29 ΙΑΝΟΥΑΡΙΟΥ 2023

ΘΕΜΑΤΑ ΕΠΙΣΤΗΜΗΣ ΔΕΔΟΜΕΜΩΝ

ΤΕΛΙΚΗ ΕΡΓΑΣΙΑ

Συντελεστές εργασίας

Χριστοφορίδης Χαράλαμπος – Π19188

Καρκάνης Ευστράτιος – Π19064

Περιεχόμενα

0. Εισαγωγή

Στο συγκεκριμένο πρόβλημα μηχανικής μάθησης, καλούμαστε να διαχειριστούμε ένα πρόβλημα χρονοσειρών (timeseries problem). Πιο συγκεκριμένα, δοθέντος δεδομένων κατανάλωσης ηλεκτρικής ενέργειας (μέσω έξυπνων μετρητών μέτρησης κατανάλωσης ρεύματος), ως data scientists πρέπει να προβλέψουμε ποια θα είναι η κατανάλωση του ηλεκτρικού ρεύματος **κάθε πελάτη** σε επόμενα χρονικά διαστήματα, δηλαδή σε χρόνους για τους οποίους δεν έχουμε δεδομένα. Η πρόβλεψη κατανάλωσης γίνεται τόσο σε βραχυχρόνια περίοδο (π.χ. μετά από τρεις ώρες), όσο και σε μακροχρόνια περίοδο (π.χ. μετά από τρεις ημέρες).

Το σύνολο δεδομένων, το οποίο χρησιμοποιήσαμε, αποτελείται από μετρήσεις κατανάλωσης ηλεκτρικής ενέργειας 370 πελατών. Κάθε μέτρηση καθενός από τους 370 πελάτες καταγράφεται ανά 15 λεπτά. Το σύνολο δεδομένων **δεν** έχει ελλιπείς τιμές (κενά) και ουσιαστικά αποτελείται από 371 στήλες (η πρώτη στήλη αναφέρει τον χρόνο και οι επόμενες 370 στήλες αναφέρουν την κατανάλωση ρεύματος για κάθε πελάτη).
Σύνδεσμος προς το σύνολο δεδομένων: https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014 .

Σύμφωνα με τα παραπάνω, το πρόβλημα που διατυπώθηκε ανάγεται σε πρόβλημα χρονοσειράς. Ουσιαστικά, οι τιμές κατανάλωσης που καταγράφονται είναι σε σχέση με τον χρόνο. Σε ένα πρόβλημα χρονοσειράς, ο χρόνος θεωρείται βασικό στοιχείο και αποτελεί την ανεξάρτητη μεταβλητή των δεδομένων μας.

1. Προεπεξεργασία των δεδομένων

Αρχικά, το dataset που χρησιμοποιούμε, αφού το κατεβάσαμε, το μετονομάσαμε σε **dataset.txt** . Αυτή η παρατήρηση είναι σημαντική, καθώς μέσα στον κώδικα διαβάζουμε το σύνολο δεδομένων μας με αυτό το όνομα.

1.1 Προετοιμασία

Το πιο σημαντικό τμήμα της μελέτης μας είναι η σωστή προετοιμασία των δεδομένων μας. Η καλή προετοιμασία σημαίνει καλύτερες και πιο σωστές προβλέψεις από τα μοντέλα που θα χρησιμοποιήσουμε στην συνέχεια.

Αφού διαβάσαμε τα δεδομένα σε ένα Dataframe, έγιναν τα ακόλουθα:

1) Αφού αναφερόμαστε σε πρόβλημα χρονοσειρών, η στήλη που αναφέρει την ημερομηνία και ώρα πρέπει να έχει τον κατάλληλο τύπο δεδομένων. Έτσι, μετατρέψαμε τον τύπο δεδομένων της στήλης αυτής από string σε **datetime**.

2) Επιπλέον, οι στήλες που αναφέρουν τις τιμές κατανάλωσης ρεύματος για κάθε πελάτη πλέον έχουν τύπο δεδομένων **δεκαδικό αριθμό** (προηγουμένως ήταν και αυτές οι στήλες αλφαριθμητικά).

3) Κάναμε την στήλη που αναγράφει τον χρόνο την **index** στήλη του συνόλου δεδομένων.

4) Διαιρέσαμε τα δεδομένα κατανάλωσης κάθε πελάτη με τον αριθμό 4. Πλέον, κάθε αριθμός στο σύνολο δεδομένων συμβολίζει την **κατανάλωση σε kWh** και όχι σε kW/15minutes που ήταν προηγουμένως.

5) Ομαδοποιήσαμε τα δεδομένα ανά ώρα. [1] Πλέον, κάθε γραμμή στο σύνολο δεδομένων αναφέρεται σε μία διαφορετική ώρα. Προηγουμένως, κάθε γραμμή περιελάβανε την κατανάλωση ρεύματος ανά τέταρτο, ενώ τώρα κάθε γραμμή αναφέρεται σε κατανάλωση ρεύματος διαφορετικής χρονικής ώρας. Για την ομαδοποίηση αυτή χρησιμοποιήθηκε ο μέσος όρος των τιμών κατανάλωσης. Θεωρούμε ότι με αυτόν τον τρόπο, η μελέτη μας θα είναι ευκολότερη.

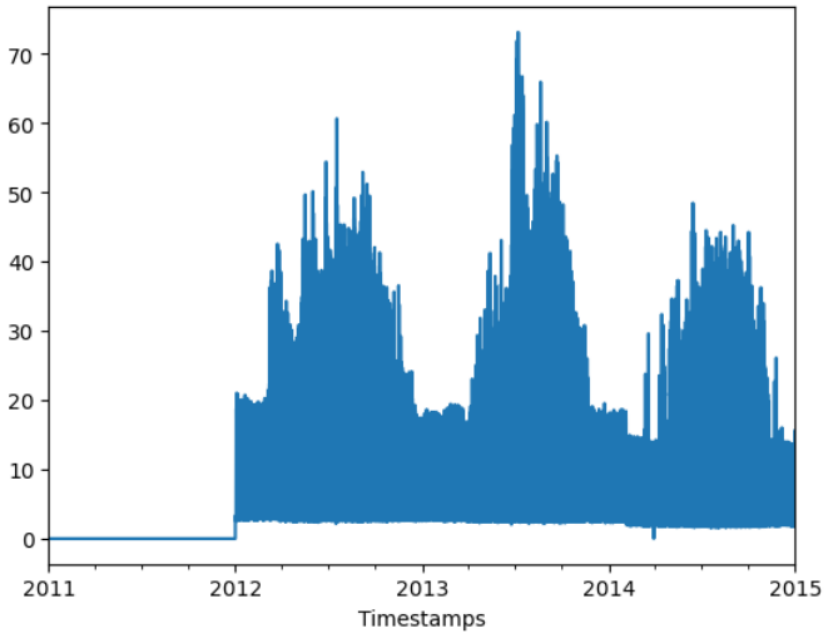
6) Ένα από τα πιο σημαντικά βήματα της προεπεξεργασίας των δεδομένων ήταν και η αποβολή των ακραίων τιμών από το dataframe. Για την

διαδικασία αυτή χρησιμοποιήσαμε τον αλγόριθμο **Isolation Forest**. [2] Μέσα στο dataframe, οι τιμές που θεωρήθηκαν outliers αντικαταστάθηκαν με τον αριθμό -1. Αυτό συνέβη έτσι ώστε να μπορούμε εύκολα να βρούμε τις ακραίες αυτές τιμές και να τις αντικαταστήσουμε με τον μέσο όρο κατανάλωσης κάθε πελάτη. Για παράδειγμα, εάν είχαν βρεθεί 2 outliers στην χρονοσειρά που αντιστοιχεί στον δεύτερο πελάτη, τότε οι ακραίες αυτές τιμές θα αντικαθίσταντο από τον μέσο όρο κατανάλωσης του δεύτερου πελάτη.

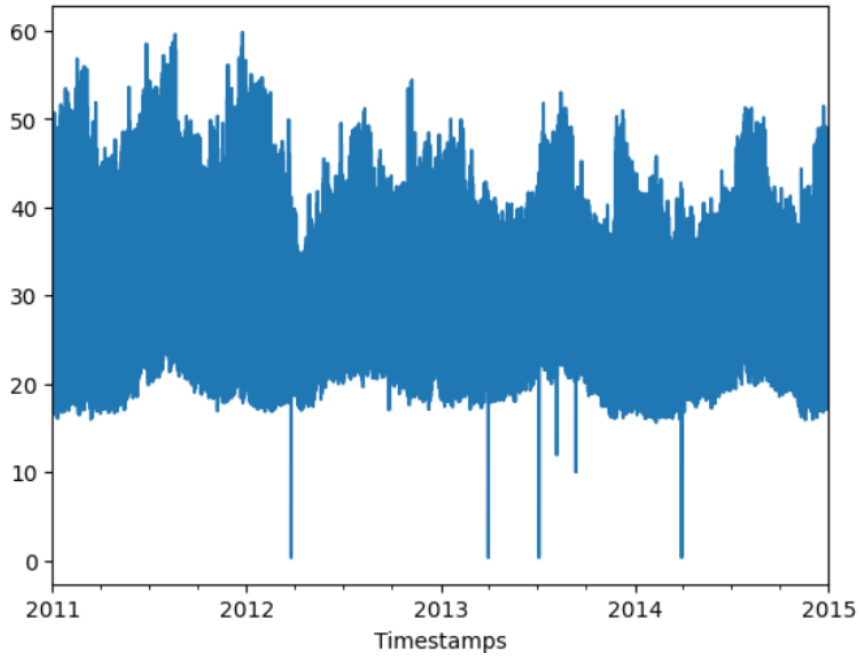
1.2 Κατανομή των τιμών κατανάλωσης

Ένα ιδιαίτερο χαρακτηριστικό που παρουσιάζει το συγκεκριμένο dataset είναι ότι αποτελείται από πολλές χρονοσειρές μαζί. Με άλλα λόγια, δεν είναι ένα κλασσικό σύνολο δεδομένων πολυδιάστατης χρονοσειράς, αλλά αποτελείται από πολλές μονομεταβλητές χρονοσειρές (κάθε πελάτης είναι ανεξάρτητος από έναν άλλο, ωστόσο η κατανάλωση κάθε πελάτη εξαρτάται από τον χρόνο).

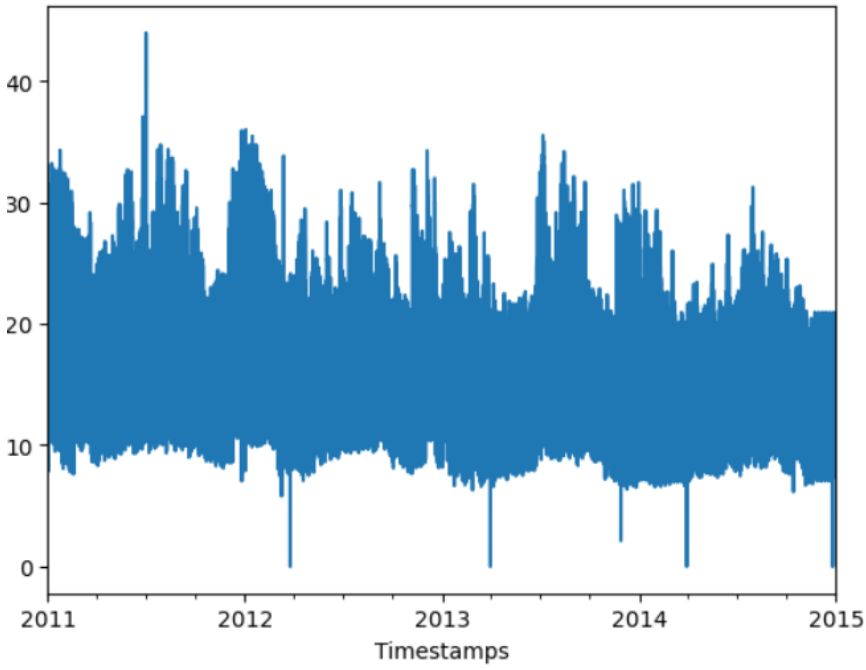
Για να αποτυπώσουμε την κατανομή των δεδομένων του dataset, επιλέξαμε τυχαία τρεις πελάτες (τρεις στήλες) από το dataset, και τις κάναμε plot. Τα αποτελέσματα φαίνονται παρακάτω:



Κατανομή δεδομένων για τον πελάτη MT_100



Κατανομή δεδομένων για τον πελάτη MT_300



Κατανομή δεδομένων για τον πελάτη MT_267

Παρατηρούμε ότι η κατανομή των δεδομένων για κάθε έναν από αυτούς τους πελάτες θυμίζει την κανονική κατανομή κατ' επανάληψη. Επομένως, το σύνολο δεδομένων περιέχει τιμές που ακολουθούν (ως επί το πλείστων) την **περιοδική κανονική κατανομή**.

Οι περισσότερες χρονοσειρές παρουσιάζουν γενικά μία αυξομείωση (δηλαδή έχουν αυξητική τάση, έπειτα χαμηλή τάση κτλ.). Επιπλέον, όσον αφορά τα φαινόμενα seasonality, τα καλοκαίρια η κατανάλωση ρεύματος είναι πιο υψηλή, ενώ τους υπόλοιπους μήνες συμβαίνει μία ύφεση. Αυτό αποδεικνύεται ότι συμβαίνει για όλες τις χρονοσειρές στο **κεφάλαιο 2** αυτής της τεκμηρίωσης. Επομένως, εδώ εμφανίζεται ένα φαινόμενο **seasonality**: κάθε χρονιά, η κατανάλωση ρεύματος αυξάνεται τα καλοκαίρια και μειώνεται τους χειμερινούς μήνες.

1.3 Δειγματοληψία των δεδομένων

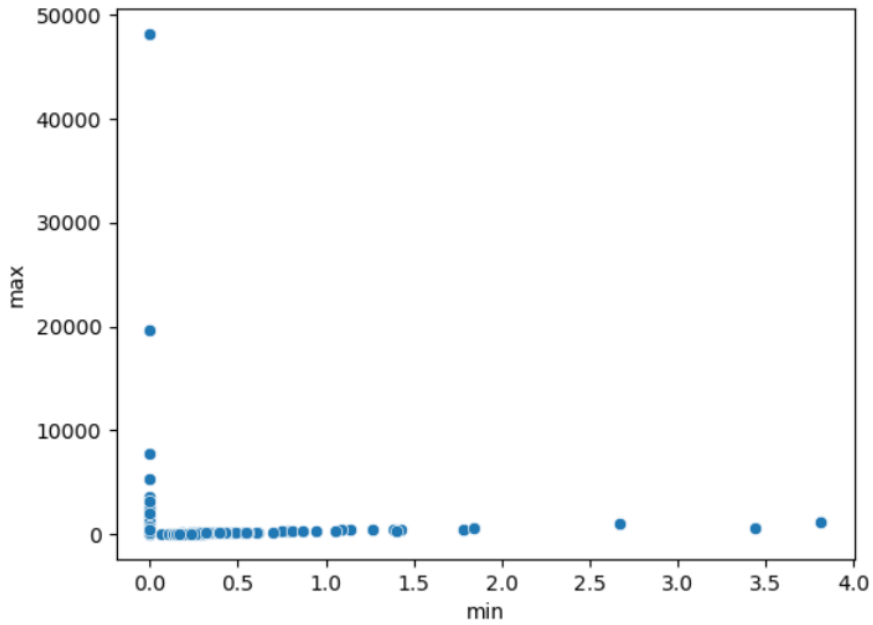
Όπως προαναφέρθηκε, η δειγματοληψία των δεδομένων είναι ανά ένα τέταρτο της ώρας. Ωστόσο, για λόγους ευκολίας και για να μειώσουμε τον αριθμό των εγγραφών, θεωρήσαμε καλύτερο να συγχωνεύσουμε τις γραμμές που αναφέρονται σε κάθε ώρα, χρησιμοποιώντας τον μέσο όρο των τιμών, ώστε να έχουμε μία τελική δειγματοληψία ανά ώρα.

Σημείωση: στο dataset δεν υπάρχουν κενά ή ελλείπεις τιμές.

1.4 Μέγιστες και ελάχιστες τιμές των δεδομένων

Για να αποτυπώσουμε τις μέγιστες και ελάχιστες τιμές πάνω στα δεδομένα, χρησιμοποιήσαμε ένα ενιαίο γράφημα, στο οποίο ο κατακόρυφος άξονας δείχνει τις μέγιστες τιμές ανά πελάτη, ενώ ο οριζόντιος άξονας δείχνει τις ελάχιστες τιμές ανά πελάτη. Σύμφωνα με το παρακάτω γράφημα, οι περισσότεροι πελάτες έχουν σαν ελάχιστη κατανάλωση ενέργειας τις 0 kWh, ενώ οι περισσότεροι δεν υπερβαίνουν τις 10 kWh σε κατανάλωση.

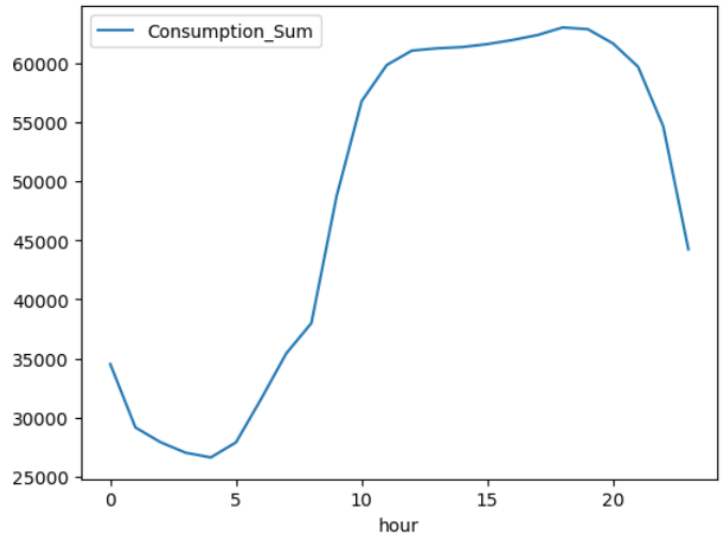
Σημείωση: το γράφημα αυτό έγινε πριν την ομαδοποίηση των δεδομένων ανά μία ώρα. Δηλαδή, στο παρακάτω γράφημα, οι ελάχιστες και μέγιστες τιμές φαίνονται με δειγματοληψία ανά τέταρτο της ώρας. Είτε η δειγματοληψία είναι ανά τέταρτο, είτε ανά ώρα, το αποτέλεσμα του γραφήματος δεν επηρεάζεται σημαντικά.



Γράφημα, στο οποίο αποτυπώνονται οι ελάχιστες τιμές και οι μέγιστες τιμές του συνόλου δεδομένων. Στον οριζόντιο άξονα βρίσκονται οι ελάχιστες τιμές ανά πελάτη, ενώ στον κατακόρυφο άξονα βρίσκονται οι μέγιστες τιμές ανά πελάτη.

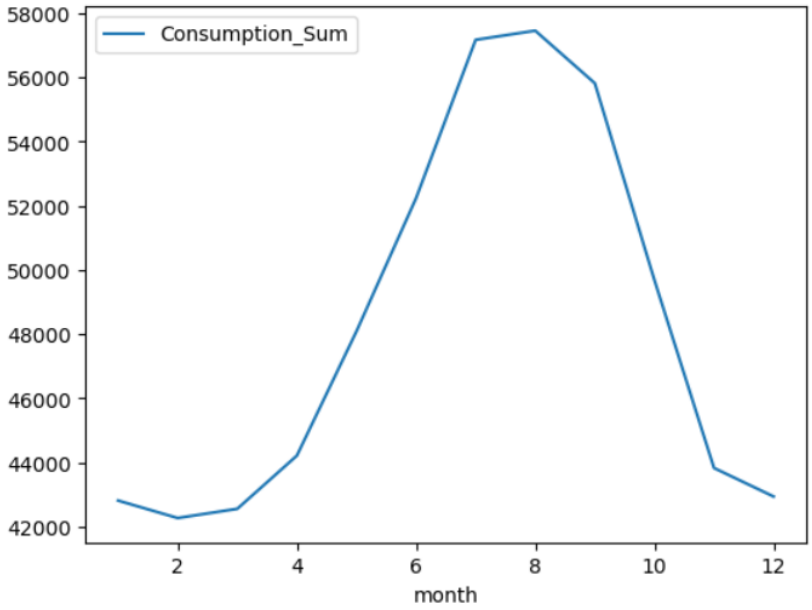
2. Οπτικοποίηση πάνω στα δεδομένα

Για να κατανοήσουμε καλύτερα τα δεδομένα μας, χρησιμοποιήσαμε μία επιπλέον στήλη στο dataset, η οποία ονομάζεται **Consumption_Sum** και περιέχει το άθροισμα κατανάλωσης ενέργειας για κάθε πελάτη ανά ώρα. Χρησιμοποιώντας την στήλη αυτή έχουμε τα επόμενα γραφήματα, τα οποία είναι σημαντικά στην μελέτη μας:



Συνολική κατανάλωση ενέργειας ως προς τις ώρες κάθε ημέρας.

Στο παραπάνω γράφημα παρατηρούμε την συνολική κατανάλωση ενέργειας ανά ώρα. Περισσότερη κατανάλωση έχουμε τις μεσημεριανές και απογευματινές ώρες, ενώ η κατανάλωση «πέφτει» τις πρωινές ώρες και τα ξημερώματα.



Συνολική κατανάλωση ενέργειας ως προς τους μήνες κάθε έτους.

Μία πιο σημαντική πληροφορία που αντλούμε από τα δεδομένα είναι ότι η κατανάλωση ενέργειας τους θερινούς μήνες φτάνει στο απόγειό της, ενώ κατά τους χειμερινούς μήνες δεν έχουμε και τόση κατανάλωση ενέργειας.

3. Ομαδοποίηση των δεδομένων

Στο συγκεκριμένο ερώτημα της εργασίας, μας ζητήθηκε να ομαδοποιήσουμε τα δεδομένα μας, να τα μοιράσουμε δηλαδή σε συστάδες (Clustering). Η εκφώνηση του ερωτήματος μας βοηθά λέγοντάς μας πως αφού έχουμε κάνει μία συζήτηση με έναν ειδικό του χώρου (domain expert), ξέρουμε ότι τα δεδομένα μας προέρχονται από 8 κατηγορίες καταναλωτών ("retail/shopping", "continuous laboring", "weekly laboring", "hotel businesses", "catering industry", "schools", "logistics" και "other").

3.1 Μείωση Διαστάσεων – PCA

Πριν ξεκινήσουμε την διαδικασία της ομαδοποίησης, θα πρέπει να χρησιμοποιήσουμε ένα αλγόριθμο μείωσης διαστάσεων (π.χ. PCA) προκειμένου να διευκολυνθεί τόσο η διαδικασία υπολογισμού αποστάσεων, όσο και η οπτικοποίηση του τελικού αποτελέσματος. Αυτό το προπαρασκευαστικό βήμα είναι απαραίτητο, λόγω του μεγάλου στο πλήθος (ως προς τη διάσταση του χρόνου) εγγραφών. [3][4]

Αφού μειώσαμε τις διαστάσεις παρατηρήσαμε ότι η απώλεια πληροφορίας ήταν ελάχιστη, κάτι που μας επιτρέπει να συνεχίσουμε στο επόμενο βήμα.

Luckily, the information loss is extremely small

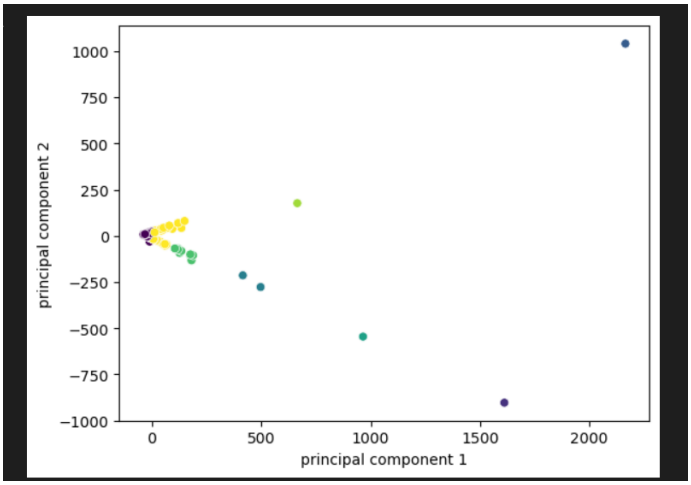
```
# print the amount of information that PCA holds after dimension reduction
print('Explained variation per principal component: {}'.format(pca.explained_variance_ratio_))
```

```
Explained variation per principal component: [0.74756256 0.19292765]
```

Απώλεια πληροφορίας μετά από την εφαρμογή της μεθόδου PCA.

3.2 Ομαδοποίηση k-means με βάση την γνώμη του domain expert

Με βάση την καθοδήγηση του domain expert (8 κατηγορίες χρηστών), μπορούμε εύκολα να διακρίνουμε την ορθή τιμή της μεταβλητής k του αλγορίθμου k-means και έτσι ξεκινάμε την διαδικασία με τα μειωμένα δεδομένα που έχουμε πλέον μετά την εκτέλεση του αλγορίθμου PCA. [5]



k-means clustering για k=8

Η παραπάνω διαδικασία δημιουργεί 8 συστάδες (0 έως 7) και κατατάσσει ανάλογα τα δεδομένα. Αναλυτικότερα, μπορούμε να δούμε τον αριθμό των τιμών σε κάθε ομάδα αλλά και το που ανήκει η κάθε χρονοσειρά - πελάτης:

```
# Print the number of clients that belong in each cluster
clusters.value_counts()
```

```
Cluster
Cluster 0    316
Cluster 7     39
Cluster 5      9
Cluster 3      2
Cluster 1      1
Cluster 2      1
Cluster 4      1
Cluster 6      1
dtype: int64
```

Αριθμός πελατών ανά συστάδα

```
# Print the cluster in which every customer belongs to
names_for_labels = [f"Cluster {label}" for label in labels1]
clusters = pd.DataFrame(zip(data.T.columns,names_for_labels),columns=["Series","Cluster"]).sort_values(by="Cluster").set_index("Series")
clusters
```

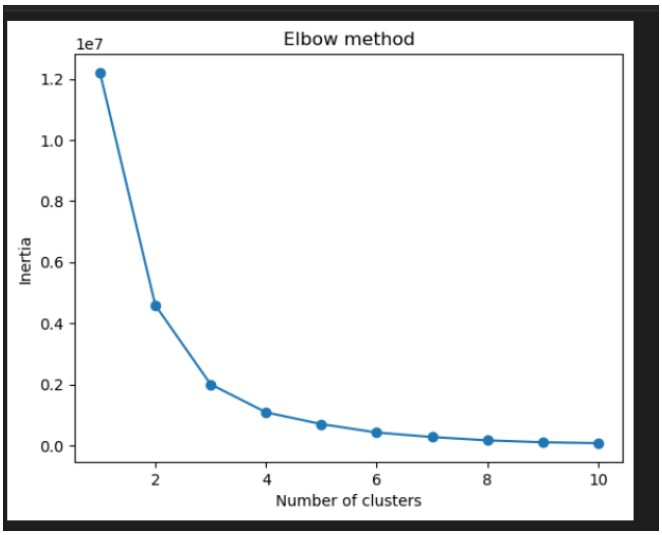
Series	Cluster
MT_001	Cluster 0
MT_244	Cluster 0
MT_243	Cluster 0
MT_242	Cluster 0
MT_240	Cluster 0
...	...
MT_369	Cluster 7
MT_224	Cluster 7
MT_204	Cluster 7
MT_213	Cluster 7
MT_345	Cluster 7

370 rows × 1 columns

Σε ποιο cluster ανήκει κάθε πελάτης

3.3 Εύρεση καλύτερου k για τον αλγόριθμο k-means

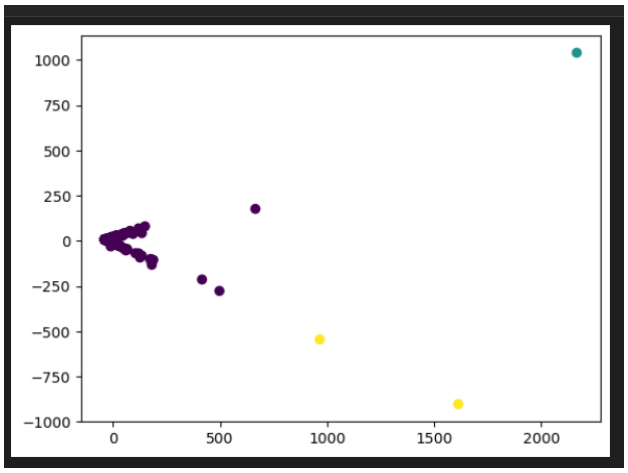
Σε αυτό το ερώτημα, η εκφώνηση μας δίνει την υπόθεση πως δεν έχουμε την πληροφορία του domain expert. Συνεπώς, καλούμαστε να βρούμε οι ίδιοι μία μέθοδο, η οποία μπορεί να μας οδηγήσει στο βέλτιστο k για την εκτέλεση του αλγορίθμου k-means. Εμείς επιλέξαμε να χρησιμοποιήσουμε την **Elbow Method** [6] για να πετύχουμε το στόχο μας:



Χρήση της μεθόδους Elbow για την εύρεση του βέλτιστου k

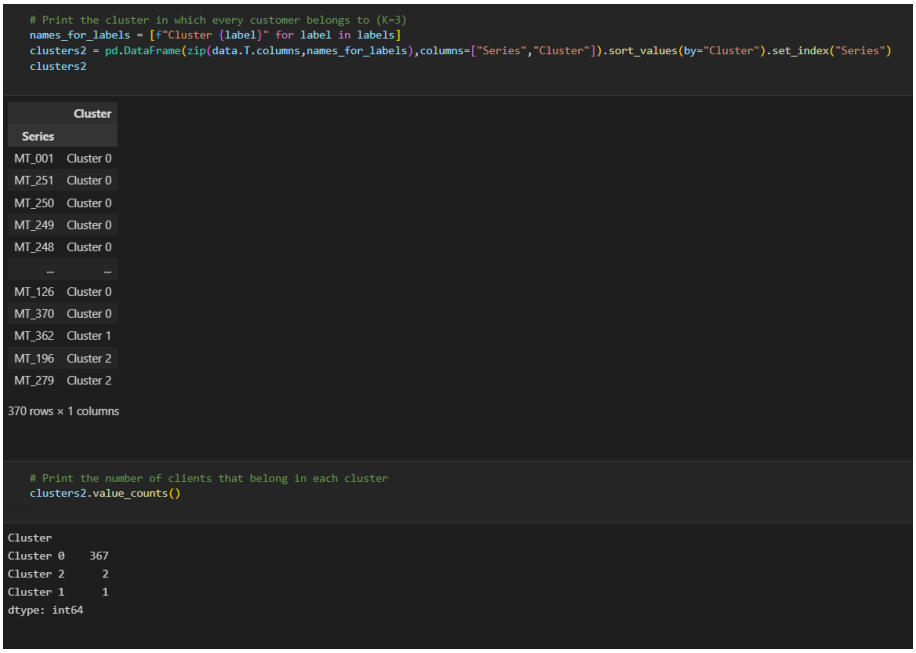
Παρατηρώντας την καμπύλη που παράχθηκε από τη μέθοδο αυτή, μπορούμε εύκολα να παρατηρήσουμε ότι το βέλτιστο k που μας υποδεικνύεται είναι το k = 3 μιας και εκεί είναι που η καμπύλη σχηματίζει κλίση (σαν αγκώνα του ανθρώπινου χεριού), ε’ ξου και το όνομά της.

Εκτελούμε ξανά τον αλγόριθμο k-means με τιμή k = 3 :



Ομαδοποίηση για k = 3

Και παίρνουμε και πάλι τα αντίστοιχα αποτελέσματα:



```
# Print the cluster in which every customer belongs to (K=3)
names_for_labels = [f"Cluster {label}" for label in labels]
clusters2 = pd.DataFrame(zip(data.T.columns,names_for_labels),columns=["Series","Cluster"]).sort_values(by="Cluster").set_index("Series")
clusters2
```

Series	Cluster
MT_001	Cluster 0
MT_251	Cluster 0
MT_250	Cluster 0
MT_249	Cluster 0
MT_248	Cluster 0
...	...
MT_126	Cluster 0
MT_370	Cluster 0
MT_362	Cluster 1
MT_196	Cluster 2
MT_279	Cluster 2

370 rows × 1 columns

```
# Print the number of clients that belong in each cluster
clusters2.value_counts()
```

```
Cluster
Cluster 0    367
Cluster 2      2
Cluster 1      1
dtype: int64
```

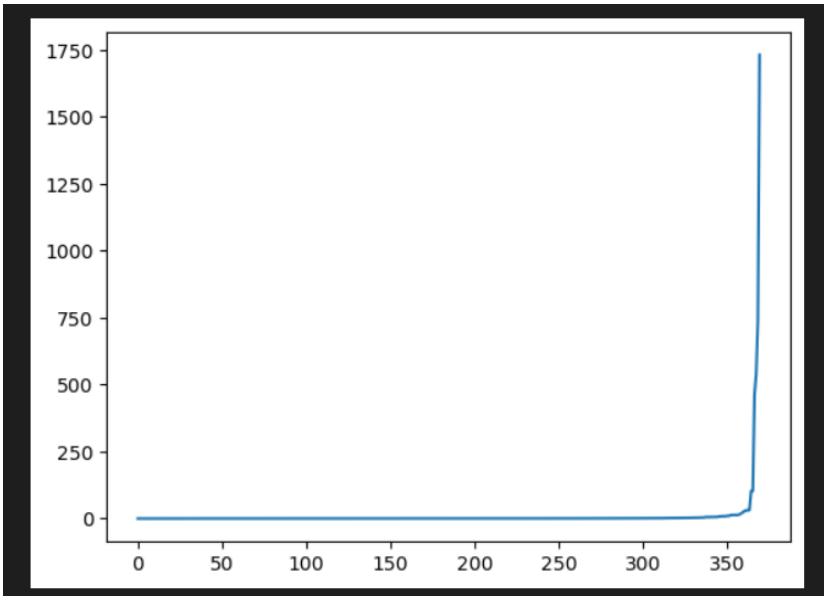
Πλήθος πελατών που ανήκουν σε μία ομάδα καθώς επίσης και ποιοι είναι αυτοί οι πελάτες

Αυτή τη φορά τα αποτελέσματα δεν μας ικανοποιούν, καθώς για k = 3 ο αλγόριθμος ομαδοποίησε σχεδόν όλα τα δεδομένα μας στην συστάδα 0 και μονάχα 3 δεδομένα πήγαν στις άλλες δύο. Ως αποτέλεσμα, θεωρούμε ότι μόνο για k = 8 μας δίνει ορθά αποτελέσματα ο αλγόριθμος.

3.4 Χρήση άλλου αλγορίθμου Clustering - DBSCAN

Σε αυτό το ερώτημα μας υποδεικνύεται από την εκφώνηση να χρησιμοποιήσουμε και έναν αλγόριθμο για density-based clustering, καθώς ο αλγόριθμος k-means έχει το μειονέκτημα πως βρίσκει μόνο σφαιροειδείς συστάδες. Γι’ αυτό το λόγο αποφασίσαμε να χρησιμοποιήσουμε τον αλγόριθμο DBSCAN. [7]

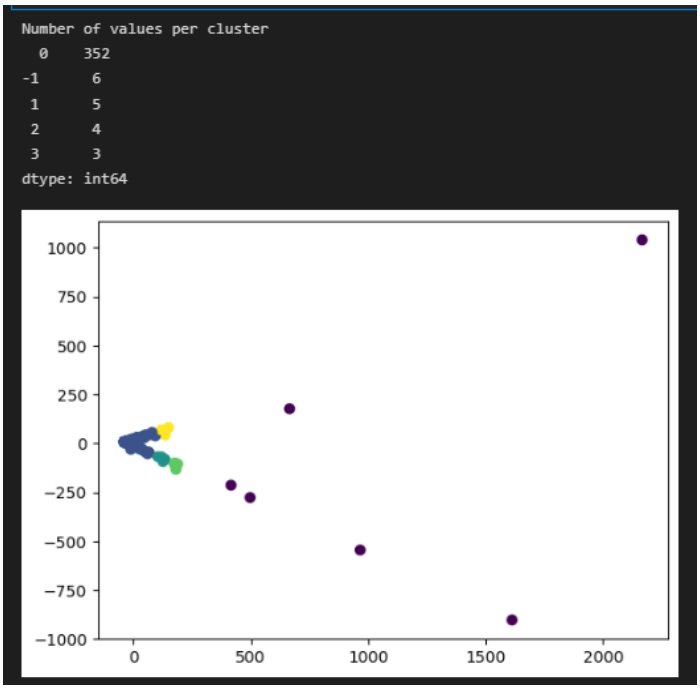
Ο συγκεκριμένος αλγόριθμος, για να λειτουργήσει, ζητάει από τον χρήστη να του προσδιορίσει την απόσταση που πρέπει να έχουν δύο σημεία για να τα ταξινομήσει στην ίδια ομάδα(eps ή e), συνεπώς θα χρησιμοποιήσουμε το μοντέλο **NearestNeighbors** [8]**,** για να μας προσδιορίσει την βέλτιστη τιμή της απόστασης αυτής στα δεδομένα μας.



Προσδιορισμός βέλτιστης τιμής για την παράμετρο e του αλγορίθμου DBSCAN.

Κοιτώντας το γράφημα που μας παρήγαγε η μέθοδος μπορούμε εύκολα να καταλάβουμε πως το βέλτιστο είναι e = 40.

Θα τρέξουμε τον αλγόριθμο ‘’DBSCAN’’ για e = 40:



Αποτελέσματα DBSCAN αλγορίθμου όταν το e = 40

3.5 Σύγκριση και παρατηρήσεις αποτελεσμάτων των δύο αλγορίθμων (k-means και DBSCAN)

Οι παρατηρήσεις που έχουν γίνει, βάση των αποτελεσμάτων, είναι οι παρακάτω:

- Οι αλγόριθμοι K-means (για k=8) και DBSCAN (για e=40) έχουν σχεδόν παρόμοια αποτελέσματα με μικρές διαφοροποιήσεις.

- Ο αλγόριθμος K-means (για k = 3) δεν παράγει σωστά αποτελέσματα ομαδοποιώντας σχεδόν όλα τα δεδομένα στην ίδια συστάδα καθιστώντας την λειτουργία του άχρηστη.

- Δεν παρατηρήσαμε κάποια ουσιαστική διαφορά στην επιλογή K-means (για k=8) και DBSCAN (για e=40) για τα συγκεκριμένα δεδομένα. Υποθέτουμε πως κάτι τέτοιο οφείλεται στην ίδια την φύση του συνόλου των δεδομένων.

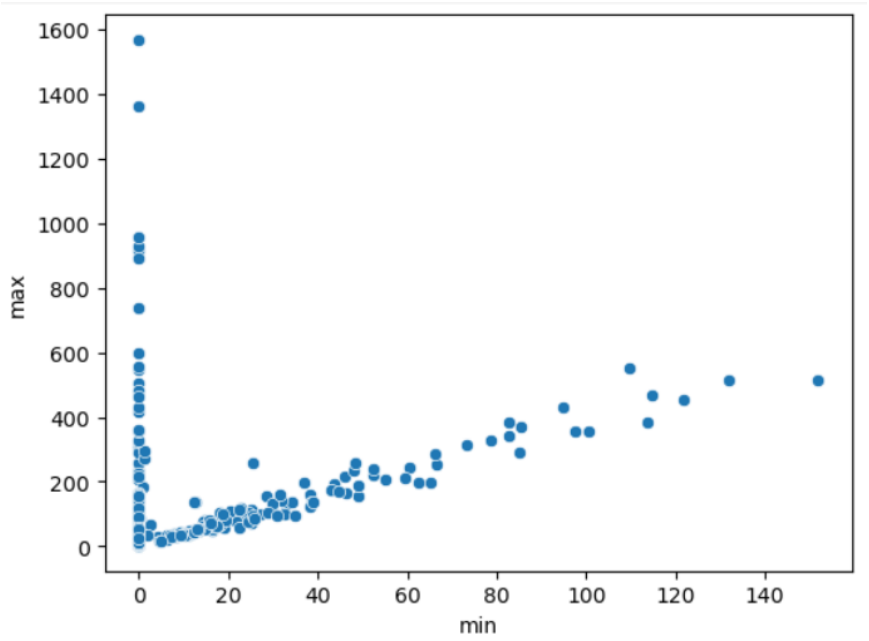
4. Αλλαγές στο σύνολο δεδομένων πριν την είσοδό τους στα μοντέλα μηχανικής μάθησης

Στο βήμα αυτό περιγράφουμε τις διαδικασίες που κάναμε στο σύνολο δεδομένων, προκειμένου να το προετοιμάσουμε κατάλληλα για τα επόμενα δύο μοντέλα μηχανικής μάθησης που χρησιμοποιήσαμε για την πρόβλεψη.

4.1 Επιλογή και επεξεργασία των δεδομένων

Τα δεδομένα που θα χρησιμοποιήσουμε για την διαδικασία της πρόβλεψης είναι <u>όλοι οι πελάτες που ομαδοποιήθηκαν σύμφωνα με τον αλγόριθμο DBSCAN στην συστάδα με αριθμό 0</u>. Όλοι αυτοί οι πελάτες παρουσιάζουν κάποια κοινά χαρακτηριστικά, δηλαδή έχουν κοινές συμπεριφορές ως προς την κατανάλωση ρεύματος. Για αυτόν τον λόγο έχουν ομαδοποιηθεί στην ίδια συστάδα από τον αλγόριθμο DBSCAN.

Πιο συγκεκριμένα, σύμφωνα με το επόμενο γράφημα, οι καταναλώσεις ρεύματος των πελατών είναι στο διάστημα [0 kWh, 1600 kWh), δηλαδή η κατανάλωση ενέργειας είναι σχετικά χαμηλή.



Γράφημα, στο οποίο αποτυπώνονται οι ελάχιστες τιμές και οι μέγιστες τιμές του συνόλου δεδομένων. Στον οριζόντιο άξονα βρίσκονται οι ελάχιστες τιμές ανά πελάτη, ενώ στον κατακόρυφο άξονα βρίσκονται οι μέγιστες τιμές ανά πελάτη.

Ακόμα, ένα πολύ σημαντικό κομμάτι της μελέτης μας είναι να κανονικοποιήσουμε τις τιμές που ανήκουν σε αυτό το «νέο» σύνολο δεδομένων, ώστε όλες οι τιμές να έχουν το ίδιο εύρος τιμών. Με αυτόν τον τρόπο, οι προβλέψεις και τα αποτελέσματα που θα λάβουμε από τα μοντέλα μας θα είναι ορθότερες και ακριβέστερες (δεν θα υπάρχει δηλαδή κάποιο bias). Η κανονικοποίηση αυτή έγινε με την μέθοδο Min-Max Normalization .

4.2 Επαύξηση των δεδομένων

Εφόσον το σύνολο των 352 πελατών που θα χρησιμοποιήσουμε εντέλει είναι μικρό ως προς το πλήθος, καλούμαστε να αυξήσουμε τον αριθμό των χρονοσειρών. Αυτό ουσιαστικά το ολοκληρώσαμε με την βοήθεια των **ημι-επικαλυπτόμενων παραθύρων**. [9]

Πιο συγκεκριμένα, χρησιμοποιώντας μία χρονοσειρά κάθε φορά (τις τιμές ενός μόνο πελάτη κάθε φορά) παράγουμε μία νέα χρονοσειρά της οποίας κάθε τιμή προκύπτει από τον μέσο όρο των τιμών της επιλεγμένης χρονοσειράς. Οι δύο τιμές που χρησιμοποιούνται κάθε φορά ανήκουν στο «παράθυρο», το οποίο σε κάθε βήμα ολισθαίνει κατά μία θέση πιο πέρα.

Κάθε νέα χρονοσειρά που παράγεται είναι ουσιαστικά και ένας νέος πελάτης. Το πλεονέκτημα εδώ πέρα είναι ότι όλες οι νέες χρονοσειρές

«μοιάζουν» με τις αρχικές 352, εφόσον χρησιμοποιείται ο μέσος όρος κάθε φορά των τιμών των χρονοσειρών. Επομένως, οι συμπεριφορές των νέων χρονοσειρών είναι παρόμοιες με τις συμπεριφορές των αρχικών χρονοσειρών.

Ως αποτέλεσμα, πλέον έχουμε 352*2=704 πελάτες (χρονοσειρές) στο σύνολο δεδομένων μας.

Στη συνέχεια, όλες οι null τιμές που δημιουργούνται στο dataset αντικαθίστανται με τον αριθμό μηδέν (0), υπονοώντας έτσι ότι ο εκάστοτε πελάτης είχε μηδενική κατανάλωση ηλεκτρικής ενέργειας.

4.3 Εύρεση των features και labels στο dataset

Πριν την είσοδο των δεδομένων στους αλγορίθμους μηχανικής μάθησης, πρέπει να επιλέξουμε ποιες στήλες θα περιέχουν δεδομένα που θα είναι γνωστά στον αλγόριθμο (features) και ποιες στήλες είναι αυτές που καλείται ο αλγόριθμος να προβλέψει, περιέχουν δηλαδή άγνωστα δεδομένα κατά το στάδιο της πρόβλεψης (label).

Επομένως, εφόσον το dataset περιέχει 704 πελάτες (έπειτα από την επαύξηση των δεδομένων), και είναι δύσκολο να κάνουμε πρόβλεψη για κάθε μία χρονοσειρά ξεχωριστά (λόγω χρόνου), αποφασίστηκε να δημιουργήσουμε μία νέα στήλη «**Consumption_Sum**», όπως προαναφέρθηκε, η οποία περιέχει το άθροισμα κατανάλωσης κάθε πελάτη ανά συγκεκριμένη χρονοσφραγίδα. Πλέον, οι διαστάσεις του dataset έχουν μειωθεί από 704 στήλες σε μόλις 1, μετατρέποντας έτσι το πρόβλημα σε πρόβλεψη μονομεταβλητής χρονοσειράς. Η στήλη αυτή αποτελεί την μόνη label στήλη του συνόλου δεδομένων.

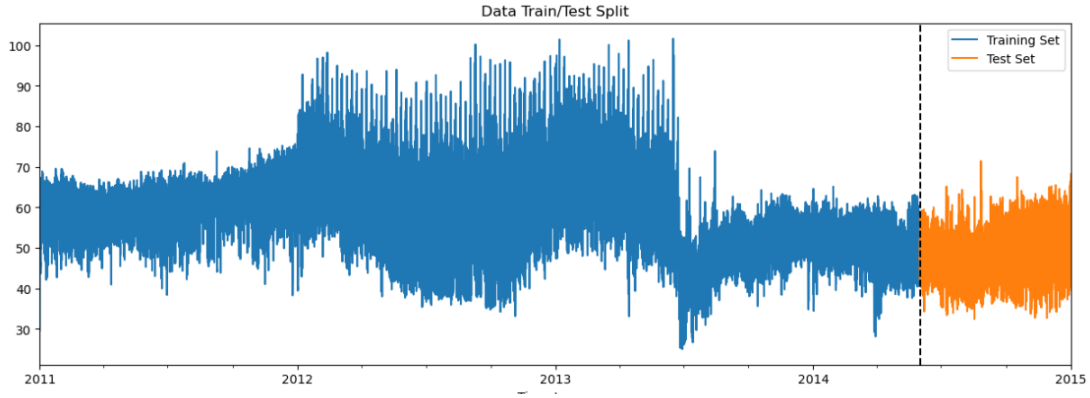
Από την άλλη, τα **features** αποφασίσαμε να είναι κάποια χαρακτηριστικά της εκάστοτε χρονοσφραγίδας. Επομένως, δημιουργήσαμε τις επιπλέον στήλες **month**, **day**, **hour** και **year**, για να συμβάλλουν κατά το στάδιο της εκπαίδευσης των αλγορίθμων. [10]

4.4 Χωρισμός των δεδομένων σε train και test sets

Ο χωρισμός των δεδομένων σε δεδομένα εκπαίδευσης και δεδομένα ελέγχου είναι μία εύκολη διαδικασία. Αποφασίστηκε, τα δεδομένα εκπαίδευσης να είναι όλες οι εγγραφές στο διάστημα **[01/01/2011,**

01/06/2014). Επομένως, όλα τα υπόλοιπα δεδομένα στο διάστημα **[01/06/2014 έως 01/01/2015]** είναι τα δεδομένα ελέγχου.

Παρακάτω, δίνεται ένα γράφημα που περιγράφει αυτόν τον διαχωρισμό:



Δεδομένα εκπαίδευσης και ελέγχου

5. Μοντέλα μηχανικής μάθησης

Στο συγκεκριμένο ερώτημα καλούμαστε να δημιουργήσουμε δύο μοντέλα μηχανικής μάθησης, τα οποία θα χρησιμοποιηθούν στις μεταγενέστερες προβλέψεις μας. Ως είσοδο για τα συγκεκριμένα μοντέλα θα χρησιμοποιήσουμε τα δεδομένα που έχουμε προ επεξεργαστεί καταλλήλως στα προηγούμενα ερωτήματα. Επειδή το πρόβλημα που έχουμε να επιλύσουμε είναι ένα **πρόβλημα regression**, καθώς καλούμαστε να προβλέψουμε μελλοντικές τιμές κατανάλωσης, τα δύο μοντέλα που αναλύουμε επιτρέπουν μία τέτοια μελέτη. Επιλέξαμε να χρησιμοποιήσουμε τα μοντέλα «**XGboost**» και «Facebook Prophet» ή απλά «**Prophet**».

5.1 Πρώτο μοντέλο (XGboost)

Το XGboost είναι το πρώτο μοντέλο που δοκιμάσαμε. [10] Οι παράμετροι που απαιτεί για την λειτουργία του βρέθηκαν μέσω της διαδικασίας «trial and error», προβήκαμε δηλαδή σε όσες δοκιμές, με διαφορετικές τιμές χρειαζόντουσαν για να βρούμε ικανοποιητικές τιμές για τις μεταβλητές (κρίνοντας πάντα από το αποτέλεσμα).

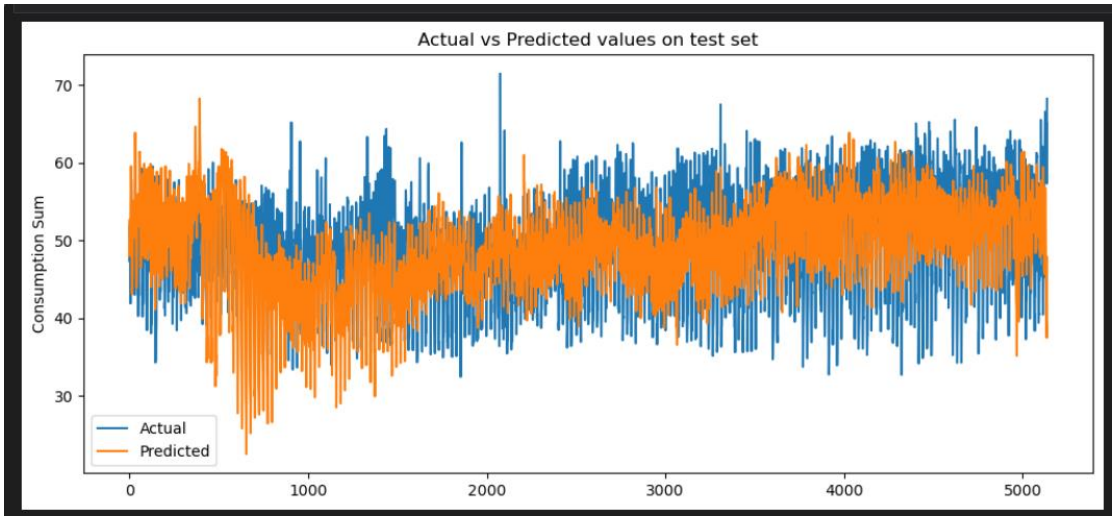
```python
# We use a regressor as our first model
reg = xgb.XGBRegressor(n_estimators=10000,
                       early_stopping_rounds=20,
                       objective='reg:squarederror',
                       learning_rate=0.01)

# Train the first model (use of validation set)
reg.fit(X_train, y_train,eval_set=[(X_train, y_train)],verbose=100)
```

```
Output exceeds the size limit. Open the full output data in a text editor
[0]     validation_0-rmse:57.82689
[100]   validation_0-rmse:21.76530
[200]   validation_0-rmse:9.20590
[300]   validation_0-rmse:5.54102
[400]   validation_0-rmse:4.73155
[500]   validation_0-rmse:4.53426
[600]   validation_0-rmse:4.45197
[700]   validation_0-rmse:4.39387
[800]   validation_0-rmse:4.34142
[900]   validation_0-rmse:4.29379
[1000]  validation_0-rmse:4.24722
[1100]  validation_0-rmse:4.20670
[1200]  validation_0-rmse:4.17062
[1300]  validation_0-rmse:4.13068
[1400]  validation_0-rmse:4.09921
[1500]  validation_0-rmse:4.07231
[1600]  validation_0-rmse:4.04920
[1700]  validation_0-rmse:4.02797
[1800]  validation_0-rmse:4.00132
[1900]  validation_0-rmse:3.97235
[2000]  validation_0-rmse:3.94888
[2100]  validation_0-rmse:3.92755
[2200]  validation_0-rmse:3.89969
[2300]  validation_0-rmse:3.87982
[2400]  validation_0-rmse:3.86451
...
[9700]  validation_0-rmse:3.04690
```

Εκπαίδευση του μηχανισμού XGboost

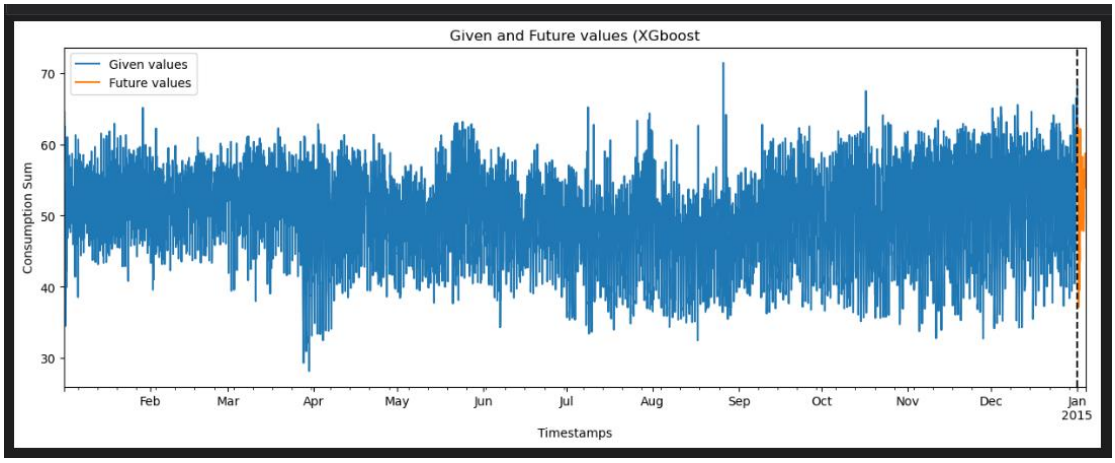
Ύστερα, χρησιμοποιώντας το εκπαιδευμένο μοντέλο, προβήκαμε στην διαδικασία των προβλέψεων τις οποίες και οπτικοποιήσαμε για να τις καταλάβουμε.



Διάγραμμα που δείχνει τις πραγματικές τιμές (με μπλε χρώμα) και τις προβλέψεις που έγιναν (με πορτοκαλί χρώμα)

Τέλος, κάναμε τις ανάλογες τροποποιήσεις για να κάνουμε το μοντέλο να μπορεί να κάνει μακροπρόθεσμες και βραχυπρόθεσμες προβλέψεις, τις οποίες και οπτικοποιήσαμε.

Για να γίνει αυτό, δημιουργήθηκε ένα νέο dataframe με τα ίδια features και labels που περιεγράφηκαν παραπάνω, το οποίο περιέχει ημερομηνίες ανά ώρα έως και 03/01/2015 (τρεις ημέρες μετά από τα δεδομένα που μας δίνονται ήδη). Επομένως, κάναμε predict (για τα καινούρια features) το ήδη εκπαιδευμένο αντικείμενο του XGboost, παίρνοντας έτσι μακροπρόθεσμες και βραχυπρόθεσμες προβλέψεις.



Στο εν λόγω γράφημα, παρατηρούνται με πορτοκαλί χρώμα οι προβλέψεις που έγιναν από το μοντέλο XGboost για έως και τρεις ημέρες μετά.

5.2 Δεύτερο μοντέλο (Facebook Prophet)

Το Facebook Prophet ή Prophet είναι το δεύτερο μοντέλο που δοκιμάσαμε [11]. Οι παράμετροι που απαιτεί για την λειτουργία του βρέθηκαν μέσω της διαδικασίας «trial and error», προβήκαμε δηλαδή σε όσες δοκιμές, με διαφορετικές τιμές χρειάζονταν, για να βρούμε ικανοποιητικές τιμές για τις μεταβλητές (κρίνοντας πάντα από το αποτέλεσμα).

Στο συγκεκριμένο μοντέλο υπάρχουν και οι παρακάτω ιδιαιτερότητες (σε σχέση με το XGboost):

- απαιτεί τη μετονομασία των στηλών του συνόλου δεδομένων για να μπορέσει να λειτουργήσει. Την στήλη «Timestamps» την

μετονομάσαμε σε «ds» και την στήλη «Consumption_Sum» σε «y»:

```
# Prophet need out DataFrame to have specific column names, in order to be functional
ModelData_2 = ModelData['Consumption_Sum'].reset_index()

# Rename Columns
ModelData_2 = ModelData_2.rename(columns={'Timestamps': 'ds','Consumption_Sum': 'y'})

# Print final dataset (modified)
ModelData_2.head()
```

	ds	y
0	2011-01-01 00:00:00	29.606851
1	2011-01-01 01:00:00	58.296219
2	2011-01-01 02:00:00	59.041983
3	2011-01-01 03:00:00	59.266659
4	2011-01-01 04:00:00	60.044196

Μετονομασία στηλών

- Για την εκπαίδευσή του δεν χρησιμοποιεί κάποιο υποσύνολο εκπαίδευσης (train) του συνόλου δεδομένων, αλλά όλο το dataset:

```
# Create the second model
prophet_model = Prophet(interval_width=0.95)
```

Train the second ML model:

```
# Train the second model
prophet_model.fit(ModelData_2)
```

```
12:50:52 - cmdstanpy - INFO - Chain [1] start processing
12:52:07 - cmdstanpy - INFO - Chain [1] done processing

<prophet.forecaster.Prophet at 0x206e77b34f0>
```

Δημιουργία και εκπαίδευση του μοντέλου.

Ύστερα, χρησιμοποιώντας το εκπαιδευμένο μοντέλο ξεκινήσαμε την διαδικασία των προβλέψεων. Το συγκεκριμένο μοντέλο, για να κάνει προβλέψεις απαιτεί τη δημιουργία ενός καινούριου Dataframe, το οποίο όπως

ορίζει και το όνομά του στην παρακάτω εικόνα, περιέχει τις προβλέψεις για μελλοντικές ημερομηνίες (‘’Timestamps’’):

Make predictions (Prophet model):

```
# Create future dates (up to 3 days next) for which we want predictions
future_dates = prophet_model.make_future_dataframe(periods=71,freq='H')
future_dates.head()
```

	ds
0	2011-01-01 00:00:00
1	2011-01-01 01:00:00
2	2011-01-01 02:00:00
3	2011-01-01 03:00:00
4	2011-01-01 04:00:00

Dataframe, στο οποίο θα εισαχθούν οι προβλέψεις του μηχανισμού Prophet.

Τέλος, κάναμε τις ανάλογες τροποποιήσεις για να κάνουμε το μοντέλο να μπορεί να κάνει μακροπρόθεσμες και βραχυπρόθεσμες προβλέψεις:

yhat column contains the predicted values that the model gave us

```
# make predictions using Prophet model
forecast = prophet_model.predict(future_dates)
forecast[['ds', 'yhat']].tail()
```

	ds	yhat
35131	2015-01-03 19:00:00	52.282362
35132	2015-01-03 20:00:00	53.762100
35133	2015-01-03 21:00:00	54.469645
35134	2015-01-03 22:00:00	53.765863
35135	2015-01-03 23:00:00	51.940444

Η στήλη «yhat» περιέχει τις προβλέψεις (συνολική κατανάλωση ενέργειας) για έως και τρεις ημέρες μετά.

5.3 MAPE score των δύο μοντέλων

Σε αυτό το ερώτημα κληθήκαμε, για κάθε μοντέλο που δημιουργήσαμε στο προηγούμενο βήμα, να συγκρίνουμε τις επιδόσεις τους στις προβλέψεις τους πάνω στο συνόλου εκπαίδευσης. Αποφασίσαμε να χρησιμοποιήσουμε την μετρική **MAPE** (Mean of all Absolute Percentage Errors) [12] και πρέπει να αποφανθούμε ποιο μοντέλο ενδείκνυται για κάθε περίπτωση πρόβλεψης.

Αρχικά υπολογίσαμε το MAPE και για τα δύο μοντέλα:

```
Estimate Score of first ML model

    MAPE_XGboost = mean_absolute_percentage_error(y_test, predictions_XGboost)
    print("The mean of all absolute percentage errors between the predicted and actual values is (%): ",MAPE_XGboost)

The mean of all absolute percentage errors between the predicted and actual values is (%):  0.11888439483459955
```

MAPE score του μοντέλου XGboost

```
Print MAPE score of prophet ML model:

    MAPE_prophet = mean_absolute_percentage_error(ModelData_2['y'], forecast['yhat'].head(35065))
    print("The mean of all absolute percentage errors between the predicted and actual values is (%): ",MAPE_prophet)

The mean of all absolute percentage errors between the predicted and actual values is (%):  0.10492044761595487
```

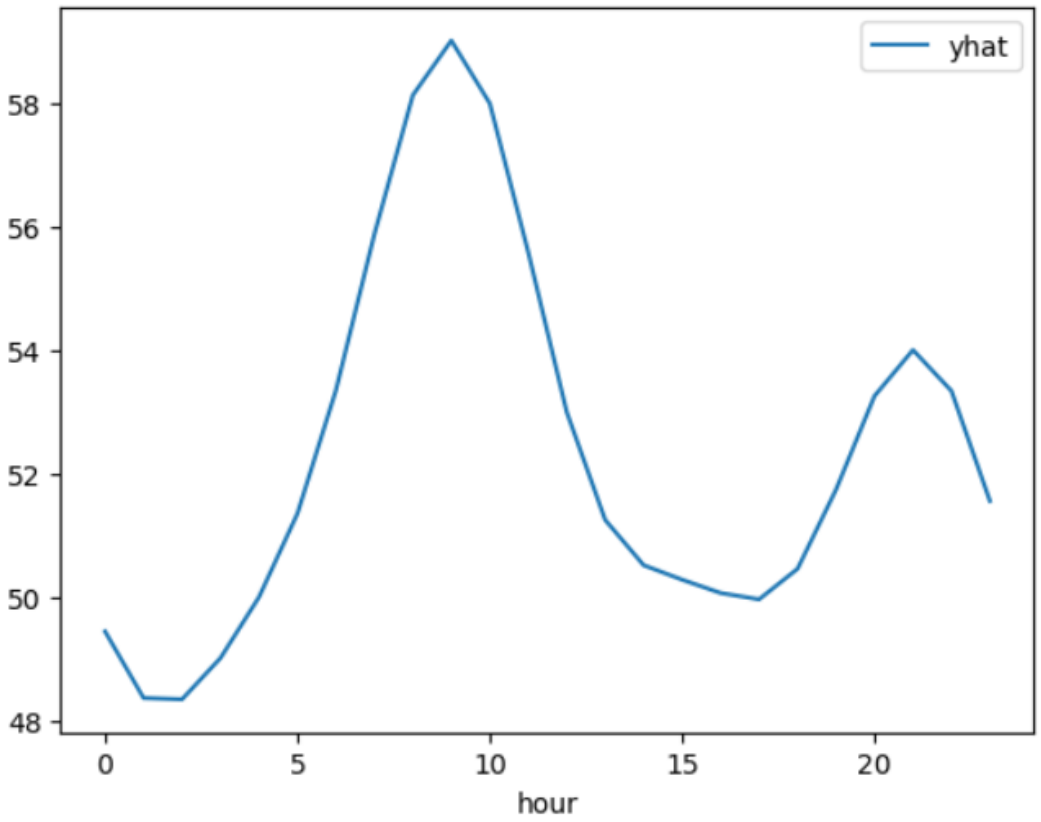
MAPE score του μοντέλου Prophet

Παρατηρούμε ότι το μοντέλο Prophet σημειώνει χαμηλότερο MAPE score από το μοντέλο XGboost, έτσι καταλήγουμε στο συμπέρασμα ότι το Prophet αποτελεί το **καταλληλότερο** για την παραγωγή προβλέψεων.

6. Συμπεράσματα

Σύμφωνα με την μελέτη μας, το μοντέλο Facebook Prophet είναι το πλέον κατάλληλο για την πρόβλεψη, τόσο για βραχυπρόθεσμη, όσο και για μακροπρόθεσμη διάρκεια. Αυτό αποδεικνύεται από την μετρική MAPE των δύο μοντέλων (αν και η διαφορά MAPE των δύο μοντέλων είναι μηδαμινή).

Στο επόμενο γράφημα φαίνεται η συνολική κατανάλωση ενέργειας (κάθε πελάτη) που προέβλεψε το μοντέλο Facebook Prophet ή Prophet στην περίοδο των επόμενων τριών ημερών. Η κατανάλωση αυτή απεικονίζεται ανά ώρα.



Μελλοντική συνολική κατανάλωση ανά ώρα

Παρατηρούμε ότι οι περισσότεροι πελάτες κάνουν κατανάλωση ενέργειας τα πρωϊνά και έπειτα τα απογεύματα. Επομένως, προκειμένου να μειωθεί η κατανάλωση ενέργειας, προτρέπουμε τους πελάτες να μειώνουν την συνεχή κατανάλωση ηλεκτρικής ενέργειας τις πρωινές ώρες και να αυξήσουν την κατανάλωση ενέργειας τις απογευματινές ώρες.

7. Βιβλιογραφία

Παρακάτω παρουσιάζονται οι πηγές που μας βοήθησαν στην μελέτη μας:

[1] “pandas.Series.resample — pandas 1.5.3 documentation.” https://pandas.pydata.org/docs/reference/api/pandas.Series.resample.html (accessed Jan. 29, 2023).

[2] “Anomaly Detection Model on Time Series Data using Isolation Forest | Engineering Education (EngEd) Program | Section.” https://www.section.io/engineering-education/anomaly-detection-model-on-time-series-data-using-isolation-forest/ (accessed Jan. 29, 2023).

[3] “Principal Component Analysis (PCA) Explained | Built In.” https://builtin.com/data-science/step-step-explanation-principal-component-analysis (accessed Jan. 29, 2023).

[4] “Principal Component Analysis (PCA) in Python Tutorial | DataCamp.” https://www.datacamp.com/tutorial/principal-component-analysis-in-python (accessed Jan. 29, 2023).

[5] “Introduction to Time Series Clustering | Kaggle.” https://www.kaggle.com/code/izzettunc/introduction-to-time-series-clustering/notebook?fbclid=IwAR1PJCXrefLcQ9xuhxWDWZY-GubIx8Y4Uv_GHOssyEXm2QqDv0d1Kk--SjI (accessed Jan. 29, 2023).

[6] “K-Means Elbow Method code for Python – Predictive Hacks.” https://predictivehacks.com/k-means-elbow-method-code-for-python/ (accessed Jan. 29, 2023).

[7] “Hotel Booking Demand DBSCAN Time Series Clustering | Kaggle.” https://www.kaggle.com/code/ceneksanzak/hotel-booking-demand-dbscan-time-series-clustering/notebook (accessed Jan. 29, 2023).

[8] “DBSCAN Python Example: The Optimal Value For Epsilon (EPS) | by Cory Maklin | Towards Data Science.” https://towardsdatascience.com/machine-learning-clustering-dbscan-determine-the-optimal-value-for-epsilon-eps-python-example-3100091cfbc (accessed Jan. 29, 2023).

[9] “pandas.DataFrame.rolling — pandas 1.5.3 documentation.” https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.rolling.html (accessed Jan. 29, 2023).

[10] “Time Series Forecasting with Machine Learning [YT] | Kaggle.” https://www.kaggle.com/code/robikscube/time-series-forecasting-with-machine-learning-yt (accessed Jan. 29, 2023).

[11] “Tutorial: Time Series Forecasting with Prophet | Kaggle.” https://www.kaggle.com/code/prashant111/tutorial-time-series-forecasting-with-prophet (accessed Jan. 29, 2023).

[12] “What is a good MAPE score? (simply explained).” https://stephenallwright.com/good-mape-score/ (accessed Jan. 29, 2023).