "success"}), 200

@app.route('/get_predictions', methods=['GET'])
def get_predictions():
    return jsonify(global_predictions), 200

@app.route('/start', methods=['POST'])
def start_simulation():
    global simulation_running
    simulation_running = True
    return jsonify({"status": "started", "running": True}), 200

@app.route('/pause', methods=['POST'])
def pause_simulation():
    global simulation_running
    simulation_running = False
    return jsonify({"status": "paused", "running": False}), 200

@app.route('/reset', methods=['POST'])
def reset_simulation():
    global global_predictions, simulation_date, simulation_running,
reset_signal, nodes_detected_reset, real_data, ml_metrics
    global_predictions = []  # Reset predictions
    real_data = []  # Reset actual power consumption data
    ml_metrics = {}  # Reset ML metrics
    simulation_date = None
    simulation_running = False
    reset_signal = True
    nodes_detected_reset.clear()
    return jsonify({"status": "reset"}), 200

@app.route('/get_reset_signal', methods=['POST'])
def get_reset_signal():
    global reset_signal, nodes_detected_reset
    data = request.json
    node_id = data.get("node_id")

    if reset_signal and node_id not in nodes_detected_reset:
        nodes_detected_reset.add(node_id)
        if len(nodes_detected_reset) >= EXPECTED_NODES:
            reset_signal = False
        return jsonify({"reset": True})

    return jsonify({"reset": False})
```

```
@app.route('/status', methods=['GET'])
def get_status():
    return jsonify({"running": simulation_running}), 200

@app.route('/get_speed', methods=['GET'])
def get_speed():
    """Returns the current simulation speed."""
    return jsonify({"speed": speed_factor}), 200

@app.route('/set_speed', methods=['POST'])
def set_speed():
    global speed_factor
    data = request.json
    speed_factor = data.get("speed", 1)
    return jsonify({"status": "speed updated", "speed":
speed_factor}), 200

@app.route('/submit_real_data', methods=['POST'])
def submit_real_data():
    """Stores actual power consumption data for comparison."""
    global real_data
    data = request.json
    real_data.append(data)
    return jsonify({"status": "success"}), 200

@app.route('/get_real_data', methods=['GET'])
def get_real_data():
    """Returns the actual recorded power consumption."""
    return jsonify(real_data), 200

@app.route('/submit_ml_metrics', methods=['POST'])
def submit_ml_metrics():
    """Receives ML evaluation metrics from edge nodes."""
    global ml_metrics
    data = request.json
    zip_code = data.get("zip_code")
    if zip_code:
        ml_metrics[zip_code] = data
    return jsonify({"status": "success"}), 200

@app.route('/get_ml_metrics', methods=['GET'])
def get_ml_metrics():
    """Returns the latest ML metrics."""
    return jsonify(ml_metrics), 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5001, debug=True)
```

3.2.στ Dashboard

Ο πίνακας ελέγχου λειτουργεί σε πραγματικό χρόνο, χρησιμοποιώντας το Dash και οπτικοποιεί τα αποτελέσματα με την χρήση του Plotly. Πρόκειται για την κεντρική διεπαφή του χρήστη, επικοινωνώντας με τον κεντρικό κόμβο (master node) μέσω HTTP αιτημάτων. Ο πίνακας επιτρέπει στους χρήστες να ελέγχουν την προσομοίωση (έναρξη, παύση,

επαναφορά), να ρυθμίζουν την ταχύτητα της, καθώς εμφανίζει και την τρέχουσα ημερομηνία προσομοίωσης.

Επίσης παρέχει ζωντανή απεικόνιση των προβλέψεων κατανάλωσης ενέργειας που υποβάλλονται από τους τοπικούς κόμβους, καθώς και των αντίστοιχων πραγματικών δεδομένων, με διακριτά γραφήματα για κάθε ταχυδρομικό κώδικα. Επιπλέον, εμφανίζει ενημερωμένες μετρικές αξιολόγησης του μοντέλου (όπως MAE, RMSE και R^2).

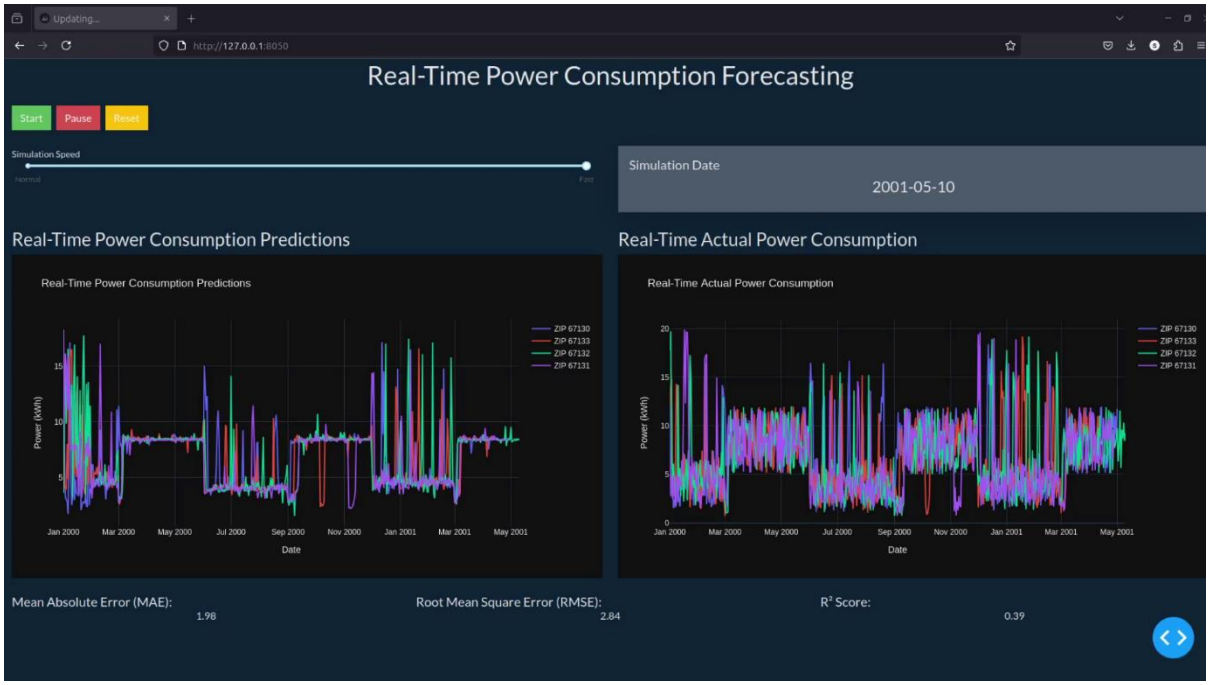
```
import dash
import dash_bootstrap_components as dbc
from dash import dcc, html
from dash.dependencies import Input, Output
import plotly.graph_objects as go
import requests
import pandas as pd

app = dash.Dash(__name__,
external_stylesheets=[dbc.themes.SUPERHERO])

app.layout = dbc.Container([
    dbc.Row([
        dbc.Col(html.H1("Real-Time Power Consumption Forecasting",
className="text-center mb-4"), width=12)
    ]),

    # Control Buttons
    dbc.Row([
        dbc.Col([
            dbc.Button("Start", id="start-button", color="success",
className="me-2"),
            dbc.Button("Pause", id="pause-button", color="danger",
className="me-2"),
            dbc.Button("Reset", id="reset-button", color="warning"),
        ], width=6)
    ], class_name="mb-4"),

    # Speed Control & Simulation Date
    dbc.Row([
        dbc.Col([
            html.Label("Simulation Speed"),
            dcc.Slider(id='speed-slider', min=1, max=100, step=1,
value=1,
                       marks={1: 'Normal', 100: 'Fast'})
        ], width=6),
        dbc.Col([
            dbc.Card([
                dbc.CardBody([
                    html.H5("Simulation Date", className="card-
title"),
                    html.H4(id="simulation-date", className="text-
center")
                ])
            ], class_name="shadow-lg")
        ], width=6)
    ], class_name="mb-4"),
```

```
    # Live Graphs for Predictions and Real Data
    dbc.Row([
        dbc.Col([
            html.H3("Real-Time Power Consumption Predictions"),
            dcc.Graph(id='live-prediction-graph', style={"height":
"500px"})
        ], width=6),
        dbc.Col([
            html.H3("Real-Time Actual Power Consumption"),
            dcc.Graph(id='live-real-data-graph', style={"height":
"500px"})
        ], width=6)
    ], class_name="mb-4"),

    # Metrics Section (Added for MAE, RMSE, R2 Scores)
    dbc.Row([
        dbc.Col([
            html.H5("Mean Absolute Error (MAE):", className="card-
title"),
            html.H6(id='mae-score', children="N/A", className="text-
center")
        ], width=4),
        dbc.Col([
            html.H5("Root Mean Square Error (RMSE):",
className="card-title"),
            html.H6(id='rmse-score', children="N/A",
className="text-center")
        ], width=4),
        dbc.Col([
            html.H5("R² Score:", className="card-title"),
            html.H6(id='r2-score', children="N/A", className="text-
center")
        ], width=4),
    ], class_name="mb-4"),

    # Interval Component for Real-Time Updates
    dcc.Interval(id='interval-component', interval=1000,
n_intervals=0, disabled=True)
], fluid=True)

# Fetch data from master node
def fetch_predictions():
    try:
        response =
requests.get("http://127.0.0.1:5001/get_predictions")
        if response.status_code == 200:
            return response.json()
    except requests.exceptions.RequestException:
        return []
    return []

def fetch_real_data():
    try:
        response =
requests.get("http://127.0.0.1:5001/get_real_data")
        if response.status_code == 200:
```

```
            return response.json()
    except requests.exceptions.RequestException:
        return []
    return []

def fetch_ml_metrics():
    try:
        response =
requests.get("http://127.0.0.1:5001/get_ml_metrics")
        if response.status_code == 200:
            return response.json()
    except requests.exceptions.RequestException:
        return {}

# Fetch simulation speed
def fetch_speed():
    try:
        response = requests.get("http://127.0.0.1:5001/get_speed")
        if response.status_code == 200:
            return response.json().get("speed", 1)
    except requests.exceptions.RequestException:
        return 1

# Update the graph with new data
@app.callback(
    [Output('live-prediction-graph', 'figure'),
     Output('live-real-data-graph', 'figure'),
     Output('simulation-date', 'children')],
    Input('interval-component', 'n_intervals')
)
def update_graph(n):
    predictions = fetch_predictions()
    real_data = fetch_real_data()

    if not predictions or not real_data:
        return go.Figure(), go.Figure(), "No Data"

    # Prepare predictions dataframe
    df_pred = pd.DataFrame(predictions)
    df_pred['date'] = pd.to_datetime(df_pred['date'])
    latest_date = df_pred["date"].max().strftime('%Y-%m-%d')

    # Prepare real data dataframe
    df_real = pd.DataFrame(real_data)
    df_real['date'] = pd.to_datetime(df_real['date'])

    # Prediction figure
    fig_pred = go.Figure()
    for zip_code in df_pred['zip_code'].unique():
        df_zip = df_pred[df_pred['zip_code'] == zip_code]
        fig_pred.add_trace(go.Scatter(x=df_zip['date'],
y=df_zip['prediction'], mode='lines', name=f'ZIP {zip_code}'))

    fig_pred.update_layout(
        title='Real-Time Power Consumption Predictions',
        xaxis_title='Date', yaxis_title='Power (kWh)',
```

```
        template="plotly_dark"
    )

    # Real data figure
    fig_real = go.Figure()
    for zip_code in df_real['zip_code'].unique():
        df_zip = df_real[df_real['zip_code'] == zip_code]
        fig_real.add_trace(go.Scatter(x=df_zip['date'],
y=df_zip['actual'], mode='lines', name=f'ZIP {zip_code}'))

    fig_real.update_layout(
        title='Real-Time Actual Power Consumption',
        xaxis_title='Date', yaxis_title='Power (kWh)',
        template="plotly_dark"
    )

    return fig_pred, fig_real, latest_date

@app.callback(
    [Output('mae-score', 'children'),
     Output('rmse-score', 'children'),
     Output('r2-score', 'children')],
    Input('interval-component', 'n_intervals')
)
def update_metrics(n):
    ml_metrics = fetch_ml_metrics()

    if not ml_metrics:
        return "N/A", "N/A", "N/A"

    # Get average across all ZIP codes
    mae_list = [data['mae'] for data in ml_metrics.values()]
    rmse_list = [data['rmse'] for data in ml_metrics.values()]
    r2_list = [data['r2'] for data in ml_metrics.values()]

    mae = sum(mae_list) / len(mae_list) if mae_list else "N/A"
    rmse = sum(rmse_list) / len(rmse_list) if rmse_list else "N/A"
    r2 = sum(r2_list) / len(r2_list) if r2_list else "N/A"

    return f"{mae:.2f}", f"{rmse:.2f}", f"{r2:.2f}"

# Control simulation state, speed, and reset functionality
@app.callback(
    Output('interval-component', 'disabled'),
    [Input('start-button', 'n_clicks'),
     Input('pause-button', 'n_clicks'),
     Input('reset-button', 'n_clicks'),
     Input('speed-slider', 'value')]
)
def control_simulation(start, pause, reset, speed):
    ctx = dash.callback_context

    if not ctx.triggered:
        return True

    trigger_id = ctx.triggered[0]['prop_id'].split('.')[0]
```

```
    if trigger_id == "start-button":
        requests.post("http://127.0.0.1:5001/start")
        return False
    elif trigger_id == "pause-button":
        requests.post("http://127.0.0.1:5001/pause")
        return True
    elif trigger_id == "reset-button":
        requests.post("http://127.0.0.1:5001/reset")
        return True
    elif trigger_id == "speed-slider":
        requests.post("http://127.0.0.1:5001/set_speed",
json={"speed": speed})

    return False

if __name__ == '__main__':
    app.run_server(debug=True)
```



Εικόνα 20 Το περιβάλλον του Πίνακα Ελέγχου

3.3 Βελτιστοποίηση Συστήματος

Κατά τη διαδικασία εκτέλεσης των πειραμάτων, παρατηρήθηκε ότι η αρχική απόδοση του συστήματος δεν ανταποκρινόταν στις προσδοκίες μας, υποδεικνύοντας την ανάγκη για συστηματική βελτιστοποίηση. Για τον λόγο αυτό, ενσωματώθηκε μια διαδικασία βελτιστοποίησης υπερπαραμέτρων (Hyperparameter Tuning), με τη χρήση του πλαισίου Optuna.

3.3.α Βελτιστοποίηση Υπερπαραμέτρων (Hyperparameter Tuning)

Οι υπερπαράμετροι (hyperparameters) είναι παράμετροι των αλγορίθμων Μηχανικής Μάθησης που δεν μαθαίνονται απευθείας από τα δεδομένα κατά τη διάρκεια της εκπαίδευσης, αλλά ρυθμίζονται πριν από την έναρξη της διαδικασίας μάθησης [56]. Παραδείγματα υπερπαραμέτρων που χρησιμοποιούμε και στους τοπικούς κόμβους είναι ο

περιλαμβάνουν τον ρυθμό μάθησης (learning rate), ο αριθμός των δέντρων σε ένα Random Forest ή τα κρυφά επίπεδα σε ένα MLP Regressor.

Παραδοσιακές μέθοδοι ρύθμισης υπερπαραμέτρων, όπως η χειροκίνητη αναζήτηση (manual search), η αναζήτηση πλέγματος (Grid Search) και η τυχαία αναζήτηση (Random Search), συχνά αποδεικνύονται χρονοβόρες και υπολογιστικά ακριβές, ειδικά όταν ο χώρος αναζήτησης των υπερπαραμέτρων είναι μεγάλος ή όταν η αξιολόγηση κάθε συνδυασμού απαιτεί σημαντικούς πόρους [57]. Για την αντιμετώπιση αυτών των περιορισμών, υιοθετήθηκαν πιο εξελιγμένες, στοχευμένες (bayesian-based or evolutionary-based) μέθοδοι βελτιστοποίησης.

3.3.β Optuna

Το Optuna είναι ένα πλαίσιο αυτόματης βελτιστοποίησης υπερπαραμέτρων (hyperparameter optimization framework) ανοιχτού κώδικα, σχεδιασμένο για να αυτοματοποιεί την αναζήτηση των βέλτιστων τιμών. Η βασική του καινοτομία έγκειται στην υιοθέτηση μιας επιτακτικής προσέγγισης (imperative approach), επιτρέποντας στους χρήστες να ορίζουν δυναμικά τον χώρο αναζήτησης για κάθε υπερπαράμετρο κατά τη διάρκεια του πειράματος [58].

Στο πλαίσιο της εργασίας μας το Optuna χρησιμοποιήθηκε στην βελτιστοποίηση των υπερπαραμέτρων των μοντέλων Μηχανικής Μάθησης που εκπαιδεύονταν τοπικά σε κάθε τοπικό κόμβο του συστήματος. Συγκεκριμένα, για κάθε κόμβο, το Optuna εφάρμοσε μια ανεξάρτητη διαδικασία αναζήτησης για τον εντοπισμό των βέλτιστων ρυθμίσεων που μεγιστοποιούσαν την απόδοση του τοπικού μοντέλου στα διαθέσιμα δεδομένα του. Έπειτα βγάλαμε έναν μέσο όρο των υπερπαραμέτρων για κάθε ΤΚ και τα εισάγαμε στον κώδικα των τοπικών κόμβων. Η αντικειμενική συνάρτηση (objective function) για τη βελτιστοποίηση ήταν η μεγιστοποίηση του συντελεστή προσδιορισμού (R^2) στο σύνολο επικύρωσης (test size) του συνόλου δεδομένων (επιλέξαμε 20%) του εκάστοτε κόμβου. Παρακάτω βλέπουμε ένα τον κώδικα για την βελτιστοποίησης υπερπαραμέτρων του xgboost, ενώ επαναλάβαμε το ίδιο και για τα υπόλοιπα μοντέλα.

```
import optuna
import xgboost as xgb
import pandas as pd
import numpy as np
import sqlite3
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split

# Parse arguments
parser = argparse.ArgumentParser(description="Edge Node for
Hyperparameter Tuning"
parser.add_argument("--zip", type=int, required=True, help="ZIP code
for the dataset table.")
args = parser.parse_args()
ZIP_CODE = args.zip

conn = sqlite3.connect("datasets/dataset.db")
query = f"SELECT * FROM '{ZIP_CODE}'"
df_zip = pd.read_sql_query(query, conn)
conn.close()

# Preprocess Data
df_zip["date"] = pd.to_datetime(df_zip["date"])
df_zip["year"] = df_zip["date"].dt.year
```

```
df_zip["month"] = df_zip["date"].dt.month
df_zip["day"] = df_zip["date"].dt.day

X = df_zip[["year", "month", "day"]]
y = df_zip["power_consumption"]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Define Optuna Objective
def objective(trial):
    params = {
        "objective": "reg:squarederror",
        "tree_method": "hist",
        "device": "cuda",
        "learning_rate": trial.suggest_float("learning_rate",
0.0001, 0.1, log=True),
        "max_depth": trial.suggest_int("max_depth", 3, 10),
        "min_child_weight": trial.suggest_int("min_child_weight", 1,
10),
        "subsample": trial.suggest_float("subsample", 0.5, 1.0),
        "colsample_bytree": trial.suggest_float("colsample_bytree",
0.5, 1.0),
        "gamma": trial.suggest_float("gamma", 0.01, 1.0, log=True),
        "reg_lambda": trial.suggest_float("reg_lambda", 0.01, 10.0,
log=True),
        "reg_alpha": trial.suggest_float("reg_alpha", 0.01, 10.0,
log=True)
    }

    model = xgb.XGBRegressor(**params)
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    r2 = r2_score(y_test, y_pred)
    return r2  # Optuna minimizes, so we negate R²

# Run Hyperparameter Optimization
study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=1000, show_progress_bar=True)

# Model Evaluation
best_params = study.best_params
print("Best Hyperparameters:", best_params)

final_model = xgb.XGBRegressor(**best_params)
final_model.fit(X_train, y_train)

y_pred = final_model.predict(X_test)
r2 = r2_score(y_test, y_pred)
print(f"Final Model Test R²: {r2:.4f}")
```

Κατά την εκτέλεση του παραπάνω προγράμματος ή έξοδος στο τερματικό είναι το ακόλουθο κείμενο:

```
[I 2025-06-05 18:25:40,913] Trial 7 finished with value:
0.012030850810589633 and parameters: {'learning_rate':
0.0015224666731020555, 'max_depth': 10, 'min_child_weight': 2,
'subsample': 0.5568366327467433, 'colsample_bytree':
0.537450890114401, 'gamma': 0.6776012118512007, 'reg_lambda':
0.18518285042640628, 'reg_alpha': 3.8497813587958833}. Best is trial
0 with value: 0.16968754925555063.
[I 2025-06-05 18:25:42,230] Trial 8 finished with value:
0.0993597779118146 and parameters: {'learning_rate':
0.0070868388646132124, 'max_depth': 7, 'min_child_weight': 10,
'subsample': 0.8051706796828653, 'colsample_bytree':
0.9741817090556639, 'gamma': 0.10940368486583352, 'reg_lambda':
0.2512770083793208, 'reg_alpha': 8.470933320166736}. Best is trial 0
with value: 0.16968754925555063.
[I 2025-06-05 18:25:43,206] Trial 9 finished with value:
0.005813341045136533 and parameters: {'learning_rate':
0.0007196110848354124, 'max_depth': 4, 'min_child_weight': 9,
'subsample': 0.806367239781983, 'colsample_bytree':
0.5627615973261737, 'gamma': 0.05297065972200769, 'reg_lambda':
0.6330493822856244, 'reg_alpha': 0.018178222019169514}. Best is
trial 0 with value: 0.16968754925555063.
[I 2025-06-05 18:25:44,627] Trial 10 finished with value:
0.0033971886251609185 and parameters: {'learning_rate':
0.00015938766787631226, 'max_depth': 7, 'min_child_weight': 6,
'subsample': 0.6370469889467454, 'colsample_bytree':
0.983650819973901, 'gamma': 0.012189206310935279, 'reg_lambda':
0.07814615295059915, 'reg_alpha': 0.5824467788769824}. Best is trial
0 with value: 0.16968754925555063.
[I 2025-06-05 18:25:45,868] Trial 11 finished with value:
0.16982846602720347 and parameters: {'learning_rate':
0.09511126508206127, 'max_depth': 6, 'min_child_weight': 5,
'subsample': 0.8598778315130678, 'colsample_bytree':
0.8579033098771166, 'gamma': 0.033810937139884784, 'reg_lambda':
1.41524414982116, 'reg_alpha': 0.09276495601769766}. Best is trial
11 with value: 0.16982846602720347.
```

ΚΕΦΑΛΑΙΟ 4 Αξιολόγηση Αποτελεσμάτων

Στο παρόν κεφάλαιο περιγράφονται οι περιορισμοί που προέκυψαν καθώς και τα αποτελέσματα από τους πειραματισμούς που έγιναν για την αξιολόγηση της απόδοσής του συστήματος.

4.1 Περιορισμοί της Εργασίας

Κατά τη διάρκεια της εργασίας, αναδείχθηκαν διάφοροι περιορισμοί που επηρέασαν την υλοποίηση, την έκταση των πειραμάτων, αλλά και τα τελικά αποτελέσματα. Η αναγνώριση αυτών των περιορισμών είναι κρίσιμη για την αντικειμενική αξιολόγηση των ευρημάτων και τη χάραξη μελλοντικών ερευνητικών κατευθύνσεων.

4.1.α Διαθεσιμότητα Δεδομένων

Ένα από τα βασικότερα προβλήματα που κληθήκαμε να αντιμετωπίσουμε και αναφέραμε αρκετές φορές στην διάρκεια ήταν η διαθεσιμότητα στοιχείων που μας ώθησε στην δημιουργία ενός δικού μας dataset. Ακόμα και αν καταβάλαμε προσπάθεια ώστε το σύνολο δεδομένων να προσεγγίζει την πραγματικότητα, δεν είναι απίθανο, να μην αντιπροσωπεύει πλήρως το εύρος και την ποικιλομορφία των πραγματικών καταναλωτικών συνηθειών σε ένα ευρύτερο γεωγραφικό φάσμα ή σε διαφορετικούς τύπους καταναλωτών.

4.1.β Υπολογιστικό Κόστος

Η χρήση αλγορίθμων όπως XGBoost, LightGBM, CatBoost, και νευρωνικών δικτύων σε συνδυασμό με την εφαρμογή του Optuna για hyperparameter tuning σε κάθε edge node, είναι υπολογιστικά έντονη. Παρά την αποδοτικότητα αυτών των εργαλείων, ο υπολογιστικός χρόνος ξεπερνούσε πολλές φορές την μια μέρα, ενώ ήταν εξαντλητικό για τους πόρους. Δεν είναι τυχαίο άλλωστε που το μεγαλύτερο ποσοστό των σχετικών εργασιών χρησιμοποιούν υπερπολλαπλάσιους πόρους [59].

Για τον ίδιο λόγο άλλωστε δεν μπορέσαμε να χρησιμοποιήσουμε μοντέλα όπως το LSTM που θεωρείται το ανώτερο μοντέλο του κλάδου για την παρούσα εφαρμογή.

4.1.γ Περιβάλλον Υλοποίησης

Η λειτουργία των τοπικών κόμβων προσομοιώθηκε σε ένα κεντρικό υπολογιστικό περιβάλλον το οποίο είναι αδύνατον να αναπαράγει πλήρως την πολυπλοκότητα, τις καθυστερήσεις επικοινωνίας και τις αστοχίες ενός πραγματικού κατανεμημένου δικτύου, γεγονός που δεν επιτρέπει την αξιολόγηση του συστήματος υπό συνθήκες πραγματικής δικτυακής κίνησης και αστοχιών.

4.1.δ Περιορισμένη Πρόσβαση σε Βιβλιογραφία

Η περιορισμένη πρόσβαση σε πλήρη επιστημονικά άρθρα, ιδιαίτερα σε πιο πρόσφατες δημοσιεύσεις, αποτέλεσε έναν σημαντικό περιορισμό. Κατά την έρευνα για σχετική εργασία παρατηρήσαμε ότι το μεγαλύτερο ποσοστό αυτής έχει εκδοθεί τα τελευταία χρόνια. Πολλά από αυτά τα ερευνητικά άρθρα και δημοσιεύσεις είναι διαθέσιμα μόνο μέσω συνδρομών ή απαιτούν πληρωμή, γεγονός που δυσχέρανε την πλήρη εξάντληση της επικαιροποιημένης βιβλιογραφίας και την ενσωμάτωση των πιο πρόσφατων εξελίξεων στον τομέα.

4.2 Κριτήρια Αξιολόγησης

Για την αντικειμενική αξιολόγηση της απόδοσης των μοντέλων πρόβλεψης που αναπτύχθηκαν, χρησιμοποιήθηκαν οι ακόλουθες, καθιερωμένες μετρικές αξιολόγησης για

προβλήματα παλινδρόμησης. Η επιλογή πολλαπλών μετρικών είναι απαραίτητη για μια σφαιρική εικόνα της απόδοσης, καθώς κάθε μετρική αναδεικνύει διαφορετικές πτυχές του σφάλματος.

4.2.α Μέσο Απόλυτο Σφάλμα (Mean Absolute Error - MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Το MAE αντιπροσωπεύει τη μέση απόλυτη διαφορά μεταξύ των πραγματικών τιμών (y_i) και των προβλεπόμενων τιμών ($\hat{y}_i$). Το σύμβολο n αντιπροσωπεύει τον συνολικό αριθμό των σημείων δεδομένων (ή παρατηρήσεων) στο σύνολο δεδομένων που αξιολογείται, ενώ το i είναι ένας δείκτης που αναφέρεται σε κάθε επιμέρους σημείο δεδομένων (από 1 έως n) [60]. Είναι μια εύκολα ερμηνεύσιμη μετρική, καθώς εκφράζεται στην ίδια μονάδα με την προς πρόβλεψη μεταβλητή και είναι λιγότερο ευαίσθητο σε ακραίες τιμές (outliers) σε σύγκριση με το MSE.

4.2.β Μέσο Τετραγωνικό Σφάλμα (Mean Squared Error - MSE):

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

Το MSE υπολογίζει το μέσο τετράγωνο των διαφορών [60]. Δίνει μεγαλύτερη σημασία σε μεγάλα λάθη. Ενώ είναι χρήσιμο, η μονάδα μέτρησής του (τετράγωνο των μονάδων της πρόβλεψης) το καθιστά λιγότερο άμεσα ερμηνεύσιμο, για αυτό τον λόγο και δεν θα το χρησιμοποιήσουμε.

4.2.γ Ρίζα Μέσου Τετραγωνικού Σφάλματος (Root Mean Squared Error - RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Το RMSE είναι η τετραγωνική ρίζα του MSE [60]. Όπως και το MSE, δίνει βάρος στα μεγάλα σφάλματα, αλλά εκφράζεται στις ίδιες μονάδες με την προβλεπόμενη μεταβλητή, κάτι που βοηθάει στην κατανόησή του.

4.2.δ Συντελεστής Προσδιορισμού (R^2, R-squared):

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

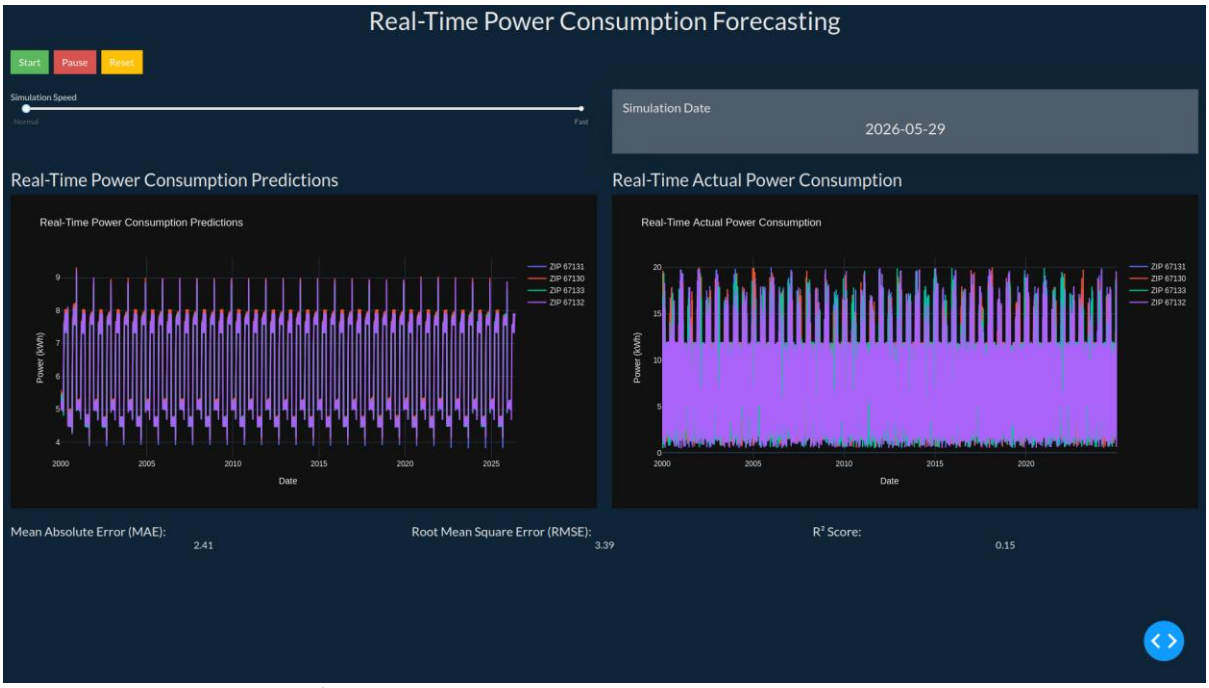
Ο συντελεστής R^2 δείχνει πόσο καλά το μοντέλο εξηγεί τη μεταβλητότητα των δεδομένων. Εδώ, το $\bar{y}$ αντιπροσωπεύει τη μέση τιμή των πραγματικών τιμών [60]. Μια τιμή κοντά στο 1 σημαίνει ότι το μοντέλο εξηγεί ένα μεγάλο μέρος της διακύμανσης, ενώ τιμές κοντά στο 0 (ή αρνητικές) δείχνουν ακόμα και μια γραμμική πρόβλεψη θα ήταν πιο αποδοτική.

4.3 Ανάλυση Αποτελεσμάτων

Κατά την πειραματική εκτέλεση και αξιολόγηση, παρατηρήθηκε μια σαφής και σημαντική βελτίωση της απόδοσης μεταξύ των διαδοχικών εκδόσεων του συστήματος, ειδικά όσον αφορά

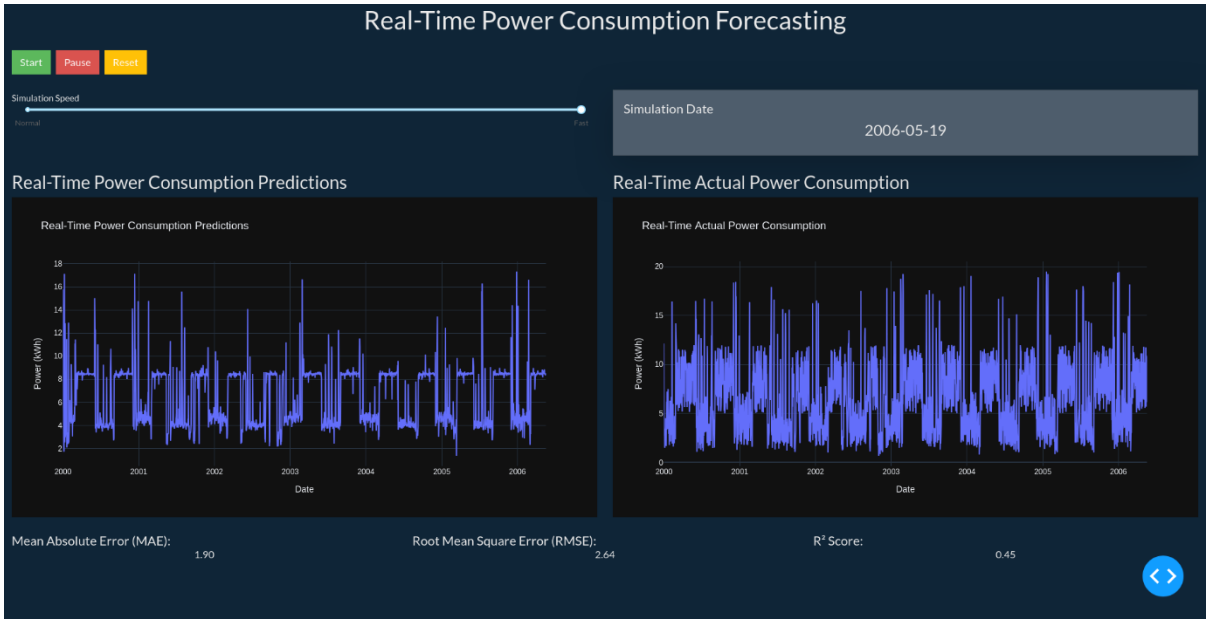
την ικανότητα των μοντέλων να εξηγούν τη μεταβλητότητα των δεδομένων, όπως αυτή αποτυπώνεται στον συντελεστή R^2.

- **Δεύτερη Έκδοση (Base Model: XGBoost & Optuna):** Κατά την αρχική φάση, όπου χρησιμοποιήθηκε ένα μόνο μοντέλο XGBoost ως βασικός μηχανισμός πρόβλεψης, με βελτίωση υπερπαραμέτρων μέσω Optuna οι τιμές του συντελεστή **R^2** κυμαίνονταν στην περιοχή του **0.17**. Αυτή η τιμή υποδηλώνει ότι το μοντέλο, αν και έδειχνε κάποια ικανότητα πρόβλεψης, εξηγούσε μόνο ένα μικρό ποσοστό της συνολικής διασποράς των δεδομένων. Αναφορικά με τα σφάλματα πρόβλεψης, οι μετρήσεις έδειξαν ένα **MAE** γύρω στο **2.50** και ένα **RMSE** περίπου στο **3.60**. Αυτές οι τιμές, υψηλότερες από αυτές της επόμενης έκδοσης, επιβεβαιώνουν ότι το μεμονωμένο μοντέλο XGBoost, παρά την ισχύ του, αντιμετώπιζε δυσκολίες στο να συλλάβει πλήρως τις πολύπλοκες και μη γραμμικές σχέσεις εντός των δεδομένων κατανάλωσης, πιθανώς λόγω της εγγενούς μεταβλητότητας και των εξωτερικών παραγόντων που επηρεάζουν την κατανάλωση ενέργειας.



Εικόνα 21 Μετρικές Δεύτερης Έκδοσης ΣΣΜΜ

- **Τρίτη Έκδοση (Ensemble Model: Stacking με Meta Model):** Με την υιοθέτηση του stacking ensemble και τη χρήση του Meta Model (που βασιζόταν επίσης σε XGBoost), παρατηρήθηκε μια σημαντική αύξηση στην απόδοση. Οι τιμές του συντελεστή **R^2** έφτασαν μέχρι το **0.5**, ενώ οι μετρικές σφάλματος βελτιώθηκαν σημαντικά, με το **MAE** να κυμαίνεται γύρω στο **1.9** και το **RMSE** περίπου στο **2.6**. Αυτή η αύξηση από το **0.17** στο **0.5** στον **R^2** και η μείωση των σφαλμάτων υπογραμμίζει την αποτελεσματικότητα της προσέγγισης ensemble. Το Meta Model κατάφερε να συνδυάσει τις προβλέψεις και τα δυνατά σημεία των επιμέρους base models (XGBoost, LightGBM, Random Forest, CatBoost, MLPRegressor), ξεπερνώντας τις αδυναμίες του κάθε μεμονωμένου μοντέλου. Η ποικιλομορφία των base models επέτρεψε στο σύστημα να μάθει διαφορετικές πτυχές των δεδομένων, με αποτέλεσμα μια πιο στιβαρή και ακριβή τελική πρόβλεψη. Η βελτίωση αυτή είναι ιδιαίτερα σημαντική, καθώς ένας **R^2** της τάξης του **0.5**, αν και όχι εξαιρετικά υψηλός, είναι πολύ πιο αποδεκτός για προβλήματα όπως πρόβλεψης κατανάλωσης ενέργειας που χαρακτηρίζονται από υψηλή πολυπλοκότητα και πιο συγκεκριμένα, με τους περιορισμούς που αντιμετωπίσαμε.



Εικόνα 22 Μετρικές Τρίτης Έκδοσης ΣΣΜΜ

Τα πειραματικά αποτελέσματα επίσης συγκρίθηκαν με αντίστοιχες μελέτες από τη διεθνή βιβλιογραφία [61] [62] [63] και ενώ το ΣΣΜΜ που αναπτύξαμε φαίνεται αρχικά υποδεέστερο, δεδομένου κυρίως περιορισμένων πόρων, αλλά όχι μόνο, το αποτέλεσμα είναι πραγματικά αξιόλογο.

ΚΕΦΑΛΑΙΟ 5 Συμπεράσματα και Μελλοντικές Επεκτάσεις

Στο τελευταίο κεφάλαιο θα παραθέσουμε τα συμπεράσματά μας, θα αξιολογήσουμε την επίτευξη των στόχων που θέσαμε στην αρχή και θα παραθέσουμε ιδέες για μελλοντικές επεκτάσεις.

5.1 Συμπεράσματα

Η παρούσα διπλωματική εργασία επικεντρώθηκε στην ανάπτυξη και αξιολόγηση ενός Συνεργατικού Σχήματος Μηχανικής Μάθησης για την πρόβλεψη φορτίου στο ΕΗΔ, αξιοποιώντας τις αρχές της κατανεμημένης μάθησης και του ensemble learning. Στόχος ήταν η επίτευξη βελτιωμένης ακρίβειας πρόβλεψης.

5.1.α Ευρήματα Εργασίας

Τα βασικά ευρήματα που προέκυψαν από την εκπόνηση και τα πειραματικά αποτελέσματα είναι τα εξής:

- **Αποτελεσματικότητα του Ensemble Learning:** Η κεντρική συνεισφορά της εργασίας αναδεικνύεται στην καταλυτική βελτίωση της προβλεπτικής ακρίβειας μέσω της υιοθέτησης του stacking ensemble. Η μετάβαση από ένα μεμονωμένο μοντέλο XGBoost (με $R^2 \approx 0.17$) σε ένα σύνθετο Meta Model που ενσωματώνει πολλαπλούς base models (XGBoost, LightGBM, Random Forest, CatBoost, MLPRegressor), οδήγησε σε μια εντυπωσιακή αύξηση του R_2 στο 0.5. Παράλληλα, παρατηρήθηκε σημαντική μείωση των σφαλμάτων (MAE από ~2.50 σε ~1.90 και RMSE από ~3.60 σε ~2.60). Αυτό αποδεικνύει αδιαμφισβήτητα ότι η συνεργασία διαφορετικών μοντέλων μπορεί να συλλάβει πιο αποτελεσματικά την πολυπλοκότητα και τη μεταβλητότητα των δεδομένων φορτίου, προσφέροντας προβλέψεις ανώτερης ποιότητας.
- **Βελτιστοποίηση με Optuna:** Η συστηματική εφαρμογή του πλαισίου Optuna για τη δυναμική βελτιστοποίηση των υπερπαραμέτρων σε κάθε τοπικό κόμβο (edge node) αποδείχθηκε θεμελιώδης. Αυτή η αυτοματοποιημένη διαδικασία εξασφάλισε ότι κάθε μοντέλο εκπαιδευόταν με τις βέλτιστες ρυθμίσεις για τα συγκεκριμένα τοπικά δεδομένα, μεγιστοποιώντας την ατομική του συνεισφορά στο τελικό ensemble. Η δυνατότητα αυτόματης και αποτελεσματικής ρύθμισης των υπερπαραμέτρων μειώνει σημαντικά το χειρωνακτικό φόρτο και βελτιώνει την αναπαραγωγιμότητα των αποτελεσμάτων.
- **Δυναμική της Κατανεμημένης Αρχιτεκτονικής:** Η σχεδιασμένη κατανεμημένη αρχιτεκτονική επιτρέπει την αποδοτική διαχείριση μεγάλων συνόλων δεδομένων, μειώνοντας την ανάγκη για κεντρική μεταφορά όλων των ακατέργαστων πληροφοριών. Αυτή η προσέγγιση είναι ιδιαίτερα κρίσιμη για εφαρμογές στο ΕΗΔ, καθώς ενισχύει την ιδιωτικότητα των δεδομένων και μειώνει τον επικοινωνιακό φόρτο στο δίκτυο, παράγοντας ταυτόχρονα ακριβείς προβλέψεις από τοπικά επεξεργασμένα δεδομένα. Η ενσωμάτωση τοπικής επεξεργασίας δεδομένων στα edge nodes αυξάνει την απόκριση του συστήματος και τη μείωση της καθυστέρησης (latency), στοιχεία κρίσιμα για τη διαχείριση σε πραγματικό χρόνο.
- **Ευελιξία και Επεκτασιμότητα Συστήματος:** Η αρθρωτή και κατανεμημένη φύση της προτεινόμενης αρχιτεκτονικής προσδίδει στο σύστημα υψηλό βαθμό ευελιξίας και επεκτασιμότητας. Αυτό σημαίνει ότι μπορούν εύκολα να ενσωματωθούν στο μέλλον νέα μοντέλα Μηχανικής Μάθησης, διαφορετικές τεχνικές ensemble ή ακόμα και πρόσθετοι edge nodes, χωρίς να απαιτείται πλήρης ανακατασκευή του συστήματος. Η δυνατότητα αυτή διασφαλίζει τη μακροβιότητα

και την προσαρμοστικότητα του συστήματος στις συνεχώς εξελισσόμενες απαιτήσεις του ΕΗΔ.

5.1.β Επίτευξη Στόχων

Η παρούσα εργασία ολοκληρώθηκε με βάση μια σειρά από σαφώς καθορισμένους στόχους, οι οποίοι επιτεύχθηκαν επιτυχώς:

- **Εμβάθυνση στις Έννοιες του ΕΗΔ, της Μηχανικής Μάθησης και του Συνεργατικού Σχήματος:** Πραγματοποιήθηκε εκτενής βιβλιογραφική ανασκόπηση και ανάλυση των θεμελιωδών εννοιών του Ευφυούς Ηλεκτρικού Δικτύου, των βασικών αρχών της Μηχανικής Μάθησης και των διαφορετικών τύπων συνεργατικών σχημάτων, με ιδιαίτερη έμφαση στις τεχνικές ensemble learning και την ομοσπονδιακή μάθηση. Αυτή η εμβάθυνση αποτέλεσε τη βάση για τον σχεδιασμό και την υλοποίηση του προτεινόμενου συστήματος.
- **Ανάπτυξη Αρχιτεκτονικής Συνεργατικού Σχήματος:** Σχεδιάστηκε και αναπτύχθηκε μια πρωτότυπη αρχιτεκτονική συνεργατικού σχήματος, η οποία επιτρέπει την εκπαίδευση μοντέλων Μηχανικής Μάθησης σε ένα κατανεμημένο περιβάλλον. Η αρχιτεκτονική αυτή εκμεταλλεύεται την υπολογιστική ισχύ των τοπικών κόμβων (edge nodes) για την εκπαίδευση base models και συνδυάζει τις προβλέψεις τους μέσω ενός Meta Model, επιτυγχάνοντας βελτιωμένη συνολική απόδοση.
- **Δημιουργία Προσομοίωσης και Πίνακα Ελέγχου:** Υλοποιήθηκε μια πλήρης προσομοίωση του προτεινόμενου συνεργατικού σχήματος, αναπαράγοντας τη λειτουργία των κατανεμημένων κόμβων και την κεντρική μονάδα συνδυασμού. Επιπλέον, σχεδιάστηκε ένας πίνακας ελέγχου (dashboard) που προσέφερε τη δυνατότητα οπτικοποίησης των αποτελεσμάτων των προβλέψεων και της παρακολούθησης της απόδοσης του συστήματος σε πραγματικό ή σχεδόν πραγματικό χρόνο.
- **Εξέταση του Ρόλου της Μηχανικής Μάθησης στην Πρόβλεψη Φορτίου και Βελτιστοποίηση:** Διερευνήθηκε διεξοδικά ο ρόλος της Μηχανικής Μάθησης στην ανάλυση δεδομένων του ΕΗΔ και την εφαρμογή της στην πρόβλεψη κατανάλωσης ηλεκτρικής ενέργειας για μια πόλη με πολλαπλούς ταχυδρομικούς κωδικούς. Ιδιαίτερη έμφαση δόθηκε στην αυτόματη υπερπαραμετροποίηση των μοντέλων μέσω του πλαισίου Optuna, γεγονός που συνέβαλε σημαντικά στην επίτευξη βέλτιστων επιδόσεων χωρίς χειροκίνητη παρέμβαση.
- **Αξιολόγηση Αποτελεσματικότητας και Αποδοτικότητας:** Πραγματοποιήθηκε συστηματική αξιολόγηση της αποτελεσματικότητας και της αποδοτικότητας του προτεινόμενου σχήματος. Τα πειραματικά αποτελέσματα κατέδειξαν τη βελτιωμένη ποιότητα της εξαγόμενης γνώσης (υψηλότερος R^2, χαμηλότερα MAE/ RMSE) και την ικανότητά του να αντιμετωπίζει τις προκλήσεις της πρόβλεψης φορτίου.

5.2 Μελλοντικές Επεκτάσεις

Η πτυχιακή μας εργασία, έθεσε τις βάσεις για την ανάπτυξη ενός συνεργατικού σχήματος Μηχανικής Μάθησης για την πρόβλεψη φορτίου στο ΕΗΔ. Παρόλα αυτά, το πεδίο αυτό είναι δυναμικό και προσφέρει εκτεταμένες ευκαιρίες για περαιτέρω έρευνα και ανάπτυξη. Οι ακόλουθες κατευθύνσεις προτείνονται ως πιθανές μελλοντικές επεκτάσεις της παρούσας εργασίας, με στόχο την ενίσχυση της στιβαρότητας, της ακρίβειας και της πρακτικής εφαρμογής του συστήματος:

5.2.α Ασφάλεια Δεδομένων

Πέρα από την ιδιωτικότητα, η ασφάλεια των δεδομένων και η ανθεκτικότητα του συστήματος σε κακόβουλες επιθέσεις είναι κρίσιμη για εφαρμογές ΕΗΔ. Μελλοντική έρευνα μπορεί να περιλαμβάνει τη διερεύνηση:

- **Ανθεκτικότητας σε δηλητηρίαση δεδομένων (data poisoning attacks):** Πώς το σύστημα μπορεί να ανιχνεύει και να μετριάζει τις επιπτώσεις κακόβουλων δεδομένων που στοχεύουν στην υποβάθμιση της απόδοσης [64] [65].
- **Ανθεκτικότητας σε επιθέσεις υποκλοπής μοντέλων (model inversion attacks):** Να διασφαλιστεί ότι οι ενημερώσεις των μοντέλων δεν αποκαλύπτουν ευαίσθητες πληροφορίες για τα αρχικά δεδομένα [66].
- **Ασφαλούς Ομοσπονδιακής Μάθησης (Secure Federated Learning):** Ενσωμάτωση πρωτοκόλλων που διασφαλίζουν την ακεραιότητα και την αυθεντικότητα των επικοινωνιών και των μοντέλων σε ένα κατανεμημένο περιβάλλον [67].

5.2.β Δημιουργία Δημοσίων Dataset

Η διαθεσιμότητα υψηλής ποιότητας, δημοσίων συνόλων δεδομένων κατανάλωσης ενέργειας αποτελεί σημαντικό περιορισμό στην ακαδημαϊκή έρευνα. Μια μελλοντική συνεισφορά θα μπορούσε να είναι η δημιουργία και διάθεση ενός ανωνυμοποιημένου και επιμελημένου dataset από τα δεδομένα που χρησιμοποιήθηκαν σε αυτή την εργασία (ή παρόμοια), ακολουθώντας βέλτιστες πρακτικές για την προστασία της ιδιωτικότητας. Ένα τέτοιο dataset θα διευκόλυνε την αναπαραγωγιμότητα της έρευνας, θα επέτρεπε σε άλλους ερευνητές να συγκρίνουν τα δικά τους μοντέλα και θα επιτάχυνε την πρόοδο στον τομέα της πρόβλεψης φορτίου [68].

5.2.γ Βελτιστοποίηση Επικοινωνίας

Μια σημαντική πτυχή για την ανάπτυξη κατανεμημένων συστημάτων είναι η αποτελεσματικότητα της επικοινωνίας μεταξύ των κόμβων. Μελλοντική εργασία θα μπορούσε να επικεντρωθεί στη βελτιστοποίηση των πρωτοκόλλων επικοινωνίας και των μηχανισμών μεταφοράς δεδομένων. Αυτό περιλαμβάνει τη διερεύνηση τεχνικών όπως η συμπίεση δεδομένων (data compression), η αποστολή μόνο των διαφορών (delta updates) στα μοντέλα αντί για ολόκληρα τα μοντέλα, ή η χρήση πιο αποδοτικών πρωτοκόλλων (π.χ., MQTT) [69] ειδικά σχεδιασμένων για IoT και Edge Computing περιβάλλοντα. Στόχος είναι η περαιτέρω μείωση του δικτυακού φόρτου και των καθυστερήσεων, καθιστώντας το σύστημα πιο βιώσιμο σε περιβάλλοντα με περιορισμένο εύρος ζώνης.

5.2.δ Hardware Acceleration

Η ενσωμάτωση επιταχυντών υλικού αποτελεί μια πολλά υποσχόμενη κατεύθυνση για την ενίσχυση της υπολογιστικής απόδοσης των μοντέλων. Η διερεύνηση της χρήσης Επεξεργαστών Γραφικών (GPUs) σε edge devices (όπου αυτό είναι εφικτό και οικονομικά βιώσιμο) μπορεί να μειώσει δραματικά τον χρόνο εκπαίδευσης και πρόβλεψης, ειδικά για πιο σύνθετα μοντέλα βαθιάς μάθησης. Η βελτιστοποίηση του κώδικα για παράλληλη επεξεργασία σε αυτές τις αρχιτεκτονικές είναι κρίσιμη για την εκμετάλλευση της πλήρους υπολογιστικής τους ισχύος [70].

5.2.ε Χρήση FPGA

Ως επέκταση της επιτάχυνσης υλικού, η αξιοποίηση των FPGA προσφέρει μοναδικά πλεονεκτήματα για εξειδικευμένες εφαρμογές ΕΗΔ. Τα FPGA επιτρέπουν την προσαρμοσμένη αρχιτεκτονική υλικού για συγκεκριμένους αλγορίθμους Μηχανικής Μάθησης, επιτυγχάνοντας εξαιρετικά υψηλή ενεργειακή αποδοτικότητα και απόδοση σε σύγκριση με τις γενικής χρήσης CPUs/ GPUs για συγκεκριμένες εργασίες. Η έρευνα στην

υλοποίηση τμημάτων του collaborative scheme (π.χ., οι πυρήνες των base models ή η διαδικασία του Meta Model) σε FPGA θα μπορούσε να οδηγήσει σε λύσεις με πολύ χαμηλότερη κατανάλωση ενέργειας και ταχύτερη απόκριση, ιδανικές για ενσωμάτωση σε περιορισμένα edge devices [70] [71].

ΠΑΡΑΡΤΗΜΑ

Εργαλεία οπτικοποίησης

Με τους παρακάτω κώδικες πραγματοποιούμε την οπτικοποιήση που μας δείχνει τις τάσεις των καταναλώσεων.

trend_viz_city_montly,py

Εμφανίζει ένα διάγραμμα της μέσης κατανάλωσης όλης της πόλης, μέσα στα χρόνια, σε σχέση με τον μήνα. Αυτό είναι ένα το πιο χρήσιμο στοιχείο οπτικοποίησης, στην δημιουργία του dataset, όπως θα δούμε παρακάτω.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import argparse
import os

# Step 1: Parse command-line arguments
parser = argparse.ArgumentParser(description="Visualize Monthly
Average Power Consumption for the Whole City and Save the Plot")
parser.add_argument("--dataset", type=str, required=True, help="Path
to the dataset CSV file")

args = parser.parse_args()
dataset_path = args.dataset  # Get the dataset file path from
arguments

# Step 2: Extract dataset name and create the output filename
dataset_filename = os.path.basename(dataset_path)  # Get only the
file name (without the full path)
dataset_name = os.path.splitext(dataset_filename)[0]  # Remove the
file extension
output_filename = f"trends_viz_city_seasonal_{dataset_name}.png"  #
New output file name

# Step 3: Load the CSV
df = pd.read_csv(dataset_path)

# Step 4: Extract Year and Month from Date
df['year'] = pd.to_datetime(df['date']).dt.year
df['month'] = pd.to_datetime(df['date']).dt.month

# Step 5: Group by Year and Month to calculate the average power
consumption for the whole city
avg_power_monthly_city = df.groupby(['year',
'month'])['power_consumption'].mean().reset_index()

# Step 6: Pivot the data for easier plotting (year as rows and
months as columns)
pivoted = avg_power_monthly_city.pivot(index='year',
columns='month', values='power_consumption')

# Step 7: Plot the trends by month
```

```
plt.figure(figsize=(14, 7))

# Plot each month's trend across the years
for month in range(1, 13):
    plt.plot(pivoted.index, pivoted[month], marker='o',
label=f'Month {month}')

# Customize the plot
plt.title('Monthly Average Power Consumption for the Whole City
(2000-2024)', fontsize=16)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Average Power Consumption (kWh)', fontsize=12)
plt.xticks(rotation=45)
plt.legend(title='Months', bbox_to_anchor=(1.05, 1), loc='upper
left')
plt.grid(True)
plt.tight_layout()

plt.savefig(output_filename, dpi=300, bbox_inches="tight")  # High-
quality PNG image
print(f"Plot saved as {output_filename}")

# Show the plot
plt.show()
```

trend_viz_city_yearly.py

Εμφανίζει ένα διάγραμμα της μέσης κατανάλωσης των νοικοκυριών, όλης της πόλης, σε σχέση με τον χρόνο.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import argparse
import os

# Step 1: Parse command-line arguments
parser = argparse.ArgumentParser(description="Visualize Annual
Average Power Consumption for the Whole City and Save the Plot")
parser.add_argument("--dataset", type=str, required=True, help="Path
to the dataset CSV file")

args = parser.parse_args()
dataset_path = args.dataset  # Get the dataset file path from
arguments

# Step 2: Extract dataset name and create the output filename
dataset_filename = os.path.basename(dataset_path)  # Get only the
file name (without the full path)
dataset_name = os.path.splitext(dataset_filename)[0]  # Remove the
file extension
output_filename = f"trends_viz_city_annual_{dataset_name}.png"  #
New output file name

# Step 3: Load the CSV
df = pd.read_csv(dataset_path)
```

```
# Step 4: Extract Year from Date
df['year'] = pd.to_datetime(df['date']).dt.year

# Step 5: Group by Year to calculate the average power consumption
for the whole city
avg_power_city =
df.groupby(['year'])['power_consumption'].mean().reset_index()

# Step 6: Plot the city-wide trend
plt.figure(figsize=(14, 7))
plt.plot(avg_power_city['year'],
avg_power_city['power_consumption'], marker='o', color='b',
label='City-Wide Average Power Consumption')

# Customize the plot
plt.title('Annual Average Power Consumption for the Whole City
(2000-2024)', fontsize=16)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Average Power Consumption (kWh)', fontsize=12)
plt.grid(True)
plt.legend()
plt.tight_layout()

# Save the plot as an image
plt.savefig(output_filename, dpi=300)  # Save as a high-quality PNG
image
print(f"Plot saved as {output_filename}")

# Show the plot
plt.show()
```

trend_viz_zip_yearly.py

Εμφανίζει ένα διάγραμμα της μέσης κατανάλωσης των νοικοκυριών, που βρίσκονται σε έναν συγκεκριμένο ΤΚ, σε σχέση με τον χρόνο.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import argparse
import os

# Step 1: Parse command-line arguments
parser = argparse.ArgumentParser(description="Visualize Annual Power
Consumption Trends by Zip Code and Save the Plot")
parser.add_argument("--dataset", type=str, required=True, help="Path
to the dataset CSV file")

args = parser.parse_args()
dataset_path = args.dataset  # Get the dataset file path from
arguments

# Step 2: Extract dataset name and create the output filename
dataset_filename = os.path.basename(dataset_path)  # Get only the
file name (without the full path)
```

```
dataset_name = os.path.splitext(dataset_filename)[0]  # Remove the
file extension
output_filename = f"trends_viz_zipcode_annual_{dataset_name}.png"  #
New output file name

# Step 3: Load the CSV
df = pd.read_csv(dataset_path)

# Step 4: Extract Year from Date
df['year'] = pd.to_datetime(df['date']).dt.year

# Step 5: Group by Zip Code and Year, then calculate the average
power consumption
avg_power_by_zip = df.groupby(['zip_code',
'year'])['power_consumption'].mean().reset_index()

# Step 6: Plot the trends for each zip code
plt.figure(figsize=(14, 7))

# Loop through each unique zip code and plot its trend over years
for zip_code in avg_power_by_zip['zip_code'].unique():
    zip_code_data = avg_power_by_zip[avg_power_by_zip['zip_code'] ==
zip_code]
    plt.plot(zip_code_data['year'],
zip_code_data['power_consumption'], label=f'Zip Code {zip_code}')

# Customize the plot
plt.title('Average Power Consumption Trend by Zip Code (2000-2024)',
fontsize=16)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Average Power Consumption (kWh)', fontsize=12)
plt.legend(title='Zip Codes', bbox_to_anchor=(1.05, 1), loc='upper
left')
plt.grid(True)
plt.tight_layout()

# Save the plot as an image
plt.savefig(output_filename, dpi=300)  # Save as a high-quality PNG
image
print(f"Plot saved as {output_filename}")

# Show the plot
plt.show()
```

Ρεαλιστικό Σύνολο Δεδομένων

Ένα ακόμα dataset που δημιουργήσαμε είναι το παρακάτω. Περιλαμβάνει ό,τι και τα υπόλοιπα, καθώς και επιπλέον παράγοντες:

- Το μέγεθος του σπιτιού
- Τις ανάγκες, ανάλογα με τα μέλη του νοικοκυριού
- Αν το σπίτι έχει φωτοβολταϊκά
- Αν η μέρα είναι καθημερινή ή Σαββατοκύριακο (διαφορετικές ώρες που βρισκόμαστε στο σπίτι, άρα διαφορετικές καταναλώσεις)

```
import pandas as pd
import numpy as np

# Step 1: Create date range
dates = pd.date_range(start="2000-01-01", end="2024-12-31",
freq="H")  # Hourly data

# Step 2: Define house IDs and zip codes
house_ids = [f"{i+1}" for i in range(3000)]
zip_codes = [67130, 67131, 67132, 67133]  # Example ZIP codes

# Randomly select 2% of houses to be vacant (not used)
vacant_houses = set(np.random.choice(house_ids,
size=int(len(house_ids) * 0.02), replace=False))

# Assign house sizes (impacts energy usage)
house_sizes = {
    house: np.random.choice(["small", "medium", "large",
"very_large"], p=[0.3, 0.4, 0.2, 0.1])
    for house in house_ids
}

# Assign solar panels to 15% of houses
solar_homes = set(np.random.choice(house_ids,
size=int(len(house_ids) * 0.15), replace=False))

# Assign different household types (affecting power usage patterns)
household_types = {
    house: np.random.choice(["family", "single", "elderly",
"vacation_home"], p=[0.4, 0.3, 0.2, 0.1])
    for house in house_ids
}

# Function to generate seasonal power consumption
def seasonal_power_consumption(date):
    month = date.month
    if month in [12, 1, 2]:  # Winter
        return np.random.uniform(12, 20)
    elif month in [6, 7, 8]:  # Summer
        return np.random.uniform(10, 18)
    else:  # Spring/Fall
        return np.random.uniform(5, 12)

# Function to apply house size multiplier
def apply_house_size_multiplier(consumption, house_size):
    size_multipliers = {"small": 0.8, "medium": 1.0, "large": 1.2,
"very_large": 1.4}
    return consumption * size_multipliers[house_size]

# Function to apply solar panel reduction
def apply_solar_reduction(consumption, date, house):
    if house in solar_homes and 8 <= date.hour <= 18:  # Daylight
hours
        return consumption * np.random.uniform(0.5, 0.8)  # Reduce
by 20-50%
    return consumption
```

```
# Function to apply weekday vs. weekend consumption patterns
def apply_weekend_adjustment(consumption, date):
    if date.weekday() >= 5:  # 5 = Saturday, 6 = Sunday
        return consumption * np.random.uniform(1.1, 1.3)  # Increase
by 10-30% on weekends
    return consumption

# Function to modify power usage based on household type
def apply_household_type_pattern(consumption, date, household_type):
    hour = date.hour

    if household_type == "family":
        if 6 <= hour <= 8 or 18 <= hour <= 22:  # Higher morning &
evening usage
            return consumption * np.random.uniform(1.2, 1.5)
    elif household_type == "single":
        if 19 <= hour <= 23:  # Higher evening usage
            return consumption * np.random.uniform(1.1, 1.3)
    elif household_type == "elderly":
        if 9 <= hour <= 17:  # Higher daytime usage
            return consumption * np.random.uniform(1.2, 1.4)
    elif household_type == "vacation_home":
        if np.random.rand() < 0.3:  # 30% chance of being unoccupied
            return consumption * np.random.uniform(0.3, 0.6)  #
Reduce by 40-70%

    return consumption

# Function to apply vacation reductions
def random_vacation_reduction(power_consumption, date,
vacation_info, season_vacation_chance):
    """Applies vacation power reductions, simulating consecutive
days of absence."""
    if vacation_info["on_vacation"] and (date -
vacation_info["start_date"]).days <
vacation_info["vacation_duration"]:
        return power_consumption * np.random.uniform(0.1, 0.4)  #
Reduce by 60-90%

    if np.random.rand() < season_vacation_chance:
        vacation_info["on_vacation"] = True
        vacation_info["start_date"] = date
        vacation_info["vacation_duration"] = np.random.randint(3,
15)  # 3-14 days
        return power_consumption * np.random.uniform(0.1, 0.4)  #
Reduce by 60-90%

    vacation_info["on_vacation"] = False
    return power_consumption

# Function to generate energy breaks (power outages for 1-6 hours)
def generate_energy_breaks(zip_codes, dates, probability=0.002):
    """Generates short power outages (1-6 hours) affecting entire
ZIP codes."""
    outages = {}
```

```
    for date in dates:
        if np.random.rand() < probability:  # Small chance of an
outage occurring
            affected_zip = np.random.choice(zip_codes)
            outage_duration = np.random.randint(1, 7)  # Outage
lasts 1-6 hours
            outages[date] = {"zip_code": affected_zip, "hours":
outage_duration}
    return outages

# Step 3: Generate power outages
energy_breaks = generate_energy_breaks(zip_codes, dates)

# Step 4: Generate dataset with all features
data = []
vacation_info = {house: {"on_vacation": False, "start_date": None,
"vacation_duration": 0} for house in house_ids}

for house in house_ids:
    zip_code = np.random.choice(zip_codes)
    house_size = house_sizes[house]
    household_type = household_types[house]

    for date in dates:
        if house in vacant_houses:
            power_consumption = np.random.uniform(0.5, 2)  # Minimal
standby power
        else:
            power_consumption = seasonal_power_consumption(date)
            power_consumption =
apply_house_size_multiplier(power_consumption, house_size)
            power_consumption =
apply_solar_reduction(power_consumption, date, house)
            power_consumption =
apply_weekend_adjustment(power_consumption, date)
            power_consumption =
apply_household_type_pattern(power_consumption, date,
household_type)

            if date.month in [12, 1, 2, 6, 7, 8]:  # Winter and
Summer months
                season_vacation_chance = 0.5  # 50% chance per day
            else:
                season_vacation_chance = 0.01  # 1% chance per day

            power_consumption =
random_vacation_reduction(power_consumption, date,
vacation_info[house], season_vacation_chance)

        if date in energy_breaks and zip_code ==
energy_breaks[date]["zip_code"]:
            outage_hours = energy_breaks[date]["hours"]
            power_consumption *= (1 - outage_hours / 24)  # Reduce
power proportionally

        data.append({
```

```
            "date": date,
            "house_id": house,
            "zip_code": zip_code,
            "house_size": house_size,
            "household_type": household_type,
            "solar_panels": house in solar_homes,
            "vacant": house in vacant_houses,
            "power_consumption": power_consumption
        })

# Step 5: Save dataset
df = pd.DataFrame(data)
df.to_csv("realistic_energy_usage_dataset.csv", index=False)
```

Το παρόν dataset δεν δοκιμάστηκε, μιας και λόγω του θεωρητικού μεγέθους του (≈32GB), δεν ήταν δυνατόν να δημιουργηθεί στο υπολογιστικό μας σύστημα, μιας και κατά την διάρκεια δημιουργίας του csv, τα δεδομένα αποθηκεύονται προσωρινή στην RAM. Με την χρήση ενός άλλου τρόπου *(parquet)* θα μπορούσε να αποθηκευτεί σε λιγότερο από 8GB αλλά και πάλι, κατά την διαδικασία επεξεργασίας των δεδομένων και κατά την πρόβλεψη, θα είχε γιγάντιο υπολογιστικό κόστος, που θα χρειαζόταν πόρους που δεν μπορούσαμε να διαθέσουμε. Επομένως, το αναφέρουμε στο Παράρτημα σαν ένα θεωρητικό σύνολο δεδομένων που θα μπορούσε να είχε χρησιμοποιηθεί.

Υλοποίηση ΣΣΜΜ: Έκδοση 1

Η πρώτη έκδοση, αποτελεί μια απλοϊκή μορφή. Τα edge nodes χρησιμοποιούν xgboost, με τυχαίες τιμές. Η ανάγνωση των δεδομένων γίνεται από έναν αρχείο .csv ενώ στον πίνακα ελέγχου μπορούμε να δούμε μόνο το γράφημα των προβλέψεων.

edge_node.py

```
import requests
import pandas as pd
import numpy as np
import xgboost as xgb
import time
import argparse
import socket
import os

parser = argparse.ArgumentParser(description="Edge Node for Power
Consumption Forecasting")
parser.add_argument("--zip", type=int, required=True, help="ZIP code
for this edge node")
args = parser.parse_args()
ZIP_CODE = args.zip

NODE_ID = f"{socket.gethostname()}_{os.getpid()}"

df =
pd.read_csv('datasets/dataset_with_seasonal_vacation_outages_and_vac
ant_houses.csv')
df['date'] = pd.to_datetime(df['date'])
```

```
df_zip = df[df['zip_code'] == ZIP_CODE].copy()
df_zip['year'] = df_zip['date'].dt.year
df_zip['month'] = df_zip['date'].dt.month
df_zip['day'] = df_zip['date'].dt.day

train_data = df_zip[df_zip['year'] < 2024]
X_train = train_data[['year', 'month', 'day']]
y_train = train_data['power_consumption']

model = xgb.XGBRegressor(objective='reg:squarederror',
n_estimators=200, learning_rate=0.1)
model.fit(X_train, y_train)

def fetch_speed():
    try:
        response =
requests.get("http://127.0.0.1:5001/get_speed").json()
        return float(response.get("speed", 1.0))
    except requests.exceptions.RequestException:
        return 1.0

def run_simulation():
    sim_date = pd.Timestamp("2000-01-01")
    last_reset_status = False

    while True:
        # Check simulation status
        status = requests.get("http://127.0.0.1:5001/status").json()
        if not status.get("running", False):
            time.sleep(1)
            continue

        # Check reset signal
        reset_response =
requests.post("http://127.0.0.1:5001/get_reset_signal",
json={"node_id": NODE_ID}).json()
        if reset_response.get("reset", False):
            print(f"Node {NODE_ID} detected reset! Restarting
simulation.")
            sim_date = pd.Timestamp("2000-01-01")
            last_reset_status = True
            continue

        speed_factor = fetch_speed()

        X_pred = pd.DataFrame([[sim_date.year, sim_date.month,
sim_date.day]], columns=['year', 'month', 'day'])
        prediction = float(model.predict(X_pred)[0])  # Ensure JSON
serialization

        response =
requests.post("http://127.0.0.1:5001/submit_prediction", json={
            "date": sim_date.strftime('%Y-%m-%d'),
            "zip_code": ZIP_CODE,
            "prediction": prediction
        })
```

```
        print(f"Node {NODE_ID} submitted prediction for
{sim_date.strftime('%Y-%m-%d')} at speed {speed_factor}, response:
{response.status_code}")

        sim_date += pd.Timedelta(days=1)

        sleep_time = max(0.01, 1.0 / speed_factor)
        time.sleep(sleep_time)

run_simulation()
```

master_node.py

```
from flask import Flask, request, jsonify

app = Flask(__name__)

global_predictions = []
simulation_date = None
simulation_running = False
reset_signal = False
nodes_detected_reset = set()
speed_factor = 1
EXPECTED_NODES = 4

@app.route('/submit_prediction', methods=['POST'])
def submit_prediction():
    global simulation_date
    data = request.json
    simulation_date = data["date"]
    global_predictions.append(data)
    return jsonify({"status": "success"}), 200

@app.route('/get_predictions', methods=['GET'])
def get_predictions():
    return jsonify(global_predictions), 200

@app.route('/start', methods=['POST'])
def start_simulation():
    global simulation_running
    simulation_running = True
    return jsonify({"status": "started", "running": True}), 200

@app.route('/pause', methods=['POST'])
def pause_simulation():
    global simulation_running
    simulation_running = False
    return jsonify({"status": "paused", "running": False}), 200

@app.route('/reset', methods=['POST'])
def reset_simulation():
    global global_predictions, simulation_date, simulation_running,
reset_signal, nodes_detected_reset
    global_predictions = []
    simulation_date = None
```

```
    simulation_running = False
    reset_signal = True
    nodes_detected_reset.clear()
    return jsonify({"status": "reset"}), 200

@app.route('/get_reset_signal', methods=['POST'])
def get_reset_signal():
    global reset_signal, nodes_detected_reset
    data = request.json
    node_id = data.get("node_id")

    if reset_signal and node_id not in nodes_detected_reset:
        nodes_detected_reset.add(node_id)
        if len(nodes_detected_reset) >= EXPECTED_NODES:
            reset_signal = False
        return jsonify({"reset": True})

    return jsonify({"reset": False})

@app.route('/status', methods=['GET'])
def get_status():
    return jsonify({"running": simulation_running}), 200

@app.route('/get_speed', methods=['GET'])
def get_speed():
    """Returns the current simulation speed."""
    return jsonify({"speed": speed_factor}), 200

@app.route('/set_speed', methods=['POST'])
def set_speed():
    global speed_factor
    data = request.json
    speed_factor = data.get("speed", 1)
    return jsonify({"status": "speed updated", "speed":
speed_factor}), 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5001, debug=True)
```

dashboard.py

```
import dash
import dash_bootstrap_components as dbc
from dash import dcc, html
from dash.dependencies import Input, Output
import plotly.graph_objects as go
import requests
import pandas as pd

app = dash.Dash(__name__,
external_stylesheets=[dbc.themes.SUPERHERO])

app.layout = dbc.Container([
    dbc.Row([
        dbc.Col(html.H1("Real-Time Power Consumption Forecasting",
```

```
classNameName="text-center mb-4"), width=12)
    ]),

    # Control Buttons
    dbc.Row([
        dbc.Col([
            dbc.Button("Start", id="start-button", color="success",
classNameName="me-2"),
            dbc.Button("Pause", id="pause-button", color="danger",
classNameName="me-2"),
            dbc.Button("Reset", id="reset-button", color="warning"),
        ], width=6)
    ], class_name="mb-4"),

    # Speed Control & Simulation Date
    dbc.Row([
        dbc.Col([
            html.Label("Simulation Speed"),
            dcc.Slider(id='speed-slider', min=1, max=100, step=1,
value=1,
                       marks={1: 'Normal', 100: 'Fast'})
        ], width=6),
        dbc.Col([
            dbc.Card([
                dbc.CardBody([
                    html.H5("Simulation Date", classNameName="card-
title"),
                    html.H4(id="simulation-date", classNameName="text-
center")
                ])
            ], class_name="shadow-lg")
        ], width=6)
    ], class_name="mb-4"),

    # Live Graph
    dbc.Row([
        dbc.Col([
            dcc.Graph(id='live-graph', style={"height": "500px"})
        ], width=12)
    ]),

    # Interval Component for Real-Time Updates
    dcc.Interval(id='interval-component', interval=1000,
n_intervals=0, disabled=True)
], fluid=True)

# Fetch data from master node
def fetch_predictions():
    try:
        response =
requests.get("http://127.0.0.1:5001/get_predictions")
        if response.status_code == 200:
            return response.json()
    except requests.exceptions.RequestException:
        return []
    return []
```

```
# Fetch simulation speed
def fetch_speed():
    try:
        response = requests.get("http://127.0.0.1:5001/get_speed")
        if response.status_code == 200:
            return response.json().get("speed", 1)
    except requests.exceptions.RequestException:
        return 1

# Update the graph with new data
@app.callback(
    [Output('live-graph', 'figure'),
     Output('simulation-date', 'children')],
    Input('interval-component', 'n_intervals')
)
def update_graph(n):
    predictions = fetch_predictions()
    if not predictions:
        return go.Figure(), "No Data"

    df_pred = pd.DataFrame(predictions)
    df_pred['date'] = pd.to_datetime(df_pred['date'])
    latest_date = df_pred["date"].max().strftime('%Y-%m-%d')

    fig = go.Figure()
    for zip_code in df_pred['zip_code'].unique():
        df_zip = df_pred[df_pred['zip_code'] == zip_code]
        fig.add_trace(go.Scatter(x=df_zip['date'],
y=df_zip['prediction'], mode='lines', name=f'ZIP {zip_code}'))

    fig.update_layout(
        title='Real-Time Power Consumption Predictions',
        xaxis_title='Date', yaxis_title='Power (kWh)',
        template="plotly_dark"
    )

    return fig, latest_date

# Control simulation state, speed, and reset functionality
@app.callback(
    Output('interval-component', 'disabled'),
    [Input('start-button', 'n_clicks'),
     Input('pause-button', 'n_clicks'),
     Input('reset-button', 'n_clicks'),
     Input('speed-slider', 'value')]
)
def control_simulation(start, pause, reset, speed):
    ctx = dash.callback_context

    if not ctx.triggered:
        return True

    trigger_id = ctx.triggered[0]['prop_id'].split('.')[0]
    if trigger_id == "start-button":
        requests.post("http://127.0.0.1:5001/start")
```

```
        return False
    elif trigger_id == "pause-button":
        requests.post("http://127.0.0.1:5001/pause")
        return True
    elif trigger_id == "reset-button":
        requests.post("http://127.0.0.1:5001/reset")
        return True
    elif trigger_id == "speed-slider":
        requests.post("http://127.0.0.1:5001/set_speed",
json={"speed": speed})

    return False
if __name__ == '__main__':
    app.run_server(debug=True)
```

Υλοποίηση ΣΣΜΜ: Έκδοση 2

Η δεύτερη έκδοση, αναπτύσσει πάνω στην προηγούμενη. Τα edge nodes, πλέον, χρησιμοποιούν xgboost, με υπερπαραμέτρους όπως αναφέραμε και στο Κεφάλαιο 3.3. Η ανάγνωση των δεδομένων γίνεται πάλι από ένα αρχείο .csv αλλά στον πίνακα ελέγχου μπορούμε πλέον να δούμε τα γραφήματα των προβλέψεων, των πραγματικών τιμών καθώς και τις μετρικές αξιολόγησης του συστήματος. Βέβαια στα πειράματα που πραγματοποιήσαμε, η μετρική του R^2 δεν ξεπερνούσε το 0.15.

edge_node.py

```
import requests
import pandas as pd
import numpy as np
import xgboost as xgb
import time
import argparse
import socket
import os
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score

historical_predictions = []
historical_actuals = []

parser = argparse.ArgumentParser(description="Edge Node for Power
Consumption Forecasting")
parser.add_argument("--zip", type=int, required=True, help="ZIP code
for this edge node")
args = parser.parse_args()
ZIP_CODE = args.zip

NODE_ID = f"{socket.gethostname()}_{os.getpid()}"

# Read dataset
df =
pd.read_csv('datasets/dataset_with_seasonal_vacation_outages_and_vac
ant_houses.csv')
df['date'] = pd.to_datetime(df['date'])
```

```
# Filter by zip code
df_zip = df[df['zip_code'] == ZIP_CODE].copy()
df_zip['year'] = df_zip['date'].dt.year
df_zip['month'] = df_zip['date'].dt.month
df_zip['day'] = df_zip['date'].dt.day

# Prepare training data
train_data = df_zip[df_zip['year'] < 2024]
X_train = train_data[['year', 'month', 'day']]
y_train = train_data['power_consumption']

# Define XGBoost model with fixed hyperparameters
model = xgb.XGBRegressor(
    objective='reg:squarederror',
    tree_method='hist',  # Enable GPU acceleration
    device='cuda',                # Use the first GPU
    learning_rate = 0.09979359710905994,
    max_depth = 4,
    min_child_weight = 9,
    subsample = 0.7225909453820122,
    colsample_bytree = 0.9568932771634039,
    gamma = 0.4520419990324307,
    reg_lambda = 2.1150201222243434,
    reg_alpha = 0.5763612495790174
)

# Fit the model directly
model.fit(X_train, y_train)

def send_real_data_and_metrics(sim_date, prediction):
    """Send real data and ML metrics to master node."""
    actual_value = df_zip[df_zip['date'] ==
sim_date]['power_consumption'].values
    actual_value = float(actual_value[0]) if len(actual_value) > 0
else None

    if actual_value is not None:
        # Store values for ML evaluation
        historical_predictions.append(prediction)
        historical_actuals.append(actual_value)

        # Send real data to master node
        requests.post("http://127.0.0.1:5001/submit_real_data",
json={
            "date": sim_date.strftime('%Y-%m-%d'),
            "zip_code": ZIP_CODE,
            "actual": actual_value
        })

        # Calculate ML metrics only if enough predictions exist
        if len(historical_predictions) > 10:  # Prevent instability
in early predictions
            mae = mean_absolute_error(historical_actuals,
historical_predictions)
            rmse = np.sqrt(mean_squared_error(historical_actuals,
historical_predictions))
```

```
            r2 = r2_score(historical_actuals,
historical_predictions)

            # Send ML metrics to master node
            requests.post("http://127.0.0.1:5001/submit_ml_metrics",
json={
                "zip_code": ZIP_CODE,
                "mae": mae,
                "rmse": rmse,
                "r2": r2
            })

def fetch_speed():
    try:
        response =
requests.get("http://127.0.0.1:5001/get_speed").json()
        return float(response.get("speed", 1.0))
    except requests.exceptions.RequestException:
        return 1.0

def run_simulation():
    sim_date = pd.Timestamp("2000-01-01")
    last_reset_status = False

    while True:
        # Check simulation status
        status = requests.get("http://127.0.0.1:5001/status").json()
        if not status.get("running", False):
            time.sleep(1)
            continue

        # Check reset signal
        reset_response =
requests.post("http://127.0.0.1:5001/get_reset_signal",
json={"node_id": NODE_ID}).json()
        if reset_response.get("reset", False):
            print(f"Node {NODE_ID} detected reset! Restarting
simulation.")
            sim_date = pd.Timestamp("2000-01-01")
            last_reset_status = True
            continue

        speed_factor = fetch_speed()

        X_pred = pd.DataFrame([[sim_date.year, sim_date.month,
sim_date.day]], columns=['year', 'month', 'day'])
        prediction = float(model.predict(X_pred)[0])  # Ensure JSON
serialization

        send_real_data_and_metrics(sim_date, prediction)

        response =
requests.post("http://127.0.0.1:5001/submit_prediction", json={
            "date": sim_date.strftime('%Y-%m-%d'),
            "zip_code": ZIP_CODE,
            "prediction": prediction
```

```
        })
        print(f"Node {NODE_ID} submitted prediction for
{sim_date.strftime('%Y-%m-%d')} at speed {speed_factor}, response:
{response.status_code}")

        sim_date += pd.Timedelta(days=1)

        sleep_time = max(0.01, 1.0 / speed_factor)
        time.sleep(sleep_time)

run_simulation()
```

master_node.py

```
from flask import Flask, request, jsonify

app = Flask(__name__)

global_predictions = []
real_data = []
ml_metrics = {}
simulation_date = None
simulation_running = False
reset_signal = False
nodes_detected_reset = set()
speed_factor = 1
EXPECTED_NODES = 4

@app.route('/submit_prediction', methods=['POST'])
def submit_prediction():
    global simulation_date
    data = request.json
    simulation_date = data["date"]
    global_predictions.append(data)
    return jsonify({"status": "success"}), 200

@app.route('/get_predictions', methods=['GET'])
def get_predictions():
    return jsonify(global_predictions), 200

@app.route('/start', methods=['POST'])
def start_simulation():
    global simulation_running
    simulation_running = True
    return jsonify({"status": "started", "running": True}), 200

@app.route('/pause', methods=['POST'])
def pause_simulation():
    global simulation_running
    simulation_running = False
    return jsonify({"status": "paused", "running": False}), 200

@app.route('/reset', methods=['POST'])
def reset_simulation():
    global global_predictions, simulation_date, simulation_running,
```

```
reset_signal, nodes_detected_reset, real_data, ml_metrics
    global_predictions = []  # Reset predictions
    real_data = []  # Reset actual power consumption data
    ml_metrics = {}  # Reset ML metrics
    simulation_date = None
    simulation_running = False
    reset_signal = True
    nodes_detected_reset.clear()
    return jsonify({"status": "reset"}), 200

@app.route('/get_reset_signal', methods=['POST'])
def get_reset_signal():
    global reset_signal, nodes_detected_reset
    data = request.json
    node_id = data.get("node_id")

    if reset_signal and node_id not in nodes_detected_reset:
        nodes_detected_reset.add(node_id)
        if len(nodes_detected_reset) >= EXPECTED_NODES:
            reset_signal = False
        return jsonify({"reset": True})

    return jsonify({"reset": False})

@app.route('/status', methods=['GET'])
def get_status():
    return jsonify({"running": simulation_running}), 200

@app.route('/get_speed', methods=['GET'])
def get_speed():
    """Returns the current simulation speed."""
    return jsonify({"speed": speed_factor}), 200

@app.route('/set_speed', methods=['POST'])
def set_speed():
    global speed_factor
    data = request.json
    speed_factor = data.get("speed", 1)
    return jsonify({"status": "speed updated", "speed":
speed_factor}), 200

@app.route('/submit_real_data', methods=['POST'])
def submit_real_data():
    """Stores actual power consumption data for comparison."""
    global real_data
    data = request.json
    real_data.append(data)
    return jsonify({"status": "success"}), 200

@app.route('/get_real_data', methods=['GET'])
def get_real_data():
    """Returns the actual recorded power consumption."""
    return jsonify(real_data), 200

@app.route('/submit_ml_metrics', methods=['POST'])
def submit_ml_metrics():
```

```
    """Receives ML evaluation metrics from edge nodes."""
    global ml_metrics
    data = request.json
    zip_code = data.get("zip_code")

    if zip_code:
        ml_metrics[zip_code] = data

    return jsonify({"status": "success"}), 200

@app.route('/get_ml_metrics', methods=['GET'])
def get_ml_metrics():
    """Returns the latest ML metrics."""
    return jsonify(ml_metrics), 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5001, debug=True)
```

dashboard.py

```
import dash
import dash_bootstrap_components as dbc
from dash import dcc, html
from dash.dependencies import Input, Output
import plotly.graph_objects as go
import requests
import pandas as pd

app = dash.Dash(__name__,
external_stylesheets=[dbc.themes.SUPERHERO])

app.layout = dbc.Container([
    dbc.Row([
        dbc.Col(html.H1("Real-Time Power Consumption Forecasting",
className="text-center mb-4"), width=12)
    ]),

    # Control Buttons
    dbc.Row([
        dbc.Col([
            dbc.Button("Start", id="start-button", color="success",
className="me-2"),
            dbc.Button("Pause", id="pause-button", color="danger",
className="me-2"),
            dbc.Button("Reset", id="reset-button", color="warning"),
        ], width=6)
    ], class_name="mb-4"),

    # Speed Control & Simulation Date
    dbc.Row([
        dbc.Col([
            html.Label("Simulation Speed"),
            dcc.Slider(id='speed-slider', min=1, max=100, step=1,
value=1,
                       marks={1: 'Normal', 100: 'Fast'})
```

```
        ], width=6),
        dbc.Col([
            dbc.Card([
                dbc.CardBody([
                    html.H5("Simulation Date", className="card-
title"),
                    html.H4(id="simulation-date", className="text-
center")
                ])
            ], class_name="shadow-lg")
        ], width=6)
    ], class_name="mb-4"),

    # Live Graphs for Predictions and Real Data
    dbc.Row([
        dbc.Col([
            html.H3("Real-Time Power Consumption Predictions"),
            dcc.Graph(id='live-prediction-graph', style={"height":
"500px"})
        ], width=6),
        dbc.Col([
            html.H3("Real-Time Actual Power Consumption"),
            dcc.Graph(id='live-real-data-graph', style={"height":
"500px"})
        ], width=6)
    ], class_name="mb-4"),

    # Metrics Section (Added for MAE, RMSE, R2 Scores)
    dbc.Row([
        dbc.Col([
            html.H5("Mean Absolute Error (MAE):", className="card-
title"),
            html.H6(id='mae-score', children="N/A", className="text-
center")
        ], width=4),
        dbc.Col([
            html.H5("Root Mean Square Error (RMSE):",
className="card-title"),
            html.H6(id='rmse-score', children="N/A",
className="text-center")
        ], width=4),
        dbc.Col([
            html.H5("R² Score:", className="card-title"),
            html.H6(id='r2-score', children="N/A", className="text-
center")
        ], width=4),
    ], class_name="mb-4"),

    # Interval Component for Real-Time Updates
    dcc.Interval(id='interval-component', interval=1000,
n_intervals=0, disabled=True)
], fluid=True)

# Fetch data from master node
def fetch_predictions():
    try:
```

```
        response =
requests.get("http://127.0.0.1:5001/get_predictions")
        if response.status_code == 200:
            return response.json()
    except requests.exceptions.RequestException:
        return []
    return []

def fetch_real_data():
    try:
        response =
requests.get("http://127.0.0.1:5001/get_real_data")
        if response.status_code == 200:
            return response.json()
    except requests.exceptions.RequestException:
        return []
    return []

def fetch_ml_metrics():
    try:
        response =
requests.get("http://127.0.0.1:5001/get_ml_metrics")
        if response.status_code == 200:
            return response.json()
    except requests.exceptions.RequestException:
        return {}

# Fetch simulation speed
def fetch_speed():
    try:
        response = requests.get("http://127.0.0.1:5001/get_speed")
        if response.status_code == 200:
            return response.json().get("speed", 1)
    except requests.exceptions.RequestException:
        return 1

# Update the graph with new data
@app.callback(
    [Output('live-prediction-graph', 'figure'),
     Output('live-real-data-graph', 'figure'),
     Output('simulation-date', 'children')],
    Input('interval-component', 'n_intervals')
)
def update_graph(n):
    predictions = fetch_predictions()
    real_data = fetch_real_data()

    if not predictions or not real_data:
        return go.Figure(), go.Figure(), "No Data"

    # Prepare predictions dataframe
    df_pred = pd.DataFrame(predictions)
    df_pred['date'] = pd.to_datetime(df_pred['date'])
    latest_date = df_pred["date"].max().strftime('%Y-%m-%d')

    # Prepare real data dataframe
```

```
    df_real = pd.DataFrame(real_data)
    df_real['date'] = pd.to_datetime(df_real['date'])

    # Prediction figure
    fig_pred = go.Figure()
    for zip_code in df_pred['zip_code'].unique():
        df_zip = df_pred[df_pred['zip_code'] == zip_code]
        fig_pred.add_trace(go.Scatter(x=df_zip['date'],
y=df_zip['prediction'], mode='lines', name=f'ZIP {zip_code}'))

    fig_pred.update_layout(
        title='Real-Time Power Consumption Predictions',
        xaxis_title='Date', yaxis_title='Power (kWh)',
        template="plotly_dark"
    )

    # Real data figure
    fig_real = go.Figure()
    for zip_code in df_real['zip_code'].unique():
        df_zip = df_real[df_real['zip_code'] == zip_code]
        fig_real.add_trace(go.Scatter(x=df_zip['date'],
y=df_zip['actual'], mode='lines', name=f'ZIP {zip_code}'))

    fig_real.update_layout(
        title='Real-Time Actual Power Consumption',
        xaxis_title='Date', yaxis_title='Power (kWh)',
        template="plotly_dark"
    )

    return fig_pred, fig_real, latest_date

@app.callback(
    [Output('mae-score', 'children'),
     Output('rmse-score', 'children'),
     Output('r2-score', 'children')],
    Input('interval-component', 'n_intervals')
)
def update_metrics(n):
    ml_metrics = fetch_ml_metrics()

    if not ml_metrics:
        return "N/A", "N/A", "N/A"

    # Get average across all ZIP codes
    mae_list = [data['mae'] for data in ml_metrics.values()]
    rmse_list = [data['rmse'] for data in ml_metrics.values()]
    r2_list = [data['r2'] for data in ml_metrics.values()]

    mae = sum(mae_list) / len(mae_list) if mae_list else "N/A"
    rmse = sum(rmse_list) / len(rmse_list) if rmse_list else "N/A"
    r2 = sum(r2_list) / len(r2_list) if r2_list else "N/A"

    return f"{mae:.2f}", f"{rmse:.2f}", f"{r2:.2f}"

# Control simulation state, speed, and reset functionality
@app.callback(
```

```
    Output('interval-component', 'disabled'),
    [Input('start-button', 'n_clicks'),
     Input('pause-button', 'n_clicks'),
     Input('reset-button', 'n_clicks'),
     Input('speed-slider', 'value')]
)
def control_simulation(start, pause, reset, speed):
    ctx = dash.callback_context

    if not ctx.triggered:
        return True

    trigger_id = ctx.triggered[0]['prop_id'].split('.')[0]
    if trigger_id == "start-button":
        requests.post("http://127.0.0.1:5001/start")
        return False
    elif trigger_id == "pause-button":
        requests.post("http://127.0.0.1:5001/pause")
        return True
    elif trigger_id == "reset-button":
        requests.post("http://127.0.0.1:5001/reset")
        return True
    elif trigger_id == "speed-slider":
        requests.post("http://127.0.0.1:5001/set_speed",
json={"speed": speed})

    return False

if __name__ == '__main__':
    app.run_server(debug=True)
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] C. W. Gellings, The smart grid: Enabling energy efficiency and demand response, River Publishers, 2009.

[2] Q. Ho and T. Le-Ngoc, "Smart Grid Communications Networks: Wireless Technologies, Protocols, Issues, and Standards," in *Handbook of Green Information and Communication Systems*, 2013, pp. 115-146.

[3] M. Aryblia, R. Bertram, A. Bolle, A. Corovessi, F. Dembski, D. Fouquet, P. Giňová, K. Księżopolski, N. Mantzaris, J. Ondřich, J. Maćkowiak-Pandera, R. Primova, A. Rüdinger, M. Santini, S. Scheuer, W. Szymalski, J. H. Torres, C. Turmes, T. Tsoutsos and M. Walsh, "ENERGY ATLAS Facts and figures about renewables in Europe," Heinrich Böll Foundation, Friends of the Earth Europe, European Renewable Energies Federation, Green European Foundation, 2018. [Online]. Available: https://gef.eu/wp-content/uploads/2018/04/energyatlas2018_facts-and-figures-renewables-europe.pdf. [Accessed 07 June 2025].

[4] M. Derrick, "Top 10: Benefits of Smart Grids," Energy Digital, 17 April 2024. [Online]. Available: https://energydigital.com/top10/top-10-benefits-of-smart-grids. [Accessed 07 June 2025].

[5] Enel Group, "Smart Grids: what they are, how they work, and their benefits," Enel Group, [Online]. Available: https://www.enel.com/company/energy-services/enel-grids/smart-grid. [Accessed 07 June 2025].

[6] U. B. T., "Smart Grid Technology: Enhancing Efficiency and Reliability," Kiu Publication Extension, September 2024. [Online]. Available: https://www.researchgate.net/publication/383875524_Smart_Grid_Technology_Enhancing_Efficiency_and_Reliability. [Accessed 07 June 2025].

[7] C. Tu, X. He, Z. Shuai and F. Jiang, "Big data issues in smart grid – A review," *Renewable and Sustainable Energy Reviews,* vol. 79, pp. 1099-1107, 2017.

[8] Y. Yan, Y. Qian, H. Sharif and D. Tipper, "A Survey on Smart Grid Communication Infrastructures: Motivations, Requirements and Challenges," *IEEE Communications Surveys & Tutorials,* vol. 15, no. 1, pp. 5-20, 2013.

[9] T. Mitchell, Machine Learning, McGraw Hill, 1997.

[10] N. A. Qarabsh, S. S. Sabry and H. A. Qarabash, "Smart grid in the context of industry 4.0: an overview of communications technologies and challenges," *Indonesian Journal of Electrical Engineering and Computer Science,* vol. 18, no. 2, pp. 656-665, 2020.

[11] M. Al-Kaabi, B. H. A. Igeb and S. Y. Ali, "An Overview of the Smart Grid Attributes, Architecture and Components," *Proceedings of the 2nd International Conference on Emerging Technologies and Intelligent Systems ,* vol. 1, p. 461–471, 2022.

[12] A. Saleem, R. Arshad and D. Khan, "Wide area Protection and Monitoring in Smart-power-grid," *Journal of Electronic Systems,* vol. 6, no. 3, pp. 89-94, 2016.

[13] M. S. Bakare, A. Abdulkarim, M. Zeeshan and A. N. Shuaibu, "A comprehensive overview on demand side energy management towards smart

grids: challenges, solutions, and future direction," *Energy Informatics,* vol. 6, no. 4, 2023.

[14] R. Ma, H.-H. Chen, H.-H. Chen and H.-H. Chen, "Smart Grid Communication: Its Challenges and Opportunities," *IEEE Transactions on Smart Grid,* vol. 4, no. 1, pp. 36-46, 2013.

[15] A. F. E., A. Y. and C. M., "Communication Technologies for Smart Grid: A Comprehensive Survey," *Sensors,* vol. 21, no. 23, 2021.

[16] Data Communications Company, "What are smart meters," Data Communications Company, [Online]. Available: https://www.smartdcc.co.uk/our-smart-network/what-are-smart-meters/. [Accessed 07 June 2025].

[17] M. Alonso, H. Amaris, D. Alcala and D. Florez R., "Smart Sensors for Smart Grid Reliability," *Sensors,* vol. 20, no. 8, 2020.

[18] D. Eckert, " Smart Grid Sensors: Improving Reliability for Distribution Grids During Energy Transition and Grid Modernization," Power Systems Technology, [Online]. Available: https://www.powersystems.technology/community-hub/technical-articles/smart-grid-sensors-improving-reliability-for-distribution-grids-during-energy-transition-and-grid-modernization.html. [Accessed 07 June 2025].

[19] T. Kotsiopoulos, P. Sarigiannidis, D. Ioannidis and D. Tzovaras, "Machine learning and deep learning in smart manufacturing: The smart grid paradigm," *Computer Science Review,* vol. 40, 2021.

[20] B. Achaal, M. Adda, M. Berger, H. Ibrahim and A. Awde, "Study of smart grid cyber-security, examining architectures, communication networks, cyber-attacks, countermeasure techniques, and challenges," *Cybersecurity ,* vol. 7, no. 10, 2024.

[21] L. Yang, H. Xue and F. Li, "Privacy-preserving data sharing in Smart Grid systems," *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm),* pp. 878-883, 2014.

[22] A. Gopstein, C. Nguyen, C. O'Fallon, N. Hastings and D. Wollman, "NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 4.0,," National Institute of Standards and Technology, Gaithersburg, MD, 2024.

[23] L. D. Castro and Joisa Dutra, "Paying for the smart grid," *Energy Economics,* vol. 40, no. 1, pp. S74-S84, 2013.

[24] Y. Zhang, T. Huang and E. Bompard, "Big data analytics in smart grids: a review," *Energy Inform,* vol. 1, no. 8, 2018.

[25] D. Dahiphale, P. Shinde, K. Patil and V. Dahiphale, "Smart Farming: Crop Recommendation using Machine Learning with Challenges and Future Ideas," June 2023. [Online]. Available: https://www.researchgate.net/publication/371589414_Smart_Farming_Crop_Recommendation_using_Machine_Learning_with_Challenges_and_Future_Ideas. [Accessed 07 June 2025].

[26] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[27] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, The MIT Press, 2016.

[28] M. Mohri, A. Rostamizadeh and A. Talwalkar, Foundations of Machine Learning, The MIT Press, 2018.

[29] R. S. Sutton and A. G. Barto, Reinforcement Learning, The MIT Press, 2018.

[30] Q. Yang, Y. Liu, T. Chen and Y. Ton, "Federated Machine Learning: Concept and Applications," *ACM Transactions on Intelligent Systems and Technology (TIST),* vol. 10, no. 2, pp. 1-19, 2019.

[31] H. B. McMahan, E. Moore, D. Ramage, S. Hampson and B. Aguera y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *International Conference on Artificial Intelligence and Statistics*, 2017.

[32] T. Li, A. K. Sahu, A. Talwalkar and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions," *IEEE Signal Processing Magazine,* vol. 37, no. 3, pp. 50-60, 2020.

[33] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawit, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu and S. Zhao, "Advances and Open Problems in Federated Learning," *now,* 2021.

[34] H. Cheng, T. Lu, R. Hao, J. Li and Q. Ai, "Incentive-based demand response optimization method based on federated learning with a focus on user privacy protection," *Applied Energy,* vol. 358, 2024.

[35] F. Wang, L. Jiao, K. Zhu, X. Lin and L. Li, "Toward Sustainable AI: Federated Learning Demand Response in Cloud-Edge Systems via Auctions," *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications,* pp. 1-10, 2023.

[36] B. Aishwarya, K. Shekar, J. S. R. V, N. Singh and H. S. Abdulaali, "Federated Learning for Distributed Optimal Power Flow Solutions in Smart Grids," *2023 International Conference on Power Energy, Environment & Intelligent Control (PEEIC),* pp. 1741-1745, 2023.

[37] Y. Du, N. Mendes, S. Rasouli, J. Mohammadi and P. Moura, " Federated learning assisted distributed energy optimization," *IET Renewable Power Generation,* vol. 18, no. 14, pp. 2524-2538, 2024.

[38] A. Grataloup, S. Jonas and A. Meyer, "A review of federated learning in renewable energy applications: Potential, challenges, and future directions," *Energy and AI,* vol. 17, 2024.

[39] R. Wei, Y. Wang, X. Wang, N. An, X. Liang and Y. Yang, "Federated Learning Based Privacy-Preserving Renewable Energy Consumption for the Local Community," *2023 5th International Conference on Power and Energy Technology (ICPET),* pp. 1594-1599, 2023.

[40] J. J, P. B. M and B. S. S, "Res2-UNeXt Combined with Federated Learning for Cyber-Attack Detection and Classification in Multi Area Smart Grid Power System," in *2024 IEEE Silchar Subsection Conference (SILCON 2024)*, 2024.

[41] N. Moumni, F. Châabane and F. Drira, "Privacy-Preserving Anomaly Detection in Smart Meter Data Via Federated Learning," in *2023 International Conference on Cyberworlds (CW)*, 2023.

[42] T. N. P, R. R. Al-Fatlawy, V. Malathy, N. Kirthiga and E. C. V. S. Mudunuri, "Fault Detection in Smart Grids by Hybrid Generative Adversarial Networks with Neuro Fuzzy Algorithm," in *2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS)*, 2024.

[43] Q. Li and W. Tang, "An Anomaly Detection Method for Smart Power Grid: A Federated Learning Framework," in *2023 6th International Conference on Data Science and Information Technology (DSIT)*, 2023.

[44] J. Chen, T. Gao, R. Si, Y. Dai, Y. Jiang and J. Zhang, "Residential Short Term Load Forecasting Based on Federated Learning," in *IEEE 2nd International Conference on Digital Twins and Parallel Intelligence (DTPI)*, 2022.

[45] B. You, C. Xing, Q. Yu, M. Zhang, Z. He, L. Zeng and C. Liu, "Federated Learning with LSTM Neural Network for Regional Load Forecasting," in *2024 21st International Conference on Harmonics and Quality of Power (ICHQP)*, 2024.

[46] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," University of California, Irvine, 2000.

[47] Hellenic Statistical Authority, "Survey on Energy Consumption in Households, 2011 - 2012," Hellenic Republic, Piraeus, 2013.

[48] ODYSSEE-MURE, "Greece | Energy profile, March 2025," ODYSSEE-MURE, 2025.

[49] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* p. 785–794, 2016.

[50] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye and T.-Y. Liu, "LightGBM: a highly efficient gradient boosting decision tree," *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17),* p. 3149–3157, 2017.

[51] L. Breiman, "Bagging predictors," *Machine Learning,* vol. 24, pp. 123-140, 1996.

[52] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush and A. Gulin, "CatBoost: unbiased boosting with categorical features," *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18),* p. 6639–6649, 2018.

[53] D. Rumelhart, G. Hinton and R. Williams, "Learning representations by back-propagating errors," *Nature ,* vol. 323, p. 533–536, 1986.

[54] A. A. Ghorbani and K. Owrangh, "Stacked generalization in neural networks: generalization on statistically neutral problems," *International Joint Conference on Neural Networks Proceedings (IJCNN'01),* vol. 3, pp. 1715-1720, 2001.

[55] R. J. Hyndman and G. Athanasopoulos, Forecasting: Principles and Practice (3rd ed), OTexts, 2021.

[56] J. Ilemobayo, O. Durodola, O. Alade, O. Awotunde, T. Adewumi, O. Falana, A. Ogungbire, A. Osinuga, D. Ogunbiyi, I. Odezuligbo, O. Edu and A. Ifeanyi, "Hyperparameter Tuning in Machine Learning: A Comprehensive Review," *Journal of Engineering Research and Reports,* vol. 26, pp. 388-95, 2024.

[57] F. Hutter, L. Xu, H. H. Hoos and K. Leyton-Brown, "Algorithm runtime prediction: Methods & evaluation," *Artificial Intelligence,* vol. 206, pp. 79-111, 2014.

[58] T. Akiba, S. Sano, T. Yanase, T. Ohta and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19),* pp. 2623-2631, 2019.

[59] M. Wang, R. Xin, M. Xia, Z. Zuo, Y. Ge, P. Zhang and H. Ye, "A federated LSTM network for load forecasting using multi-source data with homomorphic encryption," *AIMS Energy,* vol. 13, no. 2, pp. 265-289, 2025.

[60] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, O'Reilly Media, Inc., 2019.

[61] Z. Mao, H. Li, Z. Huang, Y. Tian, P. Zhao and Y. Li, "Full Data-Processing Power Load Forecasting Based on Vertical Federated Learning," *Journal of Electrical and Computer Engineering,* 2023.

[62] M. N. Fekri, K. Grolinger and S. Mir, "Asynchronous adaptive federated learning for distributed load forecasting with smart meter data," *International Journal of Electrical Power & Energy Systems,* vol. 153, 2023.

[63] A. Duttagupta, J. Zhao and S. Shreejith, "Exploring Lightweight Federated Learning for Distributed Load Forecasting," *2023 IEEE International Conference on Communications,* pp. 1-6, 2023.

[64] Z. Zhang, S. Rath, J. Xu and T. Xiao, "Federated Learning for Smart Grid: A Survey on Applications and Potential Vulnerabilities," 31 May 2025. [Online]. Available: https://arxiv.org/pdf/2409.10764. [Accessed 07 June 2025].

[65] J. Hao and Y. Tao, "Adversarial attacks on deep learning models in smart grids," *Energy Reports,* vol. 8, no. 2, pp. 123-129, 2022.

[66] Z. Zhou, J. Zhu, F. Yu, X. Li, X. Peng, T. Liu and B. Han, "Model Inversion Attacks: A Survey of Approaches and Countermeasures," 15 November 2024. [Online]. Available: https://arxiv.org/pdf/2411.10023v1. [Accessed 07 June 2025].

[67] M. A. Husnoo, A. Anwar, N. Hosseinzadeh, S. N. Islam, A. N. Mahmood and R. Doss, "A Secure Federated Learning Framework for Residential Short-Term Load Forecasting," *IEEE Transactions on Smart Grid,* vol. 15, no. 2, pp. 2044-2055, 2024.

[68] E. Altamimi, A. Al-Ali, Q. M. Malluhi and A. K. Al-Ali, "Smart grid public datasets: Characteristics and associated applications," *IET Smart Grid,* vol. 7, no. 5, pp. 503 - 530, 2024.

[69] S. Kirmani, A. Mazid, I. A. Khan and M. Abid, "A Survey on IoT-Enabled Smart Grids: Technologies, Architectures, Applications, and Challenges," *Sustainability,* vol. 15, no. 1, 2023.

[70] S. Alam, C. Yakopcic, Q. Wu, M. Barnell, S. Khan and T. M. Taha, "Survey of Deep Learning Accelerators for Edge and Emerging Computing.," *Electronics,* vol. 13, no. 15, 2024.

[71] N. Papanikolaou, N. Tzanis, E. Mylonas, M. Birbas and A. Birbas, "Hardware Accelerators based on wavelets for detection of Transient phenomena in smart grids," *12th International Conference on Modern Circuits and Systems Technologies (MOCAST),* pp. 1-4, 2023.