ARISTOTLE UNIVERSITY OF THESSALONIKI
FACULTY OF SCIENCES
SCHOOL OF INFORMATICS
DEPARTMENT OF COMPUTER SCIENCE
«KNOWLEDGE, DATA AND SOFTWARE TECHNOLOGIES»

Master Thesis:

Machine Learning Techniques for Short-Term Electric Load Forecasting

Praxitelis-Nikolaos Kouroupetroglou
UID:629

Supervisor:
Grigorios Tsoumakas, Assistant Professor

September 2017

Abstract

Economic growth in the modern world, depends directly on the availability of electric energy, especially because most societies, industries, economies etc depend almost entirely on its use. The availability of a source of continuous, cheap, and reliable energy is of foremost economic importance. Electric load forecasting is an important tool used to ensure that the energy supplied by utilities meets the consumers needs. To this end, a staff of trained personnel is needed to carry out this specialized function. Load forecasting is always defined as basically the science or art of predicting the future load on a given system, for a specified period of time ahead. These predictions may be just for a fraction of an hour ahead for operation purposes, or as much as 20 years into the future for planning purposes.

The purpose of this master thesis is to create accurate machine learning models for short-term electric load demand forecasting in short-term horizon (for 1 day ahead) for the Greek Electric Network Grid.

The load data that the Thesis uses originates from IPTO which stands for Independent Power Transmission Operator. Meteorological Features taken from Dark Sky API. To compare the Thesis predictions to the system predictions, next day current load predictions from OoEM which stands for Operator of Electricity Market in Greece was used.

The datasets were downloaded from its origins, preprocessed, cleaned and six machine learning algorithms was used. After four different types of experiments, the combination of SVM, XGBoost and Model Trees models was used in order to get 2.4% prediction error and on the other hand OoEM gives 2.53% prediction error. Hence the Thesis models improve the prediction performance by reducing the prediction error by +4.74% than the OoEM predictions.

The code of this master thesis can be found at github repo and results - visualizations can be found at shinyapps.

Praxitelis-Nikolaos Kouroupetroglou
September - 2017

Keywords: Machine Learning, Supervised Learning, Forecasting, Regression

Περίληψη

Η οικονομική ανάπτυξη του σύγχρονου κόσμου εξαρτάται άμεσα από τη διαθεσιμότητα ηλεκτρικής ενέργειας, ιδιαίτερα επειδή οι περισσότερες κοινωνίες, οι βιομηχανίες, οι οικονομίες κλπ. Εξαρτώνται σχεδόν εξ ολοκλήρου από τη χρήση της. Η διαθεσιμότητα μιας πηγής συνεχούς, φτηνής και αξιόπιστης ενέργειας έχει πρωταρχική οικονομική σημασία. Η πρόβλεψη του ηλεκτρικού φορτίου είναι ένα σημαντικό εργαλείο που χρησιμοποιείται για να εξασφαλίσει ότι η ενέργεια που παρέχεται από επιχειρήσεις κοινής ωφελείας ικανοποιεί τις ανάγκες των καταναλωτών. Για το σκοπό αυτό, απαιτείται προσωπικό εξειδικευμένο προσωπικό για να ασκεί αυτή την εξειδικευμένη λειτουργία. Η πρόβλεψη φορτίου ορίζεται πάντα ως βασικά η επιστήμη ή η τέχνη της πρόβλεψης του μελλοντικού φορτίου σε ένα δεδομένο σύστημα για μια καθορισμένη χρονική περίοδο. Αυτές οι προβλέψεις μπορεί να είναι μόνο για ένα κλάσμα μιας ώρας μπροστά για επιχειρησιακούς σκοπούς, ή μέχρι 20 χρόνια στο μέλλον για σκοπούς προγραμματισμού.

Σκοπός αυτής της διπλωματικής εργασίας είναι η δημιουργία μοντέλων με ακρίβεια πρόβλεψης της βραχυπρόθεσμης πρόβλεψης της ζήτησης ηλεκτρικού φορτίου (έως για 1 μέρα) για το Ελληνικό Ηλεκτρικό Δίκτυο.

Τα δεδομένα ηλεκτρικού φορτίου προέρχονται από την ΑΔΜΗΕ που σημαίνουν τα αρχικά της ”Ανεξάρτητος Διαχειριστής Ηλεκτρικής Ενέργειας”. Μετεωρολογικά δεδομένα αντλήθηκαν από Dark Sky API και για να συγκρίνω τις προβλέψεις της διπλωματικής εργασίας, προβλέψεις από τη ΛΑΓΗΕ που σημαίνει Λειτουργός Αγορά Ηλεκτρικής Ενέργειας.

Τα σύνολα δεδομένων αντλήθηκαν από την προέλευσή τους, προεπεξεργάστηκαν, και χρησιμοποιήθηκαν έξι αλγόριθμοι μηχανικής μάθησης. Μετά από τέσσερις διαφορετικούς τύπους πειραμάτων χρησιμοποιήθηκε ο συνδυασμός μοντέλων SVM, XGBoost και Model Trees προκειμένου να ληφθεί σφάλμα πρόβλεψης 2.4% ενώ η ΛΑΓΗΕ δίνει 2.53% σφάλμα πρόβλεψης. Έτσι, τα μοντέλα της συγκεκριμένης διπλωματικής εργασίας βελτιώνουν την απόδοση πρόβλεψης με μείωση του σφάλματος πρόβλεψης κατά +4.74% από την ΛΑΓΗΕ.

Ο κώδικας της διπλωματικής εργασίας μπορεί να βρεθεί στο αποθετήριο github και τα αποτελέσματα - οπτικοποιήσεις των αναλύσεων βρίσκονται στο: shinyapps.

Πραξιτέλης - Νικόλαος Κουρουπέτρογλου
Σεπτέμβριος 2017

Keywords: Μηχανική Μάθηση, Μάθηση με επίβλεψη, Πρόβλεψη, Παλινδρόμηση

Acknowledgments

At first, I would like to thank my parents for their support, patience for aiding me for my efforts and their love all these years. I want to thank my master thesis supervisor Dr. Grigorios Tsoumakas, who guided throw the master thesis and assisted me when I faced issues, problems and dead ends. Moreover I want to thank other professors that I met and signed in their postgraduate courses, such as Dr. Ioannis Vlahavas, Dr. Eleutherios Aggelis, Dr. Athina Vakali, Dr. Nikolaos Vaseiliadis, Dr. Anastasios Gounaris, Dr. Apostolos Papadopoulos and Dr. Konstantinos Tsichlas. Their postgraduate courses helped me to explore and understand the world of "Data Science" from many views and aspects. Lastly I want to thank all those who provide the rich and useful information from Internet sites like: stackoverflow, stats.stackexchange, datascience.stackexchange, tex.stackexchange, kdnuggets, analyticsvidhya.com, machinelearningmastery, r-bloggers and of course wikipedia. if these sites had not been there, I could not have finished the Thesis.

Contents

List of Figures

List of Tables

Chapter 1

Electric Load Forecasting - Introduction

In this chapter an introduction will be given about the subject of this master thesis and a summary of the contents that follow.

1.1 Reasoning behind the thesis

This master thesis attempts to solve the problem of electrical load demand forecasting by constructing accurate machine learning models. This is a very important issue for the electrical companies who have to meet the needs of their customers. Knowing how much load will be demanded in the near future is beneficial for an electrical company in terms of administration, economics, marketing, and transmission of electrical power.

1.2 What is the problem and why it is important

The economic progress in every society depends on the abundance of electric power. The abundance of sources for electricity production and the continuous existence of electric power is one of at most importance.

Nowadays, the electric load prediction is a vital process with applications that are used in various fields, such as in an electric company producing electrical power. Many offices like the marketing department or the trade market inside such a company can benefit from electric load prediction. The business reasons using electric load prediction are in summary the following [1]:

1. For purchasing and producing electric power.
2. For transmitting, transferring and distributing electric power.
3. For managing and maintaining the electric power sources.
4. For managing the daily electric load demand.
5. For financial and marketing planning.

We observe that the reasons for predicting the electric load demand are important and vital for an electric production company.

The next table 1.1 presents according to the lead time range of each business need as described above, the ”Minimum updating cycle” which shows the minimum time that the future load prediction must be updated and ”max horizon” of the forecasts which shows how far in the forecasts can be made.

Table 1.1: Electric Load Forecasting with Minimum Updating cycle and maximum horizon per business needs

	Minimum updating cycle	**max horizon**
purchasing and producing electric power	1 hour	10 years and above
transmitting, transferring and distributing electric power.	1 day	30 years
managing and maintaining the electric power sources.	15 minutes	2 weaks
managing the daily electric load demand	15 minutes	10 years and above
financial and marketing	1 month	10 years and above

As it is shown from the table above 1.1, the electric load prediction is an useful tool for an electric company that produce electric power because in this modern current world electric companies must provide adequate amount of electrical power for the needs of consumers. In general the electric load prediction is defined as the methodology for the load forecasting for a specific duration/horizon. The maximum horizon can be at most 20 years [2].

There is no formal or typical procedure for electric load forecasting for every type of electric company and for every duration of time. The forecasting depends on various factors such as the means and sources of electrical production by each electrical company, the demand for electric load, climate factors, economical reasons and the human activity [1].

Climate factors are those who are based on meteorological features like temperature, humidity, wind speed, precipitation etc. Based on the scientific literature temperature plays an important role for electric load consumption and many electric load forecasting systems use temperature as a feature. However temperature can be used for small horizon for example for 1-2 days of forecast and it is an unreliable factor for more than that [2].

The impact of human activity on the electric load consumption can be analyzed in various features. One type of such features are the calendar features such as the day of the week or the month of the year. Other such features concern economical factors, like the economic activities in urban areas or the economic transactions. For a long-term forecasting, e.g. forecasting for a year, economic factors play a vital role for future prediction. Moreover this kind of forecasting can use the human activities from rural areas such as the agricultural activities and the changes in rural land [1].

The different types of electric load forecasting based on the data/features available that affect the forecasting and in conjunction with the duration of time for the prediction, can be categorized in the following categories:

1. Very Short Term Load Forecasting (VSTLF). Can be used for small time-window, some hours at most and it utilizes the amount of previous hourly loads on that current day. It can not use information from economic factors or land because they do not change in this small amount of time.

2. Short Term Load Forecasting (STLF). This type of forecasting can be used from 24-hour forecasting up two 2 weeks. The difference from the previous forecasting method is that here temperature and in general climate/meteorological factors play a vital role for the forecasting. STLF can be used for decision-taking like how much energy should an electric company purchase from other electric companies in the near future.

3. Medium Term Load Forecasting (MTLF). This forecasting method can be used for 1 month to 3 years horizon. Temperature and other climate factors do not be used for forecasting due to their

unavailability of their own forecasting for this horizon. So other ways are used such as simulating their behavior and are being forecasting with this technique in order to be used of MTLF loaf forecasting. MTLF uses economical factors because their impact plays a role in daily life and for the needs of consumers. The same thing applies to the change of land in rural areas.

4. Long Term Load Forecasting (LTLF). This forecasting is used for time-window 3-10 years. Simulation techniques are used to calculate climate factors and the economic transactions/activities.

In the previous 4 categories of load forecasting, the model that is used for predictions, it describes the relationship between the electric load and the features that affect it such as the climate factors, economical factors, land use etc. However, for the load forecasting for long periods such as MTLF and LTLF, climate factors and economical features can not precisely be determined for this long period of time.

The following table 1.2 summarizes the availability of climate features, economical factors and land use for load forecasting.

Table 1.2: Availability of climate factors, economics, and land use information for load forecasting

factors / forecasting accuracy	Accurate	Inaccurate	Unreliable
climate factors	1 day	2 weeks	>2 weeks
Economics	1 month	3 years	>3 years
Land Use	3 years	5 years	>5 years

In addition, the following table 1.3 categorizes the different types of forecasting based on the horizon for the prediction and the different kind of business needs for prediction.

Table 1.3: Classification of different types of electric load forecasting based on features and factors

	Climate factors	Economics	land Use	Updating Cycle	Horizon
VSTLF	Optional	Optional	Optional	<= 1Hour	1 day
STLF	Required	Optional	Optional	1 Day	2 weeks
MTLF	Simulated	Required	Optional	1 month	3 years
LTLF	Simulated	Simulated	Required	3 years	30 years

Based on the previous table we can relate every type of forecasting with the suitable features/factors needed to be applied. Furthermore, the different types of forecasting can be related with the different business need as mentioned above. The following table 1.4 presents this relationship. Due to the fact that many business needs are being applied in various time periods/horizons, thus various of the previous forecasting techniques can be used for the same business need.

Table 1.4: Applications of different types of electric load forecasts in Business Needs

	VSTLF	STLF	MTLF	LTLF
purchasing and producing electric power	X	X	X	X
transmitting, transferring and distributing electric power		X	X	X
managing and maintaining the electric power sources	X	X		
managing the daily electric load demand	X	X	X	X
financial and marketing planning			X	X

1.3 Summary of the contents of the thesis

The Thesis is composed by the following chapters:

- Chapter 1: **Electric Load Forecasting - Introduction:**
  This chapter presents the Electricity Load Demand Forecasting, trying to describe the issue, why it is important and commercial software on the Internet that deal with it.
- Chapter 2: **Theoretical Background and Data Science Development Platforms:**
  This chapter describes the theoretical background for six models that was used for this thesis. Moreover this chapter presents a variety of statistical and data analysis software platforms and languages to implement the forecasting.
- chapter 3: **Scientific Review and Commercial Software:**
  This chapter shows statistical approaches and artificial intelligence - machine learning techniques found from the scientific literature.
- chapter 4: **Application - ML Models for STLF in Greek Electrical Grid:**
  This chapter presents the implementation in R of machine learning models, their predictions, their accuracy performance and finally the visualizations in R-Shiny, R-Markdown.
- Chapter 5: **Conclusions - Final Remarks:**
  This chapter summarize the developed application, the produced models, the results and the derived conclusions from them.
- chapter 6: **Future Work:**
  This chapter suggest future experiments and improvements for the current developed models.
- chapter 7: **Appendix of Tables:**
  This chapter is appendix of tables that presents various information about the machine learning experiments and its features in table format.

Chapter 2

Theoretical Background and Data Science Development Platforms

2.1 Machine Learning

Tom Mitchell in his book "machine Learning" [1], in his introduction, he provides a short formalism that you'll see much repeated: "Machine Learning is the process which, a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

A more understanding definition is that Machine learning is a core sub-area of artificial intelligence as it enables computers to get into a mode of self-learning without being explicitly programmed. When exposed to new data, computer programs, are enabled to learn, grow, change, and develop by themselves. Machine learning focuses on that development of computer programs that can access data and use it learn for themselves.

Machine learning is closely related to computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with data mining, where the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning. Machine learning can also be unsupervised and be used to learn and establish baseline behavioral profiles for various entities and then used to find meaningful anomalies.

Machine learning tasks are typically classified into three broad categories, depending on the nature of the learning. These are:

- **Supervised learning**: The computer is presented with example inputs and their desired outputs, and the goal is to learn a general rule that maps inputs to outputs.
- **Unsupervised learning**: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).
- **Semi-Supervised learning**: Between supervised and unsupervised learning is semi-supervised learning, where the computer is provided with an incomplete training set which is a training set with some (often many) of the target outputs missing. is a class of learning tasks and techniques that also make use of unlabeled data for training – typically a small amount of labeled data with a large amount of unlabeled data. Many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce considerable improvement in learning accuracy.

[1] Tom Mitchell - Machine Learning

- **Reinforcement learning**: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). The program is provided feedback in terms of rewards and punishments as it navigates its problem space.

Another categorization of machine learning tasks arises when one considers the desired output of a machine-learned system:

- **In classification**, inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This is typically tackled in a supervised way. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are ”spam” and ”not spam”.
- **In regression**, also a supervised problem, the outputs are continuous rather than discrete.
- **In clustering**, a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.

2.2 Supervised Learning

This project creates models in order to predict the day-ahead future loads having labeled features from meteorological and calendar data and the target - outputs are continuous load values, hence belongs to the tasks of supervised learning. Supervised learning is the machine learning task of inferring a function from labeled training data. The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a ”reasonable” way.

2.3 Theoretical Background for Various Models

For this master thesis, the theoretical background for six supervised learning models - approaches will be described. These was used for the analysis of the master thesis for STLF (Short-Term Load Forecasting) in the Greek Network. These models are: SVM, Neural Networks, K-Nearest Neighbors, Random Forest, XGBoost, and Rule Based Model Trees.

2.3.1 SVM Theoretical Background

Support Vector Machines are very specific class of algorithms, characterized by usage of kernels, absence of local minima, sparseness of the solution and capacity control obtained by acting on the margin, or on number of support vectors, etc. They were invented by Vladimir Vapnik and his co-workers, and first introduced at the Computational Learning Theory (COLT) 1992 conference with the paper. All these nice features however were already present in machine learning since 1960’s: large margin hyper planes usage of kernels, geometrical interpretation of kernels as inner products in a feature space. Similar optimization techniques were used in pattern recognition and sparsness techniques were widely discussed. Usage of slack variables to overcome noise in the data and non - separability was also introduced in 1960s. However it was not until 1992 that all these features were put together to form the maximal margin classifier, the basic Support Vector Machine, and not until 1995 that the soft margin version was introduced [36, 37].

Support Vector Machine can be applied not only to classification problems but also to the case of regression. Still it contains all the main features that characterize maximum margin algorithm: a non-linear function is leaned by linear learning machine mapping into high dimensional kernel induced feature space. The capacity of the system is controlled by parameters that do not depend on the dimensionality of feature space.

In the same way as with classification approach there is motivation to seek and optimize the generalization bounds given for regression. They relied on defining the loss function that ignores errors, which are situated within the certain distance of the true value. This type of function is often called – epsilon intensive – loss function. The figure below 2.1 shows an example of one-dimensional linear regression function with – epsilon intensive – band. The variables measure the cost of the errors on the training points. These are zero for all points that are inside the band [**?**].

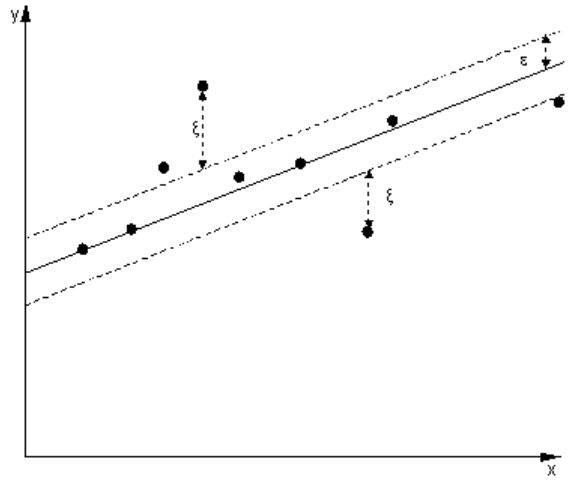

Figure 2.1: One-dimensional linear regression with epsilon intensive band.

Another figure 2.2 shows a similar situation but for non-linear regression case.

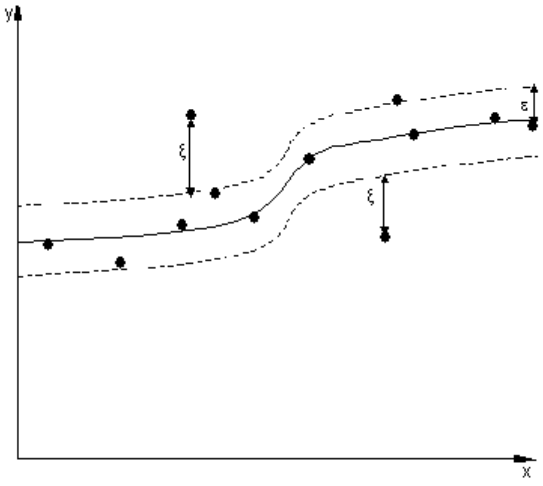

Figure 2.2: Non-linear regression function

One of the most important ideas in Support Vector Classification and Regression cases is that presenting the solution by means of small subset of training points gives enormous computational advantages. Using the epsilon intensive loss function we ensure existence of the global minimum and at the same time optimization of reliable generalization bound. In SVM regression, the input is first mapped onto a m-dimensional feature space using some fixed (nonlinear) mapping, and then a linear model is constructed in this feature space. Using mathematical notation, the linear model (in the feature space) $f(x, w)$ is

given by: $f(x,w) = \sum_{j=1}^{m} w_i g_i(x) + b$ where $g_j(x), j = 1, ..., m$ denotes a set of nonlinear transformations, and b is the "bias" term. Often the data are assumed to be zero mean (this can be achieved by preprocessing), so the bias term is dropped [**?**].

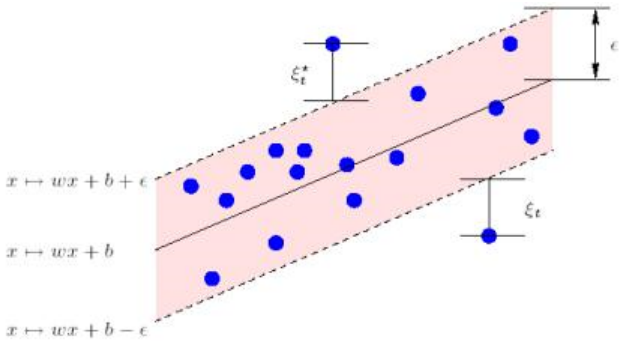

Figure 2.3: Non-linear mapping of input examples into high dimensional feature space. (Classification case, however the same stands for regression as well).

The quality of estimation is measured by the loss function $L(y, f(x,w))$. SVM regression uses a new type of loss function called -insensitive loss function proposed by Vapnik:

$$L(y, f(x,w)) = \begin{cases} 0 & if|y - f(x,w)| \leq \epsilon \\ |y - f(x,w)| - \epsilon & otherwise \end{cases}$$

The empirical risk is:

$$R_{(emp)}(w) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i, w))$$

SVM regression performs linear regression in the high-dimension feature space using ε-insensitive loss and, at the same time, tries to reduce model complexity by minimizing $||w||^2$. This can be described by introducing (non-negative) slack variables , to measure the deviation of training samples outside - insensitive zone. Thus SVM regression is formulated as minimization of the following functional [36, 37]:

$$min \frac{1}{2}||w^2|| + C \sum_{i=1}^{n} (\xi_i + \xi_i^*) \begin{cases} y_i - f(x_i, w) \leq \epsilon + \xi_i^* \\ f(x_i, w) - y_i \leq \epsilon + \xi_i \\ \xi_i, \xi_i^* \leq 0, i = 1, ..., n \end{cases} , s.t.$$

This optimization problem can transformed into the dual problem and its solution is given by:

$$f(x) = \sum_{i=1}^{n_{SV}} (a_i - a_i^*) K(x_i, x), s.t. 0 \leq a_i^* \leq C, 0 \leq a_i \leq C,$$

where n_{SV} is the number of Support Vectors (SVs) and the kernel function:

$$K(x, x_i) = \sum_{j=1}^{m} g_j(x) g_j(x_i)$$

It is well known that SVM generalization performance (estimation accuracy) depends on a good setting of meta-parameters parameters C, and the kernel parameters. The problem of optimal parameter selection is further complicated by the fact that SVM model complexity (and hence its generalization performance) depends on all three parameters. Existing software implementations of SVM regression usually treat SVM meta-parameters as user-defined inputs. Selecting a particular kernel type and kernel function parameters is usually based on application-domain knowledge and also should reflect distribution of input (x) values of the training data.

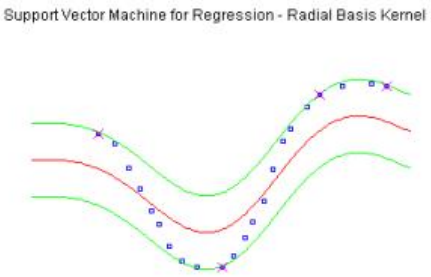

Figure 2.4: Performance of Support Vector Machine in regression case. The epsilon boundaries are given with the green lines. Blue points represent data instances.

Parameter C determines the trade off between the model complexity (flatness) and the degree to which deviations larger than are tolerated in optimization formulation for example, if C is too large (infinity), then the objective is to minimize the empirical risk only, without regard to model complexity part in the optimization formulation.

Parameter controls the width of the -insensitive zone, used to fit the training data. The value of can affect the number of support vectors used to construct the regression function. The bigger , the fewer support vectors are selected. On the other hand, bigger -values results in more "flat" estimates. Hence, both C and -values affect model complexity (but in a different way).

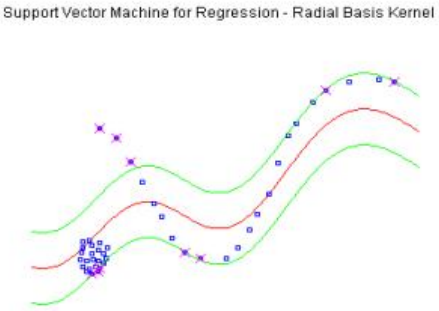

Figure 2.5: Performance of Support Vector Machine in regression case. The epsilon boundaries are given with the green lines. Blue points represent data instances.

(Additional instances are introduced in this case and after being supplied to the model are inside epsilon band. The regression function has changed. However this makes some points that were initially inside the interval to be misclassified.)

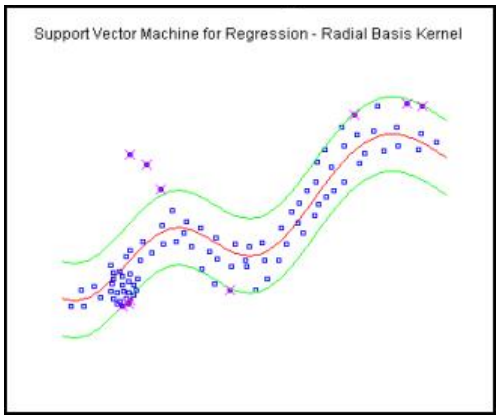

Figure 2.6: Performance of Support Vector Machine in regression case. The epsilon boundaries are given with the green lines. Blue points represent data instances.

One of the advantages of Support Vector Machine, and Support Vector Regression as the part of it, is that it can be used to avoid difficulties of using linear functions in the high dimensional feature space and optimization problem is transformed into dual convex quadratic programmes. In regression case the loss function is used to penalize errors that are grater than threshold - . Such loss functions usually lead to the sparse representation of the decision rule, giving significant algorithmic and representational advantages [36, 37].

2.3.2 Random Forest Theoretical Background

Random forests combine the predictions of multiple decision trees. The datasets are repeatedly divided into subtrees, guided by the best combination of variables. However, finding the right combination of variables can be difficult. For instance, a decision tree constructed based on a small sample might be not be generalizable to future, large samples. To overcome this, multiple decision trees could be constructed, by randomizing the combination and order of variables used. The aggregated result from these forest of trees would form an ensemble, known as a random forest [38, 39].

Random forest predictions are often better than that from individual decision trees. The chart below compares the accuracy of a random forest to that of its 1000 constituent decision trees. Only 12 out of 1000 individual trees yielded an accuracy better than the random forest. In other words, there is a 99% certainty that predictions from a random forest would be better than that from an individual decision tree [38, 39].

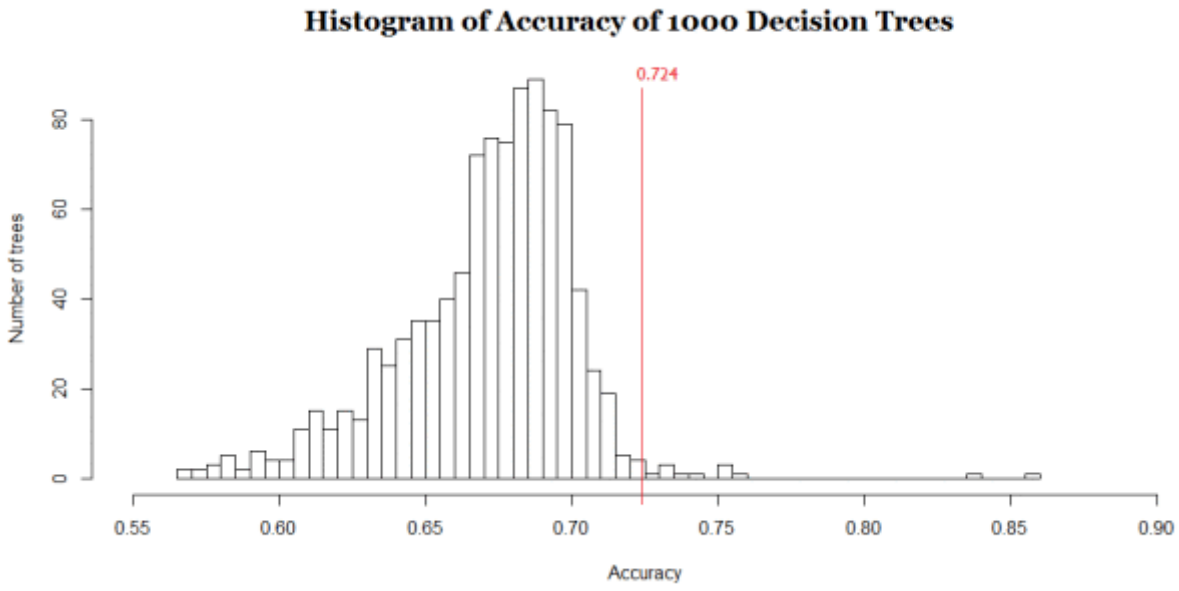

Figure 2.7: Histogram showing the accuracy of 1000 decision trees. While the average accuracy of decision trees is 67.1%, the random forest model has an accuracy of 72.4%, which is better than 99% of the decision trees.

Random forests are widely used because they are easy to implement and fast to compute. Unlike most other models, a random forest can be made more complex (by increasing the number of trees) to improve prediction accuracy without the risk of over-fitting.

A random forest is an example of an ensemble, which is a combination of predictions from different models. In an ensemble, predictions could be combined either by majority-voting or by taking averages.

Below in figure 2.8 is an illustration of how an ensemble formed by majority-voting yields more accurate predictions than the individual models it is based on [38, 39]:

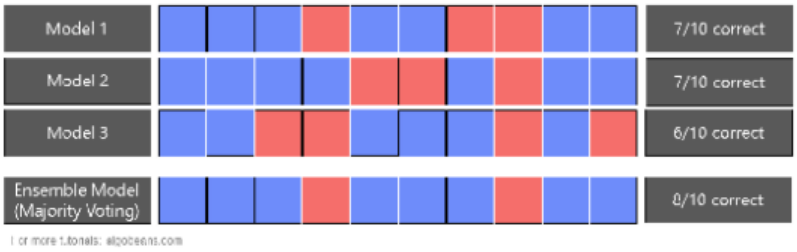

Figure 2.8: Example of three individual models attempting to predict 10 outputs of either Blue or Red. The correct predictions are Blue for all 10 outputs. An ensemble formed by majority voting based on the three individual models yields the highest prediction accuracy.

Random forest is an ensemble of multiple decision trees, it leverages "wisdom of the crowd", and is often more accurate than any individual decision tree. This is because each individual model has its own strengths and weakness in predicting certain outputs. As there is only one correct prediction but many possible wrong predictions, individual models that yield correct predictions tend to reinforce each other, while wrong predictions cancel each other out.

For this effect to work however, models included in the ensemble must not make the same kind of mistakes. In other words, the models must be uncorrelated. This is achieved via a technique called bootstrap aggregating (bagging) [38, 39].

In random forest, bagging is used to create thousands of decision trees with minimal correlation. In bagging, a random subset of the training data is selected to train each tree. Furthermore, the model randomly restricts the variables which may be used at the splits of each tree. Hence, the trees grown are dissimilar, but they still retain certain predictive power [38, 39].

Random forests are considered "black-boxes", because they comprise randomly generated decision trees, and are not guided by explicitly guidelines in predictions. We do not know how exactly the model came to the conclusion that a violent crime would occur at a specific location, instead we only know that a majority of the 1000 decision trees thought so. This may bring about ethical concerns when used in areas like medical diagnosis [38, 39].

Random forests are also unable to extrapolate predictions for cases that have not been previously encountered. For example, given that a pen costs $2, 2 pens cost $4, and 3 pens cost $6, how much would 10 pens cost? A random forest would not know the answer if it had not encountered a situation with 10 pens, but a linear regression model would be able to extrapolate a trend and deduce the answer of $20 [38].

2.3.3 k-Nearest Neighbors Theoretical Background

k-nearest neighbors is a simple algorithm that stores all available cases and predict the numerical target based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique [40].

A simple implementation of KNN regression is to calculate the average of the numerical target of the K nearest neighbors. Another approach uses an inverse distance weighted average of the K nearest neighbors. KNN regression uses the same distance functions as KNN classification [40].

The figure below shows 2.9 shows three distance measures that are only valid for continuous variables [40].

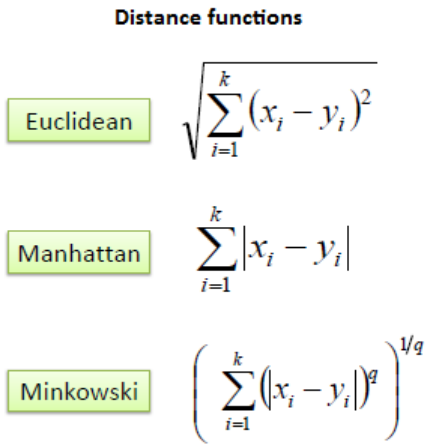

Figure 2.9: KNN distance functions

Choosing the optimal value for k is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise; however, the compromise is that the distinct boundaries within the feature space are blurred. Cross-validation is another way to retrospectively determine a good K value by using an independent data set to validate your K value. The optimal K for most datasets is 10 or more. That produces much better results than 1-NN [40].

2.3.4 Neural Networks Theoretical Background

Artificial neural networks (ANNs) were originally devised in the mid-20th century as a computational model of the human brain. Their used waned because of the limited computational power available at the time, and some theoretical issues that weren't solved for several decades (which I will detail at the end of this post). However, they have experienced a resurgence with the recent interest and hype surrounding Deep Learning. One of the more famous examples of Deep Learning.

It is theorized that because of their biological inspiration, ANN-based learners will be able to emulate how a human learns to recognize concepts or objects without the time-consuming feature engineering step. Whether or not this is true (or even provides an advantage in terms of development time) remains to be seen, but currently it's important that we machine learning researchers and enthusiasts have a familiarity with the basic concepts of neural networks.

Neural network terminology is inspired by the biological operations of specialized cells called neurons. A neuron is a cell that has several inputs that can be activated by some outside process. Depending on the amount of activation, the neuron produces its own activity and sends this along its outputs. In addition, specific input or output paths may be "strengthened" or weighted higher than other paths. The hypothesis is that since the human brain is nothing but a network of neurons, we can emulate the brain by modeling a neuron and connecting them via a weighted graph.

The artificial equivalent of a neuron is a node (also sometimes called neurons, but I will refer to them as nodes to avoid ambiguity) that receives a set of weighted inputs, processes their sum with its activation function ϕ, and passes the result of the activation function to nodes further down the graph. Note that it is simpler to represent the input to our activation function as a dot product [41]:

$$\phi\left(\sum_i w_i a_i\right) = \phi(\mathbf{w}^T\mathbf{a})$$

Figure 2.10: activation function as a dot product

Visually this looks like the following:

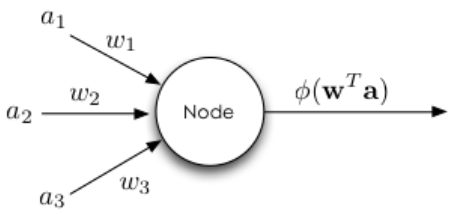

Figure 2.11: 1-node neural network

There are several canonical activation functions. For instance, the sigmoid activation function:

$$\phi(\mathbf{w}^T\mathbf{a}) = \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{a})}$$

Figure 2.12: sigmoid activation function

We can form a network by chaining these nodes together. Usually this is done in layers - one node layer's outputs are connected to the next layer's inputs.

Our goal is to train a network using labeled data so that we can then feed it a set of inputs and it produces the appropriate outputs for unlabeled data. We can do this because we have both the input x_i and the desired target output y_i in the form of data pairs. Training in this case involves learning the correct edge weights to produce the target output given the input. The network and its trained weights form a function (denoted h) that operates on input data. With the trained network, we can make predictions given any unlabeled test input [41].

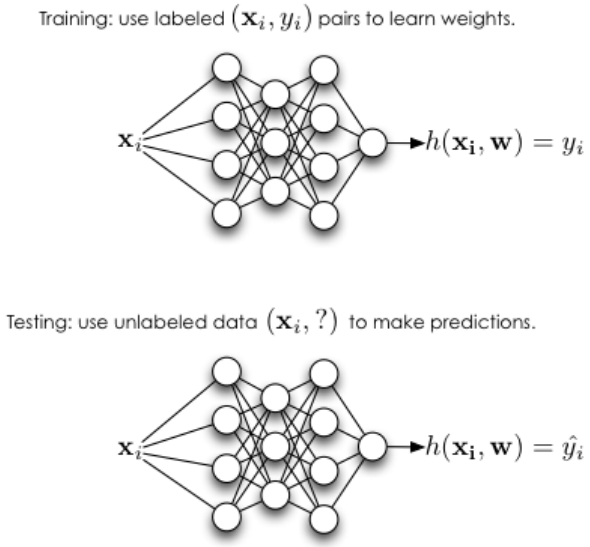

Figure 2.13: Training and testing in the neural network context. Note that a multilayer network is shown here.

In supervised training, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights which control the network. This process occurs over and over as the weights are continually tweaked. The set of data which enables the training is called the "training set." During the training of a network the same set of data is processed many times as the connection weights are ever refined.

If a network simply can't solve the problem, the designer then has to review the input and outputs, the number of layers, the number of elements per layer, the connections between the layers, the summation, transfer, and training functions, and even the initial weights themselves. Those changes required to create a successful network constitute a process wherein the "art" of neural networking occurs [41].

2.3.5 XGBoost Theoretical Background

XGBoost is short for "Extreme Gradient Boosting", is an open-source software library which provides the gradient boosting framework. The term "Gradient Boosting" is proposed in the paper Greedy Function Approximation: A Gradient Boosting Machine, by Friedman. XGBoost is based on this original model [42].

The GBM (boosted trees) has been around for really a while, and there are a lot of materials on them. It is an implementation of gradient boosting machines created by Tianqi Chen, now with contributions from many developers. It belongs to a broader collection of tools under the umbrella of the Distributed Machine Learning Community or DMLC[2] who are also the creators of the popular mxnet deep learning library[3] [42].

XGBoost is focused on computational speed and model performance. The implementation of the model supports the features of the scikit-learn and R implementations, with new additions like regularization. Three main forms of gradient boosting are supported [42]:

- Gradient Boosting algorithm also called gradient boosting machine including the learning rate.
- Stochastic Gradient Boosting with sub-sampling at the row, column and column per split levels.
- Regularized Gradient Boosting with both L1 and L2 regularization.

XGBoost algorithm was engineered for efficiency of compute time and memory resources. A design goal was to make the best use of available resources to train the model. Some key algorithm implementation features include [42]:

- Sparse Aware implementation with automatic handling of missing data values.
- Block Structure to support the parallelization of tree construction.
- Continued Training so that you can further boost an already fitted model on new data.

XGBoost dominates structured or tabular datasets on classification and regression predictive modeling problems. The evidence is that it is the go-to algorithm for competition winners on the Kaggle competitive data science platform [42].

The XGBoost library implements the gradient boosting decision tree algorithm. This algorithm goes by lots of different names such as gradient boosting, multiple additive regression trees, stochastic gradient boosting or gradient boosting machines [42].

Gradient Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made. A popular example is the AdaBoost algorithm that weights data points that are hard to predict [44].

Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models. This approach supports both regression and classification predictive modeling problems [44].

The idea of boosting came out of the idea of whether a weak learner can be modified to become better. The first realization of boosting that saw great success in application was Adaptive Boosting or

[2] DMLC

[3] mxnet deep learning library

AdaBoost for short. AdaBoost works by weighting the observations, putting more weight on difficult to classify instances and less on those already handled well. New weak learners are added sequentially that focus their training on the more difficult patterns. Predictions are made by majority vote of the weak learners' predictions, weighted by their individual accuracy. The most successful form of the AdaBoost algorithm was for binary classification problems and was called AdaBoost. AdaBoost and related algorithms were recast in a statistical framework. The statistical framework cast boosting as a numerical optimization problem where the objective is to minimize the loss of the model by adding weak learners using a gradient descent like procedure [44].

This class of algorithms were described as a stage-wise additive model. This is because one new weak learner is added at a time and existing weak learners in the model are frozen and left unchanged [44].

The generalization allowed arbitrary differentiable loss functions to be used, expanding the technique beyond binary classification problems to support regression, multi-class classification and more [44].

Gradient boosting involves three elements:

1. A loss function to be optimized.
2. A weak learner to make predictions.
3. An additive model to add weak learners to minimize the loss function.

The loss function used depends on the type of problem being solved.It must be differentiable, but many standard loss functions are supported and you can define your own. For example, regression may use a squared error and classification may use logarithmic loss. A benefit of the gradient boosting framework is that a new boosting algorithm does not have to be derived for each loss function that may want to be used, instead, it is a generic enough framework that any differentiable loss function can be used [44].

Decision trees are used as the weak learner in gradient boosting. Specifically regression trees are used that output real values for splits and whose output can be added together, allowing subsequent models outputs to be added and "correct" the residuals in the predictions. Trees are constructed in a greedy manner, choosing the best split points based on purity scores like Gini or to minimize the loss. Initially, such as in the case of AdaBoost, very short decision trees were used that only had a single split, called a decision stump. Larger trees can be used generally with 4-to-8 levels. It is common to constrain the weak learners in specific ways, such as a maximum number of layers, nodes, splits or leaf nodes. This is to ensure that the learners remain weak, but can still be constructed in a greedy manner [44].

Trees are added one at a time, and existing trees in the model are not changed. A gradient descent procedure is used to minimize the loss when adding trees. Traditionally, gradient descent is used to minimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating error or loss, the weights are updated to minimize that error. Instead of parameters, we have weak learner sub-models or more specifically decision trees. After calculating the loss, to perform the gradient descent procedure, we must add a tree to the model that reduces the loss (i.e. follow the gradient). We do this by parameterizing the tree, then modify the parameters of the tree and move in the right direction by (reducing the residual loss). Generally this approach is called functional gradient descent or gradient descent with functions.

The output for the new tree is then added to the output of the existing sequence of trees in an effort to correct or improve the final output of the model. A fixed number of trees are added or training stops once loss reaches an acceptable level or no longer improves on an external validation dataset [43, 44].

Gradient boosting is a greedy algorithm and can over-fit a training dataset quickly. It can benefit from regularization methods that penalize various parts of the algorithm and generally improve the performance of the algorithm by reducing over-fitting [43, 44].

Some of the enhancements to basic gradient boosting are the following:

- Tree Constraints
- Shrinkage

- Random sampling
- Penalized Learning

2.3.6 Rule-Based Model Trees Theoretical Background

The model tree approach described in Quinlan (1992) called M5, which is similar to regression trees except [45]:

- the splitting criterion is different
- the terminal nodes predict the outcome using a linear model (as opposed to the simple average)
- when a sample is predicted, it is often a combination of the predictions from different models along the same path through the tree.

The main implementation of this technique is a "rational reconstruction" of this model called M5', which is described by Wang and Witten (1997) and is included in the Weka software package [45].

When model trees make a split of the data, they fit a linear model to the current subset using all the predictors involved in the splits along the path. This process proceeds until there are not enough samples to split and/or fit the model. A pruning stage is later used to simplify the model [45].

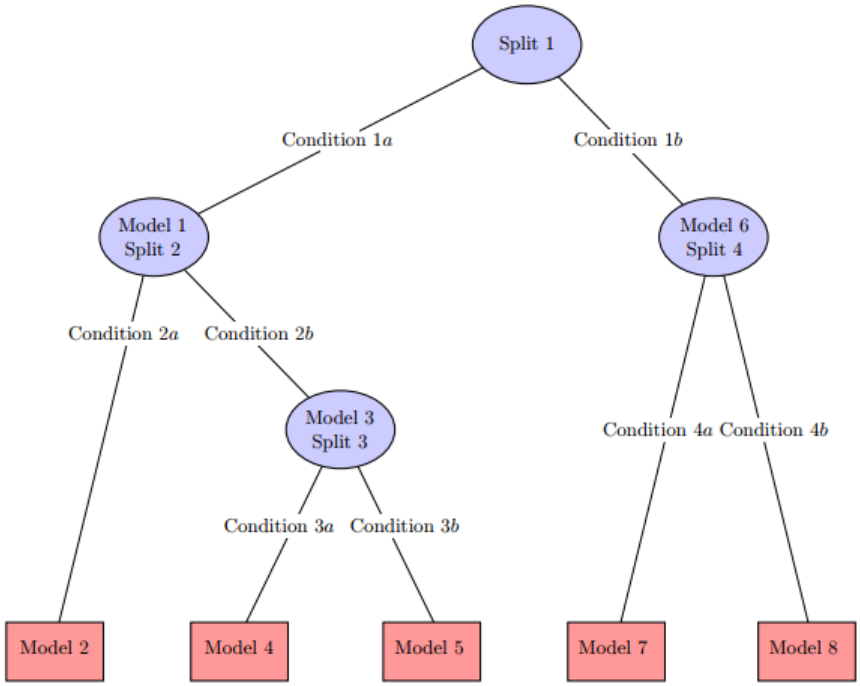

Figure 2.14: Model Trees

Tree–based models consist of one or more nested if-then statements for the predictors that partition the data. Within these partitions, a model is used to predict the outcome. For example, a very simple tree could be defined as [45]:

```
if >= 1.7 then
|   if X2 >= 202.1 then Y = 1.3
|   else Y = 5.6
else Y = 2.5
```

Figure 2.15: Regression Trees

Notice that the if-then statements generated by a tree define a unique route to one terminal node for any sample. A rule is a set of if-then conditions (possibly created by a tree) that have been collapsed into independent conditions. For the example above, there would be three rules [45]:

```
if X1 >= 1.7 & X2 >= 202.1 then Y = 1.3
if X1 >= 1.7 & X2 <  202.1 then Y = 5.6
if X1 <  1.7               then Y = 2.5
```

Figure 2.16: Regression Trees - rules

The next step here is to substitute the results from the leaves with linear regression models, creating thus Rule - Based Model Trees where each terminal node has rules that drive us to use a specific linear regression formula. This is very useful because we have predictive models with high accuracy, stability and ease of interpretation that they map non-linear relationships quite well. They are adaptable at solving any kind of problem at hand. For example the following figure shows the result from the library Cubist, it creates 2 rules and each rule ends with a linear regression function [45]:

```
Rule 1: [58 cases, mean 4.980517, range 4.477121 to 5.523746, est err 0.152796]

if zip in {z95621, z95626, z95660, z95673, z95683, z9581, z95817, z95820,
           z95822, z95823, z95824, z95826, z95827, z95828, z95832, z95838
           z95841, z95842, z95843} and
   beds <= 2 then
outcome = 7.944631 + 0.323 beds + 4e-05 sqft + 0.03 longitude

Rule 2: [126 cases, mean 5.200466, range 4.788875 to 5.662758, est err 0.090147]

if zip in {z95626, z95660, z95683, z95815, z95823, z95824, z95827, z95832,
           z95838, z95841} and
   beds > 2 then
outcome = 8.524561 - 0.056 beds + 0.000342 sqft + 0.03 longitude + 0.003 baths
```

Figure 2.17: Rule-Based Regression Model Tree

2.4 Summary of Meteorological Forecasting Factors

It is well known that electric load demand is directly affected by the weather condition. If weather changes dramatically then heating and air conditioning appliances are used, resulting in increased demand for electricity. Scientific Papers describe the use of meteorological features such as temperature, humidity, wind speed, cloudiness, dew point, wind direction, temperature-humidity index (THI) wind chill index (WCI) for electric load forecasting. From all of the previously mentioned features, The feature that is used the most is the temperature. Temperature can be used in hourly basis, or even the temperature difference with the previous hour and the current, the minimum and the maximum temperature, and even the average value of it over a few hours [5, 7–9, 14, 15, 22, 26–33].

2.5 Summary of Calendar Forecasting Factors

There are many calendar factors that affect the short-term load forecasting. Months are one of them. Months can be categorized from 4 seasons, to multiple groups of seasons, for example into 7 types to distinguish the transitions between two adjacent seasons: winter (Dec 1 – Feb 15), late winter – early spring (Feb 16 – 15 March), Spring (16 march – May 31), late spring – early summer (Jun 1 – Jun 30), Summer (Jul 1 – Sep 15), late summer – early fall (Sep 16 – 28 Oct), and Fall, (29 Oct – Nov 30) [35].

The energy consumption behavior in different days of a week may be different, which is due to many reasons. For instance, office buildings may be closed during weekend, which causes fewer loads than those of work days. People may get up late in the morning during weekend, which shifts the morning peak one or two hours later than a normal work day so seven groups of days of a week can be created. In daily basis, hours can be grouped from rush hours and non-rush hours.

Forecasting the loads of special days, e.g., holidays (national and religious), has been a challenging issue in STLF, not only because the load profiles may vary from different holidays and the same holiday of different years, but also due to the limited data history.

2.6 Evaluating Forecast Accuracy

this section will describe some metrics to evaluate prediction accuracy as a benchmark for the performance of each model being developed. We assume that Y_i indicates the i_{th} observation by the variable to predict and $\widehat{Y}_i$ the prediction of the variable Y_i.

The forecast error is simply $e_i = Y_i - \widehat{Y}_i$, which is on the same scale as the data. Accuracy measures that are based on e_i are therefore scale-dependent and cannot be used to make comparisons between series that are on different scales.

The two most commonly used scale-dependent measures are based on the absolute errors or squared errors:

- **Mean absolute error:** MAE = $mean(|e_i|)$
- **Root mean squared error:** RMSE = $\sqrt{mean(e_i^2)}$

When comparing forecast methods on a single data set, the MAE is popular as it is easy to understand and compute.

The percentage error is given by $p_i = 100e_i/y_i$. Percentage errors have the advantage of being scale-independent, and so are frequently used to compare forecast performance between different data sets. The most commonly used measure is:

- **Mean absolute percentage error:** MAPE = $mean(|p_i|)$

2.7 Data Analysis and Machine Learning Development Platforms

In this section various Data Analysis and Machine Learning platforms that can be used for this master thesis goal to develop models for short-term load forecasting in Greek electric grid will be showed.

2.7.1 R Data Analysis and Machine Learning Platform

Figure 2.18: R Language

R is an open source high-level programming language and a statistics and graphics development environment supported by the R Foundation for Statistical Computing. R language is widely used by statisticians, data miners, data analysts and its popularity has increased over the years. R is a GNU project, its source code is mainly written in C and Fortran. H R is available for free under the GNU General Public License. It has a command line environment as well as a variety of graphical environments (e.g. R-Studio). R's predecessor is named S, and S initially developed by Bell Labs. R was created by

Ross Ihaka and Robert Gentleman of the University of Auckland, New Zealand, and is currently its main development team. Named by the original letter of the names of its creators. Its development began in 1992 with an initial release in 1995 and the stable version appeared in 2000. R and its libraries implement a wide range of statistical and graphical functions, including linear and nonlinear models, classical statistical controls, time series analysis, engineering classroom classification models, clustering, etc. It can easily be expanded because it is an open source project and her community is particularly active in contributing to improving her libraries. [46]

2.7.2 Python Data Analysis and Machine Learning Platform

Figure 2.19: Python Language

Python is a widely known high-level language, created by Guido van Rossum and released in 1991. The concept is to simplify and legibility of the code by using only the blank character/s to avoid code track block using brackets as implemented with other programming languages. This language has also adopted a special way of writing to allow developers to write code with fewer lines than C ++ and Java. It makes these features a specific language suitable for easy-to-read code suitable for small or large scale and scalable. [50] Python provides several libraries for data analysis, some of which are:

- **Scikit-learn:**

Figure 2.20: Scikit-learn Data analysis platform

Scikit-learn is a free python library for data science and machine learning and its main features are classification, regression, clustering, gradient boosting, k-means, DBSCAN and is designed to work with other Python libraries (NumPy and SciPy) for scientific and numerical calculations mainly [51].

- **Pandas:**

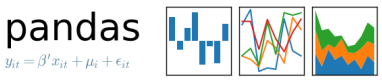

Figure 2.21: Pandas Data analysis library

Pandas is a Python library for data management and analysis. Specifically, it provides suitable data structures and methods for managing multidimensional arrays and time series. It is compatible with the BSD license and is a free library software to use [52].

- **Statsmodel:**

Figure 2.22: Statsmodel Data analysis library

Statsmodel is a Python library that allows users to explore data, evaluate different statistical models, apply statistical controls. Is a suitable library for descriptive statistics, and for descriptive graphs. It is a subset of the superset of SciPy library-based libraries [53].

- **Anaconda:**

Figure 2.23: Statsmodel Data analysis library

Anaconda was created and promoted by continium, a suite of open source libraries from both Python and R, and is suitable for large-scale data analysis and processing, forecasting and computing. Its purpose is to collect the appropriate libraries for the above functions for the most simplified management of large data. Anaconda is in line with the Open Data Science innovation, based on open source communities for better and more quality data analysis, using over 720 data analysis libraries to visualize them, for mechanical learning, deep learning and resource management large data. Includes the appropriate libraries for Apache Hadoop and Spark to handle large python data. It also includes suitable libraries for deep learning mechanical learning and image processing such as theano, tensorflow, neon, h2o, keras etc. [54]

2.7.3 Rapidminer Data Analysis and Machine Learning Platform

Figure 2.24: Rapidminer Data analysis software

Rapidminer is a commercial software and is a complete platform for engineering learning, deep learning, text analysis, and predictive analytics. Use for commercial, business, research and educational use. It includes all the machine learning processes that are used for preparation of data, visualization of data, verification of results and optimization of data Analysis. It is available in a free version but with limited capabilities of 1 logical cpu for computational resources and 10000 training examples. The commercial version provides all the software features at an initial price of $2500. The initial release was named YALE (Yet Another Learning Environment) and was developed in 2001 by Ralf Klinkenberg, Ingo Mierswa, and Simon Fischer in the Artificial Intelligence Lab at the Dortmund Polytechnic University. In 2006 its development continued with the release name Rapid-l, from Ingo Mierswa and Ralf Klinkenberg. Rapidminer has up to now enormous downloads for its free software version and over 250,000 commercial users including BMW, Cisco, GE and Samsung [55].

2.7.4 Orange Data Analysis and Machine Learning Platform

Figure 2.25: Orange Data analysis software

Orange is an open source software for data visualization, machine learning and knowledge mining from the data. Its main feature is visual programming, front-end exploratory data analysis and data visualization. Its development began at the University of Ljubljana in 1997. It is currently in version 3.4. It is based on Python, Cython, C ++, C. It is cross platform software and follows the GNU General Public License. [56]

2.7.5 KNIME Data Analysis and Machine Learning Platform

Figure 2.26: Knime Data analysis software

Knime (The Konstanz Information Miner) is an open source software for data analysis, data visualization, data mining and mechanical learning. Its graphical environment is developed so that a user can choose the tools - functions - transformations he wants to implement and with which order he wishes to do the data processing without writing any code. This provides user - friendly usability for analyzing the datasets. Knime is mostly used for pharmaceutical research, customer data analysis (CRM), business intelligence for forecasting future assets, data analysis and visualization models [57]

2.7.6 Weka Data Analysis and Machine Learning Platform

Figure 2.27: Weka Data analysis software

Weka is a free software for machine learning and data analysis written in Java. It was developed by the University of Waikato in New Zealand and it complies with the GNU General Public License. It includes a set of visualization tools for data analysis, prediction models, pre-processing data, and a user-friendly interface environment. Historically, Weka developed from the University of Waikato in New Zealand in 1993 from programming language Tcl / Tk and Makefiles. In 1997, the development of Weka from Java began. In 2005, the software was awarded from the SIGKDD Data Mining and Knowledge Discovery Service Award. And in 2006, Pentaho Corporation has attributed Weka certification as a software for business intelligence, data mining and predictive analytics. By 2011, it has received 2,487,213 downloads from sourceforge.net [47]

2.7.7 SPSS Data Analysis and Machine Learning Platform

Figure 2.28: IBM SPSS, Data Analysis Software

SPSS, is an IBM software for statistical analysis. Originally developed by SPSS Inc. and in 2009 it was acquired by IBM. SPSS mainly is used in areas such as marketing, health sciences, social sciences, text and emotion analysis, knowledge mining, data analysis. The main functions of SPSS are statistics with variable frequencies, cross tabulation, ratio statistics, and Bivariate Statistics. Correlations, ANOVA, Non-parametric controls. Predictions with multiple linear regrowth. Analysis of factors, clustering analysis, discreet analysis as well as visualization of data - examples. [48]

2.7.8 Microsoft Excel Data Analysis and Machine Learning Platform

Figure 2.29: Excel Spread-sheet Application

Microsoft Excel is a spreadsheet software and is a cross platform software for Windows, macOS and Android and one of the software packages of the Microsoft Office software suite. Its features are various computations of spreadsheet data, graphical depictions - representations, script programming with Visual Basic for Applications on spreadsheet data. It is a well-known spreadsheet software. Microsoft Excel has a key feature in its spreadsheets. This is the use of cells for data management and the use of spreadsheets cells and columns for data management, computational operations and graphical representations. Microsoft Excel includes a large set of statistical and econometric functions for displaying graphs, histograms, etc. Its first version was in 1987, and the latest version is called Excel 2016 where the stable and final version of the software was released in April 2016. An easy-to-use Microsoft Excel tool from 2010, 2013, 2016 is Power Pivot, with the main goal of developing linear regression models for spreadsheet data, and mainly uses DAX (Data Analysis Expressions) as the primary language money within the tool to create the models [49].

Chapter 3

Scientific Review and Commercial Software

In this chapter various techniques from statistics to machine learning found from the scientific literature will be presented.

3.1 Statistical Approaches

3.1.1 Regression Analysis

The short-term electricity load demand forecasting with regression analysis was analyzed a lot from the scientific literature. For example the use of regression analysis for the Pacific Gas and Electric Company for the 24-hour electricity load demand was used [3]. The fundamental techniques that was used in this approach was the least square method and the method of weighted least squares to minimize the consequences from the extreme values. The use of temperature, calendar variables such as holidays, weekends etc. was also used as independent variables. Subsequent research into the energy consumption forecast for the same company as an extension of the original survey was made with the purpose of using meteorological data such as humidity [4].

3.1.2 Time Series Approach

Time series and ARIMA models for short-term STLF forecasts methods are commonly found in the scientific literature [5]. ARIMA models are used with normalized data of electricity demand and temperature before the prediction is produced. ARIMA models along with the Box N 'Jenkins time series were used for STLF in research [6]. In this study, the non-linear relationship of temperature to the demand for electrical energy has emerged, and the third degree of polynomial function has been used for multiple linear regression. Also in a different survey for a short-term forecast of electricity consumption published in the IEEE, the use of multiple linear regression with ARIMA was used, normalization of data in combination with the use of human behavior in calendar taking into account holidays, weekends etc [7].

3.2 Artificial Intelligence - Machine Learning Approaches

3.2.1 Neural Networks

Historically, since 1990, have been published surveys for the use of artificial neural networks in short-term load forecasts. An initial survey is [8]. In this study, an algorithm of artificial neural networks was proposed in conjunction with time series and regression analysis to address the non-linear relationship of demand for electricity consumption with various meteorological features and was applied to the data

of Puget Sound Power and Light Company's from November 1988 to January 1989. Artificial neural networks were also used in the above mentioned survey of Pacific Gas and Electric Company [3] and compared the model of neural networks with that developed by regression analysis in the same set of training data from 1986 to 1990. It was shown that artificial neural networks exhibit greater prediction accuracy than regression analysis. The neural network model used was a Back Propagation model.

An extension of the previous research was based on the use of the radial basis function neural network (RBFN) [9], used for the same company and the same set of data. The new algorithm showed better performance and accuracy than the BPN model of neural networks and the use of calendar variables for holidays, holidays, etc. For Greece, there have been remarkable surveys using clusters of neural networks for short-term forecasting of electricity consumption from various research [10], and other similar investigations using artificial intelligence techniques [11–13]

3.2.2 Fuzzy Logic - Fuzzy Neural Networks

Since the late 1980s and early 1990s, many researchers have designed intelligent systems to predict power consumption using fuzzy logic. For example in 1988 the publication of the study of the prediction of electricity consumption for the Old Dominion Electric Cooperative Virginia [14, 15]. Another similar research was developed in 1990 for the Taiwan power system [16, 17]. And especially after 1990, many studies have begun to design fuzzy logic based systems (automatic fuzzy inference systems) [18–20]. Finally, since 1990 there has been a boom in the use of neural networks of fuzzy logic.

3.2.3 Deep Learning

Recent scientific / research publications have been made on the prediction of electricity consumption using modern methods of deep learning. For example, in the following study, it was predicted electricity consumption and compared the results to time series regression and analysis techniques, also it was presented the benefits of the deep learning neural networks in depth compared to traditional statistical methods and time series analysis [24]. In another recently published research in 2016, it was used the latest technology of artificial intelligence in deep learning and implemented a deep neural network model for short-term electric power consumption. Meteorological features were also used within the prediction model and performance comparisons with other prediction models. The results of the their developed deep learning neural network compared with a double seasonal Holt-Winters model, and an autoregressive integrated moving average (ARIMA) model. The metric was compared with MAPE, and deep neural network showed the fewest and lowest predictive errors compared to the rest of the models. [25].

3.2.4 SVM

Support Vector Machines are widely used as models for predicting electricity consumption. For example, in 2001 the European Intelligent Technology Organization (EUNITE) organized a competition for medium-term forecasting of electricity consumption with the best model being SVM-based [21]. SVM for MTLF was used to predict power consumption in Istanbul electric grid using average temperature data per day, calendar factors and daytime electricity load demand records [22].

3.2.5 Random Forests

Browsing the scientific literature various papers can be found for Random Forest approaches in load forecasting. A Random Forest combined with a feature selection method based on the generalized minimum redundancy and maximum relevance was proposed to improve the accuracy of short-term load forecasting (STLF) [26]. Another paper using Random Forests is the following study that proposes using Random Forest models for short-term electric load forecasting. This study uses an ensemble learning method that generates many regression trees and aggregates their results. The model operates on patterns of the time series seasonal cycles which simplify the forecasting problem especially when a time series

exhibits nonstationarity, heteroscedasticity, trend and multiple seasonal cycles. The main advantages of the model are its ability to generalization, built-in cross-validation and low sensitivity to parameter values. The proposed forecasting model was applied to historical load data in Poland and its performance is compared with some alternative models such as CART, ARIMA, exponential smoothing and neural networks [27].

3.2.6 Rule-Based

Rule-Based are also being applied to the field of Short-term load forecasting and the scientific literature verifies it. For example a study was made to investigate the possibility to apply symbolic data mining methods to the problem of load prediction. By employing rules that is used for extraction of classification or prediction from data. When new data is coming, their current rules are applied to the data and their weights are composed by the inference mechanism to the resulting weight of a given prediction. The presented approach is applied to the problem of short-term electric load forecasting [28].

Another Rule based approach is discussed from the next survey from IEEE [29]. In that survey the formulation of rules and their application in load forecasting was discussed. The use of rules was made for the classification of the load forecast parameters into weather-sensitive and nonweather-sensitive categories was described. The rationale underlying the development of rules for both the one-day and seven-day forecast is presented. This exercise leads to the identification and estimation of parameters relating to electric load such as weather variables, day types, and seasons. Moreover they provided a self-learning process is described which shows how rules governing the electric utility load can be updated [29].

3.2.7 k-Nearest Neighbors

k-Nearest Neighbors models can be found in the scientific literature too. For example kNN algorithms was used to construct a hybrid model which comprised from a Wavelet Denoising-Extreme Learning Machine and k-Nearest Neighbor Regression. This model combines k-Nearest Neighbor (kNN) and Extreme Learning Machine (ELM) based on a wavelet denoising technique was proposed for short-term load forecasting. The proposed hybrid model decomposed the time series into a low frequency-associated main signal at first, then used kNN to determine the independent and dependent variables from the low-frequency signal. The model ELM was used to get the non-linear relationship between these variables to get the final prediction result for the electric load. Compared with three other models, the Extreme Learning Machine optimized by k-Nearest Neighbor Regression (EKM), the Wavelet Denoising-Extreme Learning Machine (WKM) and Wavelet Denoising-Back Propagation Neural Network optimized by k-Nearest Neighbor Regression (WNNM). The proposed model EKM in that paper showed that can improve the accuracy efficiently. The model EKM was applied to electricity loads from New South Wales powerhouse of Australia, in order to predict electric demand for that region [30].

In another paper the k-Nearest Neighbors models was used in order for the day ahead load prediction. That paper proposes a k-nearest neighbor (kNN)-based model for predicting the next-day's load with only limited temperature forecasts, namely minimum and maximum temperature of a day, as the forecasting input. The proposed model consists of three individual kNN models which have different neighboring rules. The three are combined together by tuned weighting factors for a final forecasting output. The proposed model is tested on the Australian National Electricity Market data, showing reasonably high accuracy. It can be used as an alternative tool for day-ahead load forecasting when only limited temperature information is available [31].

3.2.8 XGBoost

Furthermore, XGBoost models can be found in the scientific literature to be used in electric load forecasting. For example in the following study, the use of XGBoost models was used in order to be built a short-term power load extreme gradient boosting (XGBoost) model with multi-information fusion by

using the gradient boosting algorithm. [32]. In another study XGBoost models was used in order to compete in Kaggle competition in 2012. The competition involved a hierarchical load forecasting problem for a US utility with 20 geographical zones. The available data consisted of the hourly loads for the 20 zones and hourly temperatures from 11 weather stations, for four and a half years. For each zone, the hourly electricity load for nine different weeks needed to be predicted without having the location of zones or stations. They used separate models for each hourly period, with component-wise gradient boosting to estimate each model using univariate penalized regression splines as base learners. The models allow for the electricity demand to change with time-of-year, day-of-week, time-of-day, and on public holidays, with the main predictors being current and past temperatures as well as past demand. The team ranked 5th among 105 teams [33].

3.3 Commercial Software

This section presents commercial software for electric load forecasting.

3.3.1 ETAP's load forecasting software

The load forecasting software from ETAP [1] provides tools for short term load forecasting. It is an ideal tool for industrial users as well as utilities to reliably and accurately forecast future short term loading in the system. Its main characteristics are:

ETAP's Load Forecasting provides many beneficial tools and utilities such as adaptive Bus Load Forecasting, real-Time load trending, predict loading up to seven days ahead, forecasting multiple load areas per individual meters, user-adjustable weather variables and load profiles, revise forecasts based on loading and weather conditions, import and export historical forecast data and finally unlimited forecast views.

ETAP Load Forecasting utilizes sophisticated algorithms to correlate multiple input variables such as predicted weather conditions along with historical data such as meter point loading and weather conditions to construct a forecast model.

3.3.2 Aiolos Forecasting Studio

The commercial software Aiolos Forecast Studio [2] is a cloud based software, user-friendly and is used for forecasting. Aiolos Forecast Studio is developed by the Scandinavian company Vitec to manage forecasts such as electricity, gas hydro power. Electricity demand is forecasted with the highly accurate Aiolos model that handles +10K forecast series at the same time. Gas forecasting, Gas demand is forecasted with the highly accurate Cilia model that handles +10K forecast series at the same time. For district heating/cooling plants the model Nestor is developed. A mixture of parameters from weather forecasts and the production plants makes this model essential for any production plant. For Hydro Power demand forecasting. The Aiolos software contains the model Achelous, which is adaptive and gives you precise forecasts on the water flow to hydro plants. For Wind power demand, Aiolos software can be connected to wind turbines and its model;h the Zephyros model provides production forecasts for entire countries if needed.

Dealing the electric load demand forecast can be made with high accuracy, used historical past data, combined with adaptive models in conjunction with the input data, providing daily forecasts.

3.3.3 Electric Load Forecasting Using Artificial Neural Networks

The commercial software Electric Load Forecasting Using Artificial Neural Networks by GMDH Shell [3] uses neural networks for reliable and accurate forecasts for electric load power demand. Is uses past

[1] https://etap.com/product/load-forecasting-software

[2] Aiolos Forecast Studio

[3] Electric Load Forecasting Using Artificial Neural Networks

historical data provided by the user. The way that it works is through minimizing the error of prediction vs the real values. When the error criterion is satisfied then the final model is given to the user.

This particular software is user-friendly and works only in Windows systems. It provides descriptive statistics tools. It uses calendar features (Weekends, Holidays, National Days etc). It can utilize ODBC Data Base connections in order to retrieve information from Data Bases. The horizon of prediction is from short-term to long-term forecasting. Moreover is has the ability to decompose time series data in its components (trend, seasonality, cyclic).

3.3.4 Escoware demand-forecasting

The Software ESCOWare® demand-forecasting solution [4] uses advanced and modern machine learning techniques for accurate electric load forecast. The ESCOWare® DFS solution utilizes proprietary, state of the art weather normalized algorithms that help to mitigate risk by accurately forecasting customer usage, allowing for predictable margins and profitability. ESCOWare® DFS is a cloud-based software solution designed specifically for the complexities of retail energy suppliers, and utilities.

3.3.5 SAS® Energy Forecasting

The commercial software SAS Energy Forecasting [5] provides reliable electric load forecasting predictions. It has the utilities to predict from short-term to long-term horizon. It is designed in order to predict the electric load forecasting providing reliable and accurate results regardless of the size of the input meteorological data that are given to be used as independent variables for prediction.

3.3.6 Statgraphics

Statgraphics [6] is a commercial statistical software for statistical analysis. It is developed in 1980 from Dr. Neil Polhemus, professor in statistics from Princeton University. The current edition is called Statgraphics Centurion XVII is realised in 2014.

In terms of data analysis for electricity consumption, Statgraphics uses data structures in the form of time series. Statgraphics has a plethora of time series visualization tools. It has controls for finding random time series, analyzing their oscillations, and finding any voltage, seasonality, periodicity, etc. In addition, it has mechanisms for descriptive data statistics, and more specifically for time series consumption data, it has autocorrelation functions to interpret how past data affect future ones. Also other tools such as the periodogram are useful for analyzing time series oscillations and whether they maintain specific frequencies at regular intervals. It also has linear and non-linear smoothing techniques for time series and especially those that hold enough noise to reliably find the time series trend. It also has time series degradation techniques in their basic components to make it possible to find seasonal patterns and produce data adapted to the seasonality-periodicity component. Finally, Statgraphics has tools for predicting time series, exploiting the past time series data to be predicted by various processes such as random walk, mobile media, trend models, simple, linear, polynomial models, exponential smoothing. ARIMA parametric models. Statgraphics also has the ability to suggest that the user selects the model itself, which minimizes the prediction error of the time series.

[4] Demand Forecasting Solution
[5] SAS® Energy Forecasting
[6] statgraphics

Chapter 4

Application - ML Models for STLF in Greek Electric Grid

4.1 Goals - Development Platform - Software Environment

For this master thesis the R Language was selected to be used as development software language. The R 3.4.1 version with R-Studio as IDE platform and R-Markdown, R-Shiny for visualizations. The main goal for the master Thesis is the development of an application in R in order to construct Machine Learning Models to analyze the load demand in the Greek Electricity Network and make predictions with less error than the existed one that OoEM provides. (ΛΑΓΗΕ in Greek). The code for this master thesis can be found in the github repo[1] and the Shiny Visualizations can be found at the IPTO-ML shinyapp [2]

4.2 Datasets

The Datasets come from 3 origins.

1. The IPTO's Loads
2. The OoEM's predictions
3. The Meteorological Data

The Datasets span from 2010-10-01 to 2017-04-30

4.2.1 IPTO Loads

The dataset for the Electricity Load Demand for Greece can be found from IPTO (ΑΔΜΗΕ in Greek)[3]. IPTO stands for Independent Power Transmission Operator in Greece. Is the Operator of the Greek Electricity Transmission System, IPTO's mission is to ensure Greece's electricity supply in a safe, efficient and reliable manner while promoting the development of competition in the Greek electricity market and guaranteeing the non-discriminatory treatment of System users. Their main goal is to be one of the most efficient electricity transmission system operators in Europe and build value for all stakeholders within a framework of sustainable development while respecting man, the environment and benefiting System Users and society as a whole. IPTO's current organizational structure is the following:

[1] github repo
[2] IPTO-ML shinyapp
[3] IPTO official site

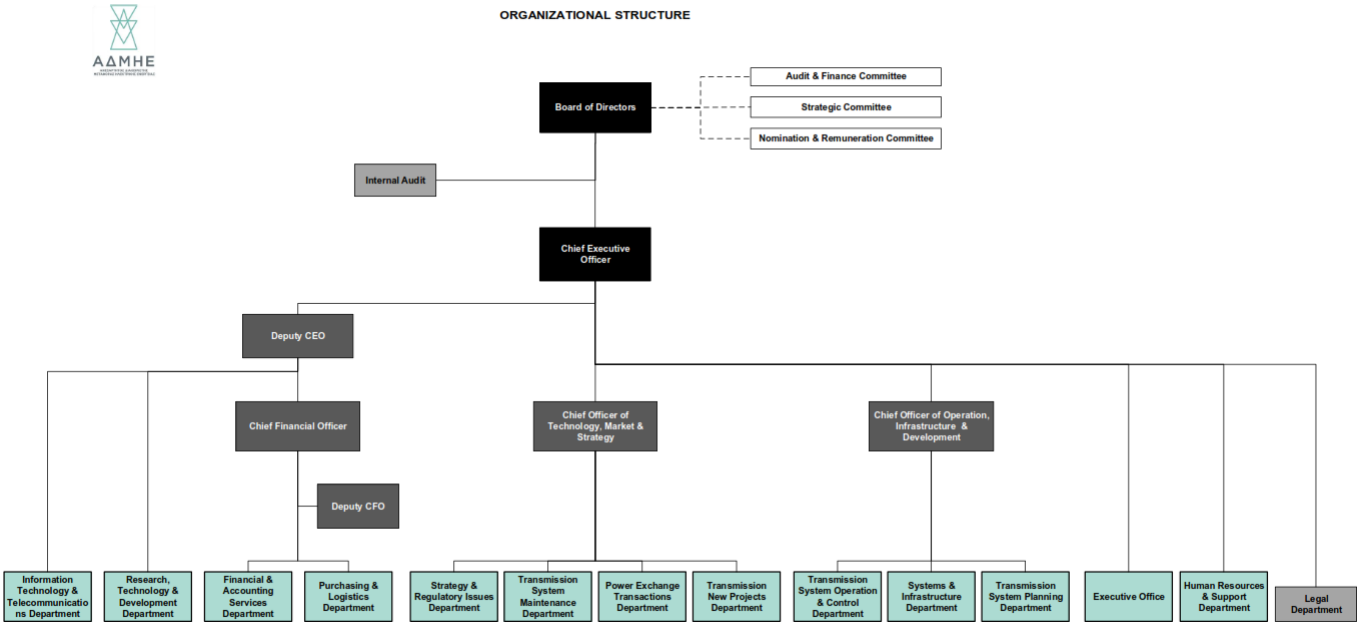

Figure 4.1: IPTO organizational Structure

The IPTO's Loads Demand Datasets are in hourly format and span from 2010-10-01 to 2017-04-30 roughly 6.5 years of data.

4.2.2 OoEM Predictions

OoEM (ΛΑΓΗΕ in Greek) [4] stands for Operator of Electricity Market in Greece. He applies the rules for the operation of the Electricity Market in accordance with the provisions of Law 4001/2011 and its delegated acts and in particular the Daily Energy Planning. It predicts the electricity load demand for the next day for the whole Greek electricity grid network. This task is very important because OoEM needs the know how many loads in MW will be needed in the network per hour because the electric power units in the Greek region must be prepared and be ready to produce the right amount of energy for the needs of consumers and if energy purchase from the neighboring countries is needed. The main goal for this master thesis is to provide with day ahead predictions with accuracy better than those from OoEM.

4.2.3 Meteorological Data

Meteorological Data as features are needed in order to buid models and make predictions. Meteorological Data was fetched from [5]. The Dark Sky Company specializes in weather forecasting and visualization. It provides an API with REST call which gives meteorological data for a geographical location in JSON format. Meteorological Data are taken from the two biggest cities of Greece, Athens and Thessaloníki. The concept behind sticking with these two cities is from a paper which dealt the same issue to predict the short-term electric load forecasting from A. Adamakos [10], but the difference here is the use of more meteorological features than the paper used. The date period of the meteorological data are the same of the IPTO's loads and OoEM predictions.

4.3 Electric Load Demand - Summary Statistics

A typical electric load demand plot from a weekday is the following:

[4] OoEM official site
[5] darksky.net

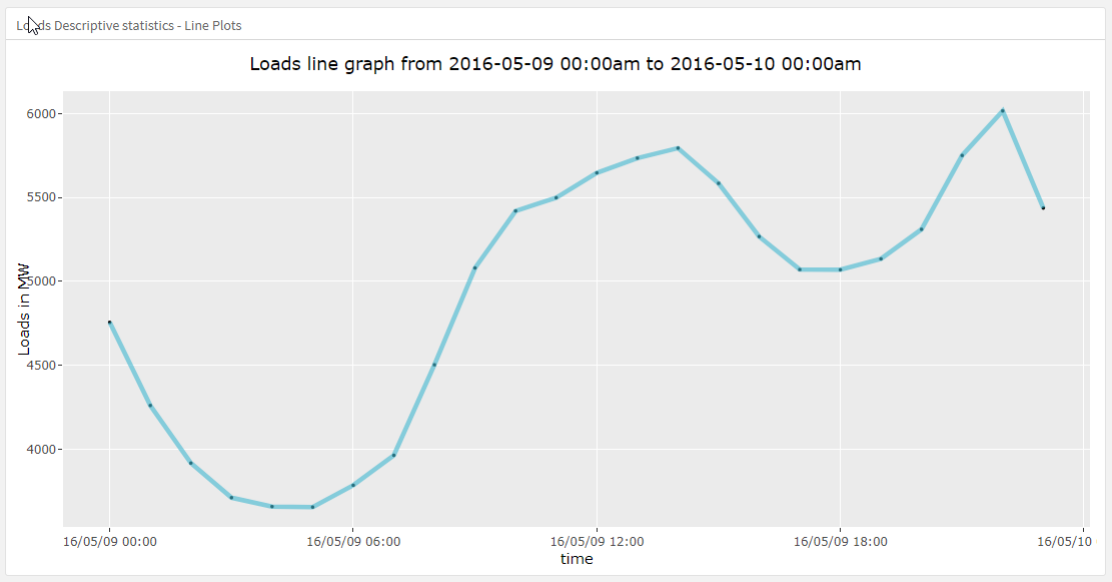

Figure 4.2: Load Demand Line Plot

We can see that from 00:00 - 7:00 (non-rush hours) load demand is decreasing, from 7:00 - 14:00 (rush hours) load are increasing, from 14:00 - 18:00 (non-rush hours) load is declining, from 18:00 - 21:00 again loads are rising and finally from 21:00 - 00:00 loads are decreasing. This is a typical load demand during weekdays. The load demand trend and cycle may vary during weekends and Holidays due to the fact that electricity load demand depends on human behavior and during weekends and holidays the human behavior is changes thus the demand alters.

4.4 Meteorological Features

The meteorological Features taken from Dark Sky API for the cities of Athens and Thessaloníki in hourly basis are the following:

- **Summary**
  Categorical variable, summary of current weather conditions.
- **Icon**
  Categorical variable, more detailed edition of current weather conditions.
- **Temperature**
  Continuous variable, the temperature in the selected location, measured in Celsius degrees.
- **Dew Point**
  Continuous variable, the dew point in the selected location, measured in Celsius degrees.
- **Humidity**
  Continuous variable, the humidity in the air in the selected location, measured with percentage.
- **Wind Speed**
  Continuous variable, the wind speed the air flows in the selected location measured in meters per second.
- **Wind Bearing**
  Continuous variable, the bearing of the wind flows in the selected location measured in degrees.
- **Visibility**
  Continuous variable, the visibility in the selected location in kilometers.

- **Cloud cover**
  Continuous variable, measuring how much the sky is covered by clouds in the selected location measured with percentage.
- **UV index**
  Continuous variable, measuring sun's radiation during the day in the selected location.

All meteorological Features measured in S.I. units.

4.5 Preprocessing - Cleaning the Data

After fetching all the datasets, one vital step is the preprocessing. Downloading the meteorological data from DarkSky API comes with many missing values. A missing value can be a problem if we leave it untouched because we can have a problematic case and may be omitted during predictions. Thus all the missing values was filled by:

1. fill the missing values by the mean of the adjacent by hour values corresponded to the same feature.
2. if the previous adjacent values are missing values too, fill the missing values by the adjacent by day values corresponded to the same feature.
3. but if and the previous values are missing values too, copy just one of the adjacent values to the missing value corresponded to the same feature.

4.6 Calendar Features

Calendar features were constructed from the dates of the meteorological dataset. The reason behind is that specific hours and specific days affect the human behavior beyond meteorological factors. Hence the following features were constructed:

- **isRushHour**, categorical variable, if a hour is rush hour or not
- **isHoliday**, categorical variable, if that day is holiday (Greek National and Religious Holidays).
- **isWeekend**, categorical variable, if that day is weekend.

4.7 Joining the Datasets and Constructing the Cases

IPTO's Loads and Dark Sky meteorological data are two different datasets that needs to be joined together in order to construct the cases for training. The common field was the time. So an inner join in the common field which is "date - time" was performed in order to join the two datasets. The final case again based on A. Adamakos paper [10] but with more meteorological and Calendar. The final case format with its features is described in the following table 4.1:

Table 4.1: Case Format and its Features

Case Format and its Features	
1	Weather forecast in Athens for the forecasted day(d).
2	Weather forecast in Thessaloniki for the forecasted day(d).
3	Calendar Features for the forecasted day(d)
4	Weather data in Athens for preceding day(d-1) of the forecasted day(d).
5	Weather data in Thessaloniki for preceding day(d-1) of the forecasted day(d).

Table 4.1 continued from previous page

	Case and its Features
6	Calendar Features for preceding day(d-1)
7	hourly load measurements of the second preceding day (d-2) of the forecasted day(d).
8	hourly load measurements of the second preceding day (d-3) of the forecasted day(d).
9	Current Loads for the forecasted day (d)

4.8 Adding Noise

Due to the fact that we have historical past observed data from the meteorological data given by Dark Sky API, we wanted to add a bit of noise to the past data where weather forecast is used (columns 1 and 2 from table 4.1), because sometimes weather observation is not accurate enough, and this stems from the fact that Dark Sky API provides meteorological data for a location / city with the aid of nearby weather stations and if weather stations do not provide weather information for that location, then Dark Sky uses its own ML models to predict the meteorological information for that location. So trying to be as precise and close to the real weather conditions for a location we added a moderate small noise (+/- 0.5 from uniform distribution) to the extracted meteorological data taken from Dark Sky API.

4.9 Feature Selection

In order to make predictions several experiments were made in order to minimize the error of prediction. One technique was the feature selection. The feature selection algorithm that was used is named "Boruta" [6] [7]. Boruta is a feature selection algorithm. Precisely, it works as a wrapper algorithm around Random Forest. This package derive its name from a demon in Slavic mythology who dwelled in pine forests. Feature selection is a crucial step in predictive modeling. This technique achieves supreme importance when a data set comprised of several variables given for model building. Boruta can be the algorithm of choice to deal with such data sets. Particularly when one is interested in understanding the mechanisms related to the variable of interest, rather than just building a black box predictive model with good prediction accuracy.

Boruta follows the following steps in during feature selection [58]:

1. Firstly, it adds randomness to the given data set by creating shuffled copies of all features (which are called shadow features).

2. Then, it trains a random forest classifier on the extended data set and applies a feature importance measure (the default is Mean Decrease Accuracy) to evaluate the importance of each feature where higher means more important. At every iteration, it checks whether a real feature has a higher importance than the best of its shadow features (i.e. whether the feature has a higher Z score than the maximum Z score of its shadow features) and constantly removes features which are deemed highly unimportant.

3. Finally, the algorithm stops either when all features gets confirmed or rejected or it reaches a specified limit of random forest runs.

In this master Thesis, the cases has a great amount of features, *1484 features for 2394 cases*. This great amount of features was created because huge variety of information from meteorological and Calendar features was used, hence feature selection is integral to the experiments. To predict the short-term load forecasting, the loads were separated by hour, so 24 target variables were created: $load.0, load.1, ..., load.24$. For each of these loads the Boruta feature selection algorithm was executed in order to find the relevant

[6] Boruta package
[7] Boruta tutorial

features per target variable. The results are from 1484 features are reduced from 180 - 220 per target variable.

Because the amount of The selected features per load.x target variable is enormous. The intercept of selected features for all load.x will be presented in table format at the end, at the appendix of tables.

4.10 Machine Learning - Experiments

During Machine Learning experimentation 6 machine Learning algorithms were used:

1. SVM: from package e1071
2. k-Nearest Neighbors: from package: FNN
3. Random Forest: from package: randomForest
4. Neural Networks: from package: RSNNS
5. XGBoost: from package: XGBoost
6. Model Trees: from package: Cubist

The load.x target variable were separated by hour, so 24 target variables were created: $load.0, load.1, ..., load.24$. Thus 24 models per machine learning algorithm were made in order to predict each of the target-variable.

The main parameters for the experiments are the following:

- **Evaluation Metrics**: Mean absolute percent error, mape
- **Parameters**: Full Features, Feature Selection, Default ML Models parameters and Grid Search
- **Classifier Selection**: Ensemble models per target variable (if applicable)

Four Experiments were performed in order to minimize the error of prediction and increase the prediction accuracy. The type of experiments are the following:

1. **Experiment #1:**
   Run the machine learning algorithms with their default parameters, and cases as is without feature selection but with full features. The dataset partition was the following: train-set: the first 5.5 years and test-set: the last 1 year
2. **Experiment #2:**
   Run the machine learning algorithms with their default parameters, and cases with feature selection. The dataset partition was the following: train-Set: the first 5.5 years and test-set: the last 1 year
3. **Experiment #3:**
   Run the machine learning algorithms and find the best tuning parameters from grid search, and cases as is without feature selection but with full features. The dataset partition was the following: train-set: the first 4.5 years, validation-set: the following 1 year and test-set: the last 1 year.
4. **Experiment #4:**
   Run the machine learning algorithms and find the best tuning parameters from grid search, and cases with feature selection. The dataset partition was the following: train-set: the first 4.5 years, validation-set: the following 1 year and test-set: the last 1 year.

4.11 Forecasting Experiment with Full Features and Default Parameters

This is the first experiment, in this experiment we ran the machine learning algorithms with their default parameters, and cases as is without feature selection but with full features. The dataset partitioned as follows: train-set: the first 5.5 years and test-set: the last 1 year.

24 Models where trained per target Variable and per machine learning algorithm as many are the hours in the day (24 hours of the day). The mape prediction error accuracy results applied to the Test-Set are presented in the following table, with green color are highlighted the mape prediction error per model and target variable with the least value. 4.2:

Table 4.2: mape evaluation performance from various models with default model parameters for the 24-hour based target variables

mape evaluation performance from various models with default model parameters for the 24-hour based target variables						
target variable / models	**svm**	**knn**	**random Forest**	**nn**	**xgboost**	**Model Trees**
load.0	2.76%	3.57%	2.91%	4.67%	2.23%	2.03%
load.1	2.63%	3.48%	2.7%	4.61%	2.43%	1.92%
load.2	2.57%	3.44%	2.58%	4.28%	2.22%	2.1%
load.3	2.7%	3.51%	2.62%	4.22%	2.53%	2.15%
load.4	2.73%	3.74%	2.72%	4.04%	2.6%	2.36%
load.5	2.71%	3.71%	2.71%	3.87%	2.56%	2.54%
load.6	3.01%	3.86%	2.92%	4.16%	2.94%	2.87%
load.7	3.78%	4.97%	3.27%	3.67%	3.3%	2.73%
load.8	5.58%	7.17%	3.74%	11.5%	3.83%	3%
load.9	6.08%	7.86%	4.22%	6.83%	3.65%	3.47%
load.10	5.72%	7.52%	3.77%	5.94%	3.3%	3.17%
load.11	5.08%	6.58%	3.68%	4.35%	3.04%	2.85%
load.12	4.53%	5.87%	3.37%	4.35%	3.07%	2.54%
load.13	4.47%	5.81%	3.21%	4.43%	2.72%	2.45%
load.14	4.68%	6.18%	3.33%	5.01%	3.12%	2.62%
load.15	5.23%	6.9%	3.77%	5.43%	2.98%	2.85%
load.16	5.63%	7.11%	4.07%	5.73%	4.07%	3.06%
load.17	5.46%	7.05%	4.14%	5.86%	3.93%	3.46%
load.18	5.67%	6.85%	4.29%	6.29%	3.83%	3.48%
load.19	5.74%	6.36%	3.99%	12.7%	3.95%	2.95%
load.20	6.03%	5.81%	3.59%	14.1%	3.35%	2.39%
load.21	4.29%	5.19%	3.06%	11.1%	2.63%	2.26%
load.22	3.57%	4.71%	3.15%	3.73%	2.74%	2.26%
load.23	3.01%	4.61%	3.42%	4.94%	2.87%	2.27%

The following table shows the mean mape from models for all the target variables per machine learning algorithm 4.3:

Table 4.3: mean mape for the 24 models per machine learning algorithm with default parameters and full features

mean mape for the 24 models per machine learning algorithm with default parameters and full features						
	svm	**knn**	**random forest**	**nn**	**xgboost**	**Model Trees**

Table 4.3 continued from previous page

mean mape for the 24 models per machine learning algorithm with default parameters and full features						
mean mape	4.31%	5.49%	3.38%	6.07%	3.07%	2.65%

Some Remarks

- Model Trees with Cubist library, performed almost perfectly, with 2.65% error. Only Model Trees and no other ML algorithm has the best accuracy per target variable.
- On the second and third place takes the XGBoost with 3.07% and RandomForest with 3.38% further experimentation will increase their accuracy.
- Having as minimum mean mape prediction error 2.65% from all models from Model Trees and the load prediction error from OoEM is 2.53%, this experiment decreases the predictive accuracy by **-4.74%** than that from OoEM's. The following table describes the ensemble of models per target variable and shows the prediction error comparison from OoEM existed model to Model Trees 4.4.

Table 4.4: classifier selection from target variable ensemble, mape evaluation and performance evaluation for experiment #1

classifier selection from target variable ensemble, mape evaluation and performance evaluation for experiment #1	
Model Trees mean mape:	2.65%
OoEM existed model mape:	2.53%
prediction error improvement:	-4.74%

4.12 Forecasting Experiment with Feature Selection and Default Parameters

This is the second experiment, in this experiment we ran the machine learning algorithms with their default parameters, and cases with feature selection. The dataset partition is the following: train-set: the first 5.5 years and test-set: the last 1 year.

24 Models where trained per target Variable and per machine learning algorithm as many are the hours in the day (24 hours of the day). The mape prediction error accuracy results applied to the Test-Set are presented in the following table 4.5. With green color are highlighted the mape prediction error per model and target variable with the least value.

Table 4.5: mape evaluation accuracy from various models with tuned parameters for the 24-hour based target variables

mape evaluation accuracy from various models with tuned parameters for the 24-hour based target variables						
target variable / models	**svm**	**knn**	**random Forest**	**nn**	**xgboost**	**Model Trees**
load.0	2.17%	3.58%	2.64%	3.4%	2.11%	2.05%
load.1	2.17%	3.49%	2.49%	3.8%	2.34%	2.23%
load.2	1.96%	3.43%	2.3%	3.68%	2.29%	1.99%
load.3	1.93%	3.48%	2.28%	3.72%	2.32%	2.12%

Table 4.5 continued from previous page

mape evaluation accuracy from various models with tuned parameters for the 24-hour based target variables						
load.4	2.16%	3.73%	2.45%	3.42%	2.52%	2.45%
load.5	2.13%	3.7%	2.49%	3.39%	2.7%	2.4%
load.6	2.29%	3.87%	2.64%	3.12%	2.68%	2.67%
load.7	2.8%	4.98%	2.95%	3.25%	2.87%	2.76%
load.8	3.82%	7.19%	3.34%	4.73%	3.59%	2.75%
load.9	4.33%	7.89%	3.7%	4.29%	3.61%	3.37%
load.10	3.96%	7.53%	3.5%	5.95%	3.23%	2.85%
load.11	3.44%	6.5%	3.27%	5.22%	3.04%	2.99%
load.12	2.96%	5.91%	3.05%	4.8%	2.85%	2.37%
load.13	2.94%	5.81%	2.95%	4.71%	2.68%	2.5%
load.14	3.07%	6.14%	2.99%	5.63%	2.95%	2.56%
load.15	3.59%	6.93%	3.33%	6.45%	3.43%	2.96%
load.16	3.69%	7.12%	3.69%	5.81%	3.78%	3.08%
load.17	3.98%	6.98%	3.76%	5.94%	3.77%	3.35%
load.18	4.19%	6.81%	3.87%	6.38%	3.67%	3.22%
load.19	4.45%	6.42%	3.5%	6.4%	3.41%	2.64%
load.20	4.18%	5.75%	3.36%	8.33%	3.64%	2.52%
load.21	2.87%	5.19%	2.83%	7.56%	2.57%	2.15%
load.22	2.38%	4.71%	2.86%	4.6%	2.64%	2.19%
load.23	2.39%	4.58%	3.02%	4.89%	2.49%	2.31%

The following table 4.6 shows the mean mape evaluation from models for all the target variables per machine learning algorithm:

Table 4.6: mean mape for the 24 models per machine learning algorithms with default parameters and feature selection

mean mape for the 24 models per machine learning algorithms with default parameters and feature selection						
	svm	**knn**	**random forest**	**nn**	**xgboost**	**Model Trees**
mean mape	3.07%	5.48%	3.05%	4.97%	2.96%	2.6%

Some Remarks

- Model Trees with Cubist library, performed again almost perfectly, it went from 2.65% prediction error to 2.6% prediction error.
- However now with feature selection SVM started to perform better than the previous experiment, its prediction error decreased from 4.31% to 3.07% and in some target variables performs better than Model Trees.
- XGBoost although is increase its accuracy than the previous experiment with feature selection, it comes second in mean accuracy but it overruns SVM in most of target variables.
- one significant tactic here is to combine different models from target variable to target variable based on the minimum mape. Here for the load target variable from 00:00 to 6:00, SVM shows greater performance that Model Trees, hence we can combine the mean mape from these SVM models and the rest of target variables from Model Trees. If we do this we get even better accuracy

with prediction error **2.55%** which is better from the mean mape from Model Trees alone which is 2.6%. The load prediction error from OoEM is 2.53%, thus this experiment again decreases the predictive performance by **-0.7%** than that from OoEM's. The following table describes the ensemble of models per target variable and shows the performance evaluation from OoEM existed model 4.7.

Table 4.7: classifier selection from target variable ensemble, mape evaluation and performance evaluation for experiment #2

classifier selection from target variable ensemble, mape evaluation and performance evaluation for experiment #2	
SVM, Model Trees mean mape:	2.55%
OoEM existed model mape:	2.53%
prediction error improvement	-0.7%

4.13 Forecasting Experiment with Full Features and Grid Search

This is the third experiment, in this experiment we ran the machine learning algorithms with tuned parameters from grid search, and cases as is with full features. The dataset partition was the following: train-set: the first 4.5 years, validation-set: the following 1 year and test-set: the last 1 year.

The grid search tuning parameters per machine learning algorithm are being presenting in the following table 4.8:

Table 4.8: Grid Search - Tuning Parameters

Grid Search - Tuning Parameters						
Parameters / Models	***SVM***	***K-Nearest Neighbors***	***Random Forest***	***Neural Networks***	***XGBoost***	***Model Trees***
parameter #1	Radial Kernel	KNN algorithm (kd-tree, brute, cover-tree)	number of trees	Resilient Back-propagation	Tree Booster	unbiased, (should unbiased rules be used?)
parameter #2	Gamma paremeter	Number of Neighbors	mtry, (Number of variables randomly sampled as candidates at each split)	Learning Rate	eta (control the learning rate)	committes = 1, how many committee models (boosting iterations) should be used? For fairness to other models, commitee = 1
parameter #3	Cost parameter			Number of Neurons at the hidden Layer	max_depth (maximum depth of a tree)	

Table 4.8 continued from previous page

Grid Search - Tuning Parameters						
parameter #4				maxit, (maximum of iterations to learn)	nrounds, (the max number of iterations)	

To select the best tuning parameters per machine learning algorithm, each combination of tuned parameters was evaluated to the Validation Set. Those who had the smaller mape were selected to train the final model and compare the accuracy results to the Test Set. The best tuned parameters per machine learning algorithm and per target variable with full features can be found at the appendix of tables.

24 Models where trained per target Variable and per machine learning algorithm as many are the hours in the day (24 hours of the day). The mape prediction error accuracy results applied to the Test-Set are presented in the following table, with green color are highlighted the mape prediction error per model and target variable with the least value. 4.9:

Table 4.9: mape evaluation performance from various models with tuned parameters and full features for the 24-hour based target variables

mape evaluation performance from various models with tuned parameters and full features for the 24-hour based target variables						
target variable/ models	**svm**	**knn**	**random Forest**	**nn**	**xgboost**	**Model Trees**
load.0	2.53%	3.55%	2.93%	2.89%	2.12%	2.06%
load.1	2.48%	3.39%	2.61%	3.64%	2.05%	1.92%
load.2	2.41%	3.23%	2.61%	2.89%	1.97%	2.12%
load.3	2.51%	3.32%	2.52%	2.9%	2.14%	2.16%
load.4	2.6%	3.55%	2.84%	3.24%	2.3%	2.32%
load.5	2.53%	3.62%	2.79%	3.16%	2.16%	2.5%
load.6	2.69%	3.85%	3.03%	2.91%	2.31%	2.87%
load.7	3.07%	4.99%	3.27%	3.47%	2.81%	2.73%
load.8	4.19%	7.23%	4.01%	4.17%	3.48%	3%
load.9	4.56%	8.01%	4.28%	4.06%	3.68%	3.47%
load.10	4.74%	7.5%	3.85%	5.15%	3.37%	3.17%
load.11	4.38%	6.36%	3.55%	4.22%	2.84%	2.85%
load.12	3.95%	5.53%	3.52%	4.42%	2.68%	2.54%
load.13	3.99%	5.45%	3.34%	3.9%	2.63%	2.45%
load.14	4.15%	5.89%	3.37%	4.61%	2.66%	2.62%
load.15	4.59%	6.66%	3.9%	4.83%	2.78%	2.85%
load.16	4.97%	7.03%	4.16%	5.7%	3.22%	3.06%
load.17	4.91%	6.86%	4.12%	4.97%	3.11%	3.46%
load.18	4.92%	6.75%	4.45%	5.42%	3.21%	3.48%
load.19	5.03%	6.24%	3.84%	6.37%	3.26%	2.98%
load.20	4.93%	5.63%	3.64%	5.06%	3.26%	2.45%
load.21	3.73%	5.03%	3.21%	4.31%	2.8%	2.26%
load.22	3.15%	4.52%	3.35%	3.85%	2.52%	2.37%
load.23	2.96%	4.39%	3.28%	3.25%	2.41%	2.27%

The following table 4.10 shows the mean mape for all the target variables per machine learning algorithm:

Table 4.10: mean mape for the 24 models per machine learning algorithms with tuned parameters and full features

mean mape for the 24 models per machine learning algorithms with tuned parameters and full features						
	svm	**knn**	**random forest**	**nn**	**xgboost**	**Model Trees**
mean mape	3.74%	5.35%	3.43%	4.14%	2.73%	2.66%

Some Remarks

- Again Model Trees has overall the best mean accuracy with 2.66% prediction error.
- XGBoost got even better better accuracy than before with prediction error 2.73%. On the other hand SVM accuracy became even worse than before. KNN can not match the in accuracy the other ML algorithms.
- Even though Model Trees has the best mean accuracy from the other five machine learning algorithms, there are some specific target variables which XGBoost models shows better performance. If we combine models per target variable and mix Model Trees and XGBoost per target variable by mape minimum prediction error we get even small mean prediction error **2.58%**. Even though the mean mape combination of XGBoost and Model Trees gives us 2.58% which is a bit better than only Model Trees 2.60%, however 2.58% is worst than the previous combination experiment which was 2.55%. This may imply that combining the two tactics, feature selection and grid search may lead us to even better results. The load prediction error from OoEM is 2.53%, thus this experiment again decreased the predictive performance by **-1.97%** and again is worst than the prediction error from OoEM. The following table describes the ensemble of models per target variable and shows the performance evaluation from OoEM existed model 4.11.

Table 4.11: classifier selection from target variable ensemble, mape evaluation and performance evaluation for experiment #3

classifier selection from target variable ensemble, mape evaluation and performance evaluation for experiment #3	
XGBoost, Model Trees mean mape:	2.58%
OoEM existed model mape:	2.53%
prediction error improvement:	-1.97%

4.14 Forecasting Experiment with Feature Selection and Grid Search

This is the forth and last experiment, in this experiment we ran the machine learning algorithms with tuned parameters from grid search, and cases as is with feature selection. The dataset partition was the following: train-set: the first 4.5 years, validation-set: the following 1 year and test-set: the last 1 year.

The grid search tuning parameters per machine learning algorithm are being presenting in the following table 4.12:

Table 4.12 continued from previous page

Grid Search - Tuning Parameters

Table 4.12: Grid Search - Tuning Parameters

Grid Search - Tuning Parameters						
Parameters / Models	***SVM***	***K-Nearest Neighbors***	***Random Forest***	***Neural Networks***	***XGBoost***	***Model Trees***
parameter #1	Radial Kernel	KNN algorithm (kd-tree, brute, cover-tree)	number of trees	Resilient Back-propagation	Tree Booster	unbiased, (should unbiased rules be used?)
parameter #2	Gamma paremeter	Number of Neighbors	mtry, (Number of variables randomly sampled as candidates at each split)	Learning Rate	eta (control the learning rate)	committes = 1, how many committee models (boosting iterations) should be used? For fairness to other models, committee = 1
parameter #3	Cost parameter			Number of Neurons at the hidden Layer	max_depth (maximum depth of a tree)	
parameter #4				maxit, (maximum of iterations to learn)	nrounds, (the max number of iterations)	

To select the best tuning parameters per machine learning algorithm, each combination of tuned parameters was evaluated to the Validation Set. Those who had the smaller mape were selected to train the final model and compare the accuracy results to the Test Set. The best tuned parameters per machine learning algorithm and per target variable with feature selection can be found at the appendix of tables.

24 Models where trained per target Variable and per machine learning algorithm as many are the hours in the day (24 hours of the day). The mape prediction error accuracy results are presented in the following table, with green color are highlighted the mape prediction error per model and target variable with the least value. 4.13:

Table 4.13: mape evaluation performance from various models with tuned model parameters and feature selection for the 24-hour based target variables

mape evaluation performance from various models with tuned model parameters and feature selection for the 24-hour based target variables						
target variable/ models	**svm**	**knn**	**random Forest**	**nn**	**xgboost**	**Model Trees**
load.0	1.78%	3.57%	2.73%	2.46%	2.01%	2.05%
load.1	1.86%	3.4%	2.53%	2.59%	1.94%	2.23%
load.2	1.77%	3.22%	2.33%	2.57%	2.05%	1.99%

Table 4.13 continued from previous page

mape evaluation performance from various models with tuned model parameters and feature selection for the 24-hour based target variables						
load.3	1.79%	3.27%	2.29%	2.55%	2.04%	2.13%
load.4	2.05%	3.52%	2.49%	2.58%	2.23%	2.43%
load.5	1.94%	3.61%	2.47%	2.99%	2.1%	2.37%
load.6	2.08%	3.83%	2.64%	2.67%	2.22%	2.67%
load.7	2.3%	4.97%	2.99%	2.59%	2.63%	2.76%
load.8	3.45%	7.21%	3.49%	4.69%	3%	2.75%
load.9	3.45%	8%	3.8%	4.47%	3.49%	3.37%
load.10	2.93%	7.49%	3.64%	11.3%	3.08%	2.85%
load.11	2.69%	6.43%	3.28%	5.32%	2.74%	2.99%
load.12	2.16%	5.61%	3.04%	4.12%	2.59%	2.37%
load.13	2.43%	5.44%	3.02%	3.83%	2.5%	2.5%
load.14	2.56%	5.84%	2.99%	6.28%	2.54%	2.56%
load.15	3.04%	6.62%	3.4%	5.12%	2.78%	2.96%
load.16	3.19%	7.02%	3.73%	4.77%	3.05%	3.08%
load.17	3.91%	6.76%	3.75%	4.84%	3.46%	3.35%
load.18	3.62%	6.75%	3.84%	4.47%	3.26%	3.22%
load.19	3.86%	6.24%	3.55%	4.51%	3.3%	2.64%
load.20	3.71%	5.68%	3.3%	3.97%	3.25%	2.55%
load.21	2.67%	5.06%	2.91%	4.14%	2.79%	2.15%
load.22	1.95%	4.52%	3.01%	3.86%	2.38%	2.19%
load.23	1.94%	4.36%	3.17%	3.21%	2.13%	2.31%

The following table shows the mean mape for all the target variables per machine learning algorithm 4.14:

Table 4.14: mean mape for the 24 models per machine learning algorithms with tuned parameters and feature selection

mean mape for the 24 models per machine learning algorithms with tuned parameters and feature selection						
	svm	**knn**	**random forest**	**nn**	**xgboost**	**Model Trees**
mean mape	2.62%	5.35%	3.09%	4.16%	2.64%	2.60%

Some Remarks

- Again Model Trees has overall the best mean accuracy with 2.60% prediction error.
- XGBoost got the an improved performance than every other experiment with prediction error 2.64%.
- SVM also also got an improved performance than every other experiment with prediction error 2.62%.
- Again if we combine and mix models from target variable to target variable by minimum mape we get the best mean combined (ensemble) mape ever which is **2.41%**. The load prediction error from OoEM is 2.53%, thus this experiment increase the predictive performance by **+4.74%**. The last experiment increased the performance accuracy by decreasing the prediction error more than that from OoEM's prediction error. The following table describes the ensemble of models per target variable and shows the performance evaluation from OoEM existed model 4.15.

Table 4.15: classifier selection from target variable ensemble, mape evaluation and performance evaluation for experiment #4

classifier selection from target variable ensemble, mape evaluation and performance evaluation for experiment #4	
SVM, XGBoost, Model Trees mean mape:	2.41%
OoEM existed model mape:	2.53%
prediction error improvement:	+4.74%

4.15 R-Shiny, R-Markdown Visualizations

To visualize the loads, the meteorological data, out predictions, the OoEM predictions, and display some descriptive statistics a R-Shiny / R-Markdown application was created. The reason behind this is that visualizations give a different perspective by visualizing the results and sometimes can find issues that can not be found from R-Studio command line.

The application is called IPTO ML, is interactive and the user can see for different dates and durations the real load values, the OoEM prediction, this project predictions, descriptive statistics and the performance of predictions in mape evaluation. The application is deployed at shinyapps, and its page is named: IPTO ML.

The initial page is the following:

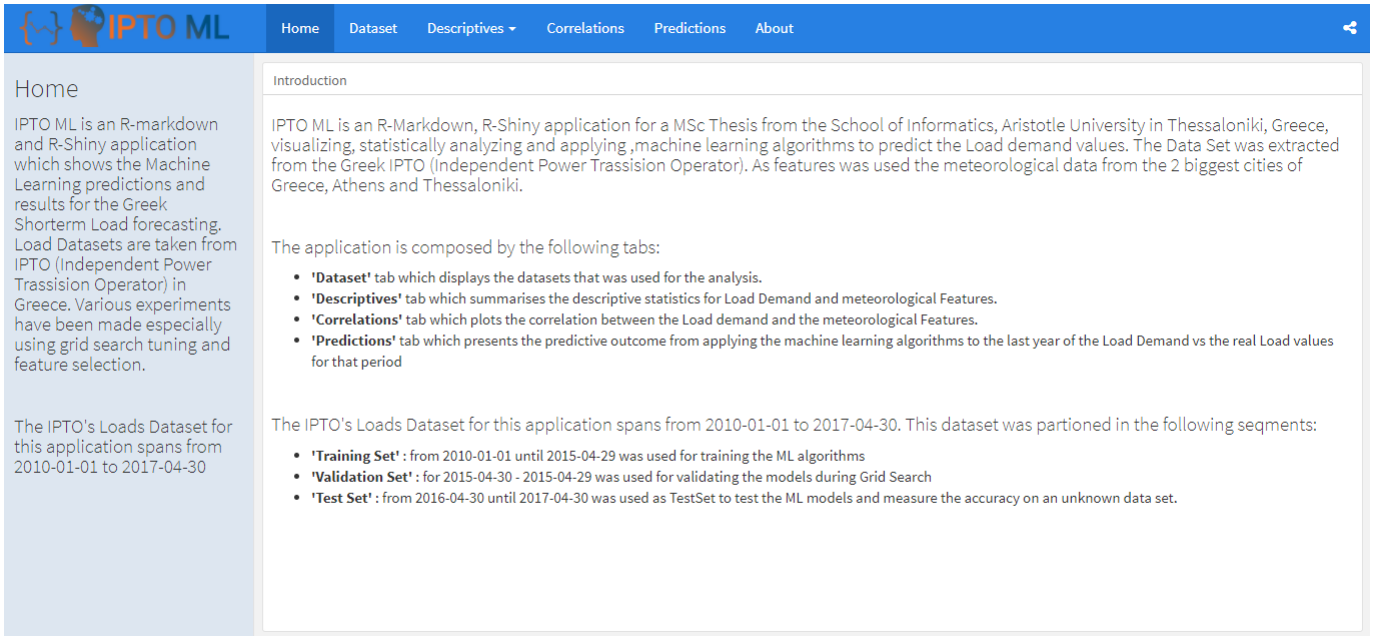

Figure 4.3: IPTO ML first page

The second page presents both loads and meteorological features in table format

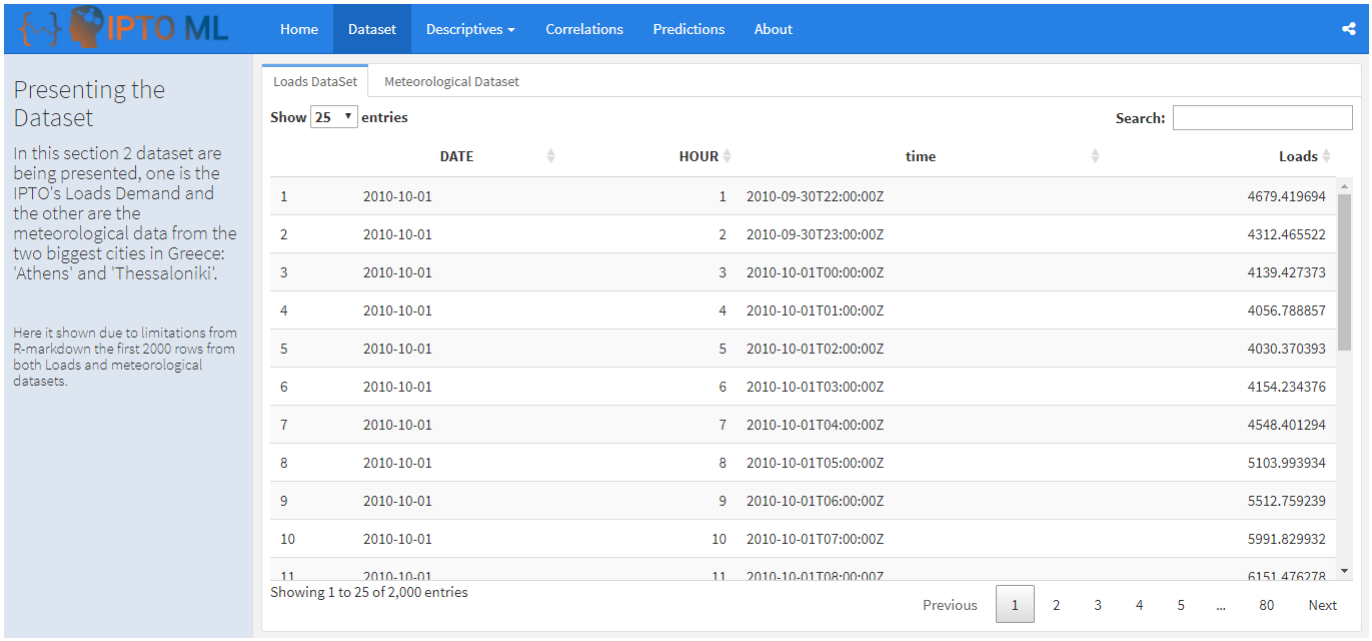

Figure 4.4: IPTO ML second page

The third page shows descriptive statistics for both loads and the meteorological features (histograms and boxplot)

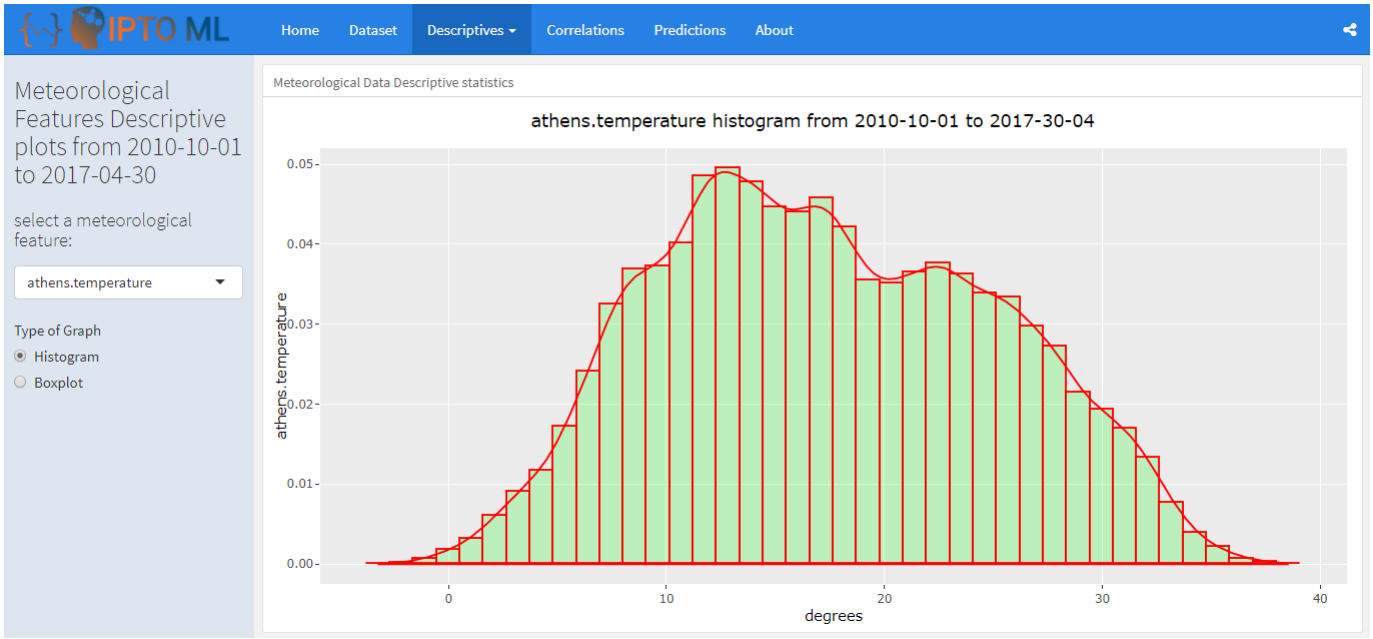

Figure 4.5: IPTO ML third page

The forth page shows the correlation between electricity load demand and meteorological features.

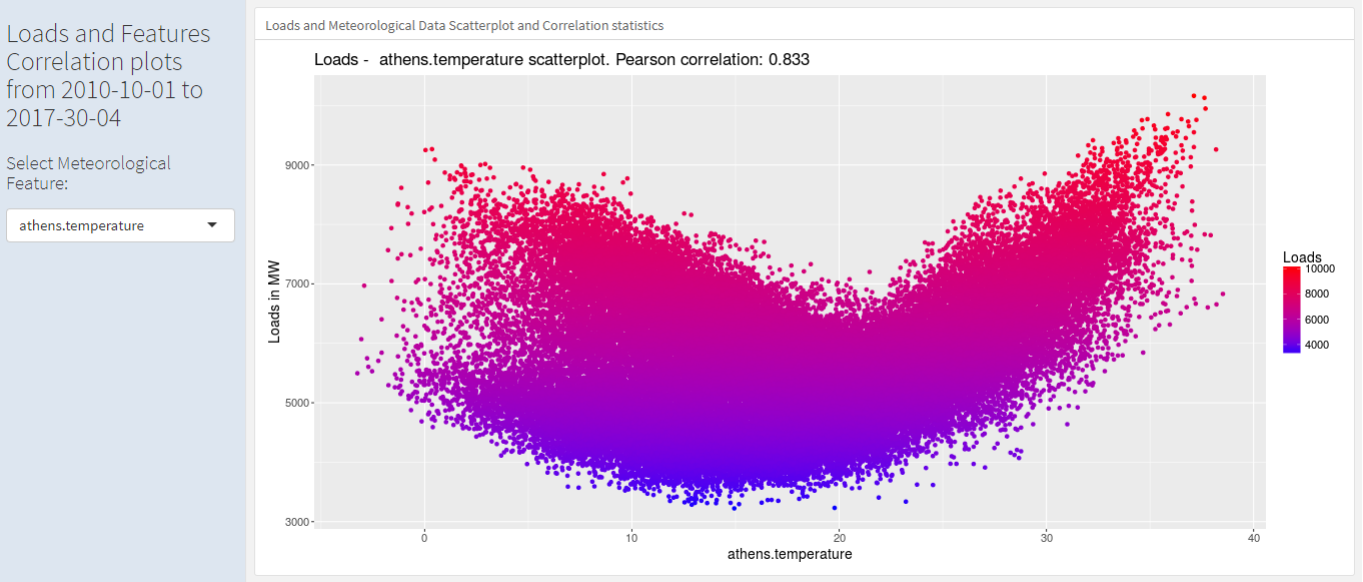

Figure 4.6: IPTO ML forth page

And the fifth page shows the correlation between electricity load demand and meteorological features

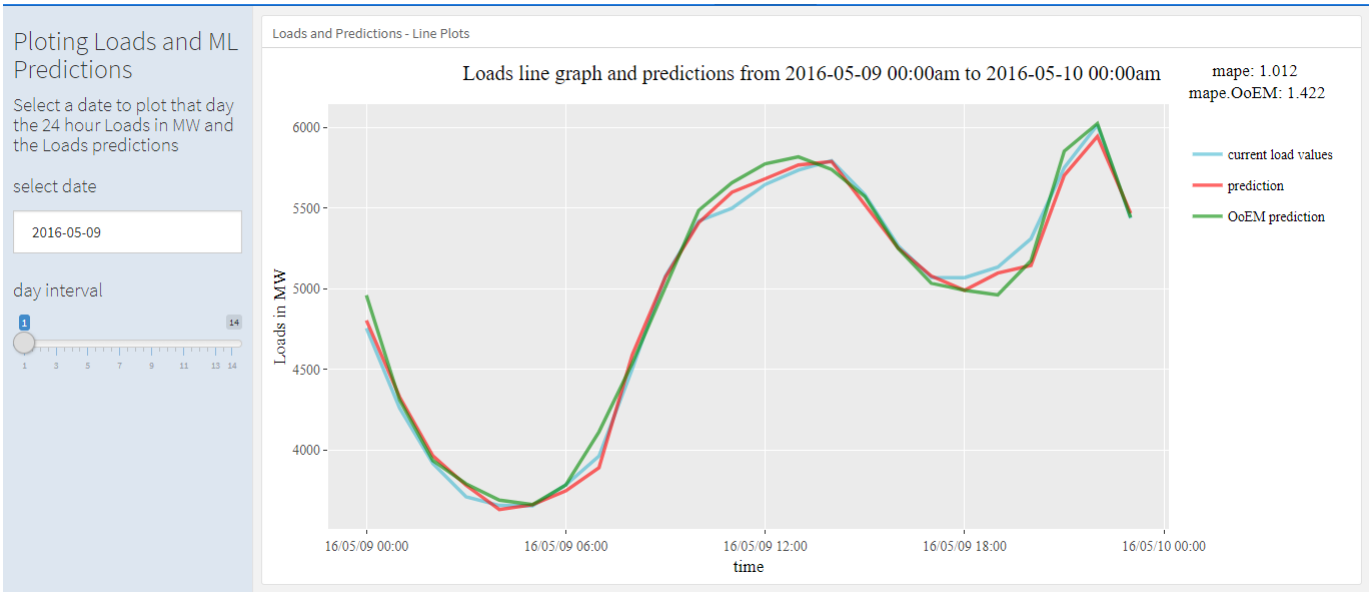

Figure 4.7: IPTO ML fifth page

Chapter 5

Conclusions - Final Remarks

As presented in the initial chapters, electric load forecasting is nowadays a vital process, due to the fact that the increased needs for electrical power from consumers and from our daily life and routine. Electrical companies should take into account this process and constantly improve their models for more accurate forecasts.

As seen in the scientific literature, many machine learning and statistical approaches exist to predict the short-term load forecasting. Applying six of them (SVM, Random Forest, k-Nearest Neighbors, Neural Networks, XGBoost, Model Trees) the prediction error accuracy increased to OoEM's by 4.74% with the best prediction error 2.41% from ensemble of models SVM, XGBoost and Model Trees. This is one interesting part, that combining models by their mape evaluation is that we can increase even more the accuracy of the prediction error from our models in total.

Chapter 6

Future Work

Many techniques have not been developed as seen from the literature. More importantly if they are developed the ensemble - combination of them should give interesting results. Furthermore techniques of multi-target regression can be applied. Moreover to increase the performance because grid search takes a lot of time, spark library for distributed computing or parallel for loops should be introduced. Two cities was used Athens and Thessaloníki and its meteorological and calendar features. What if more cities introduced to the models. Having feature selection makes introduction of more cities easier. Another future idea is to make short-term load forecasting for any future date. Having now pre-trained models we can make predictions for a future date, only new cases with meteorological and calendar values must be constructed for that date. Another idea is to give more liveness to the ipto-ml shinyapp application by fetching “on the fly” all the files from IPTO, OoEM and DarkSky and then pre-process and train new models. Finally the idea of an “Energy analytics” would be a great visualization for load forecasting.

Chapter 7

Appendix of Tables

7.1 Tables of Selected Features per target variable:

In this section the following table 7.1 with the Intercept of all selected Features for all target Variables load.x will be presented.

Table 7.1: Intercept of all selected Features for all target Variables load.x

Intercept of all selected Features for all target Variables load.x	
1	forecast.athens.temperature.0
2	forecast.thessaloniki.temperature.0
3	forecast.thessaloniki.temperature.1
4	forecast.thessaloniki.temperature.2
5	forecast.thessaloniki.temperature.3
6	forecast.athens.temperature.7
7	forecast.athens.temperature.8
8	forecast.thessaloniki.temperature.8
9	forecast.athens.temperature.9
10	forecast.thessaloniki.temperature.9
11	forecast.athens.temperature.10
12	forecast.thessaloniki.temperature.10
13	forecast.athens.temperature.11
14	forecast.thessaloniki.temperature.11
15	forecast.athens.temperature.12
16	forecast.thessaloniki.temperature.12
17	forecast.athens.temperature.13
18	forecast.thessaloniki.temperature.13
19	forecast.athens.temperature.14
20	forecast.thessaloniki.temperature.14
21	forecast.athens.temperature.15
22	forecast.thessaloniki.temperature.15
23	forecast.athens.temperature.16
24	forecast.thessaloniki.temperature.16
25	forecast.athens.temperature.17
26	forecast.thessaloniki.temperature.17
27	forecast.athens.temperature.18
28	forecast.thessaloniki.temperature.18

Table 7.1 continued from previous page

	Intercept of all selected Features for all target Variables load.x
29	forecast.athens.temperature.19
30	forecast.thessaloniki.temperature.19
31	forecast.athens.temperature.20
32	forecast.thessaloniki.temperature.20
33	forecast.athens.temperature.21
34	forecast.thessaloniki.temperature.21
35	forecast.athens.temperature.22
36	forecast.thessaloniki.temperature.22
37	forecast.athens.temperature.23
38	forecast.thessaloniki.temperature.23
39	yesterday.weather.measures.isWeekend.0
40	yesterday.weather.measures.thessaloniki.temperature.0
41	yesterday.weather.measures.thessaloniki.temperature.1
42	yesterday.weather.measures.thessaloniki.temperature.2
43	yesterday.weather.measures.athens.temperature.8
44	yesterday.weather.measures.thessaloniki.temperature.8
45	yesterday.weather.measures.athens.temperature.9
46	yesterday.weather.measures.thessaloniki.temperature.9
47	yesterday.weather.measures.athens.temperature.10
48	yesterday.weather.measures.thessaloniki.temperature.10
49	yesterday.weather.measures.athens.temperature.11
50	yesterday.weather.measures.thessaloniki.temperature.11
51	yesterday.weather.measures.athens.temperature.12
52	yesterday.weather.measures.thessaloniki.temperature.12
53	yesterday.weather.measures.thessaloniki.temperature.13
54	yesterday.weather.measures.athens.temperature.14
55	yesterday.weather.measures.thessaloniki.temperature.14
56	yesterday.weather.measures.athens.temperature.15
57	yesterday.weather.measures.thessaloniki.temperature.15
58	yesterday.weather.measures.athens.temperature.16
59	yesterday.weather.measures.thessaloniki.temperature.16
60	yesterday.weather.measures.athens.temperature.17
61	yesterday.weather.measures.thessaloniki.temperature.17
62	yesterday.weather.measures.athens.temperature.18
63	yesterday.weather.measures.thessaloniki.temperature.18
64	yesterday.weather.measures.athens.temperature.19
65	yesterday.weather.measures.thessaloniki.temperature.19
66	yesterday.weather.measures.athens.temperature.20
67	yesterday.weather.measures.thessaloniki.temperature.20
68	yesterday.weather.measures.athens.temperature.21
69	yesterday.weather.measures.thessaloniki.temperature.21
70	yesterday.weather.measures.athens.temperature.22
71	yesterday.weather.measures.thessaloniki.temperature.22
72	yesterday.weather.measures.thessaloniki.temperature.23
73	two.preceding.days.loads.Loads.0
74	two.preceding.days.loads.Loads.1
75	two.preceding.days.loads.Loads.2

Table 7.1 continued from previous page

Intercept of all selected Features for all target Variables load.x	
76	two.preceding.days.loads.Loads.3
77	two.preceding.days.loads.Loads.4
78	two.preceding.days.loads.Loads.5
79	two.preceding.days.loads.Loads.6
80	two.preceding.days.loads.Loads.7
81	two.preceding.days.loads.Loads.8
82	two.preceding.days.loads.Loads.9
83	two.preceding.days.loads.Loads.10
84	two.preceding.days.loads.Loads.14
85	two.preceding.days.loads.Loads.15
86	two.preceding.days.loads.Loads.16
87	two.preceding.days.loads.Loads.17
88	two.preceding.days.loads.Loads.18
89	two.preceding.days.loads.Loads.19
90	two.preceding.days.loads.Loads.20
91	two.preceding.days.loads.Loads.21
92	two.preceding.days.loads.Loads.22
93	two.preceding.days.loads.Loads.23
94	three.preceding.days.loads.Loads.0
95	three.preceding.days.loads.Loads.1
96	three.preceding.days.loads.Loads.2
97	three.preceding.days.loads.Loads.3
98	three.preceding.days.loads.Loads.4
99	three.preceding.days.loads.Loads.5
100	three.preceding.days.loads.Loads.6
101	three.preceding.days.loads.Loads.16
102	three.preceding.days.loads.Loads.17
103	three.preceding.days.loads.Loads.18
104	three.preceding.days.loads.Loads.19
105	three.preceding.days.loads.Loads.22
106	three.preceding.days.loads.Loads.23

7.2 Tables of Best Tuning Parameters per ML Algorithms and Target Variable with full features

7.2.1 List of Best SVM tuning Parameters per target variable with full features

Table 7.2: svm best tuning parameter with full features

svm best tuning parameter with full features		
target variable / parameters	Gamma	Cost
Load.0	5.00e-05	64
Load.1	5.00e-05	128

Table 7.2 continued from previous page

svm best tuning parameter with full features		
Load.2	5.00e-05	64
Load.3	5.00e-05	32
Load.4	5.00e-05	32
Load.5	5.00e-05	32
Load.6	5.00e-05	32
Load.7	5.00e-05	64
Load.8	5.00e-05	64
Load.9	5.00e-05	128
Load.10	5.00e-05	128
Load.11	5.00e-05	64
Load.12	5.00e-05	64
Load.13	5.00e-05	64
Load.14	5.00e-05	32
Load.15	5.00e-05	64
Load.16	5.00e-05	64
Load.17	5.00e-05	64
Load.18	5.00e-05	128
Load.19	5.00e-05	64
Load.20	5.00e-05	64
Load.21	5.00e-05	64
Load.22	5.00e-05	64
Load.23	5.00e-05	64

7.2.2 List of Best KNN tuning Parameters per target variable with full features

Table 7.3: K-Nearest Neighbors best tuning parameter with full features

K-Nearest Neighbors best tuning parameter with full features		
target variable / parameters	Number of Nearest Neighbors	KNN Algorithm
Load.0	4	kd_tree
Load.1	5	kd_tree
Load.2	7	kd_tree
Load.3	7	kd_tree
Load.4	20	kd_tree
Load.5	20	kd_tree
Load.6	19	kd_tree
Load.7	9	kd_tree
Load.8	9	kd_tree
Load.9	9	kd_tree
Load.10	10	kd_tree
Load.11	12	kd_tree
Load.12	12	kd_tree
Load.13	12	kd_tree
Load.14	15	kd_tree

Table 7.3 continued from previous page

K-Nearest Neighbors best tuning parameter with full features		
Load.15	15	kd_tree
Load.16	15	kd_tree
Load.17	15	kd_tree
Load.18	15	kd_tree
Load.19	15	kd_tree
Load.20	15	kd_tree
Load.21	14	kd_tree
Load.22	14	kd_tree
Load.23	24	kd_tree

7.2.3 List of Best Random Forest tuning Parameters per target variable with full features

Table 7.4: Random Forest best tuning parameter with full features

Random Forest best tuning parameter with full features		
target variable / parameters	number of trees	mtry
Load.0	20	259
Load.1	20	272
Load.2	20	254
Load.3	20	216
Load.4	20	211
Load.5	20	280
Load.6	20	262
Load.7	20	289
Load.8	20	267
Load.9	20	280
Load.10	20	256
Load.11	20	240
Load.12	20	274
Load.13	20	285
Load.14	20	285
Load.15	10	251
Load.16	20	267
Load.17	20	273
Load.18	10	287
Load.19	20	268
Load.20	20	287
Load.21	20	288
Load.22	10	240
Load.23	20	261

7.2.4 List of Best Neural Networks tuning Parameters per target variable with full features

Table 7.5: Neural Networks best tuning parameter with full features

Neural Networks best tuning parameter with full features			
target variable / parameters	learning rate	hidden Layer Neurons	max iteration
Load.0	0.1	10	300
Load.1	0.1	9	200
Load.2	0.2	4	300
Load.3	0.1	8	200
Load.4	0.1	8	200
Load.5	0.1	10	300
Load.6	0.1	8	200
Load.7	0.1	10	200
Load.8	0.2	10	300
Load.9	0.3	4	300
Load.10	0.2	5	200
Load.11	0.2	10	300
Load.12	0.1	9	200
Load.13	0.1	9	300
Load.14	0.1	9	300
Load.15	0.1	9	300
Load.16	0.1	9	200
Load.17	0.1	9	200
Load.18	0.2	6	200
Load.19	0.3	4	200
Load.20	0.3	4	300
Load.21	0.2	6	300
Load.22	0.2	5	200
Load.23	0.1	8	300

7.2.5 List of Best XGBoost tuning Parameters per target variable with full features

Table 7.6: XGBoost best tuning parameter with full features

XGBoost best tuning parameter with full features			
target variable / parameters	eta	depth	round
Load.0	0.1	5	200
Load.1	0.1	5	200
Load.2	0.1	5	200
Load.3	0.1	4	100
Load.4	0.1	4	100

Table 7.6 continued from previous page

XGBoost best tuning parameter with full features			
Load.5	0.1	2	200
Load.6	0.1	2	200
Load.7	0.1	4	100
Load.8	0.1	5	100
Load.9	0.1	3	500
Load.10	0.1	4	100
Load.11	0.1	5	500
Load.12	0.1	3	100
Load.13	0.1	5	500
Load.14	0.1	4	400
Load.15	0.1	5	500
Load.16	0.1	4	400
Load.17	0.1	2	400
Load.18	0.1	3	500
Load.19	0.1	3	500
Load.20	0.1	3	300
Load.21	0.2	2	200
Load.22	0.1	3	400
Load.23	0.1	3	200

7.2.6 List of Best Model Trees tuning Parameters per target variable with full features

Table 7.7: Mode Trees best tuning parameter with full features

Mode Trees best tuning parameter with full features	
target variable / parameters	unbiased
Load.0	1
Load.1	1
Load.2	1
Load.3	1
Load.4	1
Load.5	0
Load.6	0
Load.7	0
Load.8	0
Load.9	0
Load.10	0
Load.11	0
Load.12	0
Load.13	0
Load.14	0
Load.15	0
Load.16	0
Load.17	0

Table 7.7 continued from previous page

Mode Trees best tuning parameter with full features	
Load.18	0
Load.19	1
Load.20	1
Load.21	0
Load.22	1
Load.23	0

7.3 Tables of Best Tuning Parameters per ML Algorithms and Target Variable with Feature Selection

7.3.1 List of Best SVM tuning Parameters per target variable with feature selection

Table 7.8: svm best tuning parameter with feature selection

svm best tuning parameter with feature selection		
target variable / parameters	Gamma	Cost
Load.0	5.00e-04	64
Load.1	5.00e-04	16
Load.2	5.00e-04	32
Load.3	5.00e-04	64
Load.4	5.00e-05	128
Load.5	5.00e-04	16
Load.6	5.00e-04	32
Load.7	5.00e-04	32
Load.8	5.00e-04	512
Load.9	5.00e-04	512
Load.10	5.00e-04	512
Load.11	5.00e-04	512
Load.12	5.00e-04	128
Load.13	5.00e-04	128
Load.14	5.00e-04	512
Load.15	5.00e-04	256
Load.16	5.00e-04	512
Load.17	5.00e-04	512
Load.18	5.00e-04	256
Load.19	5.00e-04	128
Load.20	5.00e-04	128
Load.21	5.00e-04	256
Load.22	5.00e-04	128
Load.23	5.00e-04	64

7.3.2 List of Best K-Nearest Neighbors tuning Parameters per target variable with feature selection

Table 7.9: K-Nearest Neighbors best tuning parameters with feature selection

K-Nearest Neighbors best tuning parameters with feature selection		
target variable / parameters	Number of Nearest Neighbors	KNN Algorithm
Load.0	4	kd_tree
Load.1	5	kd_tree
Load.2	7	kd_tree
Load.3	8	kd_tree
Load.4	20	kd_tree
Load.5	24	kd_tree
Load.6	20	kd_tree
Load.7	9	kd_tree
Load.8	9	kd_tree
Load.9	9	kd_tree
Load.10	9	kd_tree
Load.11	9	kd_tree
Load.12	15	kd_tree
Load.13	12	kd_tree
Load.14	15	kd_tree
Load.15	15	kd_tree
Load.16	15	kd_tree
Load.17	14	kd_tree
Load.18	18	kd_tree
Load.19	21	kd_tree
Load.20	19	kd_tree
Load.21	15	kd_tree
Load.22	14	kd_tree
Load.23	23	kd_tree

7.3.3 List of Best Random Forest tuning Parameters per target variable with feature selection

Table 7.10: Random Forest best tuning parameter with feature Selection

Random Forest best tuning parameter with feature Selection		
target variable / parameters	number of trees	mtry
Load.0	20	50
Load.1	20	66
Load.2	20	60
Load.3	20	53
Load.4	20	61
Load.5	20	65
Load.6	20	63
Load.7	20	62

Table 7.10 continued from previous page

Random Forest best tuning parameter with feature Selection		
Load.8	20	52
Load.9	20	48
Load.10	20	57
Load.11	20	57
Load.12	20	69
Load.13	20	64
Load.14	20	110
Load.15	10	110
Load.16	20	70
Load.17	20	90
Load.18	10	110
Load.19	20	80
Load.20	20	90
Load.21	20	30
Load.22	10	30
Load.23	20	20

7.3.4 List of Best Neural Networks tuning Parameters per target variable with feature selection

Table 7.11: Neural Networks best tuning parameter with feature Selection

Neural Networks best tuning parameter with feature Selection			
target variable / parameters	learning rate	number of neurons in hidden layer	max. iteration
Load.0	0.4	9	700
Load.1	0.2	5	1000
Load.2	0.2	2	1000
Load.3	0.1	6	1000
Load.4	0.3	2	1000
Load.5	0.1	2	700
Load.6	0.4	2	500
Load.7	0.1	4	1000
Load.8	0.2	7	1000
Load.9	0.1	7	800
Load.10	0.1	2	800
Load.11	0.5	2	700
Load.12	0.1	9	400
Load.13	0.3	3	900
Load.14	0.3	3	200
Load.15	0.4	2	700
Load.16	0.4	2	700
Load.17	0.4	2	700
Load.18	0.1	3	1000
Load.19	0.5	3	900

Table 7.11 continued from previous page

Neural Networks best tuning parameter with feature Selection			
Load.20	0.1	9	1000
Load.21	0.1	8	700
Load.22	0.5	5	700
Load.23	0.4	5	600

7.3.5 List of Best XGBoost tuning Parameters per target variable with feature selection

Table 7.12: XGBoost best tuning parameters with feature Selection

XGBoost best tuning parameters with feature Selection			
target variable / parameters	eta	depth	round
Load.0	0.1	3	1000
Load.1	0.1	4	400
Load.2	0.1	5	400
Load.3	0.1	5	100
Load.4	0.1	5	100
Load.5	0.1	2	200
Load.6	0.1	2	400
Load.7	0.1	4	300
Load.8	0.1	3	500
Load.9	0.1	5	700
Load.10	0.1	5	900
Load.11	0.1	3	1000
Load.12	0.2	5	300
Load.13	0.1	4	800
Load.14	0.1	5	400
Load.15	0.1	4	900
Load.16	0.1	2	1000
Load.17	0.1	2	1000
Load.18	0.1	4	1000
Load.19	0.1	3	1000
Load.20	0.1	3	700
Load.21	0.3	3	400
Load.22	0.1	5	900
Load.23	0.1	3	800

7.3.6 List of Best Model Trees tuning Parameters per target variable with feature selection

Table 7.13: Model Trees best tuning parameter with feature selection

Model Trees best tuning parameter with feature selection	
target variable / parameters	unbiased
Load.0	1
Load.1	0
Load.2	0
Load.3	0
Load.4	1
Load.5	1
Load.6	1
Load.7	0
Load.8	0
Load.9	0
Load.10	0
Load.11	0
Load.12	0
Load.13	0
Load.14	0
Load.15	0
Load.16	0
Load.17	0
Load.18	0
Load.19	0
Load.20	1
Load.21	0
Load.22	0
Load.23	0

Bibliography

[1] Electrical load forecasting : modeling and model construction / Soliman Abdel-hady Soliman (S.A. Soliman), Ahmad M. Al-Kandari.

[2] Short Term Electric Load Forecasting by Tao Hong, A dissertation submitted to the Graduate Faculty of North Carolina State University in partial fulfillment of the requirements for the degree of Doctor of Philosophy, Operations Research and Electrical Engineering, Raleigh, North Carolina

[3] A. D. Papalexopoulos and T. C. Hesterberg, ”A regression-based approach to short-term system load forecasting,” IEEE Transactions on Power Systems.

[4] T. Haida and S. Muto, ”Regression based peak load forecasting using a transformation technique,” IEEE Transactions on Power Systems.

[5] B. Krogh, E. S. de Llinas, and D. Lesser, ”Design and Implementation of An on-Line Load Forecasting Algorithm,” IEEE Transactions on Power Apparatus and Systems.

[6] M. T. Hagan and S. M. Behr, ”The Time Series Approach to Short Term Load Forecasting,” IEEE Transactions on Power Systems.

[7] N. Amjady, ”Short-term hourly load forecasting using time-series modeling with peak load estimation capability,” IEEE Transactions on Power Systems.

[8] D. C. Park, M. A. El-Sharkawi, R. J. Marks, II, L. E. Atlas, and M. J. Damborg, ”Electric load forecasting using an artificial neural network,” IEEE Transactions on Power Systems, vol. 6, pp. 442-449, 1991.

[9] D. K. Ranaweera, N. F. Hubele, and A. D. Papalexopoulos, ”Application of radial basis function neural network model for short-term load forecasting,” IEEE Proceedings - Generation, Transmission and Distribution

[10] A. Apostolos, ”Short-Term Load Forecasting using a Cluster of Neural Networks for the Greek Energy Market”, Electrical & Computer Engineer Public Power Corporation Greece.

[11] Sp.Kiartzis, Artificial Intelligence Applications in Short-term Load Forecasting, PhD Dissertation, AUTH, 1998

[12] A.G.Bakirtzis, V.Petridis, S.J.Kiartzis, M.C.Alexiadis, A.H.Maissis, A neural network short term load forecasting model for the Greek power system, IEEE Transactions on Power Systems, Vol. 11, No.2, May 1996, pp. 858-863

[13] D. Papamilios, Load Declaration Software, Dissertation, 2012, AUTH

[14] S. Rahman and R. Bhatnagar, ”An expert system based algorithm for short term load forecast,” IEEE Transactions on Power Systems, vol. 3, pp. 392- 399, 1988.

[15] S. Rahman, ”Formulation and analysis of a rule-based short-term load forecasting algorithm,” Proceedings of the IEEE, vol. 78, pp. 805-816, 1990

[16] K.-L. Ho, Y.-Y. Hsu, C.-F. Chen, T.-E. Lee, C.-C. Liang, T.-S. Lai, and K.-K. Chen, ”Short term load forecasting of Taiwan power system using a knowledge-based expert system,” IEEE Transactions on Power Systems, vol. 5, pp. 1214-1221, 1990.

[17] Y. Y. Hsu and K. L. Ho, ”Fuzzy expert systems: an application to short-term load forecasting,” IEE Proceedings - Generation, Transmission and Distribution, vol. 139, pp. 471-477, 1992.

[18] P. A. Mastorocostas, J. B. Theocharis, and A. G. Bakirtzis, ”Fuzzy modeling for short term load forecasting using the orthogonal least squares method,” IEEE Transactions on Power Systems, vol. 14, pp. 29-36, 1999.

[19] H. Mori and H. Kobayashi, ”Optimal fuzzy inference for short-term load forecasting,” IEEE Transactions on Power Systems, vol. 11, pp. 390-396, 1996.

[20] S. Rahman, ”Formulation and analysis of a rule-based short-term load forecasting algorithm,” Proceedings of the IEEE, vol. 78, pp. 805-816, 1990

[21] N. Sapankevych and R. Sankar, ”Time Series Prediction Using Support Vector Machines: A Survey,” IEEE Computational Intelligence Magazine, vol. 4, pp. 24-38, 2009

[22] Electrical Load Forecasting using Support Vector Machines, Belgin Emre, Dilara Demren, Instanbul Technical University, Turkey.

[23] C.-M. Huang and H.-T. Yang, ”Evolving wavelet-based networks for shortterm load forecasting,” IEE Proceedings - Generation, Transmission and Distribution, vol. 148, pp. 222-228, 2001.

[24] Ensemble Deep Learning for Regression and Time Series Forecasting, Xueheng Qiu, Le Zhang, Ye Ren and P. N. Suganthan, School of Electrical and Electronic Engineering, Nanyang Technological Univeristy, Singapore, Gehan Amaratunga, Department of Engineering, University of Cambridge, UK.

[25] Deep Neural Network Based Demand Side Short Term Load Forecasting, Seunghyoung Ryu, Department of Electronic Engineering, Sogang University, 35 Baekbeom-ro, Mapo-gu, Seoul 121-742, Korea; shryu@sogang.ac.kr, Jaekoo Noh, Software Center, Korea Electric Power Corporation (KEPCO)

[26] Short Term Electrical Load Forecasting Using Mutual Information Based Feature Selection with Generalized Minimum-Redundancy and Maximum-Relevance Criteria, Nantian Huang, Zhiqiang Hu, Guowei Cai and Dongfeng Yang, School of Electrical Engineering, Northeast Dianli University, Jilin 132012 China.

[27] Short-Term Load Forecasting using Random Forests, Grzegorz Dudek, Department of Electrical Engineering, Czestochowa University of Technology, Czestochowa, Poland

[28] Rule-Based Prediction of Short-term electric Load, Petr Berka, Prof, PhD, University of Economics Prague & University of Finance and Administration Prague, Czech Republic

[29] Formulation and analysis of a rule-based short-term load forecasting algorithm, S. Rahman, Dept of Electr. Eng., Virginia Polytech. Inst. & State Univ. Blacksburg, VA, USA, ieeexplore.ieee.org/document/53400

[30] A Novel Hybrid Model Based on Extreme Learning Machine, k-Nearest Neighbor Regression and Wavelet Denoising Applied to Short-Term Electric Load Forecasting, Weide Li, Demeng Kong and Jinran Wu, School of Mathematics and Statistics, Lanzhou University, Gansu, China

[31] A composite k-nearest neighbor model for day-ahead load forecasting with limited temperature forecasts, Rui Zhang, Yan Xu, Zhao Yang Dong, Weicong Kong, Kit Po Wong, School of Electrical and Information Engineering, University of Sydney, NSW 2006, Australia ieeexplore.ieee.org/document/7741097

[32] Short-Term Electricity Load Forecasting Based on the XGBoost Algorithm, Guangye Li, Wei Li, Xiaolei Tian, Yifeng Che, State Grid Liaoning Electric Power Co., Ltd., Shenyang Liaoning

[33] A gradient boosting approach to the Kaggle load forecasting competition, Souhaib Ben Taieb, Machine Learning Group, Department of Computer Science, Faculty of Sciences, Universit´e Libre de Bruxelles, Rob J Hyndman, Department of Econometrics and Business Statistics, Monash University, Clayton, VIC 3800, Australia

[34] O. Chapelle and V. Vapnik, Model Selection for Support Vector Machines. In Advances in Neural Information Processing Systems, Vol 12, (1999)

[35] M. T. Hagan and S. M. Behr, ”The Time Series Approach to Short Term Load Forecasting,” IEEE Transactions on Power Systems, vol. 2, pp. 785-791, 1987.

[36] svm regression tutorial

[37] svm regression tutorial matlab

[38] random forest tutorial

[39] random forest explainer

[40] KNN regression

[41] neural networks regression

[42] Introduction to Boosted Trees

[43] An Introduction to XGBoost

[44] An Introduction to Gradient Boosting

[45] Cubist - Model Trees Presentation

[46] R language wiki

[47] Weka wiki

[48] SPSS wiki

[49] Excel wiki

[50] python wiki

[51] scikit-learn wiki

[52] pandas wiki

[53] statsmodel wiki

[54] Anaconda main site

[55] Rapidminer wiki

[56] Orange wiki

[57] Knime wiki

[58] Boruta Feature Selection Algorithm