ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ Η/Υ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΒΙΟΜΗΧΑΝΙΚΗΣ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ
ΔΠΜΣ: «ΤΕΧΝΟ-ΟΙΚΟΝΟΜΙΚΑ ΣΥΣΤΗΜΑΤΑ»

ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΕΒΔΟΜΑΔΙΑΙΑΣ ΠΡΟΒΛΕΨΗΣ ΩΡΙΑΙΑΣ ΕΝΕΡΓΕΙΑΚΗΣ ΚΑΤΑΝΑΛΩΣΗΣ ΚΤΙΡΙΩΝ ΜΕΣΗΣ ΤΑΣΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΧΙΛΛΕΑΣ Ε. ΜΑΡΙΝΑΚΗΣ

Επιβλέπων: Βασίλειος Ασημακόπουλος,
Καθηγητής Ε. Μ. Π.
Υπεύθυνος: Αχιλλέας Ράπτης,
Υποψήφιος Διδάκτωρ Ε. Μ. Π.

Αθήνα, Σεπτέμβριος 2015

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ Η/Υ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΒΙΟΜΗΧΑΝΙΚΗΣ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ
ΔΠΜΣ: «ΤΕΧΝΟ-ΟΙΚΟΝΟΜΙΚΑ ΣΥΣΤΗΜΑΤΑ»

ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΕΒΔΟΜΑΔΙΑΙΑΣ ΠΡΟΒΛΕΨΗΣ ΩΡΙΑΙΑΣ ΕΝΕΡΓΕΙΑΚΗΣ ΚΑΤΑΝΑΛΩΣΗΣ ΚΤΙΡΙΩΝ ΜΕΣΗΣ ΤΑΣΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΧΙΛΛΕΑΣ Ε. ΜΑΡΙΝΑΚΗΣ

Επιβλέπων: Βασίλειος Ασημακόπουλος,
Καθηγητής Ε. Μ. Π.
Υπεύθυνος: Αχιλλέας Ράπτης,
Υποψήφιος Διδάκτωρ Ε. Μ. Π.

.....	.....	.....
Βασίλειος Ασημακόπουλος Καθηγητής Ε.Μ.Π.	Ιωάννης Ψαρράς Καθηγητής Ε.Μ.Π.	Δημήτριος Ασκούνης Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2015

.....

Αχιλλέας Ε. Μαρινάκης

MSc, Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

ΠΕΡΙΛΗΨΗ

Σκοπός της παρούσας μεταπτυχιακής διπλωματικής εργασίας είναι η παραγωγή προβλέψεων αναφορικά με την ωριαία ενεργειακή κατανάλωση κτιρίων μέσης τάσης για χρονικό ορίζοντα μίας εβδομάδας. Τα δεδομένα που εξετάσαμε αφορούσαν την κατανάλωση σε KWh ενός σούπερ μάρκετ στην περιοχή του Πειραιά με συχνότητα καταγραφής τους ανά δεκαπέντε λεπτά. Τα προγράμματα που υλοποιήσαμε συντάχθηκαν στην γλώσσα προγραμματισμού Java.

Στην εισαγωγή αναφέρουμε τις κρίσιμες αποφάσεις που λάβαμε κατά την εκπόνηση της εργασίας. Συγκεκριμένα, αποφασίσαμε να ορίζουμε τις κενές τιμές της χρονοσειράς ως ημιάθροισμα της αμέσως προηγούμενης και επόμενης τιμής, ενώ το μήκος της εποχιακότητας ορίστηκε 168. Τις μεθόδους που αναπτύξαμε τις συγκρίναμε με βάση το ελάχιστο μέσο απόλυτο ποσοστιαίο σφάλμα (mape) των προβλέψεων που αυτές παράγουν.

Στη συνέχεια περιγράφουμε αναλυτικά όλο το θεωρητικό υπόβαθρο της επιστημονικής περιοχής των τεχνικών προβλέψεων, στο οποίο βασιστήκαμε για την ανάπτυξη των μεθόδων μας. Στο κεφάλαιο αυτό καταγράφονται η βασική θεωρία περί χρονοσειρών, οι κύριες μέθοδοι προβλέψεων καθώς και οι στατιστικοί δείκτες αξιολόγησης τους. Από τα παραπάνω εστιάσαμε στις μεθόδους εκθετικής εξομάλυνσης και στα σφάλματα mse και mape. Στο κεφάλαιο 3 περιγράφουμε την διαδικασία προετοιμασίας της χρονοσειράς πριν αυτή εισαχθεί στα διάφορα μοντέλα πρόβλεψης, η οποία περιλαμβάνει την συμπλήρωση των κενών τιμών και την μετέπειτα συνάθροισή των ανά τέταρτο δεδομένων στα αντίστοιχα ωριαία.

Στα κεφάλαια 4, 5 και 6 αναλύονται αντιστοίχως οι τρεις μέθοδοι πρόβλεψης που υλοποιήσαμε. Η πρώτη είναι η απλή μέθοδος εκθετικής εξομάλυνσης (SES) με μήκος εποχιακότητας 168, όπου η τιμή του αρχικού επιπέδου ισούται με την τιμή του σταθερού επιπέδου της απλής γραμμικής παλινδρόμησης, ενώ για τον συντελεστή εξομάλυνσης a χρησιμοποιήθηκε το κριτήριο της ελαχιστοποίησης του μέσου τετραγωνικού σφάλματος (mse). Οι εν λόγω αποφάσεις ισχύουν για όλες τις μεθόδους. Η δεύτερη μέθοδος είναι μία παραλλαγή της πρώτης χωρίς την επίδραση της μέσης ημερήσιας θερμοκρασίας στα δεδομένα. Στα πλαίσια της τρίτης μεθόδου, «σπάμε» την χρονοσειρά σε 168 διαφορετικές χρονοσειρές, κάθε μία από τις οποίες περιέχει τιμές που αντιστοιχούν μόνο σε μία συγκεκριμένη ημέρα και ώρα της εβδομάδας. Στη συνέχεια για κάθε χρονοσειρά εφαρμόζουμε την SES.

Τέλος, στο κεφάλαιο 7 συγκρίνουμε τα αποτελέσματα των τριών μεθόδων με βάση το mape και συμπεραίνουμε ότι η τρίτη μέθοδος και μάλιστα στην βελτιωμένη μορφή της (παράγραφοι 1.4.3.1 και 6.4), παράγει τα καλύτερα αποτελέσματα. Στο σύνολο των δεδομένων το μέσο ελάχιστο mape της μεθόδου αυτής ήταν 7,02%.

Λέξεις κλειδιά: Τεχνικές Προβλέψεων, Ενεργειακή Κατανάλωση, Απλή Εκθετική Εξομάλυνση, Εποχιακότητα Μήκους 168

ABSTRACT

The scope of this master thesis is to predict the hourly energy consumption of medium voltage buildings within a weekly forecast horizon. The data we have examined concerned the consumption, in KWh, of a supermarket in Piraeus on a recording frequency of fifteen minutes. Java programming language has been used for the development of all the programs we implemented.

Firstly, we refer to the issues resolved during our drafting. More specifically, we have defined the blank values of the time series as the semi – sum of the previous and the next value and we have set the seasonality length as 168. Moreover, we decided to compare the methods developed, on the basis of the minimum mean absolute percentage error (mape).

Subsequently, we describe the theoretical background of forecasting, upon which the methods developed in the thesis are based, i.e. the basic time series theory, the main forecasting methods and the statistical indicators for their evaluation with a focus on exponential smoothing methods and mse and mape errors. In chapter 3 we describe the preparation process for the time series, which includes the completion of empty values and the subsequent aggregation of the per quarter data in the corresponding hourly ones.

In chapters 4, 5 and 6 the three forecasting methods that we implemented are respectively analyzed. The first method is the simple exponential smoothing (SES) with seasonality length 168, where the value of the initial level is the one of the fixed level of simple linear regression and in addition, the criterion of minimizing the mean square error (mse) is used for estimating the smoothing factor a. The latter are valid for all methods. The second method is a variation of the first one without the influence of the average daily temperature in the data. In the context of the third method we “break” the time series in 168 different time series, each of which contains values corresponding to only one specific day and time of the week. Then for each time series we apply the SES.

Finally, in chapter 7 we compare the results of the three methods implemented, on the basis of mape and we conclude that the third method, in its improved version (paragraphs 1.4.3.1 and 6.4), produces the best results. In the specific dataset the minimum average mape of this method was 7.02%.

Key words: Forecasting Techniques, Energy Consumption, Simple Exponential Smoothing, Seasonality Length 168

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω, πάνω από όλους, τον καθηγητή μου κύριο Βασίλειο Ασημακόπουλο, που με εμπιστεύτηκε για την εργασία και μου παρείχε τις πολύτιμες γνώσεις του και την βοήθειά του σε οποιοδήποτε πρόβλημα αντιμετώπισα. Η καθοδήγησή του υπήρξε καταλυτική και η εν γένει συνεργασία μας καθ' όλη τη διάρκεια εκπόνησης της παρούσας μεταπτυχιακής διπλωματικής εργασίας ήταν άψογη.

Θερμές ευχαριστίες θα ήθελα επίσης να απευθύνω στον υποψήφιο διδάκτορα ΣΗΜΜΥ ΕΜΠ Αχιλλέα Ράπτη, η συμβολή του οποίου υπήρξε καθοριστική για την εκπόνηση της εργασίας. Οι συμβουλές του τόσο σε οργανωτικό, όσο και επιστημονικό επίπεδο υπήρξαν πολύτιμες.

Τέλος, θα ήθελα να ευχαριστήσω τον αδερφό μου και συνάδελφό μου Νίκο, τους γονείς μου Βαγγέλη και Θέκλα και την φίλη μου Αντιγόνη Παπαδάτου, για την ηθική τους συμπαράσταση κατά την διάρκεια των μεταπτυχιακών σπουδών μου.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

1. ΕΙΣΑΓΩΓΗ

Σκοπός της παρούσας μεταπτυχιακής διπλωματικής εργασίας είναι η παραγωγή προβλέψεων για την ωριαία ενεργειακή κατανάλωση κτιρίων μέσης τάσης. Για τον σκοπό αυτό υλοποιήσαμε τρεις διαφορετικές μεθόδους, τα συγκριτικά αποτελέσματα των οποίων παρουσιάζονται στο κεφάλαιο 7.

1.1. ΠΕΡΙΓΡΑΦΗ ΔΕΔΟΜΕΝΩΝ

Τα δεδομένα που χρησιμοποιήσαμε προκειμένου να εφαρμόσουμε τις διάφορες μεθόδους προβλέψεων, αφορούν την ενεργειακή κατανάλωση ενός super market στην ευρύτερη περιοχή του Πειραιά. Η μονάδα μέτρησης των δεδομένων είναι η KWh και η χρονική περίοδός τους εκτείνεται από την 01/01/2012 έως και την 31/12/2013 (δηλαδή δύο έτη). Τα (ακατέργαστα) δεδομένα ήταν διαθέσιμα (σε αρχεία excel) με συχνότητα ενός τετάρτου της ώρας, αλλά με δεδομένο ότι θέλαμε να προβλέψουμε την ωριαία κατανάλωση του κτιρίου, έπρεπε να συναθροίζουμε τα δεδομένα αυτά σε ωριαία βάση. Η διαδικασία αυτή περιγράφεται αναλυτικά στην παράγραφο 3.2. Στην αρχή της υποπαραγράφου 3.3.4, παρουσιάζεται ένα υποσύνολο των (ακατέργαστων αυτών) δεδομένων που χρησιμοποιήσαμε για να δοκιμάσουμε τις μεθόδους μας.

1.2. ΣΤΟΙΧΕΙΑ ΠΡΟΒΛΕΨΕΩΝ

Δύο κρίσιμοι παράγοντες κατά την διαδικασία των προβλέψεων είναι τα ιστορικά δεδομένα που χρησιμοποιούνται και ο χρονικός ορίζοντας της πρόβλεψης. Αναφορικά με τις προβλέψεις που παρήχθησαν στα πλαίσια της εργασίας μας, το χρονικό εύρος των ιστορικών δεδομένων ήταν από 1 έως 12 μήνες και ο χρονικός ορίζοντας ήταν 1 εβδομάδα (δηλαδή 7 * 24 = 168 ωριαίες τιμές). Συγκεκριμένα προβλέψαμε την ωριαία ενεργειακή κατανάλωση του κτιρίου, καθ' όλη τη διάρκεια της δεύτερης εβδομάδας όλων των μηνών του έτους 2013. Για την υλοποίηση των μεθόδων I (κεφάλαιο 4) και II (κεφάλαιο 5), το πλήθος των ιστορικών δεδομένων ήταν από 1 έως 12 μήνες (Ιανουάριος 2012 έως Νοέμβριος 2013, ανάλογα με τον υπό πρόβλεψη μήνα) + τις τιμές που αφορούσαν την κατανάλωση της πρώτης εβδομάδας κάθε μήνα. Αντιθέτως, για την υλοποίηση της μεθόδου III (κεφάλαιο 6), το πλήθος των ιστορικών δεδομένων ήταν από 3 έως 12 μήνες + τις τιμές της πρώτης εβδομάδας κάθε μήνα. Τα mape των διαφόρων προβλέψεών μας παρουσιάζονται στην Εικόνα 22, στην Εικόνα 34, Εικόνα 38, στην Εικόνα 43 και στην Εικόνα 45.

1.3. ΚΡΙΣΙΜΕΣ ΑΠΟΦΑΣΕΙΣ

Κατά την διάρκεια του σχεδιασμού των μεθόδων πρόβλεψής μας, χρειάστηκε να λάβουμε κάποιες πολύ κρίσιμες αποφάσεις:

- ***Διαδικασία εκτίμησης κενών τιμών***: Όπως αναλυτικά περιγράφουμε στην παράγραφο 3.1, αποφασίσαμε οι κενές τιμές της χρονοσειράς μας να ορίζονται ως το ημιάθροισμα (μέσος όρος) της προηγούμενης και της επόμενης πραγματικής παρατήρησης.

- ***Μήκος εποχιακότητας χρονοσειράς***: Η έννοια της χρονοσειράς αναλύεται στην υποπαράγραφο 2.2.2, ενώ τα ποιοτικά χαρακτηριστικά της (όπως είναι η εποχιακότητα) περιγράφονται στην 2.2.4. Ύστερα από πολλές πειραματικές μετρήσεις, διαπιστώσαμε ότι οι τιμές της χρονοσειράς παρουσιάζουν εποχιακότητα σε σχέση με την ώρα και την ημέρα κάθε εβδομάδας. Συνεπώς, το μήκος της εποχιακότητας τέθηκε ίσο με 168, στα πλαίσια της μεθόδου I (κεφάλαιο 4) και της μεθόδου II (κεφάλαιο 5), όπου λαμβάνει χώρα η διαδικασία της κλασικής μεθόδου αποσύνθεσης (2.3.1.2 και 4.1).

- ***Βασική μέθοδος πρόβλεψης***: Όλες οι μέθοδοι που αναπτύξαμε έχουν σαν βάση τους την απλή εκθετική εξομάλυνση – **SES** (2.4.4.1). Ύστερα από πολλές πειραματικές μετρήσεις, διαπιστώσαμε ότι η **SES** αποδίδει καλύτερα σε σχέση με τις Naive, LRL, Holt, Damped και Theta, που επίσης δοκιμάστηκαν (2.4), σε σχέση πάντοτε και με τον απαραίτητο υπολογιστικό χρόνο.

- ***Αρχικό επίπεδο SES***: Αποφασίσαμε η τιμή του αρχικού επιπέδου στη SES να ισούται με την τιμή του σταθερού επιπέδου από το μοντέλο της απλής γραμμικής παλινδρόμησης.

- ***Συντελεστής εξομάλυνσης SES***: Το κριτήριο που χρησιμοποιήσαμε για τον προσδιορισμό του συντελεστή εξομάλυνσης a, είναι η ελαχιστοποίηση του μέσου τετραγωνικού σφάλματος (MSE), που είναι και το επικρατέστερο κριτήριο στην βιβλιογραφία.

- ***Κριτήριο σύγκρισης των μεθόδων***: Για την σύγκριση της προβλεπτικής ικανότητας των διαφόρων μεθόδων που αναπτύξαμε, χρησιμοποιήσαμε ως κριτήριο το μικρότερο mape της κάθε μίας από αυτές.

- ***Γλώσσα προγραμματισμού***: Όλα τα προγράμματα υλοποιήθηκαν σε **Java**, ενώ σε κάποιες περιπτώσεις χρησιμοποιήθηκε (άτυπα) και η R για λόγους επαλήθευσης.

1.4. ΠΕΡΙΓΡΑΦΗ ΜΕΘΟΔΩΝ

Σε κάθε μία από τις παρακάτω μεθόδους η χρονοσειρά που εισέρχεται σαν είσοδος στο μοντέλο, είναι αυτή που προκύπτει όταν στα αρχικά δεδομένα εφαρμόσουμε, πρώτα την συμπλήρωση των κενών τιμών (παράγραφος 3.1) και έπειτα την ωριαία συνάθροιση των δεδομένων (παράγραφος 3.2).

1.4.1. ΜΕΘΟΔΟΣ Ι

Στην μέθοδο Ι (κεφάλαιο 4), η χρονοσειρά αποεποχικοποιείται σύμφωνα με την κλασική μέθοδο αποσύνθεσης (2.3.1.2 και 4.1), με μήκος εποχιακότητας ίσο με 168. Στην συνέχεια, με βάση την αποεποχικοποιημένη χρονοσειρά, παράγονται προβλέψεις με την μέθοδο **SES** (2.4.4.1) και τέλος αυτές οι προβλέψεις επαναεποχικοποιούνται παράγοντας έτσι τις τελικές προβλέψεις.

1.4.2. ΜΕΘΟΔΟΣ ΙΙ

Η μέθοδος ΙΙ (κεφάλαιο 5) είναι παρόμοια με την μέθοδο Ι, με την διαφορά ότι οι τιμές της χρονοσειράς διαιρούνται με την μέση θερμοκρασίας Κελσίου της αντίστοιχης ημέρας, προτού λάβει χώρα η ίδια διαδικασία της αποεποχικοποίησης. Ομοίως, αφού παραχθούν οι «ενδιάμεσες» προβλέψεις με την **SES,** αυτές πολλαπλασιάζονται με την αντίστοιχη μέση θερμοκρασία και επαναεποχικοποιούνται, για να παραχθούν οι τελικές προβλέψεις.

1.4.3. ΜΕΘΟΔΟΣ ΙΙΙ

Στα πλαίσια της μεθόδου ΙΙΙ (κεφάλαιο 6), «σπάμε» την χρονοσειρά μας σε 168 διαφορετικές χρονοσειρές, κάθε μία από τις οποίες περιέχει τιμές που αντιστοιχούν μόνο σε μία συγκεκριμένη ημέρα και ώρα της εβδομάδας. Στη συνέχεια για κάθε χρονοσειρά παράγουμε μία πρόβλεψη με την μέθοδο **SES.** Το mape της μεθόδου ΙΙΙ ισούται με τον μέσο όρο όλων των mape των 168 αυτών διαφορετικών χρονοσειρών.

1.4.3.1 ΒΕΛΤΙΩΜΕΝΗ ΜΕΘΟΔΟΣ ΙΙΙ

Η μέθοδος ΙΙΙ πετυχαίνει καλύτερα αποτελέσματα εάν πραγματοποιήσουμε την εξής τροποποίηση (παράγραφος 6.4): ελέγχουμε εάν κάποια από τις υπό πρόβλεψη τιμές ανήκει σε διαφορετικό «είδος» ημέρας (αργία ή όχι αργία) από την αντίστοιχη αμέσως προηγούμενή της, δηλαδή την τελευταία τιμής της χρονοσειράς εισόδου στην μέθοδο **SES**. Εάν ισχύει αυτό, τότε η τιμή αυτή αφαιρείται από την χρονοσειρά και η πρόβλεψη βασίζεται στις υπόλοιπες τιμές. Ο λόγος που μας οδήγησε σε αυτή την τροποποίηση είναι ότι διαπιστώσαμε, μελετώντας τα δεδομένα μας, ότι η ενεργειακή κατανάλωση του super market ήταν πολύ διαφορετική τις αργίες σε σχέση με τις εργάσιμες μέρες, γεγονός που είναι απολύτως λογικό και αναμενόμενο.

Περιπτώσεις εφαρμογής

1. Ιανουάριος 2013: για την πρόβλεψη των εργάσιμων ωρών (6:00 - 22:00) της Τρίτης 8 Ιανουαρίου 2013 αφαιρέσαμε τις αμέσως 2 προηγούμενες τιμές (1/1/2013 και 25/12/2012) γιατί αυτές ήταν αργίες (Πρωτοχρονιά και Χριστούγεννα αντίστοιχα). Το ίδιο κάναμε και για την Τετάρτη 9 Ιανουαρίου καθώς στις 2/1 τα super market κάνουν απογραφή και 26/12 είναι επίσης αργία. Αντίθετα δεν κάναμε το ίδιο για την αργία των Φώτων (6/1/2013), που πιθανόν να επηρέαζε τις προβλέψεις για τις 13/1/2013, γιατί συνέπεσε να είναι Κυριακή, δηλαδή έτσι κι αλλιώς αργία. Πράγματι, μετά από αυτήν την τροποποίηση το min mape για τον Ιανουάριο έπεσε από το 12,22% (Εικόνα 38) στο 6,22% (Εικόνα 43).

2. Μάιος 2013: για την πρόβλεψη των εργάσιμων ωρών της Δευτέρας 13/5/2013 αφαιρέσαμε την αμέσως προηγούμενη τιμή, γιατί η 6/5/2013 ήταν Δευτέρα του Πάσχα. Πράγματι, μετά από αυτήν την τροποποίηση το min mape για τον Μάιο έπεσε από το 7,85% (Εικόνα 38) στο 5,13% (Εικόνα 45).

2. ΤΕΧΝΙΚΕΣ ΠΡΟΒΛΕΨΕΩΝ

2.1. ΓΕΝΙΚΑ ΠΕΡΙ ΠΡΟΒΛΕΨΕΩΝ

Οι προβλέψεις πάντα αποτελούσαν αναπόσπαστο κομμάτι της ανθρώπινης φύσης, τόσο σε απλές καθημερινές περιστάσεις όσο και σε σημαντικότερες αποφάσεις που αφορούν το ατομικό ή το συλλογικό μας μέλλον. Από την απόφαση για το αν θα πάρουμε ομπρέλα μαζί μας το πρωί έως την απόφαση για την επένδυση εκατομμυρίων ευρώ, η πρόβλεψη του καιρού από την μία πλευρά και της παγκόσμιας οικονομίας από την άλλη, μας οδηγούν στην λήψη της απόφασης. Ειδικότερα, η σημασία των προβλέψεων έχει αυξηθεί ραγδαία από το 1980 και μετά και η αύξηση αυτή έχει γίνει εμφανής τόσο σε ακαδημαϊκό επίπεδο όσο και στο επίπεδο των επιχειρήσεων.

Αναπόφευκτα λοιπόν, συμβολή σε αυτή την ανάπτυξη του κλάδου των προβλέψεων έχουν από την μία πλευρά οι ακαδημαϊκοί με την εξέλιξη της επιστήμης των προβλέψεων, κυρίως με την δημιουργία ενός μεγάλου συνόλου μεθόδων και από την άλλη πλευρά τα στελέχη των επιχειρήσεων με την πρακτική εφαρμογή των μεθόδων αυτών στο εσωτερικό των επιχειρήσεων.

Το ενδιαφέρον για τις προβλέψεις πηγάζει κυρίως από την αβεβαιότητα για το μέλλον, η οποία αποτελεί ένα βασικό χαρακτηριστικό των οικονομιών, οι οποίες πλήττονται από την οικονομική ανασφάλεια. Από τους διοικητές των επιχειρήσεων και τους υπεύθυνους για την λήψη πολιτικών αποφάσεων έως τους απλούς πολίτες στην καθημερινή τους ζωή, όλοι μας βρισκόμαστε αντιμέτωποι με την αβεβαιότητα. Η κατάσταση αβεβαιότητας γίνεται όλο και πιο έντονη και έχει επιβάλλει μια πιο συστηματική και προσεκτική έρευνα του μέλλοντος. Τόσο τα ιστορικά δεδομένα όσο και οι προβλέψεις καθαυτές χρησιμοποιούνται σαν δεδομένα σε όλες τις κατηγορίες σχεδιασμού, πολιτικού σχεδιασμού, χρονικού προγραμματισμού καθώς και πλήθος δραστηριοτήτων λήψης απόφασης. Για τους παραπάνω λόγους, η αναγκαιότητα της πρόβλεψης είναι κάτι παραπάνω από επιβεβλημένη. Αυτή τη χρονική περίοδο η μεγαλύτερη πρόκληση στον τομέα των προβλέψεων είναι να γίνει η διαδικασία των προβλέψεων όσο το δυνατόν πιο χρήσιμη, αποδοτική και ακριβής.

Η αβεβαιότητα που αποτελεί και το πιο σημαντικό «εχθρό» της επιστήμης των προβλέψεων έχει κατηγοριοποιηθεί από τον Μακρυδάκη και τους συνεργάτες του στο βιβλίο «Χορεύοντας με την Τύχη» [1]. Αναφέρουν λοιπόν δυο είδη αβεβαιότητας που συναντάμε καθημερινά, την «αβεβαιότητα του μετρό» και την «αβεβαιότητα της καρύδας». Η «αβεβαιότητα του μετρό» αναφέρεται σε συνεχείς μικρές τυχαίες διακυμάνσεις τόσο της ιδιωτικής όσο και της επιχειρηματικής καθημερινότητας. Ο όρος προήλθε από την επιπλέον χρονική διάρκεια που θα χρειαστεί ένας συρμός για την διαδρομή μεταξύ δυο απομακρυσμένων σταθμών, λόγω ενός τεχνικού προβλήματος, της πολυκοσμίας ή μιας στάσης εργασίας. Από την άλλη, η «αβεβαιότητα της καρύδας» αναφέρεται σε ένα εντελώς απρόσμενο και σπάνιο γεγονός που να έχει σημαντικές επιδράσεις και συνέπειες. Η αβεβαιότητα αυτού του είδους πήρε το όνομά της από το απρόσμενο γεγονός που μπορεί να συμβεί καθώς

περπατάμε στο δρόμο και πέφτει στο κεφάλι μας μια καρύδα! Η «αβεβαιότητα της καρύδας» αντιπαραβάλλεται με μεγάλες φυσικές ή οικονομικές καταστροφές που είναι δύσκολο να προβλεφθεί το πότε και το που θα συντελεστούν καθώς και πόσο μεγάλες θα είναι οι επιδράσεις τους.

Όλα αυτά τα χρόνια, η επιστήμη των προβλέψεων έχει δεχθεί σφοδρές κριτικές και είχε να αντιμετωπίσει τη μεγάλη δυσαρέσκεια σχετικά με την ανικανότητα των μεθόδων να προειδοποιήσουν έγκαιρα για επερχόμενες αλλαγές καθώς και για τα μεγάλα σφάλματα στις προβλέψεις. Συγχρόνως όμως, λανθασμένες ενέργειες οι οποίες προκαλούνται από ασταθή περιβάλλοντα, μη αναμενόμενες εξελίξεις, ασυνέχειες και άλλα αυξάνουν την ανάγκη για πραγματοποίηση προβλέψεων. Όταν δεν υπάρχει αβεβαιότητα στο περιβάλλον και τα πάντα κυλούν ομαλά και ακολουθούν την αναμενόμενη πορεία τους, δεν υπάρχει καμία ουσιαστική ανάγκη για προβλέψεις. Σε περιόδους όμως, οι οποίες χαρακτηρίζονται από συνεχείς και απότομες μεταβολές και συνεπώς η ικανότητα πρόβλεψης είναι αισθητά μειωμένη, η ζήτηση και η ανάγκη για προβλέψεις παρουσιάζει κατακόρυφη αύξηση. Πρακτικά το παραπάνω γεγονός επιβεβαιώνεται σε περιόδους οικονομικών και άλλων κρίσεων κατά την διάρκεια των οποίων η αναζήτηση για συμβούλους προβλέψεων αυξάνεται.

2.2. ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΧΡΟΝΟΣΕΙΡΩΝ

2.2.1. ΕΙΣΑΓΩΓΗ

Για την εξαγωγή προβλέψεων αλλά και γενικότερα για την στατιστική ανάλυση και τη μελέτη μιας μεταβλητής, το πρώτο σημαντικό βήμα είναι η συλλογή και η οργάνωση των ιστορικών στοιχείων της μεταβλητής αυτής. Τα δεδομένα που θα συλλεχθούν θα πρέπει να είναι όσο το δυνατόν πιο έγκυρα και πιο επικαιροποιημένα έτσι ώστε η μετέπειτα επεξεργασία τους με κάποια μέθοδο πρόβλεψης να μας επιτρέψει να επιτύχουμε όσο το δυνατόν καλύτερη ακρίβεια.

Μια από τις διάφορες κατηγοριοποιήσεις δεδομένων η οποία έχει επικρατήσει στον τομέα των προβλέψεων είναι ο διαχωρισμός των δεδομένων σε δυο βασικές κατηγορίες. Τα ***διαστρωματικά στοιχεία*** (cross – sectional data) και οι ***χρονολογικές σειρές*** (time series), που έχουν επικρατήσει με την ονομασία ***χρονοσειρές,*** αποτελούν τις δύο αυτές κατηγορίες. Η βασική διαφορά τους είναι ότι στα διαστρωματικά δεδομένα για ένα συγκεκριμένο μέγεθος και για το ίδιο χρονικό διάστημα υπάρχουν πολλές παρατηρήσεις ενώ οι χρονοσειρές αποτελούνται από μια αλληλουχία διαχρονικών παρατηρήσεων του ίδιου μεγέθους. Οι μέθοδοι που αναπτύχθηκε στα πλαίσια της συγκεκριμένης μεταπυχιακής εργασίας εφαρμόζεται πάνω σε χρονοσειρές και για αυτό θα δοθεί ιδιαίτερη έμφαση στην ανάλυση των χαρακτηριστικών τους.

2.2.2. ΑΝΑΛΥΣΗ ΕΝΝΟΙΑΣ ΧΡΟΝΟΣΕΙΡΑΣ

Οι χρονοσειρές αποτελούν ένα σύνολο διαδοχικών παρατηρήσεων της τιμής κάποιου φυσικού ή άλλου μεγέθους ανηγμένες στο χρόνο. Βάσει της αλληλουχίας τιμών των διαδοχικών δεδομένων των χρονοσειρών μπορεί να γίνει και ο διαχωρισμός τους. Οι δυο βασικές κατηγορίες χρονοσειρών με βάση των τρόπο προσδιορισμού των μελλοντικών δεδομένων είναι οι ***ντετερμινιστικές*** χρονοσειρές και οι ***στοχαστικές*** χρονοσειρές. Στις ντετερμινιστικές χρονοσειρές οι διαδοχικές παρατηρήσεις της χρονοσειράς δεν είναι ανεξάρτητες μεταξύ τους και οι μελλοντικές τιμές μπορούν να υπολογιστούν από τις προηγούμενες. Αντίθετα, στις στοχαστικές χρονοσειρές, οι τιμές των μελλοντικών παρατηρήσεων προκύπτουν από μια στοχαστική διαδικασία και δεν περιγράφονται πλήρως από το παρελθόν των αντίστοιχων τιμών.

Ιδιαίτερο ενδιαφέρον παρουσιάζει η παρακολούθηση, η ανάλυση και η πρόβλεψη πραγματικών χρονοσειρών, καθώς η εξέλιξή τους είναι εν γένει άγνωστη και επιδέχεται πρόβλεψη. Στην πραγματικότητα όμως το μέλλον των πραγματικών χρονοσειρών καθορίζεται μερικώς μόνο από το παρελθόν, αφού η πλειοψηφία των χρονοσειρών που εμφανίζονται στον πραγματικό κόσμο επηρεάζονται από κάποιο «τυχαίο παράγοντα». Έτσι, θεωρείται ότι οι χρονοσειρές αντιπροσωπεύουν κυρίως στοχαστικές διαδικασίες και όχι ντετερμινιστικά συστήματα.

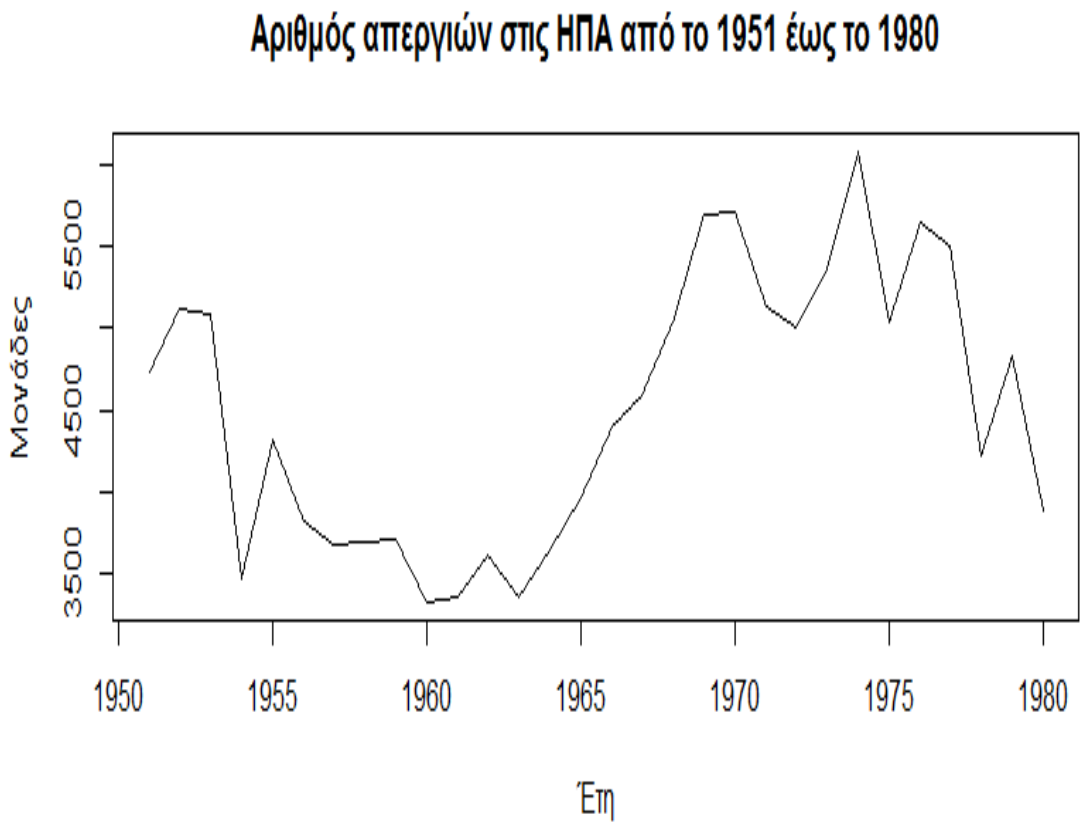
Ο διαχωρισμός των χρονοσειρών που αναφέρθηκε δεν είναι πάντα τόσο προφανής ώστε άμεσα να μπορούμε να χαρακτηρίσουμε μια χρονοσειρά. Παρόλα αυτά, η κατηγοριοποίηση είναι απαραίτητη για την αναγνώριση και την κατανόηση των παραμέτρων που επηρεάζουν την εξέλιξη μιας χρονοσειράς ανεξάρτητα με το είδος των δεδομένων τους.

2.2.3. ΑΝΑΠΑΡΑΣΤΑΣΗ ΧΡΟΝΟΣΕΙΡΩΝ

Όπως αναλύσαμε στην προηγούμενη παράγραφο η χρονοσειρά δεν είναι τίποτα παραπάνω από μια σειρά παρελθουσών τιμών για την περιγραφή ενός μεγέθους η μιας μεταβλητής. Οι τιμές αυτές αποτελούν την ιστορική πληροφορία της χρονοσειράς, ή όπως έχει επικρατήσει να ονομάζονται τα ιστορικά δεδομένα. Όταν τα ιστορικά δεδομένα αυτά αρχίζουν να μεγαλώνουν σε όγκο (πχ ιστορικά δεδομένα της τιμής μιας μετοχής του Χρηματιστηρίου ανά λεπτό από το 1990), ή ακόμα πιο σύνθετα όταν αρχίζουμε να μελετάμε αρκετά ομοειδή μεγέθη (πχ ιστορικά δεδομένα όλων των τιμών των μετοχών του Χρηματιστηρίου ανά λεπτό από το 1990) τότε το πρόβλημα αρχίζει να γίνεται εμφανές και ένας άνθρωπος αδυνατεί να ανταποκριθεί. Είναι λοιπόν προφανής η ανάγκη για μια πιο απτή αναπαράσταση των ιστορικών δεδομένων τόσο για την μελέτη της χρονοσειράς όσο και για την παρουσίασή της.

Η λύση στο πρόβλημα αυτό δόθηκε με τη δισδιάστατη γραφική αναπαράσταση των ιστορικών δεδομένων κάθε χρονοσειράς. Οι κύριοι τύποι γραφημάτων που χρησιμοποιούνται για την γραφική αναπαράσταση χρονοσειρών είναι:

1 ***Διαγράμματα Χρόνου*** (time plots): Είναι το πλέον προφανές και χρησιμοποιούμενο διάγραμμα και αναπαριστά τα διαθέσιμα δεδομένα στην πάροδο του χρόνου. Μέσω των διαγραμμάτων αυτών γίνονται άμεσα αντιληπτά τα βασικά χαρακτηριστικά των χρονοσειρών όπως η τάση και η εποχιακότητα. Τα γραφήματα αυτού του είδους χρησιμοποιούνται ευρέως σχεδόν σε όλους τους τομείς των επιχειρήσεων αλλά και της καθημερινότητας (χρηματιστήρια, πωλήσεις προϊόντων, εισαγωγές-εξαγωγές προϊόντων, κοινωνικοπολιτικά στοιχεία, κ.τ.λ.), ενώ παράλληλα γίνονται με ευκολία κατανοητά από την πλειοψηφία των ανθρώπων.



Εικόνα 1: Παράδειγμα Διαγράμματος Χρόνου - Αριθμός απεργιών στις ΗΠΑ από το 1951 έως το 1980

2 ***Εποχιακά Διαγράμματα*** (seasonal plots): Τα διαγράμματα αυτά ενδείκνυνται για χρονοσειρές που εμφανίζουν έντονη εποχιακότητα.

Εικόνα 2: Παράδειγμα Εποχιακού Διαγράμματος - Μηνιαία παραγωγή γάλακτος ανά αγελάδα για 14 χρόνια

3 ***Διαγράμματα Διασποράς*** (scatter plots): Η χρήση αυτού του τύπου γραφημάτων γίνεται κυρίως για σύγκριση μεταξύ διαφορετικών προϊόντων, υπηρεσιών ή οποιαδήποτε άλλη δυνατή σύγκριση. Το γράφημα παρουσιάζει τα διαφορετικά αυτά επιλεγμένα δεδομένα και αποτυπώνει την σύγκριση σε σχέση μεταξύ δύο διαφορετικών μεγεθών ή χαρακτηριστικών που αφορούν αυτά τα δεδομένα.

Από την γραφική αναπαράσταση και εν γένει την οπτικοποίηση των ιστορικών δεδομένων καθίσταται ευκολότερη η διαδικασία αναγνώρισης των βασικών χαρακτηριστικών της χρονοσειράς καθώς και η εύρεση ακραίων και ιδιαίτερων τιμών, των οποίων η διόρθωση ή η αντιμετώπιση είναι απόφαση του αναλυτή.

2.2.4. ΠΟΙΟΤΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΩΝ ΧΡΟΝΟΣΕΙΡΩΝ

Όπως ήδη έχουμε αναφέρει οι χρονοσειρές δομούνται από κάποια βασικά χαρακτηριστικά και φυσικά με κατάλληλες τεχνικές μπορούν, υπό προϋποθέσεις, να αναλυθούν σε αυτά. Οι παραδοσιακές μέθοδοι ανάλυσης των χρονοσειρών ασχολούνται με την ανάλυση της διακύμανσης της χρονοσειράς σε τέσσερα βασικά συστατικά: την ***τάση***, την ***κυκλικότητα***, την ***εποχιακότητα*** και τις ***μη κανονικές διακυμάνσεις***. Η προσέγγιση αυτή είναι χρήσιμη όχι μόνο για την εφαρμογή της κλασικής μεθόδου αποσύνθεσης, ένα μέρος της οποίας χρησιμοποιείται σε κάποιες μεθόδους μας για την αποεποχικοποίηση των χρονοσειρών, αλλά και για την ανάλυση κάθε χρονοσειράς σε επιμέρους στοιχεία ώστε ο αναλυτής να είναι σε θέση να χειριστεί την κάθε χρονοσειρά με την ενδεικνυόμενη για κάθε περίπτωση τεχνική ή μέθοδο. Θα αναφερθούμε ξεχωριστά σε κάθε ένα από τα στοιχεία στα οποία μπορεί να αποσυντεθεί μια χρονοσειρά:

Αρχικά η ***τάση***, αποτελεί το πρώτο συστατικό μιας χρονοσειράς και ορίζεται ως μια «μακροπρόθεσμη» μεταβολή του μέσου επιπέδου τιμών αυτής. Ο ορισμός της τάσης, αν και κοινά αποδεκτός, δημιουργεί ένα πρόβλημα σχετικά με ποια μεταβολή θεωρείται μακροπρόθεσμη έτσι ώστε να μπορεί να εξεταστεί η αντίστοιχη αύξηση ή μείωση στο μέσο επίπεδο. Η απάντηση στο ερώτημα αυτό ποικίλει ανάλογα με την φύση των εξεταζόμενων δεδομένων. Απαραίτητη σε κάθε περίπτωση είναι η ύπαρξη ικανοποιητικού όγκου ιστορικών δεδομένων έτσι ώστε να μπορεί με ασφάλεια να εξαχθεί κάποιο συμπέρασμα σχετικά με την τάση. Η τάση στην γενική της εικόνα μπορεί να είναι ανοδική, πτωτική ή σταθερή και μπορεί να εκτιμηθεί κατά προσέγγιση με μια ευθεία γραμμή ή μια εκθετική καμπύλη ή οποιαδήποτε άλλη οικογένεια καμπυλών.

Η ***κυκλικότητα*** είναι το δεύτερο συστατικό μιας χρονοσειράς και αντιπροσωπεύει μια μεταβολή που εμφανίζεται κατά περιόδους. Η κυκλικότητα οφείλεται κατά κύριο λόγο σε εξωγενείς παράγοντες και το μήκος των περιόδων εμφάνισής της είναι πάντα μεγαλύτερο από ένα έτος (πενταετία, δεκαετία, κλπ). Στις γραφικές παραστάσεις των χρονοσειρών παρουσιάζεται ως μια κυματοειδής γραμμή η οποία κινείται ανάμεσα στις ακραίες στάθμες της χρονοσειράς. Κυκλικότητα εμφανίζεται κυρίως σε οικονομικές χρονοσειρές όπως το Ακαθάριστο Εθνικό Προϊόν, οι τιμές μετοχών και αμοιβαίων κεφαλαίων και οι τιμές πετρελαίου και χρυσού. Αυτό οφείλεται στις γενικότερες οικονομικές συνθήκες που χαρακτηρίζονται από διαδοχικές ανόδους και υφέσεις των παγκόσμιων και εγχώριων οικονομιών και για αυτό οι μεταβολές στις οικονομικές χρονοσειρές είναι γνωστές με την ονομασία επιχειρηματικός κύκλος.

Η ***εποχιακότητα*** είναι το τρίτο ποιοτικό χαρακτηριστικό μιας χρονοσειράς και ορίζεται ως μια περιοδική διακύμανση η οποία έχει σταθερό και μικρότερο ή ίσο μήκος από ένα έτος. Η εποχιακότητα είναι μετά την τάση το πιο εύκολα αναγνωρίσιμο χαρακτηριστικό μιας χρονοσειράς από την γραφική της αναπαράσταση. Επίσης εύκολος είναι και ο τρόπος αντιμετώπισης της επίδρασης της εποχιακότητας στα δεδομένα μιας χρονοσειράς καθώς οι αλλαγές που προκαλεί στα δεδομένα επαναλαμβάνονται κατά την πάροδο του χρόνου, στα ίδια χρονικά διαστήματα και με την ίδια ποσοστιαία αλλαγή. Η εποχικότητα ακριβώς λόγω της κανονικότητάς της, αντιμετωπίζεται με την εύρεση δεικτών εποχιακότητας για τα αντίστοιχα χρονικά

διαστήματα, η διαίρεση των οποίων με τα πραγματικά δεδομένα μας απαλλάσσει από την επίδραση της εποχιακότητας και μας επιτρέπει να παράγουμε μια νέα χρονοσειρά χωρίς εποχιακότητα που ονομάζεται αποεποχικοποιημένη χρονοσειρά.

Παραδείγματα εποχιακών χρονοσειρών είναι οι πωλήσεις παγωτών που εκτοξεύονται το καλοκαίρι ή ο αριθμός επιβατών πλοίων και αεροπλάνων που πολλαπλασιάζονται κατά τις περιόδους των γιορτών. Από τα παραπάνω καταλαβαίνουμε ότι η βασική διαφορά μεταξύ της εποχιακότητας και της κυκλικότητας ότι και τα δυο χαρακτηριστικά της χρονοσειράς μας δείχνουν επανάληψη κάποιου είδους μοτίβου, η μεν εποχιακότητα με συχνότητα μέρας, εβδομάδας ή μήνα ενώ η κυκλικότητα σε επίπεδο πενταετίας, δεκαετία ή ακόμα και αιώνα.

Οι ***μη κανονικές διακυμάνσεις*** ή αλλιώς ***ασυνέχειες***, είναι εκείνες οι παρατηρήσεις που εμφανίζονται στην γραφική απεικόνιση της χρονοσειράς ως απότομες αλλαγές στο πρότυπο συμπεριφοράς της. Τέτοιες αλλαγές, που έχουν είτε παροδικό είτε ακόμα και μόνιμο χαρακτήρα, δεν θα μπορούσαν να προβλεφθούν με χρήση αποκλειστικά των ιστορικών δεδομένων. Οι αλλαγές με παροδική διάρκεια που η επίδρασή τους διαρκεί για σύντομο χρονικό διάστημα ονομάζονται ***outliers*** ή ***special events.*** Η αναγνώριση τους δεν είναι μια απλή διαδικασία που θα μπορούσε να επιτύχει ένας απλός χρήστης καθώς απαιτείται τόσο θεωρητική γνώση του υπό μελέτη μεγέθους όσο και κριτική ικανότητα από την πλευρά του αναλυτή. Ένα **outlier** αποτελεί μια ασυνήθιστη παρατήρηση της χρονοσειράς που οφείλεται σε κάποιο εξαιρετικό ή απρόβλεπτο γεγονός. Παραδείγματος χάρη, μια απεργία μπορεί να προκαλέσει δραματική μείωση των παραγόμενων προϊόντων μιας επιχείρησης, ενώ μια διαφημιστική εκστρατεία μπορεί να αυξήσει τις πωλήσεις προϊόντων αυτής.

Από την άλλη πλευρά οι αλλαγές με μόνιμο χαρακτήρα στην χρονοσειρά που οι αλλαγές τους θα συνεχιστούν και στο μέλλον ονομάζονται ***level shifts.*** Ένα παράδειγμα αλλαγής τύπου **level shift** είναι η πτώση του επιπέδου των πωλήσεων μιας εταιρείας λόγω εισαγωγής στην αγορά μιας ανταγωνίστριας εταιρείας στον ίδιο κλάδο. Μετά από την απότομη μείωση των πωλήσεων κατά την είσοδο του ανταγωνισμού στην αγορά, οι πωλήσεις σταθεροποιούνται και πάλι απλά σε χαμηλότερο από το αρχικό επίπεδο.

Τέλος υπάρχει και η συνιστώσα της τυχαιότητας ή αλλιώς στοιχείο σφάλματος. Ως τυχαιότητα ορίζεται η διαφορά ανάμεσα στην συνδυασμένη επίδραση των τριών πρώτων συνιστωσών των χρονοσειρών (τάση, κυκλικότητα και εποχιακότητα) και των πραγματικών δεδομένων.

2.3. ΚΑΤΗΓΟΡΙΕΣ ΜΕΘΟΔΩΝ ΠΡΟΒΛΕΨΗΣ

Οι τρεις μεγάλες κατηγορίες στις οποίες εντάσσονται οι τεχνικές προβλέψεων που έχουν αναπτυχθεί μέχρι σήμερα είναι οι εξής:

- ***Ποσοτικές*** (quantitative)
- ***Κριτικές*** (judgmental)
- ***Τεχνολογικές*** (technological)

Στα πλαίσια της μεταπτυχιακής διπλωματικής εργασίας μας θα ασχοληθούμε με την εφαρμογή ποσοτικών μεθόδων και ειδικότερα μεθόδων χρονοσειρών, αλλά για χάρη πληρότητας πρέπει να αναφερθούν όλες οι μέθοδοι προβλέψεων.

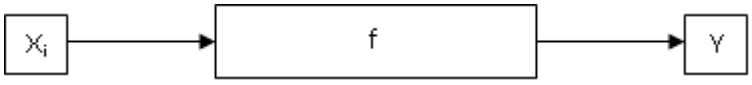
2.3.1. ΠΟΣΟΤΙΚΕΣ ΜΕΘΟΔΟΙ

Για την εφαρμογή των ποσοτικών μεθόδων πρόβλεψης υπάρχει η απαίτηση η πληροφορία που θέλουμε να προβλέψουμε να ποσοτικοποιείται με την μορφή αριθμητικών δεδομένων και τα ιστορικά δεδομένα να διατηρούν το πρότυπο συμπεριφοράς τους στο μέλλον. Οι ποσοτικές μέθοδοι ανάλογα με το μοντέλο που χρησιμοποιείται μπορεί να ταξινομηθούν στα μοντέλα χρονοσειρών, τα οποία και χρησιμοποιούμε στις μεθόδους μας και τα αιτιοκρατικά μοντέλα. Για να κατανοήσουμε ευκολότερα τις βασικές υποθέσεις πάνω στις οποίες στηρίζεται κάθε μία από τις δυο ποσοτικές μεθόδους καθώς και για να εντοπίσουμε τα πλεονεκτήματα και τα μειονεκτήματά τους θα πρέπει να μελετήσουμε τις ιδιότητες και τα χαρακτηριστικά της καθεμίας.

2.3.1.1 ΜΕΘΟΔΟΙ ΧΡΟΝΟΣΕΙΡΩΝ

Οι μέθοδοι χρονοσειρών αποτελούν το πιο διαδομένο είδος ποσοτικού μοντέλου πρόβλεψης, ειδικά στον κλάδο των επιχειρήσεων. Το μοντέλο αυτό βασίζεται στην υπόθεση ότι η μεταβολή της τιμής του υπό μελέτη μεγέθους ακολουθεί ένα λανθάνον πρότυπο που επαναλαμβάνεται στο χρόνο και παραμένει όσο το δυνατόν πιο σταθερό χωρίς την εμφάνιση ασυνήθιστων τιμών. Το λανθάνον αυτό πρότυπο υποθέτουμε ότι αναγνωρίζεται μονοσήμαντα με βάση τα δεδομένα. Οι προβλέψεις, λοιπόν, παράγονται με την αναγνώριση του προτύπου αυτού και την προέκταση του στο μέλλον. Έτσι βασιζόμενοι στις παρελθούσες τιμές της υπό εξέταση μεταβλητής, ανιχνεύουμε το πρότυπο το οποίο ακολουθούν και παράγουμε σημειακές προβλέψεις επεκτείνοντας το πρότυπο αυτό στο μέλλον. Για την υλοποίηση της διαδικασίας αυτής πρέπει να έχουμε διαθέσιμα, ικανοποιητικά σε αριθμό ιστορικά στοιχεία της μεταβλητής σε σταθερές χρονικές περιόδους. Όσο μεγαλύτερη ιστορικότητα έχουμε στην διάθεσή μας τόσο καλύτερα θα εντοπίσουμε το πρότυπο συμπεριφοράς της χρονοσειράς και θα παράγουμε ακριβέστερες προβλέψεις.

Το μοντέλο χρονοσειρών μπορεί να παρασταθεί σχηματικά με το απλό σχήμα που φαίνεται στην Εικόνα 3:



Εικόνα 3: Το μοντέλο χρονοσειρών

Στο παραπάνω σύστημα η είσοδος του συστήματος X_i (όπου i είναι η αντίστοιχη χρονική περίοδος) αναπαριστά τα ιστορικά δεδομένα της μελετώμενης μεταβλητής. Η έξοδος Y είναι η τελική παραγόμενη πρόβλεψη και η συνάρτηση f είναι το μοντέλο πρόβλεψης που χρησιμοποιούμε.

2.3.1.2 ΜΕΘΟΔΟΙ ΑΠΟΣΥΝΘΕΣΗΣ

Αντικείμενο των μεθόδων αποσύνθεσης είναι ο διαχωρισμός κύριων χαρακτηριστικών των χρονοσειρών και η απομόνωσή τους. Τα κύρια αυτά χαρακτηριστικά των χρονοσειρών όπως έχουν ήδη αναφερθεί στην παράγραφο 2.2.4 είναι: η **τάση**, ο **κύκλος**, η **εποχιακότητα** και η **τυχαιότητα**. Οι κυριότερες μέθοδοι αποσύνθεσης από τη βιβλιογραφία είναι:

- Fixed Additive Method – σταθερή προσθετική μέθοδος
- Fixed Multiplicative Method – σταθερή πολλαπλασιαστική μέθοδος ή κλασική μέθοδος αποσύνθεσης.
- Moving Additive Method – κινητή προσθετική μέθοδος
- Moving Multiplicative Method – κινητή πολλαπλασιαστική μέθοδος
- Zaycoff's Method
- Μέθοδος Cwnsus X-II
- CPB Method
- KVF Method
- SABL Method

Σε κάποιες μεθόδους που εφαρμόζουμε στην εργασία, έχουν ενσωματωθεί κάποια από τα βήματα της κλασικής μεθόδου αποσύνθεσης, με σκοπό την εύρεση των δεικτών εποχιακότητας της χρονοσειράς και της αποεποχικοποίησής της. Αυτό γίνεται γιατί έχει δειχθεί ότι είναι πιο εύκολο και δίνει και πιο ακριβείς προβλέψεις η προέκταση μιας αποεποχικοποιημένης χρονοσειράς και έπειτα ή επαναεποχικοποίηση των παραγόμενων προβλέψεων ώστε να προκύψουν οι τελικές προβλέψεις.

2.3.1.3 ΜΕΘΟΔΟΙ ΕΞΟΜΑΛΥΝΣΗΣ

Οι μέθοδοι εξομάλυνσης, χρησιμοποιούνται εν γένει για βραχυπρόθεσμες προβλέψεις των μελλοντικών τιμών της σειράς και σκοπός των μεθόδων αυτών είναι να διακρίνουν το βασικό πρότυπο, εξομαλύνοντας τα ιστορικά δεδομένα. Οι μέθοδοι αυτές διακρίνονται στις μεθόδους κινητού μέσου όρου, στις οποίες οι παρελθούσες τιμές της μεταβλητής συμμετέχουν με την ίδια βαρύτητα στον υπολογισμό της πρόβλεψης και στις μεθόδους εκθετικής εξομάλυνσης, όπου χρησιμοποιούνται διαφορετικοί συντελεστές βαρύτητας για τα ιστορικά δεδομένα οι οποίοι φθίνουν με εκθετικό τρόπο από την πιο πρόσφατη τιμή των δεδομένων ως την πιο μακρινή.

2.3.1.4 ΑΥΤΟΠΑΛΙΝΔΡΟΜΙΚΕΣ ΜΕΘΟΔΟΙ ΚΙΝΗΤΟΥ ΜΕΣΟΥ ΟΡΟΥ (ARIMA)

Οι αυτοπαλινδρομικές μέθοδοι κινητού μέσου όρου είναι στοχαστικά μαθηματικά μοντέλα τα οποία χρησιμοποιούνται για την περιγραφή της διαχρονικής εξέλιξης κάποιου φυσικού μεγέθους. Τα στοχαστικά μοντέλα περιέχουν τον τυχαίο παράγοντα, τις τιμές του μεγέθους για τις προηγούμενες χρονικές στιγμές όπως και άλλους στοχαστικούς παράγοντες συνήθως. Το μοντέλο που προκύπτει τελικά είναι ένας γραμμικός συνδυασμός των παραπάνω ποσοτήτων. Τα αυτοπαλινδρομικά μοντέλα βασίζονται στην παραδοχή της αλληλεξάρτησης μεταξύ των τιμών που λαμβάνει η χρονοσειρά τις διάφορες χρονικές στιγμές.

2.3.1.5 ΕΠΕΞΗΓΗΜΑΤΙΚΕΣ (ΑΙΤΙΟΚΡΑΤΙΚΕΣ) ΜΕΘΟΔΟΙ

Στις επεξηγηματικές μεθόδους αναγνωρίζονται μεταβλητές οι οποίες σχετίζονται με την σειρά δεδομένων που υπάρχει και βάσει αυτών των μεταβλητών αναπτύσσεται κάποιο μοντέλο για να εκφράσει την σχέση αυτή. Δεν είναι απαραίτητα κάποια χρονική εξάρτηση καθώς η πρόβλεψη εκφράζεται ως συνάρτηση του συγκεκριμένου αριθμού παραγόντων που έχει αναγνωριστεί ότι επηρεάζουν τις μελλοντικές τιμές. Η ανάπτυξη μία τέτοιας μεθόδου διευκολύνει την κατανόηση των συνθηκών και επιτρέπει τον πειραματισμό με διάφορους συνδυασμούς δεδομένων για τη βαθύτερη μελέτη των επιδράσεων τους στην τελική πρόβλεψη. Στις επεξηγηματικές μεθόδους ανήκουν οι μέθοδοι παλινδρόμησης όπως και οι οικονομετρικές μέθοδοι.

2.3.2. ΚΡΙΤΙΚΕΣ ΜΕΘΟΔΟΙ

Οι στατιστικές μέθοδοι εντάσσονται στην κατηγορία των ποσοτικών μεθόδων και επιτρέπουν γενικά την αναγνώριση κάποιων προτύπων ή σχέσεων που διακρίνονται στις χρονοσειρές, με στόχο την προέκταση των χρονοσειρών αυτών για εύρεση μελλοντικών τους τιμών. Βασική υπόθεση που γίνεται όμως, είναι ότι θα συνεχιστεί να ισχύει το συγκεκριμένο πρότυπο ή σχέση που έχει παρατηρηθεί. Όμως στην πραγματική ζωή, αλλαγές συμβαίνουν διαρκώς και όσο γρηγορότερα αναγνωριστούν τόσο πιο πιθανή είναι η αποφυγή μεγάλου και συχνά ακριβού λάθους στις προβλέψεις. Όταν λοιπόν τέτοιες αλλαγές αναγνωριστούν, τότε εισέρχεται στις μεθόδους πρόβλεψης η ανθρώπινη κριτική ικανότητα. Η ικανότητα αυτή είναι η μόνη βιώσιμη εναλλακτική για να προβλέπει τόσο την έκταση όσο και την επίδραση των αλλαγών αυτών στις προβλέψεις. Επίσης είναι απαραίτητη για να μπορεί να ενσωματωθεί στις πληροφορίες η εμπειρία και η γνώση των managers όπως επίσης και των experts. Συνοψίζοντας, οι κριτικές μέθοδοι έχουν εν γένει ως δεδομένα, προϊόντα διαίσθησης, κρίσης και συσσωρευμένης γνώσης και χρησιμοποιούνται σε επιχειρήσεις και οργανισμούς. Η πρόβλεψη μπορεί να βασίζεται είτε στις γνώσεις και

την κρίση ενός ατόμου (ατομικές μέθοδοι) είτε στο συνδυασμό απόψεων των μελών κάποιας επιτροπής (μέθοδοι επιτροπής).

2.3.3. ΤΕΧΝΟΛΟΓΙΚΕΣ ΜΕΘΟΔΟΙ

Οι τεχνολογικές μέθοδοι πρόβλεψης χρησιμοποιούνται για μακροπρόθεσμες προβλέψεις σχετικά με τεχνολογικά, οικονομικά, κοινωνικά και πολιτικά θέματα. Διακρίνονται σε δύο κατηγορίες: στις **διερευνητικές (exploratory)** και στις **κανονιστικές (normative).** Οι διερευνητικές μέθοδοι ξεκινούν από το παρελθόν ή το παρόν και εξετάζοντας όλες τις πιθανές περιπτώσεις οδηγούνται στο μέλλον. Στον αντίποδα, υπάρχουν οι κανονιστικές μέθοδοι που πρώτα καθορίζουν όλους τους μελλοντικούς στόχους και έπειτα εξετάζουν τη δυνατότητα επίτευξης τους λαμβάνοντας υπ όψιν τους περιορισμούς, τους διαθέσιμους πόρους αλλά και τις τεχνολογίες.

2.4. ΚΥΡΙΟΤΕΡΕΣ ΜΕΘΟΔΟΙ ΠΡΟΒΛΕΨΗΣ

Στην παρούσα μεταπτυχιακή εργασία, χρησιμοποιήσαμε έξι βασικές μεθόδους πρόβλεψης χρονοσειρών, οι οποίες είναι οι εξής:

1. Naive ή Απλοϊκή Μέθοδο
2. Απλή Γραμμική Παλινδρόμηση – LRL
3. Απλή Εκθετική Εξομάλυνση – SES
4. Εκθετική Εξομάλυνση Γραμμικής Τάσης – Holt
5. Εκθετική Εξομάλυνση Μη Γραμμικής Τάσης – Damped
6. Κλασική Μέθοδο Theta

Στις παρακάτω παραγράφους θα αναπτυχθούν αναλυτικά τα μοντέλα αυτά, αλλά και για χάρη πληρότητας και μερικά άλλα τα οποία δεν υλοποιήθηκαν στα πλαίσια της εργασίας.

2.4.1. ΑΠΛΟΙΚΗ ΜΕΘΟΔΟΣ (NAIVE)

Η μέθοδος **Naive** αποτελεί την πιο απλή μέθοδο πρόβλεψης. Η συγκεκριμένη μέθοδος δίνει ως πρόβλεψη για την επόμενη χρονική περίοδο την ίδια τιμή με την παρατήρηση που είχε σημειωθεί την προηγούμενη ακριβώς χρονική περίοδο. Έχει καλή απόδοση για προβλέψεις μίας περιόδου μπροστά σε αποεποχικοποιημένες χρονοσειρές καθώς η αναμενόμενη τιμή της πρόβλεψης δεν διαφέρει σημαντικά από την τελευταία παρατήρηση που είναι διαθέσιμη. Συνήθως όμως, δεν παράγει ακριβείς προβλέψεις, με αποτέλεσμα να μην χρησιμοποιείται τόσο ως μέθοδος πρόβλεψης, όσο ως σημείο

αναφοράς (benchmark) για άλλες, πιο πολύπλοκες μεθόδους. Η μαθηματική σχέση που περιγράφει αυτή τη μέθοδο πρόβλεψης είναι:

$$F_{\mathrm{t}} = Y_{t-1}$$

όπου Ft είναι η πρόβλεψη την χρονική στιγμή t και Yt-1 η πραγματική τιμή της χρονικής στιγμής t-1 (αμέσως προηγούμενη).

2.4.2. ΜΕΘΟΔΟΙ ΚΙΝΗΤΟΥ ΜΕΣΟΥ ΟΡΟΥ

Οι μέθοδοι μέσων όρων, εκτός από την χρησιμότητά τους για την εξομάλυνση των ιστορικών δεδομένων και κατά συνέπεια την ομαλοποίηση των χρονοσειρών, μπορούν να χρησιμοποιηθούν και ως μέθοδοι πρόβλεψης. Κάποιοι από τους μέσους όρους που χρησιμοποιούμε για την παραγωγή προβλέψεων είναι ο απλός μέσος όρος και ο κινητός μέσος όρος.

2.4.2.1 ΑΠΛΟΣ ΜΕΣΟΣ ΟΡΟΣ

Η μέθοδος του απλού μέσου όρου στηρίζεται στην εύρεση του μέσου όρου όλων των παρατηρήσεων και στη χρήση αυτής της τιμής για πρόβλεψη. Συνεπώς η πρόβλεψη δίνεται βάσει της παρακάτω σχέσης:

$$F_{t+1} = \frac{1}{t} \cdot \sum_{i=1}^{t} Y_i$$

Η χρήση αυτής της μεθόδου ενδείκνυται για περιπτώσεις που οι παρατηρήσεις δεν παρουσιάζουν τάση ή αξιοπρόσεκτη εποχιακότητα. Εν γένει προτείνεται για χρονοσειρές που παρουσιάζουν σταθερότητα στην πάροδο του χρόνου, λόγω του μεγάλου όγκου ιστορικών δεδομένων που ισοβαρώς συμπεριλαμβάνονται στον υπολογισμό της μεθόδου.

2.4.2.2 ΚΙΝΗΤΟΣ ΜΕΣΟΣ ΟΡΟΣ

Ένας τρόπος να διαχειριστεί κανείς την επιρροή των παρελθουσών παρατηρήσεων στην πρόβλεψη, όταν έχει επιλεχθεί ως μέθοδος πρόβλεψης αυτή των μέσων όρων, είναι να καθοριστεί το μήκος του μέσου όρου των παρατηρήσεων που θα ληφθούν υπ’ όψη στην εξαγωγή της πρόβλεψης. Έτσι το μοντέλο καθώς μια νέα παρατήρηση

γίνεται διαθέσιμη θα ανανεώνεται με αποτέλεσμα να γίνεται πιο ακριβές αφού λαμβάνει υπόψη του δεδομένα που βρίσκονται πιο κοντά στο παρόν. Η επιλογή του μήκους του μέσου όρου ο οποίος θα υπολογιστεί και η τιμή του που θα οριστεί ως πρόβλεψη μετατρέπει τον απλό μέσο όρο σε κινητό μέσο όρο.

Ο όρος κινητός μέσος όρος χρησιμοποιείται για να περιγράψει τη διαδικασία καθώς όταν μία νέα παρατήρηση γίνεται διαθέσιμη, τότε υπολογίζεται ο νέος μέσος όρος των τελευταίων παρατηρήσεων του συγκεκριμένου μήκους που έχει επιλεχθεί. Αυτός ο νέος μέσος όρος θα είναι η τιμή της πρόβλεψης που παράγεται από αυτήν την μέθοδο για την επόμενη χρονική περίοδο. Είναι σημαντικό να τονιστεί ότι το πλήθος των παρατηρήσεων που χρησιμοποιούνται για την εξαγωγή του μέσου όρου παραμένει σταθερό καθ' όλη τη διαδικασία πρόβλεψης και περιλαμβάνει πάντα τις πιο πρόσφατες παρατηρήσεις. Η σχέση που χρησιμοποιείται για την εφαρμογή της μεθόδου του κινητού μέσου όρου, οποίος συμβολίζεται: ΚΜΟ(k), όπου k το μήκος, είναι:

$$F_{t+1} = \frac{1}{\kappa} \cdot \sum_{i=t-k+1}^{t} Y_i$$

Μία ενδεχόμενη σύγκριση του κινητού μέσου όρου και του απλού μέσου όρου θα παρουσίαζε ενδιαφέρον για να γίνει αντιληπτή η διαφορά στην χρήση τους. Το πλεονέκτημα του κινητού μέσου όρου είναι η σημασία που δίνεται πάντα στις τελευταίες ίσου πλήθους παρατηρήσεις, ενώ τα μειονεκτήματα που παρουσιάζει είναι ότι απαιτεί περισσότερο χώρο αποθήκευσης δεδομένων, διότι πρέπει να αποθηκευτούν όλες οι παρατηρήσεις από τις οποίες θα εξάγεται κάθε φορά ο μέσος όρος και όχι απλά η τιμή του μέσου όρου.

Το μειονέκτημα των μεθόδων μέσων όρων είναι ότι καμία από τις δύο μεθόδους δεν μπορεί να διαχειριστεί με επιτυχία κύρια χαρακτηριστικά των χρονοσειρών όπως είναι η τάση και η εποχιακότητα κατά την εφαρμογή τους για την παραγωγή προβλέψεων. Για αυτό το λόγο τα μοντέλα μέσων όρων χρησιμοποιούνται κυρίως για την εξάλειψη της εποχιακότητας και της τυχαιότητας από τις χρονοσειρές, ώστε να προκύψει μια εκτίμηση της γραμμής τάσης-κύκλου. Έτσι λοιπόν τα μοντέλα αυτά θα μπορούσαμε να πούμε ότι είναι κατά κύριο λόγο, εργαλεία αποσύνθεσης και όχι εργαλεία πρόβλεψης.

2.4.3. ΑΠΛΗ ΓΡΑΜΜΙΚΗ ΠΑΛΙΝΔΡΟΜΗΣΗ

Η μέθοδος της απλής γραμμικής παλινδρόμησης βασίζεται στην υπόθεση ύπαρξης σχέσης ανάμεσα στη μεταβλητή πρόβλεψης (εξαρτημένη μεταβλητή) και σε μια άλλη μεταβλητή (ανεξάρτητη μεταβλητή). Εκτός από την υπόθεση ότι υπάρχει μια τέτοια σχέση, υποθέτουμε ότι η σχέση αυτή είναι και γραμμική. Σκοπός λοιπόν της απλής γραμμικής παλινδρόμησης είναι η έκφραση της σχέσης της μεταβλητής Y από μία ανεξάρτητη μεταβλητή X με την εξίσωση μιας ευθείας γραμμής:

$$Y = a + bX + e$$

Όπου α είναι το αρχικό σημείο (για b= 0) και b είναι η κλίσης της ευθείας, ενώ ο όρος e, δηλώνει το σφάλμα, δηλαδή την απόκλιση της παρατήρησης από της ευθεία που παριστάνεται από την παραπάνω σχέση.

Στόχος της απλής γραμμικής παλινδρόμησης είναι η εκτίμηση των παραμέτρων α και β έτσι ώστε η ευθεία:

$$Y = a + bX$$

να αποτελεί τη "βέλτιστη", δηλαδή να προσαρμόζεται όσο το δυνατόν καλύτερα στα δεδομένα. Το σφάλμα προσαρμογής μπορεί να θεωρηθεί σαν την κατακόρυφη απόκλιση της παρατήρησης από την ευθεία προσαρμογής και δίδεται ως εξής:

$$e_t = Y_i - \widehat{Y}_i$$

όπου η τιμή $\widehat{Y}_i$ αντιπροσωπεύει την εκτιμώμενη τιμή (από της ευθεία παλινδρόμησης) και η τιμή Y_i αντιστοιχεί στην πραγματική παρατήρηση. Σαν βέλτιστη ευθεία προσαρμογής, επιλέγεται αυτή για την οποία το άθροισμα των τετραγώνων των σφαλμάτων γίνεται ελάχιστο. Η μέθοδος είναι γνωστή σαν μέθοδο Ελαχίστων Τετραγώνων.

Η μαθηματική σχέση από την οποία προκύπτει το σφάλμα μπορεί να γραφτεί συναρτήσει των α και b ως εξής:

$$\sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (Y_i - \widehat{Y}_i)^2 = \sum_{i=1}^{n} (Y_i - a - bX_i)^2$$

οπότε και είναι δυνατός ο υπολογισμός των παραμέτρων a και b που δίνουν την εξίσωση βέλτιστης ευθείας.

Οι μαθηματικές σχέσεις που προκύπτουν είναι:

$$b = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n} (X_i - \widehat{X}_i)^2}$$

$$a = \bar{Y} - b\bar{X}$$

Όπου $\bar{X}$ και $\bar{Y}$ οι μέσες τιμές των διανυσμάτων X και Y και n ο αριθμός των παρατηρήσεων με βάση τις οποίες υπολογίζεται η ευθεία παλινδρόμησης.

2.4.3.1 ΚΩΔΙΚΑΣ JAVA

```
package metaptyxiakh;

import static java.lang.Math.pow;
import java.util.ArrayList;

public class LinearRegression {

    public Double[] lrl (ArrayList <Double> data) {

        Double[] coefficients = new Double[2];
        Double xMean = 0.0;
        Double xSquareMean = 0.0;
        Double yMean = 0.0;
        Double xyMean = 0.0;
        Double a = 0.0;
        Double b = 0.0;

        for (int i=0; i<data.size(); i++) {

        xMean = xMean + i + 1;
                                          }
        xMean = xMean/data.size();


        for (int i=0; i<data.size(); i++) {

        xSquareMean = xSquareMean + pow(i + 1,2);
                                          }
        xSquareMean = xSquareMean/data.size();


        for (int i=0; i<data.size(); i++) {

        yMean = yMean + data.get(i);
                                          }
        yMean = yMean/data.size();


        for (int i=0; i<data.size(); i++) {

        xyMean = xyMean + (i+1)*data.get(i);
                                          }
        xyMean = xyMean/data.size();


        b = (xyMean - xMean*yMean) / (xSquareMean - pow(xMean,2));
        a = yMean - b*xMean;

        coefficients[0] = a;
        coefficients[1] = b;
```

```
        return coefficients;
    }

}
```

Εικόνα 4: Κώδικας Java για απλή γραμμική παλινδρόμηση

Ο παραπάνω κώδικας δέχεται σαν είσοδο το σύνολο των δεδομένων και επιστρέφει σαν έξοδο τους συντελεστές a και b της απλής γραμμικής παλινδρόμησης σύμφωνα με τους τύπους της παραγράφου 2.4.3.

2.4.4. ΜΕΘΟΔΟΙ ΕΚΘΕΤΙΚΗΣ ΕΞΟΜΑΛΥΝΣΗΣ

Μία επέκταση των μεθόδων μέσου όρου, είναι οι μέθοδοι πρόβλεψης με σταθμισμένο μέσο όρο, δηλαδή όλες οι παρατηρήσεις να μην έχουν τη ίδια βαρύτητα για την εξαγωγή των προβλέψεων. Είναι συχνό φαινόμενο, οι πιο πρόσφατες παρατηρήσεις να είναι καλύτερος οδηγός για την πρόβλεψη της μελλοντικής τιμής. Γι' αυτόν τον λόγο, δημιουργήθηκε η ανάγκη για μοντέλα πρόβλεψης που θα χρησιμοποιούν τις παλαιότερες παρατηρήσεις με μειωμένη βαρύτητα συγκριτικά με τις πιο πρόσφατες.

Σε αυτήν την παράγραφο λοιπόν θα περιγραφούν μέθοδοι οι οποίες εφαρμόζουν εκθετική μείωση του συντελεστή βαρύτητας όσο πιο παλαιά είναι μια παρατήρηση. Αυτός είναι και ο λόγος που αυτές οι μέθοδοι καλούνται μέθοδοι εκθετικής εξομάλυνσης.

Γενικότερα, η εκθετική εξομάλυνση είναι μέθοδος πρόβλεψης, η οποία προεκτείνει στοιχεία του προτύπου των ιστορικών δεδομένων στο μέλλον. Το μοντέλο της αντίστοιχης πρόβλεψης εφαρμόζεται στη δοθείσα χρονοσειρά αφού πρώτα τα αντίστοιχα δεδομένα έχουν εξομαλυνθεί έτσι ώστε να απομονωθούν τα πραγματικά πρότυπα από τις καθαρά τυχαίες διακυμάνσεις.

Οι μέθοδοι εκθετικής εξομάλυνσης παρουσιάστηκαν για πρώτη φορά γύρω στο 1940 και η άνθισή τους ήρθε το 1960 μαζί με την άνθιση της επιστήμης της πληροφορικής. Οι μέθοδοι εκθετικής εξομάλυνσης είναι ιδιαίτερα δημοφιλείς στο πεδίο των προβλέψεων λόγω της απλότητας τους, των περιορισμένων απαιτήσεων τους για αποθήκευση δεδομένων και του μειωμένου υπολογιστικού φόρτου που απαιτούν. Επίσης παρά την απλότητα που τις διακρίνει, σύμφωνα με αποτελέσματα πρακτικών μελετών, παρουσιάζουν ικανοποιητικά ποσοστά ακρίβειας σε σχέση με πιο πολύπλοκες μεθόδους, διότι δεν επηρεάζονται από τις ιδιομορφίες των προτύπων των δεδομένων ούτε από τυχαία εμφανιζόμενες ακραίες τιμές. Βέβαια αντίθετα με την ακρίβεια των προβλέψεων, οι εμπειρικές μελέτες έδειξαν ότι τα μοντέλα γραμμικής και εκθετικής τάσης εμφανίζουν μια υπεραισιοδοξία με αποτέλεσμα να οδηγούν σε ιδιαίτερα υψηλές τιμές προβλέψεων και κατ΄ επέκταση σε αυξημένες τιμές του στατιστικού δείκτη Mean Error.

Τα μοντέλα εκθετικής εξομάλυνσης χωρίζονται σε κατηγορίες ανάλογα με τη γενική μορφή της γραφικής παράστασης της χρονοσειράς. Σύμφωνα λοιπόν με την κατηγοριοποίηση αυτή προκύπτουν τέσσερα μοντέλα τάσης: τα **μοντέλα σταθερού επιπέδου, γραμμικής τάσης, εκθετικής τάσης** και **φθίνουσας τάσης**.

Πριν την ανάλυση κάθε μοντέλου, κρίνεται σκόπιμο να αναφερθούν κάποια γενικά χαρακτηριστικά τους συνοπτικά και για τα τέσσερα μοντέλα έτσι ώστε να υπάρχει μία γενική εικόνα για τη χρήση του καθενός.

Αρχικά λοιπόν, το **μοντέλο σταθερού επιπέδου** υποθέτει την απουσία τάσης από τα δεδομένα. Εν γένει χρησιμοποιείται για τις προβλέψεις ενός βήματος, διότι η πρόβλεψη για οποιαδήποτε μελλοντική χρονική στιγμή γίνεται με την προέκταση μιας οριζόντιας ευθείας γραμμής. Αντιθέτως, το μοντέλο γραμμικής τάσης είναι πρακτικά πιο διαδεδομένο διότι η πρόβλεψη γίνεται με προέκταση μίας ευθείας γραμμής συμπεριλαμβανομένης την ύπαρξης της τάσης με αυτόν τον τρόπο. Χαρακτηριστικά παραδείγματα που ενδείκνυται η εφαρμογή μοντέλου εκθετικής τάσης είναι το ποσοστό αύξησης των πωλήσεων στην αρχή του κύκλου ζωής ενός προϊόντος. Λόγω του προβλήματος υπεραισιοδοξίας των μοντέλων γραμμικής και εκθετικής τάσης που αναφέραμε, προέκυψε η ανάγκη εφαρμογής ενός άλλου μοντέλου, όπως του μοντέλου φθίνουσας τάσης, το οποίο αποτελεί την καλύτερη προσέγγιση σχετικά με τις μακροχρόνιες προβλέψεις, διότι μειώνεται σταδιακά το μέγεθος κατά το οποίο αυξάνονται οι τιμές της χρονοσειράς κάθε χρονική περίοδο. Πιο αναλυτικά ακολουθεί μελέτη και παρουσίαση κάθε μοντέλου για την καλύτερη κατανόηση του.

2.4.4.1 ΜΟΝΤΕΛΟ ΣΤΑΘΕΡΟΥ ΕΠΙΠΕΔΟΥ – ΑΠΛΗ ΕΚΘΕΤΙΚΗ ΕΞΟΜΑΛΥΝΣΗ (SIMPLE EXPONENTIAL SMOOTHING – SES)

Το μοντέλο σταθερού επιπέδου περιγράφεται από τις εξής εξισώσεις:

$$e_t = Y_t - \mathrm{F_t}$$
$$S_t = S_{t-1} + \mathrm{a} \cdot e_t$$
$$\mathrm{F_{t+1}} = S_t$$

όπου e_t είναι το σφάλμα της πρόβλεψης το οποίο προκύπτει από τη διαφορά της πραγματικής τιμής της χρονοσειράς και της πρόβλεψης για την ίδια χρονική περίοδο t. Ο δείκτης t λοιπόν, αντιπροσωπεύει την χρονική περίοδο. Το S_t είναι το επίπεδο της χρονοσειράς στο τέλος της χρονικής περιόδου t και είναι το επίπεδο της προηγούμενης χρονικής περιόδου και ενός ποσοστού του σφάλματος. Ιδιαίτερο ενδιαφέρον παρουσιάζει η τιμή του ποσοστού αυτού διότι αντιπροσωπεύει την τιμή του συντελεστή εξομάλυνσης και λαμβάνει τιμές από 0 έως 1. Τέλος η τιμή $F_t(m)$ είναι η πρόβλεψη που πραγματοποιείται στο τέλος κάθε περιόδου t και αναφέρεται σε m περιόδους μπροστά. Χαρακτηριστικό της μεθόδου σταθερού επιπέδου είναι ότι η πρόβλεψη για κάθε χρονική περίοδο είναι ίση με το επίπεδο S_t.

Αν θέλαμε να περιγράψουμε με λίγες λέξεις την λειτουργία του μοντέλου σταθερού επιπέδου θα είχαμε την παρακάτω διαδικασία. Σε κάθε χρονική στιγμή, υπολογίζεται το σφάλμα με σκοπό να κρατήσει την τιμή της πρόβλεψης αρκετά κοντά στο επίπεδο της πραγματικής χρονοσειράς. Για την παραγωγή της πρόβλεψης κάθε χρονική στιγμή, πρέπει να έχει υπολογιστεί η τιμή του επιπέδου από την πραγματική χρονοσειρά για την προηγούμενη χρονική στιγμή.

Η ερώτηση που προκύπτει από την λειτουργία του μοντέλου είναι, τι θα γίνει με την πρόβλεψη για την πρώτη χρονική περίοδο για την οποία δεν υπάρχουν ιστορικά δεδομένα. Η απάντηση δίνεται από το αρχικό επίπεδο, σαν πρώτη πρόβλεψη στο συγκεκριμένο μοντέλο χρησιμοποιείται το αρχικό επίπεδο. Είναι λοιπόν δεδομένη η σημασία της σωστής επιλογής του αρχικού επιπέδου του μοντέλου πρόβλεψης για την παραγωγή προβλέψεων με ακρίβεια.

Αρχικό Επίπεδο

Συνήθεις μεθοδολογίες για την πρώτη τιμή του επιπέδου της απλής εκθετικής εξομάλυνσης σταθερού επιπέδου είναι:

- Ο μέσος όρος όλων των παρατηρήσεων
- Ο μέσος όρος των ν πρώτων παρατηρήσεων
- Η πρώτη παρατήρηση
- Το σταθερό επίπεδο από το μοντέλο της απλής γραμμικής παλινδρόμησης

Η τιμή της αρχικοποίησης του επιπέδου για την εφαρμογή του μοντέλου, αφήνεται στο ερευνητή και εξαρτάται και από τα χαρακτηριστικά της εκάστοτε χρονοσειράς.

Συντελεστής Εξομάλυνσης

Όπως αναφέραμε, η παράμετρος a αποτελεί το συντελεστή εξομάλυνσης της μεθόδου. Οι τιμές που λαμβάνει ανήκουν στο διάστημα 0 έως 1 και εν γένει το κριτήριο που χρησιμοποιείται για τον προσδιορισμό της είναι η ελαχιστοποίηση του μέσου τετραγωνικού σφάλματος (MSE). Το κριτήριο αυτό ως επικρατέστερο στην βιβλιογραφία, έχει χρησιμοποιηθεί και από εμάς κατά τα στάδια υλοποίησης των μεθόδων. Βέβαια η τιμή της παραμέτρου δεν είναι μοναδική καθώς μπορεί να επιλεχθούν άλλα κριτήρια για την επιλογή της αντίστοιχης τιμής όπως είναι η ελαχιστοποίησης κάποιου άλλου σφάλματος.

Η εύρεση της τιμής της σταθεράς εξομάλυνσης βρίσκεται με αλγοριθμικό τρόπο είτε με γραμμική αναζήτηση ψάχνοντας το ελάχιστο σφάλμα είτε με κάποια άλλη βελτιστοποίηση. Η γραμμική αναζήτηση εν γένει έχει το πρόβλημα χρόνου ιδιαίτερα όταν οι χρονοσειρές που αναφερόμαστε έχουν πολλά δεδομένα.

Ο βέλτιστος συντελεστής εξομάλυνσης καθορίζεται από δύο κύριους παράγοντες οι οποίοι αλληλοεξαρτώνται. Ο πρώτος είναι το ποσοστό θορύβου που υπάρχει στην χρονοσειρά. Όσο περισσότερος θόρυβος υπάρχει στα δεδομένα της χρονοσειράς τόσο

μικρότερη πρέπει να είναι η τιμή του συντελεστή εξομάλυνσης για να αποφευχθεί η υπερβολική αντίδραση στον θόρυβο. Ο άλλος παράγοντας είναι η σταθερότητα του μέσου όρου της χρονοσειράς. Αν ο μέσος όρος της χρονοσειράς μεταβάλλεται, ο συντελεστής εξομάλυνσης θα πρέπει να είναι μεγάλος έτσι ώστε οι προβλέψεις να παρακολουθούν τις αντίστοιχες μεταβολές των δεδομένων. Αντίθετα αν ο μέσος όρος είναι σχετικά σταθερός, τότε η τιμή του συντελεστή εξομάλυνσης θα είναι μικρή.

Οι ακραίες τιμές του συντελεστή εξομάλυνσης έχουν καθοριστική σημασία για την παραγωγή της πρόβλεψης. Μηδενική τιμή του συντελεστή εξομάλυνσης σημαίνει ότι η πρόβλεψη θα μείνει ίδια για όλες τις χρονικές περιόδους και ίση με το αρχικό επίπεδο. Γι αυτό εν γένει χρησιμοποιείται ένα κατώτατο όριο για την τιμή της παραμέτρου έτσι ώστε να αποφεύγεται η μηδενική τιμή. Αντίθετα, η μέγιστη τιμή του συντελεστή εξομάλυνσης, δηλαδή να είναι ίσος με τη μονάδα, οδηγεί την τιμή της πρόβλεψης να ταυτίζεται κάθε φορά με την τιμή της προηγούμενης χρονικής περιόδου, δηλαδή να είναι ίση με την τελευταία τιμή της χρονοσειράς και έτσι το μοντέλο μετατρέπεται στην απλοϊκή μέθοδο (Naive).

Κώδικας Java

```
package metaptyxiakh;

import java.util.ArrayList;

public class SimpleExponentialSmoothing {


    public ArrayList <Double> ses (ArrayList <Double> data, Double a,
Double initialLevel, Integer m) {

            ArrayList <Double> levels = new ArrayList();
            ArrayList <Double> forecasts = new ArrayList();
            ArrayList <Double> errors = new ArrayList();

            levels.add(initialLevel);

            for (int i=0; i<data.size(); i++) {
                forecasts.add(levels.get(i));
                errors.add(data.get(i) - forecasts.get(i));
                levels.add(levels.get(i) + a*errors.get(i));

            }

            for (int i=1; i<(m+1); i++) {

                forecasts.add(levels.get(levels.size() - 1));
            }

            return forecasts;

    }

}
```

Εικόνα 5: Κώδικας Java για απλή εκθετική εξομάλυνση

Ο παραπάνω κώδικας δέχεται τις εξής παραμέτρους:

- data: το σύνολο των δεδομένων
- a: ο συντελεστής εξομάλυνσης
- initialLevel: το αρχικό επίπεδο
- m: ο αριθμός των περιόδων μπροστά για τις οποίες ζητείται πρόβλεψη

Ο κώδικας δίνει σαν έξοδο τις προβλέψεις που ζητήθηκαν.

2.4.4.2 ΜΟΝΤΕΛΟ ΓΡΑΜΜΙΚΗΣ ΤΑΣΗΣ (HOLT EXPONENTIAL SMOOTHING)

Το μοντέλο εξομάλυνσης γραμμικής τάσης αποτελεί επέκταση του μοντέλου απλής εκθετικής εξομάλυνσης, η οποία μπορεί επιπρόσθετα να διαχειριστεί την συνιστώσα της τάσης που συχνά συναντάμε στις πραγματικές επιχειρησιακές χρονοσειρές. Παρουσιάζει ομοιότητες με το μοντέλο παλινδρόμησης, όμως σταδιακά αποδίδεται μεγαλύτερη βαρύτητα στα πιο πρόσφατα δεδομένα και το αρχικό σημείο και η κλίση επαναϋπολογίζονται σε κάθε χρονική περίοδο. Πρακτικές μελέτες έχουν δείξει ότι οι εξομαλυμένες τιμές του αρχικού σημείου και της κλίσης είναι πολύ πιο ακριβείς από τις αντίστοιχες τιμές που υπολογίζονται αν στα δεδομένα εφαρμοστεί απλή γραμμική παλινδρόμηση. Το μοντέλο της εξομάλυνσης γραμμικής τάσης (Holt Exponential Smoothing, λόγω της εισαγωγής του στην επιστήμη των προβλέψεων το 1957 από τον Holt [2]) μαθηματικά περιγράφεται από τις παρακάτω εξισώσεις:

$$e_t = Y_t - \mathrm{F_t}$$
$$S_t = S_{t-1} + T_{t-1} + \mathrm{a} \cdot e_t$$
$$T_t = T_{t-1} + \mathrm{b} \cdot e_t$$
$$\mathrm{F_{t+m}} = S_t + m \cdot T_t$$

Όπου e_t είναι το σφάλμα της πρόβλεψης το οποίο προκύπτει από τη διαφορά της πραγματικής τιμής της χρονοσειράς και της πρόβλεψης για την ίδια χρονική περίοδο t. Το S_t, είναι το επίπεδο της χρονοσειράς στο τέλος της χρονικής περιόδου t και είναι ίσο με το άθροισμα το επιπέδου της χρονικής περιόδου t-1, της τάσης την χρονική περίοδο t-1 και ενός ποσοστού το σφάλματος πρόβλεψης. Το ποσοστό αυτό καθορίζεται από τον συντελεστή a ο οποίος ορίζεται ως ο συντελεστής εξομάλυνσης του επιπέδου και το πεδίο τιμών του είναι από το 0 έως το 1. Η τάση T_t αντιπροσωπεύει την τάση που υπάρχει στην χρονοσειρά για την περίοδο t και είναι ίση με το άθροισμα της τάσης της χρονικής περιόδου t-1 και ενός ποσοστού του σφάλματος της πρόβλεψης. Το ποσοστό αυτό συμβολίζεται με τον συντελεστή b ο οποίος καλείται συντελεστής εξομάλυνσης της τάσης και το πεδίο τιμών του είναι επίσης από το 0 έως το 1. Η ποσότητα F_{t+m} που υπάρχει στην τελευταία από τις

σχέσεις που περιγράφουν το μοντέλο είναι η πρόβλεψη που πραγματοποιείται στο τέλος της περιόδου t και αναφέρεται σε m περιόδους μπροστά. Η πρόβλεψη είναι ίση με το άθροισμα του επιπέδου S_t και της τάσης T_t πολλαπλασιασμένη με τον αριθμό m περιόδων του ορίζοντα πρόβλεψης.

Από την ανάλυση των σχέσεων που προηγήθηκε, γίνονται εμφανείς και οι διαφορές μεταξύ του μοντέλου γραμμικής τάσης και του μοντέλου γραμμικής παλινδρόμησης. Το επίπεδο στο μοντέλο γραμμικής τάσης είναι το αρχικό σημείο μίας γραμμής τάσης η οποία αντιστοιχεί στη συγκεκριμένη μόνο χρονική περίοδο και μεταβάλλεται ανάλογα με τα δεδομένα κάθε χρονικής περιόδου.

Αρχικό Επίπεδο και Αρχική Τάση

Η αρχικοποίηση τόσο του επιπέδου όσο και της τάσης είναι εξαιρετικά σημαντική και στο μοντέλο γραμμικής εξομάλυνσης. Το αρχικό επίπεδο υπολογίζεται όπως και στην απλή εκθετική εξομάλυνση. Ως αρχική τάση συνήθως χρησιμοποιείται:

- η διαφορά της δεύτερης και πρώτης παρατήρησης $(Y_2 - Y_1)$
- η διαφορά την ν- οστής παρατήρησης και πρώτης διαιρεμένη με ν-1
- η σταθερά κλίσης της κλίσης από το μοντέλο της απλής γραμμικής παλινδρόμησης

Η τελική επιλογή γίνεται πάντα σύμφωνα με τα χαρακτηριστικά και το είδος της χρονοσειράς την οποία θέλουμε να μελετήσουμε.

Συντελεστές Εξομάλυνσης α και b

Σχετικά με τις τιμές των συντελεστών εξομάλυνσης, όπως και στο μοντέλο εκθετικής εξομάλυνσης σταθερού επιπέδου, ποικίλουν οι τιμές τους στο εύρος από το 0 έως το 1, ανάλογα με το κριτήριο επιλογής που θα χρησιμοποιηθεί, όπως η ελαχιστοποίηση του μέσου τετραγωνικού σφάλματος που συνηθίζεται, ή η ελαχιστοποίηση του μέσου απόλυτου σφάλματος. Από την ελαχιστοποίηση του μέσου τετραγωνικού σφάλματος προκύπτουν τιμές και για τις δύο παραμέτρους της μεθόδου. Ουσιαστικά προκύπτει ο καλύτερος συνδυασμός των παραμέτρων σχετικά με την ακρίβεια. Είναι προφανές πως λόγω ύπαρξης δύο παραμέτρων εξομάλυνσης, η εύρεση της καλύτερης τιμής τους γίνεται πιο πολύπλοκη διαδικασία που απαιτεί και περισσότερο χρόνο. Συνεπώς η χρήση αποτελεσματικού αλγορίθμου προς εύρεση αυτών των παραμέτρων κρίνεται πολύ σημαντική.

Κώδικας Java

```
package metaptyxiakh;

import java.util.ArrayList;
```

```
public class HoltExponentialSmoothing {

    public ArrayList <Double> holt (ArrayList <Double> data, Double a,
Double b, Double initialLevel, Double initialTrend, Integer m) {

        ArrayList <Double> levels = new ArrayList();
        ArrayList <Double> trends = new ArrayList();
        ArrayList <Double> forecasts = new ArrayList();
        ArrayList <Double> errors = new ArrayList();

        levels.add(initialLevel);
        trends.add(initialTrend);

        for (int i=0; i<data.size(); i++) {

        forecasts.add(levels.get(i) + trends.get(i));
        errors.add(data.get(i) - forecasts.get(i));
        levels.add(levels.get(i) + trends.get(i) + a*errors.get(i));
        trends.add(trends.get(i) + a*b*errors.get(i));
                                        }

        for (int i=1; i<(m+1); i++) {

            forecasts.add(levels.get(levels.size() - 1) +
i*trends.get(trends.size() - 1));
        }

        return forecasts;

    }
}
```

Εικόνα 6: Κώδικας Java για το μοντέλο γραμμικής τάσης

Ο παραπάνω κώδικας δέχεται τις εξής παραμέτρους:

- data: το σύνολο των δεδομένων
- a: ο πρώτος συντελεστής εξομάλυνσης
- b: ο δεύτερος συντελεστής εξομάλυνσης
- initialLevel: το αρχικό επίπεδο
- initialTrend: η αρχική τάση
- m: ο αριθμός των περιόδων μπροστά για τις οποίες ζητείται πρόβλεψη

Ο κώδικας δίνει σαν έξοδο τις προβλέψεις που ζητήθηκαν.

2.4.4.3 ΜΟΝΤΕΛΟ ΜΗ ΓΡΑΜΜΙΚΗΣ ΤΑΣΗΣ (DAMPED EXPONENTIAL SMOOTHING)

Το μοντέλο γραμμικής τάσης που περιγράψαμε παραπάνω, μπορεί να μεταβληθεί κατάλληλα ώστε να προσαρμόζεται και σε μη γραμμικές τάσεις. Αυτό επιτυγχάνεται με την χρήση μιας παραμέτρου επιπλέον που ελέγχει τον ρυθμό αύξησης των τιμών

των προβλέψεων. Αυτή ονομάζεται παράμετρος διόρθωσης της τάσης, συμβολίζεται με φ και περιγράφηκε το 1985 από τους Gardner και McKenzie [3]. Το μοντέλο μη γραμμικής τάσης περιγράφεται μαθηματικά ως ακολούθως:

$$e_t = Y_t - \mathrm{F_t}$$
$$S_t = S_{t-1} + \varphi \cdot T_{t-1} + \mathrm{a} \cdot e_t$$
$$T_t = \varphi \cdot T_{t-1} + \mathrm{b} \cdot e_t$$
$$\mathrm{F_{t+m}} = S_t + \sum_{\mathrm{i=1}}^{\mathrm{m}} \varphi^i \cdot T_t$$

Αρχικά, όπως και στις προηγούμενες μεθόδους, υπολογίζεται το σφάλμα της πρόβλεψης e_t το οποίο προκύπτει από τη διαφορά της πραγματικής τιμής της χρονοσειράς και της πρόβλεψης για την ίδια χρονική περίοδο t. Το S_t,είναι το επίπεδο της χρονοσειράς στο τέλος της χρονικής περιόδου t και είναι ίσο με το άθροισμα του επιπέδου της χρονικής περιόδου t-1, της τάσης κατά την χρονική περίοδο t-1 και ενός ποσοστού το σφάλματος πρόβλεψης. Το ποσοστό αυτό καθορίζεται από τον συντελεστή a ο οποίος ορίζεται ως ο συντελεστής εξομάλυνσης του επιπέδου και το πεδίο τιμών του είναι από το 0 έως το 1. Η τάση T_t αντιπροσωπεύει την τάση που υπάρχει στην χρονοσειρά για την περίοδο t και είναι ίση με το άθροισμα της τάσης της χρονικής περιόδου t-1 και ενός ποσοστού του σφάλματος της πρόβλεψης. Το ποσοστό αυτό συμβολίζεται με τον συντελεστή b ο οποίος καλείται συντελεστής εξομάλυνσης της τάσης και το πεδίο τιμών του είναι επίσης από το 0 έως το 1. Όπως εύκολα γίνεται αντιληπτό, οι εξισώσεις αυτές είναι όμοιες με αυτές του γραμμικού μοντέλου. Η διαφορά έγκειται στην τελευταία εξίσωση στην οποία αντί να υπολογίζεται μια γραμμική αύξηση της τάσης μέσω του ορίζοντα πρόβλεψης m, πραγματοποιείται ένας μη γραμμικός υπολογισμός αυτής μέσω της παραμέτρου εξομάλυνσης φ.

Ιδιαίτερο ενδιαφέρον παρουσιάζει η μελέτη της τιμής της νέας παραμέτρου φ που χρησιμοποιείται, η οποία δεν έχει κάποιο πάνω ή κάτω όριο και δύναται να πάρει οποιαδήποτε τιμή. Αν η παράμετρος είναι μεγαλύτερη της μονάδας, τότε προκύπτει εκθετική τάση και το μέγεθος κατά το οποίο αυξάνει η τιμή των προβλέψεων μεγαλώνει κάθε φορά. Αν η τιμή του συντελεστή φ όμως είναι μικρότερη από την μονάδα τότε προκύπτει φθίνουσα τάση και το μέγεθος κατά το οποίο αυξάνει η τιμή των προβλέψεων μικραίνει κάθε χρονική περίοδο.

Αρχικό Επίπεδο και Αρχική Τάση

Όπως περιγράψαμε και στα δυο προηγούμενα μοντέλα, η αρχικοποίηση το μοντέλου γίνεται εν γένει με εφαρμογή της γραμμικής παλινδρόμησης στα ιστορικά δεδομένα όπου οι αρχικές τιμές του επιπέδου και της τάσης λαμβάνουν τις τιμές του αρχικού σημείου και της κλίσης της ευθείας της γραμμικής παλινδρόμησης αντίστοιχα. Επίσης

είναι δυνατή και η χρήση και των άλλων τρόπων που έχουν ήδη περιγραφεί στη γραμμική εκθετική εξομάλυνση.

Συντελεστές Εξομάλυνσης α,b,φ

Σχετικά με τις τιμές των συντελεστών εξομάλυνσης και της παραμέτρου διόρθωσης της τάσης, ελέγχονται κάποιες δοκιμαστικές τιμές. Το κριτήριο επιλογής είναι στις περισσότερες περιπτώσεις η ελαχιστοποίηση το μέσου τετραγωνικού σφάλματος (MSE) αν και θα μπορούσαν να χρησιμοποιηθούν και άλλα είδη σφαλμάτων όπως είναι το ποσοστιαίο σφάλμα.

Βέβαια, όσο αυξάνεται το πλήθος των παραμέτρων, τόσο αυξάνεται και η υπολογιστική πολυπλοκότητα του προβλήματος άρα και του χρόνου που απαιτείται για την εύρεση των παραμέτρων. Οπότε η υλοποίηση ενός αποδοτικού αλγορίθμου που χρησιμοποιεί το κριτήριο της ελαχιστοποίησης του μέσου τετραγωνικού σφάλματος ώστε να εξάγει το βέλτιστο δυνατό συνδυασμό πέρα από την γραμμική αναζήτηση η οποία είναι χρονοβόρα, κρίνεται επιτακτική.

Είναι σημαντικό να αναλυθεί περαιτέρω η φυσική σημασία της τιμής της παραμέτρου διόρθωσης και οι διαφορές που προκαλούν οι αλλαγές της τιμής της στο εν λόγο μοντέλο. Αρκετές φορές λοιπόν, οι προβλέψεις που προκύπτουν από το μοντέλο μη γραμμικής τάσης είναι ίδιες με αυτές του μοντέλου απλής εκθετικής εξομάλυνσης ή του μοντέλου γραμμικής τάσης. Για παράδειγμα αν στα δεδομένα μας δεν υπάρχει τάση και εφαρμοστεί το μοντέλο μη γραμμικής τάσης και το μοντέλο σταθερού επιπέδου, οι προβλέψεις που θα παραχθούν θα είναι κατά προσέγγιση ίσες. Αυτό γιατί η τιμή της παραμέτρου διόρθωσης της τάσης φ που θα προκύψει με την προαναφερόμενη διαδικασία εύρεσης της, θα είναι πολύ κοντά στο 0. Και πράγματι, αν στις μαθηματικές εξισώσεις περιγραφής του μοντέλου μη γραμμικής τάσης εξαλείψουμε την παράμετρο φ (θεωρώντας την αμελητέα και άρα ίση με το 0) , προκύπτει το μοντέλο σταθερού επιπέδου συνεπώς οι προβλέψεις είναι ακριβώς οι ίδιες.

Από την παραπάνω διαπίστωση γίνεται φανερό ότι μπορεί να χρησιμοποιηθεί το μοντέλο μη γραμμικής τάσης σαν ένα αυτόματο σύστημα πρόβλεψης για κάθε τύπο μη εποχιακής χρονοσειράς. Για κάθε τιμή της παραμέτρου διόρθωσης της τάσης φ έχουμε αντίστοιχη σε ένα από τα παρακάτω μοντέλα εξομάλυνσης:

- $\phi = 0$, απλή εκθετική εξομάλυνση σταθερού επιπέδου
- $\phi < 1$, μοντέλο φθίνουσας τάσης
- $\phi = 1$, μοντέλο γραμμικής τάσης
- $\phi > 1$, μοντέλο εκθετικής τάσης

Η ακρίβεια των προβλέψεων του μοντέλου μη γραμμικής τάσης είναι σημαντικά μεγαλύτερη από τις αντίστοιχες του μοντέλου γραμμικής τάσης. Γενικά το μοντέλο μη γραμμικής τάσης δίνει ικανοποιητικά αποτελέσματα σε περιπτώσεις όπου είναι αδύνατη η εύρεση κάποιου συγκεκριμένου μοντέλου για την παραγωγή προβλέψεων κάποιας χρονοσειράς.

Άλλο ένα πλεονέκτημα του μοντέλου μη γραμμικής τάσης είναι η καταλληλότητα του για παραγωγή προβλέψεων μεγάλου χρονικού ορίζοντα. Πραγματικά οι πρακτικές έρευνες έχουν δείξει ότι όσο πιο μακρινός είναι ο ορίζοντας πρόβλεψης, τόσο πιο πολύ πλεονεκτεί το μοντέλο μη γραμμικής τάσης σε ακρίβεια έναντι των άλλων μοντέλων.

Κώδικας Java

```
package metaptyxiakh;

import static java.lang.Math.pow;
import java.util.ArrayList;

public class DampedExponentialSmoothing {

    public ArrayList <Double> damped (ArrayList <Double> data, Double
a, Double b, Double initialLevel, Double initialTrend, Double f,
Integer m) {

        ArrayList <Double> levels = new ArrayList();
        ArrayList <Double> trends = new ArrayList();
        ArrayList <Double> forecasts = new ArrayList();
        ArrayList <Double> errors = new ArrayList();

        levels.add(initialLevel);
        trends.add(initialTrend);

        for (int i=0; i<data.size(); i++) {

        forecasts.add(levels.get(i) + f*trends.get(i));
        errors.add(data.get(i) - forecasts.get(i));
        levels.add(levels.get(i) + f*trends.get(i) + a*errors.get(i));
        trends.add(f*trends.get(i) + a*b*errors.get(i));
                                        }

        Double fcoefficient = 0.0;

        for (int i=1; i<(m+1); i++) {

        fcoefficient = fcoefficient + pow(f,i);
        forecasts.add(levels.get(levels.size() - 1) +
fcoefficient*trends.get(trends.size() - 1));
                                    }

        return forecasts;

    }
}
```

Εικόνα 7: Κώδικας Java για το μοντέλο μη γραμμικής τάσης

Ο παραπάνω κώδικας δέχεται τις εξής παραμέτρους:

- data: το σύνολο των δεδομένων
- a: ο πρώτος συντελεστής εξομάλυνσης

- b: ο δεύτερος συντελεστής εξομάλυνσης
- φ: ο τρίτος συντελεστής εξομάλυνσης
- initialLevel: το αρχικό επίπεδο
- initialTrend: η αρχική τάση
- m: ο αριθμός των περιόδων μπροστά για τις οποίες ζητείται πρόβλεψη

Ο κώδικας δίνει σαν έξοδο τις προβλέψεις που ζητήθηκαν.

2.4.5. ΜΟΝΤΕΛΟ THETA

Τέλος, μια μέθοδος που ενσωματώθηκε στο σύστημα και αναπτύχθηκε από δύο μέλη της Μονάδας Προβλέψεων και Στρατηγικής το 2000 είναι η μέθοδος Theta [4]. Η μέθοδος Theta (Assimakopoulos και Nikolopoulos, 2000; Νικολόπουλος, 2002) είναι μία μονοδιάστατη μέθοδος πρόβλεψης, η οποία βασίζεται στην μεταβολή των τοπικών καμπυλοτήτων μιας χρονοσειράς μέσα από την παράμετρο θ που εφαρμόζεται πολλαπλασιαστικά στις διαφορές δεύτερης τάξης των δεδομένων. Η καινούργια χρονοσειρά που δημιουργείται διατηρεί την μέση τιμή και κλίση της αρχικής χρονοσειράς αλλά όχι και τις τοπικές καμπυλότητες και τη διακύμανση. Οι χρονοσειρές που παράγονται με αυτή τη διαδικασία ονομάζονται γραμμές Theta. Βασικό ποιοτικό χαρακτηριστικό αυτών των γραμμών είναι η καλύτερη προσέγγιση της μακροπρόθεσμης συμπεριφοράς των δεδομένων ή ανάδειξη και τονισμός των βραχυπρόθεσμων χαρακτηριστικών, ανάλογα με την τιμή της παραμέτρου θ (μικρότερη ή μεγαλύτερη της μονάδας αντίστοιχα).

Η προτεινόμενη μέθοδος αποσυνθέτει (διαχωρίζει) την αρχική χρονοσειρά σε δύο ή περισσότερες γραμμές Theta. Η κάθε γραμμή Theta προεκτείνεται στο μέλλον ξεχωριστά, με την ίδια ή και με διαφορετικές μεθόδους πρόβλεψης και οι παραγόμενες προβλέψεις συνδυάζονται για να προκύψει η τελική πρόβλεψη. Ο απλός συνδυασμός δύο γραμμών Theta, για θ=0 (ευθεία γραμμή) και για θ=2 (διπλασιασμός των τοπικών καμπυλοτήτων) χρησιμοποιήθηκε για την παραγωγή προβλέψεων για τις 3003 χρονοσειρές του διεθνούς διαγωνισμού προβλέψεων M3 και παρήγαγε πολύ καλά αποτελέσματα, με μικρά σφάλματα προβλεπτικής ακρίβειας.

Τα βήματα που ουσιαστικά περιγράφουν τη μεθοδολογία της κλασικής μεθόδου theta είναι τα παρακάτω:

Βήμα 0. Έλεγχος εποχιακότητας. Ελέγχεται η κάθε χρονοσειρά για στατιστικά σημαντική εποχιακή συμπεριφορά.

Βήμα 1. Αποεποχικοποίηση. Μέσω της κλασικής μεθόδου πολλαπλασιαστικής αποσύνθεσης, εφόσον αποδειχθεί ότι η χρονοσειρά έχει σημαντική εποχιακότητα.

Βήμα 2. Αποσύνθεση. Η κάθε χρονοσειρά αποσυντίθεται σε γραμμές Theta, την ευθεία γραμμικής παλινδρόμησης (θ = 0) και τη γραμμή Theta με παράμετρο θ = 2.

Βήμα 3. Πρόβλεψη. Η γραμμή Theta με παράμετρο θ = 0, που αναπαριστά την ευθεία γραμμικής παλινδρόμησης, προεκτείνεται με τον συνηθισμένο τρόπο, ενώ η δεύτερη γραμμή προεκτείνεται μέσω της απλής εκθετικής εξομάλυνσης.

Βήμα 4. Συνδυασμός. Οι παραγόμενες προβλέψεις των δύο γραμμών Theta συνδυάζονται με ίσα βάρη.

Βήμα 5. Εποχικοποίηση. Οι τελικές προβλέψεις εποχικοποιούνται, χρησιμοποιώντας τους δείκτες εποχιακότητας που υπολογίστηκαν στο βήμα 1.

Ακολούθως περιγράφεται ο υπολογισμός των δύο γραμμών Theta που προτείνονται στην αναφερόμενη μεθοδολογία. Δεδομένου πως η Theta Line(0) ισοδυναμεί με την ευθεία ελαχίστων τετραγώνων (LRL) που περιγράφηκε στην προηγούμενη παράγραφο, απομένει ο υπολογισμός της Theta Line(2). Έτσι προκύπτουν οι εξής σχέσεις:

$$Y_t = \frac{1}{2}\left(Y_t^{\theta=1-\alpha} + Y_t^{\theta=1-\alpha}\right) \overset{\alpha=1}{\Longleftrightarrow}$$

$$Y_t = \frac{1}{2}\left(Y_t^{\theta=0} + Y_t^{\theta=2}\right) \overset{Y_t^{\theta=0}=LRL_t}{\Longleftrightarrow}$$

$$Y_t^{\theta=2} = 2 \cdot Y_t - LRL_t$$

Η τελευταία εξίσωση οδηγεί σε έναν εναλλακτικό τρόπο παραγωγής της Theta Line(2) αφού η LRL μπορεί εύκολα να παραχθεί σύμφωνα με τη θεωρία της παλινδρόμησης. Ένας άλλος τρόπος υπολογισμού οποιασδήποτε γραμμής Theta είναι σύμφωνα με τον Νικολόπουλο και του συνεργάτες το 2008 [5]:

$$Theta\ Line(\theta)_t = Y_t^{\theta} = LRL_t + \theta \cdot e_t$$

όπου:

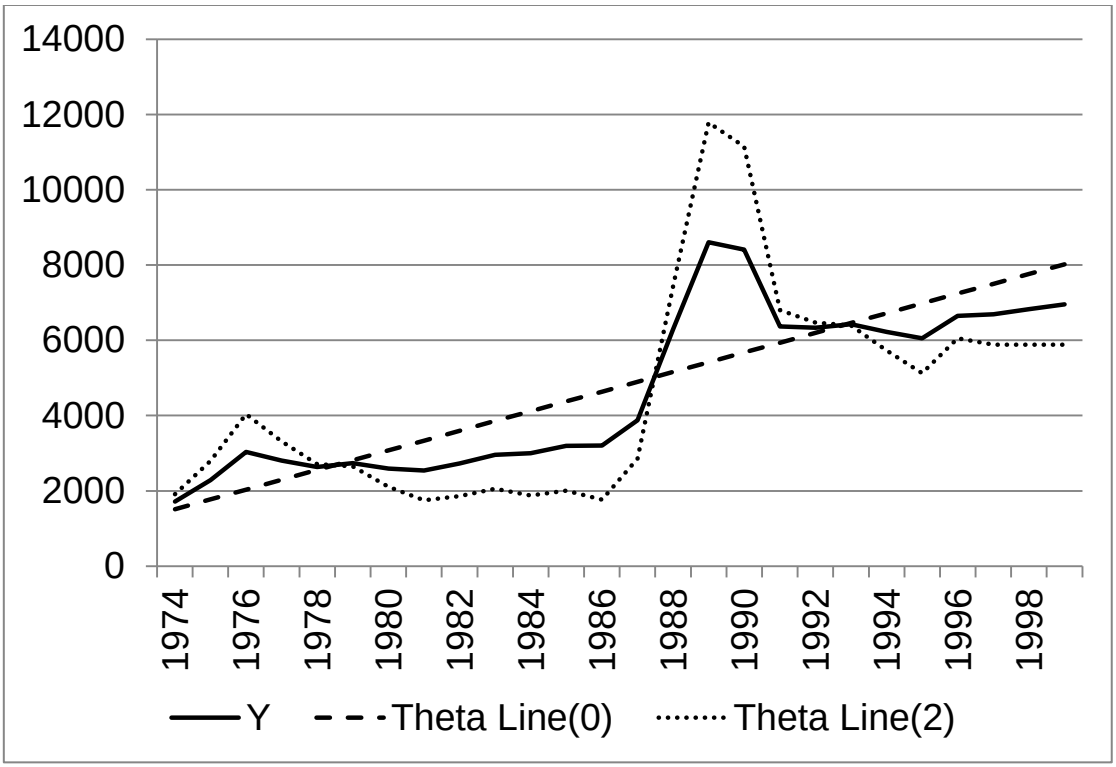
$$e_t = Y_t - LRL_t$$

Ενώ οι Hyndman και Billah (2003) [6] πρότειναν τον εξής τρόπο:

$$Y_t^{\theta} = \theta \cdot Y_t + \alpha_{\theta} + b_{\theta} \cdot (t-1)$$

Στην μέθοδο Theta, η μακροπρόθεσμη τάση εξασφαλίζεται από της προέκταση της γραμμής θ = 0. Ταυτόχρονα η ύπαρξη και της γραμμής θ=2 λειτουργεί σαν αντίβαρο στην χρησιμοποίηση μόνο της απλής γραμμικής παλινδρόμησης και εξασφαλίζει την αξιοποίηση και της βραχυπρόθεσμης πληροφορίας. Ως αποτέλεσμα, το σημείο εκκίνησης των προβλέψεων πετυχαίνει καλύτερη προσέγγιση του σωστού επιπέδου

και εξασφαλίζει μία συντηρητική μεν αλλά σταθερή δε συνέχιση της μακροπρόθεσμης τάσης.



Εικόνα 8: Η μέθοδος Theta

2.4.5.1 ΚΩΔΙΚΑΣ JAVA

```
package metaptyxiakh;

import java.util.ArrayList;

public class Theta {

    public ArrayList <Double> thetaf (ArrayList <Double> data, Double
a, Integer p, Integer m) {

        ArrayList <Double> normalisedSeasonalityIndexes = new
ArrayList();
        ArrayList <Double> deSeasonalisedData = new ArrayList();
        Double[] coefficientsLRL = new Double[2];
        ArrayList <Double> thetaZero = new ArrayList();
        ArrayList <Double> thetaTwo = new ArrayList();
        Double initialLevel = 0.0;
        ArrayList <Double> thetaTwoSmoothed = new ArrayList();
        ArrayList <Double> forecasts = new ArrayList();

        SeasonalDecomposition seasonalDecomposition = new
SeasonalDecomposition();
```

```
        normalisedSeasonalityIndexes = seasonalDecomposition.sd(data,
p);

        for (int i=normalisedSeasonalityIndexes.size(); i<data.size()
+ m; i++) {


normalisedSeasonalityIndexes.add(normalisedSeasonalityIndexes.get(i-
p));
        }

        for (int i=0; i<data.size(); i++) {

            deSeasonalisedData.add(100 * data.get(i) /
normalisedSeasonalityIndexes.get(i));
        }

        LinearRegression linearRegression = new LinearRegression();
        coefficientsLRL = linearRegression.lrl(deSeasonalisedData);

        for (int i=1; i<deSeasonalisedData.size() + 1 + m; i++) {

            thetaZero.add(coefficientsLRL[0] + coefficientsLRL[1] *
i);
        }

        for (int i=0; i<deSeasonalisedData.size(); i++) {

            thetaTwo.add(thetaZero.get(i) + 2 *
(deSeasonalisedData.get(i) - thetaZero.get(i)));
        }

        for (int i=0; i<p; i++) {

            initialLevel = initialLevel + thetaTwo.get(i);
        }

        initialLevel = initialLevel/p;


        SimpleExponentialSmoothing simpleExponentialSmoothing = new
SimpleExponentialSmoothing();
        thetaTwoSmoothed = simpleExponentialSmoothing.ses(thetaTwo, a,
initialLevel, m);


        for (int i=0; i<deSeasonalisedData.size() + m; i++) {

            forecasts.add((0.5 * thetaZero.get(i) + 0.5 *
thetaTwoSmoothed.get(i)) * normalisedSeasonalityIndexes.get(i) / 100);
        }

        return forecasts;
    }

}
```

Εικόνα 9: Κώδικας Java για τη μέθοδο Theta

Ο παραπάνω κώδικας δέχεται τις εξής παραμέτρους:

- data: το σύνολο των δεδομένων
- a: ο συντελεστής εξομάλυνσης της SES
- p: το μήκος της εποχιακότητας των δεδομένων
- m: ο αριθμός των περιόδων μπροστά για τις οποίες ζητείται πρόβλεψη

Ο κώδικας δίνει σαν έξοδο τις προβλέψεις που ζητήθηκαν

2.4.6. ΕΠΙΛΟΓΗ ΤΗΣ ΚΑΤΑΛΛΗΛΗΣ ΜΕΘΟΔΟΥ ΠΡΟΒΛΕΨΗΣ

Η εξαγωγή των προβλέψεων λόγω της μεγάλης τεχνολογικής εξέλιξης που υπάρχει, δεν μπορεί να χαρακτηριστεί ως μία δύσκολη διαδικασία, οι απαιτούμενοι υπολογισμοί μπορούν να υλοποιηθούν σε οποιαδήποτε γλώσσα προγραμματισμού και συνήθως σε λίγα δευτερόλεπτα να μας δώσουν προβλέψεις. Όμως η επιλογή της κατάλληλης μεθόδου η οποία θα χρησιμοποιηθεί για τους υπολογισμούς αυτούς δεν είναι μια εύκολη διαδικασία που μπορεί να γίνει από απλούς χρήστες ενός προγράμματος. Όπως έχει αναφερθεί προηγουμένως, οι μέθοδοι πρόβλεψης κατατάσσονται σε διάφορες κατηγορίες ανάλογα με τις εφαρμογές τους αλλά και τα κύρια χαρακτηριστικά τους, έτσι ώστε να γίνει η διαδικασία της επιλογής τους ανά περίπτωση πιο εύκολη διαδικασία. Κινούμενοι προς αυτήν την κατεύθυνση θα αναφερθούν κάποιοι βασικοί παράγοντες που αντικατοπτρίζουν τις δυνατότητες εφαρμογής των διαθέσιμων μεθόδων. Οι κυριότεροι λοιπόν παράγοντες είναι:

- *Χρονικός ορίζοντας*. Ανάλογα το χρονικό διάστημα στο μέλλον στο οποίο θα αναφέρεται η πρόβλεψη συχνά επιλέγεται και η αντίστοιχη μέθοδος που θα χρησιμοποιηθεί. Επίσης σημαντικό στοιχείο είναι και το πλήθος των περιόδων για το οποίο απαιτείται πρόβλεψη.
- *Πρότυπο συμπεριφοράς των δεδομένων.* Δεν είναι δυνατή η εφαρμογή κανενός μοντέλου πρόβλεψης αν πρώτα δεν αναγνωριστεί ένα βασικό πρότυπο συμπεριφοράς των δεδομένων, το οποίο θα αποτελέσει βάση της τεχνικής πρόβλεψης που θα εφαρμοστεί. Τα τέσσερα βασικά πρότυπα που συμπεριφοράς που συχνά εμφανίζονται στις χρονοσειρές και τις περισσότερες φορές συνυπάρχουν είναι το σταθερό πρότυπο, το πρότυπο της τάσης, το εποχιακό και το κυκλικό πρότυπο.
- *Κόστος.* Αναφερόμενοι σε μία μέθοδο πρόβλεψης, το κόστος της σχετίζεται άμεσα με τον όγκο των δεδομένων που αποτελούν τα ιστορικά στοιχεία και από την πολυπλοκότητα κατά την εφαρμογή της. Το κόστος μπορεί να αναφέρεται τόσο σε υπολογιστικό όσο και σε χρηματικό κόστος για την υλοποίηση των μεθόδων.
- *Αξιοπιστία.* Η αξιοπιστία σχετικά με τις προβλέψεις, συνδέεται με το επίπεδο λεπτομέρειας που απαιτείται στην αντίστοιχη περίπτωση. Υπάρχουν

περιπτώσεις όπου ένα ποσοστό ακρίβεια της πρόβλεψης 10% είναι ικανοποιητικό ενώ άλλες που ακόμα και το μισό ποσοστό από το προαναφερόμενο μπορεί να αποδειχθεί καταστροφικό.

- *Απλότητα και ευκολία στην εφαρμογή της.* Απλές και εύληπτες μέθοδοι εν γένει προτιμούνται καθώς είναι και πιο εύκολες στην εφαρμογή τους. Από την μία, μπορεί ο συνδυασμός μεθόδων να δώσει καλύτερα αποτελέσματα, αλλά από την άλλη έχει αποδειχθεί πειραματικά ότι απλές μέθοδοι πηγαίνουν καλύτερα σε σύγκριση με πολύπλοκες μεθόδους.

Η μελέτη για την επιλογή της κατάλληλης μεθόδου πρόβλεψης είναι ένα θέμα το οποίο έχει απασχολήσει ιδιαίτερα την επιστημονική κοινότητά, και έχει μελετηθεί από πάρα πολλούς ερευνητές με κυριότερα αποτελέσματα να προέρχονται από τον Davis Wright και τους συνεργάτες του (1986) [7], τον Yokum, J. T. και τον Armstrong, J. S. (1995) [8] καθώς και τον Tashman, L.J. (1991 [9], 2000 [10]). Ο αλγόριθμος που χρησιμοποιείται για την επιλογή της κατάλληλης μεθόδου βασίζεται στην ελαχιστοποίηση του In Sample Mean Squared Error, Chatfield (1988) [11].

2.4.7. ΣΥΝΔΥΑΣΜΟΙ ΜΕΘΟΔΩΝ ΠΡΟΒΛΕΨΗΣ

Ανάλογα με τις συνθήκες και τα κύρια χαρακτηριστικά που εμφανίζονται γίνεται και η επιλογή της κατάλληλης μεθόδου. Εντούτοις, η αναγνώριση των χαρακτηριστικών και των συνθηκών που επικρατούν δεν είναι εύκολη και γρήγορη διαδικασία. Σε άλλες πάλι περιπτώσεις ακόμα και να αναγνωριστούν σωστά οι διάφοροι παράγοντες, είναι δύσκολη η εύρεση μίας μεθόδου αποκλειστικά που να ικανοποιεί πλήρως όλες τις απαιτήσεις του προβλήματος για την πρόβλεψη. Παρατηρήθηκε ότι ένας τρόπος αύξησης της ακρίβειας των προβλέψεων, είναι ο συνδυασμός διαφορετικών μεθόδων πρόβλεψης. Διαφορετικές μέθοδοι πρόβλεψης, εφαρμοζόμενες στις ίδιες χρονοσειρές, παράγουν διαφορετικά αποτελέσματα καθώς η κάθε πρόβλεψη παρέχει και διαφορετική πληροφορία. Ο συνδυασμός τους παράγει προβλέψεις, αξιοποιώντας περισσότερη πληροφορία στην ίδια τιμή, γεγονός που ενισχύει την ακρίβεια της πρόβλεψης καθώς προσεγγίζεται καλύτερα η πραγματικότητα. Αξιοπρόσεχτο όμως σε αυτό το σημείο είναι να αναφερθεί ο τρόπος με τον οποίο γίνεται να πραγματοποιηθεί ο συνδυασμός των μεθόδων πρόβλεψης. Δύο βασικές τεχνικές επικρατούν για την υλοποίηση του εν λόγω συνδυασμού. Ο απλός μέσος όρων όλων των προβλέψεων των μεθόδων που θα επιλεγούν είναι η πρώτη τεχνική, ενώ ο υπολογισμός του μέσου όρου αλλά με χρήση συντελεστών βαρύτητας είναι η δεύτερη. Σχετικά με τη δεύτερη τεχνική, οι συντελεστές βαρύτητας που γενικά χρησιμοποιούνται εξαρτώνται από την σχετική ακρίβεια της μεθόδου και από την συνδιακύμανση των σφαλμάτων πρόβλεψης. Το μέγεθος της ανομοιότητας των μεθόδων όπως επίσης και το μέτρο της σχετικής ακρίβειας κάθε μεθόδου σε μια συγκεκριμένη περίπτωση είναι δείκτες για την τιμή των συντελεστών βαρύτητας. Μελέτες που έχουν πραγματοποιηθεί έχουν αποδείξει ότι ο υπολογισμός του απλού μέσου όρου προβλέψεων οδηγεί σε αποτελέσματα το ίδιο ικανοποιητικά με αυτά το πολύπλοκων τεχνικών συνδυασμού.

Πάντως γενικότερα η στρατηγική του συνδυασμού των διαφορετικών προβλέψεων, είναι εξαιρετικά εποικοδομητική καθώς από έρευνες και μελέτες όπως αυτές του Clemen (1989) [12] και του Armstrong (2001) [13] που έχουν πραγματοποιηθεί μέχρι σήμερα αποδεικνύεται μείωση του μεγέθους του σφάλματος έως και 6%.

Στα πλαίσια της συγκεκριμένης μεταπτυχιακής εργασίας έχουν χρησιμοποιηθεί μόνο απλές στατιστικές μέθοδοι και όχι συνδυασμός.

2.5. ΔΕΙΚΤΕΣ ΑΞΙΟΛΟΓΗΣΗΣ ΠΡΟΒΛΕΨΕΩΝ

Η στατιστική ανάλυση είναι ουσιαστικά η εύρεση βασικών στατιστικών δεικτών και αποτελεί και την διαδικασία ανάλυσης κάθε χρονοσειράς για την μετέπειτα ορθότερη αντιμετώπιση της. Επιτρέπει στους αναλυτές, να έχουν μια γρήγορη, δομημένη και ταυτόχρονα συνολική εικόνα για το σύνολο της χρονοσειράς και για αυτό η στατιστική ανάλυση τόσο των χρονοσειρών, όσο και των προβλέψεων και κατ' επέκταση και των μοντέλων πρόβλεψης έχει αποτελέσει αντικείμενο πολλών ερευνητών του τομέα των προβλέψεων, Makridakis (1993) [14] και Armstrong (2001) [15]. Σε συνδυασμό με την γραφική αναπαράσταση της χρονοσειράς, είναι δυνατή η επιλογή ακολούθως των κατάλληλων μεθοδολογιών και διαδικασιών πρόβλεψης. Η στατιστική ανάλυση αποτελείται από τρεις κατηγορίες που αναλύονται ακολούθως.

2.5.1. ΒΑΣΙΚΗ ΣΤΑΤΙΣΤΙΚΗ ΑΝΑΛΥΣΗ

Αποτελείται από βασικούς στατιστικούς δείκτες όπως:

- Μέση τιμή:

$$\bar{Y} = \frac{1}{n} \cdot \sum_{i=1}^{n} Y_i$$

- Μέγιστη και Ελάχιστη τιμή (Maximum and Minimum) της χρονοσειράς

- Τυπική απόκλιση (Standard Deviation):

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}{n}}$$

- Διακύμανση(Variance) : ορίζεται ως το τετράγωνο της τυπικής απόκλισης

- Συνδιακύμανση (Covariance):

$$Cov(X,Y) = \frac{1}{n} \cdot \sum_{i=1}^{n} [(X_i - \bar{X}) \cdot (Y_I - \bar{Y})]$$

$Cov(X,Y) > 0$: μεταβάλλονται ανάλογα τα δύο μεγέθη
$Cov(X,Y) < 0$:μεταβάλλονται αντιστρόφως ανάλογα τα δύο μεγέθη
$Cov(X,Y) = 0$: τα δύο μεγέθη είναι ασυσχέτιστα

- Συντελεστής γραμμικής συσχέτισης (Linear Correlation Coefficient):

$$r_{XY} = \frac{\sum_{i=1}^{n} [(X_i - \bar{X}) \cdot (Y_I - \bar{Y})]}{\sqrt{\sum_{i=1}^{n} [(X_i - \bar{X})^2]} \sqrt{\sum_{i=1}^{n} [(Y_I - \bar{Y})^2]}}$$

$r_{XY} = \pm 1$: τέλεια γραμμική συσχέτιση

$-0{,}3 < r_{XY} < 0{,}3$: δεν υπάρχει γραμμική συσχέτιση

- Συντελεστής αυτοσυσχέτισης (Autocorrelation Coefficient):

$$ACK_k = \frac{\sum_{i=1}^{n} [(X_i - \bar{X}) \cdot (Y_I - \bar{Y})]}{\sqrt{\sum_{i=1}^{n} [(Y_I - \bar{Y})^2]}}$$

ACK_k= 0: μηδενική συσχέτιση των παρατηρήσεων χρονικής υστέρησης k

$ACK_k = 1$: μεγάλη συσχέτιση των παρατηρήσεων χρονικής υστέρησης k

- Συντελεστής Μεταβλητότητας (Coefficient of Variation):

$$C_V = \frac{\sigma}{\bar{Y}} \cdot 100(\%)$$

2.5.2. ΣΤΑΤΙΣΤΙΚΗ ΑΝΑΛΥΣΗ ΑΚΡΙΒΕΙΑΣ ΠΡΟΒΛΕΨΕΩΝ

Σε αυτήν την κατηγορία πέρα από την πραγματική σειρά των παρατηρήσεων που είναι αναγκαία όπως και στην βασική στατιστική ανάλυση, απαιτείται και μία δεύτερη σειρά πρόβλεψης που προκύπτει από την εφαρμογή κάποιας κατάλληλης μεθόδου επί της πραγματικής χρονοσειράς. Στα προηγούμενα κεφάλαια έγινε ανάλυση τόσο των μεθόδων πρόβλεψης όσο και της μεθοδολογίας επιλογής της καταλληλότερης μεθόδου. Με βάση την πραγματική σειρά των δεδομένων και την επιλεγμένη μέθοδο πρόβλεψης μπορούν να υπολογιστούν οι δείκτες ακρίβειας των προβλέψεων.

Η σημασία της συγκεκριμένης κατηγορίας της στατιστικής ανάλυσης στον κλάδο των προβλέψεων είναι καίριας σημασίας καθώς αποτελεί βασικό εργαλείο για την

αξιολόγηση μεθόδων αλλά και για τον χαρακτηρισμό τους σχετικά με τον τρόπο προσέγγισης της μεθοδολογίας πρόβλεψης και της πραγματική χρονοσειράς.

Κύρια έννοια για να οριστούν οι μετέπειτα δείκτες της στατιστικής ακρίβειας προβλέψεων αποτελεί το σφάλμα, δηλαδή η διαφορά μεταξύ της πραγματικής τιμής και της πρόβλεψης για μία περίοδο, το οποίο ορίζεται ως εξής:

$$e_i = Y_i - F_i$$

Είναι προφανές πως η τιμή του σφάλματος δεν μπορεί να υπολογιστεί αν δεν υπάρχουν για την ίδια χρονική περίοδο τόσο οι πραγματικές τιμές της χρονοσειράς όσο επίσης και οι τιμές πρόβλεψης. Οπότε μπορεί να γίνει διαχωρισμός των σφαλμάτων, σε σφάλμα του μοντέλου πρόβλεψης (in – sample error) το οποίο προκύπτει από τις διαφορές των πραγματικών τιμών της χρονοσειράς που είναι ήδη διαθέσιμες και των τιμών του μοντέλου πρόβλεψης για αυτές τις χρονικές περιόδους και στο πραγματικό σφάλμα (out – of – sample error) που προκύπτει από τη διαφορά της πραγματικής μελλοντικής τιμής της χρονοσειράς που θα γίνει γνωστή μετά από το αντίστοιχο χρονικό διάστημα και τις πρόβλεψης που έχει παραχθεί από το αντίστοιχο μοντέλο για εκείνη τη χρονική περίοδο.

Οι δείκτες που ακολουθούν είναι ορισμένοι έτσι ώστε να εκφράζουν το σφάλμα της μεθόδου πρόβλεψης για n περιόδους , αλλά με κατάλληλες αλλαγές στις περιόδους που αναφέρονται μπορεί να εκφράσουν και το πραγματικό σφάλμα της πρόβλεψης. Παρακάτω λοιπόν, γίνεται παράθεση των βασικότερων δεικτών που έχουν υλοποιηθεί και υπολογίζονται από την εφαρμογή:

- Μέσο σφάλμα (Mean Error):

$$ME = \frac{1}{n} \cdot \sum_{i=1}^{n} (Y_i - F_i)$$

- Μέσο απόλυτο σφάλμα (Mean Absolute Error):

$$MAE = \frac{1}{n} \cdot \sum_{i=1}^{n} |Y_i - F_i|$$

- Μέσο τετραγωνικό σφάλμα (Mean Squared Error):

$$MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (Y_i - F_i)^2$$

- Ρίζα Μέσου τετραγωνικού σφάλματος (Root Mean Squared Error):

$$RMSE = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} (Y_i - F_i)^2}$$

- Μέσο απόλυτο ποσοστιαίο σφάλμα (Mean Absolute Percentage Error):

$$MAPE = \frac{1}{n} \cdot \sum_{i=1}^{n} \left| \frac{Y_i - F_i}{Y_i} \right| \cdot 100\ (\%)$$

- Συμμετρικό μέσο απόλυτο ποσοστιαίο σφάλμα (Symmetric Mean Absolute Percentage Error):

$$sMAPE = \frac{1}{n} \cdot \sum_{i=1}^{n} \left| \frac{Y_i - F_i}{\left(\frac{Y_i + F_i}{2}\right)} \right| \cdot 100\ (\%) = \frac{1}{n} \cdot \sum_{i=1}^{n} \left| \frac{2 \cdot (Y_i - F_i)}{Y_i + F_i} \right| \cdot 100\ (\%)$$

- Μέσο απόλυτο κανονικοποιημένο σφάλμα (Mean Absolute Scaled Error):

$$MAsE = \frac{= \frac{1}{n} \cdot \sum_{i=1}^{n} |Y_i - F_i|}{\frac{1}{n-1} \cdot \sum_{i=2}^{n} |Y_i - Y_{i-1}|}$$

Από τα προαναφερόμενα σφάλματα είναι εμφανές είναι ότι τα τέσσερα πρώτα σφάλματα, δηλαδή τα Mean Error (ME), Mean Absolute Error (MAE), Mean Squared Error (MSE) και Root Mean Squared Error (RMSE) βασίζονται κατά κύριο λόγο στις διαφορές των τιμών πρόβλεψης από τις τιμές των πραγματικών τιμών. Τα δύο επόμενα σφάλματα Mean Absolute Percentage Error (MAPE) και Symmetric Mean Absolute Percentage Error (sMAPE) εκφράζουν τα σφάλματα σε ποσοστιαία μορφή και είναι εξαιρετικά χρήσιμη μία τέτοια προσέγγιση καθώς καθιστά δυνατή τη σύγκριση σφαλμάτων από χρονοσειρές διαφορετικού επιπέδου μέσης τιμής, όπως έχουν μελετηθεί εκτενέστερα από τους Goodwin, P. και Lawton, R. (1999) [16], ενώ το τελευταίο Mean Absolute Scaled Error (MAsE) είναι ένα σχετικό μέτρο (relative measure) που χρησιμοποιεί το σφάλμα μίας μεθόδου, στην συγκεκριμένη περίπτωση τη μέθοδο naive ως benchmark – ορόσημο για να το συγκρίνει με το σφάλμα της υπό εξέτασης μεθόδου πρόβλεψης.

2.5.2.1 ΚΩΔΙΚΑΣ JAVA ΓΙΑ ΤΟΝ ΥΠΟΛΟΓΙΣΜΟ ΣΦΑΛΜΑΤΩΝ

```
package metaptyxiakh;

import static java.lang.Math.abs;
import java.util.ArrayList;
import static java.lang.Math.pow;
```

```
public class Errors {

    public Double mse (ArrayList <Double> data, ArrayList <Double>
forecasts) {

        Double MSE = 0.0;

        for (int i=0; i<data.size(); i++) {

            MSE = MSE + pow(data.get(i) - forecasts.get(i), 2);

        }

        MSE = MSE/data.size();

        return MSE;
    }

    public Double me (ArrayList <Double> data, ArrayList <Double>
forecasts) {

        Double ME = 0.0;

        for (int i=0; i<data.size(); i++) {

            ME = ME + (data.get(i) - forecasts.get(i));

        }

        ME = ME/data.size();

        return ME;
    }

    public Double mae (ArrayList <Double> data, ArrayList <Double>
forecasts) {

        Double MAE = 0.0;

        for (int i=0; i<data.size(); i++) {

            MAE = MAE + abs(data.get(i) - forecasts.get(i));

        }

        MAE = MAE/data.size();

        return MAE;
    }

    public Double mape (ArrayList <Double> data, ArrayList <Double>
forecasts) {

        Double MAPE = 0.0;

        for (int i=0; i<data.size(); i++) {

            MAPE = MAPE + abs((data.get(i) - forecasts.get(i)) /
data.get(i));

        }
```

```
        MAPE = 100*MAPE/data.size();

        return MAPE;
    }

    public Double smape (ArrayList <Double> data, ArrayList <Double>
forecasts) {

        Double SMAPE = 0.0;

        for (int i=0; i<data.size(); i++) {

            SMAPE = SMAPE + abs(2*(data.get(i) - forecasts.get(i)) /
(data.get(i) + forecasts.get(i)));

        }

        SMAPE = 100*SMAPE/data.size();

        return SMAPE;
    }
}
```

Εικόνα 10: Κώδικας Java για τον υπολογισμό των σφαλμάτων

Όλες οι μέθοδοι της Java κλάσης "Errors" δέχονται σαν είσοδο τις πραγματικές τιμές και τις αντίστοιχές τους προβλέψεις και επιστρέφουν σαν έξοδο το συγκεκριμένο σφάλμα που ζητήθηκε.

2.5.3. ΡΥΘΜΟΣ ΑΝΑΠΤΥΞΗΣ

Για λόγους πληρότητας των δεικτών στατιστικής ανάλυσης κρίνεται σκόπιμο να παρουσιαστεί και ο ρυθμός ανάπτυξης. Ο δείκτης του ρυθμού ανάπτυξης είναι ένα ποσοστό που εκφράζει το μέτρο της αυξητικής ή φθίνουσας πορείας μίας σειράς δεδομένων για ένα συγκεκριμένο χρονικό διάστημα. Ορίζεται ως εξής:

$$Growth\ Rate = \frac{\frac{1}{ppy} \cdot \sum_{i=n-ppy+1}^{n} Y_i - \frac{1}{n-ppy} \cdot \sum_{i=1}^{n-ppy} Y_i}{\frac{1}{n-ppy} \cdot \sum_{i=1}^{n-ppy} Y_i} \cdot 100(\%)$$

Όπου Y είναι το διάνυσμα των n παρατηρήσεων και ppy είναι το πλήθος των περιόδων στο μήκος ενός έτους.

3. ΠΡΟΕΤΟΙΜΑΣΙΑ ΧΡΟΝΟΣΕΙΡΑΣ

3.1. ΔΙΑΧΕΙΡΙΣΗ ΚΕΝΩΝ ΚΑΙ ΜΗΔΕΝΙΚΩΝ ΤΙΜΩΝ

Η συλλογή και διαχείριση των δεδομένων που αποτελούν τις χρονοσειρές, μία από τις βασικότερες εισόδους στις τεχνικές προβλέψεων, δεν είναι πάντα εύκολη διαδικασία. Υπάρχουν περιπτώσεις ελλειπουσών ή μηδενικών τιμών, οι οποίες δημιουργούν προβλήματα στην εφαρμογή των περισσότερων στατιστικών μεθόδων πρόβλεψης και, ως εκ του τούτου, πρέπει να αντιμετωπισθούν.

Οι κενές τιμές αφορούν περιπτώσεις όπου η τιμή κάποιων περιόδων δεν είχε καταγραφεί και αποθηκευτεί στη βάση δεδομένων. Ο λόγος μπορεί να οφείλεται σε αστοχία του πληροφοριακού συστήματος ή σε λάθος χειρισμό από την πλευρά του υπεύθυνου χρήστη. Ανεξαρτήτως αιτίας, ακολουθείται μία από τις παρακάτω διαδικασίες εκτίμησης της ελλειπούσας τιμής, ανάλογα με την περίπτωση [17]:

- Γίνεται προσπάθεια εύρεσης της κενής τιμής από άλλες πηγές ή απευθείας ορισμός αυτής, αν υπάρχει ασφαλής κριτική εκτίμηση για το ύψος στο οποίο κυμάνθηκε.
- Η κενή τιμή ορίζεται ως το ημιάθροισμα (μέσος όρος) της προηγούμενης και της επόμενης παρατήρησης, όταν η χρονοσειρά χαρακτηρίζεται από στασιμότητα και δεν παρατηρείται εποχιακή συμπεριφορά.
- Αν η χρονοσειρά παρουσιάζει σαφή εποχιακή συμπεριφορά, τότε η κενή τιμή ορίζεται ως ο μέσος όρος των τιμών των αντίστοιχων περιόδων.

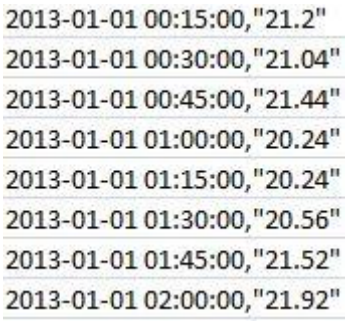
Στο πλαίσιο της παρούσας εργασίας, οι κενές τιμές ορίστηκαν ως το ημιάθροισμα της προηγούμενης και της επόμενης παρατήρησης. Στην περίπτωση όπου υπήρχαν δύο ή περισσότερες διαδοχικές κενές τιμές, τότε όλες ελάμβαναν την ίδια τιμή, δηλαδή το ημιάθροισμα της προηγούμενης και της επόμενης πραγματικής τιμής.

Οι μηδενικές τιμές αφορούν μία ειδική κατηγορία των «προβληματικών» τιμών μιας χρονοσειράς. Αυτές μπορεί να αφορούν κενές τιμές, οι οποίες καταγράφηκαν αυτόματα από το πληροφοριακό σύστημα ως μηδενικές ή πραγματικά μηδενικές τιμές (δηλαδή μηδενική ζήτηση). Στην πρώτη περίπτωση εφαρμόζεται η κατάλληλη διαδικασία διαχείρισης κενών τιμών. Η δεύτερη περίπτωση αναφέρεται σε χρονοσειρές διακοπτόμενης ζήτησης και δεν απαιτείται καμία διαδικασία μεταβολής αυτών.

Κατά την καταγραφή των δεδομένων μας, δεν αντιμετωπίσαμε κανένα πρόβλημα μηδενικής τιμής.

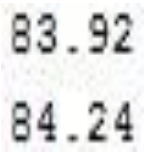
3.2. ΣΥΝΑΘΡΟΙΣΗ ΤΩΝ ΑΝΑ ΤΕΤΑΡΤΟ ΔΕΔΟΜΕΝΩΝ ΣΕ ΑΝΤΙΣΤΟΙΧΑ ΩΡΙΑΙΑ

Όπως αναφέρθηκε στην παράγραφο 1.1, τα δεδομένα που συλλέξαμε ήταν ανά ένα τέταρτο της ώρας. Καθώς σκοπός μας ήταν να προβλέψουμε την ωραία ενεργειακή κατανάλωση του super market, έπρεπε να συναθροίσουμε τέσσερις τιμές σε μία, σύμφωνα με το εξής παράδειγμα:



Εικόνα 11: Δεδομένα πριν την ωριαία συνάθροιση

Οι παραπάνω οκτώ τιμές αντικαθίστανται από τις εξής δύο:



Εικόνα 12: Δεδομένα μετά την ωριαία συνάθροιση

Η τιμή 83.92 αντιστοιχεί στην ωριαία κατανάλωση μέχρι τη 1 η ώρα και η 84.24 μέχρι τις 2.

3.3. ΚΩΔΙΚΑΣ JAVA ΓΙΑ ΔΙΑΧΕΙΡΙΣΗ ΚΕΝΩΝ ΤΙΜΩΝ ΚΑΙ ΩΡΙΑΙΑ ΣΥΝΑΘΡΟΙΣΗ ΔΕΔΟΜΕΝΩΝ - ΕΠΕΞΗΓΗΣΗ

Στην παράγραφο αυτή παραθέτουμε τον κώδικα Java που γράψαμε προκειμένου να υλοποιήσουμε τις διαδικασίες που αναφέρονται στις παραγράφους 3.1 και 3.2, ενώ οι υποπαράγραφοι που ακολουθούν εξηγούν τις επιμέρους λειτουργίες του κώδικα καθώς και την εν γένει διαδικασία προετοιμασίας και εκτέλεσης του προγράμματος.

```
package metaptyxiakh;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import static java.lang.Math.round;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.TimeZone;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;

public class MissingValuesAndHourlyAggregation {

    public ArrayList <Double> mvaha(ArrayList <Double> data, ArrayList
<String> timestamps, SimpleDateFormat sdf, Integer
numberOfValuesPerHour) throws ParseException, FileNotFoundException,
IOException, org.json.simple.parser.ParseException {

        ArrayList <Long> epochTimestamps = new ArrayList();
        Date date = new Date();
        Integer numberOfMissingValues = 0;
        Date dateStart = new Date();
        String timestampStart = "";
        Long startEpoch = 0L;
        SimpleDateFormat sdfStart = new SimpleDateFormat("yyyy-MM-
dd");

        String pathToConfFile
="C:\\Users\\user\\Desktop\\configuration.json";
        FileReader configurationFile = new FileReader(pathToConfFile);
        JSONParser parser = new JSONParser();
        JSONObject configurationObject = (JSONObject)
parser.parse(configurationFile);
        JSONObject start = (JSONObject)
configurationObject.get("Start_Time");

        timestampStart = (String) start.get("Year") + "-" + (String)
start.get("Month") + "-01";
        //System.out.println(timestampStart + "\n");
        sdfStart.setTimeZone(TimeZone.getTimeZone("GMT"));
        dateStart = sdfStart.parse(timestampStart);
        startEpoch = dateStart.getTime();

        for (int i=0; i<timestamps.size(); i++) {

            date = sdf.parse(timestamps.get(i));
            long epoch = date.getTime();
            epochTimestamps.add(epoch);
        }


        if(round((double)(epochTimestamps.get(0)-
startEpoch)/(60000*60/numberOfValuesPerHour)) - 1.0 >0) {
                //System.out.println(timestampStart);
                double nextValue = data.get(0);
                double interval = round((double)(epochTimestamps.get(0)
- startEpoch)/(60000*60/numberOfValuesPerHour)) - 1.0;
                int intInterval = (int)interval;
                //System.out.println(intInterval);
```

```
                for (int j=0; j<intInterval; j++) {
                    data.add(numberOfMissingValues+j, nextValue);
//System.out.println(data.get(numberOfMissingValues+j));
                }
                //System.out.println("\n");
                numberOfMissingValues = numberOfMissingValues +
intInterval;
            }

        for (int i=0; i<epochTimestamps.size()-1; i++) {

            if(round((double)(epochTimestamps.get(i+1) -
epochTimestamps.get(i))/(60000*60/numberOfValuesPerHour)) - 1.0 >0) {
                System.out.println(timestamps.get(i));
                double previousValue =
data.get(i+numberOfMissingValues);
                double nextValue = data.get(i+1+numberOfMissingValues);
                double interval =
round((double)(epochTimestamps.get(i+1) -
epochTimestamps.get(i))/(60000*60/numberOfValuesPerHour)) - 1.0;
                int intInterval = (int)interval;
                System.out.println(intInterval);
                for (int j=0; j<intInterval; j++) {
                    data.add(i+1+numberOfMissingValues+j,
(previousValue + nextValue)/2); System.out.println(data.get(i+1+
numberOfMissingValues+j));
                }
                numberOfMissingValues = numberOfMissingValues +
intInterval;
                System.out.println("\n");
            }
        }


        Integer numberOfLastValues = 0;
        numberOfLastValues = data.size() % numberOfValuesPerHour;
        int intInterval = numberOfValuesPerHour - numberOfLastValues;
        if (intInterval != numberOfValuesPerHour) {
            for (int j=0; j<intInterval; j++) {
                data.add(data.get(data.size()-1));
//System.out.println(data.get(data.size()-1));
            }
            //System.out.println("\n");
            numberOfMissingValues = numberOfMissingValues +
intInterval;
        }

        //System.out.println(numberOfMissingValues);
        //System.out.println("\n");

        for (int i=0; i<data.size(); i++) {
            if (i % numberOfValuesPerHour == numberOfValuesPerHour-1)
{

                Double aggregation = 0.0;
                for (int j=i; j>i-numberOfValuesPerHour; j--) {
                    aggregation = aggregation + data.get(j);

                }
                data.set(i, aggregation);
            }
        }
```

```
            //System.out.println(data.size());
            //System.out.println("\n");

            for (int i=data.size()-1; i>=0; i--) {
                if (i % numberOfValuesPerHour != numberOfValuesPerHour-1)
{
                    data.remove(i);
                }
            }

            return data;
        }
}
```

Εικόνα 13: Java κλάση "MissingValuesAndHourlyAggregation"

3.3.1. ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗ

Η κλάση "MissingValuesAndHourlyAggregation" διαθέτει μία μέθοδο την "mvaha" η οποία και υλοποιεί τόσο την διαχείριση των κενών τιμών όσο και την ωριαία συνάθροιση των δεδομένων. Στη συνέχεια επεξηγούμε τις διάφορες παραμέτρους της μεθόδου:

1. data: τα αρχικά, ακατέργαστα δεδομένα που συλλέξαμε (παράγραφος 1.1)
2. timestamps: οι χρονικές στιγμές των παραπάνω δεδομένων
3. sdf: η μορφή αναπαράστασης των παραπάνω χρονικών στιγμών (στην περίπτωση μας "yyyy-MM-dd HH:mm:ss")
4. numberOfValuesPerHour: ο αριθμός αυτός είναι 4 εάν τα αρχικά δεδομένα είναι ανα τέταρτο, 2 εάν είναι ανά μισάωρο κ.ο.κ.

Οι παραέμτροι data και timestamps εξάγονται κατ' ευθείαν από το αρχείο των δεδομένων (M135.csv), ενώ οι παράμετροι sdf και numberOfValuesPerHour διαβάζονται από το αρχείο παραμετροποιήσης (configuration.json), το οποίο συμπληρώνεται από τον χρήστη της μεθόδου. Ένα παράδειγμα του αρχείου παραμετροποίησης παρουσιάζεται παρακάτω:

```
{
    "Latitude":"37.951575",
    "Longitude":"23.66700232",
    "Start_Time":{
        "Month":"1",
        "Year":"2013",
        "Leap_Year":"no"
    },
    "End_Time":{
        "Month":"4",
        "Year":"2013",
        "Leap_Year":"no"
    },
    "Data_Timestamp_Format":"yyyy-MM-dd HH:mm:ss",
    "Number_of_Values_per_Hour":"4"
}
```


Εικόνα 14: Παράδειγμα αρχείου παραμετροποίησης I

Τα πεδία "Data_Timestamp_Format" και "Number_of_Values_per_Hour" περιγράφονται παραπάνω, ενώ τα υπόλοιπα θα περιγραφούν σε επόμενα κεφάλαια. Το αρχείο παραμετροποίησης γράφτηκε σε μορφή JSON [18], συνεπώς είναι απαραίτητη η προσθήκη της Java βιβλιοθήκης “json-simple-1.1.jar”.

3.3.2. ΚΕΝΕΣ ΤΙΜΕΣ

Το παρακάτω κομμάτι κώδικα εντοπίζει την ύπαρξη κενών τιμών στα αρχικά δεδομένα μας και τις συμπληρώνει ως το ημιάθροισμα της προηγούμενης και επόμενης πραγματικής τιμής:

```
for (int i=0; i<epochTimestamps.size()-1; i++) {

            if(round((double)(epochTimestamps.get(i+1) -
epochTimestamps.get(i))/(60000*60/numberOfValuesPerHour)) - 1.0 >0) {
                System.out.println(timestamps.get(i));
                double previousValue =
data.get(i+numberOfMissingValues);
                double nextValue = data.get(i+1+numberOfMissingValues);
                double interval =
round((double)(epochTimestamps.get(i+1) -
epochTimestamps.get(i))/(60000*60/numberOfValuesPerHour)) - 1.0;
                int intInterval = (int)interval;
                System.out.println(intInterval);
                for (int j=0; j<intInterval; j++) {
```

```
                    data.add(i+1+numberOfMissingValues+j,
(previousValue + nextValue)/2); System.out.println(data.get(i+1+
numberOfMissingValues+j));
                }
                numberOfMissingValues = numberOfMissingValues +
intInterval;
                System.out.println("\n");
            }

        }
```

Εικόνα 15: Επιμέρους κώδικας για διαχείριση κενών τιμών

Ο κώδικας χρησιμοποιεί τις έτοιμες Java κλάσεις “Date” [19] και “SimpleDateFormat” [20], με σκοπό την μετατροπή των timestamps που διαβάζονται από το αρχείο δεδομένων, σε μορφή μεγάλων ακεραίων (long). Αυτή η μορφή αναπαράστασης των timestamps, που είναι γνωστή σαν epoch time format και έχει σαν μονάδα μέτρησης το ms, μας δίνει τη δυνατότητα να τα συγκρίνουμε χρησιμοποιώντας απλές αλγεβρικές πράξεις. Κατά συνέπεια, η ύπαρξη κενής τιμής εντοπίζεται εάν η χρονική διαφορά μεταξύ δύο διαδοχικών timestamps είναι μεγαλύτερη από 1 ώρα (=60.000*60 ms) / numberOfValuesPerHour, που στην περίπτωση μας (numberOfValuesPerHour=4) ισούται με 15 λεπτά. Κάθε φορά που ο κώδικας εντοπίζει μία κενή τιμή, τυπώνει το αμέσως προηγούμενο πραγματικό timestamp, το πλήθος των διαδοχικών κενών τιμών, καθώς και την τιμή με την οποία συμπληρώνονταιβ (ημιάθροισμα της προηγούμενης και επόμενης πραγματικής τιμής).

3.3.3. ΩΡΙΑΙΑ ΣΥΝΑΘΡΟΙΣΗ

Αμέσως μετά τον εντοπισμό και την συμπλήρωση των κενών τιμών, λαμβάνει χώρα διαδικασία της ωριαίας συνάθροισης των δεδομένων, όπως αυτή περιγράφεται στην παράγραφο 3.2. Ακολουθεί το σχετικό κομμάτι κώδικα:

```
for (int i=0; i<data.size(); i++) {
            if (i % numberOfValuesPerHour == numberOfValuesPerHour-1)
{

                Double aggregation = 0.0;
                for (int j=i; j>i-numberOfValuesPerHour; j--) {
                    aggregation = aggregation + data.get(j);

                }
                data.set(i, aggregation);
            }
        }

        //System.out.println(data.size());
        //System.out.println("\n");

        for (int i=data.size()-1; i>=0; i--) {
```

```
            if (i % numberOfValuesPerHour != numberOfValuesPerHour-1)
{
                data.remove(i);
            }
        }
```

Εικόνα 16: Επιμέρους κώδικας για ωριαία συνάθροιση δεδομένων

3.3.4. ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΗΣ – ΕΚΤΕΛΕΣΗ ΚΩΔΙΚΑ

2013-02-04 00:15:00,"22.96"
2013-02-04 00:30:00,"24.88"
2013-02-04 00:45:00,"24.16"
2013-02-04 01:00:00,"24.4"
2013-02-04 01:15:00,"25.12"
2013-02-04 01:30:00,"23.6"
2013-02-04 01:45:00,"24.64"
2013-02-04 02:00:00,"24.72"
2013-02-04 02:15:00,"24.96"
2013-02-04 02:30:00,"24.96"
2013-02-04 02:45:00,"26.8"
2013-02-04 03:00:00,"25.92"
2013-02-04 03:15:00,"26.56"
2013-02-04 03:30:00,"27.6"
2013-02-04 03:45:00,"26"
2013-02-04 04:00:00,"26.96"
2013-02-04 04:15:00,"27.52"
2013-02-04 04:30:00,"27.84"
2013-02-04 04:45:00,"26.4"
2013-02-04 05:00:00,"25.04"
2013-02-04 05:15:00,"25.76"
2013-02-04 05:30:00,"27.2"
2013-02-04 05:45:00,"39.44"
2013-02-04 06:00:00,"41.04"
2013-02-04 06:15:00,"43.36"
2013-02-04 06:30:00,"47.36"
2013-02-04 06:45:00,"48.24"
2013-02-04 07:00:00,"47.28"
2013-02-04 07:15:00,"44.16"
2013-02-04 07:30:00,"29.6"
2013-02-04 08:15:00,"9.76"
2013-02-04 08:30:00,"45.84"
2013-02-04 08:45:00,"49.52"

2013-02-04 09:00:00,"46.72"
2013-02-04 09:15:00,"47.52"
2013-02-04 09:30:00,"48.48"
2013-02-04 09:45:00,"46.8"
2013-02-04 10:00:00,"46.32"
2013-02-04 10:15:00,"49.2"
2013-02-04 10:30:00,"46.96"
2013-02-04 10:45:00,"45.44"
2013-02-04 11:00:00,"44.24"
2013-02-04 11:15:00,"46.16"
2013-02-04 11:30:00,"45.52"
2013-02-04 11:45:00,"46.16"
2013-02-04 12:00:00,"46.32"
2013-02-04 12:15:00,"46.48"
2013-02-04 12:30:00,"47.2"
2013-02-04 12:45:00,"48.32"
2013-02-04 13:00:00,"47.52"
2013-02-04 13:15:00,"47.76"
2013-02-04 13:30:00,"47.12"
2013-02-04 13:45:00,"45.84"
2013-02-04 14:00:00,"45.52"
2013-02-04 14:15:00,"46.8"
2013-02-04 14:30:00,"44.64"
2013-02-04 14:45:00,"45.44"
2013-02-04 15:00:00,"45.44"
2013-02-04 15:15:00,"46.24"
2013-02-04 15:30:00,"47.12"
2013-02-04 15:45:00,"46.08"
2013-02-04 16:00:00,"47.04"
2013-02-04 16:15:00,"47.6"
2013-02-04 16:30:00,"46.8"
2013-02-04 16:45:00,"43.92"
2013-02-04 17:00:00,"44.96"
2013-02-04 17:15:00,"48"
2013-02-04 17:30:00,"47.28"
2013-02-04 17:45:00,"47.2"
2013-02-04 18:00:00,"48.24"
2013-02-04 18:15:00,"47.92"
2013-02-04 18:30:00,"48.4"
2013-02-04 18:45:00,"48.4"
2013-02-04 19:00:00,"48.24"
2013-02-04 19:15:00,"47.52"
2013-02-04 19:30:00,"47.52"
2013-02-04 19:45:00,"47.2"
2013-02-04 20:00:00,"47.92"
2013-02-04 20:15:00,"48.56"

2013-02-04 20:30:00,"46.24"
2013-02-04 20:45:00,"48.72"
2013-02-04 21:00:00,"49.76"
2013-02-04 21:15:00,"47.2"
2013-02-04 21:30:00,"44"
2013-02-04 21:45:00,"27.36"
2013-02-04 22:00:00,"28.16"
2013-02-04 22:15:00,"27.28"
2013-02-04 22:30:00,"25.36"
2013-02-04 22:45:00,"23.76"
2013-02-04 23:00:00,"21.6"
2013-02-04 23:15:00,"21.92"
2013-02-04 23:30:00,"21.76"
2013-02-04 23:45:00,"21.28"
2013-02-04 23:58:58,"24.24"

Θα εκτελέσουμε τον κώδικα της παραγράφου 3.3 στο παραπάνω υποσύνολο δεδομένων , που αφορά τις τιμές μίας συγκεκριμένης (τυχαίας) ημέρας (04/02/2013). Για να καταστεί αυτό δυνατό χρειάζεται η εξής μετατροπή στον κώδικα:

Από

```
timestampStart = (String) start.get("Year") + "-" + (String) start.get("Month") + "-01";
```

Σε

```
timestampStart = (String) start.get("Year") + "-" + (String) start.get("Month") + "-04";
```

Επίσης φτιάξαμε την δοκιμαστική Java κλάση "TestMissingValuesAndHourlyAggregation" την οποία τρέχουμε:

```
package metaptyxiakh;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.TimeZone;
import org.json.simple.JSONObject;
```

```
import org.json.simple.parser.JSONParser;

public class TestMissingValuesAndHourlyAggregation {


    public static void main(String[] args) throws
FileNotFoundException, IOException, ParseException,
UnsupportedEncodingException, org.json.simple.parser.ParseException {

        ArrayList <Double> data = new ArrayList();
        ArrayList <String> timestamps = new ArrayList();

        String csvFile = "C:\\Users\\user\\Desktop\\data\\M135.csv";

        String pathToConfFile
="C:\\Users\\user\\Desktop\\configuration.json";
        FileReader configurationFile = new FileReader(pathToConfFile);
        JSONParser parser = new JSONParser();
        JSONObject configurationObject = (JSONObject)
parser.parse(configurationFile);

br = null;
        br = new BufferedReader(new FileReader(csvFile));
        String line = br.readLine();

        while ((line = br.readLine()) != null) {
            if((line.split(",")[0].substring(4,5).equals("3") && 2<=
Integer.parseInt(line.split(",")[0].substring(6,8)) &&
Integer.parseInt(line.split(",")[0].substring(6,8))<3) &&
line.split(",")[0].substring(9,11).equals("04")) {

data.add(Double.parseDouble(line.split(",")[1].replace("\"", "")));
                timestamps.add(line.split(",")[0].replace("\"", ""));
            }
        }

        System.out.println("πλήθος αρχικών δεδομένων = " + data.size()
+ "\n");

        SimpleDateFormat sdf = new SimpleDateFormat((String)
configurationObject.get("Data_Timestamp_Format"));
        sdf.setTimeZone(TimeZone.getTimeZone("GMT"));
        MissingValuesAndHourlyAggregation examo = new
MissingValuesAndHourlyAggregation();
        data = examo.mvaha(data, timestamps, sdf,
Integer.parseInt((String)
configurationObject.get("Number_of_Values_per_Hour")));

        System.out.println("πλήθος συγκεντρωτικών δεδομένων = " +
data.size()+ "\n");
        System.out.println("πρώτη συγκεντρωτική (ωριαία) τιμή = " +
data.get(0));
        System.out.println("τελευταία συγκεντρωτική (ωριαία) τιμή = "
+ data.get(data.size()-1)+"\n");
}

}
```

Εικόνα 17: Java κλάση "TestMissingValuesAndHourlyAggregation"

και λαμβάνουμε τα παρακάτω αποτελέσματα:

```
πλήθος αρχικών δεδομένων = 94

2013-02-04 07:30:00
2
19.68
19.68


πλήθος συγκεντρωτικών δεδομένων = 24

πρώτη συγκεντρωτική (ωριαία) τιμή = 96.4
τελευταία συγκεντρωτική (ωριαία) τιμή = 89.2
```

Εικόνα 18: Αποτελέσματα Java κλάσης "TestMissingValuesAndHourlyAggregation"

Πράγματι:

- Υπάρχουν 2 κενές τιμές στις 7:45:00 και 8:00:00 (δηλαδή αμέσως μετά τις 7:30:00) οι οποίες και συμπληρώνονται με την τιμή (29.6 + 9.76) / 2 = 19.68
- Το πλήθος των αρχικών δεδομένων είναι 24 ώρες * 4 τιμές / ώρα – 2 κενές τιμές = 94
- Το πλήθος των συγκεντρωτικών δεδομένων πρέπει να είναι 24 (όσες και οι ώρες της ημέρας)
- Η πρώτη συγκεντρωτική τιμή ισούται με το άθροισμα των τεσσάρων πρώτων αρχικών, δηλαδή 22.96 + 24.88 + 24.16 + 24.4 = 96.4
- Η τελευταία συγκεντρωτική τιμή ισούται με το άθροισμα των τεσσάρων τελευταίων αρχικών, δηλαδή 21.92 + 21.76 + 21.28 + 24.24 = 89.2

4. ΜΕΘΟΔΟΣ Ι: SES ΜΕ ΕΠΟΧΙΑΚΟΤΗΤΑ ΜΗΚΟΥΣ 168

Η περιγραφή της μεθόδου Ι βρίσκεται στην υποπαράγραφο 1.4.1.

4.1. ΚΩΔΙΚΑΣ JAVA ΓΙΑ ΚΛΑΣΙΚΗ ΜΕΘΟΔΟ ΑΠΟΣΥΝΘΕΣΗΣ

Σε αυτήν την υποπαράγραφο παρουσιάζουμε τον κώδικα που υλοποιεί την σταθερή πολλαπλασιαστική μέθοδο ή κλασική μέθοδο αποσύνθεσης (Fixed Multiplicative Method). Η Java κλάση "SeasonalDecompostion" περιέχει τη μέθοδο "sd", η οποία δέχεται σαν είσοδο το σύνολο των δεδομένων (data) καθώς και το μήκος της εποχιακότητας (p). Σαν έξοδο δίνει τους ποσοστιαίους δείκτες εποχιακότητας της χρονοσειράς:

```
package metaptyxiakh;

import java.util.ArrayList;

public class SeasonalDecomposition {

    public ArrayList <Double> sd (ArrayList <Double> data, Integer p)
{

        ArrayList <Double> simpleMovingAverage = new ArrayList();
        ArrayList <Double> centeredMovingAverage = new ArrayList();
        ArrayList <Double> seasonalityRatios = new ArrayList();
        ArrayList <Double> periodSeasonalityRatios = new ArrayList();
        ArrayList <Double> normalisedSeasonalityIndexes = new
ArrayList();
        Double sumNonNormalisedSeasonalityIndexes = 0.0;


            for (int i=0; i<data.size() - p + 1; i++) {

                Double currentSMA = 0.0;

                for (int j=0; j<p; j++) {

                    currentSMA = currentSMA + data.get(i+j);
                }

                simpleMovingAverage.add(currentSMA/p);

            }

            if (p % 2 == 0) {
            for (int i=0; i<simpleMovingAverage.size() - 1; i++) {

                centeredMovingAverage.add((simpleMovingAverage.get(i)
+ simpleMovingAverage.get(i+1))/2);
```

```
            }

                        }
            else        {
                centeredMovingAverage = simpleMovingAverage;
                        }

            for (int i=(int)(p/2); i<data.size() - (int)(p/2); i++) {


seasonalityRatios.add(data.get(i)/centeredMovingAverage.get(i -
(int)(p/2))*100);
            }

                for (int i=0; i<p; i++) {

                    Double maxValue = null;
                    Double minValue = null;
                    Double meanRatio = 0.0;

                    for (int j=0; j<seasonalityRatios.size(); j++) {

                        if (j % p == i) {


periodSeasonalityRatios.add(seasonalityRatios.get(j));

                                        }
                                                                    }

                    if (seasonalityRatios.size()/p > 2) {

                    maxValue = periodSeasonalityRatios.get(0);
                    minValue = periodSeasonalityRatios.get(0);

                    for (int j=0; j<periodSeasonalityRatios.size();
j++) {

                            if (maxValue <
periodSeasonalityRatios.get(j)) {

                                maxValue =
periodSeasonalityRatios.get(j);

}

                            if (minValue >
periodSeasonalityRatios.get(j)) {

                                minValue =
periodSeasonalityRatios.get(j);

}

}
                    periodSeasonalityRatios.remove(maxValue);
                    periodSeasonalityRatios.remove(minValue);
                                                    }

                    for (int j=0; j<periodSeasonalityRatios.size();
j++) {
```

```
                        meanRatio = meanRatio +
periodSeasonalityRatios.get(j);

}

normalisedSeasonalityIndexes.add(meanRatio/periodSeasonalityRatios.siz
e());

                    periodSeasonalityRatios.clear();
                                        }

                for (int i=0; i<normalisedSeasonalityIndexes.size();
i++) {

                    sumNonNormalisedSeasonalityIndexes =
sumNonNormalisedSeasonalityIndexes +
normalisedSeasonalityIndexes.get(i);

}

                for (int i=0; i<normalisedSeasonalityIndexes.size();
i++) {

                    normalisedSeasonalityIndexes.set(i,
normalisedSeasonalityIndexes.get(i) * p * 100 /
sumNonNormalisedSeasonalityIndexes);


}

                for (int i=0; i<(int)(p/2); i++) {

                    normalisedSeasonalityIndexes.add(i,
normalisedSeasonalityIndexes.get(normalisedSeasonalityIndexes.size() -
(int)(p/2) + i) );
                                                        }

                for (int i=0; i<(int)(p/2); i++) {


normalisedSeasonalityIndexes.remove(normalisedSeasonalityIndexes.size(
) - 1);
                                                        }

                for (int i=p; i<data.size(); i++) {


normalisedSeasonalityIndexes.add(normalisedSeasonalityIndexes.get(i-
p));
                                                         }

        return normalisedSeasonalityIndexes;
    }
}
```

Εικόνα 19: Κώδικας Java για την κλασική μέθοδο αποσύνθεσης

4.1.1. ΕΚΤΕΛΕΣΗ ΚΩΔΙΚΑ

Προκειμένου να επαληθεύσουμε τον κώδικα της παραγράφου 4.1, φτιάξαμε την δοκιμαστική Java κλάση “TestSeasonalDecompostion”:

```
package metaptyxiakh;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;

public class TestSeasonalDecomposition {

    public static void main(String[] args) throws
FileNotFoundException, IOException {

        String csvFile =
"C:\\Users\\user\\Desktop\\inputDecomposePage76.txt"; //page 76
        BufferedReader br = null;
        br = new BufferedReader(new FileReader(csvFile));
        String line = br.readLine();
        ArrayList <Double> data = new ArrayList();
        ArrayList <Double> indexes = new ArrayList();

        String[] stringValues = line.split(",");
        for (int i=0; i<stringValues.length; i++) {
            data.add(Double.parseDouble(stringValues[i]));
        }

        SeasonalDecomposition ex = new SeasonalDecomposition();
        indexes = ex.sd(data, 4);
        for (int i=0; i<indexes.size(); i++) {
            System.out.println(indexes.get(i));
        }
    }

}
```

Εικόνα 20: Java κλάση “TestSeasonalDecomposition”

Τα δεδομένα που χρησιμοποιούνται για την επαλήθευση του κώδικα είναι αυτά της σελίδας 76 του βιβλίου [17], τα οποία και παρατίθενται παρακάτω για ευκολία:

```
4109,3874,3842,3946,4207,3850,4030,4260,4193,4051,4126,4445,4344
,4319,4571,4576,4699,4614,4613,4738
```

Τρέχοντας την δοκιμαστική κλάση λαμβάνουμε τα εξής αποτελέσματα:

```
101.79171899247102
97.4806226306173
98.5375338251076 9
102.1901245518039 8
101.79171899247102
97.4806226306173
98.5375338251076 9
102.1901245518039 8
101.79171899247102
97.4806226306173
98.5375338251076 9
102.1901245518039 8
101.79171899247102
97.4806226306173
98.5375338251076 9
102.1901245518039 8
101.79171899247102
97.4806226306173
98.5375338251076 9
102.1901245518039 8
```

Εικόνα 21: Αποτελέσματα Java κλάσης "TestSeasonalDecomposition"

Τα οποία και ταυτίζονται με τα αντίστοιχα του βιβλίου.

Ο κώδικας που υλοποιήσαμε για την κλασική μέθοδο αποσύνθεσης επαληθεύτηκε και με πολλά άλλα σύνολα δεδομένων.

4.2. ΚΩΔΙΚΑΣ ΜΕΘΟΔΟΥ Ι

```
package metaptyxiakh;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import static java.lang.Math.sqrt;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.TimeZone;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;


public class SESWithSeasonality168 {
```

```
    public static void main(String[] args) throws
FileNotFoundException, IOException, ParseException,
UnsupportedEncodingException, org.json.simple.parser.ParseException {

        ArrayList <Double> data = new ArrayList();
        ArrayList <String> timestamps = new ArrayList();
        ArrayList <Double> indexes = new ArrayList();
        ArrayList <Double> forecasts = new ArrayList();
        ArrayList <Double> validation = new ArrayList();
        Double[] coef = new Double[2];
        ArrayList <Double> dataTest = new ArrayList();
        ArrayList <Double> forecastsTest = new ArrayList();
        Integer k = 0;

        String csvFile = "C:\\Users\\user\\Desktop\\data\\M135.csv";

        String pathToConfFile
="C:\\Users\\user\\Desktop\\configuration.json";
        FileReader configurationFile = new FileReader(pathToConfFile);
        JSONParser parser = new JSONParser();
        JSONObject configurationObject = (JSONObject)
parser.parse(configurationFile);
        JSONObject start = (JSONObject)
configurationObject.get("Start_Time");
        JSONObject end = (JSONObject)
configurationObject.get("End_Time");
        Integer startMonth =
Integer.parseInt((String)start.get("Month"));
        Integer endMonth = Integer.parseInt((String)end.get("Month"));
        Integer startYear =
Integer.parseInt((String)start.get("Year"));

        if (startYear == 2012) {

br = null;
        br = new BufferedReader(new FileReader(csvFile));
        String line = br.readLine();

        while ((line = br.readLine()) != null) {
            if((line.split(",")[0].substring(4,5).equals("2") &&
startMonth<= Integer.parseInt(line.split(",")[0].substring(6,8)) &&
Integer.parseInt(line.split(",")[0].substring(6,8))<13) ||
(line.split(",")[0].substring(4,5).equals("3") && 1<=
Integer.parseInt(line.split(",")[0].substring(6,8)) &&
Integer.parseInt(line.split(",")[0].substring(6,8))<(endMonth + 1))) {

data.add(Double.parseDouble(line.split(",")[1].replace("\"", "")));
               timestamps.add(line.split(",")[0].replace("\"", ""));
            }
        }
        }
        else {

            BufferedReader br = null;
            br = new BufferedReader(new FileReader(csvFile));
            String line = br.readLine();

        while ((line = br.readLine()) != null) {
            if((line.split(",")[0].substring(4,5).equals("2") && 13<=
Integer.parseInt(line.split(",")[0].substring(6,8)) &&
Integer.parseInt(line.split(",")[0].substring(6,8))<13) ||
```

```
(line.split(",")[0].substring(4,5).equals("3") && startMonth<=
Integer.parseInt(line.split(",")[0].substring(6,8)) &&
Integer.parseInt(line.split(",")[0].substring(6,8))<(endMonth + 1))) {

data.add(Double.parseDouble(line.split(",")[1].replace("\"", "")));
                timestamps.add(line.split(",")[0].replace("\"", ""));
            }
        }

        }


        SimpleDateFormat sdf = new SimpleDateFormat((String)
configurationObject.get("Data_Timestamp_Format"));
        sdf.setTimeZone(TimeZone.getTimeZone("GMT"));
        MissingValuesAndHourlyAggregation aggr = new
MissingValuesAndHourlyAggregation();
        data = aggr.mvaha(data, timestamps, sdf,
Integer.parseInt((String)
configurationObject.get("Number_of_Values_per_Hour")));

        if
(endMonth==1||endMonth==3||endMonth==5||endMonth==7||endMonth==8||endM
onth==10||endMonth==12) {k=24;}
        if (endMonth==4||endMonth==6||endMonth==9||endMonth==11)
{k=23;}
        if (endMonth==2 &&
end.get("Leap_Year").toString().equals("no")) {k=21;}
        if (endMonth==2 &&
end.get("Leap_Year").toString().equals("yes")) {k=22;}

        Integer n = data.size()-k*24;

        for (int i=0; i<n; i++) {
            validation.add(data.get(i));
        }

        SeasonalDecomposition dec = new SeasonalDecomposition();
        indexes = dec.sd(validation, 168);

        for (int i=0; i<validation.size(); i++) {
            validation.set(i,
100*validation.get(i)/indexes.get(i%168));
        }

        SimpleExponentialSmoothing test = new
SimpleExponentialSmoothing();
        LinearRegression lr = new LinearRegression();
        coef = lr.lrl(validation);
        Errors err = new Errors();
        Double minMSE = 1000000.0;
        Double amin = 0.0;
        //Double forecastFinal = 0.0;

        for (Integer i=0; i<10001; i++) {
            forecasts = test.ses(validation, i.doubleValue()/10000,
coef[0], 1);
            if (err.mse(validation, forecasts)<minMSE) {
                minMSE = err.mse(validation, forecasts); amin =
i.doubleValue()/10000;
                //forecastFinal = forecasts.get(forecasts.size()-1);
            }
```

```
        }

        //System.out.println(forecastFinal);
        //System.out.println(sqrt(minMSE));
        System.out.println("βέλτιστο a = " + amin +"\n");

        forecasts.clear();
        forecasts = test.ses(validation, amin, coef[0], 168);

        for (int i=n; i<n+168; i++) {
            dataTest.add(data.get(i));

forecastsTest.add(forecasts.get(i)*indexes.get(i%168)/100);
        }
        if (dataTest.size() == forecastsTest.size() && dataTest.size()
== 168) {
            System.out.println("mape = " + err.mape(dataTest,
forecastsTest));
            System.out.println("mse = " + sqrt(err.mse(dataTest,
forecastsTest)) + "\n");
            //System.out.println(sqrt(exam.mse(forecastsTest,
dataTest)));
        }

    }
```

Εικόνα 22: Κώδικας Java για την μέθοδο Ι (κλάση "SESWithSeasonality168")

4.3. ΑΠΟΤΕΛΕΣΜΑΤΑ

ΠΛΗΘΟΣ ΙΣΤΟΡΙΚΩΝ ΜΗΝΩΝ (+ 1η ΕΒΔΟΜΑΔΑ)	2η ΕΒΔΟΜΑΔΑ 2013	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
12		17,37	16,54	7,82	8,51	5,70	14,27	14,71	14,47	6,70	14,58	6,39	15,42
11		17,40	16,14	7,61	9,69	6,25	14,38	14,68	14,28	6,64	14,81	6,80	15,98
10		17,23	15,77	7,15	10,81	6,49	14,36	14,40	14,15	6,52	15,14	7,39	16,49
9		16,75	14,00	7,17	11,17	7,43	14,82	14,34	14,02	6,27	14,46	8,14	16,80
8		15,47	13,11	7,16	11,72	8,05	15,16	14,21	13,71	5,73	13,56	8,42	16,31
7		14,56	12,76	7,15	13,21	8,68	15,64	14,25	13,68	5,60	12,90	8,46	15,03
6		14,19	12,51	7,45	14,22	8,71	17,28	14,35	14,53	5,56	14,61	8,04	13,96
5		14,02	11,85	7,96	16,15	9,98	14,27	15,43	16,12	5,18	15,84	8,41	13,77
4		13,08	10,84	9,25	16,58	7,55	11,25	16,62	14,14	4,06	17,47	8,63	15,53
3		12,10	9,41	10,13	14,60	4,66	9,58	15,92	12,71	5,09	18,06	8,57	16,92
2		10,79	9,34	11,06	10,63	4,83	7,27	13,96	11,04	7,34	17,16	10,44	16,38
1		10,79	6,88	10,81	8,25	3,87	6,89	9,90	11,58	6,13	10,98	11,64	13,96
MIN MAPE		10,79	6,88	7,15	8,25	3,87	6,89	9,90	11,04	4,06	10,98	6,39	13,77
AVERAGE MAPE		14,48	12,43	8,39	12,13	6,85	12,93	14,40	13,70	5,90	14,96	8,44	15,55

Εικόνα 23: Συγκεντρωτικά αποτελέσματα της μεθόδου Ι

4.4. ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΗΣ – ΕΚΤΕΛΕΣΗ ΚΩΔΙΚΑ

Προκειμένου κανείς να εξάγει κάποιο από τα αποτελέσματα της παραγράφου 4.3, πρέπει πρώτα να συμπληρώσει κατάλληλα το αρχείο παραμετροποίησης που περιγράφεται στην Εικόνα 14. Παραδείγματος χάρη, για να εξάγουμε το mape που αντιστοιχεί στην πρόβλεψη της ωριαίας ενεργειακής κατανάλωσης για την δεύτερη εβδομάδα του Φεβρουαρίου 2013 (6,88%) με πλήθος ιστορικών μηνών 1 (Ιανουάριος 2013), το αρχείο πρέπει να έχει την ακόλουθη μορφή:

```
{
    "Latitude":"37.951575",
    "Longitude":"23.66700232",
    "Start_Time":{
        "Month":"1",
        "Year":"2013",
        "Leap_Year":"no"
    },
    "End_Time":{
        "Month":"2",
        "Year":"2013",
        "Leap_Year":"no"
    },
    "Data_Timestamp_Format":"yyyy-MM-dd HH:mm:ss",
    "Number_of_Values_per_Hour":"4"
}
```

Εικόνα 24: Παράδειγμα αρχείου παραμετροποίησης II

Το υποπεδίο “Leap_Year” του πεδίου “Start_Time” παίρνει την τιμή “yes”, μόνο όταν στα ιστορικά δεδομένα εμπεριέχονται και οι τιμές που αντιστοιχούν στον Φεβρουάριο του 2012 (δίσεκτο έτος). Αυτό συμβαίνει όταν προσπαθούμε να προβλέψουμε τις τιμές για την δεύτερη εβδομάδα του Ιανουαρίου 2013 με πλήθος ιστορικών μηνών 12 (17,37%) και 11 (17,40%) ή αντίστοιχα τις τιμές για την δεύτερη εβδομάδα του Φεβρουαρίου 2013 με πλήθος ιστορικών μηνών 11 (16,14%). Επειδή το 2013 δεν ήταν δίσεκτο έτος, το υποπεδίο “Leap_Year” του πεδίου “End_Time” παίρνει πάντα την τιμή “no”.

Μετά την ορθή εισαγωγή των παραμέτρων, τρέχουμε τον κώδικα της παραγράφου 4.2 και λαμβάνουμε το εξής αποτέλεσμα:

```
2013-02-04 07:30:00
2
19.68
19.68


2013-02-21 19:30:00
3
5.76
5.76
5.76


βέλτιστο a = 0.9356

mape = 6.879024711303009
mse = 14.036866377431041
```

Εικόνα 25: Παράδειγμα εκτέλεσης του κώδικα της μεθόδου I

Το πρόγραμμα, εκτός από το mape, τυπώνει το mse αλλά και την τιμή του συντελεστή εξομάλυνσης a, με την οποία παρήχθησαν οι προβλέψεις. Επίσης τυπώνονται οι χρονικές στιγμές αμέσως μετά από τις οποίες εντοπίζονται κενές τιμές στα ιστορικά δεδομένα, καθώς και οι τιμές με τις οποίες αυτές συμπληρώνονται, ακριβώς όπως στην Εικόνα 18.

5. ΜΕΘΟΔΟΣ ΙΙ: ΜΕΘΟΔΟΣ Ι ΜΕ ΑΦΑΙΡΕΣΗ ΕΠΙΔΡΑΣΗΣ ΘΕΡΜΟΚΡΑΣΙΑΣ

Η περιγραφή της μεθόδου ΙΙ βρίσκεται στην υποπαράγραφο 1.4.2.

5.1. ΚΩΔΙΚΑΣ JAVA ΓΙΑ ΕΥΡΕΣΗ ΜΕΣΗΣ ΗΜΕΡΗΣΙΑΣ ΘΕΡΜΟΚΡΑΣΙΑΣ ΜΕΣΩ ΔΙΑΔΙΚΤΥΟΥ - ΕΠΕΞΗΓΗΣΗ

Σε αυτήν την υποπαράγραφο παρουσιάζουμε τον κώδικα που υλοποιήσαμε προκειμένου να αντλούμε τις μέσες ημερήσιες θερμοκρασίες μέσω του διαδικτυακού τόπου "http://www.wunderground.com" [21]. Η Java κλάση "WundergroundConnection" περιέχει τη μέθοδο "getTemperatures" η οποία δέχεται σαν είσοδο τις εξής παραμέτρους:

- latitude: το γεωγραφικό πλάτος της περιοχής όπου μας ενδιαφέρει να βρούμε τις θερμοκρασίες
- longitude: το γεωγραφικό μήκος της περιοχής όπου μας ενδιαφέρει να βρούμε τις θερμοκρασίες
- startMonth: ο πρώτος μήνας της χρονικής περιόδου που μας ενδιαφέρει
- startYear: το ημερολογιακό έτος στο οποίο υπάγεται ο πρώτος μήνας
- isStartLeapYear: εάν το startYear είναι δίσεκτο έτος και ο startMonth (σαν Integer) <=2, τότε η παράμετρος πρέπει να λάβει την τιμή "yes", αλλιώς την τιμή "no"
- endMonth: ο τελευταίος μήνας της χρονικής περιόδου που μας ενδιαφέρει
- endYear: το ημερολογιακό έτος στο οποίο υπάγεται ο τελευταίος μήνας
- isEndLeapYear: εάν το endYear είναι δίσεκτο έτος και ο endMonth (σαν Integer) >=2, τότε η παράμετρος πρέπει να λάβει την τιμή "yes", αλλιώς την τιμή "no"

Όλες οι παραπάνω παράμετροι δηλώνονται από τον χρήστη στο αρχείο παραμετροποίησης (Εικόνα 31).

Σαν έξοδο επιστρέφει τις ζητούμενες μέσες ημερήσιες θερμοκρασίες.

```
package metaptyxiakh;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
```

```
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.Objects;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;

public class WundergroundConnection {

    public ArrayList <Double> getTemperatures(Double latitude, Double
longitude, Integer startMonth, Integer startYear, String
isStartLeapYear, Integer endMonth, Integer endYear, String
isEndLeapYear) throws FileNotFoundException,
UnsupportedEncodingException, IOException, ParseException {


        ArrayList <Double> temperatures = new ArrayList();
        String id = "";
        Boolean completePWS = false;


        URL url = new URL
("http://api.wunderground.com/api/49a120bc6584c08a/geolookup/q/" +
latitude + "," + longitude + ".json");
        HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
        connection.setRequestMethod("GET");
        connection.setRequestProperty("Content-Length","0");
      int responseCode = connection.getResponseCode();
      //System.out.println("Sending 'GET' request to URL : " + url
+"\n");
      //System.out.println("Response Code : " + responseCode +"\n");
in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
inputLine;
response = new StringBuffer();
      while ((inputLine = in.readLine()) != null) {
         response.append(inputLine);
        }


        JSONParser parser = new JSONParser();
        JSONObject responseObject = (JSONObject)
parser.parse(response.toString());
        JSONObject location = (JSONObject)
responseObject.get("location");
        JSONObject nearbyWeatherStations = (JSONObject)
location.get("nearby_weather_stations");
        JSONObject pws = (JSONObject)
nearbyWeatherStations.get("pws");
        String stationString = pws.get("station").toString();
        Object obj = parser.parse(stationString);
        org.json.simple.JSONArray jsonObject =
(org.json.simple.JSONArray) obj;

        for (int l=0; l<jsonObject.size(); l++) {
            JSONObject closestStation = (JSONObject)
jsonObject.get(l);
            id = closestStation.get("id").toString();
            //System.out.println(l);
```

```
            System.out.println("αναγνωριστικό μετεωρολογικού σταθμού =
" + id);
            //System.out.println(startYear);
            //System.out.println(endYear);

            if (Objects.equals(startYear, endYear)) {

            for (int i=startMonth; i<endMonth + 1; i++) {

                URL url2 = new URL
("http://www.wunderground.com/weatherstation/WXDailyHistory.asp?ID=" +
id + "&graphspan=month&month=" + i + "&day=1&year=" + startYear +
"&format=1");
                HttpURLConnection connection2 = (HttpURLConnection)
url2.openConnection();
                connection2.setRequestMethod("GET");
                connection2.setRequestProperty("Content-Length","0");
             int responseCode2 = connection.getResponseCode();
             //System.out.println("Sending 'GET' request to URL : " +
url2);
             //System.out.println("Response Code : " + responseCode2
+"\n");
             BufferedReader in2 = new BufferedReader(new
InputStreamReader(connection2.getInputStream()));
             String inputLine2;
             StringBuffer response2 = new StringBuffer();
             while ((inputLine2 = in2.readLine()) != null) {
                 response2.append(inputLine2);
                }

                String[] stringValues =
response2.toString().split("<br>");
                System.out.println("ημέρες μήνα = " +
(stringValues.length-1));
                if (i==1||i==3||i==5||i==7||i==8||i==10||i==12) {if
(stringValues.length-1<31) {completePWS = false; break;} else
{completePWS = true;}}
                if (i==4||i==6||i==9||i==11) {if (stringValues.length-
1<30) {completePWS = false; break;} else {completePWS = true;}}
                if(i==2&&isStartLeapYear.equals("yes")) {if
(stringValues.length-1<29) {completePWS = false; break;} else
{completePWS = true;}}
                if(i==2&&isStartLeapYear.equals("no")) {if
(stringValues.length-1<28) {completePWS = false; break;} else
{completePWS = true;}}
                for (int j=1; j<stringValues.length; j++) {
                    for (int k=0; k<24; k++) {

temperatures.add(Double.parseDouble(stringValues[j].split(",")[2]));
                    }
                }
            }

            if (completePWS) {
                //System.out.println("PWS=" + l);
                break;} else {temperatures.clear();}
            //System.out.println("\n");

        }
        else {
                for (int i=startMonth; i<13; i++) {
```

```
                URL url2 = new URL
("http://www.wunderground.com/weatherstation/WXDailyHistory.asp?ID=" +
id + "&graphspan=month&month=" + i + "&day=1&year=" + startYear +
"&format=1");
                HttpURLConnection connection2 = (HttpURLConnection)
url2.openConnection();
                connection2.setRequestMethod("GET");
                connection2.setRequestProperty("Content-Length","0");
            int responseCode2 = connection.getResponseCode();
            //System.out.println("Sending 'GET' request to URL : " +
url2);
            //System.out.println("Response Code : " + responseCode2
+"\n");
            BufferedReader in2 = new BufferedReader(new
InputStreamReader(connection2.getInputStream()));
            String inputLine2;
            StringBuffer response2 = new StringBuffer();
            while ((inputLine2 = in2.readLine()) != null) {
                response2.append(inputLine2);
                }

                String[] stringValues =
response2.toString().split("<br>");
                System.out.println("ημέρες μήνα = " +
(stringValues.length-1));
                if (i==1||i==3||i==5||i==7||i==8||i==10||i==12) {if
(stringValues.length-1<31) {completePWS = false; break;} else
{completePWS = true;}}
                if (i==4||i==6||i==9||i==11) {if (stringValues.length-
1<30) {completePWS = false; break;} else {completePWS = true;}}
                if(i==2&&isStartLeapYear.equals("yes")) {if
(stringValues.length-1<29) {completePWS = false; break;} else
{completePWS = true;}}
                if(i==2&&isStartLeapYear.equals("no")) {if
(stringValues.length-1<28) {completePWS = false; break;} else
{completePWS = true;}}
                for (int j=1; j<stringValues.length; j++) {
                    for (int k=0; k<24; k++) {

temperatures.add(Double.parseDouble(stringValues[j].split(",")[2]));
                    }
                }
            }

                for (int i=1; i<endMonth + 1; i++) {

                if (!completePWS) {break;}

                URL url2 = new URL
("http://www.wunderground.com/weatherstation/WXDailyHistory.asp?ID=" +
id + "&graphspan=month&month=" + i + "&day=1&year=" + endYear +
"&format=1");
                HttpURLConnection connection2 = (HttpURLConnection)
url2.openConnection();
                connection2.setRequestMethod("GET");
                connection2.setRequestProperty("Content-Length","0");
            int responseCode2 = connection.getResponseCode();
            //System.out.println("Sending 'GET' request to URL : " +
url2);
            //System.out.println("Response Code : " + responseCode2
+"\n");
```

```
            BufferedReader in2 = new BufferedReader(new
InputStreamReader(connection2.getInputStream()));
            String inputLine2;
            StringBuffer response2 = new StringBuffer();
            while ((inputLine2 = in2.readLine()) != null) {
                response2.append(inputLine2);
               }

               String[] stringValues =
response2.toString().split("<br>");
               System.out.println("ημέρες μήνα = " +
(stringValues.length-1));
               if (i==1||i==3||i==5||i==7||i==8||i==10||i==12) {if
(stringValues.length-1<31) {completePWS = false; break;} else
{completePWS = true;}}
               if (i==4||i==6||i==9||i==11) {if (stringValues.length-
1<30) {completePWS = false; break;} else {completePWS = true;}}
               if(i==2&&isEndLeapYear.equals("yes")) {if
(stringValues.length-1<29) {completePWS = false; break;} else
{completePWS = true;}}
               if(i==2&&isEndLeapYear.equals("no")) {if
(stringValues.length-1<28) {completePWS = false; break;} else
{completePWS = true;}}
               for (int j=1; j<stringValues.length; j++) {
                   for (int k=0; k<24; k++) {

temperatures.add(Double.parseDouble(stringValues[j].split(",")[2]));
                   }
               }
           }
             if (completePWS) {System.out.println("PWS=" + l);
break;} else {temperatures.clear();}
           System.out.println("\n");
       }

       }
           if (!completePWS) {System.out.println("There are no
complete temperatures values for these coordinates");}
           //System.out.println(temperatures.size());
           //System.out.println("\n");
       return temperatures;
   }

}
```

Εικόνα 26: Κώδικας Java για την εύρεση των μέσων ημερήσιων θερμοκρασιών απο τον διαικτυακό τόπο wunderground

5.1.1. ΕΥΡΕΣΗ ΠΛΗΣΙΕΣΤΕΡΩΝ ΜΕΤΕΩΡΟΛΟΓΙΚΩΝ ΣΤΑΘΜΩΝ (PWS)

Το παρακάτω κομμάτι κώδικα επικοινωνεί, μέσω του πρωτοκόλλου HTTP, με τον διαδικτυακό τόπο http://www.wunderground.com και, με βάσει τις γεωγραφικές συντεταγμένες που εισάγει ο χρήστης στο αρχείο παραμετροποίησης, αντλεί έναν πίνακα με όλους τους πλησιέστερους διαθέσιμους μετεωρολογικούς σταθμούς. Οι σταθμοί αυτοί ξεχωρίζονται μεταξύ τους μέσω ενός αναγνωριστικού (id).

```
URL url = new URL
("http://api.wunderground.com/api/49a120bc6584c08a/geolookup/q/" +
latitude + "," + longitude + ".json");
        HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
        connection.setRequestMethod("GET");
        connection.setRequestProperty("Content-Length","0");
      int responseCode = connection.getResponseCode();
      //System.out.println("Sending 'GET' request to URL : " + url
+"\n");
      //System.out.println("Response Code : " + responseCode +"\n");
in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
inputLine;
response = new StringBuffer();
      while ((inputLine = in.readLine()) != null) {
         response.append(inputLine);
        }


        JSONParser parser = new JSONParser();
        JSONObject responseObject = (JSONObject)
parser.parse(response.toString());
        JSONObject location = (JSONObject)
responseObject.get("location");
        JSONObject nearbyWeatherStations = (JSONObject)
location.get("nearby_weather_stations");
        JSONObject pws = (JSONObject)
nearbyWeatherStations.get("pws");
        String stationString = pws.get("station").toString();
        Object obj = parser.parse(stationString);
        org.json.simple.JSONArray             jsonObject             =
(org.json.simple.JSONArray) obj;
```

Εικόνα 27: Επιμέρους κώδικας για εύρεση πλησιέστερων μετεωρολογικών σταθμών (pws)

5.1.2. ΑΝΤΛΗΣΗ ΜΕΣΩΝ ΗΜΕΡΗΣΙΩΝ ΘΕΡΜΟΚΡΑΣΙΩΝ

Αφού το πρόγραμμα λάβει τους διαθέσιμους σταθμούς (5.1.1), ξεκινώντας από τον πλησιέστερο, ξαναεπικοινωνεί, μέσω του πρωτοκόλλου HTTP, με τον διαδικτυακό τόπο http://www.wunderground.com, και παίρνει πίσω όλες τις θερμοκρασίες (υψηλότερη, μέση, χαμηλότερη) για όλους τους μήνες του χρονικού διαστήματος του ενδιαφέροντος του χρήστη. Στη συνέχεια ο κώδικας ελέγχει ότι όλες οι ζητούμενες θερμοκρασίες είναι διαθέσιμες από τον εν λόγω σταθμό, και αν δεν είναι τις αναζητά στον αμέσως πλησιέστερο, αλλιώς σταματάει την αναζήτηση σταθμού και απομονώνει τις μέσες ημερήσιες θερμοκρασίες (καθεμία την αντιγράφει άλλες 23 φορές για να συμπληρωθούν 24 ώρες) και τις επιστρέφει. Ο έλεγχος αυτός ήταν απολύτως απαραίτητος καθώς κάποιοι σταθμοί διέθεταν ελλειπή δεδομένα. Για παράδειγμα, ο σταθμός με αναγνωριστικό IATTICAU4, που είναι ο πλησιέστερος στην

ευρύτερη περιοχή του Πειραιά, δίνει την ακόλουθη απάντηση για το μήνα Αύγουστο του 2012:

(http://www.wunderground.com/weatherstation/WXDailyHistory.asp?ID=IATTICAU4&graphspan=month&month=8&day=1&year=2012&format=1

Date,TemperatureHighC,TemperatureAvgC,TemperatureLowC,Dewp
2012-8-1,35.1,31.3,28.1,22.1,19.8,16.3,63,51,39,967,964,16,3,31,0.00
2012-8-2,35.3,30.9,26.5,25.8,22.3,18.9,80,62,45,968,967,10,1,19,0.00
2012-8-3,37.6,28.7,26.4,23.7,21.1,17.8,83,66,36,968,968,5,0,13,0.00
2012-8-6,37.6,37.6,37.6,20.4,20.4,20.4,37,37,37,968,968,0,0,0,0.00

Εικόνα 28: Παράδειγμα έλλειψης διαθέσιμων θερμοκρασιών

δηλαδή θερμοκρασίες μόνο για τέσσερις από τις συνολικά τριάντα μία ημέρες του μήνα.

Το κομμάτι του κώδικα που αφορά στην λειτουργία της υποπαραγράφου 5.1.2 είναι:

```
for (int l=0; l<jsonObject.size(); l++) {
            JSONObject closestStation = (JSONObject)
jsonObject.get(l);
            id = closestStation.get("id").toString();
            //System.out.println(l);
            System.out.println("αναγνωριστικό μετεωρολογικού σταθμού =
" + id);
            //System.out.println(startYear);
            //System.out.println(endYear);

            if (Objects.equals(startYear, endYear)) {

            for (int i=startMonth; i<endMonth + 1; i++) {

                URL url2 = new URL
("http://www.wunderground.com/weatherstation/WXDailyHistory.asp?ID=" +
id + "&graphspan=month&month=" + i + "&day=1&year=" + startYear +
"&format=1");
                HttpURLConnection connection2 = (HttpURLConnection)
url2.openConnection();
                connection2.setRequestMethod("GET");
                connection2.setRequestProperty("Content-Length","0");
             int responseCode2 = connection.getResponseCode();
             System.out.println("Sending 'GET' request to URL : " +
url2);
             //System.out.println("Response Code : " + responseCode2
+"\n");
             BufferedReader in2 = new BufferedReader(new
InputStreamReader(connection2.getInputStream()));
             String inputLine2;
             StringBuffer response2 = new StringBuffer();
```

```
            while ((inputLine2 = in2.readLine()) != null) {
                response2.append(inputLine2);
                }

                String[] stringValues =
response2.toString().split("<br>");
                System.out.println("μέρες μήνα = " +
(stringValues.length-1));
                if (i==1||i==3||i==5||i==7||i==8||i==10||i==12) {if
(stringValues.length-1<31) {completePWS = false; break;} else
{completePWS = true;}}
                if (i==4||i==6||i==9||i==11) {if (stringValues.length-
1<30) {completePWS = false; break;} else {completePWS = true;}}
                if(i==2&&isStartLeapYear.equals("yes")) {if
(stringValues.length-1<29) {completePWS = false; break;} else
{completePWS = true;}}
                if(i==2&&isStartLeapYear.equals("no")) {if
(stringValues.length-1<28) {completePWS = false; break;} else
{completePWS = true;}}
                for (int j=1; j<stringValues.length; j++) {
                    for (int k=0; k<24; k++) {

temperatures.add(Double.parseDouble(stringValues[j].split(",")[2]));
                    }
                }
            }

            if (completePWS) {
                //System.out.println("PWS=" + l);
                break;} else {temperatures.clear();}
            //System.out.println("\n");

        }
        else {
                for (int i=startMonth; i<13; i++) {

                URL url2 = new URL
("http://www.wunderground.com/weatherstation/WXDailyHistory.asp?ID=" +
id + "&graphspan=month&month=" + i + "&day=1&year=" + startYear +
"&format=1");
                HttpURLConnection connection2 = (HttpURLConnection)
url2.openConnection();
                connection2.setRequestMethod("GET");
                connection2.setRequestProperty("Content-Length","0");
             int responseCode2 = connection.getResponseCode();
             System.out.println("Sending 'GET' request to URL : " +
url2);
             //System.out.println("Response Code : " + responseCode2
+"\n");
             BufferedReader in2 = new BufferedReader(new
InputStreamReader(connection2.getInputStream()));
             String inputLine2;
             StringBuffer response2 = new StringBuffer();
             while ((inputLine2 = in2.readLine()) != null) {
                 response2.append(inputLine2);
                }

                String[] stringValues =
response2.toString().split("<br>");
                System.out.println("μέρες μήνα = " +
(stringValues.length-1));
```

```
                if (i==1||i==3||i==5||i==7||i==8||i==10||i==12) {if
(stringValues.length-1<31) {completePWS = false; break;} else
{completePWS = true;}}
                if (i==4||i==6||i==9||i==11) {if (stringValues.length-
1<30) {completePWS = false; break;} else {completePWS = true;}}
                if(i==2&&isStartLeapYear.equals("yes")) {if
(stringValues.length-1<29) {completePWS = false; break;} else
{completePWS = true;}}
                if(i==2&&isStartLeapYear.equals("no")) {if
(stringValues.length-1<28) {completePWS = false; break;} else
{completePWS = true;}}
                for (int j=1; j<stringValues.length; j++) {
                    for (int k=0; k<24; k++) {

temperatures.add(Double.parseDouble(stringValues[j].split(",")[2]));
                    }
                }
            }

                for (int i=1; i<endMonth + 1; i++) {

                if (!completePWS) {break;}

                URL url2 = new URL
("http://www.wunderground.com/weatherstation/WXDailyHistory.asp?ID=" +
id + "&graphspan=month&month=" + i + "&day=1&year=" + endYear +
"&format=1");
                HttpURLConnection connection2 = (HttpURLConnection)
url2.openConnection();
                connection2.setRequestMethod("GET");
                connection2.setRequestProperty("Content-Length","0");
            int responseCode2 = connection.getResponseCode();
            System.out.println("Sending 'GET' request to URL : " +
url2);
            //System.out.println("Response Code : " + responseCode2
+"\n");
            BufferedReader in2 = new BufferedReader(new
InputStreamReader(connection2.getInputStream()));
            String inputLine2;
            StringBuffer response2 = new StringBuffer();
            while ((inputLine2 = in2.readLine()) != null) {
                response2.append(inputLine2);
                }

                String[] stringValues =
response2.toString().split("<br>");
                System.out.println("μέρες μήνα = " +
(stringValues.length-1));
                if (i==1||i==3||i==5||i==7||i==8||i==10||i==12) {if
(stringValues.length-1<31) {completePWS = false; break;} else
{completePWS = true;}}
                if (i==4||i==6||i==9||i==11) {if (stringValues.length-
1<30) {completePWS = false; break;} else {completePWS = true;}}
                if(i==2&&isEndLeapYear.equals("yes")) {if
(stringValues.length-1<29) {completePWS = false; break;} else
{completePWS = true;}}
                if(i==2&&isEndLeapYear.equals("no")) {if
(stringValues.length-1<28) {completePWS = false; break;} else
{completePWS = true;}}
                for (int j=1; j<stringValues.length; j++) {
                    for (int k=0; k<24; k++) {
```

```
temperatures.add(Double.parseDouble(stringValues[j].split(",")[2]));
                    }
                }
            }
              if (completePWS) {System.out.println("PWS=" + l);
break;} else {temperatures.clear();}
            System.out.println("\n");
        }

        }
            if (!completePWS) {System.out.println("There are no
complete temperatures values for these coordinates");}
            //System.out.println(temperatures.size());
            //System.out.println("\n");

        return temperatures
```

Εικόνα 29: Επιμέρους κώδικας για άντληση μέσων ημερήσιων θερμοκρασιών

5.1.3. ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΗΣ – ΕΚΤΕΛΕΣΗ ΚΩΔΙΚΑ

Για να εκτελέσουμε τον κώδικα της παραγράφου 5.1 φτιάξαμε την δοκιμαστική Java κλάση " TestWundergroundConnection":

```
package metaptyxiakh;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;

public class TestWundergroundConnection {

    public static void main(String[] args) throws
FileNotFoundException, IOException, ParseException {

        ArrayList <Double> temps = new ArrayList();

        String pathToConfFile
="C:\\Users\\user\\Desktop\\configuration.json";
        FileReader configurationFile = new FileReader(pathToConfFile);
        JSONParser parser = new JSONParser();
        JSONObject configurationObject = (JSONObject)
parser.parse(configurationFile);
        JSONObject start = (JSONObject)
configurationObject.get("Start_Time");
        JSONObject end = (JSONObject)
configurationObject.get("End_Time");

        WundergroundConnection wc = new WundergroundConnection();
```

```
        temps = wc.getTemperatures(Double.parseDouble((String)
configurationObject.get("Latitude")), Double.parseDouble((String)
configurationObject.get("Longitude")), Integer.parseInt((String)
start.get("Month")), Integer.parseInt((String) start.get("Year")),
(String) start.get("Leap_Year"), Integer.parseInt((String)
end.get("Month")), Integer.parseInt((String) end.get("Year")),
(String) end.get("Leap_Year"));

        System.out.println("πλήθος θερμοκρασιών = " + temps.size());

        for (int i=0; i<temps.size(); i++) {

            if (i % 24 == 0) {
                System.out.println(temps.get(i));
            }
        }

    }
}
```

Εικόνα 30: Java κλάση "TestWundergroundConnection"

Εάν για παράδειγμα θέλουμε να λάβουμε όλες τις μέσες θερμοκρασίες του Φεβρουαρίου 2012 για την ευρύτερη περιοχή του Πειραιά (γεωγραφικές συντεταγμένες 37,951575 23,66700232) το αρχείο παραμετροποίησης πρέπει να συμπληρωθεί, σύμφωνα και με όσα περιγράφονται στην αρχή της παραγράφου 5.1, ως εξής:

```
{
    "Latitude":"37.951575",
    "Longitude":"23.66700232",
    "Start_Time":{
        "Month":"2",
        "Year":"2012",
        "Leap_Year":"yes"
    },
    "End_Time":{
        "Month":"2",
        "Year":"2012",
        "Leap_Year":"yes"
    },
    "Data_Timestamp_Format":"yyyy-MM-dd HH:mm:ss",
    "Number_of_Values_per_Hour":"4"
}
```

Εικόνα 31: Παράδειγμα αρχείου παραμετροποίησης III

Τρέχοντας την δοκιμαστική κλάση λαμβάνουμε τα ακόλουθα αποτελέσματα:

```
αναγνωριστικό μετεωρολογικού σταθμού = IATTICAU4
ημέρες μήνα = 29
πλήθος θερμοκρασιών = 696
3.8
6.6
12.0
14.8
15.2
14.0
12.0
6.7
4.8
6.7
10.6
10.7
12.3
10.7
9.5
8.8
6.4
9.4
11.3
10.9
12.7
11.1
12.1
13.5
14.8
14.2
9.7
5.6
6.8
```

Εικόνα 32: Αποτελέσματα Java κλάσης "TestWundergroundConnection"

Το πλήθος των θερμοκρασιών είναι (29*24 = 696), αλλά προφανώς τυπώνουμε μόνο τις 29 (όσες και οι ημέρες), αφού οι υπόλοιπες απλά επαναλαμβάνονται.

Για να επαληθεύσουμε τον κώδικα αντλούμε τις ίδιες θερμοκρασίες (Φεβρουάριος 2012) κατ' ευθείαν από έναν browser μέσω της διεύθυνσης:

http://www.wunderground.com/weatherstation/WXDailyHistory.asp?ID=IATTICAU4&graphspan=month&month=2&day=1&year=2012&format=1

```
Date,TemperatureHighC,TemperatureAvgC,TemperatureLowC,Dewpoi
2012-2-1,6.7,3.8,1.7,1.7,-0.3,-1.7,83,75,63,996,991,16,3,34,0.00
2012-2-2,11.5,6.6,3.7,9.8,5.3,0.6,99,92,76,991,986,10,1,18,0.13
2012-2-3,18.3,12.0,8.1,14.0,10.4,7.1,98,91,75,993,990,8,0,13,0.00
2012-2-4,20.8,14.8,9.9,17.0,13.6,9.8,99,92,78,993,989,5,1,14,0.03
2012-2-5,18.6,15.2,13.5,15.9,14.3,13.1,99,94,79,990,985,10,2,18,0.00
2012-2-6,16.9,14.0,10.7,14.3,12.4,9.7,99,91,71,986,961,27,3,55,1.50
2012-2-7,16.0,12.0,7.6,12.6,10.8,6.6,99,93,78,975,961,11,2,21,0.28
2012-2-8,10.7,6.7,5.2,6.7,3.4,2.4,93,80,68,988,975,16,3,31,0.00
2012-2-9,6.3,4.8,3.8,3.8,2.7,2.0,92,86,79,990,988,13,2,32,0.00
2012-2-10,8.8,6.7,4.8,5.2,4.0,2.8,88,83,74,995,990,13,2,24,0.00
2012-2-11,18.3,10.6,7.3,13.1,8.1,4.4,92,85,70,994,991,5,0,13,0.00
2012-2-12,13.7,10.7,9.2,13.6,10.3,9.1,99,98,88,991,985,5,0,13,3.68
2012-2-13,19.8,12.3,9.3,15.7,10.3,7.0,99,88,63,986,980,5,0,14,0.00
2012-2-14,16.5,10.7,7.4,11.2,8.1,5.7,95,85,64,982,978,5,0,18,0.10
2012-2-15,18.1,9.5,6.0,10.3,5.7,3.5,93,78,58,982,978,10,1,34,0.03
2012-2-16,12.3,8.8,6.1,8.0,5.3,1.5,91,79,67,978,973,16,1,34,0.10
2012-2-17,9.9,6.4,4.0,2.3,-0.4,-1.6,76,62,49,989,978,16,2,37,0.00
2012-2-18,16.2,9.4,2.8,6.4,1.1,-3.2,71,57,38,992,988,5,0,14,0.00
2012-2-19,18.6,11.3,5.9,8.7,5.8,1.8,84,70,48,998,992,3,0,10,0.00
2012-2-20,15.1,10.9,7.4,11.7,8.9,6.3,96,88,77,1000,998,5,0,13,0.00
2012-2-21,19.1,12.7,8.0,12.6,8.9,7.0,99,79,56,1000,996,13,1,23,0.00
2012-2-22,11.8,11.1,10.3,10.8,9.8,8.7,99,92,82,996,992,10,2,21,0.56
2012-2-23,16.1,12.1,10.2,10.7,8.9,7.1,98,82,68,993,988,10,1,80,0.03
2012-2-24,20.9,13.5,7.0,14.1,9.3,5.8,93,76,55,989,985,3,0,10,0.00
2012-2-25,21.6,14.8,10.2,15.1,10.9,6.8,91,78,63,987,983,3,0,8,0.00
2012-2-26,18.9,14.2,10.5,15.0,12.3,8.6,99,88,69,983,973,2,0,6,0.00
2012-2-27,12.9,9.7,7.0,12.3,8.0,4.5,97,89,84,976,971,21,4,60,0.13
2012-2-28,8.9,5.6,2.5,4.7,1.3,-3.7,99,75,53,985,976,27,5,55,0.18
2012-2-29,12.7,6.8,2.1,6.2,1.3,-4.9,99,69,49,985,972,13,1,37,0.15
```

Πράγματι, οι μέσες θερμοκρασίες (“TemperatureAvgC”) είναι οι ίδιες με τις αντίστοιχες που μας έδωσε το πρόγραμμα (Εικόνα 32).

5.2. ΚΩΔΙΚΑΣ ΜΕΘΟΔΟΥ ΙΙ

```
package metaptyxiakh;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import static java.lang.Math.sqrt;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.TimeZone;
```

```
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;


public class SESWithSeasonality168Temps {

    public static void main(String[] args) throws
FileNotFoundException, IOException, ParseException,
UnsupportedEncodingException, org.json.simple.parser.ParseException {

        ArrayList <Double> data = new ArrayList();
        ArrayList <String> timestamps = new ArrayList();
        ArrayList <Double> indexes = new ArrayList();
        ArrayList <Double> forecasts = new ArrayList();
        ArrayList <Double> validation = new ArrayList();
        ArrayList <Double> temps = new ArrayList();
        Double[] coef = new Double[2];
        ArrayList <Double> dataTest = new ArrayList();
        ArrayList <Double> forecastsTest = new ArrayList();
        Integer k = 0;

        String csvFile = "C:\\Users\\user\\Desktop\\data\\M135.csv";

pathToConfFile ="C:\\Users\\user\\Desktop\\configuration.json";
        FileReader configurationFile = new FileReader(pathToConfFile);
        JSONParser parser = new JSONParser();
        JSONObject configurationObject = (JSONObject)
parser.parse(configurationFile);
        JSONObject start = (JSONObject)
configurationObject.get("Start_Time");
        JSONObject end = (JSONObject)
configurationObject.get("End_Time");
        Integer startMonth =
Integer.parseInt((String)start.get("Month"));
        Integer endMonth = Integer.parseInt((String)end.get("Month"));
        Integer startYear =
Integer.parseInt((String)start.get("Year"));

        if (startYear == 2012) {

br = null;
        br = new BufferedReader(new FileReader(csvFile));
        String line = br.readLine();

        while ((line = br.readLine()) != null) {
            if((line.split(",")[0].substring(4,5).equals("2") &&
startMonth<= Integer.parseInt(line.split(",")[0].substring(6,8)) &&
Integer.parseInt(line.split(",")[0].substring(6,8))<13) ||
(line.split(",")[0].substring(4,5).equals("3") && 1<=
Integer.parseInt(line.split(",")[0].substring(6,8)) &&
Integer.parseInt(line.split(",")[0].substring(6,8))<(endMonth + 1))) {

data.add(Double.parseDouble(line.split(",")[1].replace("\"", "")));
                timestamps.add(line.split(",")[0].replace("\"", ""));
            }
        }
        }
        else {

            BufferedReader br = null;
            br = new BufferedReader(new FileReader(csvFile));
            String line = br.readLine();
```

```
        while ((line = br.readLine()) != null) {
            if((line.split(",")[0].substring(4,5).equals("2") && 13<=
Integer.parseInt(line.split(",")[0].substring(6,8)) &&
Integer.parseInt(line.split(",")[0].substring(6,8))<13) ||
(line.split(",")[0].substring(4,5).equals("3") && startMonth<=
Integer.parseInt(line.split(",")[0].substring(6,8)) &&
Integer.parseInt(line.split(",")[0].substring(6,8))<(endMonth + 1))) {

data.add(Double.parseDouble(line.split(",")[1].replace("\"", "")));
                timestamps.add(line.split(",")[0].replace("\"", ""));
            }
        }

        }

        SimpleDateFormat sdf = new SimpleDateFormat((String)
configurationObject.get("Data_Timestamp_Format"));
        sdf.setTimeZone(TimeZone.getTimeZone("GMT"));
        MissingValuesAndHourlyAggregation aggr = new
MissingValuesAndHourlyAggregation();
        data = aggr.mvaha(data, timestamps, sdf,
Integer.parseInt((String)
configurationObject.get("Number_of_Values_per_Hour")));


        if
(endMonth==1||endMonth==3||endMonth==5||endMonth==7||endMonth==8||endM
onth==10||endMonth==12) {k=24;}
        if (endMonth==4||endMonth==6||endMonth==9||endMonth==11)
{k=23;}
        if (endMonth==2 &&
end.get("Leap_Year").toString().equals("no")) {k=21;}
        if (endMonth==2 &&
end.get("Leap_Year").toString().equals("yes")) {k=22;}

        Integer n = data.size()-k*24;

        for (int i=0; i<n; i++) {
            validation.add(data.get(i));
        }



        WundergroundConnection wc = new WundergroundConnection();
        temps = wc.getTemperatures(Double.parseDouble((String)
configurationObject.get("Latitude")), Double.parseDouble((String)
configurationObject.get("Longitude")), Integer.parseInt((String)
start.get("Month")), Integer.parseInt((String) start.get("Year")),
(String) start.get("Leap_Year"), Integer.parseInt((String)
end.get("Month")), Integer.parseInt((String) end.get("Year")),
(String) end.get("Leap_Year"));



        for (int i=0; i<n; i++) {
            validation.set(i, validation.get(i)/temps.get(i));
        }

        SeasonalDecomposition dec = new SeasonalDecomposition();
        indexes = dec.sd(validation, 168);
```

```
        for (int i=0; i<validation.size(); i++) {
            validation.set(i,
100*validation.get(i)/indexes.get(i%168));
        }

        SimpleExponentialSmoothing test = new
SimpleExponentialSmoothing();
        LinearRegression lr = new LinearRegression();
        coef = lr.lrl(validation);
        Errors err = new Errors();
        Double minMSE = 1000000.0;
        Double amin = 0.0;
        //Double forecastFinal = 0.0;

        for (Integer i=0; i<10001; i++) {
            forecasts = test.ses(validation, i.doubleValue()/10000,
coef[0], 1);
            if (err.mse(validation, forecasts)<minMSE) {
                minMSE = err.mse(validation, forecasts); amin =
i.doubleValue()/10000;
                //forecastFinal = forecasts.get(forecasts.size()-1);
            }
        }

        System.out.println("\n");
        //System.out.println(forecastFinal);
        //System.out.println(sqrt(minMSE));
        System.out.println("βέλτιστο a = " + amin +"\n");

        forecasts.clear();
        forecasts = test.ses(validation, amin, coef[0], 2208);

        for (int i=n; i<n+168; i++) {
            dataTest.add(data.get(i));

forecastsTest.add(temps.get(i)*forecasts.get(i)*indexes.get(i%168)/100
);
        }
        if (dataTest.size() == forecastsTest.size() && dataTest.size()
== 168) {
            System.out.println("mape = " + err.mape(dataTest,
forecastsTest));
            System.out.println("mse = " + sqrt(err.mse(dataTest,
forecastsTest)) + "\n");
            //System.out.println(sqrt(exam.mse(forecastsTest,
dataTest)));
        }
    }

}
```

Εικόνα 33: Κώδικας Java για την μέθοδο ΙΙ (κλάση "SESWithSeasonality168Temps")

5.3. ΑΠΟΤΕΛΕΣΜΑΤΑ

ΠΛΗΘΟΣ ΙΣΤΟΡΙΚΩΝ ΜΗΝΩΝ (+ 1η ΕΒΔΟΜΑΔΑ)	2η ΕΒΔΟΜΑΔΑ 2013	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
12		30,31	34,10	22,07	8,34	15,63	15,54	14,18	13,36	7,16	46,87	8,25	31,82
11		31,36	32,23	22,17	8,74	15,36	15,63	14,29	13,16	7,32	47,07	8,41	32,45
10		31,52	32,08	23,95	8,98	14,95	15,70	13,98	13,03	7,43	47,26	9,06	32,87
9		31,98	30,83	26,19	9,16	14,09	15,82	13,86	12,86	7,39	46,55	9,37	33,15
8		32,66	29,91	26,81	9,37	13,61	15,69	13,96	12,63	6,10	44,69	9,46	32,46
7		32,73	29,53	28,28	10,01	13,06	15,60	13,93	12,61	5,73	43,11	9,93	31,58
6		33,11	28,50	28,82	10,71	13,04	16,46	13,95	12,95	5,16	45,64	9,84	30,40
5		33,42	28,11	30,04	11,75	12,86	13,64	14,78	14,08	5,40	48,78	10,39	30,23
4		34,14	27,15	38,47	12,57	15,60	9,55	15,63	12,19	7,08	50,76	10,66	31,87
3		35,26	12,74	40,03	10,93	15,30	8,37	13,77	10,75	10,06	50,56	10,44	32,77
2		30,03	12,49	39,39	11,39	20,29	7,20	13,21	9,83	13,30	49,73	12,30	32,07
1		34,61	12,44	28,44	13,64	20,23	11,95	10,61	12,15	9,31	39,28	13,03	28,93
MIN MAPE		30,03	12,44	22,07	8,34	12,86	7,20	10,61	9,83	5,16	39,28	8,25	28,93
AVERAGE MAPE		32,59	25,84	29,56	10,47	15,34	13,43	13,85	12,47	7,62	46,69	10,10	31,72

Εικόνα 34: Συγκεντρωτικά αποτελέσματα της μεθόδου II

5.4. ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΗΣ – ΕΚΤΕΛΕΣΗ ΚΩΔΙΚΑ

Προκειμένου κανείς να εξάγει κάποιο από τα αποτελέσματα της παραγράφου 5.3, πρέπει πρώτα να συμπληρώσει κατάλληλα το αρχείο παραμετροποίησης, με την ίδια ακριβώς διαδικασία που περιγράφεται αναλυτικά στην παράγραφο 4.4. Συνεπώς, για να εξάγουμε το mape που αντιστοιχεί στην πρόβλεψη της ωριαίας ενεργειακής κατανάλωσης για την δεύτερη εβδομάδα του Αυγούστου 2013 (9,83%) με πλήθος ιστορικών μηνών 2 (Ιούνιος 2013 – Ιούλιος 2013), το αρχείο πρέπει να έχει την ακόλουθη μορφή:

```
{
    "Latitude":"37.951575",
    "Longitude":"23.66700232",
    "Start_Time":{
        "Month":"6",
        "Year":"2013",
        "Leap_Year":"no"
    },
    "End_Time":{
        "Month":"8",
        "Year":"2013",
        "Leap_Year":"no"
    },
    "Data_Timestamp_Format":"yyyy-MM-dd HH:mm:ss",
    "Number_of_Values_per_Hour":"4"
}
```

Εικόνα 35: Παράδειγμα αρχείου παραμετροποίησης III

Μετά την ορθή εισαγωγή των παραμέτρων, τρέχουμε τον κώδικα της παραγράφου 5.2 και λαμβάνουμε το εξής αποτέλεσμα:

```
αναγνωριστικό μετεωρολογικού σταθμού = IATTICAU4
ημέρες μήνα = 30
ημέρες μήνα = 31
ημέρες μήνα = 29
αναγνωριστικό μετεωρολογικού σταθμού = IMOSCHAT2
ημέρες μήνα = 0
αναγνωριστικό μετεωρολογικού σταθμού = IU0391U019
ημέρες μήνα = 30
ημέρες μήνα = 31
ημέρες μήνα = 31


βέλτιστο a = 0.9709

mape = 9.828764679922404
mse = 26.07949166880497
```

Εικόνα 36: Παράδειγμα εκτέλεσης του κώδικα της μεθόδου ΙΙ

6. ΜΕΘΟΔΟΣ ΙΙΙ: SES ΜΕ 168 ΔΙΑΦΟΡΕΤΙΚΑ ΜΟΝΤΕΛΑ

Η περιγραφή της μεθόδου ΙΙΙ βρίσκεται στην υποπαράγραφο 1.4.3.

6.1. ΚΩΔΙΚΑΣ ΜΕΘΟΔΟΥ ΙΙΙ

```
package metaptyxiakh;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.TimeZone;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;


public class SESWith168DifferentModels {

    public static void main(String[] args) throws
FileNotFoundException, IOException, ParseException,
UnsupportedEncodingException, org.json.simple.parser.ParseException {

        ArrayList <Double> data = new ArrayList();
        ArrayList <String> timestamps = new ArrayList();
        ArrayList <Double> forecasts = new ArrayList();
        ArrayList <Double> validation = new ArrayList();
        ArrayList <Double> mapes = new ArrayList();
        Double[] coef = new Double[2];
        ArrayList <Double> dataTest = new ArrayList();
        ArrayList <Double> forecastsTest = new ArrayList();
        Integer z = 0;

        String csvFile = "C:\\Users\\user\\Desktop\\data\\M135.csv";

        String pathToConfFile
="C:\\Users\\user\\Desktop\\configuration.json";
        FileReader configurationFile = new FileReader(pathToConfFile);
        JSONParser parser = new JSONParser();
        JSONObject configurationObject = (JSONObject)
parser.parse(configurationFile);
        JSONObject start = (JSONObject)
configurationObject.get("Start_Time");
        JSONObject end = (JSONObject)
configurationObject.get("End_Time");
        Integer startMonth =
Integer.parseInt((String)start.get("Month"));
        Integer endMonth = Integer.parseInt((String)end.get("Month"));
```

```
        Integer startYear =
Integer.parseInt((String)start.get("Year"));

      if (startYear == 2012) {

br = null;
        br = new BufferedReader(new FileReader(csvFile));
        String line = br.readLine();

        while ((line = br.readLine()) != null) {
            if((line.split(",")[0].substring(4,5).equals("2") &&
startMonth<= Integer.parseInt(line.split(",")[0].substring(6,8)) &&
Integer.parseInt(line.split(",")[0].substring(6,8))<13) ||
(line.split(",")[0].substring(4,5).equals("3") && 1<=
Integer.parseInt(line.split(",")[0].substring(6,8)) &&
Integer.parseInt(line.split(",")[0].substring(6,8))<(endMonth + 1))) {

data.add(Double.parseDouble(line.split(",")[1].replace("\"", "")));
                timestamps.add(line.split(",")[0].replace("\"", ""));
            }
        }
        }
        else {

            BufferedReader br = null;
            br = new BufferedReader(new FileReader(csvFile));
            String line = br.readLine();

        while ((line = br.readLine()) != null) {
            if((line.split(",")[0].substring(4,5).equals("2") && 13<=
Integer.parseInt(line.split(",")[0].substring(6,8)) &&
Integer.parseInt(line.split(",")[0].substring(6,8))<13) ||
(line.split(",")[0].substring(4,5).equals("3") && startMonth<=
Integer.parseInt(line.split(",")[0].substring(6,8)) &&
Integer.parseInt(line.split(",")[0].substring(6,8))<(endMonth + 1))) {

data.add(Double.parseDouble(line.split(",")[1].replace("\"", "")));
                timestamps.add(line.split(",")[0].replace("\"", ""));
            }
        }

        }


        SimpleDateFormat sdf = new SimpleDateFormat((String)
configurationObject.get("Data_Timestamp_Format"));
        sdf.setTimeZone(TimeZone.getTimeZone("GMT"));
        MissingValuesAndHourlyAggregation examo = new
MissingValuesAndHourlyAggregation();
        data = examo.mvaha(data, timestamps, sdf,
Integer.parseInt((String)
configurationObject.get("Number_of_Values_per_Hour")));


        if
(endMonth==1||endMonth==3||endMonth==5||endMonth==7||endMonth==8||endM
onth==10||endMonth==12) {z=24;}
        if (endMonth==4||endMonth==6||endMonth==9||endMonth==11)
{z=23;}
        if (endMonth==2 &&
end.get("Leap_Year").toString().equals("no")) {z=21;}
```

```
        if (endMonth==2 &&
end.get("Leap_Year").toString().equals("yes")) {z=22;}


        Integer n = data.size()-z*24;


        for (int k=n; k<n+168; k++) {
        //System.out.println(k);
        for (int i=0; i<n; i++) {
            validation.add(data.get(i));
        }

        for (int i=validation.size()-1; i>=0; i--) {
            if (i % 168 != k %168) {
                validation.remove(i);
            }
        }

        SimpleExponentialSmoothing ex = new
SimpleExponentialSmoothing();
        LinearRegression lr = new LinearRegression();
        coef = lr.lrl(validation);
        Errors exam = new Errors();
        Double minMSE = 1000000.0;
        Double amin = 0.0;
        Double forecastFinal = 0.0;

        for (Integer i=0; i<10001; i++) {
            forecasts = ex.ses(validation, i.doubleValue()/10000,
coef[0], 1);
            if (exam.mse(validation, forecasts)<minMSE) {
                minMSE = exam.mse(validation, forecasts); amin =
i.doubleValue()/10000;
                forecastFinal = forecasts.get(forecasts.size()-1);
            }
        }

        //System.out.println(forecastFinal);
        //System.out.println(sqrt(minMSE));
        //System.out.println("βέλτιστο a = " + amin +"\n");

        forecasts.clear();
        forecasts = ex.ses(validation, amin, coef[0], 1);

        for (int i=k; i<k+1; i++) {
            dataTest.add(data.get(i));
            forecastsTest.add(forecastFinal);
        }
        if (dataTest.size() == forecastsTest.size() && dataTest.size()
== 1) {
            mapes.add(exam.mape(dataTest, forecastsTest));
        }
        validation.clear();
        forecasts.clear();
        dataTest.clear();
        forecastsTest.clear();
        }

        Double sum =0.0;
        for (int i=0; i<mapes.size(); i++) {
            sum = sum + mapes.get(i);
```

```
            //System.out.println(mapes.get(i));
        }

        //System.out.println(mapes.size()+"\n");
        System.out.println("mape = " + sum/mapes.size());

    }
```

Εικόνα 37: Κώδικας Java για την μέθοδο III (κλάση "SESWith168DifferentModels")

6.2. ΑΠΟΤΕΛΕΣΜΑΤΑ

ΠΛΗΘΟΣ ΙΣΤΟΡΙΚΩΝ ΜΗΝΩΝ (+ 1η ΕΒΔΟΜΑΔΑ)	2η ΕΒΔΟΜΑΔΑ 2013	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
12		12,39	8,17	7,56	5,55	10,00	8,12	9,36	5,71	6,95	10,33	5,77	8,95
11		12,49	8,30	7,54	5,17	9,17	8,15	9,40	5,54	6,99	11,06	5,80	8,57
10		12,55	8,38	7,61	5,69	9,84	8,25	9,17	5,61	6,38	11,33	5,99	8,40
9		12,67	8,40	7,57	5,37	9,78	8,10	9,23	5,52	6,70	11,07	6,15	8,28
8		12,40	8,40	7,55	5,35	9,79	8,11	10,99	5,33	6,82	10,92	6,23	8,25
7		12,22	8,52	7,52	5,40	9,76	8,81	10,77	5,25	6,97	10,92	6,17	8,46
6		12,46	8,67	7,46	5,82	9,01	9,23	10,51	5,22	6,92	11,02	6,31	8,11
5		12,55	9,30	7,92	6,93	7,85	9,08	10,30	5,03	7,24	11,29	6,64	8,31
4		13,33	9,35	9,22	6,64	7,92	9,55	10,18	5,09	7,08	13,37	6,55	7,75
3		13,50	7,85	8,71	5,65	8,87	9,70	10,11	5,19	7,38	13,11	6,65	7,60
MIN MAPE		12,22	7,85	7,46	5,17	7,85	8,10	9,17	5,03	6,38	10,33	5,77	7,60
AVERAGE MAPE		12,66	8,53	7,87	5,76	9,20	8,71	10,00	5,35	6,94	11,44	6,23	8,27

Εικόνα 38: Συγκεντρωτικά αποτελέσματα της μεθόδου III

6.3. ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΗΣ – ΕΚΤΕΛΕΣΗ ΚΩΔΙΚΑ

Προκειμένου κανείς να εξάγει κάποιο από τα αποτελέσματα της παραγράφου 6.2, πρέπει πρώτα να συμπληρώσει κατάλληλα το αρχείο παραμετροποίησης, με την ίδια ακριβώς διαδικασία που περιγράφεται αναλυτικά στην παράγραφο 4.4. Συνεπώς, για να εξάγουμε το mape που αντιστοιχεί στην πρόβλεψη της ωριαίας ενεργειακής κατανάλωσης για την δεύτερη εβδομάδα του Μαίου 2013 (7,85%) με πλήθος ιστορικών μηνών 5 (Δεκέμβριος 2012 – Απρίλιος 2013), το αρχείο πρέπει να έχει την ακόλουθη μορφή:

```
{
    "Latitude":"37.951575",
    "Longitude":"23.66700232",
    "Start_Time":{
        "Month":"12",
        "Year":"2012",
        "Leap_Year":"no"
    },
    "End_Time":{
        "Month":"5",
        "Year":"2013",
        "Leap_Year":"no"
    },
    "Data_Timestamp_Format":"yyyy-MM-dd HH:mm:ss",
    "Number_of_Values_per_Hour":"4"
}
```

Εικόνα 39: Παράδειγμα αρχείου παραμετροποίησης IV

Μετά την ορθή εισαγωγή των παραμέτρων, τρέχουμε τον κώδικα της παραγράφου 6.1 και λαμβάνουμε το εξής αποτέλεσμα:

```
2013-02-04 07:30:00
2
19.68
19.68


2013-02-21 19:30:00
3
5.76
5.76
5.76


2013-03-31 03:00:00
4
23.439999999999998
23.439999999999998
23.439999999999998
23.439999999999998


2013-04-22 22:00:00
3
7.88
7.88
7.88


mape = 7.846568681769978
```

Εικόνα 40: Παράδειγμα εκτέλεσης του κώδικα της μεθόδου III

6.4. ΒΕΛΤΙΩΣΗ ΜΕΘΟΔΟΥ ΙΙΙ

Η περιγραφή της βελτιωμένης μεθόδου ΙΙΙ βρίσκεται στην υποπαράγραφο 1.4.3.1.

6.4.1. ΚΩΔΙΚΑΣ

```
package metaptyxiakh;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.TimeZone;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;


public class SESWith168DifferentModelsImprovement {

    public static void main(String[] args) throws
FileNotFoundException, IOException, ParseException,
UnsupportedEncodingException, org.json.simple.parser.ParseException {

        ArrayList <Double> data = new ArrayList();
        ArrayList <String> timestamps = new ArrayList();
        ArrayList <Double> forecasts = new ArrayList();
        ArrayList <Double> validation = new ArrayList();
        ArrayList <Double> mapes = new ArrayList();
        Double[] coef = new Double[2];
        ArrayList <Double> dataTest = new ArrayList();
        ArrayList <Double> forecastsTest = new ArrayList();
        Integer z = 0;

        String csvFile = "C:\\Users\\user\\Desktop\\data\\M135.csv";

        String pathToConfFile
="C:\\Users\\user\\Desktop\\configuration.json";
        FileReader configurationFile = new FileReader(pathToConfFile);
        JSONParser parser = new JSONParser();
        JSONObject configurationObject = (JSONObject)
parser.parse(configurationFile);
        JSONObject start = (JSONObject)
configurationObject.get("Start_Time");
        JSONObject end = (JSONObject)
configurationObject.get("End_Time");
        Integer startMonth =
Integer.parseInt((String)start.get("Month"));
        Integer endMonth = Integer.parseInt((String)end.get("Month"));
```

```
        Integer startYear =
Integer.parseInt((String)start.get("Year"));
        Integer endYear = Integer.parseInt((String)end.get("Year"));

      if (startYear == 2012) {

br = null;
        br = new BufferedReader(new FileReader(csvFile));
        String line = br.readLine();

        while ((line = br.readLine()) != null) {
            if((line.split(",")[0].substring(4,5).equals("2") &&
startMonth<= Integer.parseInt(line.split(",")[0].substring(6,8)) &&
Integer.parseInt(line.split(",")[0].substring(6,8))<13) ||
(line.split(",")[0].substring(4,5).equals("3") && 1<=
Integer.parseInt(line.split(",")[0].substring(6,8)) &&
Integer.parseInt(line.split(",")[0].substring(6,8))<(endMonth + 1))) {

data.add(Double.parseDouble(line.split(",")[1].replace("\"", "")));
                timestamps.add(line.split(",")[0].replace("\"", ""));
            }
        }
        }
        else {

            BufferedReader br = null;
            br = new BufferedReader(new FileReader(csvFile));
            String line = br.readLine();

        while ((line = br.readLine()) != null) {
            if((line.split(",")[0].substring(4,5).equals("2") && 13<=
Integer.parseInt(line.split(",")[0].substring(6,8)) &&
Integer.parseInt(line.split(",")[0].substring(6,8))<13) ||
(line.split(",")[0].substring(4,5).equals("3") && startMonth<=
Integer.parseInt(line.split(",")[0].substring(6,8)) &&
Integer.parseInt(line.split(",")[0].substring(6,8))<(endMonth + 1))) {

data.add(Double.parseDouble(line.split(",")[1].replace("\"", "")));
                timestamps.add(line.split(",")[0].replace("\"", ""));
            }
        }

        }


        SimpleDateFormat sdf = new SimpleDateFormat((String)
configurationObject.get("Data_Timestamp_Format"));
        sdf.setTimeZone(TimeZone.getTimeZone("GMT"));
        MissingValuesAndHourlyAggregation examo = new
MissingValuesAndHourlyAggregation();
        data = examo.mvaha(data, timestamps, sdf,
Integer.parseInt((String)
configurationObject.get("Number_of_Values_per_Hour")));


        if
(endMonth==1||endMonth==3||endMonth==5||endMonth==7||endMonth==8||endM
onth==10||endMonth==12) {z=24;}
        if (endMonth==4||endMonth==6||endMonth==9||endMonth==11)
{z=23;}
        if (endMonth==2 &&
end.get("Leap_Year").toString().equals("no")) {z=21;}
```

```
        if (endMonth==2 &&
end.get("Leap_Year").toString().equals("yes")) {z=22;}


        Integer n = data.size()-z*24;


        for (int k=n; k<n+168; k++) {
        //System.out.println(k);
        for (int i=0; i<n; i++) {
            validation.add(data.get(i));
        }

        for (int i=validation.size()-1; i>=0; i--) {
            if (i % 168 != k %168) {
                validation.remove(i);
            }
        }

        if (endYear == 2013 && endMonth == 1) {
            if((k>=n+5 && k<=n+21) || (k>=n+30 && k<=n+45)) {
                validation.remove(validation.size()-1);
                validation.remove(validation.size()-1);
            }
        }

        if (endYear == 2013 && endMonth == 5) {
            if(k>=n+126 && k<=n+140) {
                validation.remove(validation.size()-1);
            }
        }


        SimpleExponentialSmoothing ex = new
SimpleExponentialSmoothing();
        LinearRegression lr = new LinearRegression();
        coef = lr.lrl(validation);
        Errors exam = new Errors();
        Double minMSE = 1000000.0;
        Double amin = 0.0;
        Double forecastFinal = 0.0;

        for (Integer i=0; i<10001; i++) {
            forecasts = ex.ses(validation, i.doubleValue()/10000,
coef[0], 1);
            if (exam.mse(validation, forecasts)<minMSE) {
                minMSE = exam.mse(validation, forecasts); amin =
i.doubleValue()/10000;
                forecastFinal = forecasts.get(forecasts.size()-1);
            }
        }

        //System.out.println(forecastFinal);
        //System.out.println(sqrt(minMSE));
        //System.out.println("βέλτιστο a = " + amin +"\n");

        forecasts.clear();
        forecasts = ex.ses(validation, amin, coef[0], 1);

        for (int i=k; i<k+1; i++) {
            dataTest.add(data.get(i));
            forecastsTest.add(forecastFinal);
```

```
        }
        if (dataTest.size() == forecastsTest.size() && dataTest.size()
== 1) {
            mapes.add(exam.mape(dataTest, forecastsTest));
        }
        validation.clear();
        forecasts.clear();
        dataTest.clear();
        forecastsTest.clear();
        }

        Double sum =0.0;
        for (int i=0; i<mapes.size(); i++) {
            sum = sum + mapes.get(i);
            //System.out.println(mapes.get(i));
        }

        //System.out.println(mapes.size()+"\n");
        System.out.println("mape = " + sum/mapes.size());
    }

}
```

Εικόνα 41: Κώδικας Java για την βελτίωση της μεθόδου III (κλάση " SESWith168DifferentModelsImprovement")

6.4.2. ΑΠΟΤΕΛΕΣΜΑΤΑ

Τρέχοντας τον κώδικα της υποπαραγράφου 6.4.1 και με αρχείο παραμετροποίησης το:

```
{
    "Latitude":"37.951575",
    "Longitude":"23.66700232",
    "Start_Time":{
        "Month":"6",
        "Year":"2012",
        "Leap_Year":"no"
    },
    "End_Time":{
        "Month":"1",
        "Year":"2013",
        "Leap_Year":"no"
    },
    "Data_Timestamp_Format":"yyyy-MM-dd HH:mm:ss",
    "Number_of_Values_per_Hour":"4"
}
```

Εικόνα 42: Παράδειγμα αρχείου παραμετροποίησης V

λαμβάνουμε το εξής mape:

```
2012-09-23 13:00:00
7
23.12
23.12
23.12
23.12
23.12
23.12
23.12

2012-11-11 01:00:00
2
20.520000000000003
20.520000000000003

2012-11-12 07:00:00
3
16.560000000000002
16.560000000000002
16.560000000000002

mape = 6.218344832926391
```

Εικόνα 43: Βελτίωση της μεθόδου III - Περίπτωση I

Η περίπτωση I περιγράφεται στην υποπαράγραφο 1.4.3.1 (***Περιπτώσεις εφαρμογής 1***)

Παρατηρούμε ότι το 6,22% είναι σαφώς μικρότερο από το 12,22% (αποτέλεσμα μεθόδου III – Ιανουάριος 2013 – παράγραφος 6.2).

Ομοίως τρέχοντας τον κώδικα της υποπαραγράφου 6.4.1 και με αρχείο παραμετροποίησης το:

```
{
    "Latitude":"37.951575",
    "Longitude":"23.66700232",
    "Start_Time":{
        "Month":"1",
        "Year":"2013",
        "Leap_Year":"no"
    },
    "End_Time":{
        "Month":"5",
        "Year":"2013",
        "Leap_Year":"no"
    },
    "Data_Timestamp_Format":"yyyy-MM-dd HH:mm:ss",
    "Number_of_Values_per_Hour":"4"
}
```

Εικόνα 44: Παράδειγμα αρχείου παραμετροποίησης VI

λαμβάνουμε το εξής mape:

```
2013-02-04 07:30:00
2
19.68
19.68


2013-02-21 19:30:00
3
5.76
5.76
5.76


2013-03-31 03:00:00
4
23.439999999999998
23.439999999999998
23.439999999999998
23.439999999999998


2013-04-22 22:00:00
3
7.88
7.88
7.88


mape = 5.134168916447202
```

Εικόνα 45:Βελτίωση της μεθόδου ΙΙΙ - Περίπτωση ΙΙ

Η περίπτωση ΙΙ περιγράφεται στην υποπαράγραφο 1.4.3.1 (***Περιπτώσεις εφαρμογής 2***).

Παρατηρούμε ότι το 5,13% είναι σαφώς μικρότερο από το 7,85% (αποτέλεσμα μεθόδου ΙΙΙ – Μάιος 2013 – παράγραφος 6.2).

7. ΣΥΜΠΕΡΑΣΜΑΤΑ

Σκοπός αυτού του κεφαλαίου είναι να συγκρίνουμε τα αποτελέσματα όλων των μεθόδων που υλοποιήσαμε, προκειμένου, εάν αυτό είναι δυνατό, να εξάγουμε χρήσιμα συμπεράσματα σχετικά με την ενεργειακή κατανάλωση του κτιρίου. Για λόγους ευκολίας παραθέτουμε εδώ τα συνοπτικά αποτελέσματα:

ΠΛΗΘΟΣ ΙΣΤΟΡΙΚΩΝ ΜΗΝΩΝ (+1η ΕΒΔΟΜΑΔΑ)	2η ΕΒΔΟΜΑΔΑ 2013	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
MIN MAPE		10,79	6,88	7,15	8,25	3,87	6,89	9,90	11,04	4,06	10,98	6,39	13,77
AVERAGE MAPE		14,48	12,43	8,39	12,13	6,85	12,93	14,40	13,70	5,90	14,96	8,44	15,55

Εικόνα 46: Συνοπτικά αποτελέσματα της μεθόδου I

ΠΛΗΘΟΣ ΙΣΤΟΡΙΚΩΝ ΜΗΝΩΝ (+1η ΕΒΔΟΜΑΔΑ)	2η ΕΒΔΟΜΑΔΑ 2013	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
MIN MAPE		30,03	12,44	22,07	8,34	12,86	7,20	10,61	9,83	5,16	39,28	8,25	28,93
AVERAGE MAPE		32,59	25,84	29,56	10,47	15,34	13,43	13,85	12,47	7,62	46,69	10,10	31,72

Εικόνα 47: Συνοπτικά αποτελέσματα της μεθόδου II

ΠΛΗΘΟΣ ΙΣΤΟΡΙΚΩΝ ΜΗΝΩΝ (+1η ΕΒΔΟΜΑΔΑ)	2η ΕΒΔΟΜΑΔΑ 2013	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
MIN MAPE		12,22	7,85	7,46	5,17	7,85	8,10	9,17	5,03	6,38	10,33	5,77	7,60
AVERAGE MAPE		12,66	8,53	7,87	5,76	9,20	8,71	10,00	5,35	6,94	11,44	6,23	8,27
ΜΙΝ ΜΑΡΕ ΧΩΡΙΣ ΤΗΝ(ΤΙΣ) ΑΡΓΙΑ(ΕΣ) ΑΜΕΣΩΣ ΠΡΙΝ ΤΗΝ ΠΡΟΒΛΕΨΗ		6,22				5,13							

Εικόνα 48: Συνοπτικά αποτελέσματα της μεθόδου III

Παρατηρώντας τα παραπάνω συνοπτικά αποτελέσματα συμπεραίνουμε ότι:

Η μέθοδος ΙΙ, δηλαδή η προσπάθειά μας να βελτιώσουμε την μέθοδο Ι αφαιρώντας την επίδραση της μέσης θερμοκρασίας από τα δεδομένα της χρονοσειράς, σε γενικές γραμμές δεν οδήγησε σε καλύτερα αποτελέσματα. Ιδιαίτερα, οι προβλέψεις μας, με αυτήν τη μέθοδο, για τους φθινοπωρινούς και χειμερινούς μήνες παρουσίασαν μεγάλα σφάλματα. Μία εξήγηση γι' αυτό το γεγονός είναι ότι η χρονοσειρά των δεδομένων και η αντίστοιχη των μέσων ημερήσιων θερμοκρασιών έχουν συντελεστή γραμμικής συσχέτισης $r_{XY} = 0{,}3069426$, δηλαδή ουσιαστικά δεν έχουν γραμμική συσχέτιση μεταξύ τους.

Η βελτιωμένη μέθοδος ΙΙΙ (παράγραφοι 1.4.3.1 και 6.4) παράγει σαφώς τις καλύτερες προβλέψεις από όλες τις μεθόδους. Η βασική διαφορά της με την μέθοδο Ι είναι ότι στην μέθοδο ΙΙΙ δεν λαμβάνει χώρα η κλασική αποσύνθεση. Είναι πιθανό, λόγω του πολύ μεγάλου μήκους εποχιακότητας (το 168 είναι πολύ μεγάλο ως αριθμός), οι δείκτες εποχιακότητας να μην αποτυπώνουν πλήρως την εποχιακή συμπεριφορά της χρονοσειράς. Συνεπώς, ήταν καλύτερο να θεωρήσουμε την χρονοσειρά μας σαν 168 διαφορετικές χρονοσειρές, στις οποίες παράγουμε προβλέψεις. Πράγματι το μέσο min mape της βελτιωμένης μεθόδου ΙΙΙ είναι **7,02%**, το οποίο είναι πολύ ικανοποιητικό για την φύση των δεδομένων που μελετήσαμε.

8. ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Makridakis, S., Hogarth R. and Gaba A. (2010). Dance with Chance: Making Luck Work for You, Oneworld Publications

[2] Holt, C. C. (1957). Forecasting seasonals and trends by exponentially weighted averages. O. N. R. Memorandum 52/1957. Pittsburgh: Carnegie Institute of Technology. Reprinted with discussion in 2004. International Journal of Forecasting, 20, 5-13.

[3] Gardner, Jr. E. S., Jr., & McKenzie, E. (1985). Forecasting trends in time series. Management Science, 31, 1237-1246.

[4] Assimakopoulos, V. and Nikolopoulos, N. (2000) "The theta model: a decomposition approach to forecasting', International Journal of Forecasting, Vol. 16, No. 4, pp. 521-530.

[5] Nikolopoulos, K., Assimakopoulos, V., Bougioukos, N. and Petropoulos F. (2008) "Advances in Theta model", Working Paper No. 0023, University of Peloponnese, Department of Economics.

[6] Hyndman, R. J. and Billah, B. (2003). Unmasking the Theta method. International Journal of Forecasting, 19:287–290.

[7] Wright, D.J., Capon, G., Page, R., Quiroga, J., Taseen, A.A. and Tomasini, F. (1986) "Evaluation of forecasting methods for decision support", International Journal of Forecasting, Vol. 2, No. 2, pp. 139–153.

[8] Yokum, J. T. and Armstrong, J. S. (1995) "Beyond accuracy: Comparison of criteria used to select forecasting methods", International Journal of Forecasting, Vol. 11, pp. 591-597.

[9] Tashman, L.J. and Leach, M.L. (1991), "Automatic forecasting software: a survey and evaluation'', International Journal of Forecasting, Vol. 7, No 2, pp. 209-30.

[10] Tashman L. J. (2000) "Out-of-sample tests of forecasting accuracy: an analysis and review", International Journal of Forecasting, Vol. 16, No. 4, pp. 437-450.

[11] Chatfield, C. (1988) "Apples, Oranges and Mean Square Error", International Journal of Forecasting, Vol. 4, pp. 515-518.

[12] Clemen, R.T. (1989) "Combining forecasts: A review and annotated biography (with discussion)" International Journal of Forecasting, Vol. 5, pp. 559-583.

[13] Armstrong, J. S. (2001b). "Combining forecasts", In: J. S. Armstrong (Ed.), Principles of forecasting: A handbook for researchers and practitioners, Boston, MA: Kluwer Academic Publishing, pp. 417 – 439.

[14] Makridakis, S. (1993) "Accuracy measures - theoretical and practical concerns", International Journal of Forecasting, Vol. 9, pp. 527-529.

[15] Armstrong, J. S. (2001a). "Evaluating forecasting methods", In: J. S. Armstrong (Ed.), Principles of forecasting: A handbook for researchers and practitioners, Boston, MA: Kluwer Academic Publishing, pp. 443 – 472.

[16] Goodwin, P. and Lawton, R. (1999) "On the asymmetry of the symmetric MAPE", International Journal of Forecasting, Vol. 15, pp. 405-408.

[17] Πετρόπουλος Φ., Ασημακόπουλος Β., (2011). "Επιχειρησιακές Προβλέψεις". εκδόσεις συμμετρία, Αθήνα

[18] JSON: http://json.org/

[19] Java κλάση Date: http://docs.oracle.com/javase/7/docs/api/java/util/Date.html
[20] Java κλάση SimpleDateFormat: https://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html
[21] Weather Underground: http://www.wunderground.com/