

ARISTOTLE UNIVERSITY OF THESSALONIKI
SCHOOL OF ECONOMICS
Msc. Of Logistics and Supply Chain Management



MACHINE LEARNING APPLICATIONS IN SUPPLY CHAIN MANAGEMENT

Master Thesis Dissertation
MAVRIDIS GEORGIOS

Supervisor: Athanasios Tsadiras

THESSALONIKI
MARCH 2021

Table of Contents

Table of Contents	1
Figures Table	3
List of Abbreviations	4
Abstract	6
1. Introduction	7
2. Research Questions.....	7
3. Literature Review	8
4. Machine Learning.....	9
4.1 Machine Learning types	10
4.1.1 Supervised Learning	11
4.1.2 Unsupervised Learning	11
4.1.3 Reinforcement Learning	11
4.2 Machine Learning Methods	11
4.2.1 Polynomial regression.....	12
4.2.2 Gaussian process regression.....	12
4.2.3 Artificial neural networks	12
4.3 More specific examples of machine learning methods	13
4.3.1 Decision trees.....	13
4.3.2 K-means clustering	13
4.3.3 Dimensionality reduction.....	14
4.3.4 Multi-criteria inventory classification.....	15
4.3.5 Classification criteria	15
4.3.6 Machine learning classifiers.....	15
4.3.7 SVM with radial basis function kernel.....	16
4.3.8 Deep neural network	16
5. Machine Learning Applications.....	18
5.1 Machine learning and Industry 4.0	18
5.2 Big data analytics in supply chain	19
5.3 Different machine learning applications.....	20
5.3.1 Supervised Machine Learning Application for effective supplier classification	20
5.3.2 Machine Learning Application for sales forecasting	21
5.3.3 Stock-out Prediction Algorithm	21

5.4 Machine learning applications to SCs and manufacturing	22
5.5 Machine learning approach to find business partners and building mutual relationships	26
5.6 Hybrid demand forecasting models.....	27
6. Data in supply chains	28
6.1 Demand management in supply chains.....	29
6.2 Big Data Analytics for demand forecasting in SCM.....	30
6.3 Big data in manufacturing: a systematic mapping study.....	31
6.4 Production planning.....	32
7. Timeseries forecasting	33
7.1 Definition of a Timeseries	34
7.2 Components of a Time Series	35
7.3 Introduction to Timeseries Analysis.....	35
8. Chosen algorithms for our research	36
8.1 Demand prediction with ARIMA forecasting model	37
8.1.1 Pros and Cons of ARIMA Forecasting	40
8.1.2 Four key steps in forecasting when using the ARIMA model	41
8.2 SARIMAX forecasting model	41
8.3 Random forest regressor	43
8.3.1 Advantages of Random Forest:.....	45
8.3.2 Disadvantages of Random Forest:	45
8.4 Holt Winters Algorithm	46
8.4.1 Holt-Winters' additive method	47
8.4.2 Holt-Winters' multiplicative method	48
8.5 Simulations based on the space state model	49
8.6 Performance measure errors.....	50
9. Data set analysis.....	51
10. Implementation of Algorithms in Python	54
10.1 Python Modules	54
10.1.1 Pandas	54
10.1.2 Sklearn	55
10.1.3 Stat models.....	56
10.1.4 Tkinter.....	56
10.1.5 Numpy.....	56

10.1.6 XlsxWriter.....	57
10.2 Script for 1 timeseries description in steps.....	57
10.3 Script for n timeseries description in steps	58
10.4 Main commands which use in python scripts.....	58
10.5 Results of python script for algorithm comparison in 1 timeseries.....	60
10.6 Results of python script for algorithm comparison for n timeseries.....	69
11. Conclusions	72
12. Future Work	73
References	74
Figures references.....	77
APPENDIX 1:Python script code for the comparison of the algorithm errors applicable in one timeseries	78
APPENDIX 2: Python script code for the comparison of the algorithm errors applicable in many timeseries with the for loop.....	90
APPENDIX 3: Split the data in sheets based on the Product_Code	101
APPENDIX 4: Put all the sheets in ascending order	103
APPENDIX 5: Copy all demand rows side by side in one new combined sheet.....	104
APPENDIX 6: Erase the columns	105

Figures Table

Figure: 1 Articles per year graph.....	9
Figure: 2 Machine Learning Types	10
Figure: 3 k-means clustering	14
Figure: 4 Deep Neural Network	17
Figure: 5 Machine Learning Applications.....	18
Figure: 6 the four industrial revolutions	19
Figure: 7 Demand Forecast	33
Figure: 8 Time series	34
Figure: 9 Random Forest.....	44
Figure: 10 Excel file for the monthly demand.....	53
Figure: 11 Timeseries data graph.....	61
Figure: 12 ETS decomposition.....	61
Figure: 13 estimators for the random forest regressor	62
Figure: 14 Random Forest Regressor results	62
Figure: 15 Random Forest Regressor results graph.....	63
Figure: 16 ARIMA model evaluator.....	63
Figure: 17 ARIMA model results	64
Figure: 18 ARIMA model results graph	64

Figure: 19 ARIMA model summary	65
Figure: 20 SARIMAX model results	65
Figure: 21 SARIMAX model graph with simulations	66
Figure: 22 SARIMAX results graph	66
Figure: 23 Holt Winters model selection	67
Figure: 24 Holt Winters graph with simulations	67
Figure: 25 Holt Winters Results graph	68
Figure: 26 Holt Winters model summary	68
Figure: 27 Errors of the compared algorithms.....	69
Figure: 28 column start and end & estimators for random forest regressor	69
Figure: 29 Script output screen errors	70
Figure: 30 xlsx file script output.....	70
Figure: 31 Comparison sheet in xlsx file	71

List of Abbreviations

Literature review (LR)

Machine learning (ML)

Field of research (FoR)

Gaussian process regression (GPR)

Artificial neural network (ANN)

Multi-layer perceptron (MLP)

Background distribution algorithm (BP)

K-nearest neighbor algorithm (k-NN)

Multi-criteria inventory class (MCIC)

Support vector machines (SVM)

Deep neural networks (DNN)

Neural Networks (NN)

Radial Basis Function (RBF)

Information and communication technology (ICT)

Cyber-physical systems (CPS)

Internet of Things (IoT)

Industrial internet (IIoT)

Virtual Internet (PI)

Artificial intelligence (AI)

Deep learning (DL)
Supply Chain Management (SCM)
Vector Auto Regression (VAR)
Inventory level (IL)
Supply Chain (SC)
Supervised Machine Learning (SML)
Autoregressive Integrated Moving Average (ARIMA)
Autocorrelation function (ACF)
Partial autocorrelation functions (PACF)
Akaike Information Criterion (AIC)
Big Data Analytics (BDA)
Holt winters (HW)
Random forest (RF)
Random forest regression (RFR)
Mean error (ME)
Mean absolute error (MAE)
Mean squared error (MSE)
Root means square error (RMSE)
Mean percentage error (MPE)
Mean absolute percentage error (MAPE)

Abstract

Our scope in this thesis is to analyze the Machine Learning applications in supply chain management and to implement a model-tool in one of the most important of these applications, which is the demand forecasting. Our literature review has as main theme the analysis and the presentation of ML types, methods and applications in supply chain management with some specific examples. Moreover, we do research about the data in supply chain, the demand management in supply chain and the usage of Big Data Analytics in demand forecasting and in SCM. Based on these we conclude about the importance of demand forecasting in SCM and how useful it is when companies have an easy to use tool to decide which algorithm is more accurate in order to have the best possible forecast. We develop two python scripts, which compare demand-forecasting results of four different algorithms of the categories auto regression, exponential smoothing, and machine learning ones by compare their performance metrics. This comparison take place by using an .xlsx file where we rank the algorithms based on their performance metrics results. The input in the scripts is the dataset which contains the historical data from 2012-2016 of demand from a big firm. In these data, we do preprocessing and form them as monthly demand timeseries in order to be able to work with this time step. The algorithms and the performance metrics we use, we find them in our literature review and by providing and analyzing them. We conclude and decide to use the ARIMA as the most well known algorithm in demand forecasting. Next algorithm is the SARIMAX one of most suitable algorithms for seasonal demand with exogenous factors. Then the Holt Winters as one of the most suitable algorithms for timeseries demand forecasting and finally the Random Forest Regressor as a very precise and well-known ML algorithm for demand forecasting.

Our thesis structure is as below, first the introduction, which introduce us in the aspect of machine learning and its applications in SCM. The research questions, which we have form, based on our main scope and based in our motivations. Next, is the first main part which is the literature review where we analyze more specific the Machine Learning, the Machine Learning types, the Machine Learning Methods, examples of machine learning methods, Machine Learning Applications in general, Machine learning and Industry 4.0, Big data analytics in supply chain, Different machine learning applications like applications to SCs and manufacturing and in demand management in supply chains and in production planning. After these chapters, we analyze the timeseries forecasting and the chosen algorithms for our research and the performance metrics. The second part of our thesis is the technical programming; here we present our steps to make the python scripts. First is the data set analysis and preprocess, then is the scripts analysis and description of the used python modules and functions. In addition, we present details of each script with pictures and steps of the execution and the results. Moreover, we present our conclusions and our suggestions for future work. At the end, we present the full code of both scripts in our appendixs with comments in order to be easily understand.

Keywords: machine learning, demand forecasting, timeseries, python scripts, ARIMA, SARIMAX, Holt Winters, Random forest regressor, performance metrics

1. Introduction

Valid and correct forecasts have an important role in many sectors like science, industry, commerce and economy. In nowadays business consumer-centric environment, companies look for high sales performance and often need to keep a balance between satisfying them customers' demand and control the cost of logistics and inventory. Keeping in stock bigger inventory provides us the ability to satisfy the clients' demand at any time, but may result in over-stocking, leading to issues such as tied-up capital, written down inventory, and lower profit margins. In comparison, lower stock can reduce the cost of a product, but can lead to cost overruns arising from non-existent sales opportunities, lower customer satisfaction, and other issues. In this thesis, we study about the ML applications in supply chain management and more specific the demand forecasting applications. Forecasting is the process of determining future demand and forecasts can be used to store the required amount of inventory to prevent less or additional issues. Forecast results can affect the company's financial planning, marketing, customer management, logistics planning and other company sectors. As a result, improving the accuracy of demand forecasts it becomes an important part of the company's performance. Demand forecasting is a traditional but very effective method that predicts the timeseries future values. Timeseries forecasts are forming the base for the performance of any process over time based in the previous historical data. Predictability is determined by using historical data and forecasts. We dedicate much effort in recent decades to develop and improve various types of forecasting with many models and algorithms. The elements of the time series are actually noisy, unstable, out of line, poorly constructed and many factors affecting the political, economic and psychological factors that have made so many exchange-related applications such as complex applications for financial forecasting strategies. Sales forecasting plays a key role in financial planning and business conduct in any organization to assess past and current demand statistics and predict future performance. Overall, direct demand forecasts provide more efficiency and effectiveness in company in order to saving money on predicting methods or forecasts described as Statistical Modeling, ML. For these reasons, we decide to conduct a literature review to find all ML applications in supply chain management and more specific these regarding to demand forecasting. Moreover, by developing two python scripts that compare the demand forecasting results of four different algorithms of the categories auto regression, exponential smoothing, and ensemble machine learning ones by compare their performance metrics. After compare the performance metrics, the script provide us the result which of the algorithm is better for the specific dataset, which we import in the scripts. One script is for one timeseries data and the other script we develop it in order to make forecasts for many timeseries and based the results of errors provide us the best most accurate algorithm by using the help of Microsoft excel to make the algorithms order and see which of them is the best.

2. Research Questions

In this thesis, we have express two research questions to can successfully accomplish our scope as follow:

RQ1: Which is the most suitable machine learning (or statistical) algorithm for forecasting the demand in the area of logistics?

Motivation: The motivation of this research question is to study and find among the machine learning and statistical algorithms the best one to use for forecasting of demand in logistics sector based on timeseries historical data.

RQ2: Which are the most suitable performance metrics in order to find this algorithm, which provides us the best forecasting for the product demand based on the historical data of our demand dataset?

Motivation: The motivation of this research question is to find the most suitable ML algorithm among the chosen ones after compare their results based on the performance metrics (errors). We define the most suitable algorithms for demand forecasting after conduct our literature review and choose them to answer RQ1. We analyze and test the chosen algorithms, and we select the most efficient algorithm based on the results obtained after the performance evaluation answers the RQ2.

3. Literature Review

A literature review (LR) is carried out as first part of our thesis to distinguish the machine learning (ML) applications and the algorithms which been used in order to solve many problems in logistics area and in supply chain, more specific we focus in ML applications in demand forecasting. We choose some algorithms to see those benefits and we choose the best based on the performance metrics, which also we select based on our LR.

Inclusion Criteria:

- Articles published and released during 2005 - 2020.
- Articles published in books, Journals, conferences and magazines.
- Articles published in English.
- Articles fully available with or without institutional access.

Exclusion Criteria:

- Articles not published in English.
- Articles without complete text.
- Articles prior to 2005

Our LR is conducted by using academic sources, which are Scopus, Emerald insight, Ieee Xplore, Elsevier, Related textbooks, Wiley Online Library, Research Gate etc. We search on these by using the keywords strings ‘machine learning’, ‘timeseries forecasting’, ‘demand forecasting by using machine learning’, ‘machine learning applications in supply chain management’ and as secondary we search under combinations of the followings:

- Machine learning
- Supply chain management
- Big data
- Industry 4.0
- Timeseries
- Forecasting

- Demand
- ARIMA
- SARIMAX
- HOLT WINTERS
- RANDOM FOREST REGRESSOR
- Forecasting metrics

The advantage of this type of review is that similar results should be obtained if the procedure is repeated. We choose this specific FoR in order to limit the number of papers analyzed; otherwise, our approach would not be replicable and objective. Our screening process takes place as mentioned below steps:

- First step was to make the search online using the mentioned sources and keywords.
- Second step was to decide which papers we will keep according to our FoR and our thesis thematic.
- Third step was to download and write down the articles and their authors and publish date in an .xlsx file in order to categorize them according to the year and the content to be easier the review. Also we make the graph in Figure1 which denotes the number of articles per year. As we can see we use mostly new articles in our LR.
- Final step is to analyze and compare all the articles in order to form our LR.

During our research, we found 110 related articles of which we finally use in our LR 53 of them.

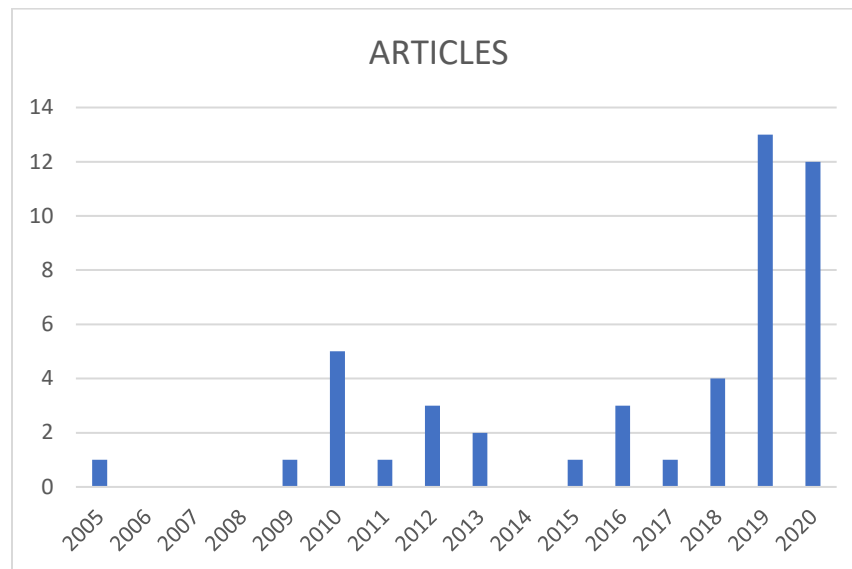


Figure: 1 Articles per year graph

4. Machine Learning

Machine learning (ML) is programming computers to optimize a performance criterion using example data or past experience (historical data). ML is to build algorithms that can either receive input data or use statistical analysis to predict (Predictive) an output while updating outputs as new data becomes available. ML is an area under artificial intelligence and represents another system

approach. Sample data replaces strict system calculation rules. From the data of the given model, learning methods or algorithms produce statistical flexibility, and represent those that are in the model state. Genres can respond to new, unknown data and categorize or generate predictions. ML works with a computer-assisted model and the realization of learning environments. It is defined as a process that uses information in order to improve performance or make concrete predictions. The experience is based on past data, which is provided by the process from electronic data collection. ML incorporates the construction of effective and intuitive algorithms. (Mohri, Rostamizadeh and Talwalkar, 2012 p. 1-5).

4.1 Machine Learning types

As mentioned in literature by many authors like (Djordje Cica,2020) (Wenzel et.al ,2019) (Géron, 2017, p. 22-31) and (Tarallo et al 2019), machine learning algorithms are classified as supervised, unsupervised or reinforcement learning. In the supervised algorithms, a training set has the appropriate answers and, based on this training application, the algorithm performs a good response to all possible given data. Unsupervised machine learning algorithms incorporate patterns from a data set without referring to the already known or labeled results. Unlike supervised machine learning, unsupervised machine learning methods cannot be applied directly to a retrospective or partition problem because the values of the output data are unknown, making it difficult to train the algorithm in the normal way. In reinforcement learning algorithm learns the policy of how to do it given the view of the world. All actions have a certain impact on the environment, and nature provides the answer that guides the learning algorithm.

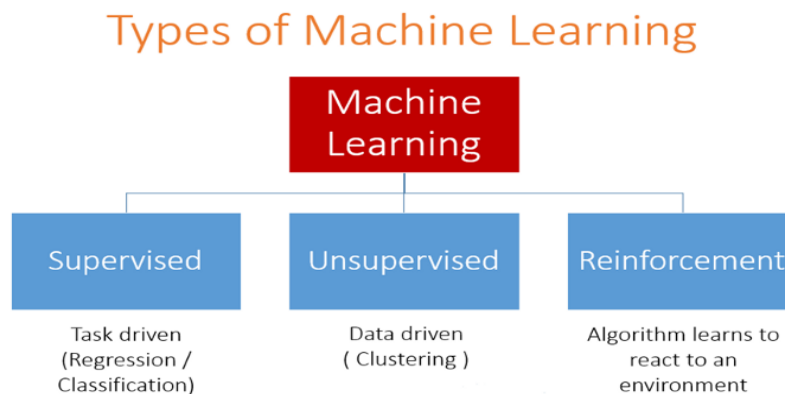


Figure: 2 Machine Learning Types

(7wdata.be)

Although machine learning in general and supervised methods in particular, are usually associated with predictive analytics, which can also be applied to prescriptive tasks. In some problems (e.g. transactions acceptance), converting a prediction into a prescription is straightforward (if the transaction is predicted to be fraudulent, it should be rejected). When that is not the case, there are

two main ML paradigms to devise a prescription. The first is to generate good and optimal solutions, and use a supervised method to infer some rule from those solutions. The second is to generate, combine and simulate different rules, and output the best of them.

4.1.1 Supervised Learning

The most known type of ML is Supervised Learning. Supervised Learning is a process where a software is trained by using known model data. As output is also known, this learning process is made to find the connection in the form of rules, which integrates the input data with output data and ultimately applies learned rules to the new data. The software is currently undergoing training. Two important functions of this ML type are data classification and regression.

4.1.2 Unsupervised Learning

Unsupervised learning describes a ML process that is capable to predict knowledge. Real data are not given, thus, there are no predefined target values. This approach is also called "learning without a teacher". A well-known function of unsupervised learning is clustering. Clustering identifies similarities in the input values in order to categorize them by common patterns.

4.1.3 Reinforcement Learning

In reinforcement learning, the best solution is unknown to the system at the start of the learning phase and after that, we must be determined iteratively. In this process, sensible approaches are rewarded, and wrong steps tend to be punished. With this approach, the system is highly possible to takes into account complicated environmental factors that provide influence to the results and reacts accordingly. Hence, the system finds its own solutions separately through directional rewards and punishments.

4.2 Machine Learning Methods

We group the ML methods according to task they are solving. In addition, there are several different algorithms for each method. Part of ongoing research in ML is the development of additional methods, models and algorithms. Choosing a model and algorithm depends on many factors and it is not available a solution for all - also known as "No free lunch with statistics". Choosing the right method for a given problem with known dataset is one of the most challenging tasks in data analysis.

Based on (Djordje Cica,2020) paper, by using ML methods and algorithms, some challenges may occur. Two different reasons can be mentioned when it comes to the development of error sources. On the one hand, the problem can be found in the data. The availability of a large amount of data which is necessary for the training of a model is still a frequent challenge, as well as insufficiently representative training data. Another challenge is poor data quality. In order to make patterns visible in data sets, they first must be detected. If the data contains errors or outliers, it is difficult to identify such patterns. Another issue is redundant features, which do not provide added value to the model. An important step for the success of a ML method is therefore to select suitable features for training. On the other hand, a poorly chosen algorithm can create difficulties in the form of overfitting and underfitting. Overfitting means that ML algorithms can sometimes generalize incorrectly although the model works on a training data set. A model can be too complex, which

means that the model follows noise or errors in the data too closely. Underfitting is the opposite of overfitting and describes a model that is too simple for the structure of the data to be recognized and learned.

4.2.1 Polynomial regression

Regression analysis is one of the most significant factors of analysis and ML. The purpose of the regression analysis is to show the relationship between dependent and independent variables during the estimation process of the future values. A simple method of regression function is linear regression, in which the output (response) variable is followed as a line combination of independent variable (input). Advanced regression models include multiple retrospective analysis where dependent variables are also equated with independent variables. The hypothetical linear relationship between dependent and independent variables may not be sufficient to define a particular relationship. In the polynomial retrospective model, relationships between dependent variables and autonomy are formed in the form of a polynomial equation. Since polynomial retraction models are considered special cases for multiple line retraction models, filing these models at least does not present a new problem and fossil analysis can be used to determine model fit.

4.2.2 Gaussian process regression

Gaussian process regression (GPR) for machine learning was initially implemented by Williams and Rasmussen. Compared to other recovery strategies based on the kernel process, like SVM, the GPR is a model that is based on traditional Bayesian methods. The GPR is very simple to deal with complex issues of high magnitude, inequality and a small number of training parameters. Due to its excellent performance, GPR has been widely used in recent years in various engineering fields.

4.2.3 Artificial neural networks

Over the past few decades, ML strategies, the majority of the artificial neural network (ANN), have captured the interest of many researchers in almost every field of engineering. ANN encouraged the processing of human brain information in an effort to achieve human functioning. Because of the moderate capacity of offline function, noise resistance, adaptability and general performance, ANN is particularly useful in modeling machine processes that are characterized by many highly related parameters. Different types of ANN are suggested in the literature, but multi-layer perceptron (MLP) is the most widely used. MLP is a type of feed-forward ANN consisting of neurons divided into three types of layers: (i) input layer, (ii) output layer and (iii) hidden layers (one or more). Each layer consists of a group of neurons (nodes) connected by neurons from other layers by connecting between neurons. Each neuron within a network is a simple processing unit where basic calculations are performed to process one or more inputs and produce relevant results. The connections between neurons, or synapses, have a corresponding mass that controls the release of neurons. ANN results can be modified by adjusting synaptic values. This weight adjustment is

designed to be on the side that reduces the difference between the ANN output and the current response loads. The background distribution algorithm (BP) is probably one of the most popular methods in the ANN field. Therefore, in the present study, an optional feed for more advanced ANN based BP algorithm was selected to improve the model prediction. Input layer consists of a set of neurons representing the process inputs features. Hidden layers and the nodes per hidden layer is usually determined through a ‘trial and error’ method, by increasing or the number of hidden layer and neurons during training. The last layer acts as the network output layer. The neurons in the last layer that works as the network output layer is the same as the number of functions being approximated by the model.

(Wenzel et.al ,2019) (Géron, 2017, p. 22-31) (Tarallo et al 2019)

4.3 More specific examples of machine learning methods

In the supervised machine learning, according to the researches of (Álvaro Silva de Melo,2019) (Alpaydin, 2009) and (Taiwo, 2010) we point out methods like:

4.3.1 Decision trees. These models use observations approximately sure actions and become aware of an ideal direction for arriving at a favored outcome. Neural Networks can actually carry out some of regression and/or classification obligations right away, although commonly every network performs only one. In the good-sized majority of instances, therefore, the network could have an only output variable, even though within the case of many-state category troubles, this could correspond to a number of output units. In pattern popularity, the k-nearest neighbor algorithm (k-NN) is a non-parametric technique used for classification and regression. In each of instances, the input includes the k closest training examples in the characteristic area. The output relies upon on whether or not the k-NN used for classification or regression.

In addition, in unsupervised machine learning we will face methods like:

4.3.2 K-means clustering which is one of the best ways to learn unsupervised learning algorithms who gives solutions to a known interaction problem. This procedure is implement in an easy and simple way to separate the given data set by a certain number of clusters (think groups k) that form a priori. This model collects a certain number of data points from a specific number of collections based on similar factors.

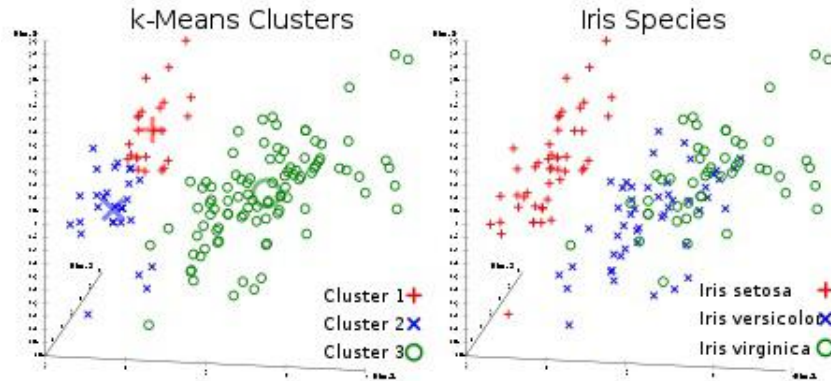


Figure: 3 k-means clustering

(wikimedia)

4.3.3 Dimensionality reduction.

As the name suggests, the usage of this method is to remove the most important information (sometimes unwanted columns) in the data set. Basically, the process of obtaining integrated high-end data presentations.

In their paper (Lolli et al, 2019) after read and reviewed other before years' papers, they focused on classification of products for inventory control. When a class framework is designed for the inventories, it determines how important the products and the size of the controls placed on them are. A well-fitted reorder coverage wishes to be selected for all merit so that the application of the class-selected re-ordering policy rather than compliance makes it an easy way to deal with every stock management problem. However, classifying gadgets by multi-criteria inventory class (MCIC) is a different way of finding the right techniques for all aesthetics. The full operation of the MCIC strategy does not involve the operation of the inventory management machine and the low interest paid on the effect of the durability of the stock type simultaneously with the use of the asset. The only goal of reducing all operating costs is left behind when MCIC and stock structures are not made simultaneously. Clearly, a complete simulation of the inventory machine performed made at the same level of material can end up not being used to the MCIC by carefully arranging all the items set out in terms of the cost of the invention, solving the open problem mentioned above. However, a complete simulation of the total number of items requires very high collection efforts in a variety of time settings with a huge number of items that need to be controlled.

They additionally investigate the supervised machine learning classifiers as important and needed tools for MCIC, support vector machines with Gaussian kernel (SVM) and deep neural networks (DNN). Machine learning knowledge demands a big volume of data in order to work well and enables complex decision-making techniques to be automatic. MCIC applied to lots of items is the sort of complex methods, and ML tools may additionally, therefore be promising in MCIC. However, the literature is especially poor in this particular area of studies. From a methodological aspect, this two-stage automatic forecasting and stock manage for intermittent call for the demand for spare parts is irregular, and the literature has investigated the success of demand forecasting and stock manage methodologies in said context. A simple exponential smoothing is implemented

to each variable whilst the demand takes place and then an estimator of the predicted cost of call for according to period is evaluated by means of the ratio of these smoothing results. Artificial intelligence has also been implemented for forecasting intermittent demand because of its capacity to generalize a non-linear technique without requiring any distributional assumptions. When the intermittent demand styles contain seasonal and trend additives, SARIMA model considers promising consequences. Although demand forecasting and inventory control contributes equally toward the performance of inventory management, it must be emphasized that the contrast of forecasting models was implemented by the authors in phrases of forecasting accuracy, which does not always guide to better performance of the inventory forecasting machine in phrases of total applicable price. Concerning the inventory manage of sporadic demand profiles, the literature can be separated into two essential streams. The first stream gives a huge set of compound distributions for modelling purposes. In unique, at the statistical evaluation of intermittent call for. The second referring to stock manage focuses on the overall performance of various inventory structures on a big variety of items taken from commercial case research. Clearly, this stream of research additionally matches MCIC. The effectiveness of stock systems within the case of sporadic demand has been investigated earlier than years in numerous researches, but the item type is once more created earlier than testing the inventory machine on the generated instructions, and for this reason seems disjointed.

4.3.4 Multi-criteria inventory classification

A newly developed contribution on MCIC provided in their study. They analyzed a classification model, which applies AHP in order to classify products in terms of annual cost, average unit cost, criticality and the lead-time. This acts for the one of the earliest attempts to overcome the weaknesses identified by the mono-criteria by the value of use in representing all values.

4.3.5 Classification criteria

In this work, classification based on the inventory. The method that is expected to influence the performance of the inventory management system, and as a result the classification of the item, is preferred. These methods fall into two distinct groups, referring to time series calculations and object elements, respectively. The first group sets the required parameters to set the production demand. They refer to two stochastic variables of the periodic demand process, e.g. Distinguish between successive demands and the size of the positive demand size. The second decision-making process is based on the inventory management system and on the method used to select the best policy for each product.

4.3.6 Machine learning classifiers

They use two different ML classifiers: SVM with radial foundation function kernel and DNN. Both algorithms can be efficient in non-linear contexts and provides an optimized and moderate big variety of meta-parameters. For the SVM, these meta-parameters are: Box constraint (C), which is applied to incorrectly separated products. Kernel scale (scale), the radial foundation function kernel is accelerated through this scale factor. While for the DNN, the related meta-parameters are: Number of hidden layers (Nhidden). Number of neurons in step with hidden layer (Nneurons). For each one of the algorithms and the meta-parameter, a tenfold cross validation is applied to evaluate the combination performance. A fold makes use of nine tenths of the gadgets

to be had (education set) to train the technique and the last one 10th to are expecting the training (validation set). The universal performance evaluation takes place through combining the validation set forecasts and remaking the unique dataset, these predictions are evaluated in opposition to the authentic instructions of gadgets and the ratio of accurate predictions is measured. A most useful set of meta-data for each algorithm is recognized as the one that is maximizing the forecast performance. This cross-validation method allows the meta-data optimization even as leveraging the complete dataset and proscribing the effect of the unmarried fold department on the measured overall performance.

4.3.7 SVM with radial basis function kernel

A standard SVM is a two-phase line divider that separates feature space and hyperplane. The hyperplane is made to magnify the margin (m), increasing twice the distance of the hyperplane and its nearest object to the element space, while maintaining the correct phase separation. It is possible to use SVM in cases where is not possible to separate the data sequentially. In these cases, the system can differentiate unfairly other models, thereby paying a price for the intended function. The incorrect classification error is a box constraint C multiplied by the classification error; we define this SVM as soft margin SVM.

4.3.8 Deep neural network

Dnn is a non-linear classifier with many classes that gives the product functions thru a set of layers and outputs a vector in which each layer is a vector associated with the subsequent one thru a matrix of weights w_l and a set of non-linear capabilities. If high-quality linear equations have been carried out, the very last end result will be a multivariate logistic regression. That is due to the reality that an aggregate of linear features is a linear feature. To add non-linearity, the sigmoid function ought to be used after every linear mixture. The use of a linear classifier within the final layer isn't always uncommon to dnn and svm with the radial foundation feature kernel; each algorithms growth the features place and use linear classifiers to gain non-linear classifications inside the proper space. They carry out this optimization with the useful resource of enhancing the weights and biases of every layer, which include the output one. On the manner to understand a way to regulate weights and biases, the gradient of pass is acquired. Using the multivariable chain rule, it's miles feasible to calculate the derivatives of cross. This permits a green reuse of calculations with a sizeable computational time saving. Once every by-product has been computed, a gradient descend set of rules is carried out to transport the charge of go downwards, accordingly developing the dnn overall performance on the schooling set. This system is called backpropagation.

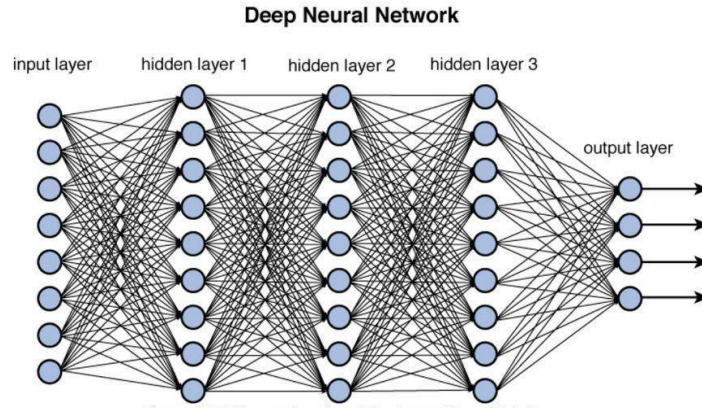


Figure: 4 Deep Neural Network

(Medium.com)

A DNN could be a multiclass non-linear classifier that passes the item options through a collection of layers and outputs a vector wherever every layer could be a vector connected to consequent one through a matrix of weights W_l and a collection of non-linear functions. If only linear equations were applied, the ultimate result would be a multivariate logistic regression. That happens because of a mix of specific functions is an active line. So as to feature non-linear, the sigmoid operate ought to be used when every linear integration. the utilization of a linear classifier within the final layer is common to DNN and SVM with the radial basis operate kernel; each of those algorithms expand the options area and use linear classifiers to get non-linear classifications within the original area. By modifying the weights and biases of every layer, they dispensed this optimization, as well as the output one. To understand using modify weights and biases, the gradient of cross is obtained. By undermine the multivariable chain rule, it's attainable to calculate the derivatives of cross. This helps to re-use the pc with efficiency by saving heaps of calculation time. Once every derivative has been calculated, a gradient descend formula is applied to move the value of cross downwards, therefore increasing the DNN performance on the training set. This procedure is called backpropagation. (Lolli et al,2019)

Researchers like (Samiul et al ,2020) and (Makkar et al,2019) also found that the normal performance of the ML methods did not exceed traditional methods, but the SVM was trained in a series of demands, it produced forecasts that are more accurate. The same researchers expanded their research activities using SVM and NN. They found that the techniques for using ML models provided significant improvements compared to traditional models. The results of that study showed that the performance of the SVR forecast was higher than the Radial Basis Function (RBF), as the SVR produced less square results of the estimated error and higher clarity of the buyer's predictions. To reduce procurement costs and asset management costs, the risk management framework uses the Bayesian Belief Network. They also suggest using different ML algorithms such as SVM and NN to ensure possible performance improvements. Competition between different ML strategies provide us high accuracy forecasts that improve the decisions needed to increase revenue. The author also talked about the cost-benefit of backorder prediction. However, they do not discuss the indication of the backward condition in this paper. In the study they develop

an order system model based on performance monitoring policy which use ARIMA models, Theta method, and many interim integration strategies. Decision trees are the supervised learning strategies that they use in order to solve spatial planning and energy distribution problems. Each branch of the decision tree is highly traceable, and is easy to translate the tree model without the expert's knowledge in the forecasting industry. These hallmark symbols make them popular among organizational decision-makers to solve various decision-making problems.

5. Machine Learning Applications

According to (Wenzel et.al, 2019) the machine learning applications in real world are mostly developed as part of a bigger project. During this period, many process model frameworks are implement for this kind of projects. The process defines six stages with specific tasks. Beginning from "Business Understanding" to "Data Understanding", "Data Preparation", "Modeling", the model "Evaluation" and the end with "Deployment". The scope of the framework is to improve results while minimizing the duration and the cost for the project.

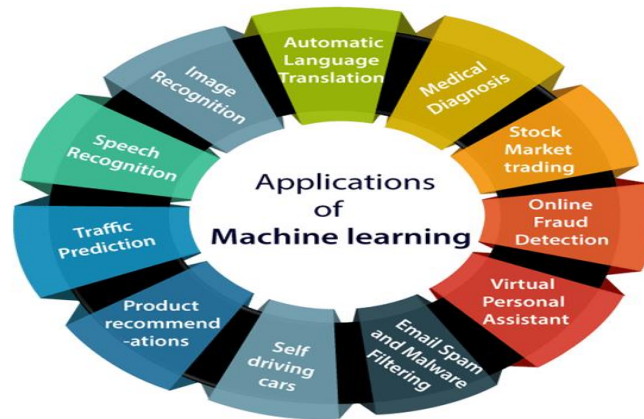


Figure: 5 Machine Learning Applications

(Javatpoint)

5.1 Machine learning and Industry 4.0

The fourth industrial revolution (Industry 4.0) has projects and technologies that should be used to increase the competitiveness of industrial firms with a focus on communications, digital production, and automation. In this context, Smart Logistics helps at the successful implementation of chains that provide intelligence and reliance based on agile networks and partnerships with affiliated firms. In addition, the information exchange is made with the use of modern information and communication technology (ICT), data networks, characters and sensors, and automated diagnostic technology and asset tracking technology. The automated transportation, conversion, and storage systems, supported by private transport vehicles, should enable in some way by having the complete control of systems. Finally, Smart Logistics can be established by using the technological aspects of cyber-physical systems (CPS), Internet of Things (IoT), and the industrial internet (IIoT), and virtual Internet (PI).

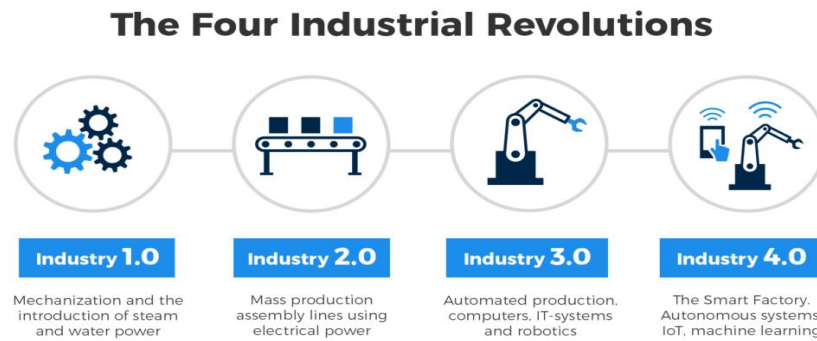


Figure: 6 the four industrial revolutions

(knowledgepublisher.com)

(Woschank et.al,2020) in their research study about the use of artificial intelligence, machine learning, and in-depth learning concepts which can be considered as very serious factors in success in the course of digital transformation. Will be use the results of this study as a base for future research on the use of AI, ML, and DL technology in the area of Smart Logistics in industrial enterprises, and provide a framework for industry-leading staff to better modernize technology. Therefore, it is important to integrate the various areas of research, e.g., information technology, material management, mechanical engineering, industrial engineering, mathematics and mathematics, into future research projects. In this case, AI is the science and engineering of intelligent machines with a special focus on intelligent software. ML is an integral part of AI, which refers to the automatic detection of logical patterns in data sets. ML tools has the purpose to increase the lustiness of algorithms by ensuring the ability to learn and adapt according to big-data analytics. In addition, DL, is defined as a category of ML within AI-technologies that explores multiple layers of information processing that is not compatible with output and / or uncontrolled output and transformation features, as well as pattern analysis and classification. In nowadays, AI, ML, and DL have gained increasing importance in many sectors of research such as engineering, medicine, economics, and business management. Therefore, the authors conduct a literature review based on AI, ML, and DL projects implemented over the period 2014 to 2019. In the sector of Smart Logistics, the use of AI, ML, and DL methods is in the early stages of development. Most of the already known subjects are concepts, laboratory tests, or in the first stage of testing. Complete applications for mature industries are not yet available. However, continuous reporting of mechanical settings, machine circuits, high-quality settings, speculation, decision support systems, advanced planning, and planning in the sector of inventory research, flow problems, traditional store planning problems, production process, and to make better the performance of operating equipment, e.g. detection and tracking methods, can be seen as promising areas within the scope of Smart Logistics.

5.2 Big data analytics in supply chain

Several papers mention the usage of big data in procurement research and discuss the use of three main stages of big data analytics in supply chains. This includes descriptive, speculative and descriptive analytics. Supply chain metrics, such as total stock, turnover per customer and year after year differences in sales would be some examples that provide historical views regarding the

company's financial and operational progress. Predictive analytics contains also the use of predictive algorithms in order to forecast the future state of affairs in supply chains. Like the forecasting of the customers' behavior, these are the purchasing patterns and identifying trends in sales activities. Similarly, the definition of forecasting statistics includes marketing and financial aspects of supply chains, such as the telecommunications sector, the trading of exchange rates. Defined analytics include building in predictions to increase the current state of the material in order to take the step to the most desirable stage.

According to (Brintrup et al,2020) research, one example is the, prediction of instabilities in traded products. This may cause adjustments in price or inventory level, using the data of demand that each customer has per product may cause us to optimize the spare parts inventory for maintenance, traffic data may cause optimizations in the logistic routes in order to avoid predicted traffic disruptions, or demand data to warehouse inventory adjustments. Depending on the internal production system, must connect to all external partners including suppliers and customers, to evaluate the use of large amounts of data in asset management and automatic asset management. Similarly, use analytics to optimize inventory-ordering decisions. It is important that in above examples they use the term supply chain freely as the scope of analysis includes all business structures with strategies, strategies and operations, including various business activities including finance, marketing and sales, while others have focused on the more confined side of supply chains contained logistics and procurement. This may be due to the fact big analytics in some operations appear to affect procurement management, and in part because the purpose of data analytics is understood as combining both multiple business operations and data from a supply chain by merging data together. Except these sectors, descriptive and predictive data analytics in supply chain have gain the least attention, whilst optimization has been the most popular.

5.3 Different machine learning applications

5.3.1 Supervised Machine Learning Application for effective supplier classification

In nowadays, global supply chain, needs incorporation of information technology and increasing of competitiveness in order to have an effective supply chain management. As (Ozden et.al, 2009) mention in their research the SCM is take into account as a competitive strategy that connects suppliers and customers with the scope of making better, more flexible and responsive the organization. Supplier assessment has an important role in satisfy the strategic aims of the firm. Identifying the right suppliers promotes corporate competition and decision-makers believe that supplier testing is a very important function of the procurement department. Procurement is an important contributor to achieving the company's strategic goals, but needs vary according to customer preferences. In today's world of uncertain consumer preferences, market volatility, and new procurement, policies require a quick and comprehensive process of testing suppliers across all organizations. Therefore, supplier testing is one of the most significant steps in the procurement process. The supplier testing process is an ongoing process and the process of changing the testing of each product and organization, the supplier trial process helps to lower the risk nd uncertainty by increasing the total value of the procurement network. The type of decision-making is entirely dependent on the process used to select suppliers, which can be of quality and quantity. Due to the dual nature of decision-making, provider testing is considering as a multidisciplinary process (MCDM). In order to solve the MCDM problem, the decision taking analysis based on human

choices and on certain factors. The weights given by human judgments are subject to the decision maker and the end result remains biased. Finally, the effectiveness of making enhanced decisions regarding the supplier assessments by incorporating appropriate ML programs that are able to provide effective feedback during decision-making.

5.3.2 Machine Learning Application for sales forecasting

We can divide sales forecasting methods into judgmental processes, exclusivities and causal mechanisms. Add-ons methods only use data of the actual task time series in order to generate a prediction. Some strategies range from a simple distance with a smooth, clear family to a complex problem like the Box-Jenkins approach. While identifying and adding trend-timing trends, time of year and autocorrelation effectively, we do not take in mind external factors such as price changes and promotions. VAR models extend the Box - Jenkins modes to include other variables, but their complexity makes balancing difficult. Because of the prediction inclusion the development of measurement models using inputs to represent the events that they believe are the outcome driver; it can be as simple as a model backing line with a variety of promotions. ML models like the SVM or neural networks; they do not mention the specific relationships between variables and include a search with a functional form space and parameter measurement. Marketing books have found a number of things that will greatly drive promotional sales. This includes the size of the depreciation of the proposed item in the frequency of the same product promotions. The relationship of the last two factors are close enough to the foreclosure of a product promoted by the same organization. Obviously, to get better predictions where promotions exists, it is important to systematically collect and use data about promotional features. Note that the main purpose of marketing studies is to find the main outcomes that contribute to more sales, rather than predicting the specific amount sold. The authors find out in their research that the Holt Winters exponential smoothing model is an effective method under stable economic conditions, and that advanced network models work better when faced with strong offline and seasonal patterns. Compare neural networks and the performance of multiple logos model in predicting product allocation - rather than sales - in food categories and find that neural networks work better. They use the information, price, and promotions of each product as inclusion. In addition, they remove and season the timeseries date in order to improve significantly the Neural Network performance. Use the SARIMA (ARIMA seasonal) model in the trading time series and train the neural network to predict future residues using the residuals from the last k periods, as well as the inclusion of promoters and special day dummies to predict SKU sales at a supermarket. They found that the SARIMA hybrid model and neural network work better than any other single model. They propose a hybrid-forecasting model that is based on compounding and classifying items according to their marketing behavior using tree classification to predict sales in the textile market.

5.3.3 Stock-out Prediction Algorithm

Oroojlooy (2019) develops an approach in order to make stock-out forecasts for multi-echelon networks when historical data are available. Author creates the algorithm based on the deep learning and more specific in DNN. DNN is a non-parametric ML algorithm that means it does not make strong estimations about the functional relationship that has the output with the input values. In the space of supply chain, author implements the DNN to demand forecasting and quantile regression. By taking on mind that a multi-echelon supply chain network with x nodes,

with inconsistent structure. For each one of the nodes in the network, he has an idea of the history of its IL, as IL he defines the owned and stored inventory of which will deduct the backorders, and of the inventory-in-transit, which is the goods that have been shipped to the node but have not arrived yet. However, author makes the DNN algorithms to work with limited inputs; in contrast, the vector changes its size in every step. Therefore, when we use them we only take in mind historical data from more recent times than the full history. This leaves some information that can be useful in the network, it also decreasing the input size, and choosing a k that is big enough and provides a good information level about the whole system. Moreover, by not keep all the available historical data, ignores the effect of any outlier observations after k periods. A DNN is a network of nodes, starting with an input stage (representing the inputs), ending with an output stage (representing the yt vector), and one or more stages in the middle of the time. All the nodes use a mathematical function, called an as an activation function, to turn the input they receive into the results that send to the next stage, with the aim of keeping a close relationship with the input and the output results. In a fully connected network, all the nodes of each stage are connected to the nodes of the next by other coefficients, called metals, which are established randomly. Network "training" involves prediction of the correct values for those weights, by the use of offline methods. The weight loss function is been used in order to check the quality of a given set of weights. The loss function calculates the distance of the forecasted values and the already known output values. Author takes in mind the following loss functions, which are mainly used for binary outputs such as Oroojlooy presents:

- Hinge loss function
- Euclidean loss function
- Soft-max loss function

The DNN algorithm provides one forecast, where false positives are good and those with the same weight. However, the character should be able to control the probability of predicting stock outages, that is, the balance of the false positive and the false negative errors. To this end, will benefit from a loss function that can provide control over the probability of stock forecasting, as the release of DNN is directly affected by its loss function. The loss functions mentioned above do not have a weighted factors, and set the equilibrium ratio between select 0 (forecasting no stock-out) and 1 (forecasting stock-out). To remedy this, suggest to estimate the amount of work lost in each output, 0 and 1, using the weights cn and cp , which represent the cost of positive and negative errors, respectively. In this way, when $cp < cn$, DNN tries to have a small number of cases where it returns False but is actually 0, so it forecasts an end to stock that will provides a small number of false positive errors and a large number of positive errors. Similarly, when $cp > cn$, DNN forecasts a small stock expiration to avoid cases where it returns True but is actually 1. Therefore, make a small number of positive errors and a large number of negative positive errors. If $cn = cp$, the revised loss function has the same results as the original loss function.

5.4 Machine learning applications to SCs and manufacturing

According to (Ian M. Cavalcante et al,2019) and (Hokey Min,2010) papers, we see that the ML can be applied in flexible SCs. Application of ML to the dynamic selection of replenishment policies according to SC environmental dynamics. They apply ML techniques in order to notice

bottlenecks, bad tasks and events to realize adequate production rescheduling propose framework supported data processing for job shop planning issues (JSSPs) that identifies the essential parameters and states of explicit dynamic planning environments. They also develop associate RL primarily based Q-learning formula (QLA) and notice that the QLA performance is superior to the present policies and apply associate RL approach to resolve a dismantlement line leveling drawback with uncertainty. Implement a hybrid methodology that integrates ML with multi-criteria decision-making techniques so as to execute multi-attribute inventory analysis. The authors enforced naive Bayes, Bayesian network, NN, and SVM models to forecast categories of at first determined stock things in an exceedingly large-scale automotive company. Another application of ML to producing is prediction of LT and cycle time key performance indicators. Most production designing and programming strategies suppose LTs. The authors perform associate LT prediction supported regression algorithms for a true flow-shop atmosphere exposed to frequent changes and uncertainties ensuing from the dynamic client order stream. use SML approaches to perform LT prediction supported historical production information obtained from producing execution systems. CT prediction one in every of the foremost necessary problems for production coming up with in terms of maintaining high delivery liableness in semiconductor manufacturing systems. Technologies like oftenest identification (RFID) and Bluetooth low energy devices, e.g., beacons, modify the gathering of information pools from producing shop floors. The authors use nearest neighbor, weighted k-NN and Bayesian reasoning techniques. Their analysis suggests that ML techniques effectively will not to harness detector systems for improved operational use cases. Similarly, use RFID technology to capture period production knowledge then apply two ML techniques: k-means cluster and gradient descent improvement. The authors state that valid predictions regarding the expected overall producing time for a given variety of producing batch inputs is obtained. ML has been wont to improve producing at the method level. three bunch algorithms are compared – k-means, hierarchical collective and Gaussian mixture models – in terms of their contribution to spindle performance data throughout high outturn machining operation. additionally use SVM and DT integrated with a mathematical programming approach to supplement existing provider choice strategies in an exceedingly biomass-to-biofuel SC. though ML isn't a positive methodology for all industrial issues, encouraging the appliance of learning algorithms will contribute to the action of autonomous production systems. Therefore, since ML provides intelligent outcomes from knowledge, an in depth follow up during this analysis field is prime to innovation in an exceedingly resilient data-driven producing setting. Despite of serious advances in ML applications to SC and operations management achieved recently, the literature doesn't specify directions on a way to build use of digital knowledge and to utilize the ML benefits to make resilient provider portfolios. As a result, it's not nonetheless clear however ML will contribute to the abstract and technological frameworks of resilient provider choice. This additionally implies that the causes of SC performance perturbations thanks to disruptions in offer base haven't been entirely disentangled from the chance profiles of provider performance. During this paper, they tend to introduce a brand new approach to resilient provider choice that utilizes the advances in knowledge analytics whereas avoiding two major inconveniences, specifically the requirement to estimate the chance of disruptions and foretelling the performance impacts. One problem in managing the resilient provider portfolios mistreatment disruption chance estimations could be a relative rarity of risk events that area unit too intermittent and irregular to be accurately

known, estimated, and forecasted. rather than estimating chances of extremely unpredictable events, the stress of their study shifts to utilizing the benefits of digital knowledge in good producing systems to predict the provider disposition to disruptions, and therefore the associated impact on SC performance. They direct a particular focus of research toward the resilient provider choice in digital producing. The take a look at cases were performed in an exceedingly digital make-to order producing setting employing a simulation tool. The results indicate that the employment of SML algorithms will support the resilient provider choice decision-making method, resulting in a lot of sure delivery from suppliers and enhancements in risk mitigation decision-making. the appliance of this approach needs a modification of outlook relating to the customer-supplier relationship, which means that these relations ought to be a lot of transient and data-oriented in order that resilient provider portfolios is developed and resilient SCs is achieved. two vital contributions emerge. First, indicate that deviations from the SC robust performance profile and supplier risk profiles is clearly outlined. by a mix of SML and simulation. Second, the results of this study improve understanding of however ML and impersonation is combined and the way to form digital SC twins, and thru these twins improves resilience. The result of this study will emerge in an exceedingly variety of helpful insights for managers like a development of most important suppliers, re-engineering of provider base, investments in SC resilience, order allocation improvement or perhaps a procurement of a risky however vital provider. The findings counsel that this model is important in revealing latent, bad provider portfolios, and prioritizing risk mitigation efforts. within the experiment, the suppliers had restrictions on production capability in bound periods and were diagrammatic in an exceedingly dataset divided by classes, like order date and order amount. The SML model was ready to predict the performance of the suppliers once variations in these categorizations had occurred. Then the employment of SML will contribute to provider choice as a risk mitigation strategy that would assist improvement and resilience management models. With the arrival of huge knowledge handiness, decision-making in producing can become more and more enthusiastic about applied math strategies. Hence, it's essential to pave the method for exchange abstractions with ML models in producing risk management processes, in order that price creation is perceived by practitioners and real knowledge shared, resulting in a virtuous circle of improvement. Finally, some limitations and future analysis avenues is also highlighted. First, the benefits of mistreatment ML techniques will become a lot of evident once considering larger knowledge sets. They manifest those benefits in quicker process times and higher relation recognition as compared to ancient applied math strategies. Since the spatial property of our knowledge set is kind of little and restricted to two parameters (i.e., delivery time and quantity), different applied math strategies might are used for our specific model, however on the opposite hand, such strategies couldn't be possible in real applications. In real provider databases, there would be multiple parameters within the SC resilience analysis. The employment of ML might suit higher to such associate inflated complexness and might be important at producing corporations with a data-driven culture. Second, though the model considers random variations to approximate to a true case, the model continues to be supported fictional data: the results area unit subject to variations in real case eventualities. For real case applications in data-oriented corporations, many options can exist thanks to the rise in knowledge handiness. In these cases, previous feature choice is wont to determine the foremost relevant options within the prediction model, or deep learning techniques ought to be thought-about. Thereto finish, the

simulation model is extended by adding product variability, transport prices, and different made-to-order options. additionally, it's attainable to analyze totally different SML algorithms, further as new strategies of mixing two or a lot of those algorithms whereas considering the individual accuracy of every.

In their research (Wenzel et.al,2019) present one example case study of the demand forecast model implemented in DM one of the largest drugstore companies in Germany, DM, uses ML algorithms to predict future needs. With 3,350 stores worldwide, the DM drug store chain has six distribution centers to ensure the availability of goods. Distribution centers should ensure that the incoming goods of industry partners are organized in individual stores in such a way that no product shortages and customer satisfaction remain high. Usually, stores require delivery of products within a short period of time, but manufacturers have longer delivery times. This remains a major challenge for distribution centers. Well-planned planning goes hand in hand with long-term orders, leading to higher storage costs and revenue consolidation. To overcome this problem, they use artificial intelligence algorithms to create weekly demand forecasts based on the SKU level for 6 months. In algorithm training, they use the data from 2.5 years ago of a particular distribution center and attention was paid to the annual feature. As a result of these artificial intelligence algorithms, demand forecasts are so well-defined over a period of six months that significant improvements in forecasting quality can be achieved and industry partners are now able to plan ahead. The reliability of the delivery and availability of the product can thus be greatly improved, which is supported by the automatic flow of information for future needs of industry partners. The benefits of using algorithms are obvious to industry partners and the DM drug market. Industrial partners can rely on increased security planning and ordering security through valid predictions and the drug store chain has secure product availability and much lower stocks. Results customers' satisfaction with low cost.

Several works like (Paolo Priore et al,2018), propose machine learning-based frameworks for managing the inventory at all nodes of the supply chain in a coordinated manner. Their solutions employ different algorithms for reinforcement learning, e.g. Q-learning, to determine near-optimal ordering policies. They use simulation methods to test the effectiveness of the supply chain in a variety of contexts. Proposed solutions take decisions based on the vector state system, which is often defined as the building of a list of different supply areas. In these activities, the learning-based approach is demonstrated beyond the specific measurement policies. The former has used learning reinforcements to determine the complete reseller policy for the full seller. Their solution, when considering two products, includes the number of trucks shipped by the delivery center to the vendor set. Lastly, it focuses on determining the appropriate manufacturer's production policy. They use case-based thinking using a continuous algorithm close to K. These works point out that a learning-based approach effectively increases the profitability of procurement rather than traditional methods. The usefulness of ML to manage the flow of goods through the automatic configuration of the supply chain has also been investigated. In both cases, this flexible approach, which facilitates the configuration of learning-based strategies, significantly achieves a single set of financial shooting. Finally, several authors evaluate the effectiveness of these demanding methods, which are an important part of asset management. They contribute to the literature by creating a learning-based framework for setting the most appropriate replenishment policy over

time in dynamic environments. They design their solution to respond to environmental changes; therefore, by considering the scope of both internal and external factors, in contrast to previous works in this field. Although there is a very high level of expertise, that use informative learning as it creates a complete understanding of decision-making. In this sense, decision trees can be interpreted as ‘white box’ systems, which allow for in-depth analysis of the impact factors; unlike many ML techniques. By using reading learning, instead of other ML strategies, as it makes sense of the decision-making process. As a result, they have gained some insight into the impact of flexibility on the suitability of asset policies. In this regard, the leading policy relies heavily on node-building costs. In addition, results indicate that appropriate policy is more sensitive to the establishment policy of higher markets than that of lower purchasing shares. It is interesting to note that the correct wholesaler policy depends largely on whether or not the merchant policy minimizes the effects of the Bullwhip Effect.

5.5 Machine learning approach to find business partners and building mutual relationships

The purpose of the papers of (Junichiro Mori et al, 2012) and (Chitriki Thotappa, 2010) finding new business partners as suppliers and customers, and building relationships between them with the help of AI programs. To this purpose, they propose a ML model and predict customer relationships with suppliers based on the clearly available profiles of firms and their trading relationships. With strong data available, they create several features that reflect customer-supplier relationships. After that they use those features, represented by a very large vector, with SVM in order to learn the model of customer-supplier relationships. As a result, the approach may automatically predict business partners who are provided with strong profiles and their existing contractual relationships.

In their paper (E. A. Beardslee & T. B. Trafalis, 2005) realize that among the old problems facing any transaction, that of deciding how much of a given item should be kept on the shelf of a store or store, and what stages of implementation of the completion of this offer are thoroughly evaluated and thoroughly researched. However, using current data mining methods to support the determination of effective stock levels and economic restructuring points remains to be fully considered. A specific area that needs research is to use the transaction history of orders, receipts, stocks shortages, and stock shortages to assist asset planners and stock managers with the task of predicting or predicting future stock deficits. Using data mining, in a “reduced” trading environment where common asset classification systems are volatile, it is possible to find patterns, organizations, classes or categories within the transaction distribution that provides clues to subsequent stock shortages or stock losses. Using a number of current data mining techniques, we plan to review the capabilities of each method of providing reliable guessing capabilities to the inventory manager. They use each method on the same set of transaction data covering three years of stock bench operation. This transaction set contains orders, receipts, stock expiration, and stock shortages. Therefore, the timing of the removal or removal of an item from the inventory is unknown. The lack of this data point strengthens the problem space by eliminating significant transactions; otherwise, finding the expected demand becomes very difficult.

In their research (Makridakis et al, 2019) mention that there are inconsistencies in the performance of ML methods for demand forecasting. For example, some experts have also concluded that

traditional or mathematical methods such as exponential smoothing and Holt Winters exponential smoothing have produced relatively accurate or sometimes better predictions than ANNs. In any predictive attempt, there are three sources of uncertainty, including model uncertainty, parameter uncertainty and data uncertainty. A potential cause for inconsistency is the inability of pure and unconventional methods of seeking predictions caused by the inability to manage all sources of uncertainty. This prediction has been used for many studies by combining various ML methods and mathematical models to establish hybrid strategies such as ARIMA integrated with ANNs. Repeatedly, they find the evidence in the rise of integrated predictions especially in combining traditional time series with machine learning methods. Nowadays, much attention has been focused on using AI and ML to predict demand. However, there is little evidence of research into the effectiveness of these methods in improving acquisition performance and accounting inconsistencies with ML-based forecasting methods. Specifically, using an ML-based approach produced mixed results in a consistent and complex database. Also in this study, an ML-based demand predictive model is developed to address this lack of understanding using the corresponding database. Using observational data collected from a steel company, we analyze the effect of the new method on climate accuracy, asset reversal, and the currency conversion cycle and its interactions as procurement performance metrics.

5.6 Hybrid demand forecasting models

Javad Feizabadi (2020) in order to develop the ML-based forecasting model rests on two established models: 1) autoregressive integrated moving average with exogenous variables (ARIMAX) and 2) two-layer feed forward NN with backpropagation learning. ARIMAX is similar to the generic ARIMA model with the inclusion of exogenous variables, such as the macro-economic factors. The basic ARIMA (p, d, q) is the generic form of the time series models that are commonly used by econometricians. It models the time series as autoregressive (AR), integration or differencing order (I), and moving average (MA) components. The parameters of (p, d, q) represent the number of time-lagged in AR, the number of differences as I, and the number of time-lagged forecast errors, respectively. Overall, ARIMA models have performed inferior in demand forecasting against exponential smoothing methods (Makridakis et.al, 2019) and both methods grounded on the premise that future demand is a function of the past. However, ARIMAX is based not only on past demand time series but also on leading indicators time series. By analyzing ACF and PACF of the timeseries they observe a significant autocorrelation in AR(1) and AR (3) series while the correlation was not sizable in MA series. Thus considering ACF and PACF, the p and q parameters were set at 1 and 0 in the first model and 3 and 0 in the second model suggest that if ARIMA models are used for forecasting, a high differencing order is not recommended and most time series can be modelled with no more than two differences (i.e. d , 2) and AR/MA terms up to order 5 (i.e. p, q ≤ 5). The ARIMAX (1,1,0) model is developed following the rule of minimizing the corrected AIC. The AIC, in general, is an estimate of lost information. It optimizes the trade-off between the model goodness-of-fit and the simplicity of the model. The ARIMAX (1,1,0) model is specified as:

$$y_t = aFt Byt - 1 G(yt - 1 - yy - 2) \#j = 1djwj et$$

Where y_t is the demand at time t , F_t is the time-series forecast using Holt-Winter's method at time t , a is the coefficient for the term F_t , y_{t-1} is the demand lagged by one time period, b is the y_{t-1} coefficient, g is the coefficient of the difference term ($y_t - 1 - y_{t-2}$), w_j and d_j denote macro-economic factors and their coefficients respectively and e_t is the error term. The better fit of higher-order ARIMA demand process in upstream of supply chains is also observed in prior research and it is largely due to demand variance amplification propagated from downstream to upstream of the chain. Thus, the ARIMAX (3, 0, 0) model can be specified as below:

$$y_t = aF_t + 3i = 1biy_{t-1} - i + 3j = 1gjw_j + e_t$$

As one of the deep learning methods, the two-layer feedforward neural network is a popular ML algorithm used in statistical modelling, pattern recognition, etc. A feedforward network characterizes a mapping $y = f(x; u)$ and learns the value of u that provides the best function approximation. The information flows through the function being evaluated at x , through the intermediate computation used to define f and finally to the output y . The two-layer structure of the network can be specified as $f(x) = f_2(f_1(x))$. Because the training data doesn't show the value of other layers than output layer, the other layer is called hidden layers and activation functions should be used to compute the value of hidden layers. The artificial neural networks handle the nonlinear relationship between input variables and the learning occurs as the algorithm compute the gradients of some complicated functions. Back propagation algorithm is used to compute the gradients.

The NN model has a similar input variable structure to ARIMAX, including both AR term and exogenous variables, along with the Holt-Winter's forecast. The output of each neuron is determined by the following equations: $v_i = \sum h = 1w_{ih}x_h$. Where v_i is the summation of the weights w connecting to the inputs x for neuron i . They compute the weights by using backpropagation learning algorithm. The error is calculated at the output and distributed back through the network layers. The final weights are determined based on minimum possible error at the output. There are m inputs with corresponding weights in the previous layer h . If layer h is a hidden layer, each x_h is an output of a neuron from

the previous layer. Then we have: $Y_i = s(v_i) = \frac{1}{1 + \exp(-v_i)}$

Where Y_i is the output of neuron i , $s(v_i)$ is the sigmoid activation function and a is the constant which influences the gradient of the function. In this study, two hidden layers, each with 5 and 2 neurons, respectively, were used to generate the forecast. Two-layer was specified in the model to reflect the more in-depth learning of the algorithm. The number of neurons in the hidden layers is typically within the range of input to output neurons (7 and 1, in this case). In training and testing stages, the 5- and 2-neuron hidden layers structure gave the best forecast accuracy when compared with 6- and 3-neuron hidden layers, 5- and 3-neuron hidden layers, and a single 4-neuron hidden layer models.

6. Data in supply chains

(Seyedan et al, 2020) in their literature review point out too several sources of big data within the supply chains with many varieties of balances in variety, value, velocity and veracity attributes.

Although, the predictions needed for supply chains are of low volume limits, velocity, and variability, however, these predictions will use information from several sources altogether supply departments from low-volume / varied / velocity on-the-shelf setup reports. Trailing of volume / variability / velocity given by IoT. This mixture of information sources utilized in SC seeks prediction, with its varied temporal and abstraction options, places nice stress on the utilization of huge amounts of data in supply chains, mainly, and prediction efforts, especially. The big data analytics functions in supply chain demand prediction was mentioned in each varieties of ML, supervised and unsupervised learning. In supervised learning, data are going to be connected with labels, which means that the inputs and outputs area unit specific. The supervised learning algorithms giving the mentioned relationships that has to do with the inputs and outputs in a try to point out the inputs to the regarding outputs given a replacement unlabeled dataset. As an example, for instance case of a demand-driven learning model for demand prediction, future demands are often expected supported historical data on product demand. In unsupervised learning, data are unknown, and therefore the BDA algorithms attempt to realize the underlying patterns among them, the inputs and their analysis of interrelationships. A good example of unsupervised learning in supply chains that integrates very different teams of consumers consistent with their similarities is the client segregation. Multiple machine learning / data algorithms will create it easier for supervised learning (excluding input-output relationships) and unrestrained reading (input, output and their relationships).

6.1 Demand management in supply chains

Moreover in (Seyedan et al,2020) research they outline the 2 different ways for demand management. A forward approach that appearance at potential demand over following many years and a backward approach that depends on past or current capabilities in responding to demand. In forward demand management, the foremost necessary factors are the demand statement and planning and marketing methods. Demand prediction and planning check with prediction the timings and the quantities of customers' demands. Such statement aims to realize client satisfaction by meeting their desires in an exceedingly timely manner. Correct prediction of demand will improve the potency and hardiness of production processes (and associated supply chains) as resources are going to be tailored to the requirements that cause the reduction of inventory and waste. Based on the above facts, there are variety of projected strategies within the literature and in conducting prediction and order planning. Program models, spreadsheet strategies (such as moving averages), and benchmark-based judgments are among these strategies. Today, the foremost wide used statement and editing tool is excel. The most well-known issue where the spreadsheet models used for predicting demand is that they cannot be enclosed in big data. Additionally, the complexness and uncertainty in SCM (quantity and variability of would like and provision) can not be underestimated, analyzed, and handled by easy mathematical strategies like measuring or visual fluency. Over the past decade, traditional SC solutions look for to predict and set up and face several challenges in reducing prices and reducing inventories. In several cases, the planned solutions have improved day pay; raise the price of SC as a burden to suppliers. Big data time and highly calculated analytics have enabled you to manage data with a high potency, speed, simplicity, and reduced considerations regarding data storage and assortment for cloud services. The importance of latest technologies in data storage and analytics and the proliferation

of high-quality data have created new opportunities for predicting and programming required data. They improve the search for prediction accuracy by data-mining algorithm techniques and tools that may filter data, analyze results, and find out about the relationships concerned. This may cause much-needed prediction models that learn from data and should be accessible to SCM. Within the next section, a review of BDA applications to SCM is given. These programs are compiled supported the strategies went to establish data drives predictions.

6.2 Big Data Analytics for demand forecasting in SCM

(Seyedan et al,2020) found that there's a steady increasing trend within the scope of articles published from 2005 to 2019. It's expected that such growth continue in 2020. Reviewing the past fifteen years of analysis on big data analytics and ML applications in SC demand prediction, they tend to known sixty-four research papers and classify them with reference to the methodologies they implement for the demand prediction. It shall be mentioned that there have been many articles exploitation multiple of those techniques. Many papers study the utilization of big data analytics in SCM. However, this analysis points out the vital theme “demand forecasting” in SCM to explore BDA modules in line with this explicit subtopic in SCM. Typical ways have featured variety of limitations for demand forecasting within the context of SCs. There are many factors, which affect the demand in supply chains; however, most of them weren't captured in studies mistreatment typical ways for the sake of simplicity. During this, the forecasts may solely give a partial understanding of demand variations in offer chains. Additionally, the unexplained demand variations can be merely thought-about as applied math noise. Typical approaches may give shorter process times in exchange for a compromise on lustiness and accuracy of predictions. Typical SC demand prognostication approaches are principally made manual with big reliance on the planner’s skills and domain data. It might be worthy to modify the forecasting method to cut back such a dependency. Finally, data-driven techniques may learn to include non-linear behaviors and will so give bigger estimations in demand forecasting in comparison to traditional ways that are principally derived supported linear models. There’s a big level of non-linearity in demand manners in SC significantly because of the big competition among suppliers, the bullwhip result, and pair between supply and demand. To extract valuable data from a huge quantity of data, BDA is employed as a complicated analytics technique to get the info required for decision-making. Reduced operational prices, improved SC nimbleness, and augmented client satisfaction are mentioned among the advantages of applying BDA in SCM. Researchers used varied BDA techniques and algorithms in SCM context, like classification, situation analysis, and optimization. Machine-learning techniques are wont to forecast demand in SCs, subject to uncertainties in costs, markets, competitors, and client behaviors, to manage SCs in an additional economical and profitable manner. BDA has been applied all told stages of offer chains, as well as acquisition, storage, logistics/transportation, producing, and management of sales. BDA includes the descriptive analytics, predictive analytics, and prescriptive analytics. Descriptive analysis is the process that describing and categorizing what happened within the past. Predictive analytics are accustomed predict future events and see predictive patterns inside data by using mathematical models such us data processing, web mining and text mining. Prescriptive analytics use data and mathematical models for decision-making. Multi-criteria decision-making, optimization, and

simulation are among the prescriptive analytics tools that facilitate to enhance the accuracy of forecasting. Predictive analytics are those principally used in SC demand and acquisition prediction.

(Lauer and Legner,2019) with their research contribute in the application of ML on master production programming, which was gaining small attention by then. With this research, the stability factor proved suitable. However, the accuracy of the predictive results shows that the influence by the strong relationships and demand in advanced ML is big, leading to high accuracy in the third quarter of the planning period. To can be able to take a deviation from this pattern by the impact of other factors, continuous data and features are required, e.g., of volume. Another solution would be to study different models of different parts of the editing limit. This could mean that algorithm-training data is separated to produce individual models. They test some algorithms predictions accuracy compared to other mathematical prediction methods. With regard to mitigation of instability, the company must decide the way to set a target for the sale of sustainable production and response systems in a changing environment. Typically, this approach can be used to make decisions, where appropriate systems are not suitable for simulation and modeling of statistics. Tests and results promote the basic concept and field of application, but ask for further analysis.

6.3 Big data in manufacturing: a systematic mapping study

O'Donovan, (2015) in his research note that modern production facilities are data-rich areas that support the transfer, distribution and analysis of information across networks that generate productive intelligence. Potential benefits of manufacturing ingenuity include improvements in efficiency, innovation, and environmental impact, to name a few. However, like other industries and domains, current information systems that support business intelligence and productivity are tasked with storing more large data sets (Big Data), as well as supporting real-time processing of this 'Big Data' using advanced statistics. The predicted exponential growth in data production will be a result of an increase in the number of instruments that record measurements from physical environments and processes, as well as an increase in the frequency at which these devices record and persists measurements. The technologies that transmit this raw data will include legacy automation and sensor networks, in addition to new and emerging paradigms, such as the IoT and CPS. The low-profile data captured by this technology can be used by analytics and modeling applications to enable producers to develop a better understanding of their functions and processes to obtain information that can improve existing functions. Focusing on high-tech information technology in production areas is a new area of research covering a wide range of fields, including automation, engineering, information technology and data analytics, to name a few. Now, it is imperative to understand the current state of research related to high-tech information technology in production, as well as to identify areas where future research efforts should be made to support next-generation infrastructure and technology. Therefore, this study aims to differentiate current research efforts, identify outstanding research themes, and identify gaps in current literature. This study uses a second known and formal research method to create a systematic map to capture the most comprehensive and varied research threads currently related to the technology of big data in practice. The contribution of this study is a comprehensive report on the current state of research involving high-tech information technology in production, including the type of research being

conducted, the areas where large data research is conducted, and the results from these large data research efforts. The research methodology employed in this study is guided by the systematic mapping process described by the combination of these reviews, combined with a systematic map presented in this study, can serve to provide a complete overview of the main research related to big data in production.

6.4 Production planning

In the fresh food industry, the small shelf life and the need to keep quality during the storage and the distribution process make demand-forecasting accuracy a serious variable for production planning. This in order to minimizing the sales lose due to a stock out of products, reducing the returns due to the approaching of the expiration date, and making better the availability to customers, by reflecting on organization results and even reducing any causing environmental damage. As for the human factor in the forecasting, shows that discriminatory and systematic errors in demand prediction often happens in the procurement decision-making process or in the sales and operational planning process, as influenced by personal judgment. There are points that neural networks and ML are part of a collection of so-called cognitive technologies, which try to mimic human thinking and that can use a lot of information for analysis, which can be used in business processes. The benefits of ML forecasting strategies in the food industry, such as supermarkets, grocery stores, restaurants, bakeries and confectionery. The major advantages noted were the reduction in human bias due to greater automation of the forecasting process, higher level of predictive accuracy, and volatile variability. Challenges in the author's companies have not accounted for the lack of detailed history information and a large number of learning algorithms available, which can make the right choice difficult. A common feature in various business sectors is the importance of marketing promotion, which is actions that encourage consumers to buy faster consumer products.

(Tarallo et al 2019) in their research mention that ML applications, together with the newest methods like the DL, as well as the combination of the various strategies used in the required prediction models, can give us advantages regarding to the production sides and sellers of fast-moving consumer goods. The main advantage achieved was a good forecast of demand. As a result, factories can plan better sales and operations plans, adjust production size and logistics, and improve inventory balancing across the supply chain. Retail shops is possible to have high efficiency in them management, stock replenishment processes, and maintaining suitable stock quantity with fewer stock outs or excesses, thereby resulting in smaller cost, increased profit, and better clients satisfaction based to the availability of clients' needed products. Comparing with simple statistical methods have shown better sales predictions, as provided by bigger accuracy than before models, the flexibility of managing large amounts of data flexibility and the ability to process large data sizes. Studies says that complicated analysis, such as the impact of promotions, is better to be handled by ML models. Well known business sectors such as fast-fashion, technology consumer products, general retail, short shelf-life foods, and general foods. Many of the business challenges of emerging sectors were the need to constantly predict, new products with no history of sales and to reduce food loss due to expiration date, which could affect sustainability in the new food sector. Because of the high digitalization in the retail sector, researches in future can study DL capabilities to handle different type of data such as pictures, data from the IoT

sensors and information on consumer behavior. These data will enable better predictions and even real-time decisions that could affect the entire supply chain. Other concerns regarding the adoption of ML predictive data may be the variety of algorithms available, thus making the choice of a good alternative difficult. New research on actual initiation cases and adoption strategies can help with that problem. Therefore, this study contributes to the identification of the benefits and features of ML, which are used to improve the accuracy of forecasting, required which is a key competitive advantage.

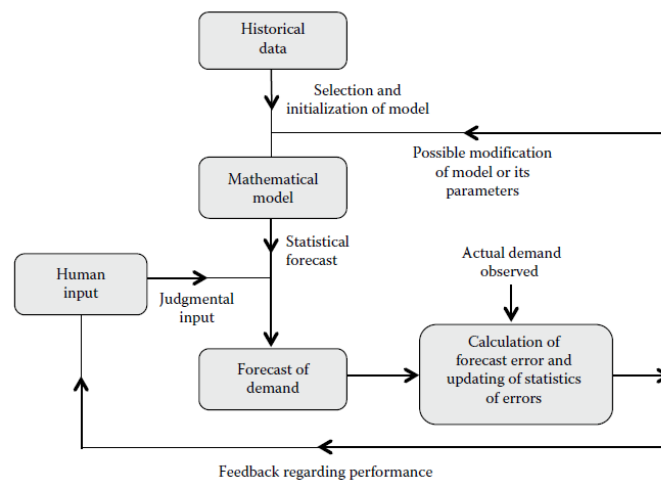


Figure: 7 Demand Forecast

(Tarallo et al 2019)

7. Timeseries forecasting

Timeseries, as defined by (Mahya seyedan et al 2020) are ways for mining difficult and sequent data varieties. Sequence data in timeseries, are made with long series of numeric data, written down at equal time points like per minute, per hour, or per day. Several natural and human-made processes, like stock markets, diagnosing, or natural phenomena, will offer timeseries datasets. Combos of timeseries strategies with product or market options have gained a lot of attention in demand forecasting. Authors plan and build a demand trend-mining algorithm for forecasting life cycle style. In their methodology, they mixed 3 models: a decision tree model made with big horizon historical data classification, a discrete selection analysis for present and historical demand modeling, and an automatic timeseries prediction model for future trend analysis. They tried and applied their 3-level mixed model in an exceedingly smartphone design, producing and remanufacturing. Timeseries approach is additionally used for prediction of search traffic subject to changes in shopper attitudes. Demand forecasting are often predicted by using timeseries models using exponential smoothing so as to create forecasts for short, mid-term, and long-term demand trends across the SC. Additionally, employing a timeseries approach for SC demand prediction with customer-responsive. Just in case of spoilable products, with short life cycles, having acceptable (short-term) forecasting is extraordinarily vital. They found that the HW model in time series contains a higher goodness-of-fit based on MAPE. What is more just in case of ARIMA model, the accuracy of predictions might show wherever it is present a big uncertainty of the future

patterns of parameters. HW model forecasting will provide us higher accuracy as compared to ARIMA. HW is easy model and simple to use, particularly in time series. However, data horizon couldn't be larger than a seasonal cycle. Otherwise, the accuracy of forecasts can decrease quickly. This happens because of the inputs of the HW model are forecasted values subject to longer-term possible inaccuracies and uncertainties.

7.1 Definition of a Timeseries

(Ratnadip Adhikari and R. K. Agrawal, 2013) define that timeseries as a consecutive set of data points, calculated typically over successive time points. It is mathematically defined as a set of vectors $x(t), t = 0, 1, 2, \dots$ where t represents the time passed. The variable $x(t)$ is taken in mind as a random variable. The values measured during an event in a timeseries are arranged in a proper chronological time order. The time series consist of a single variable defined as univariate. Nevertheless, if you look at the records of one variety, it is called a multivariate. The timeseries can be continuous or discrete. In a continuous timeseries observations are calculated at each period of time, besides a discrete time series have observations calculated at discrete points of time. For example, temperature readings, flow of a river, concentration of a chemical process etc. can be defined as a continuous time series. On the other hand, the value of a particular city, the production of a company, and the exchange rates between two different currencies may reflect a different timeseries. Frequently in a separate timeseries consecutive observations are recorded at intermediate time intervals such as hourly, daily, weekly, monthly or yearly. The observed variance in a limited time series is assumed to be measured as a continuous variance using a real number scale. In addition, a continuous series of time can be easily transformed into a non-existent one by combining data together within a set period.

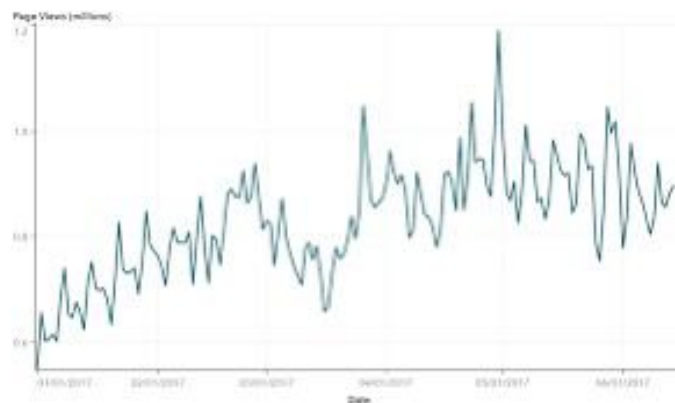


Figure: 8 Time series

(rpubs.com)

7.2 Components of a Time Series

Timeseries generally is meant to be plagued by four main parts, which might be separated from the determined knowledge. These parts are trend, cyclical, seasonal and irregular parts. A quick description of those four parts is given here. The final tendency of a time series to extend, decrease or stagnate over a protracted amount of your time is termed as laic Trend or just Trend. Thus, it are often same that trend could be a long-run movement in a timeseries. For instance, series with reference to growth, variety of homes in a town etc. show upward trend, whereas downward trend are often determined in series with reference to mortality rates, epidemics, etc. seasonal differences in a time series are fluctuations among a year during the season. The necessary factors inflicting seasonal differences are: climate and weather conditions, customs, traditional habits, etc. For instance, sales of ice-cream increase in summer, sales of woollen cloths increase in winter. Seasonal variation is a crucial issue for executives, merchandiser and producers for creating correct plans. The cyclic variation in a very statistic describes the medium-term changes within the series, caused by circumstances that repeat in cycles. The length of a cycle extends over longer amount of your time, typically 2 or a lot of years. Most of the economic and money statistic show some reasonably cyclic variation. For example, a business cycle consists of four phases 1. Prosperity, 2. Decline, 3. Depression and 4. Recovery. A four-phase business cycle Irregular or random variations during a timeseries are caused by unpredictable influences, that don't seem to be regular and also don't repeat in a specific pattern. These variations are caused by incidences like war, strike, earthquake, flood, revolution, etc. there's no outlined statistical technique for measure random fluctuations in a timeseries. Considering the consequences of those four parts, 2 differing types of models are typically used for a time series.

- Multiplicative Model: $Y(t) = T(t) \times S(t) \times C(t) \times I(t)$
- Additive Model: $Y(t) = T(t) + S(t) + C(t) + I(t)$

$Y(t)$ is the observation

$T(t)$ $S(t)$ $C(t)$ $I(t)$ are respectively the trend, seasonal, cyclical and irregular variation at time (t).

Multiplicative model relies on the assumption that the four parts of a timeseries don't seem to be essentially independent and that they will have an effect on one another; whereas within the additive model it's assumed that the four elements are independent of each alternative. Some samples of timeseries data timeseries observations are often encountered in several domains like business, economics, industry, engineering and science, etc. looking on the character of research and sensible would like, there may be varied totally different sorts of timeseries. to ascertain the fundamental pattern of the info, sometimes a timeseries is portrayed by a graph, wherever the observations are aforethought against corresponding time. (Ratnadip Adhikari and R. K. Agrawal, 2013)

7.3 Introduction to Timeseries Analysis

According to (Ratnadip Adhikari and R. K. Agrawal, 2013), in practice a suitable model is fitted to a given timeseries and the corresponding parameters are calculated by using the known data values. The process by which we will do the fitting a timeseries to a suitable model they mention in the literature as Timeseries Analysis. It contains tasks that used in order to can understand the

nature of the series and is often useful for future forecasting and simulation. In timeseries forecasting, past data are collected and analyzed in order to create a suitable model, which consists of the underlying data and generates the process for the series. They forecast future events by using this model. This method is especially useful when there is not much information on a mathematical pattern followed by sequential observations or when there is a need for a satisfactory descriptive model. Timeseries forecasting has important applications in various fields. Often important strategic decisions and precautionary measures are taken depending on the effects of the weather. Therefore, to make a good prediction, i.e. including a sufficient model in the time series is very important. Over the past few decades, researchers have developed and refined real-time forecasting models.

8. Chosen algorithms for our research

We decide to use the below algorithms because we want to have the most well known and widely used algorithms for timeseries forecast. Moreover, we need a machine-learning algorithm and a simple statistical algorithm in order to see which method is better based on the errors we will calculate.

Forecasting machine algorithms according to Jason Brownlee (2019) belong to the following seven categories.

- Baseline is a method that uses heuristics, simple summary statistics, randomness, or simple machine learning to create predictions for a dataset. In this category belong simple statistical forecasting algorithms such as persistence and averages.
- Auto regression (AR) models, are a demonstration of random process which we use to describe certain time-varying processes. In addition, they specify that the output variable depends linearly on its own previous values and on a stochastic term. In this category belong algorithms like the moving-average (MA) model, autoregressive integrated moving average (ARIMA) and the related with ARIMA seasonal algorithms like SARIMA, SARIMAX etc.
- Exponential smoothing, are similar in that a prediction is a weighted sum of past observations, but the model explicitly uses an exponentially decreasing weight for past observations. There are many algorithms like simple, double and triple exponential smoothing. Triple Exponential Smoothing is an expansion of simple Exponential Smoothing that explicitly adds support for seasonality to the univariate time series. It is sometimes called Holt-Winters Exponential Smoothing, named for two contributors to the method: Charles Holt and Peter Winters.
- Linear ML algorithms, linear models generate a formula to create a best-fit line to forecast unknown values. Linear models can be trained relatively quickly and are generally more straightforward to interpret. In this category we see algorithms like Linear regression, which is used for regression (numerical predictions) and Logistic regression.
- Non Linear ML algorithms, are models that build a non-linear relationship between the dependent and independent variables. We use them when the data shows a curvy trend, and linear regression would not produce very accurate results when compared to non-linear regression. Some examples are Knn, decision trees, SVM etc.
- Ensemble ML algorithms, Ensemble methods are models composed of multiple weaker models that are independently trained and their forecasts are combined in some way to make the overall prediction. This is a very powerful category and is also very popular. Some of these algorithms are

Weighted Average (Blending), Stacked Generalization (Stacking), Gradient Boosting Machines (GBM), Gradient Boosted Regression Trees (GBRT), Random Forest.

- Deep learning, these algorithms are a modern update to Artificial Neural Networks that exploit abundant cheap computation. They are concerned constructing bigger and more complicated neural networks and many methods are concerned with very large datasets of labelled analog data, such as image, audio and video. The most popular deep learning algorithms are: Convolutional Neural Network (CNN), Recurrent Neural Networks (RNNs), Long Short-Term Memory Networks (LSTMs) and hybrid models.

For this reason, we decide use ARIMA as the most known algorithm for forecast in simple timeseries, from Autoregression algorithms. From the same category the SARIMAX, which is suitable for seasonality models, forecast which are affected from exogenous factors. From exponential smoothing category we choose to use the Holt Winters as one of the most known algorithms to use in timeseries forecasting and analysis. From the ensemble machine learning category we choose the random forest regressor which is a ML algorithm with high accuracy widely used in demand forecasting.

8.1 Demand prediction with ARIMA forecasting model

One mathematical approach to forecasting timeseries is well known as the Box Jenkins technique and was enforced 1st by Box and Jenkins (1970). Technically, the Box Jenkins technique is an integration of the autoregressive and therefore the moving average strategies, thus it's additionally named ARIMA (Autoregressive, Integrated, Moving Average) model. Since its 1st introduction, this ARIMA approach has become documented in several fields like specification, estimation, and diagnostic. The ARIMA methodology could be a method for analyzing and building a statement model that best represents a statistic by modeling the correlations within the information. within the research, there are many refernces of the ARIMA model were found and support the ARIMA as a correct means in particularly short statistic statement. Taking advantage of its strictly applied math approach, the ARIMA technique solely needs the historical information of a timeseries to will create the forecast. Hence, the ARIMA technique will increase the forecast accuracy whereas keeping the amount of parameters to a minimum.

The AR and MA model can be mixed and, give us a third class of general models called ARMA, a particular ARIMA $p\ q\ (d, 0)$ model. With non-seasonal differences d added to the model, the ARIMA $p\ d\ q\ (, ,)$ model has the ability to can used in a big variety of timeseries forecasting questions. Here p is the number of autoregressive terms, d is the number of non-seasonal differences, and q is the number of lagged forecast errors in the prediction equation. The ARIMA $p\ d\ q\ (, ,)$ model use combinations of historical values and forecasting errors and offer a potential for fitting models that could not be well fitted if using just an AR or an MA model itself. Furthermore, the addition of the differencing eliminates most non-stationarity in the series. A significant difference between the ARIMA methodology and previous methods is that ARIMA does not make estimations about the number of the terms or the relative weights to be addressed to these terms. In order to specify the model, we have to select the appropriate order by choosing the number of $p\ d\ q$ terms. (Zhu and Shen, 2012)

Demand prediction is important in inventory management. The product prices used to be based on demand forecasts. In fact, improper estimates of demand may result in extra costs, proving that the

process has not improved. As a result, many systems are investing heavily in demand forecasting models in order to avoid "stock-outs." Another confusing issue is that some demands could also be within the middle, which implies there's a time once we now not need it and over again, we've a series of demands. Medium requirements present many of the difficulties of traditional mathematical predictions. In general; there are many methods of predicting numbers among them in which we find tangible smoothness, for example. However, when we use these methods, must have a detailed history. Initially, there is no information about the past, using estimates based on similar conditions or developer experiences. In this case, we have a great deal of uncertainty that will be avoided in time. The variable is most often demand, but it can also be something else, such as supply or price. Forecasting is the operation of making assumption about the future values of studied variables. In manufacturing, forecasting demands is among the most crucial issues in inventory management it can be used in various operational planning activities during the production process: capacity planning, used-product acquisition management. For both types of supply chain processes "push/pull," the predictions of the demand are defined as the base of supply chain's planning.

The classical ARIMA model isn't smart in several cases, it's not possible to outline a model, once allowance order is high or its diagnostics fail to indicate that point series is stationary seasonal adjustment allowance. In such cases, the static parameters of the classical ARIMA model area unit thought of the principal constraint to prediction high variable seasonal demand. Another constraint of the classical ARIMA approach is that it needs an outsized range of observations to work out the best-fit model for a knowledge series.

An ARIMA model is labeled as an ARIMA model (p,d,q), wherein:

- p is the number of autoregressive terms.
- d is the number of differences.
- q is the number of moving averages.

An autoregressive process of the p, AR(p), is defined mathematically as a weighted linear sum of the before p values plus white noise: $y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + z_t$

Where:

- $\phi_1, \phi_2, \dots, \phi_p$ define the coefficients of the AR order.
- z_t defines the error term with zero mean and variance.

Using the back-shift operator B , $B y_t = y_{t-1}$, the AR(p) can be shown as:

$$\phi(B)y_t = z_t$$

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \dots - \phi_p B^p$$

Where: $\phi(B)$ is a polynomial in B of order P.

An MA(q) task is a weighted linear sum of the last q white noise error terms:

$$y_t = z_t + \theta_1 z_{t-1} + \theta_2 z_{t-2} + \dots + \theta_q z_{t-q}$$

Where:

- $\theta_1, \theta_2, \dots, \theta_q$ defines the coefficients of the MA order.
- z_t defines the white noise terms with zero mean and constant variance.

Using the back-shift operator B , the MA(q) can be written as:

$$\theta(B) = 1 + \theta_1 B + \dots + \theta_q B^q$$

Where: $\theta(B)$ is a polynomial in B of order q .

An ARIMA (p, d, q) is a multi-autoregressive moving average model of p autoregressive terms and q moving average terms which are differenced d times:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + z_t + \theta_1 z_{t-1} + \theta_2 z_{t-2} + \dots + \theta_q z_{t-q}$$

In other words: Forecasted y_t = a constant linear combination of lagged values linear combination of current and past error values

Using the back-shift operator B , the ARIMA(p, d, q) can be defined as:

$$\phi(B)(1 - B)^d y_t = \theta(B)z_t$$

Where: $\phi(B)(1 - B)^d$ is the combined autoregressive operator.

The autoregressive method based on (Fattah et. al ,2018), every observation consists of a random element (random shock, ϵ) and a linear combination of the previous observations. α_1 during this equation is that the self-regression constant. The integrated method. The behavior of the timeseries is also suffering from the additive result of some processes. As an example, stock standing is continually changed by consumption and provide, however the common level of stocks is basically dependent on the additive result of the instant changes over the period between inventories. Though short-term stock values might fluctuate with giant contingencies around this average value, the extent of the series over the future can stay unchanged. A timeseries determined by the cumulative impact of an activity belongs to the category of integrated processes. Though the behavior of a series is erratic, the variations from one observation to successive may be comparatively low or perhaps oscillate around a relentless value for a method determined at very different time intervals. This stationarity of the series of variations for an integrated method may be a crucial characteristic viewed from the statistical analysis aspect of the timeseries. Integrated processes are the example of nonstationary series. The moving average method. The current value of a moving averaging method may be a linear combination of the present disturbance with one or many previous perturbations. The moving average order indicates the amount of previous periods embedded within the current value to develop a sensible approach in order to perform ARIMA models. The principal rule to spot the model is that if a timeseries is obtained from an ARIMA method, it should have some theoretical autocorrelation properties. The statistical characteristics of a stationary timeseries such as the mean and also the autocorrelation structure are constant over time. We sometimes have to be compelled to apply differencing and power transformation to the data to get rid of the trend and stabilize the variance before an ARIMA model is fitted. After that, it becomes easy to calculate the model parameters and so specify the model. These parameters are

calculable in order that the general error is reduced. Finally, we have a tendency to move to the diagnostic checking of model adequacy. During this last step, we confirm that the hypothesis we created concerning the errors are contended. The diagnostic statistics and plots of residuals can be used to assess the adequacy of future values to our data. If the model isn't adequate, we've to form alternative estimation of parameters followed by the model validation. Diagnostic data will facilitate us return up with new models. Researchers approve that the estimation of parameters needs an oversized variety of observations. Consequently, there are some limits for mistreatment ARIMA model. However, once we have a tendency to apply ARIMA model, we have a tendency to reach a top quality within the opposite of the time series models.

Chaudhuri et.al (2020) conduct a research and found that other authors used ARIMA to forecast electricity prices and modeling in a time series data for forecasting purpose. They also found works regarding the instability in area and production of wheat and maize in many markets.

8.1.1 Pros and Cons of ARIMA Forecasting

- a) ARIMA needs at least 50 and preferably, 100 observations of data to be available in order to make a correct model used most often for hourly or daily data, however with application to several high frequency cases and a few helpful quarterly application. It would like an extended series of data while not structural modification notably major problem if handling seasonal information in 3 years of monthly data, we've got thirty six observations, but, solely 3 observations on every of the monthly seasons.
- b) ARIMA has not automatic updates unlike simple naive or smooth models there is no automatic update feature as new data is available throughout the modeling process should be repeated especially the diagnostic testing phase as the model is likely to reduce the fact that these types are often the instability cause many economists to look suspicious.
- c) ARIMA has high implementation cost because of the large data needs, the lack of convenient updating procedures, and the fact that they must be estimated using nonlinear estimation procedures. The amount of subjective input at the identification stage additionally makes them somewhat a lot of AN art than a science.
- d) ARIMA model tend to be unstable, same with relevancy changes in observations and changes in model specification. Several specifications can yield no results, and like most nonlinear estimation techniques, the results might not be distinctive.
- e) ARIMA works for short run forecasts with high frequency data results can be difficult to beat. We will show in the last section of the term that these types are equal to others under special circumstances (i.e., structured economics, or clear slides) and have a greater chance of being more resistant to the basic assumptions of data fluctuations than many other systems and real data nearby.

8.1.2 Four key steps in forecasting when using the ARIMA model

- a) Looking at ACF and PACF is a process involved in identification to determine which form is most appropriate for data. If the ACF does not move quickly, the chain may become stagnant and some changes will be mandatory. Take repeated differences, if problems are a habit try log modification or otherwise, the problem is related to variability. Frequently the estimated functions are not clear cut and several models may seem appropriate.
- b) Estimation the purposeful style of the ARIMA model is extremely nonlinear thanks to the presence of the moving average terms. like all nonlinear estimation routine, beginning values should lean. Estimation issues like a failure to induce convergence oft results is that the knowledge series isn't stationary.
- c) Diagnostic checking models should be check for validity before the forecasts area unit generated if the model passes then we tend to proceed, if not should come back to the identification stage and decide ensuing most probable model. it's needed the requirement for checking coefficients for significance the coefficients of order p and Q ought to be important (interior needn't be) residuals ought to (must) be white noise. A take a look at of whether or not the residuals kind a white method is given by a changed version of the Box-Pierce Q data point mentioned higher than within the form: wherever letter hat is that the autocorrelations of the residuals, d is that the order of differencing to get a stationary series, T is that the length of the series, and K is that the variety of autocorrelations being checked. Here if Q is larger than the essential price for the chi square distribution with K-p-q degrees of freedom, the model ought to contemplate inadequate.
- d) Predicting this program is straightforward; a forecast is an expectation, that is assessed somewhere in time. Confidence moments can even be simply found in common residual errors.

We may consider timeseries as a raw material, which processed by statistical or econometric methods can highlight recurring issues, analogies, conditionings or benchmarks. More specifically, timeseries passed through the filter of specific analysis/forecast methods can provide information on: the existence of a dominant evolutionary direction which applies particularly to conditions of normality of the process, appearance of systematic periodic oscillations with chances of repeating as effect and scale in the future, evolution's inertial character of some processes or relatively predictable appearance of the evolution of some processes as a response to some deviations from the past. (PETRICĂ et al. ,2016) (Meyler et.al, 1998)

8.2 SARIMAX forecasting model

According to Adhikari and Agrawal (2013), in order to can understand the structure of the SARIMAX model, we have to introduce first the Seasonal ARIMA (SARIMA). ARIMA model is widely known due to its flexibility to be possible used in different types of timeseries with simplicity as well as the associated Box-Jenkins method for optimal model building process. However, the serious limitation of these models is the pre-assumed linear form of the regarding timeseries which becomes inadequate in many practical circumstances. To overcome this limit, various non-linear stochastic models have been proposed in literature. However, from

implementation point of view these are not so simple as the ARIMA is. Box and Jenkins have generalized this model in order to can work with seasonality. Their proposed model is defined as the Seasonal ARIMA (SARIMA) model. In this model, seasonal differencing of the correct order is used to remove non-stationarity from the timeseries. A first order seasonal difference is the difference between an observation and the corresponding observation from the previous year. For monthly time series $s = 12$ and for quarterly time series $s = 4$. This model is generally termed as the s SARIMA(p, d, q) \times (P, D, Q) model.

Based on (Vagropoulos et.al ,2016) research, SARIMAX model is an extension of the SARIMA model, improved by having the ability to embed exogenous variables, in order to increase its forecasting performance. This multivariate version of SARIMA model, called Seasonal ARIMA with exogenous factor (i.e. SARIMAX). The stationarity and inevitability conditions are equal to those of ARMA models. In the examined application, the response variable of the SARIMAX model is the time (months). As the data is nonstationary, the key point for the successful implementation of a SARIMAX model, is the differentiation of both response and exogenous time series before the model's estimation, otherwise there is the hazard of "spurious regression". The selection process of the seasonal (P, D, Q) and non-seasonal (p, d, q) terms was based mainly in certain information criteria (AIC and FPE), but the ACF and PACF plots were also observed and, therefore, the most suitable candidate models were selected.

In addition, Shadkam (2020) points that, SARIMAX methodology meliorates the traditional time-series methods by direct dealing with seasonality in the dataset and has the ability to take in mind external variables. These specifications of SARIMAX make it the most suitable model for seasonal demand forecasting. Author implements the SARIMAX methodology to predict the electricity demand by using the day of the week, holidays, and temperature given as external variables. The SARIMAX method was clearly good in forecasting sudden increases in demand. This kind of ARIMA model is using in data with seasonality by using seasonal AR, MA, and differencing the terms in the model. We can also add external variables to the model by using an exogenous regressor term. Seasonal ARIMA with exogenous regressors (SARIMAX) allows user to add the results of external variables to the model. Exogenous variables are named as variables that affects a model but the model does not affect them. The time is assumed as an exogenous variable in a demand forecast model. A SARIMAX model is defined as SARIMAX (p, d, q) (P, D, Q) s

Where:

- p is the order of the AR term.
- d is the order of differencing needed to make the data stationary.
- q is the order of the MA term.
- P is the order of the seasonal AR term.
- D is the order of the seasonal differencing needed to make data stationary.
- Q is the order of the seasonal MA term.
- S is the number of periods in one season.

We will see below SARIMAX (p, d, q) (P, D, Q)s mathematical expression:

$$y_t = \beta_0 + \beta_1 X_{1,t} + \beta_2 X_{2,t} + \dots + \beta_k X_{k,t} + (1 - \theta_1 B - \theta_2 B^2 - \dots - \theta_Q B^Q)(1 - \Phi_1 B^s - \Phi_2 B^{2s} - \dots - \Phi_P B^{Ps})(1 - \phi_1 B - \phi_2 B^2 - \dots - \phi_p B^p)(1 - \phi_1 B^s - \phi_2 B^{2s} - \dots - \phi_P B^{Ps}) z_t$$

Where:

- y_t defines the value of the series at the time t .
- $X_{1,t}, X_{2,t}, \dots, X_{k,t}$ signify observations of the exogenous variables.
- $\beta_0, \beta_1, \dots, \beta_k$ signify the parameters of the regression part.
- $\phi_1, \phi_2, \dots, \phi_p$ signify the weight of the non-seasonal autoregressive terms.
- $\Phi_1, \Phi_2, \dots, \Phi_P$ signify the weight of the seasonal autoregressive terms.
- $\theta_1, \theta_2, \dots, \theta_Q$ signify the weight of the non-seasonal moving average terms.
- $\Theta_1, \Theta_2, \dots, \Theta_Q$ signify the weight of the seasonal moving average terms.
- B^s signify the backshift operator such that $B^s y_t = y_{t-s}$
- z_t signify the white noise terms.

8.3 Random forest regressor

(Ceh et.al, 2018) in their research define that the Random forest algorithm in general is a classification and regression algorithm which is based on the Bootstrap aggregating and random subspace methods. The idea of Bootstrap aggregating is to create an ensemble of learners, each trained on a bootstrap sample (D_b) acquired from the original dataset (D) with the usage of the following sampling procedure: provided a D with N examples, it makes a D_b by randomly choosing k examples from D with replacement. After removing double values, if N is big and $k = N$, it is await that D_b has two-thirds of the examples from D . The forecast of the ensemble is made from the separate decisions by classification or the regression. It has been shown that bagging can low down the variance in the last model when compared to the basic models and can avoid overfitting. Regression trees are used as base learners in the Random Forest Regression algorithm. After choosing the number of the trees in the forest, each regression tree is grown on a different sample of bootstrap found in the initial training details. Each node in the tree represents a binary test compared to the selected predictive value selected to reduce the remaining total square squares that flow through both branches (left and right). Terminal nodes include more than the maximum set number of examples where the target value is obtained by estimation. To avoid the overlap between the trees in the forest, the process of selecting the best-cracked forecaster on the tree is adjusted to choose between randomly selected predictions only - selecting the default sub-location for the first n -dimensional problem. For a fixed parameter, m ($m \ll n$).

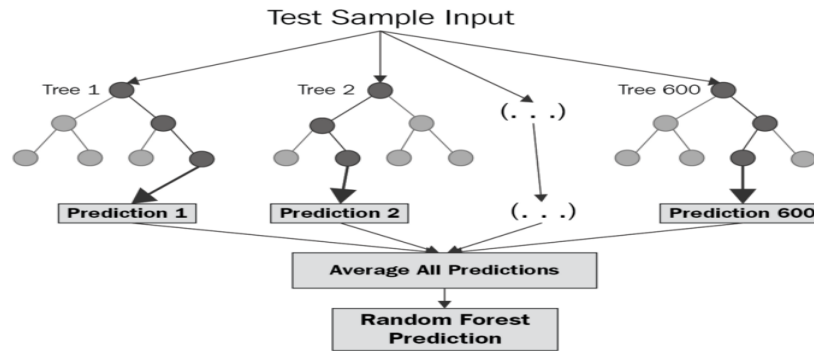


Figure: 9 Random Forest

(corporatefinanceinstitute.com)

Based on (Ouedraogo et.al, 2019) the use of recent data mining or ML models avoids many of those limitations. An emerging method that utilizes ensembles of regressions is receiving specific interest in alternative fields of information. Ensemble learning algorithms use a similar base algorithm to provide continual multiple predictions, that are averaged in order to provide a singular model. They'll reduce bias and improve prediction potency. RFR is associate example of such associate ensemble regression technique. RFR is nonparametric, and so data big ought to come from a selected distribution (e.g. Gaussian) and might contain very variables. Moreover, RFR works well with very giant numbers of predictors. These strategies will cope with model choice uncertainty, as predictions are primarily based upon an accord of the many models and not merely one model chosen with some measure of goodness of work. RFR is suitable for illustrating the nonlinear impact of variables; it will handle complicated interactions among variables and isn't stricken by have been. Major applications of RFR are found in environmental and ecological modeling, Eco hydrological distribution modeling, landslide condition mapping, and remote sensing. In groundwater analysis, RF ways are applied to model nitrate and arsenic in aquifers of the southwestern United States, nitrate in associate loose formation in southern Spain and nitrate in shallow and deep wells of the United States of America Central valley. A perceived disadvantage of traditional ML strategies like ANN is their black-box nature: while not calculable coefficients, it's tough to point out vital relations between the response and predictor variables. RFR relies on bootstrap aggregation of regression trees, and usually outperforms traditional models like logistic regression. Moreover, the parameterization of RFR is extremely straightforward and it's computationally less stern. RFR provides superb results compared to different machine learning techniques like SVM or ANN, or to different call tree algorithms. This study uses RFR to elucidate and predict groundwater nitrate contamination at the continental scale and compares the performance of RFR with ML techniques. The study uses a nitrate contamination information set compiled through a meta-analysis. The modeling of nitrate contamination at this scale will offer steerage for the look and implementation of groundwater watching programs, especially the look of transboundary groundwater management ways adjusted to the conditions in several regions of Africa. RFR associate RFR model was known on a similar information set. RFR modeling is associate ensemble ML technique for classification and regression that operates by constructing a large number of call trees. The philosophy behind ensemble learning techniques relies on the premise that its accuracy is above different ML algorithms as a result of the mixture

of predictions performs a lot of accurately than any single constituent model. The individual call trees in RFR tend to be told extremely irregular patterns, i.e. they over fit their coaching information sets. RFR may be a means that of averaging multiple call trees, trained on totally different components of a similar coaching information set, with the goal of reducing the prediction variance. RFR modeling is suitable for modeling the nonlinear impact of variables. It will handle complicated interactions among variables, and isn't stricken by multiple correlation. RFR will assess the consequences of all instructive variables at the same time, and mechanically ranks the importance of those variables in raining order.

RF is a bagging method and not a boosting one. The trees in random forests are run in parallel. There is no interaction between these trees while building them. It works by building a pile of decision trees during training and issuing a class which is a way of class (division) or predicting the meaning (return) of each tree. A random forest is a meta-estimator, which accumulates many decision trees, with some helpful modifications: The features that can be split on at each node is limited to a percentage of the total (which is defined as the hyper parameter). This ensures that the integration model is not too dependent on any independent factor, and makes good use of all the predictive features. Each tree draws a random sample from the actual data set when it produces its own variance, inserting another random object that prevents overheating. The above changes help prevent trees from becoming too closely related.

8.3.1 Advantages of Random Forest:

- It is an accurate machine learning algorithms available. For many data sets, it produces a highly accurate classifier.
- It runs with high efficiency on large databases.
- It can handle thousands of input variables without delete any of them.
- It gives estimates of what variables that are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method to can estimate missing data and keeps the accuracy when a large proportion of the data are missing.

8.3.2 Disadvantages of Random Forest:

- Random forests have been implement to can fit in some datasets with noisy classification and regression tasks.

- For data that have categorical variables with different number of levels, random forests are prejudiced in side of those attributes with more levels. Therefore, the variable importance scores from random forest are not reliable for this type of data.

(Afroz Chakure ,2019)

A supervised learning algorithm that uses ensemble learning method for regression is the Random Forest Regression. Ensemble learning method combines forecasts from multiple ML algorithms to make a more accurate forecast than just a model. In order to understand better the Random Forest algorithm, we will analyze its steps:

1. Take a random k data points from our training set.
2. Build a decision tree based on these k data points.
3. Choose the number N of the trees we want to build and repeat steps 1 and 2.
4. For a new data point, make each one of the N -tree trees predict the value of y for the data point in question and give the new data point to the average across all of the predicted y values.

Simply said: Random forest forms many decision-making trees and binds them together to obtain accurate and stable predictions. As the trees grow the Random Forest adds more random to the model. The Random Forest has the same parameters as the decision tree or bagging classifier. The Random Forest adds more randomness to the mode while the trees are growing. Instead of looking for the most important feature when taking apart a node, it search for the best specification in a random subset of features. This results in variations that often lead to better modeling. Random Forest is popular for its high speed of prediction and low memory usage because it selects only a set of features from all the partitions.

8.4 Holt Winters Algorithm

Holt (1957) and Winters (1960) update Holt's model in order to be able to handle the seasonality. The Holt-Winters seasonal contains the forecast equation and three smoothing ones — one for the level ℓ_t , one for the trend b_t , and one for the seasonal factor s_t , with corresponding smoothing parameters α , β and γ . (Hyndman and Athanasopoulos, 2018) in their paper use m to define the frequency of the seasonality, i.e., the number of seasons exists in a year. They use, for quarterly data $m=4$, and for monthly data $m=12$. In our case bellow we will also use $m=12$ as we will face monthly seasonality.

In literature there are 2 kinds of this methodology that distinction is within the nature of the seasonal element. The additive methodology is most well-liked once the differences due to the season are well constant through the series and therefore the increasing methodology, that is most well-liked once the differences due to the season area unit changing comparable to the level of the

series. With the additive methodology, the seasonal issue is expressed in absolute terms within the scale of the determined series, and within the level equation the series is seasonally adjusted by subtracting the seasonal element. among annually, the seasonal element can add up to almost zero. With the increasing methodology, the seasonal element is incontestable in relative terms (percentage), and therefore the series is seasonally adjusted by dividing through by the seasonal issue. among annually, the seasonal issue can total up to almost m.

8.4.1 Holt-Winters' additive method

The HW additive method is a type of the HW model, wherein the seasonal part is expressed in fixed terms in the scale of the observed series; while at the equation level (overall smoothing), and the series is seasonally adjusted by deduct the seasonal part. The component form for the additive method we will see it described by Hyndman and Athanasopoulos (2018) and (Hansun et.al, 2019) as follows:

$$S_t = \alpha y_t - I_{t-L} + 1 - \alpha S_{t-L} + b_{t-L}$$

$$b_t = \gamma S_t - S_{t-L} + 1 - \gamma b_{t-L}$$

$$I_t = \beta y_t - S_{t-L} - b_{t-L} + 1 - \beta I_{t-L}$$

$$F_{t+m} = S_t + mb_t + I_{t-L}$$

where

- S_t is the whole (level) smoothing
- b_t is the trend smoothing
- I_t is the seasonal smoothing
- F_{t+m} is the forecast at the m periods ahead
- y_t is the observation (real data)
- t is an index which defines the period of time
- L refers to the number of data points, which defines the new season's begin.
- α, β , and γ are constants that take a number between 0 and 1, which must be estimated in such a way that the other error measurements have the lower possible value.

The component form for the additive method is:

$$\begin{aligned}
\hat{y}_t + h|t &= \ell_t + hbt + st + h - m(k+1)\ell_t = \alpha(yt - st - m) + (1 - \alpha)(\ell_t - 1 + bt - 1)bt \\
&= \beta * (\ell_t - \ell_t - 1) + (1 - \beta *)bt - 1st \\
&= \gamma(yt - \ell_t - 1 - bt - 1) + (1 - \gamma)st - m, \hat{y}_t + h|t = \ell_t + hbt + st + h - m(k+1)\ell_t \\
&= \alpha(yt - st - m) + (1 - \alpha) * (\ell_t - 1 + bt - 1)bt = \beta * (\ell_t - \ell_t - 1) + (1 - \beta *)bt - 1st \\
&= \gamma * (yt - \ell_t - 1 - bt - 1) + (1 - \gamma)st - m,
\end{aligned}$$

Where:

k is the integer part of $(h - 1)/m$, which ensures that the estimates of the seasonal indices used for forecasting come from the final year of the sample.

The level equation shows a weighted average between the seasonally adjusted observation $(yt - st - m)$ and the non-seasonal forecast $(\ell_t - 1 + bt - 1)$ for time t.

The trend equation is identical to Holt's linear method. The seasonal equation shows a weighted average between the current seasonal index, $(yt - \ell_t - 1 - bt - 1)$, and the seasonal index of the same season last year.

The equation for the seasonal component is often expressed as $st = \gamma * (yt - \ell_t) + (1 - \gamma)st - m$. If we substitute ℓ_t from the smoothing equation for the level of the component form above, we get $st = \gamma * (1 - \alpha)(yt - \ell_t - 1 - bt - 1) + [1 - \gamma * (1 - \alpha)]st - m$, which is identical to the smoothing equation for the seasonal component we specify here, with $\gamma = \gamma * (1 - \alpha)$.

The usual parameter restriction is $0 \leq \gamma * \leq 1$, which translates to $0 \leq \gamma \leq 1 - \alpha$.

8.4.2 Holt-Winters' multiplicative method

The component form for the multiplicative method is:

$$\begin{aligned}
\hat{y}_t + h|t &= (\ell_t + hbt)st + h - m(k+1)\ell_t = \alpha y_t st - m + (1 - \alpha)(\ell_t - 1 + bt - 1)bt \\
&= \beta * (\ell_t - \ell_t - 1) + (1 - \beta *)bt - 1st \\
&= \gamma y_t (\ell_t - 1 + bt - 1) + (1 - \gamma)st - m
\end{aligned}$$

The HW model is one of the well-known methods from the Exponential Smoothing models in forecasting. HW is famous because it is simple, has low data storage requirements, and can be implemented automatically. It also has the ability to adapt in changes in trends and seasonal patterns in time series data whenever needs. There are two types of the HW method with them difference in the nature of the seasonal component. These are the additive method, in which the seasonal variations are roughly constant throughout the series, and the multiplicative method, where the seasonal variations are changing according to the level of the series (Hyndman, R.J., & Athanasopoulos, G. , 2018)

They also developed a special HW model in order to predict the abnormality detection in network traffic. In the medical sector also, the HW method has been used in order to can predict assisted childbirths. Moreover, HW hybrid models have also been developed and implemented, used the damped-trend HW and Feedforward Multilayer Neural Network to sales forecasting.

8.5 Simulations based on the space state model

(Aamir and Shabri, 2016) defines the simulation smoother in state space time series analysis as a procedure for drawing samples from the conditional distribution of state or disturbance vectors given the observations. They present a new technique for this, which is both simple and computationally efficient. The treatment includes models with diffuse initial conditions and regression effects. Computational comparisons are made with the previous standard method. The process of state space model has two steps. First, a state vector it is designed for the essentials accumulated to the end. The vector that is called the state vector is the smallest and it concludes the past behavior of the full system. It has the key role of state space modelling which defines the state vector. The smoothing in the form of two linear equations is explained by the state space modelling. The first equation provides the relationships between the current observation and unobserved states which is also called the observation equation. The second equation provides the progress in the states during the time called state equation. The state vector is updated live through the state equation. The state equation is a vector comprising of undetected components such as trend, seasonality and level or Autoregressive and Moving Averages factors of a timeseries. The simulation in our model is based on the Holt-Winter's state space model methods. The state space model define that the true value at time t is randomly distributed around the forecasted value. If using the additive error model, this means:

$$y_t = \hat{y}_t + e_t \sim N(0, \sigma^2) \quad y_t = \hat{y}_t + e_t \sim N(0, \sigma^2)$$

Using the multiplicative error model:

$$y_t = \hat{y}_t \cdot (1 + e_t) \sim N(0, \sigma^2) \quad y_t = \hat{y}_t \cdot (1 + e_t) \sim N(0, \sigma^2)$$

Adding these equations into the smoothing equation formulation leads to the state space equations. The notation used here follows.

Additionally,

$$B_t L_t S_t Y_t = b_t - 1 \circ d \phi = l_t - 1 \circ b B_t = s_t - m = L_t \circ s S_t, B_t = b_t - 1 \circ d \phi L_t = l_t - 1 \circ b B_t S_t = s_t - m Y_t = L_t \circ s S_t,$$

Where:

$\circ d$ is the operation connecting trend and damping term (multiplication if the trend is additive, power if the trend is multiplicative)

$\circ b$ is the operation connecting level and trend (addition if the trend is additive, multiplication if the trend is multiplicative)

$\circ s$ is the operation connecting seasonality to the rest.

The state space equations can then be formulated as

$$\begin{aligned} y_t l_t b_t s_t &= Y_t + \eta \cdot e_t = L_t + \alpha \cdot (M_e \cdot L_t + \kappa l) \cdot e_t = B_t + \beta \cdot (M_e \cdot B_t + \kappa b) \cdot e_t \\ &= S_t + \gamma \cdot (M_e \cdot S_t + \kappa s) \cdot e_t y_t = Y_t + \eta \cdot e_t l_t = L_t + \alpha \cdot (M_e \cdot L_t + \kappa l) \cdot e_t b_t \\ &= B_t + \beta \cdot (M_e \cdot B_t + \kappa b) \cdot e_t s_t = S_t + \gamma \cdot (M_e \cdot S_t + \kappa s) \cdot e_t \end{aligned}$$

With:

$\eta_{Me} = \begin{cases} Y_{t+1} & \text{if error is multiplicative} \\ 1 & \text{else} \end{cases}$ $\eta = \begin{cases} Y_{t+1} & \text{if error is multiplicative} \\ 1 & \text{else} \end{cases}$

When using the additive error model:

$\kappa_{kbs} = \begin{cases} 1 & \text{if seasonality is multiplicative} \\ k & \text{else} \end{cases}$ $\kappa_{lt} = \begin{cases} 1 & \text{if trend is multiplicative} \\ l & \text{else} \end{cases}$ $\kappa_{lts} = \begin{cases} 1 & \text{if seasonality is multiplicative} \\ l & \text{else} \end{cases}$ $\kappa_{lts} = \begin{cases} 1 & \text{if seasonality is multiplicative} \\ l & \text{else} \end{cases}$

When using the multiplicative error model:

$\kappa_{kbs} = \begin{cases} 0 & \text{if seasonality is multiplicative} \\ S_t & \text{else} \end{cases}$ $\kappa_{lt} = \begin{cases} 1 & \text{if trend is multiplicative} \\ l & \text{else} \end{cases}$ $\kappa_{lts} = \begin{cases} 0 & \text{if seasonality is multiplicative} \\ L_t & \text{else} \end{cases}$

(Hyndman & Athanasopoulos, 2018)

8.6 Performance measure errors

In order to get the error value, we deduct the predictive value from the real value of the data at the period. To assess the value of this prediction error, we may use the mean error or the mean absolute error. Therefore, the ME is just the average error. The MAE is computed by considering the absolute value of the difference of the forecasted value and the real value at the same time so that negative prices do not cancel the positive prices. A rating of these absolute values is considered to determine the mean absolute error. A MSE measures the variability of the prediction errors. Seemingly, the variability in the performance measure errors should be low. ME, MAE and MSE are all measures based on the measurement of the forecast accuracy, i.e., their values are expressed in terms of initial measurement units. RMSE estimates the average dimension of the error. In RMSE the difference of the predicted and the real values are each squared and then averaged over the sample. Since the errors are squared before they are averaged, RMSE provides the highest weight for large errors. That means that RMSE is very useful when large errors are notably undesirable. MPE is the number of error at a specific time point in a series. The average percentage error measurement throughout the series is a general measure of useful equity by comparing the equations of different models. This measure adds all the percentage errors each time and divides them by the number of time points. Since the positive and negative errors may be self-reported, MPE statistics are often reversed with the MAPE. This error is a more accurate statistical index because the rate is a certain percentage and the unit of prediction series will not affect its validity. The closer the MAPE is to zero, the better the forecasting results. We calculate predictability errors by using all the performance measures namely MAE, MSE, RMSE and MAPE in all four algorithms we choose to compare in this thesis.

According to Nicolas Vandepuut (2019) the performance measures equations which will use in calculations are:

- First of all the error is defined as $et = ft - dt$ where ft is the forecast and dt is the real demand.
- $MAE = \frac{1}{n} \sum_{i=1}^n |et|$ where n = the number of errors, $|et|$ = the absolute errors. Is very good to measure forecast accuracy. As its name defines, it is the mean of the absolute errors.
- $MAPE = \frac{1}{n} \sum_{t=1}^n \left| \frac{et}{dt} \right|$ MAPE is the sum of the individual absolute errors divided by the demand (each period separately). It is the average of the percentage errors.

- $MSE = \frac{1}{n} \sum_{t=1}^n (et)^2$ Many algorithms use MSE as it is faster to compute and easier to manipulate than RMSE. However, it is not scaled to the original error (because the error is squared), for this reason, we cannot relate to the original demand scale.
- $RMSE = \sqrt{MSE}$ It is defined as the square root of the mean squared error.

Additionally, (Saigal and Mehrotra, 2012) mention in their research the following observations in order to can decide and evaluate the best performance metric:

- RMSE is one of the best report the rather than MSE, because the RMSE is computed by using the same units as the dataset has, rather than in squared units, and it is a representation of the size of a "typical" error.
- MAE is also estimated in the same component as the original data, and is almost similar in consequence to, but little smaller than, the root mean squared error. Those who are challenged by mathematics often find this to be a simpler figure than RMSE.
- MAPE is usually useful for reporting, because it is provided in generic percentage, which would make some kind of idea even for someone who has absolutely no idea what a "big" error based on the products sold and the money spent. The MAPE can only be given by taking in mind the data that are mandatory to be strictly positive.
- The RMSE can face the large error case with better precision than the rest: the squaring process gives out of proportion weight to huge errors. If one of these is not an error in decision-making, then the MAE or MAPE should be a more suitable evaluating metric. In many cases, these statistics will vary without been the case when the distribution of error is present as an "outlier."
- The confidence intervals in the forecasts that are one-step ahead are mostly based on the RMSE, the confidence intervals for the extended forecasts can be computed by time series models highly depended on the underlying modeling estimations, particularly cases about the alterability of the trend.

(Saigal and Mehrotra, 2012)

9. Data set analysis

Our purpose in this thesis is to make forecasts and compare based on the errors the above algorithms. In order to do this, we have to choose a suitable data set, which includes product demand data.

After do some research in kaggle, google data set search, data.gov and other site we decide to use the dataset from Felix Zhao published in Kaggle

(<https://www.kaggle.com/felixzhao/productdemandforecasting>) which includes the historical daily demand of 2172 products from a manufacturing company with global presentation. The company provides these products within dozens of product categories. There are four central warehouses to ship products. Since the production takes place in different locations all over the world, it normally takes more than one month to ship products via sea to different central warehouses. If forecasts for each product in different central with reasonable accuracy for the monthly demand after next can be achieved, it would be beneficial to the company in multiple ways.

In our case, we do not take in mind the warehouse and the category. We use just the product code, date and order demand in order to forecast the monthly demand of each product with 4 different algorithms and compare the actual values with the predicted ones, then we calculate the errors Mean Absolute Error, Mean Squared Error, Mean Absolute Percentage Error, Root Mean Squared Error and evaluate the best algorithm. In order to do this preprocess in our dataset, we use Microsoft's Excel software and Visual Basic on it in order to can modify the dataset according to our needs before use it in our Python Scripts to make the forecasts.

First, we erase the column Warehouse and Product_Category and then we put the data in order from the oldest to the newest according to the date column by using the order tool in Excel. Next, we erase all null dates and the dates of years 2011 and 2017 because they have not complete data from all months January to December. After that we fix the demand values which are not just a number but also have some symbols like () or – and we cannot use without remove these. In order to do this, we use the “find and replace” tool in Excel and ask it to find and replace each symbol with a ‘space’.

Next, we put in order the data according to the Product_Code column and we sum the same date demands by using the below functions, this happens because we erase the Warehouse column and now we have many lines with same Product_Code and same date demand. By using the below Excel's functions, first we sum them up in one line. After that, we erase the other lines and then we split the data according to the Product_Code in separate sheets and then merge them in one sheet in order to save it in .csv format and use it as input in our Python scripts to do our forecasts.

Steps

1. We insert the data from the original csv file into an xlsx file in order to modify the data by using the below functions and processes.
2. We create an extra column next to the last used column in our file
3. In second line of the new column we use the function =IF(DATE NOW = DATE BEFORE LINE;SUM OF THE DEMANDS;ONLY NOW DEMAND)
4. In next line of the same column we use the function =IF(DATE NOW=DATE BEFORE;BEFORE LINE DEMAND+NOW DEMAND;NOW DEMAND)
5. We repeat this function of step 3 in all the below lines of the new column.
6. We create one more new column and in its first line if the date is same with the second line we write by hand the number 1 and if not we write FALSE.

7. In the second line of this column we use the function `=IF(NOW DATE=DATE BEFORE;1;FALSE)`
8. Then in all lines with the FALSE value we keep the demand and the line as it is
9. In all the lines with the value 1 on them, we keep only the line and the demand with the biggest demand value which is the sum of all the same demand lines and we erase the rest lines.
10. In order to split the data in sheets based on the Product_Code because we must sum up the month demands and form our timelines; we use the visual basic script from exceltip.com. (Appendix 3)
11. In order now to put all the sheets in ascending order we use the visual basic macros script from the howtogeek.com which are suggested by Lori Kaufman,2016 (Appendix 4)
12. Because we need the demand per month we make an array in column D1 we write month and we use the D2–D6 to put the years 2012-2016 and in the E1-P1 we put the months by using numbers 1-12 (Jan-Dec). In order to fill the array we use the function `=SUMPRODUCT((MONTH(DATE)=E$1)*(YEAR(DATE)=$D2)*(DEMAND))` and by this excel fill the array with the sum of the demand for the specific month. (we do this with all sheets selected in order to be done in all at same time and no need to do by hand in each sheet)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1	Date	Order_Demand	month																					
2	9/2/2012	27		2012	44	72	73	61	80	53	51	71	41	49	69	16								
3	2/11/2012	20		2013	38	56	43	73	64	41	69	23	33	66	37	65								
4	5/2/2013	12		2014	52	29	24	46	25	33	67	49	45	35	24	31								
5	23/4/2014	10		2015	23	33	12	20	44	20	47	28	12	48	31	14								
6	28/10/2016	10		2016	7	31	63	19	55	5	18	36	53	42	26	25								
7	14/4/2016	9																						
8	1/12/2016	9		Jan-12	44	44																		
9	8/3/2016	8		Feb-12	72	72																		
10	30/5/2013	7		Mar-12	73	73																		
11	22/10/2012	6		Apr-12	61	61																		
12	14/8/2014	6		May-12	80	80																		
13	19/8/2015	6		Jun-12	53	53																		
14	11/2/2015	5		Jul-12	51	51																		
15	1/8/2012	4		Aug-12	71	71																		
16	27/2/2012	3		Sep-12	41	41																		
17	9/8/2012	1		Oct-12	49	49																		
18	13/11/2012	1		Nov-12	63	63																		
19	23/6/2015	1		Dec-12	16	16																		
20	18/3/2015	0		Jan-13	38	38																		
21	20/1/2012	1		Feb-13	56	56																		
22	23/1/2012	1		Mar-13	43	43																		
23	30/1/2012	1		Apr-13	73	73																		
24	29/3/2012	1		May-13	64	64																		
25	1/4/2012	1		Jun-13	41	41																		
26	27/4/2012	1		Jul-13	69	69																		
27	13/6/2012	1		Aug-13	23	23																		
28	27/6/2012	1		Sep-13	33	33																		
29	6/7/2012	1		Oct-13	66	66																		
30	8/8/2012	1		Nov-13	37	37																		
31	9/8/2012	1		Dec-13	65	65																		
32	20/9/2012	1		Jan-14	52	52																		
33	3/10/2012	1		Feb-14	29	29																		
34	17/10/2013	1		Mar-14	24	24																		

Figure: 10 Excel file for the monthly demand

13. We put in a column separate the months per year like this 1-1-2012 for Jan of 2012 and 2-1-2012 for the Febr of 2012 etc. And we copy and paste by using transpose each line of year from the array and make it column. But because of this first copy paste is with formulas we copy this transposed column and paste the values. (we do this with all sheets selected in order to be done in all at same time and no need to do by hand in each sheet)

14. After that, we delete the rest columns, keep just the date and demand in order to formulate our timelines. We can do this at once in all sheets. We can do this in just one sheet with all sheets selected in order to do it at same time in all sheets.
15. Finally we copy all demand rows side by side in one new combined sheet using this code script from mrexcel.com suggested by (alansidman ,2013) (Appendix5)
16. We find the sum of zero values in each column of products by using the function =COUNTIF(B2:B61,0) where B2:B61 are the cells of the timeseries when we want count the zero vlues.
17. We erase the timeseries with sum of zeros more than 13 (we choose 13 as limit because of if have sum more than 13 that means we lose 1 year of our history data from timeseries). In addition, we do this because we need normal timeseries and not special timeseries with many zeros on them in order to have a valid comparison between the algorithms. We erase the columns by using this visual basic script (Appendix 6)
18. We use these remain 1380 timeseries on sheet in order to form the timeseries and make forecast, we save it as .csv file to can use it in our python scripts. In order to be suitable the data we format them according our script needs that is the date must be month-day-year (mm-dd-yyyy) format and the rest columns with the timeseries date must be number without decimals.

These are the steps we conduct for the data set preprocess before can use it in our research. Now we present the code and the scripts, which we use in order to do the algorithms evaluation.

10. Implementation of Algorithms in Python

We implement a python script in two versions. We use the first version in case we want to examine the algorithms and those results and error for just one timeseries. The second version is the same script but that it do is to examine the algorithms in multiple timeseries and save the errors in one file in order to compare them and find the most suitable algorithm based on its errors.

In our scripts, we use many specific modules like pandas, matplotlib, statsmodels, sklearn, math, tkinter, numpy, warnings, pylab and xlswriter. In next chapter, we describe the most important of them and which packages we import and use in our code.

10.1 Python Modules

10.1.1 Pandas

According to its developers, **pandas** is a fast, powerful, flexible and easy to use open-source data analysis and manipulation tool, built on top of the Python programming language.

Pandas start its grow in 2008 at AQR Capital Management. By the end of 2009 it had been open sourced, and is supported in our days by a community of like-minded individuals around the world who dedicate their precious time and energy to help make open-source *pandas* possible. Since 2015, *pandas* is a NumFOCUS sponsored project. In addition, this help ensure the success of development of *pandas* as a world-class open-source project.

Timeline

- 2008: Development of *pandas* begin
- 2009: *pandas* become open source
- 2012: First edition of *Python for Data Analysis* is published
- 2015: *pandas* become a NumFOCUS sponsored project
- 2018: First in-person core developer sprint

A fast and efficient DataFrame object for data manipulation with integrated indexing;

Tools for reading and writing data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format;

- Smart data coordination and integrated management of lost data: find automatic label-based alignment in the compilation and easy use of contaminated data in an orderly manner.
- Flexible reconstruction and testing of data sets.
- Smart label-based pruning, advanced indexing, and resetting big data sets.
- Columns can be inserted and removed from data structures by size variation.
- Consolidating or modifying data with a dynamic team with an engine that allows the integration of integration into data sets.
- Integration of high performance and joining data sets;
- Hierarchical axis index provides an accurate way to work with high-volume data in a low-level data structure.
- Timeseries performance: date range production and frequency conversion, moving window statistics, day variables and remainder. Or create a temporary domain payment and join a time series without losing data.
- Highly customizable, with the most sophisticated codecs written in Cython or C.
- Python and pandas are used in a variety of educational and commercial topics, including finance, Neuroscience, Economics, Mathematics, Marketing, Web Analytics, and more.

pandas mission is the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader aim of becoming the most powerful and flexible open-source data analysis / manipulation tool available in any programming language.

(<https://pandas.pydata.org/about/>)

10.1.2 Sklearn

One other library we use in our script is the scikit-learn (sklearn) which is an ML library which assumes a very basic working knowledge of ML practices (model fitting, predicting, cross-validation, etc.). Scikit-learn is an opensource ML library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection and evaluation, and many other utilities. Scikit-learn provides dozens of built-in ML algorithms and models. We use this library for the Random Forest Regressor. A random forest is a Meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-

sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree. The sklearn has also the module **sklearn.metrics** that implements several losses, score, and utility functions to measure regression performance. Some of those are enhanced to handle the multioutput cases: **mean_squared_error**, **mean_absolute_error** etc. We use this module in order to calculate the errors in our script. (Pedregosa *et al.*,2011)

10.1.3 Stat models

Statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration. An extensive list of result statistics is available for each estimator. The results are tested against existing statistical packages to ensure that they are correct. The package is released under the open source Modified BSD (3-clause) license. The online documentation is hosted at statsmodels.org. As background it has the models module of `scipy.stats` which was originally written by Jonathan Taylor. For some time `statmodels` was part of `scipy` but was later pulled out. During the Google Summer of Code 2009, `statsmodels` was fixed, tested, improved and released as a new package. Since then, the `statsmodels` team has continued to add new models, tools, and statistical methods. In our script we use this module in order to implement the ARIMA, SARIMAX and Holt Winters models by using the appropriate commands. (Seabold, et.al 2010)

10.1.4 Tkinter

The `tkinter` package (“Tk interface”) is the standard Python interface to the Tk GUI toolkit. Both Tk and `tkinter` are available on most Unix platforms, as well as on Windows systems. (Tk itself is not part of Python; it is maintained at ActiveState.) The `tkinter.filedialog` module provides classes and factory functions for creating file/directory selection windows, which we use in order to choose our dataset input file for our script.

(<https://docs.python.org/3/library/tkinter.html#module-tkinter>)

10.1.5 Numpy

According to (Harris, et.al 2020) NumPy is an open-source project with the purpose to make possible the numerical computing with Python. It was made in 2005, building on the early work of the Numerical and Numarray libraries. NumPy will always be 100% open-source module. Also is one of the main packages for scientific calculations in Python. It is a Python module that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and a big set of routines in order to do fast calculations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

In the heart of the NumPy module, is the `ndarray` object. This encapsulates n-dimensional arrays of homogeneous data types, with many tasks being performed in compiled code for performance. There are many and serious differences between NumPy arrays and the standard Python commands:

- NumPy arrays have a fixed size when made, unlike Python lists (which can change dynamically). Changing the size of a `ndarray` create a new array and delete the original one.

- The elements in a NumPy array are mandatory to be of the same type, and thus will be the same size in memory. The exclusion is that one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
- NumPy arrays make more advanced mathematical and other types of operations on large numbers of data. Typically, such operations are implemented more efficiently and with less code than is possible using Python's built-in functions.
- A huge amount of scientific and mathematical Python-based packages are using NumPy arrays; though these they mainly support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays.
- The points about sequence size and speed are particularly important in scientific computing.

In our case we mainly use the `asarray` and the `append` command which append values to the end of an array. We use these in our model functions where we have many arrays to fill and edit for our algorithm models.

10.1.6 XlsxWriter

Jmcnamara defines XlsxWriter as a Python module for writing and modifying files in the Excel 2007+ .xlsx file format. It can be used in order to write text, numbers, and formulas in multiple sheets and it can use features like the formatting of columns, images, charts, page setup, automatic filters, conditional formatting and many other functions. XlsxWriter has advantages and disadvantages against the rest alternatives in Python modules for writing Excel files.

Advantages:

- Is capable to use many Excel features than any other of the alternatives.
- It is loyal with files produced by Excel. In most of the cases, the files, which produce, are almost same to files produced by Excel.
- The documentation is huge, there are many example files and tutorials.
- It is speedy and is made to use small amount of memory even for very large output files.

Disadvantages:

- It is not capable to modify or read Excel .xlsx files.

For these reasons, we choose to use this module in order to create the .xlsx file that the script, which we use for n timeseries provide us as output in order to can compare and find the best algorithm based on its errors.

10.2 Script for 1 timeseries description in steps

1. First the script asks us to choose the file with the timeseries data.
2. Then the script plots in a graph the data of timeseries.
3. Then the script plots seasonal decomposition in a figure.
4. After that, we type the number of estimators for the random forest regressor.
5. Prints the expected and predicted values of the random forest regressor side by side.

6. Plots the expected and predicted values in a graph with different color in order to can compare.
7. Next, runs and evaluates many orders in the ARIMA model and checks them based to the RMSE error and chooses the best in order to run the ARIMA model with this.
8. Prints the predicted and estimated values from ARIMA model side by side.
9. Plots the values in a graph with different colors in order to compare them.
10. Prints the ARIMA model summary.
11. Plots the SARIMAX model graph with the forecast and simulations of the algorithm in order to see how the algorithm fits in our data.
12. Plots the SARIMAX model graph with the real, historical and predicted values with different colors to can compare.
13. Prints the model summary of SARIMAX.
14. After that prints the type of holt winters model which use based on our timeseries data (additive or multiplicative).
15. Plots the HOLT WINTERS forecasts with the simulations of the algorithm in order to see how the algorithm fits in our data.
16. Then plots the model graph with real, historical and predicted values of holt winters algorithm with different colors to can compare.
17. Prints the HOLT WINTERS model summary.
18. Prints the ERRORS of all the algorithms.

10.3 Script for n timeseries description in steps

1. Frist the script creates an xlsx file in order to store the algorithm errors.
2. Then the script fills the file with the names for the sheets, 1 sheet per algorithm and the error names in the first columns.
3. Then the script asks to choose our timeseries data file.
4. After that, we type the start and end number for the loop (how many timeseries will try).
5. Next, we type the number of estimators for the random forest regressor.
6. Script runs the algorithms RANDOM FOREST REGRESSOR, ARIMA (first finds the best order for ARIMA), SARIMAX (with same order as ARIMA and with a fixed seasonal order which is (1,2,1,12)), HOLT WINTERS model additive or multiplicative (script tests the data and decides which to implement).
7. In order to see that it works correct it prints three times 'OK' .
8. Next, the script prints all algorithm errors.
9. After that, the script prints the name of the timeseries (header).
10. Fills these data in the specific file cells and proceeds in the next timeseries.
11. When finishes with all timeseries it saves the file and close.

10.4 Main commands which use in python scripts

From pandas we use 'pandas.read_csv' in order to read our timeseries file more specific

```
df = pd.read_csv(myfile, header=0, infer_datetime_format=True, parse_dates=[0],
index_col=[0],delimiter=';', usecols=[0, keep_rows])
```

In this command we define the filename, header line, if we have dates, the index column, the delimiter and the column we use (in the code of the script for one timeseries we do not use the usecols= because we have just one timeseries in the file).

From the pandas, we use the dataframe.iloc that is the purely integer-location based indexing for selection by position. In our case, we use it first to split our data in train and test set, as we can see the train set is the timeseries components-12 and the test set is the last 12 remain timeseries components (the latest 12).

```
df_train = df.iloc[:-12]
```

```
df_test = df.iloc[-12:]
```

We use the df.values which store the values of our data frame in a variable in order to use them in our forecasts.

```
values = df.values
```

Moreover, in each algorithm we use some functions which we call to make the predictions such as the **def walk_forward_validation(data, n_test)** and the

def random_forest_forecast(train, testX) which both are used in the random forest regressor model. The main function for this algorithm is the **RandomForestRegressor(n_estimators=estim, n_jobs=2)** which included in the sklearn library

We also create the function that make evaluations of many ARIMA model orders to choose the most suitable based on the error in order to use this for the model.

```
def evaluate_arima_model(X, arima_order)
```

```
def evaluate_models(dataset, p_values, d_values, q_values)
```

By using the bellow command, we implement our ARIMA model with the most suitable order which we found and optimize this model, too. This command is included in statmodels module.

```
ARIMA(history, order=(ord1)).fit(optimized=True, use_brute=True)
```

For the SARIMAX model, we use the bellow command that also included in statmodels module.

```
model1 = SARIMAX(df_train, order = (ord1), seasonal_order =(2, 1, 1, 12)).fit(optimized=True, use_brute=True)
```

For the Holt Winters model, we count the zeros in our dataset and if there are no zeros on it, we use the multiplicative model by using the below command which is included in statmodels module.

```
model=HWES(df_train,seasonal_periods=12,trend='add',seasonal='mul',initialization_method="estimated").fit(optimized=True,use_brute=True)
```

If our dataset has one or more zeros, we use the additive model as if use the multiplicative we have errors as it is not possible to do divisions with zero. Additive model be implemented by using the below command.

```
model = HWES(df_train, seasonal_periods=12, trend='add', seasonal='add',  
initialization_method="estimated").fit(optimized=True, use_brute=True)
```

We make simulations based on the state space model for 12 steps and 100 repetitions in order to see a general fit of the algorithm in our data. We do the simulations in SARIMAX and Holt Winters algorithms, by using the below commands.

```
simulations1 = model1.simulate(12, repetitions=100, error='mul', anchor='end')  
SARIMAX
```

```
simulations = model.simulate(12, repetitions=100, error='mul') HOLT WINTERS
```

From the module `xlswriter` we use the below commands in order to create our `.xlsx` file with the errors for the comparison.

```
wb = xlswriter.Workbook ('ALGORITHMS COMPARISON____.xlsx') which create  
the file
```

```
sheet1 = wb.add_worksheet('ERRORS HOLT WINTERS') which add the worksheets  
and name them according to each algorithm
```

```
ws = wb.get_worksheet_by_name('ERRORS HOLT WINTERS') in order to access the  
worksheets and fill them
```

```
ws.write(1, 0, 'Mean Absolute Error') in order to write in the cells of the specific  
worksheet
```

10.5 Results of python script for algorithm comparison in 1 timeseries

We run the file as exe or with python shell (for better visual we suggest use python IDE shell) and we take the below results. First, we choose in the open file dialog the csv file with our timeseries data in which we want to test the algorithms.

The script plot the timeseries in a diagram like this

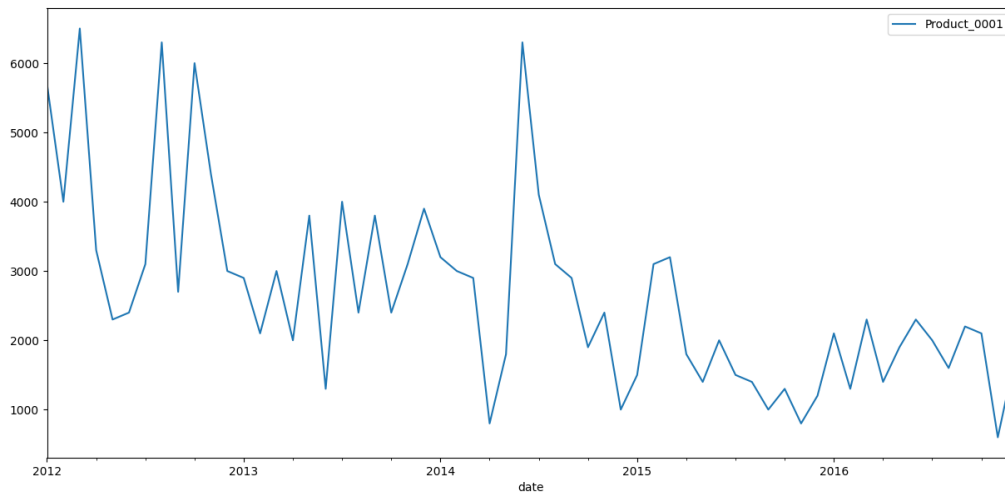


Figure: 11 Timeseries data graph

Next, we close the window of the diagram and the script make ETS decomposition by plot in same figure the timeseries data, the trend, the seasonality and the residuals.

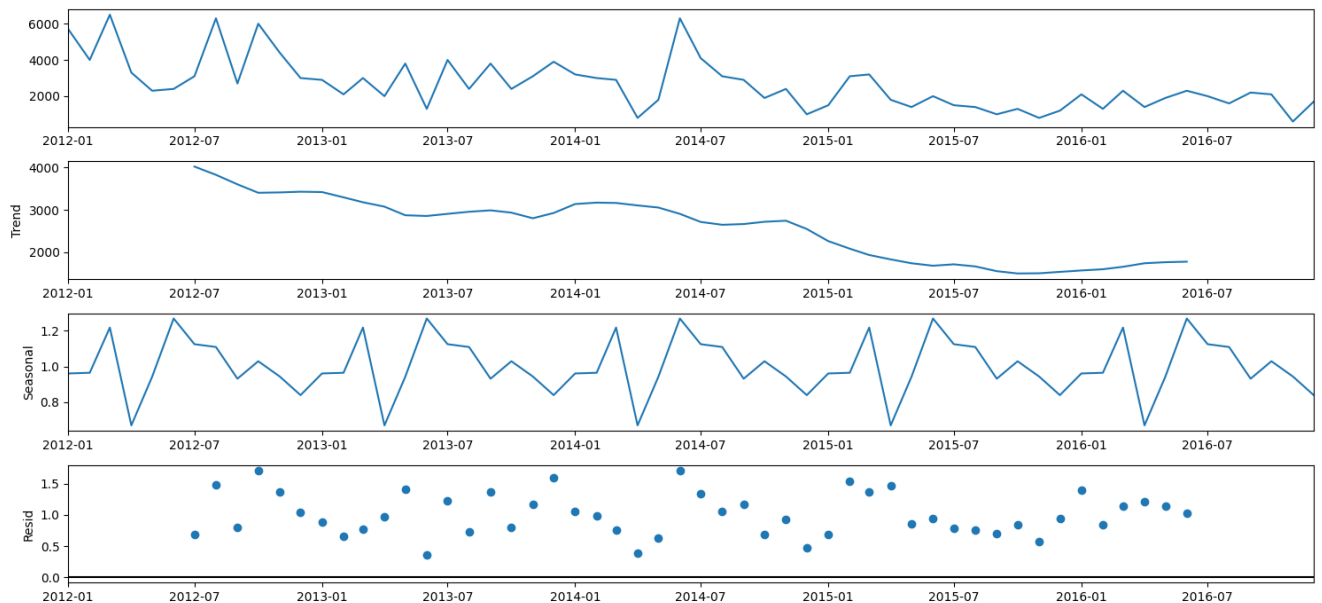
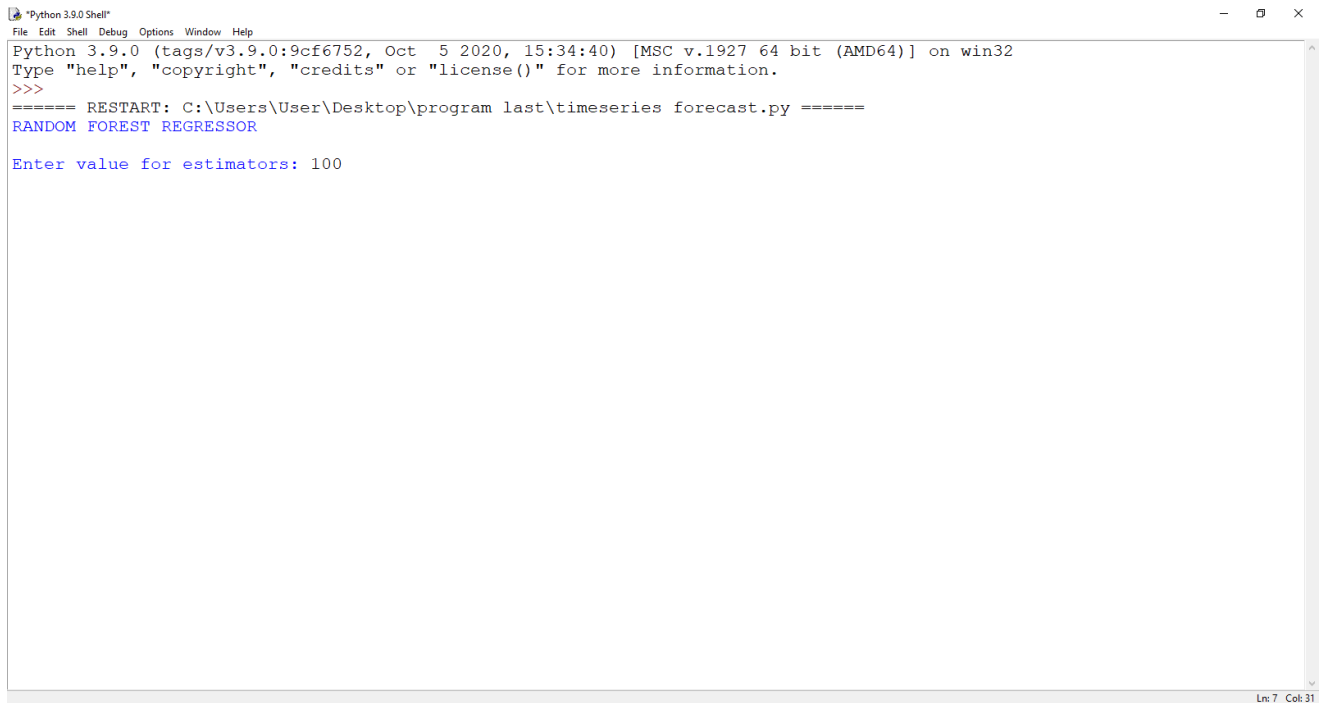


Figure: 12 ETS decomposition

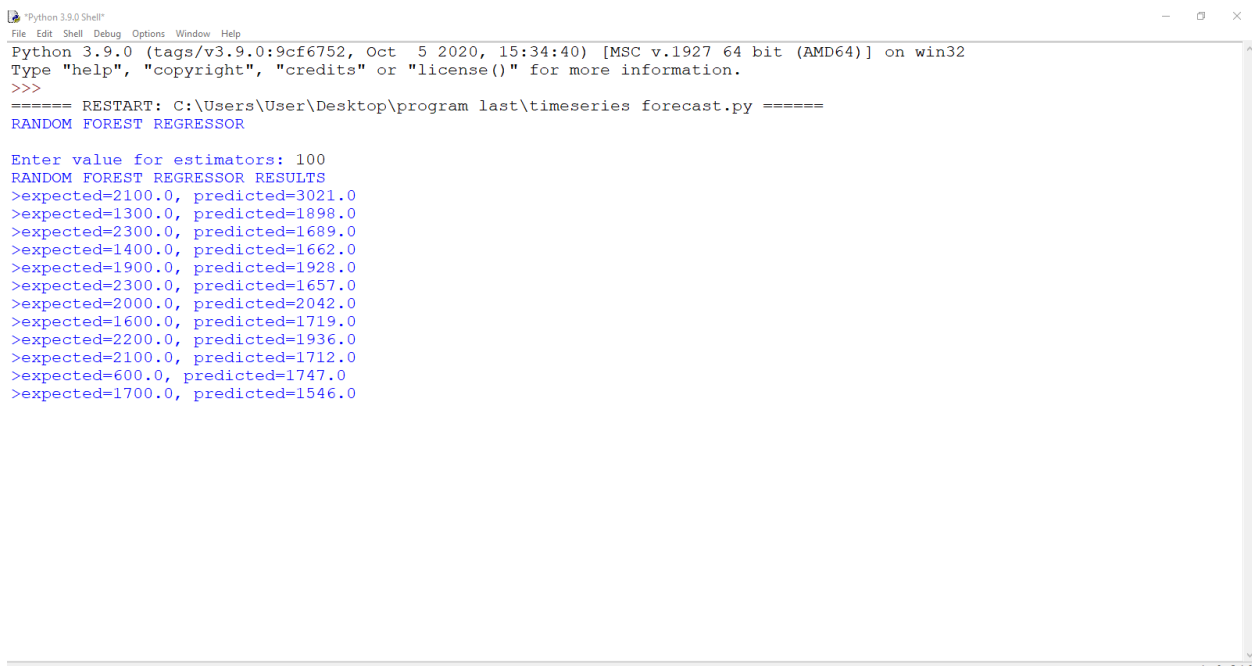
After we close this window and script ask us to insert the number of estimators for the Random forest regressor algorithm. We type the number and then press enter



```
Python 3.9.0 Shell
File Edit Shell Debug Options Window Help
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct 5 2020, 15:34:40) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\User\Desktop\program last\timeseries forecast.py =====
RANDOM FOREST REGRESSOR
Enter value for estimators: 100
```

Figure: 13 estimators for the random forest regressor

Then the script prints the detailed predicted and expected values of random forest regressor for 12 months side by side and when finish it plots a graph with expected and predicted values in order to can see the comparison. We close the graph window and then the script asks us to press ‘enter’ in order to proceed to the next algorithm.



```
Python 3.9.0 Shell
File Edit Shell Debug Options Window Help
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct 5 2020, 15:34:40) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\User\Desktop\program last\timeseries forecast.py =====
RANDOM FOREST REGRESSOR

Enter value for estimators: 100
RANDOM FOREST REGRESSOR RESULTS
>expected=2100.0, predicted=3021.0
>expected=1300.0, predicted=1898.0
>expected=2300.0, predicted=1689.0
>expected=1400.0, predicted=1662.0
>expected=1900.0, predicted=1928.0
>expected=2300.0, predicted=1657.0
>expected=2000.0, predicted=2042.0
>expected=1600.0, predicted=1719.0
>expected=2200.0, predicted=1936.0
>expected=2100.0, predicted=1712.0
>expected=600.0, predicted=1747.0
>expected=1700.0, predicted=1546.0
```

Figure: 14 Random Forest Regressor results

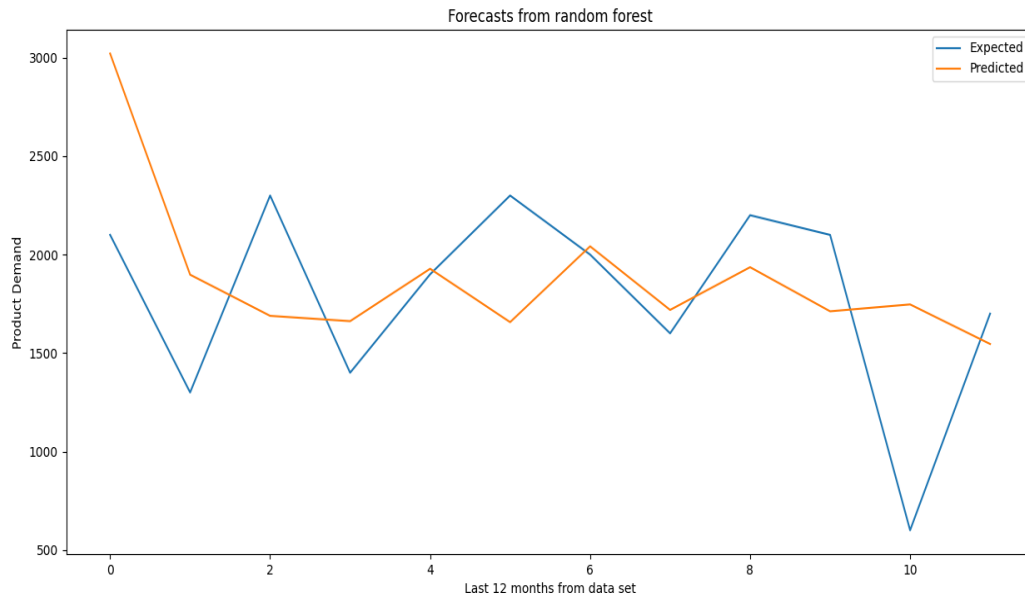


Figure: 15 Random Forest Regressor results graph

Now the script runs the ARIMA algorithm evaluation in a big set of p,q,d parameters and with the comparison of the RMSE error that has in each combination of p,q,d keeps the best order.

```

Python 3.9.0 Shell
File Edit Shell Debug Options Window Help
>expected=2000.0, predicted=2042.0
>expected=1600.0, predicted=1719.0
>expected=2200.0, predicted=1936.0
>expected=2100.0, predicted=1712.0
>expected=600.0, predicted=1747.0
>expected=1700.0, predicted=1546.0

Press enter to go to the next model)
ARIMA MODEL

ARIMA(0, 0, 0) RMSE=1394.578
ARIMA(0, 0, 1) RMSE=1219.930
ARIMA(0, 0, 2) RMSE=1065.176
ARIMA(0, 1, 0) RMSE=732.649
ARIMA(0, 1, 1) RMSE=646.645
ARIMA(0, 1, 2) RMSE=638.128
ARIMA(0, 2, 0) RMSE=1213.082
ARIMA(0, 2, 1) RMSE=760.084
ARIMA(0, 2, 2) RMSE=805.342
ARIMA(1, 0, 0) RMSE=1095.377
ARIMA(1, 0, 2) RMSE=957.999
ARIMA(1, 1, 0) RMSE=651.575
ARIMA(1, 1, 1) RMSE=636.148
ARIMA(1, 2, 0) RMSE=1058.591
ARIMA(2, 0, 0) RMSE=952.168
ARIMA(2, 0, 1) RMSE=812.534
ARIMA(2, 0, 2) RMSE=1036.198
ARIMA(2, 1, 0) RMSE=607.041
ARIMA(2, 1, 1) RMSE=624.154
ARIMA(2, 1, 2) RMSE=682.905
ARIMA(2, 2, 0) RMSE=946.615
ARIMA(2, 2, 1) RMSE=653.151
Best ARIMA(2, 1, 0) RMSE=607.041
Press enter to go to the results of ARIMA with best order)
Ln: 41 Col: 28

```

Figure: 16 ARIMA model evaluator

After that we press 'enter' and script proceeds and gives us the ARIMA best order model detailed results in the same way as in random forest side by side and in a graph in order to see the comparison between them.

```

Python 3.9.0 Shell
File Edit Shell Debug Options Window Help
ARIMA(0, 1, 1) RMSE=646.645
ARIMA(0, 1, 2) RMSE=638.128
ARIMA(0, 2, 0) RMSE=1213.082
ARIMA(0, 2, 1) RMSE=760.084
ARIMA(0, 2, 2) RMSE=805.342
ARIMA(1, 0, 0) RMSE=1095.377
ARIMA(1, 0, 2) RMSE=957.999
ARIMA(1, 1, 0) RMSE=651.575
ARIMA(1, 1, 1) RMSE=636.148
ARIMA(1, 2, 0) RMSE=1058.591
ARIMA(2, 0, 0) RMSE=952.168
ARIMA(2, 0, 1) RMSE=812.534
ARIMA(2, 0, 2) RMSE=1036.198
ARIMA(2, 1, 0) RMSE=607.041
ARIMA(2, 1, 1) RMSE=624.154
ARIMA(2, 1, 2) RMSE=682.905
ARIMA(2, 2, 0) RMSE=946.615
ARIMA(2, 2, 1) RMSE=653.151
Best ARIMA(2, 1, 0) RMSE=607.041
Press enter to go to the results of ARIMA with best order)ARIMA(2, 2, 1) RMSE=653.151
ARIMA MODEL RESULTS
predicted=895.718415, expected=2100.000000
predicted=1372.255851, expected=1300.000000
predicted=1470.099510, expected=2300.000000
predicted=1736.858495, expected=1400.000000
predicted=1631.312395, expected=1900.000000
predicted=1641.279185, expected=2300.000000
predicted=1870.960619, expected=2000.000000
predicted=2002.472805, expected=1600.000000
predicted=1776.015295, expected=2200.000000
predicted=1813.569768, expected=2100.000000
predicted=1953.345558, expected=600.000000
predicted=1382.579447, expected=1700.000000
Ln: 58 Col: 0

```

Figure: 17 ARIMA model results

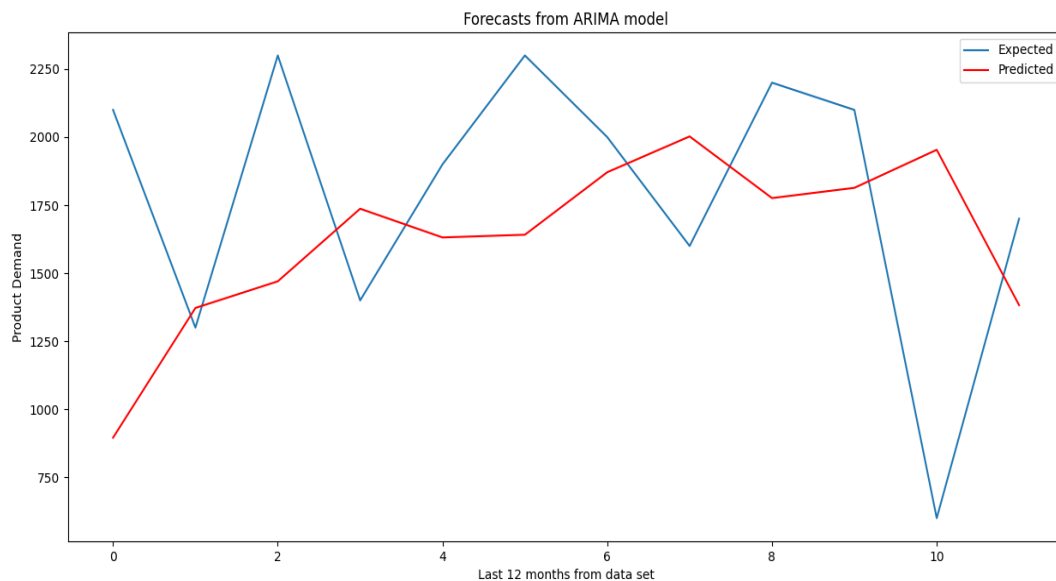


Figure: 18 ARIMA model results graph

After we close the graph window, the script prints the ARIMA model results summary.

```

Python 3.9.0 Shell
File Edit Shell Debug Options Window Help
predicted=1736.858495, expected=1400.000000
predicted=1631.312395, expected=1900.000000
predicted=1641.279185, expected=2300.000000
predicted=1870.960619, expected=2000.000000
predicted=2002.472805, expected=1600.000000
predicted=1776.015295, expected=2200.000000
predicted=1813.569768, expected=2100.000000
predicted=1953.345558, expected=600.000000
predicted=1382.579447, expected=1700.000000
ARIMA Model Results
=====
Dep. Variable:          D.y      No. Observations:          58
Model:                ARIMA(2, 1, 0)  Log Likelihood          -498.015
Method:               css-mle    S.D. of innovations      1292.650
Date:                 Thu, 18 Mar 2021  AIC              1004.030
Time:                 22:19:11      BIC              1012.271
Sample:               1          HQIC              1007.240

=====
              coef      std err          z      P>|z|      [0.025      0.975]
-----
const          -69.4279      96.377          -0.720      0.471     -258.324      119.468
ar.L1.D.y       -0.5916       0.129          -4.575      0.000       -0.845     -0.338
ar.L2.D.y       -0.1868       0.131          -1.422      0.155       -0.444       0.071
=====
Roots
=====
              Real          Imaginary      Modulus      Frequency
-----
AR.1          -1.5838          -1.6870j          2.3140          -0.3700
AR.2          -1.5838           +1.6870j          2.3140           0.3700
=====
Press enter to go to the next model)

```

Figure: 19 ARIMA model summary

After that the script asks us press 'enter' to go in next model.

The script prints the next algorithm summary and graph. Next algorithm is SARIMAX, which runs with the same order p,q,d as the ARIMA but with a specific predefined seasonal order, in our case we choose (1, 2, 1, 12).

```

Python 3.9.0 Shell
File Edit Shell Debug Options Window Help
AR.2          -1.5838          +1.6870j          2.3140           0.3700
=====
Press enter to go to the next model)
SARIMAX Results
=====
Dep. Variable:          Product_0001  No. Observations:          48
Model:                SARIMAX(2, 1, 0)x(2, 1, [1], 12)  Log Likelihood          -307.017
Date:                 Thu, 18 Mar 2021  AIC              626.035
Time:                 22:24:51      BIC              635.367
Sample:               01-01-2012  HQIC              629.256
                  - 12-01-2015
Covariance Type:      opg
=====
              coef      std err          z      P>|z|      [0.025      0.975]
-----
ar.L1          -0.5703      0.279          -2.042      0.041       -1.117     -0.023
ar.L2          -0.1878      0.293          -0.642      0.521       -0.762     0.386
ar.S.L12       -0.2261      0.361          -0.626      0.531       -0.934     0.482
ar.S.L24       -0.2640      0.297          -0.888      0.375       -0.847     0.319
ma.S.L12       -0.9676      0.361          -2.682      0.007       -1.675     -0.260
sigma2         1.553e+06      2.49e-07      6.23e+12      0.000      1.55e+06      1.55e+06
=====
Ljung-Box (L1) (Q):          0.26      Jarque-Bera (JB):          27.16
Prob(Q):                   0.61      Prob(JB):              0.00
Heteroskedasticity (H):      0.44      Skew:                  1.44
Prob(H) (two-sided):         0.17      Kurtosis:              6.22
=====
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 3.14e+28. Standard errors may be unstable.

```

Figure: 20 SARIMAX model results

In this graph, we see also with grey color the simulations of the algorithm in order to see how it fits in general terms with our timeseries data.

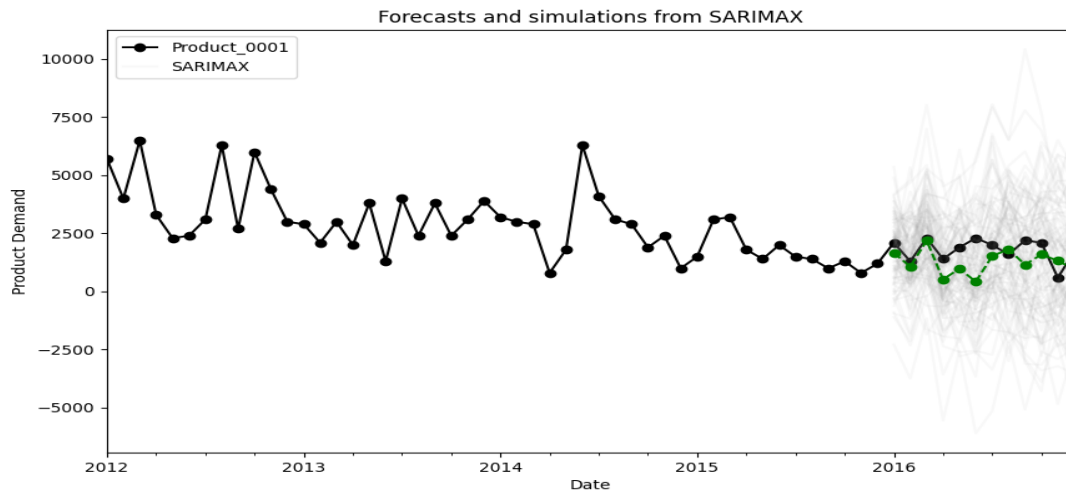


Figure: 21 SARIMAX model graph with simulations

After we close this graph window we get a graph without the simulations which shows us only the historical data of our timeseries, the predicted and the expected all in different colors to can compare them.

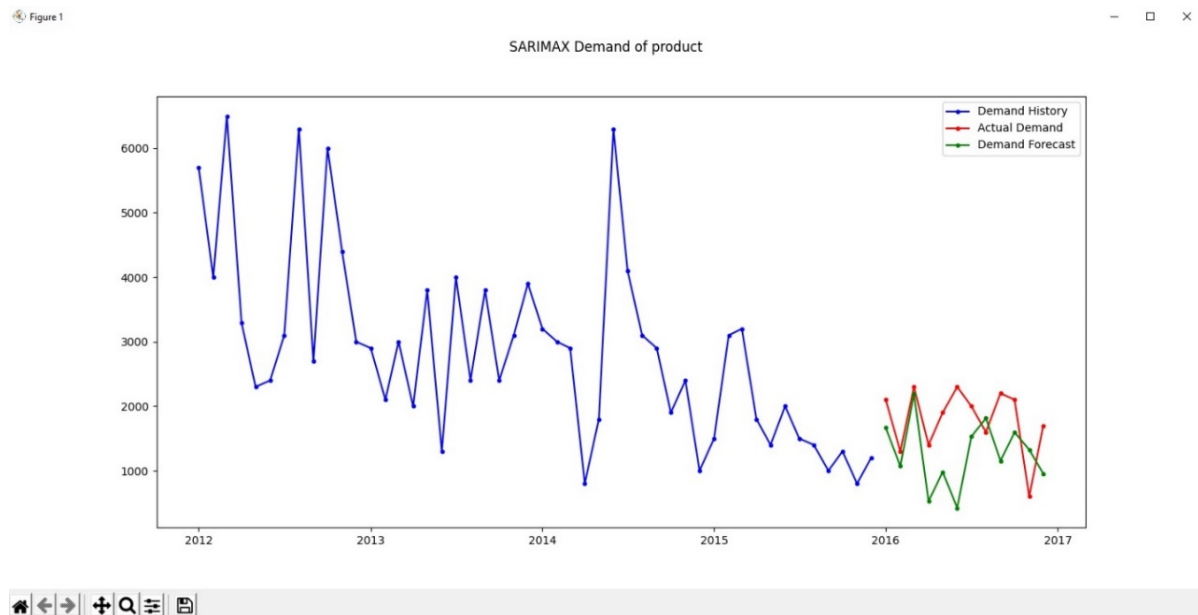


Figure: 22 SARIMAX results graph

After we close this window the script again asks us to press 'enter' in order to go to the next algorithm model.

```

Python 3.5.0 Shell
File Edit Shell Debug Options Window Help

Press enter to go to the next model)

===== SARIMAX Results =====
Dep. Variable: Product_0001 No. Observations: 48
Model: SARIMAX(2, 1, 0)x(2, 1, [1], 12) Log Likelihood: -307.017
Date: Thu, 18 Mar 2021 AIC: 626.035
Time: 22:24:51 BIC: 635.367
Sample: 01-01-2012 HQIC: 629.256
- 12-01-2015
Covariance Type: opg
=====
coef std err z P>|z| [0.025 0.975]
-----
ar.L1 -0.5703 0.279 -2.042 0.041 -1.117 -0.023
ar.L2 -0.1878 0.293 -0.642 0.521 -0.762 0.386
ar.S.L12 -0.2261 0.361 -0.626 0.531 -0.934 0.482
ar.S.L24 -0.2640 0.297 -0.888 0.375 -0.847 0.319
ma.S.L12 -0.9676 0.361 -2.682 0.007 -1.675 -0.260
sigma2 1.553e+06 2.49e-07 6.23e+12 0.000 1.55e+06 1.55e+06
=====
Ljung-Box (L1) (Q): 0.26 Jarque-Bera (JB): 27.16
Prob(Q): 0.61 Prob(JB): 0.00
Heteroskedasticity (H): 0.44 Skew: 1.44
Prob(H) (two-sided): 0.17 Kurtosis: 6.22
=====

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 3.14e+28. Standard errors may be unstable.

Press enter to go to the next model)
multiplicative

```

Figure: 23 Holt Winters model selection

Next is the Holt Winters algorithm. The script checks our timeseries data if have any zeros in order to apply the suitable model, additive if have any zeros or multiplicative if not have any zeros. It prints us that model uses and then prints the comparison graph with the simulations.

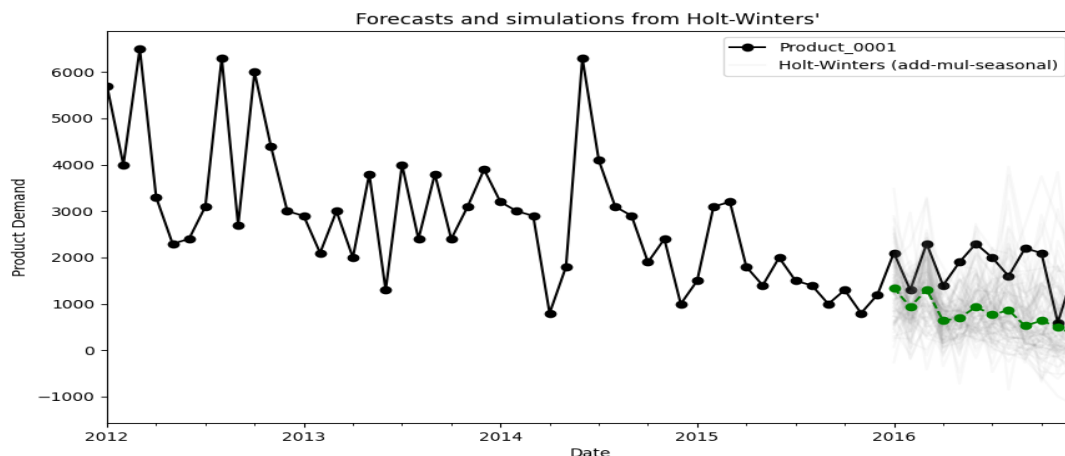


Figure: 24 Holt Winters graph with simulations

After we close the graph window, the script prints the graph with historical, expected and predicted date with different colors to can compare and we finally after we get the model summary printed.

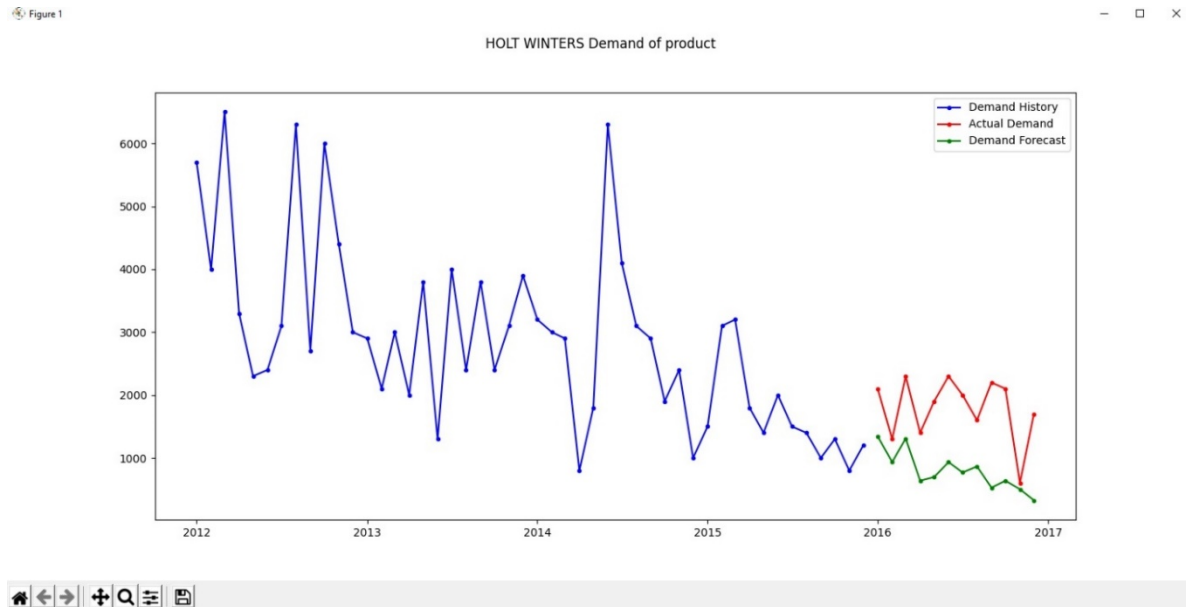


Figure: 25 Holt Winters Results graph

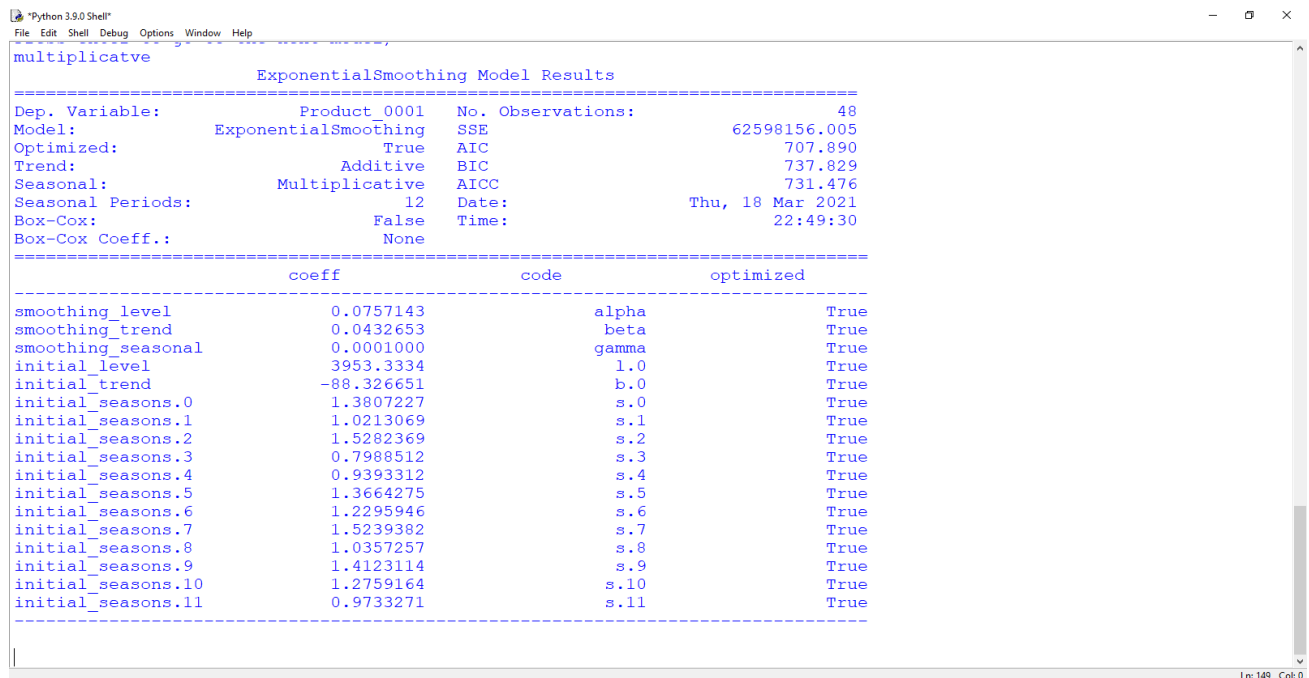


Figure: 26 Holt Winters model summary

After close this graph window the script prints all the algorithm errors in order to can see which algorithm is better to use according each error.

```

Python 3.9.0 Shell
File Edit Shell Debug Options Window Help
-----
initial_seasons.4      0.9393312      s.4      True
initial_seasons.5      1.3664275      s.5      True
initial_seasons.6      1.2295946      s.6      True
initial_seasons.7      1.5239382      s.7      True
initial_seasons.8      1.0357257      s.8      True
initial_seasons.9      1.4123114      s.9      True
initial_seasons.10     1.2759164      s.10     True
initial_seasons.11     0.9733271      s.11     True
-----

HOLT WINTERS ERRORS
Mean Absolute Error: 1000.9160962647406
Mean Squared Error: 1204833.6590042133
Mean Absolute Percentage Error: 0.5285132611888783
Root Mean Squared Error: 1097.649151142665

SARIMAX ERRORS
Mean Absolute Error: 679.2166225917823
Mean Squared Error: 673358.4666228506
Mean Absolute Percentage Error: 0.42404656988817924
Root Mean Squared Error: 820.584222747946

ARIMA ERRORS
Mean Absolute Error: 523.6165062263209
Mean Squared Error: 428047.540725034
Mean Absolute Percentage Error: 0.39548150030294776
Root Mean Squared Error: 654.2534224022324

RANDOM FOREST REGRESSOR ERRORS
Mean Absolute Error: 431.417
Mean Squared Error: 303127.750
Mean Absolute Percentage Error: 0.337
Root Mean Squared Error: 550.5703860543173
Press enter to exit ;)
Ln 149 Col 0

```

Figure: 27 Errors of the compared algorithms

The script is finished and ask us to press ‘enter’ to exit.

10.6 Results of python script for algorithm comparison for n timeseries

Our python script for the n timeseries is a lite version (without plots and prints of the models) of the above script for the one timeseries with some extra modifications. It provides us as result an .xlsx excel file with 4sheets 1 per chosen algorithm. In each sheet, script has save the errors for each product demand timeseries in separate column. When run it opens the open file dialog in order to choose our input file. Then it asks us to enter the start column of our file (if 1st column is the index (dates) then we type 0 and hit ‘enter’, then we add the last column number and hit ‘enter’ again. After these numbers, the script asks us to type the value of the estimators for the random forest regressor.

```

Python 3.9.0 Shell
File Edit Shell Debug Options Window Help
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct 5 2020, 15:34:40) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\User\Desktop\program last\timeseries forecast n.py =====
Enter value for the start column of the file (if 1st is the index (dates) type 0): 0
Enter value for the last column of the file: 1381
Enter value for estimators of random forest regressor: 100
OK
|
Ln 9 Col 0

```

Figure: 28 column start and end & estimators for random forest regressor

```
Python 3.8.0 Shell
File Edit Shell Debug Options Window Help
Mean Absolute Error: 475.750
Mean Squared Error: 349457.917
Mean Absolute Percentage Error: 292358675810134720.000
Root Mean Squared Error: 591.149656742408
Product_0017
OK
OK
OK
HOLT WINTERS ERRORS
Mean Absolute Error: 530.36336462355
Mean Squared Error: 338420.27710360393
Mean Absolute Percentage Error: 3.5127071908973875e+17
Root Mean Squared Error: 581.7390111584438

SARIMAX ERRORS
Mean Absolute Error: 821.3704924377499
Mean Squared Error: 1003407.5095351697
Mean Absolute Percentage Error: 6.091445506643706e+17
Root Mean Squared Error: 1001.70230584499

ARIMA ERRORS
Mean Absolute Error: 351.96979916526647
Mean Squared Error: 179633.46133638328
Mean Absolute Percentage Error: 2.992367671701433e+17
Root Mean Squared Error: 423.83187862215283

RANDOM FOREST REGRESSOR ERRORS
Mean Absolute Error: 465.083
Mean Squared Error: 308175.250
Mean Absolute Percentage Error: 400069766898079040.000
Root Mean Squared Error: 555.1353438576939
Product_0018
OK
```

After script ends and close, we open the file and after confirm that it is correct filled. In our case we have approx. 1380 timeseries from years 2012-2016 and the script execution time was approx. 24hours by using PC with CPU i5-2410M 2.3GHz 4,00GB RAM.

Figure: 30 xlsx file script output

Then we create one extra sheet named “comparison” and we fill it in same way as the other sheets. In the place of the errors now, we use this function that tell us for each product and for each type of error which algorithm has the smallest error and is the best to use for more accurate forecast in this timeseries.

	Product_1	Product_2	Product_3	Product_4	Product_5	Product_6	Product_7	Product_8	Product_9	Product_10
Mean Absolute Error	RANDOM F	RANDOM F	ARIMA	ARIMA	RANDOM FOREST	RANDOM F	HOLT WINTERS	ARIMA	HOLT WIN	RANDOM
Mean Squared Error	RANDOM F	RANDOM F	ARIMA	ARIMA	RANDOM FOREST	RANDOM F	ARIMA	ARIMA	HOLT WIN	RANDOM
Mean Absolute Percentage Error	RANDOM F	RANDOM F	HOLT WINTERS	ARIMA	RANDOM FOREST	RANDOM F	HOLT WINTERS	SARIMAX	SARIMAX	HOLT WIN
Root Mean Squared Error	RANDOM F	RANDOM F	ARIMA	ARIMA	RANDOM FOREST	RANDOM F	ARIMA	ARIMA	HOLT WIN	RANDOM
TOTAL BEST ALGORITHM PER ERROR										
Mean Absolute Error	COUNT	RANK								
ARIMA	955	1								
HOLT WINTERS	189	3								
RANDOM FOREST	202	2								
SARIMAX	34	4								
Mean Squared Error	COUNT	RANK								
ARIMA	1027	1								
HOLT WINTERS	152	3								
RANDOM FOREST	161	2								
SARIMAX	40	4								
Mean Absolute Percentage Error	COUNT	RANK								
ARIMA	640	1								
HOLT WINTERS	330	2								
RANDOM FOREST	257	3								
SARIMAX	153	4								

Figure: 31 Comparison sheet in xlsx file

```
=IF((MIN('ERRORS HOLT WINTERS'!B2;'ERRORS SARIMAX'!B2;'ERRORS
ARIMA'!B2;'ERRORS RANDOM FOREST'!B2)='ERRORS HOLT WINTERS'!B2);"HOLT
WINTERS";
(IF((MIN('ERRORS HOLT WINTERS'!B2;'ERRORS SARIMAX'!B2;'ERRORS
ARIMA'!B2;'ERRORS RANDOM FOREST'!B2)='ERRORS SARIMAX'!B2);"SARIMAX";
(IF((MIN('ERRORS HOLT WINTERS'!B2;'ERRORS SARIMAX'!B2;'ERRORS
ARIMA'!B2;'ERRORS RANDOM FOREST'!B2)='ERRORS
ARIMA'!B2);"ARIMA";"RANDOM FOREST")))))
```

In function where B2 is the cell with the error, we want to compare in algorithms. It is a set of IF and MIN excel functions which give us as result

- HOLT WINTERS if the min error comes from Holt Winters’ algorithm
- SARIMAX if the min error comes from SARIMAX algorithm
- ARIMA if the min error comes from ARIMA algorithm
- RANDOM FOREST if the min error comes from Random Forest Regressor algorithm

Next step in results processing, is to find which algorithm is better in each product timeseries per error. This because is possible in each error to have different best algorithm and for that reason we must implement a function which finds the algorithm which is the best in each error in order to take it as the best algorithm for this product forecast and for this error. Moreover, we rank the comparing algorithms based on the number of the timeseries we found them as best. We do this by using the bellow steps three times, one for each error (except the RMSE because it has same results as the MSE) in order to find the algorithm based on the error type.

1. We create an array with the name of each algorithm in one column, the counts of how many times find as best in second column and the rank of them in third column.
2. The counts finds be using the function =COUNTIF(\$B\$7:\$U\$7;A12) where B7:A7 is the line in which we have all the best algorithms and A12 is the name of the algorithm we want to count.
3. The rank we find by using the function =RANK(B12;\$B\$12:\$B\$15) where B12 is the count of the algorithm we want to rank and B12:B15 is the column with all the counts.
4. Final step is to put in ascending order based on ranks and after that we have first the TOP best algorithm.

So, after all these processes we conclude that the best algorithms rank per error as bellow

TOTAL BEST ALGORITHM PER ERROR		
Mean Absolute Error	COUNT	RANK
<u>ARIMA</u>	<u>955</u>	<u>1</u>
HOLT WINTERS	189	3
RANDOM FOREST	202	2
SARIMAX	34	4
Mean Squared Error	COUNT	RANK
<u>ARIMA</u>	<u>1027</u>	<u>1</u>
HOLT WINTERS	152	3
RANDOM FOREST	161	2
SARIMAX	40	4
Mean Absolute Percentage Error	COUNT	RANK
<u>ARIMA</u>	<u>640</u>	<u>1</u>
HOLT WINTERS	330	2
RANDOM FOREST	257	3
SARIMAX	153	4

11. Conclusions

Our literature review conclude that in supply chain management ML has many applications like usage of Supervised ML application for effective supplier classification, stock-out prediction algorithms based in ML, ML usage to find business partners and building mutual relationship, ML

demand forecasting models, production planning models based in ML and many more. In this thesis we focus on demand forecasting, which is a main part of the financial planning and logistics planning of business for any organization. It can be defined as one tool which uses the statistics of the past and the now sales in order to give us forecast for the future demand. To can implement our work on this we decide to work with timeseries demand datasets. For this reason, we analyze timeseries and we understand that especially in seasonality demand is very important to have the demand structured in timeseries in time step based on our needs (in our case we work with most used which is monthly demand data). We make a script, which is very helpful tool to can choose the algorithm, which is more suitable for our timeseries, and can give us, the most accurate forecast of demand. Our script helps to find a solution, we compare ML models implemented with algorithms such as Random Forest Regressor with the rest wide used statistic like ARIMA, SARIMAX and Holt Winters and we find the best option. After that, we have a clear view which algorithm, we use in order to get the best possible forecast for demand in our data. We conclude that the best algorithm for all type of errors is the ARIMA algorithm according to the chosen dataset and thus satisfy the aim of this thesis. Next comes the RANDOM FOREST REGRESSOR in all errors except Mean Absolute Percentage Error. ARIMA has the best forecasting results in all errors in this dataset of timeseries. Is possible that in other dataset with different kind of timeseries the ranking can be different. Moreover, as we can see the rest algorithms have small differences in them counts. SARIMAX model has the smallest ranking in all errors that make this algorithm not valid for our dataset.

We also conclude that the python script is a very helpful tool, which can run multiple models, and algorithms per time. Instead of other software like Weka that cannot do this job so easy because can run only one model at a time and are not able to give automatically and fast, this kind of results as an output like python scripts do. Having in mind all these conclusions our scripts can be used very easily with real data and can help companies or logistic service providers forecast them product demand (or in case of service providers, them clients' needs and base on these forecast them needs in space and in transportation) by using them historical data.

12. Future Work

As future work for this thesis we suggest the comparison of the performance evaluation results of the chosen algorithm with real time data (live acquired by interconnections with ERP or WMS software) or even bigger dataset of timeseries in order to establish the results in a better more 'solid' way. We propose this for future research as our thesis use a dataset with historical data which we modify for our needs, if have a live interconnection with the software we can use only the data we need without have any other program usage but only our script and we do not need any dataset preparation, too.

Moreover, as an extra, we can implement in our script a routine, which test seasonal order in SARIMAX like the one we do for the ARIMA model, but this will increase the execution time too much and this will be an issue especially in cases we need fast results. We assume approximately that if we will keep the order from ARIMA and make tests in seasonal order the time will be 4 times longer than the now script. Is possible this delay to can be solve by using other programming

language or other better modules which can implement faster the calculations, this can be also a proposal for future research , to can make the scripts faster.

One more proposal for future research is, as be using the scripts we have evaluated the algorithms based on them errors and rank them, we can make predictions by using the most suitable one in order to make forecasts for demand without have real data available to compare the forecasts as in our thesis example. For example, predict next month or next year demand and after the time pass and we will have in our possession the real demand data we will compare and see if the forecast is correct and help us to have good predictions in demand and seasonality. The better the forecast, the higher will be the satisfied clients' needs based on this forecast or the forecast was not correct and cause us stock outs or over stock and many other problems which can lead to unsatisfied clients and bad company reputation of our firm.

We can use algorithms from the rest categories like baseline, linear, nonlinear ML and of course deep learning and Hybrid models in order to see and evaluate them reaction in the same way as we do in this thesis with the selected algorithms.

Finally, if we want to propose a ML application except the demand forecasting which we implement in this thesis we will propose a same structure future research but focused in the production planning and more specific to the machine task allocation problem which can be solved by using ML approaches and algorithms like classifiers. This sector of supply chain management has same big importance as the demand forecasting.

References

1. Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. (2018) *Foundations of Machine Learning*. 2nd ed. MIT Press
2. Djordje Cica, Branislav Sredanovic, Sasa Tesic and Davorin Kramar. (2020) Predictive modeling of turning operations under different cooling/lubricating conditions for sustainable manufacturing with machine learning techniques. Published in Applied Computing and Informatics. 2210-8327 DOI 10.1016/j.aci.2020.02.001
3. Wenzel, Hannah; Smit, Daniel; Sardesai, Saskia (2019): A literature review on machine learning in supply chain management, In: Kersten, Wolfgang Blecker, Thorsten Ringle, Christian M. (Ed.): Artificial Intelligence and Digital Transformation in Supply Chain Management: Innovative Approaches for Supply Chains. Proceedings of the Hamburg International Conference of Logistics (HICL), Vol. 27, ISBN 978-3-7502-4947-9, epubli GmbH, Berlin, pp. 413-441, <http://dx.doi.org/10.15480/882.2478>
4. Aurélien Géron (2017) *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media, Inc. ISBN: 9781491962299
5. Elcio Tarallo, Getúlio K. Akabane, Camilo I. Shimabukuro, Jose Mello and Douglas Amancio. (2019) Machine Learning in Predicting Demand for Fast-Moving Consumer Goods: An Exploratory Research. IFAC
6. Álvaro Silva de Melo (2019) A Machine Learning Approach to the Optimization of Inventory Management Policies. FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO
7. Ethem Alpaydın (2010) *Introduction to Machine Learning*. 2nd ed. The MIT Press Cambridge, Massachusetts London, England

8. Taiwo Oladipupo Ayodele. (2010) Types of machine learning algorithms. In New advances in machine learning. IntechOpen
9. F. Lolli, E. Balugani, A. Ishizaka, R. Gamberini, B. Rimini & A. Regattieri (2019) Machine learning for multi-criteria inventory classification applied to intermittent demand, *Production Planning & Control*, 30:1, 76-89, DOI: 10.1080/09537287.2018.1525506
10. Samiul Islam and Saman Hassanzadeh Amin (2020) Prediction of probable backorder scenarios in the supply chain using Distributed Random Forest and Gradient Boosting Machine learning techniques. *J Big Data* 7:65 <https://doi.org/10.1186/s40537-020-00345-2>
11. Makkar, Sandhya & G.Naga Rama Devi, Dr & Solanki, Vijender. (2020). Applications of Machine Learning Techniques in Supply Chain Optimization. 10.1007/978-981-13-8461-5_98.
12. Manuel Woschank, Erwin Rauch and Helmut Zsifkovits (2020) A Review of Further Directions for Artificial Intelligence, Machine Learning, and Deep Learning in Smart Logistics. *Sustainability* 2020, 12, 3760; doi:10.3390/su12093760
13. Alexandra Brintrup , Johnson Pak , David Ratiney , Tim Pearce , Pascal Wichmann , Philip Woodall & Duncan McFarlane (2020) Supply chain data analytics for predicting supplier disruptions: a case study in complex asset manufacturing, *International Journal of Production Research*, 58:11, 3330-3341, DOI: 10.1080/00207543.2019.1685705
14. Özden Gür Ali, Serpil Sayın, Tom van Woensel and Jan Fransoo (2009) SKU demand forecasting in the presence of promotions. *Expert Systems with Applications* 36 12340–12348
15. Oroojlooy, Afshin, "Applications of Machine Learning in Supply Chains" (2019). Theses and Dissertations. 4364. <https://preserve.lehigh.edu/etd/4364>
16. Ian M. Cavalcantea, Enzo M. Frazzonb, Fernando A. Forcellinia and Dmitry Ivanovc (2019) A supervised machine learning approach to data-driven simulation of resilient supplier selection in digital manufacturing. *International Journal of Information Management* 49 (2019) 86–97
17. Hokey Min (2010) Artificial intelligence in supply chain management: theory and applications, *International Journal of Logistics: Research and Applications*, 13:1, 13-39, DOI:10.1080/13675560902736537
18. Paolo Priore, Borja Ponte, Rafael Rosillo & David de la Fuente (2019) Applying machine learning to the dynamic selection of replenishment policies in fast-changing supply chain environments, *International Journal of Production Research*, 57:11, 3663-3677, DOI:10.1080/00207543.2018.1552369
19. Junichiro Mori, Yuya Kajikawa, Hisashi Kashima and Ichiro Sakata (2012) Machine learning approach for finding business partners and building reciprocal relationships, *Expert Systems with Applications*, Volume 39, Issue 12, Pages 10402-10407, ISSN 0957-4174, <https://doi.org/10.1016/j.eswa.2012.01.202>
20. Chitriki Thotappa, Dr. K.Ravindranath (2010) Data mining Aided Proficient Approach for Optimal Inventory Control in Supply Chain Management. *Proceedings of the World Congress on Engineering 2010 Vol 1 WCE 2010*, June 30 - July 2, 2010, London, U.K.
21. Beardslee, E.A., & Trafalis, T.B. (2005). Data Mining Methods In A Metrics-deprived Inventory Transactions Environment. *WIT Transactions on Information and Communication Technologies*, 35.
22. Makridakis, Spyros & Hyndman, Rob & Petropoulos, Fotios. (2019). Forecasting in social settings: The state of the art. *International Journal of Forecasting*. 36. 10.1016/j.ijforecast.2019.05.011.
23. Javad Feizabadi (2020) Machine learning demand forecasting and supply chain performance, *International Journal of Logistics Research and Applications*, DOI: 10.1080/13675567.2020.1803246

24. Mahya Seyedan and Fereshteh Mafakheri (2020) Predictive big data analytics for supply chain demand forecasting: methods, applications, and research opportunities. *J Big Data* 7:53
25. T. Lauer, S. Legner (2019) Plan Instability prediction by machine learning in master production planing. *IEEE 15th international conference on automation science and engineering*
26. Peter O'Donovan, Kevin Leahy, Ken Bruton and Dominic T. J. O'Sullivan (2015) Big data in manufacturing: a systematic mapping study. *Journal of Big Data* 2:20 DOI 10.1186/s40537-015-0028-x
27. Elcio Tarallo, Getúlio K. Akabane, Camilo I. Shimabukuro, Jose Mello and Douglas Amancio (2019) Machine learning in predicting demand for fast moving consumer goods: an exploratory research. *IFAC PapersOnLine* 52-13 737–742.
28. Mahya Seyedan and Fereshteh Mafakheri (2020) Predictive big data analytics for supply chain demand forecasting: methods, applications, and research opportunities. *J Big Data* 7:53 <https://doi.org/10.1186/s40537-020-00329-2>
29. Ratnadip Adhikari, R. K. Agrawal (2013) *An Introductory Study on Time Series Modeling and Forecasting*. LAP Lambert Academic Publishing, Germany
30. Jason Brownlee (2019) *A Tour of Machine Learning Algorithms* [WWW] Available from: <https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/> [Accessed 20/3/2020]
31. X. Zhu and M. Shen, (2012) "Based on the ARIMA model with grey theory for short term load forecasting model," *International Conference on Systems and Informatics*, Yantai, pp. 564-567, doi: 10.1109/ICSAI.2012.6223060.
32. Fattah, Jamal & Ezzine, Latifa & Aman, Zineb & Moussami, Haj & Lachhab, Abdeslam. (2018). Forecasting of demand using ARIMA model. *International Journal of Engineering Business Management*. 10. 184797901880867. 10.1177/1847979018808673.
33. Tinni Chaudhuri, Prashant Verma, Mukti Khetan (2020) Modeling and Forecasting of Rice Production in Some Major States of India using ARIMA. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)* ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429 Volume 8 Issue XII
34. Andreea-Cristina PETRICĂ, Stelian STANCU and Alexandru TINDECHE (2016) Limitation of ARIMA models in financial and monetary economics. *Theoretical and Applied Economics* Volume XXIII, No. 4(609), Winter, pp. 19-42
35. Meyler, Aidan and Kenny, Geoff and Quinn, Terry (1998) *Forecasting Irish inflation using ARIMA models* Central Bank and Financial Services Authority of Ireland.
36. Stylianos I. Vagropoulos, G. I. Chouliaras, E. G. Kardakos, C. K. Simoglou, A. G. Bakirtzis (2016) Comparison of SARIMAX, SARIMA, Modified SARIMA and ANN-based Models for Short-Term PV Generation Forecasting, 978-1-4673-8463-6/16/\$31.00 IEEE
37. Shadkam, A. (2020). Using SARIMAX to forecast electricity demand and consumption in university buildings (T). *University of British Columbia*. Retrieved from <https://open.library.ubc.ca/collections/ubctheses/24/items/1.0391009>
38. Marjan, Čeh & Kilibarda, Milan & Lisec, Anka & Bajat, Branislav. (2018). Estimating the Performance of Random Forest versus Multiple Regression for Predicting Prices of the Apartments. *ISPRS International Journal of Geo-Information*. 7. 168. 10.3390/ijgi7050168.
39. Ouedraogo, I., Defourny, P. & Vanclooster, M. (2019) Application of random forest regression and comparison of its performance to multiple linear regression in modeling groundwater nitrate concentration at the African continent scale. *Hydrogeol J* 27, 1081–1098. <https://doi.org/10.1007/s10040-018-1900-5>

40. Afroz Chakure. (2019) Random Forest Regression Along with its implementation in Python. Medium (The startup). Weblog [Online] 29 Jun. Available from: <https://medium.com/swlh/random-forest-and-its-implementation-71824ced454f>. [Accessed 27/1/2021].
41. Hyndman, R.J., & Athanasopoulos, G. (2018) Forecasting: principles and practice, 2nd edition, OTexts: Melbourne, Australia. OTexts.com/fpp2. [Accessed 28/1/2021].
42. Hansun, Seng & .V, Charles & Indrati, Ch & Seno Saleh, Subanar. (2019). Revisiting the Holt-Winters' Additive Method for Better Forecasting. International Journal of Enterprise Information Systems. 15. 43-57. 10.4018/IJEIS.2019040103.
43. Muhammad Aamir and Ani Shabri (2016) Modelling and Forecasting Monthly Crude Oil Price of Pakistan: A Comparative Study of ARIMA, GARCH and ARIMA Kalman Model. AIP Conference Proceedings 1750, 060015
44. Saigal S. and Mehrotra D. (2012) Performance Comparison of Time Series Data Using Predictive Data Mining Techniques. Advances in Information Mining, ISSN: 0975-3265 & E-ISSN: 0975-9093, Volume 4, Issue 1, pp.-57-66.
45. Nicolas Vandeput (2019) Forecast KPIs: RMSE, MAE, MAPE & Bias [WWW] Available from: <https://towardsdatascience.com/forecast-kpi-rmse-mae-mape-bias-cdc5703d242d> [Accessed 20/3/2020]
46. Exceltip (n.d.) Split Excel Sheet Into Multiple Files Based On Column Using VBA [WWW] Exceltip. Available from: <https://www.exceltip.com/general-topics-in-vba/split-excel-sheet-into-multiple-files-based-on-column-using-vba.html> [Accessed 20/11/2020]
47. Alansidman (2013) macro to copy column B & C on all sheets in workbook to one master sheet side by side [WWW] mrexcel. Available from: <https://www.mrexcel.com/board/threads/macro-to-copy-column-b-c-on-all-sheets-in-workbook-to-one-master-sheet-side-by-side.718664> [Accessed 20/11/2020]
48. Pandas (n.d.) About pandas. [WWW] Pandas. Available from: <https://pandas.pydata.org/about/> [Accessed 20/1/2021]
49. Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay (2011) Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830
50. Seabold, Skipper, and Josef Perktold. (2010) "statsmodels: Econometric and statistical modeling with python." Proceedings of the 9th Python in Science Conference.
51. Harris, C.R., Millman, K.J., van der Walt, S.J. et al.(2020) Array programming with NumPy. Nature 585, 357–362. DOI: 0.1038/s41586-020-2649-2.
52. Python (n.d.) tkinter — Python interface to Tcl/Tk [WWW] Python Available from: <https://docs.python.org/3/library/tkinter.html> [Accessed 20/1/2021]
53. Jmcnamara (2020) xlsx writer [WWW] github Available from: <https://github.com/jmcnamara/XlsxWriter> [Accessed 20/1/2021]

Figures references

1. https://www.7wdata.be/wp-content/uploads/2017/08/ml_typespng.png [Accessed 20/11/2020]
2. https://upload.wikimedia.org/wikipedia/commons/thumb/1/10/Iris_Flowers_Clustering_kMeans.svg/450px-Iris_Flowers_Clustering_kMeans.svg.png
3. https://miro.medium.com/max/1199/1*N8UXaiUKWurFLdmEhEHjWg.jpeg
4. <https://www.javatpoint.com/applications-of-machine-learning>

5. <https://www.knowledgepublisher.com/assets/industrial-revolutions.png>
6. <https://rpubs.com/Edna/634742>
7. <https://corporatefinanceinstitute.com/resources/knowledge/other/random-forest/>

APPENDIX 1:Python script code for the comparison of the algorithm errors applicable in one timeseries

```
import pandas as pd

from matplotlib import pyplot as plt

from statsmodels.tsa.holtwinters import ExponentialSmoothing as HWES

from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing, Holt

from sklearn.metrics import mean_squared_error

from math import sqrt

from sklearn.metrics import
mean_absolute_error,mean_squared_error,mean_absolute_percentage_error

from matplotlib.pylab import rcParams

from tkinter.filedialog import askopenfilename

from statsmodels.tsa.seasonal import seasonal_decompose

from statsmodels.tsa.statespace.sarimax import SARIMAX

from statsmodels.tsa.arima_model import ARIMA

from sklearn.ensemble import RandomForestRegressor

from numpy import asarray

import numpy

import warnings

import pylab

# Ignore harmless warnings
```

```

warnings.filterwarnings("ignore")

#figures size
rcParams['figure.figsize'] = 15, 7

#open file
myfile = askopenfilename()

#read the data file. the date column must be in the mm-dd-yyyy format.
df = pd.read_csv(myfile, header=0, infer_datetime_format=True, parse_dates=[0],
index_col=[0],delimiter=';')
df.head()

#plot the data from file
df.plot()
plt.show()

# make ETS Decomposition
result = seasonal_decompose(df,model = 'multiplicative')

# plot the ETS
result.plot()
plt.show()

#split between the training and the test data sets. The last 12 periods form the test data
df_train = df.iloc[:-12]
df_test = df.iloc[-12:]

#RANDOM FOREST
print ('RANDOM FOREST REGRESSOR')

```



```

print()

#we will ask user to type the n estimators for random forest regressor
estims = input("Enter value for estimators: ")
estim = int(estims)

# transform a time series dataset into a supervised learning dataset
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1 if type(data) is list else data.shape[1]
    dfs = pd.DataFrame(data)
    cols = list()
    # input sequence (t-n, ... t-1)
    for i in range(n_in, 0, -1):
        cols.append(dfs.shift(i))
    # forecast sequence (t, t+1, ... t+n)
    for i in range(0, n_out):
        cols.append(dfs.shift(-i))
    # put it all together
    agg = pd.concat(cols, axis=1)
    # drop empty rows without values (NaN)
    if dropnan:
        agg.dropna(inplace=True)
    return agg.values

# split a univariate dataset into train/test groups
def train_test_split(data, n_test):
    return data[:-n_test, :], data[-n_test:, :]

# fit a random forest regressor model and make one step ahead forecast

```

```

def random_forest_forecast(train, testX):
    # transform list into array
    train = asarray(train)
    # split into input and output columns
    trainX, trainy = train[:, :-1], train[:, -1]
    # fit the model
    model = RandomForestRegressor(n_estimators=estim, n_jobs=2)
    model.fit(trainX, trainy)
    # make a one-step ahead forecast
    yhat = model.predict([testX])
    return yhat[0]

# walk-forward validation for the univariate data
def walk_forward_validation(data, n_test):
    predictions = list()
    # split the dataset
    train, test = train_test_split(data, n_test)
    # feed the history with the training dataset
    history = [x for x in train]
    # step ahead each step in the test data group
    for i in range(len(test)):
        # split test row into input and output columns
        testX, testy = test[i, :-1], test[i, -1]
        # fit model on history and make a forecast
        yhat = random_forest_forecast(history, testX)
        # store forecast in list of predictions
        predictions.append(yhat)
    # add actual observation to history array in order to implement the next loop

```

```

history.append(test[i])
# summarize progress
print('>expected=%.1f, predicted=%.1f % (testy, yhat))
# estimate prediction error
error = mean_absolute_error(test[:, -1], predictions)
error1 = mean_squared_error(test[:, -1], predictions)
error2 = mean_absolute_percentage_error(test[:, -1], predictions)
return error, error1, error2, test[:, -1], predictions

# load the dataset
values = df.values
print('RANDOM FOREST REGRESSOR RESULTS')
# transform the time series data into supervised learning data
data = series_to_supervised(values, n_in=6)
# evaluate the model
mae, mse, mape, y, yhat = walk_forward_validation(data, 12)

# plot forecasts against actual values
plt.ylabel("Product Demand")
plt.xlabel("Last 12 months from data set")
plt.plot(y, label='Expected')
plt.plot(yhat, label='Predicted')
plt.title("Forecasts from random forest")
plt.legend()
plt.show()
print()
input("Press enter to go to the next model")

```

```

# ARIMA Model

print('ARIMA MODEL')

print()

# evaluate an ARIMA model for a given order (p,d,q)

def evaluate_arima_model(X, arima_order):

    # prepare the training dataset
    train_size = int(len(X) * 0.66)
    train, test = X[0:train_size], X[train_size:]
    history = [x for x in train]

    # make forecasts
    predictions = list()

    for t in range(len(test)):
        model = ARIMA(history, order=arima_order).fit(optimized=True, use_brute=True)
        yhat = model.forecast()[0]
        predictions.append(yhat)
        history.append(test[t])

    # calculations of sample error
    rmse = sqrt(mean_squared_error(test, predictions))

    return rmse

# function which evaluates combinations of p, d and q values for an ARIMA model

def evaluate_models(dataset, p_values, d_values, q_values):

    dataset = dataset.astype('float32')

    best_score, best_cfg = float("inf"), None

    for p in p_values:
        for d in d_values:

```

```

for q in q_values:
    order = (p,d,q)
    try:
        rmse = evaluate_arma_model(dataset, order)
        if rmse < best_score:
            best_score, best_cfg = rmse, order
        print('ARIMA%s RMSE=%.3f % (order,rmse))
    except:
        continue
    print('Best ARIMA%s RMSE=%.3f % (best_cfg, best_score))
    return best_cfg

series = df

# evaluation parameters for the ARIMA model order (*we define them range)
p_values = [0, 1, 2]
d_values = range(0, 3)
q_values = range(0, 3)
ordl=evaluate_models(series.values, p_values, d_values, q_values)
input("Press enter to go to the results of ARIMA with best order")

print('ARIMA MODEL RESULTS')
# split the data set into train and test groups
X = values
size = int(len(X) * 0.8)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictionsar = list()
# walk-forward validation

```

```

for t in range(len(test)):
    modelar = ARIMA(history, order=(ord1)).fit(optimized=True, use_brute=True)
    output = modelar.forecast()
    yhat = output[0]
    predictionsar.append(yhat)
    obs = test[t]
    history.append(obs)
    print('predicted=%f, expected=%f' % (yhat, obs))

# plot forecasting against the actual values
plt.ylabel("Product Demand")
plt.xlabel("Last 12 months from data set")
plt.plot(test, label='Expected')
plt.plot(predictionsar, label='Predicted', color='red')
plt.title("Forecasts from ARIMA model")
plt.legend()
plt.show()

#print the model summary
print(modelar.summary())
print()
input("Press enter to go to the next model")

#SARIMAX model
model1 = SARIMAX(df_train, order = (ord1), seasonal_order =(2, 1, 1,
12)).fit(optimized=True, use_brute=True)

#print the model summary
model1.summary()

```

```

print(model1.summary())

print()

#define the start and end steps for forecasts
start = len(df_train)
end = len(df_train) + len(df_test) - 1

# forecasting for 12 months against the test set
predictions = model1.predict(start, end, typ = 'levels').rename("Predictions")

#make the simulations of the model
simulations1 = model1.simulate(12, repetitions=100, error='mul', anchor='end')

#plot simulations of the model
ax = df.plot(figsize=(10,6), marker='o', color='black',
              title="Forecasts and simulations from SARIMAX" )
ax.set_ylabel("Product Demand")
ax.set_xlabel("Date")
simulations1.plot(ax=ax, style='-', alpha=0.05, color='grey', legend=False)
predictions.rename('SARIMAX').plot(ax=ax, style='--', marker='o', color='green', legend=True)
plt.show()

# plot predictions and the actual values aside
figS = plt.figure()
figS.suptitle('SARIMAX Demand of product')
pasT, = plt.plot(df_train.index, df_train, 'b.-', label='Demand History')
futurE, = plt.plot(df_test.index, df_test, 'r.-', label='Actual Demand')
predicted_futurE, = plt.plot(df_test.index, predictions, 'g.-', label='Demand Forecast')

```

```

plt.legend(handles=[pasT, futurE, predicted_futurE])
plt.show()

input("Press enter to go to the next model")

#checking for zeros inside the dataset
sum0 = 0
for i in range(len(df)):
    if df.values[i] == 0:
        sum0 = sum0 + 1

#if sum of zero values is zero will run multiplicative holt winters' model
if sum0 == 0:
    #holt winters' build and train the model based on the training data group
    print('multiplicative')

    model = HWES(df_train, seasonal_periods=12, trend='add', seasonal='mul',
initialization_method="estimated").fit(optimized=True, use_brute=True)

    #if sum of zero values is not zero will run additive holt winters' model
else:
    #holt winters' build and train the model based on the training data group
    print('additive')

    model = HWES(df_train, seasonal_periods=12, trend='add', seasonal='add',
initialization_method="estimated").fit(optimized=True, use_brute=True)

#make simulations of model
simulations = model.simulate(12, repetitions=100, error='mul')

#plot the simulations of the algorithm
ax = df.plot(figsize=(10,6), marker='o', color='black',

```



```

        title="Forecasts and simulations from Holt-Winters' " )
ax.set_ylabel("Product Demand")
ax.set_xlabel("Date")
simulations.plot(ax=ax, style='-', alpha=0.05, color='grey', legend=False)
model.forecast(12).rename('Holt-Winters (add-mul-seasonal)').plot(ax=ax, style='--', marker='o',
color='green', legend=True)

plt.show()


#print the model summary
print(model.summary())

print()


#create an out of sample forecast for the next 12 steps ahead the final data point in the training
data set
sales_forecast = model.forecast(steps=12)


#plot the training data, the test data and the forecast on the same graph
fig = plt.figure()
fig.suptitle('HOLT WINTERS Demand of product')
past, = plt.plot(df_train.index, df_train, 'b.-', label='Demand History')
future, = plt.plot(df_test.index, df_test, 'r.-', label='Actual Demand')
predicted_future, = plt.plot(df_test.index, sales_forecast, 'g.-', label='Demand Forecast')
plt.legend(handles=[past, future, predicted_future])
plt.show()


#print all the errors for each algorithm model
print('HOLT WINTERS ERRORS')
print('Mean Absolute Error:',mean_absolute_error(df_test, sales_forecast))
print('Mean Squared Error:',mean_squared_error(df_test, sales_forecast))

```

```
print('Mean Absolute Percentage Error:',mean_absolute_percentage_error(df_test,
sales_forecast))
```

```
rmse=sqrt(mean_squared_error(df_test, sales_forecast))
```

```
print('Root Mean Squared Error:',rmse)
```

```
print()
```

```
print('SARIMAX ERRORS')
```

```
print('Mean Absolute Error:',mean_absolute_error(df_test,predictions))
```

```
print('Mean Squared Error:',mean_squared_error(df_test,predictions))
```

```
print('Mean Absolute Percentage Error:',mean_absolute_percentage_error(df_test,predictions))
```

```
rmse1=sqrt(mean_squared_error(df_test,predictions))
```

```
print('Root Mean Squared Error:',rmse1)
```

```
print()
```

```
print('ARIMA ERRORS')
```

```
print('Mean Absolute Error:',mean_absolute_error(test, predictionsar))
```

```
print('Mean Squared Error:',mean_squared_error(test, predictionsar))
```

```
print('Mean Absolute Percentage Error:',mean_absolute_percentage_error(test, predictionsar))
```

```
rmse1=sqrt(mean_squared_error(test, predictionsar))
```

```
print('Root Mean Squared Error:',rmse1)
```

```
print()
```

```
print('RANDOM FOREST REGRESSOR ERRORS')
```

```
print('Mean Absolute Error: %.3f % mae)
```

```
print('Mean Squared Error: %.3f % mse)
```

```
print('Mean Absolute Percentage Error: %.3f % mape)
```

```
rmser=sqrt(mse)
```

```
print('Root Mean Squared Error:',rmser)
```

```
input("Press enter to exit ;)")
```

APPENDIX 2: Python script code for the comparison of the algorithm errors applicable in many timeseries with the for loop

```
import pandas as pd

from statsmodels.tsa.holtwinters import ExponentialSmoothing as HWES
from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing, Holt
from sklearn.metrics import mean_squared_error
from math import sqrt

from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_absolute_percentage_error
import

from tkinter.filedialog import askopenfilename

from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.arima_model import ARIMA
from sklearn.ensemble import RandomForestRegressor
from numpy import asarray
import numpy
import warnings
import xlswriter

# Ignore harmless warnings
warnings.filterwarnings("ignore")

# Workbook is created
wb = xlswriter.Workbook ('ALGORITHMS COMPARISON1111.xlsx')

# add_sheet is used to create sheets and name them.
```

```

sheet1 = wb.add_worksheet('ERRORS HOLT WINTERS')
sheet2 = wb.add_worksheet('ERRORS SARIMAX')
sheet3 = wb.add_worksheet('ERRORS ARIMA')
sheet4 = wb.add_worksheet('ERRORS RANDOM FOREST')

```

#we write in each sheet the error names in first column

```

ws = wb.get_worksheet_by_name('ERRORS HOLT WINTERS')
ws.write(1, 0, 'Mean Absolute Error')
ws.write(2, 0, 'Mean Squared Error')
ws.write(3, 0, 'Mean Absolute Percentage Error')
ws.write(4, 0, 'Root Mean Squared Error')

ws = wb.get_worksheet_by_name('ERRORS SARIMAX')
ws.write(1, 0, 'Mean Absolute Error')
ws.write(2, 0, 'Mean Squared Error')
ws.write(3, 0, 'Mean Absolute Percentage Error')
ws.write(4, 0, 'Root Mean Squared Error')

ws = wb.get_worksheet_by_name('ERRORS ARIMA')
ws.write(1, 0, 'Mean Absolute Error')
ws.write(2, 0, 'Mean Squared Error')
ws.write(3, 0, 'Mean Absolute Percentage Error')
ws.write(4, 0, 'Root Mean Squared Error')

ws = wb.get_worksheet_by_name('ERRORS RANDOM FOREST')
ws.write(1, 0, 'Mean Absolute Error')
ws.write(2, 0, 'Mean Squared Error')
ws.write(3, 0, 'Mean Absolute Percentage Error')
ws.write(4, 0, 'Root Mean Squared Error')

```

```

#open file
myfile = askopenfilename()

#import of the range values for the loop
aa = input("Enter value for the start column of the file (if 1st is the index (dates) type 0): ")
aa = int(aa)
bb = input("Enter value for the last column of the file: ")
bb = int(bb)

#RANDOM FOREST
print()

#we will ask user to type the n estimators for random forest regressor
estims = input("Enter value for estimators of random forest regressor: ")
estim = int(estims)

#now in this for loop we execute our main code in order to calculate and put in xls file all the errors
for f in range(aa , bb):

    keep_rows = f+1

    #read the data file. the date column must be in the mm-dd-yyyy format.

    df = pd.read_csv(myfile, header=0, infer_datetime_format=True, parse_dates=[0],
index_col=[0],delimiter=';', usecols=[0, keep_rows])

    df.head()

    #split the dataset into training and the test groups. The last 12 periods form the test data
    df_train = df.iloc[:-12]
    df_test = df.iloc[-12:]

    # transform a time series dataset into a supervised learning dataset
    def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
        n_vars = 1 if type(data) is list else data.shape[1]
        dfs = pd.DataFrame(data)

```

```

cols = list()

# input sequence (t-n, ... t-1)
for i in range(n_in, 0, -1):
    cols.append(dfs.shift(i))

# forecast the sequence (t, t+1, ... t+n)
for i in range(0, n_out):
    cols.append(dfs.shift(-i))

# put it all together
agg = pd.concat(cols, axis=1)

# drop empty rows without values (NaN)
if dropnan:
    agg.dropna(inplace=True)

return agg.values


# split a univariate dataset into train and test group
def train_test_split(data, n_test):
    return data[:-n_test, :], data[-n_test:, :]


# implement a random forest regressor model and make a forecast one step ahead
def random_forest_forecast(train, testX):
    # transform list into array
    train = asarray(train)

    # split data into input and output columns
    trainX, trainy = train[:, :-1], train[:, -1]

    # fit the model
    model = RandomForestRegressor(n_estimators=estim, n_jobs=2)
    model.fit(trainX, trainy)

    # forecasting one step ahead

```

```

    yhat = model.predict([testX])
    return yhat[0]

# walk-forward validation for the univariate data
def walk_forward_validation(data, n_test):
    predictions = list()
    # split the dataset
    train, test = train_test_split(data, n_test)
    # feed the history with the training dataset
    history = [x for x in train]
    # step ahead each step in the test data group
    for i in range(len(test)):
        # split the test row into input and output columns
        testX, testy = test[i, :-1], test[i, -1]
        # fit the model on history and make a forecast
        yhat = random_forest_forecast(history, testX)
        # add the forecast in list of predictions
        predictions.append(yhat)
        # add real observation to history array in order to implement the next loop
        history.append(test[i])

    # estimate the errors of the model
    error = mean_absolute_error(test[:, -1], predictions)
    error1 = mean_squared_error(test[:, -1], predictions)
    error2 = mean_absolute_percentage_error(test[:, -1], predictions)
    return error, error1, error2, test[:, -1], predictions

# load the dataset

```

```

values = df.values

# transform the timeseries data into supervised learning data
data = series_to_supervised(values, n_in=6)

# evaluation based on the errors
mae, mse, mape, y, yhat = walk_forward_validation(data, 12)

print('OK')

# ARIMA Model

# evaluate an ARIMA model for a given order (p,d,q)
def evaluate_arima_model(X, arima_order):
    # prepare training dataset
    train_size = int(len(X) * 0.66)
    train, test = X[0:train_size], X[train_size:]
    history = [x for x in train]
    # make predictions
    predictions = list()
    for t in range(len(test)):
        model = ARIMA(history, order=arima_order).fit(optimized=True,
use_brute=True)
        yhat = model.forecast()[0]
        predictions.append(yhat)
        history.append(test[t])
    # calculation of the sample error
    rmse = sqrt(mean_squared_error(test, predictions))
    return rmse

# evaluate combinations of p, d and q values for the ARIMA model
def evaluate_models(dataset, p_values, d_values, q_values):
    dataset = dataset.astype('float32')

```



```

best_score, best_cfg = float("inf"), None
for p in p_values:
    for d in d_values:
        for q in q_values:
            order = (p,d,q)
            try:
                rmse = evaluate_arima_model(dataset, order)
                if rmse < best_score:
                    best_score, best_cfg = rmse, order
            except:
                continue

    return best_cfg

#dataset
series = df

# evaluate parameters
p_values = [0, 1, 2]
d_values = range(0, 3)
q_values = range(0, 3)
ord1=evaluate_models(series.values, p_values, d_values, q_values)

# spare the data into train and test sets
X = df.values
X = X.astype('float32')
size = int(len(X) * 0.8)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictionsar = list()
# walk-forward validation for the best order calculated

```

```

for t in range(len(test)):

    modelar = ARIMA(history, order=(ord1)).fit(optimized=True, use_brute=True)

    output = modelar.forecast()

    yhat = output[0]

    predictionsar.append(yhat)

    obs = test[t]

    history.append(obs)

print('OK')

#SARIMAX model

#runing SARIMAX model with the same order as ARIMA and a fixed seasonal order

model1 = SARIMAX(df_train, order = (ord1), seasonal_order =
(1,2,1,12)).fit(optimized=True, use_brute=True)

#define start and end length for the forecast

start = len(df_train)

end = len(df_train) + len(df_test) - 1

# Predictions for one-year against the test set

predictions = model1.predict(start, end, typ = 'levels').rename("Predictions")

print('OK')

#checking for zeros inside the dataset

sum0 = 0

for i in range(len(df)):

    if df.values[i] == 0:

        sum0 = sum0 + 1

#if sum of zero values is zero we run multiplicative holt winters' model

if sum0 == 0:

```

```

    #holt winters' build and train the model on the training data

    model = HWES(df_train, seasonal_periods=12, trend='add', seasonal='mul',
initialization_method="estimated").fit(optimized=True, use_brute=True)

    #if sum of zero values is not zero we run additive holt winters' model
    else:

        #holt winters' build and train the model on the training data

        model = HWES(df_train, seasonal_periods=12, trend='add', seasonal='add',
initialization_method="estimated").fit(optimized=True, use_brute=True)

    #create an out of sample forecast for the next 12 steps beyond the final data point in the training
data set

    sales_forecast = model.forecast(steps=12)


    #print all errors for each algorithm
    print('HOLT WINTERS ERRORS')
    print('Mean Absolute Error:',mean_absolute_error(df_test, sales_forecast))
    print('Mean Squared Error:',mean_squared_error(df_test, sales_forecast))
    print('Mean Absolute Percentage Error:',mean_absolute_percentage_error(df_test,
sales_forecast))
    rmse=sqrt(mean_squared_error(df_test, sales_forecast))
    print('Root Mean Squared Error:',rmse)
    print()

    print('SARIMAX ERRORS')
    print('Mean Absolute Error:',mean_absolute_error(df_test,predictions))
    print('Mean Squared Error:',mean_squared_error(df_test,predictions))
    print('Mean Absolute Percentage Error:',mean_absolute_percentage_error(df_test,predictions))
    rmse2=sqrt(mean_squared_error(df_test,predictions))

```

```

print('Root Mean Squared Error:',rmse2)
print()

print('ARIMA ERRORS')
print('Mean Absolute Error:',mean_absolute_error(test, predictionsar))
print('Mean Squared Error:',mean_squared_error(test, predictionsar))
print('Mean Absolute Percentage Error:',mean_absolute_percentage_error(test,
predictionsar))

rmse1=sqrt(mean_squared_error(test, predictionsar))
print('Root Mean Squared Error:',rmse1)
print()

print('RANDOM FOREST REGRESSOR ERRORS')
print('Mean Absolute Error: %.3f % mae)
print('Mean Squared Error: %.3f % mse)
print('Mean Absolute Percentage Error: %.3f % mape)
rmser=sqrt(mse)
print('Root Mean Squared Error:',rmser)

#write te product number in prod value
for col in df.columns:
    prod=col
    print(prod)

#print product number and errors in the xls file sheets
ws = wb.get_worksheet_by_name('ERRORS RANDOM FOREST')
ws.write(0, keep_rows, prod)
ws.write(1, keep_rows, mae)
ws.write(2, keep_rows, mse)

```

```
ws.write(3, keep_rows, mape)
```

```
ws.write(4, keep_rows, rmser)
```

```
ws = wb.get_worksheet_by_name('ERRORS ARIMA')
```

```
ws.write(0, keep_rows, prod)
```

```
ws.write(1, keep_rows, mean_absolute_error(test, predictionsar))
```

```
ws.write(2, keep_rows, mean_squared_error(test, predictionsar))
```

```
ws.write(3, keep_rows, mean_absolute_percentage_error(test, predictionsar))
```

```
ws.write(4, keep_rows, rmse1)
```

```
ws = wb.get_worksheet_by_name('ERRORS SARIMAX')
```

```
ws.write(0, keep_rows, prod)
```

```
ws.write(1, keep_rows, mean_absolute_error(df_test,predictions) )
```

```
ws.write(2, keep_rows, mean_squared_error(df_test,predictions))
```

```
ws.write(3, keep_rows, mean_absolute_percentage_error(df_test,predictions))
```

```
ws.write(4, keep_rows, rmse2)
```

```
ws = wb.get_worksheet_by_name('ERRORS HOLT WINTERS')
```

```
ws.write(0, keep_rows, prod)
```

```
ws.write(1, keep_rows, mean_absolute_error(df_test, sales_forecast) )
```

```
ws.write(2, keep_rows, mean_squared_error(df_test, sales_forecast))
```

```
ws.write(3, keep_rows, mean_absolute_percentage_error(df_test, sales_forecast))
```

```
ws.write(4, keep_rows, rmse)
```

```
#save the file and exit the script
```

```
wb.close()
```

```
input("Press enter to exit ;)")
```

APPENDIX 3: Split the data in sheets based on the Product_Code

```
Sub SplitIntoSheets()
```

```
With Application
```

```
    .ScreenUpdating = False
```

```
    .DisplayAlerts = False
```

```
End With
```

```
    ThisWorkbook.Activate
```

```
    Sheet1.Activate
```

```
'clearing filter if any
```

```
    On Error Resume Next
```

```
    Sheet1.ShowAllData
```

```
    On Error GoTo 0
```

```
    Dim lsrClm As Long
```

```
    Dim lstRow As Long
```

```
'counting last used row
```

```
    lstRow = Cells(Rows.Count, 1).End(xlUp).Row
```

```
    Dim uniques As Range
```

```
    Dim clm As String, clmNo As Long
```

```
    On Error GoTo handler
```

```
    clm = Application.InputBox("From which column you want create files" & vbCrLf & "E.g. A,B,C,AB,ZA  
etc.")
```

```
    clmNo = Range(clm & "1").Column
```

```
    Set uniques = Range(clm & "2:" & clm & lstRow)
```

```
'Calling Remove Duplicates to Get Unique Names
```

```
    Set uniques = RemoveDuplicates(uniques)
```

```
    Call CreateSheets(uniques, clmNo)
```

```
With Application
```

```
    .ScreenUpdating = True
```

```
    .DisplayAlerts = True
```

```

        .AlertBeforeOverwriting = True

        .Calculation = xlCalculationAutomatic

    End With

    Sheet1.Activate

    MsgBox "Well Done!"

    Exit Sub

    Data.ShowAllData

handler:

    With Application

        .ScreenUpdating = True

        .DisplayAlerts = True

        .AlertBeforeOverwriting = True

        .Calculation = xlCalculationAutomatic

    End With

End Sub

Function RemoveDuplicates(uniques As Range) As Range

ThisWorkbook.Activate

    Sheets.Add

    On Error Resume Next

    ActiveSheet.Name = "uniques"

    Sheets("uniques").Activate

    On Error GoTo 0

    uniques.Copy

    Cells(2, 1).Activate

    ActiveCell.PasteSpecial xlPasteValues

    Range("A1").Value = "uniques"

    Dim lstRow As Long

    lstRow = Cells(Rows.Count, 1).End(xlUp).Row

    Range("A2:A" & lstRow).Select

    ActiveSheet.Range(Selection.Address).RemoveDuplicates Columns:=1, Header:=xlNo

    lstRow = Cells(Rows.Count, 1).End(xlUp).Row

    Set RemoveDuplicates = Range("A2:A" & lstRow)

```

```

End Function

Sub CreateSheets(uniques As Range, clmNo As Long)

    Dim lstClm As Long

    Dim lstRow As Long

    For Each unique In uniques

        Sheet1.Activate

        lstRow = Cells(Rows.Count, 1).End(xlUp).Row

        lstClm = Cells(1, Columns.Count).End(xlToLeft).Column

        Dim dataSet As Range

        Set dataSet = Range(Cells(1, 1), Cells(lstRow, lstClm))

        dataSet.AutoFilter field:=clmNo, Criteria1:=unique.Value

        lstRow = Cells(Rows.Count, 1).End(xlUp).Row

        lstClm = Cells(1, Columns.Count).End(xlToLeft).Column

        Debug.Print lstRow; lstClm

        Set dataSet = Range(Cells(1, 1), Cells(lstRow, lstClm))

        dataSet.Copy

        Sheets.Add

        ActiveSheet.Name = unique.Value2

        ActiveCell.PasteSpecial xlPasteAll

    Next unique

End Sub

```

APPENDIX 4: Put all the sheets in ascending order

```

' Prompt the user as which direction they wish to
' sort the worksheets.

iAnswer = MsgBox("Sort Sheets in Ascending Order?" & Chr(10) _
    & "Clicking No will sort in Descending Order", _
    vbYesNoCancel + vbQuestion + vbDefaultButton1, "Sort Worksheets")

For i = 1 To Sheets.Count

    For j = 1 To Sheets.Count - 1

        ' If the answer is Yes, then sort in ascending order.

        If iAnswer = vbYes Then

            If UCase$(Sheets(j).Name) > UCase$(Sheets(j + 1).Name) Then

```



```

        Sheets(j).Move After:=Sheets(j + 1)
    End If

    ' If the answer is No, then sort in descending order.
    ElseIf iAnswer = vbNo Then
        If UCase$(Sheets(j).Name) < UCase$(Sheets(j + 1).Name) Then
            Sheets(j).Move After:=Sheets(j + 1)
        End If
    End If

End If

Next j

Next i

End Sub

```

APPENDIX 5: Copy all demand rows side by side in one new combined sheet

```

Sub AMS()

    Dim sh As Worksheet
    Dim DestSh As Worksheet
    Dim LR As Long
    Dim CopyRng As Range
    Dim lColumn As Long
    With Application
        .ScreenUpdating = False
        .EnableEvents = False
    End With
    ' Delete the summary sheet if it exists.
    Application.DisplayAlerts = False
    On Error Resume Next
    ActiveWorkbook.Worksheets("MergeSheet").Delete
    On Error GoTo 0
    Application.DisplayAlerts = True
    ' Add a new summary worksheet.
    Set DestSh = ActiveWorkbook.Worksheets.Add
    DestSh.Name = "MergeSheet"
    ' Loop through all worksheets and copy the data to the summary worksheet.'

```

```

For Each sh In ActiveWorkbook.Worksheets
    If sh.Name <> DestSh.Name Then
        ' Find the last column with data on the summary worksheet.
        'Last = LastColumn(DestSh)

        lColumn = DestSh.Cells(1, Columns.Count).End(xlToLeft).Column
        LR = sh.Range("B" & Rows.Count).End(xlUp).Row
        ' Specify the range to place the data'
        Set CopyRng = sh.Range("B1:C" & LR)
        ' This statement copies values and formats from each worksheet.'
        CopyRng.Copy
        With DestSh.Cells(1, lColumn + 1)
            .PasteSpecial xlPasteValues
            .PasteSpecial xlPasteFormats
            Application.CutCopyMode = False
        End With
    End If
Next
ExitTheSub:
    Application.Goto DestSh.Cells(1)
    ' AutoFit the column width in the summary sheet.
    DestSh.Columns.AutoFit
    With Application
        .ScreenUpdating = True
        .EnableEvents = True
    End With
End Sub

```

APPENDIX 6: Erase the columns

```

Sub DeleteCols()
    For c = Cells(63, Columns.Count).End(xlToLeft).Column To 1 Step -1
        If Cells(63, c) >= 13 Then Columns(c).Delete
    Next c
End Sub

```