Supply chain optimization using machine learning methods.

A manufacturing case study.

Manasas Vasileios

SID: 33081170010

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

December 2019, Thessaloniki - Greece

Supply chain optimization using machine learning methods.

A manufacturing case study.

Manasas Vasileios

SID: 33081170010

Supervisor: Prof. Konstantinos Diamantaras

Supervising Committee Members: -

-

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

Abstract

This dissertation was written as a part of the MSc in Data Science at the International Hellenic University. The aim of the thesis, is to present ways that machine learning methods can facilitate in supply chain management, specifically as regards lead time prediction and reduction. Since lead time has been acknowledged as a key factor in supply chain planning and in customer's satisfaction as well, it has been seen fit to research this topic in depth. These methods were applied to a big multi- product and multi-stage aluminum manufacturing group of companies headquartered in Greece. In detail two predictive models were examined to predict the lead time of the company's major product groups, architectural aluminum profiles and accessories, and one model for the demand forecasting of aluminum accessories to prevent stock outs which heavily affect customer's orders lead time. The results of this case study were more than satisfactory, having outperformed the performance of existing systems concerning lead time prediction and demand forecasting.

For the completion of this research, I would like to sincerely thank my supervisor, Konstantinos Diamantaras, for his constructive advice, as well as the company's staff for their valuable help in collecting the material and business knowledge used in this thesis.

Manasas Vasileios
December 2019.

Table of Contents

1 Introduction

Order lead time is the time period between the customer's order placement and the receiving of the order. For years, organizations and businesses of all sectors are trying to add business value by minimizing the lead time, delivering products at the shortest possible time and by providing an accurate estimation of the order delivery date at the placement of the order. While the reduction of LT (Lead Time) through demand forecasting and the prediction of its value is a key factor in production programming and scheduling, it has proven to be a deciding factor for customer satisfaction too. The complexity of the in between processes, the huge range of products and the variability that external factors bring in, is making the modeling of this matter extremely complicated. In this thesis we are attempting to minimize the lead time of orders placed by customers of Alumil, a big manufacturing industry in Greece, by forecasting the demand for inventory control, thus preventing stock-outs which can cause serious delays in the delivery of the products. Also, it is attempted to build a software that dynamically predicts each order's delivery date based on historical data.

Case Study – Alumil SA

ALUMIL S.A. is a big manufacturing group of companies that designs, produces and distributes aluminum architectural systems. Alumil's data containing customer's orders, production routings in a granular level was utilized for this thesis. As the main webpage of ALUMIL SA states: [1] "The company is considered as one of the largest industries in the extrusion of aluminum in Greece and southern Europe. Specifically, the company is specialized in the research, development and production of aluminum architectural systems. The company is headquartered in the Industrial Area of Stavrochori Kilkis, Greece and has more than 2,000 employees. ALUMIL S.A. operates worldwide with over 30 subsidiaries

ALUMIL, has its own production plants in Greece, Serbia, Bosnia and Albania where the demands of its customers for aluminum profiles are being met. Specifically, the facilities that the company is equipped with are the following:

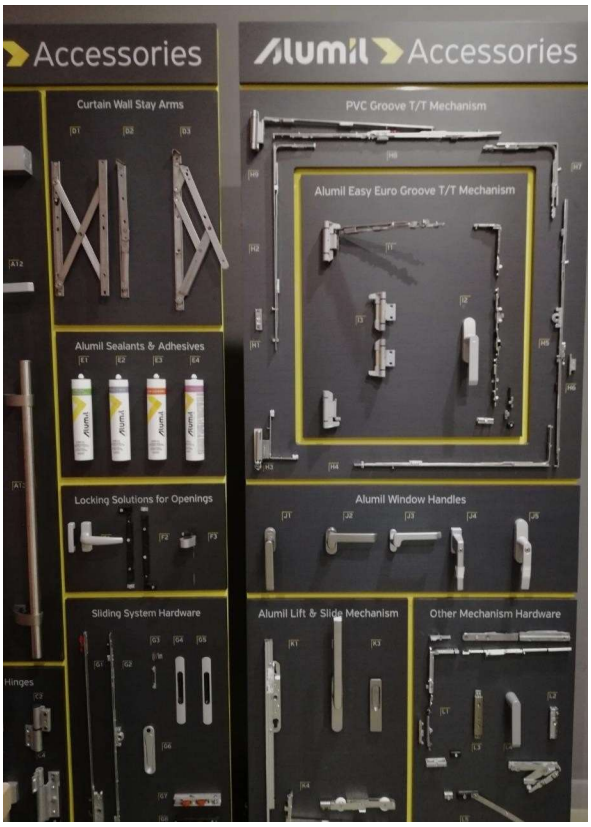
- 11 aluminum extrusion lines with a capacity of 100.000 tons per year,
- 8 powder coating lines (7 horizontal, 1 vertical).
- 3 sublimation lines for wood imitation and special effect colors.
- 3 anodizing plants.
- 2 foundry for aluminum billet production with a capacity of 80.000 tons per year,
- 8 thermal break assembly lines.

- 1 production site for various prefabricated entrance & interior doors
- 1 manufacturing plant for roll-formed aluminum-foam filled profiles.
- 1 production site for elevators and automated systems through METRON, an autonomous subsidiary specialized in automation.
- 3 manufacturing plants for production, processing and assembling of accessories.
- 1 manufacturing plant for aluminum composite panels with a capacity of 950.000 m2.
- 1 manufacturing plant for polycarbonate sheets".

1.1 Problem statement

In this thesis, is presented ways that machine learning can support the supply chain of a big industry, in this case ALUMIL S.A. in two ways;

First by providing an accurate demand forecast for aluminum accessories used in architectural



Picture 1: Picture of aluminum accessories taken at the company's headquarters

and industrial aluminum systems and by estimating the lead time between the order and the delivery date in architectural profiles and accessories.

While aluminum profiles are being produced by the production plants inside the company, accessories that are part of aluminum architectural systems, are for the most part, ordered by external suppliers. The reason these two supply chain aspects were chosen for this thesis, is the

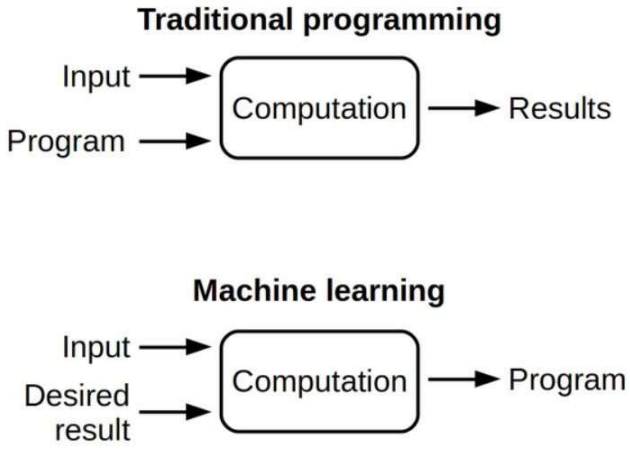
challenges that the company has to face regarding the timely order of aluminum accessories from suppliers and the accurate prediction of the lead time between customer's order date and order delivery date. In detail, customer's demand for aluminum accessories has to be predicted in order to optimize supply decisions. Stock outs are a common phenomenon that is observed, due to unexpected demand. Consequently, the order lead time, of an unpredicted demand in accessories is rising. As for the lead time prediction, high accuracy can lead to customer satisfaction.

2 Background and preliminaries

This thesis is going to address some of the issues of supply chain optimization, modeling the given problems with machine learning. In this chapter, background theory about machine learning is going to be laid out. Specifically, since the problems that are going to be modeled in this thesis have numerical targets, meaning that the algorithms which are applied attempt to predict numerical values, regression techniques are below presented.

2.1 Machine Learning

Machine learning is the study of statistically modeling a problem to perform a task without specifically setting the rules and instructions to do so. It's a counter intuitive set of techniques which changes the order of tasks in comparison to traditional programming. In traditional programming a well-defined set of rules are applied to existing data in order to calculate the desired outcome. In machine learning programming pre-known data and desired outcome are modeled in a way that a previously unknown set of rules is discovered. This relatively new approach to problems can be found extremely useful in business in general but especially in the supply chain management. The number of hidden and volatile factors that interrelates and correlates with the desired outcome in a complex ecosystem like the supply chain, can make the effort of its modeling with traditional methods, extremely difficult or even impossible.



Picture 2: Traditional Vs Machine learning programming.

Machine learning consists of five steps (on a high level):

1. Data Collection (Feature Selection)
2. Data preparation (Feature Engineering)
3. Model selection and training
4. Model Evaluation
5. Prediction

Supply Chain Analytics

Analytics and predictive modeling can be utilized practically in every stage of the supply chain, to enable efficient and accurate planning in both sales, operations and inventory level. In detail:

- Sourcing: The process of acquiring and purchasing essential goods, like raw materials. Predictive modeling may be used to detect the balance point between supply and demand, to identify the cost factors
- Production: Quality control, optimized scheduling based on inventories and production stage capacities.
- Warehousing: Workload optimization, stock relocation.
- Transportation: Routing optimization and scheduling
- Consumer: Credit scoring, recommender systems, fraud detection.

Machine learning techniques

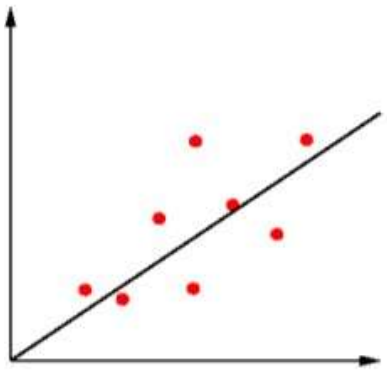
Knowledge data discovery (KDD) is the process of gaining insights and value from the thorough analysis of data. While in medicine and biology, finance and innovation related fields the process of KDD have been applied for several decades now, in the field of production and supply chain planning and control the research interest has not been this vivid, until the past few years. Rainer states that the application of KDD processes in production and supply chain areas in business such as analytic methods had resulted in a return of at least ten times of their investment.

KDD processes (also found in literature as "Data mining") are divided into two big categories, that are, descriptive and predictive analytics. Descriptive analytics focuses on the discovery of patterns, rules, associations in the data, like the name indicates, its aim is to "describe" the data. Some applications are association rule learning, causal discovery techniques and clustering algorithms like KNN (K- Nearest Neighbors) algorithm. Predictive analytics aims at predicting future values of one or many target variables. Depending on the target variable, the predictive analytics can be assorted in continuous or categorical. Examples of continuous predictive analytics is the famous regression technique with its many variants.

Categorical predictive analytics like classification, aims to classify correctly the target variable into two or more categories.

The well-applied KDD techniques in production planning and supply chain management all belong to APS thus far (Advanced planning and scheduling, fault diagnosis, quality improvement and defect analysis). Lead time (else known as flow time) prediction is a field which only in the last decade has been dealt with analytical techniques.

2.1.1 Linear Regression



inputs	outputs
x1 = 1	y1 = 1
x2 = 3	y2 = 2.2
x3 = 2	y3 = 2
x4 = 1.5	y4 = 1.9
x5 = 4	y5 = 3.1

Picture 3: Each data point of feature "x" corresponds to a respective target "y"

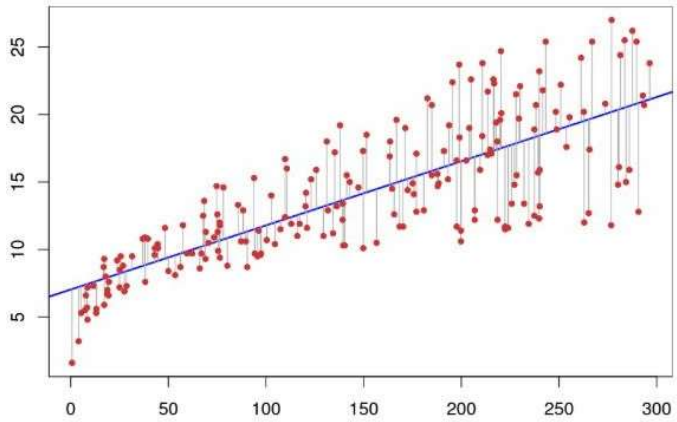
Linear regression, is a machine learning technique for the prediction of a continuous numerical target variable. Its approach is simple, yet it manages to capture the linear relationships (and near linear) between the features applied in the model. It stands as a basis for more advanced techniques, so the understanding of linear regression is fundamental for approaching more complex techniques. [2]

$$Y = \beta_0 + \beta_1 X$$

The equation above captures the problem of the regression problem. The Y variable (dependent variable) is the target continuous variable, the X is the independent variable, with the betas are called the coefficients. Regression is trying to estimate the betas so that the right side of the equation is as close as possible to the actual target. This is done by minimizing the squares of sum of squared errors. The difference between the actual target variable and the predicted one are called residuals and it is captured by the following equation.

$$e_i = y_i - \hat{y}_i$$

The reason why the evaluation metric is the squared error, is that the prediction may be below or above the desired outcome. So the difference between the true and predicted value may be positive or negative. Lest we square the errors, the sum of errors will be decreased, because the positive differences will be canceled out by the negative difference. One additional advantage of squaring the errors is that it penalizes the bigger difference from the desired target values, thus securing a more accurate model.



Picture 4: Linear regression: The straight line that minimizes the sum of squared residuals.

Above in the visual, the red dots represent the true values, the blue line represents the linear model while the grey lines are the difference between the true values and the predicted ones. The blue line the linear model that minimizes the summation of the squared errors. Another issue that has to be addressed is the relevancy of the betas coefficients concerning the target variable. P-values indicate the statistical significance. Every modeling task starts with the hypothesis that there is correlation between the chosen features and the target variable. Therefore the null hypothesis is that there is no correlation between the dependent variable and the independent ones. The metric of p-values for every beta coefficient indicates whether the independent variable (predictor) is statistically significant in relation with the target variable. A value of 0.05 or less is generally considered in the literature as an indication of strong relationship between the dependent variable and the predictor. Lastly, two other KPIs for the linear model are the R^2 statistic and the RSE (Standard error of the residuals).

2.1.2 Multiple Regression

Just like linear regression, multiple regression models attempt to predict a target continuous variable, but this time multiple features are used. In real life one independent variable is not enough to explain the variation of the dependent variable, so two or more variables are added. The equation remains the same as in the simple linear regression, with the addition of the variable and their corresponding betas coefficients. Furthermore, unlike simple regression's p- value for examining the relevancy of the independent variables, in the case of multiple regression the F- statistic metric is measured. The F- statistic formula is displayed by the equation:

$$F = \frac{\frac{TSS - RSS}{p}}{\frac{RSS}{(n - p - 1)}}$$

p: the number of independent variables.

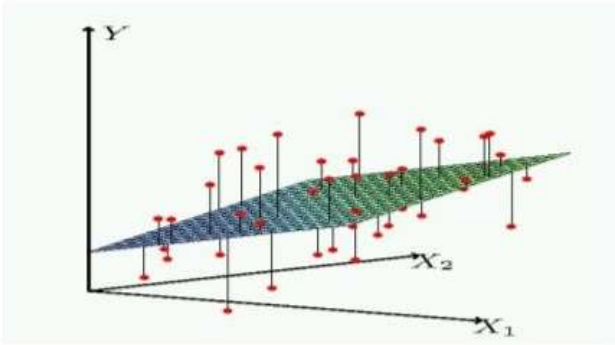
n: is the number of data rows.

TSS: Total Sum of Squares

RSS: Residual Sum of Squares

Unlike p-value, which examines each feature individually, the F-statistic formula assess the whole model. As a rule of thumb, if the relationship is strong enough, the F measure will be bigger than 1, otherwise it will be close to the value of one. (Actually, the value of F-statistic must be assessed in companion with the number of data points, since in big dataset even slightly bigger value than 1indicate strong relationship, while in small ones, the value of the measure

must be significantly bigger than one. In the case of many predictors, the use of p-values is not trustworthy, because unimportant features may be evaluated as significant because of their low p-value. The F-statistic can overcome this obstacle, and this is the reason it is used in the case of multiple regression. The R^2 is again the measure which can be used to evaluate the interpretability of the model. It is a fact that adding more features to the model will lead to an increase of the model but it does not mean necessarily that the performance of the algorithm will increase as well.



Picture 5: In multiple regression the goal is to fit a hyper-plane than minimizes the sum of squared errors.

In multiple regression there critical issues that need to be addressed such as multicollinearity, heteroscedasticity and autocorrelation. The aim of this paper is to be present practical applications of machine learning applications in supply chain's operational management, and it is not seen fit to delve into regression's issues in more detail.

2.1.3 Ensemble Methods for numerical prediction (regression)

Ensemble learning is the technique of combining multiple machine learning algorithms on a problem in order to obtain better accuracy. The three most popular ensemble practices are:

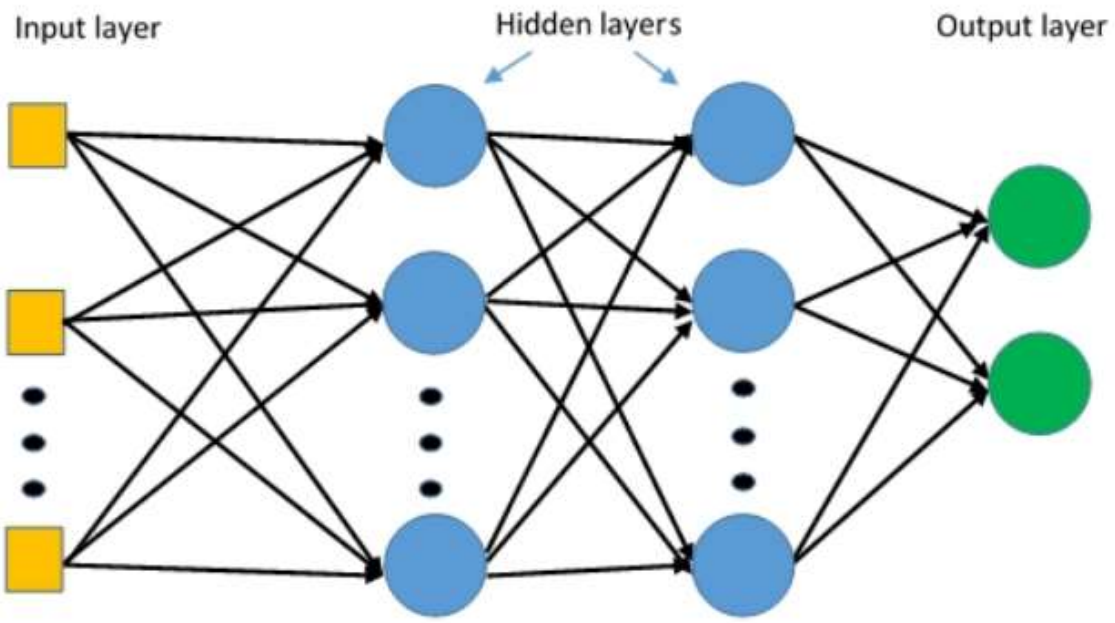
1. Bagging: Using the same training data, use different subsamples to apply multiple models on them.
2. Boosting: Construct many models of the same type, each one taking account the error of the previous model, learning to fix the errors.
3. Voting: Multiple models of different type are applied to the training data and then aggregate the result to combine the predictions (max voting, mean voting)

In more detail:

Bagging, is considered a bootstrap technique which subsamples the training set and applies and trains a regression model on each random distributed sample. Each sub-dataset of

the total N samples, is created by sampling with replacement. Some data points may occur in many of the subsamples, while others may be left out at all. After the training of the model to each subsample, the outcome is averaged and it is presented as the model outcome. Another ensemble approach which uses subsamples of the original training data is the boosting technique. What differentiates boosting from bagging is that each algorithm is aware of the errors that the prior algorithm/ subset of data has produced. Instead of using randomly distributed samples of the data, each subsequent algorithm chooses the data that the previous algorithm has not predicted accurately. After several algorithm steps, all the outcomes of the models are weighted averaged. The weights are related to each model's accuracy, that is, bigger accuracy implies bigger weight. An example of this ensemble technique is the additive regression model. Voting is using a variety of machine learning models to the same training dataset and then it combines the different predictions. Especially in the case of implementing voting in regression, using the averaging technique, it is expected a good performance, based that the majority of the algorithms will be correct when their "opinions" are close to each to other. Another ensemble method is called stacked generalization, also known as stacking. Stacking combines predictions of different machine learning algorithms in levels. Level zero is the original dataset and all the algorithms are applied to this level. Level one is composed by the by the outputs of the base models and so on and so forth.

2.1.4 Artificial Neural Networks



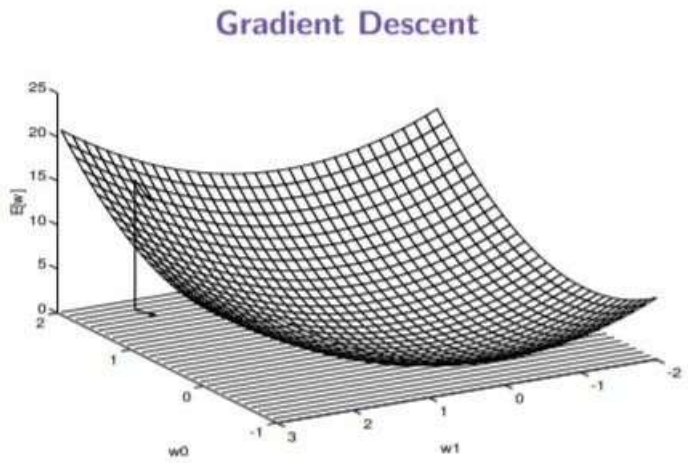
Picture 6: An example of an artificial neural network. [Image downloaded from https://deepai.org/machine-learning-glossary-and-terms/multilayer-perceptron in December 2019]

Artificial neural networks are computing networks which are inspired by the way biological networks are working. In comparison to traditional programming, the neural network learns to perform tasks from the training data without being given explicit rules. Simple components, called neurons perform simple numerical operations, which combined with some predefined thresholds (activation functions) perform decisions. These neurons are ordered into layers. The first layer is the input layer the second one is called the hidden layer and the last one (in a shallow networks) is called the output layer. The number of input layer is equal with the distinct number of features of the training dataset and the output layer is comprised of one neuron for regression tasks (numerical outcome). If the network consists of more than one hidden layer is called a deep neural network. Adding hidden layers leads to more complex network which is able to reproduce complex functions, thus achieving more accurate predictions.

The learning process of the network takes place in the updating of the parameters (weights) of the neurons. It is an iterative process of forward - propagation and back - propagation trying to find the optimal values for the weights of the neurons, so that the prediction is as much closer as possible.

According to the Universal approximation theorem [3], it has been proven that *"a feed-forward network with a single hidden layer containing a finite number of neurons can*

approximate continuous functions on compact subsets of R^n (Euclidean Space)*, under mild assumptions on the activation function".*



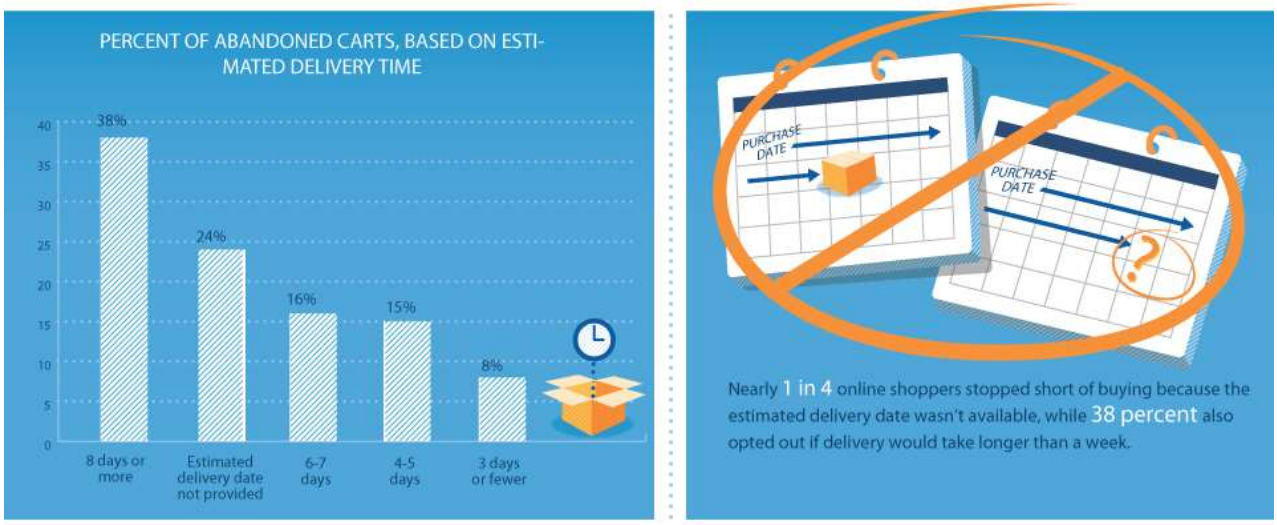
Picture 7: Gradient descent, a core aspect of machine learning.

3 Literature review

3.1 Relevance of reducing and predicting LT

In this chapter the relevance and importance of reducing and predicting Lead Times is going to be examined. This thesis is trying to address the issue of inventory control through the use of demand forecasting.

"The No. 1 thing that has made us successful by far is an obsessive-compulsive focus on the customer as opposed to obsession over the competitor," [4] said Jeff Bezos, CEO, and founder of "Amazon" in a talk at the Economic Club of Washington. Nowadays it is a common knowledge among big industries and companies that customer satisfaction is a key factor, if not -the- key factor, to the growth of an organization. Towards this goal, this consulting project is an effort to improve "Alumil" customer satisfaction by estimating and distributing a realistic order delivery date, utilizing business analytics. Business analytics is defined as "the scientific process of transforming data into insight for better decision making". According to a study carried out by Milo [5] (titled: "No cart left behind"), the bigger the estimated delivery time is, the bigger is the probability that a customer will not complete the order but rather abandon it. While the above study was carried out for the business to customer industry, it remains true for the business to business organizations. This project's aim is to provide an estimated delivery time closest to the real one.



Picture 8: Part of "milos"' infographic about customer's behavior

Muddassir Ahmed, in his article [6]"*Seven reasons why you need to forecast in supply chain*" in 2016, states that demand forecasting plays a critical role in the supply chain function regarding stock outs, especially when the suppliers are delivering the product in long lead times. This type of forecasting can ensure efficiency in two key aspects of the business, namely sales fulfilment time and short inventory times that can lead to reducing warehousing costing in general. Also, having in mind the customer satisfaction factor, Ahmed suggested that demand forecasting can help increase the business' customer's satisfaction by having already available in stock the products that they are ordering in the time of the ordering. In addition to these benefits, Ahmed concluded that manufacturing demands for production, planning of new products by the team of product managers and planning for promotions by the marketing team can all be harmonized together by accurate demand forecasting. This fact could lead to the reduction of safety stock necessities thus reducing the associated costs. In their 2010 released book, "Operations Management", Slack, Chambers and Johnston claimed that accurate demand forecasting aids the business at achieving effective production schedule by maintaining the balance between supply and with SIOP (Sales Inventory and Operations Planning).

3.2 Lead Time prediction

Little's Law:

Queuing theory, a mathematical sub discipline of the theory of probability, states that the average number of customers on a system is equal to the product of their average arrival rate and the average lead time that is the amount of time that the customer spends in the stationary system. This is famously known as [7]"Little's Law".

$$L = \lambda W$$

In the formula above the “L” declare the work in progress, also known as “WIP”. Put simply, it is the number of items/ products/ services inside the queuing system that is being examined. Lambda stands for the average arrival and departure rate and W represents the time that each item must spend on the system, also known as lead time. For years, this simple yet effective formula was utilized in the manufacturing business world with quite remarkable results. The reason for that is that the formula is not influenced by external factors such as service distribution and order or the arrival distribution.

Since the decade of the seventies, considerable amount of researchers have been publishing papers concerning the determination of production lead times. Weeks, 1979 [8], published his research on the impact of the predicted lead time in production based on statistical measures. His three hypotheses that he put to the test were: Lead time rules based on forecasting of individual jobs’ delivery time have a better outcome than rules based on total work, concerning workshop congestion. Rules focused on lead time perform better than the “shortest imminent-processing” rule**.** Lead time tends to be bigger when the workshop system becomes more complicated. Weeks claimed that this research was just a beginning, and the topic of predicting lead times has many unanswered questions and plenty of room for further investigating. In 1991, [9] Vig and Dooley, utilized existing data of completed orders to obtain rules and patterns in them concerning the orders’ lead time. They discovered that existing data can be of use, since they concluded that the characteristics of the placed order and the kind of production affect the lead time and are very important for the forecasting. In 1994, [10] Enns stated that two were the factors of success for job shop (a manufacturing practice in which batches of custom products are made) customers: high supply reliability and short lead times. A methodology that describes the success factors of negotiating the due dates in a production environment, with complex processes and high variety of products where research by Lawrence S.R., 1994 [11]. Production order’s flow time in small and medium-sized enterprises in a dynamic market was the research topic for Wiendahl and Dammann in 2006. They quantified the influences inside this dynamic market and developed a tool that prescripts which response is the best to this influences. He used a model for production and capacity control which was targeted at order’s delivery time. In 1983, [12]Tatsiopoulos I. and Kingsman in their paper “Lead Time Management”, presented their research which dealt with defining manufacturing flow times used in PPC. The main two methods they used included were: A probabilistic approach which treats manufacturing lead time as unpredictable and chaotic. The second approach highlights the importance of control in manufacturing in order to manage the lead time to confront to predefined ranges. In the paper it is concluded, that the second approach is

more reliable, but it requires very close cooperation between the functions of production and commercial inside the organization to achieve that. [13] Kingsman, in his research "A structural methodology for managing manufacturing lead times in make-to-order companies", 1989, proposed a different approach to controlling manufacturing lead times. While in the past, the scheduling of orders in the shop floor were made in order to meet their predefined delivery dates, he proposed that the ordering and scheduling must be defined at a higher level when the customer inquiry arrives and the prices and the delivery dates are determined. The same researcher, in the year of 2000 issued an article in the "International journal of production economics" under the title: [14]"Modelling input–output workload control for dynamic capacity planning in production planning systems". In his research of workload control in PPC, he concluded that in the produce-to-order companies the arrival of orders cannot be predicted, and that managing is a better practice than forecasting lead times. Ooijen and Bertrand in 2001 [15], researched the tradeoff between the reliability of the company and the length of lead times from an economic perspective. They tried to find a cost optimal delivery due date, having in mind lead time and cost related factors.

Following the review of research projects in general there the pattern that has been revealed is that most of the researchers' work is concentrated on the whole lead time, gathering data from simulation processes and applying few machine learning algorithms. In 2006, [16] Ozturk found that because lead time does not include only the processing time in the shop floor, but contains also transportation and queue, it is very a very difficult task to forecast it. In this research "Manufacturing lead time estimation using data mining", the group of researchers attempted to predict lead times using basic machine learning algorithms like regression trees and linear regression. In 2018, Pfeifer A. [17] applied and evaluated the performance of three different machine learning models and presented the outcomes to the 6th CIRP Global Web Conference – Envisaging the future manufacturing, design, technologies and systems in innovation era. Data were extracted by simulation processes (discrete event) with 8 features. The random forest machine learning outperformed the other two models. Gyulai D. in the 16th IFAC Symposium on Information Control Problems in Manufacturing (2018, Bergamo, Italy) [18] presented a paper which contacted machine learning and analytical techniques and methods to a manufacturing execution system, which due to changing order stream is exposed to changes and is overall complex. [19] Meidan in his paper stated that the research should not focus on lead times in general but rather in waiting times between different stages in the manufacturing flow shop environment. The team of researchers applied neural networks, Bayesian classifiers and decision trees. After excessive feature engineering they concluded that only the 11% of original's dataset features were enough to perform the forecast. [20] Alenezi in 2008, compared the performance of an SVM (support vector machine) model with an ANN (artificial neural network) and time series models for flow time prediction in real time problem. Another

comparison between different machine learning models concerning lead time prediction was contacted by Mori in 2015 [21] . He estimated lead time in production using Bayesian networks and compared its results with SVM and ANN models in the steel industry. An example of classification model was conducted by De Cos Juez [22] . Specifically he built a machine learning algorithm which classifies a batch of metallic components of aerospace engines into two categories: completed in the forecasted time or not. Susanto [23] , Raaymakers are two other researchers who utilized ANN to estimate accurately the lead time of customer's orders. To our knowledge this is the first attempt to predict customer's order lead time in the aluminum manufacturing industry with machine learning and non-simulated data.

3.3 Demand forecasting

Most of the research conducted concerning demand forecasting contained traditional time series analysis methods such as Autoregressive Integrated Moving Average (ARIMA) or exponential smoothing method for decades. In last years of the decade of 1990 it is observed that along traditional time series forecasting methods, various machine learning methods were implemented, starting from the simplest ones such as linear/ multiple regression and getting to the more complex ones like ensemble methods, SVM artificial and neural networks.

The ARIMA technique was used in a variety of demand forecasting studies. Williams B. in the year of 2003 [24] , applied the forecasting technique of ARIMA in the vehicular traffic flow. Non-stationary time series data, like international tourism was the object of forecast of Lim in 2002 [25] . She used Box Jenkins ARIMA and evaluated her forecasting accuracy by MAPE (Mean Absolute Percentage Error) and RMSE (Root Mean Squared Error). In 2003, [26] Ching-Wu Chu compared linear methods such as ARIMA with nonlinear models like ANN. He concluded that ANN were performing better with de-seasonilized data while ARIMA was performing better with the original data. Mupparaju K. in 2018 [27] performed a comparative study of various machine learning models for demand forecasting of grocery items. The main comparison was between LGBM (Light Gradient Boosting) and ANN. While ANN are very reliable and efficient with high dimensional data, they are not accurate if the training data is not large enough. Furthermore there is always an interpretability – accuracy trade off. ANN are known as "black boxes" meaning that even if the accuracy of the model is high there is no way of interpreting the causal relationships and the calculations that took place inside the model and how eventually the model arrived in the given result. Two similar approaches of neural networks to time series data, recurrent backpropagation and Long Short Term Memory (LSTM) model were utilized by Goyal A., who observed that LSTM Neural Network outperformed other baseline models, it is fairly simple to implement and it does not require much feature engineering. LSTM models are storing information for extended time interval, truncating the gradient where the model sees fit. While Graves in 2013 [28] used LSTM to forecast next

sequence of text, Mupparaju K. in 2018 [27] , implemented LSTM for sales forecasting. Liu Yue in 2007 [29] performed demand forecasting with a variant of SVM model, which he called "SHEnSVM" (Selective and Heterogeneous Ensemble of Support Vector machines) in which each SVM is trained with different samples of data which is generated by a bootstrap algorithm. He then applied this model to forecast the demand of beer in a retail company with good generalization ability. He then concluded that forecasting accuracy can be increased by using ensemble- learning techniques. I. Watanabe and his research team published an "AI based demand forecasting" [30] method in the Fujitsu scientific & technical journal. The team combined conventional time series modeling with machine learning. They also proposed an attribute decomposition method for the forecasting of new products which have no previous sales. Last, they concluded that the fact that the optimum forecasting method differs depending to the life cycle of the product is a major challenge.

4 Data Engineering & technologies involved

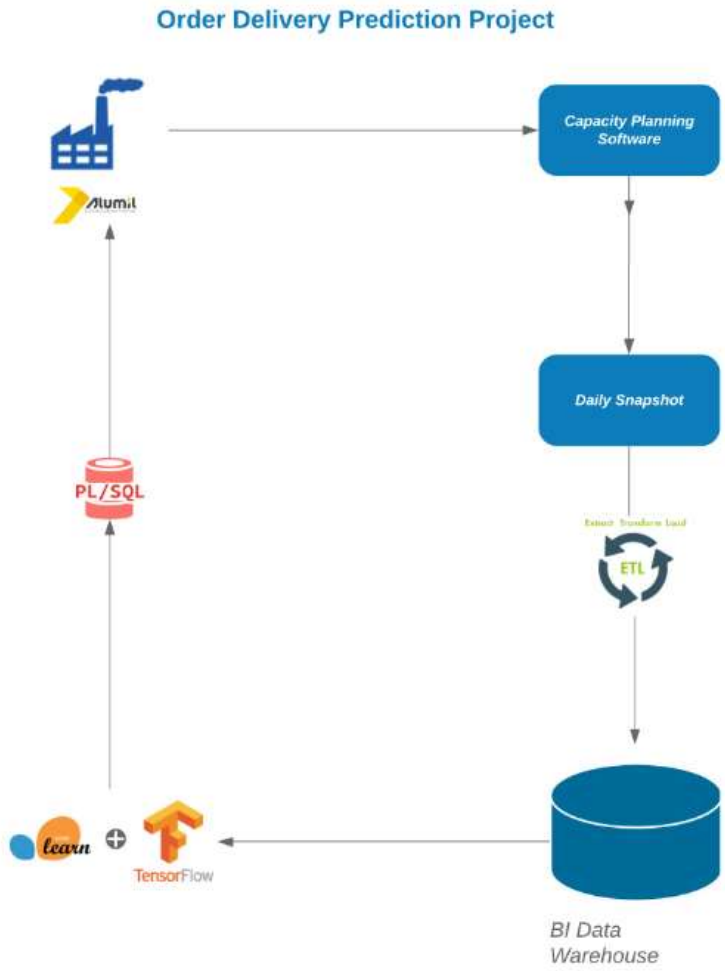
Technologies Involved in this thesis

All raw, transactional data reside inside Oracle's relational databases. From there, using SQL and PLSQL data are integrated into a unique data repository (data warehouse). Data exploration is performed using Microsoft's business intelligence platform PowerBI which live connects to the data warehouse. Lastly, data from the data warehouse are analyzed and used in analysis, feature engineering and predictive modeling with Python's programming language, utilizing its most well-known libraries and APIs, like Pandas, NumPy, ScikitLearn, Tensorflow and Keras.

Data Engineering

Data engineering is the process of acquiring, wrangling, cleaning and modeling the data from disperse and different sources into a single and unified data repository [31]. The data that are going to be utilized in the predictive modeling is not going to be static, but dynamic. ETL pipelines (Extract – Transform – Load) had been constructed so that the data is being refreshed daily. After that, the data is modeled in a star schema and it resides in a multidimensional data warehouse. The original data, resides in the transactional databases of the application that the organization is using for its everyday needs. Specifically, the main resource of data is the company's ERP. Besides that the company uses a complex B2B software designed for customer to place their orders, explore the products and the pricelist, a CRM software, an Advanced Planning and scheduling software and a production management software.

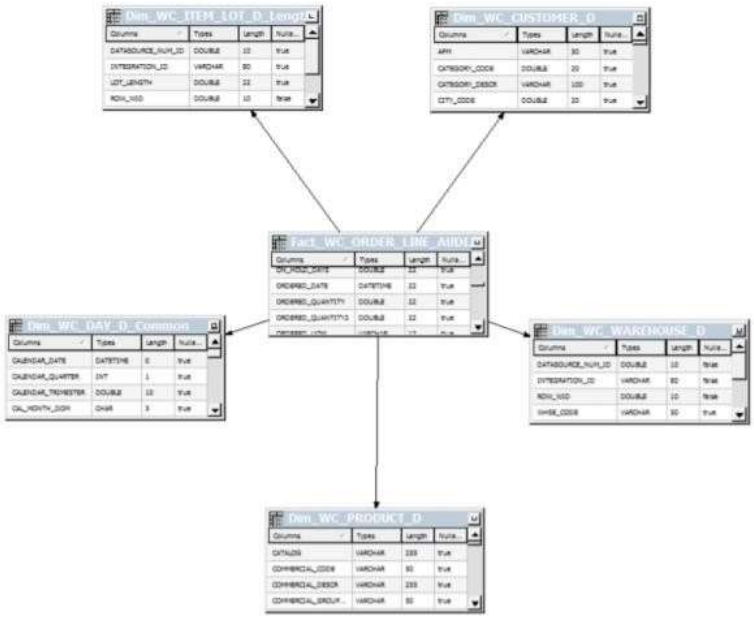
For years, the company made sure that historical data of every day's pending orders are stored in the transactional database of planning's software. This means that an order is present through all its stages, from the first date that it was registered, to the last date, when all the requested kilos of the aluminum profile were ready for loading in the warehouse. About 24 million rows worth of data is stored for reporting reasons (for example, the number of open orders per day, kilos that are pending and must go through various production lines). This huge volume of raw data triggered the initial idea, that it can be utilized in order to gain predictive and prescriptive insights, rather than plain descriptive, using machine learning methods.



Picture 9: The flow of predictive modeling: data from source systems, data warehouse, predictive modeling, database programming and back to business users.

As expected, even simple queries on this huge dataset can be restrictive on a transactional database. The attempt of providing predictive analytics with a machine or worse, deep learning with data that comes from this database was unsuccessful. For this reason, the data needed for

this purpose is daily extracted from the software's database, transformed and then loaded into the data warehouse, which is built for analytical purposes, with the method of ETL (Extract – Transform – Load). The data needs a lot of manipulation, pre-processing and cleansing in order to be used in the correct form by machine learning algorithms. This process was decided to be implemented with PL/SQL and not with Python since PL/SQL engine resides inside the Oracle database (the data warehouse used is Oracle's) and the data manipulation is much faster there. (Analysis inside python takes place inside RAM memory, making analysis infeasible there, due to the volume of data). The historical data of orders reside inside the data warehouse in the well-known architecture of a star schema. That means that a fact table holds the measurable, quantitative data about each order line and dimension tables are linked with the fact table. In each dimension table, there is a primary key relating to one of the columns in the fact table with a one to many relationship.



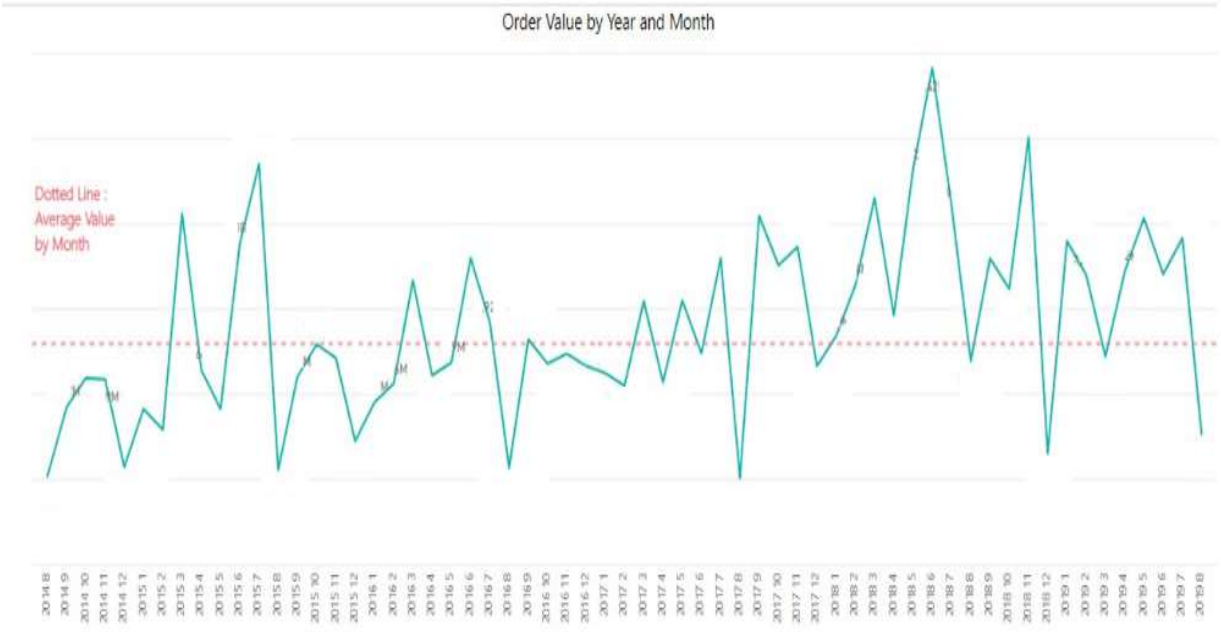
Picture 10: Star schema used in company's data warehouse

The data in the data warehouse is refreshed daily. An automated job is scheduled inside the database to extract the new data from the transactional databases (capacity planning software, ERP system) and insert it into the data warehouse. On top of this star schema structure, a database (materialized) view is constructed, containing all the information in the form that is needed for the machine learning algorithm (a materialized view is selected for performing issues). The application of machine learning modeling is implemented in a Python 3 environment. Access to the Oracle database is given to python with the cx_Oracle version 7.1 module, and specifically to the view that was mentioned before. The data from the view is then inserted into a pandas' data frame for further preprocessing and analysis.

Data Visualization and exploration

Prior business knowledge is essential for the predictive modeling of demand and lead time. In addition to a priori knowledge, data exploration and visualization can lead to discoveries, trends and patterns that may prove of usefulness to the machine learning algorithms later used. For this purpose the business intelligence platform of Microsoft's PowerBI is used.

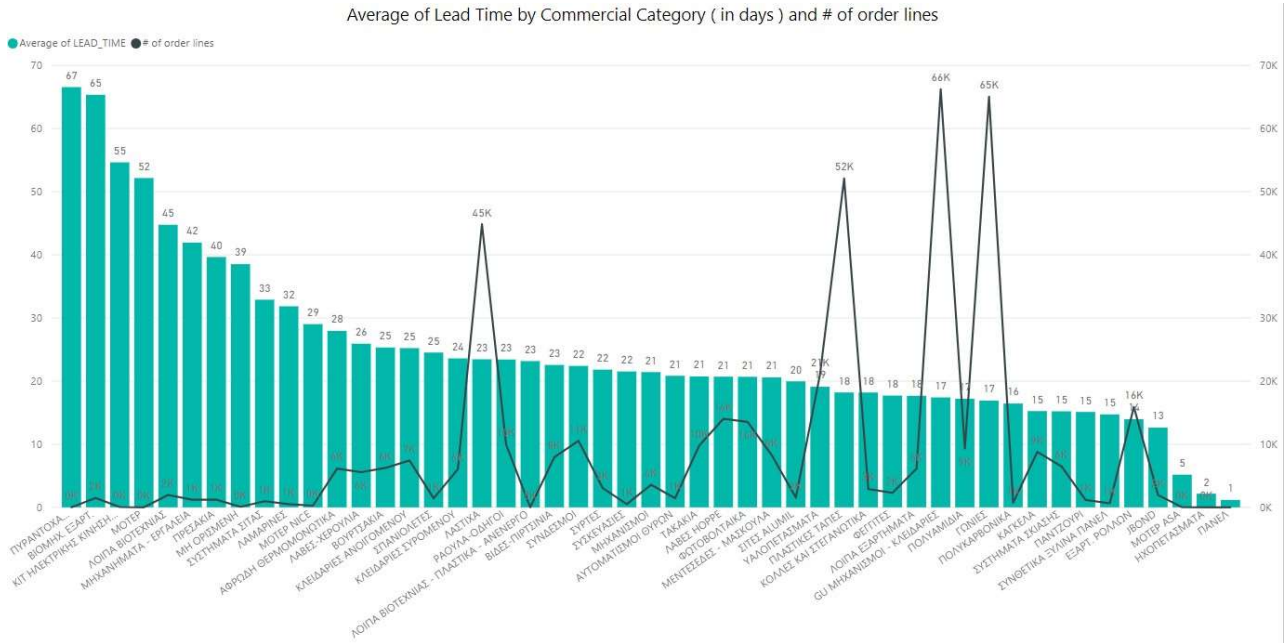
The great usefulness of this platform is that the visualizations that are built with the data are being refreshed daily (or even hourly) in sync with the data in the data warehouse, so that the analyst with a single click can explore daily the new data and find anomalies or new trends previously unknown. Some diagrams exported from the business intelligence platform are displayed below:



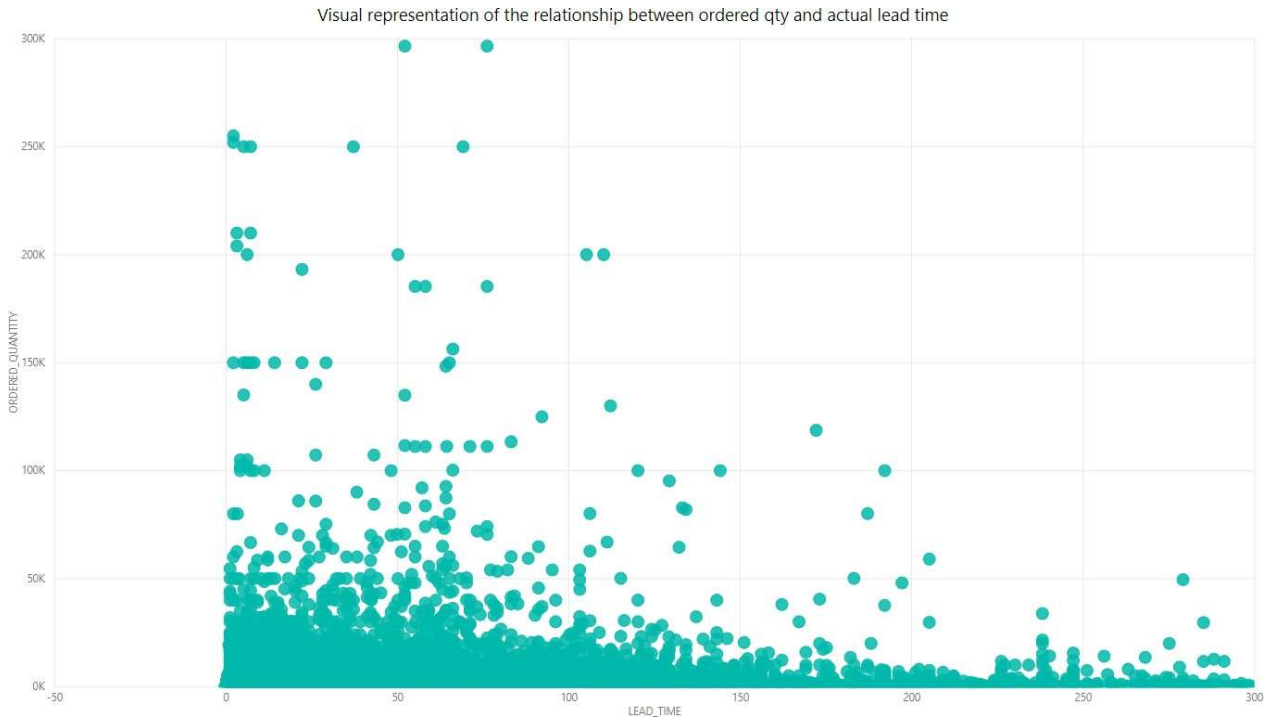
Picture 11: Order value by month. Line chart example from BI Platform

In the line chart above, a clear trend and seasonality is discovered. It is obvious that the value of orders during the August is much less. It is concluded that the month feature will greatly help the algorithm explain the variance of the problem.

Picture 12: Examples of visual representation that helped to understand the problem of lead time prediction (1)



Picture 13: Examples of visual representation that helped to understand the problem of lead time prediction (2)



Picture 14: Examples of visual representation that helped to understand the problem of lead time prediction (3)

5 Feature engineering

Feature engineering is the method of applying domain knowledge of the modeling problem to create features that will make the machine learning model able to predict a desired outcome. Feature engineering is a core aspect of machine learning, and as Andrew Ng states: "Coming up with features is difficult, time-consuming, requires expert knowledge. «Applied machine learning» is basically feature engineering."

There are three predictive objectives that this thesis is dealing with:

1. Lead Time Prediction for customers' orders of aluminum profiles
2. Lead Time Prediction for customers' orders of aluminum accessories
3. Demand forecasting for customers' orders of aluminum accessories.

Combining these three results the supply chain will be in position to perform a "what if" analysis. In detail, utilizing the results of lead time prediction and the demand forecasting the logistics professional will estimate the amount of product loadings for the next month.

The three major categories of data that explains the variability in demand forecasting

1. Static product characteristics
2. Dynamic product characteristics
3. Product's life cycle state
4. External factors (aluminum raw material price, temperature, competing materials price (steel)).

5.1 Lead time prediction (Accessories)

Aluminum accessories are the second most sold (and ordered) item category in the group after aluminum profiles. Its sales account for the 15% of the total revenue that the group produces. Accessories are being grouped into commercial categories and groups. While most of the accessories products (like door handles and hinges) are parts of aluminum systems, some commercial categories, like solar panels, are being sold separately and are not part of the aluminum systems for windows & doors. The table below displays the most common commercial categories that accessories belong to:

Accessories Grouping - Commercial Category	
Adhesives And Sealants	Other Accessories
Aluminum Sheets	Other Industrial
Connectors	Packaging
Corners	Photovoltaics (Pv)
Cremones	Pl Polyamides
Door Automations	Plastic Covers
Electric Motion Kit For Doors	Polyamides
Facade Accessories	Polycarbonates
Fire Rated Accessories / Doors	Premade Pergola
Flyscreen Systems	Pressed Panels
Foam Insulation	Punching Machines
Gaskets	Pvc Profiles
Mechanisms - Locks	Railing Systems
Handles	Roller Shutter Accessories
Hinges	Rollers - Inox Rails
Hoppe Handles	Screws And Rivets
Industrial Accessories	Second Sash Latches

Interno	Shading Systems
Jbond	Shutters Accessories
Levelers / Wedges	Skylights
Locks For Opening Systems	Slats-Axles
Locks For Sliding Systems	Soundwalls
Machines - Tools	Special Steel Systems
Motor	Steel Goods
Motor Asa	T/T Mechanisms
Motors Nice	Wood Composite Panels

A very important distinction which takes place in both the profiles and the accessories is the line of business categorization. Each product can either belong to the architectural or the industrial line of business. While the architectural products are part of window and door, interior and facade systems the industrial products are utilized in the industry (automotive and construction industry, machine construction, ladders scaffoldings facilities). This characteristic is very important especially in the lead time estimation problem, because the industrial products are always "make to order" and are being delivered based on contracts with the customers, while the architectural ones are for the most part "make to stock" since there is a pretty steady order flow from the customers. The majority of the accessories products are being delivered by external suppliers.

Feature selection:

Several meetings with the supply chain stakeholders took place in order to conclude to several features that are considered to affect the accessories' orders lead time. Again, order's details like the date of order placement, product characteristics and most importantly daily data from the accessories' capacity planning software (MRP) were taken into account as features for the predictive modeling. In detail:

Features taken into account:
Order Number,
Ordered Date,
Line Number,
Ordered Quantity,
Item Code,

Line Of Business,
Log Category,
Commercial Code,
Commercial Description
Commercial Group Code,
Commercial Group Description,
Proforma Hold Flag,
Credit Hold Flag,
Mrp Available Stock,
Mrp Production Quantity,
Mrp Reserved Quantity,
Mrp Expected Quantity,
Mrp Remaining Quantity,
Mrp Receipt Stock,
Mrp MTO Status,
Mrp Status,
Schedule Ship Date Estimated,
Mrp Estimated Days,
Actual Delivery Date

Feature explanation:

- Order Number: Each order obtain a unique id
- Order Date: The date that the order was placed by the customer
- Line Number: Each order may contain one or more lines. Each line is a different product and may contain the ordered quantity. Each line in the order has a unique identifier
- Ordered Quantity: The ordered quantity for each product in different unit of measure (it may be pair, pack, piece etc.)
- Item Code: The product's item code

- Line of business: Distinction between the values of line of business is described above.
- Log Xar: This feature indicates whether the product is manufactured in house or it is being delivered by an external supplier.
- Commercial Category Code/ Description: The commercial category described before with its respective code.
- Commercial Group Code/ Description: A more detailed categorization concerning only accessories.
- Proforma/ Credit Hold Flags: Reasons to hold a customer's order based on his credit view. For example, if a customer's balance exceeds his credit limit, his orders are being held. It is expected that this feature affects the order lead time.
- MTO status: Based on each product's stock turnover rate, each item code is characterized with a MTO status.
- MRP (Stocks): MRP (abbreviation for Materials Requirement Planning) is an in house built software which calculates the materials and several components which are needed to manufacture and offer a product. It consists of three steps:
    1. Considers the available inventory of materials and components and distributes the stock to the customers' orders (mostly FIFO).
    2. Identifies which additional resources are needed
    3. Provides action for scheduling of purchasing/ production

  The distributed stock by the MRP software (MRP Production Quantity, MRP Reserved Quantity, MRP Expected Quantity, MRP Remaining Quantity, and MRP Receipt Stock) is a feature that affects the customer's order lead time.

5.2 Lead Time Prediction for architectural profiles

Features for LT Prediction of Architectural Profiles	
Order Year	Expressorder Flag
Order Day Name	Express Line Flag
Order Month	Is Sm Replenishment
Order Number	Preason
Line Number	Prioritydescr
Line Id	Hasloaded
Ordered Date	Ordercomplete
Schedule Ship Date	Remainnotloadedkilos

Request Date	Pickedqtykilos
Order Line Status	Remainpicklistkilos
Proforma Hold Flag	Prosdesmeusikilos
Ordered Quantity	Diadikasiaparagogiskilos
Ordered Quantity2	Programmatismenakilos
Ordered Uom	Apoapothikikilos
Ordered Uom2	Apoparagogikilos
Schedule Ship Date Estimated	Diekperaiosikgs
Country	Subsidiaty
Wms Dt	Is Industrial
Predicttime Dt	Express Flag
Shipment Dt	Ral Categories
Shipment No	Is Thermo
On Hold Days	Supreme Flag
Abc Code	Daily Load Kgs
Promised Dt Lag	Orders Day Count
Orderremarks	In Warehouse Dt
Has Remarks	Planning Difference
Notproformapayed	Delivery Days

Feature explanation:

1. Data describing the product that the order contains:

Architectural or industrial aluminum profile. This project is focused on architectural profiles. The process of satisfying industrial aluminum profiles' demand is different and cannot be described and modeled in the same way of architectural profiles.
Heat insulation. Architectural profiles can be insulated or not. This feature is very important for the determination of the production duration, given that heat insulated profiles must come through an extra production stage, insulation foil production line.
Profile's surface treatment. There are 4 available surfaces for the profiles: Not painted, electrostatic painting, anodized and wood effect painting (sublimation). The surface of each product, affects the production duration. For example, if the product contained in order is not painted it is not necessary to go through the powder coating line. If the requested product is anodized, then it must come through the anodizing plant. Each production stage has its own production capacity and lead time (the latency between

the initiation and execution of each production's stage process).

ABC Product Priority. Depending on the stock turnover ratio and popularity of each product, each aluminum profile is characterized as:

- A
- B
- C
- D
- E (Express Order)
- MTO (Make to Order)

2. Data describing the details of the order:

The amount of the ordered kgs, per order line.

The amount of kgs that are already available in the warehouse, per order line.
The amount of kgs that must come through the production stages, per order line.
Customer data (subsidiary or not, credit control flag),

Seasonality of order. The year, month and day name of the order placement day is taken into account as a feature that can affect the lead time,
The existence of comments on order.

Line ID/Number: Each order may contain multiple lines. Prediction is going to be performed on the order line level and then is going to be aggregated to the order level. (Maximum delivery date of all line numbers in an order is going to be the delivery date of the whole order).

3. Data describing the daily production load:

Number of open orders that must be satisfied through the production facilities
Pending kilos of aluminum profiles that are not yet produced.

Feature engineering

The part of data engineering, is followed by feature engineering. Machine learning models are based on data that can interpret the problem and sufficiently explain the complexity of the given problem, in a way the algorithm can comprehend. For example, most of the machine

learning algorithms don't know how to handle categorical variables.

For both demand forecasting and lead time prediction the following data preprocessing steps are taken:

1. Outlier and anomalies detection
2. Filling missing values
3. Creating rolling window / lag features
4. Encoding categorical values

6 Predictive modeling – Machine learning methods

Since the desired outcome of the predictive algorithms that we are going to use is numerical this thesis is concerned with Regression Techniques.

Regression algorithms are machine learning techniques for predicting continuous numerical values. They are supervised learning tasks which means they require labelled training examples.

Demand Forecasting:

The aim of demand forecasting that this thesis is concerned is to predict the value of the orders that are going to be placed in the next month for accessories.

As for now ensemble methods, namely "xgboost" and "catboost" have been implemented, in the process of predicting the orders' value of June 2019. In total for all aluminum accessories except the solar commercial category, predicted values display a 3% difference from actual ones.

The strategy for predictive future (next month's) sales orders was as follows:

- Load the data into python
- Perform exploratory analysis, remove outliers
- Create features related with commercial categories/ commercial groups
- Create a matrix, which is the Cartesian product of all products and calendar dates and fill it with data from sales orders. If an item was not ordered fill the blanks with zero values. This is especially important, because we the model needs to be informed about zero sales too.

- Add lag features
- add mean encoded data
- add price trend data
- add month
- add days
- add months since last sale/months since first sale features
- cut first year and drop columns which cannot be calculated for the test set
- fit the model, predict and clip targets for the test set

The best performing algorithm for the demand forecasting was the xgboost regressor with the following parameters:

```
max_depth= 8,
n_estimators= 500,
min_child_weight= 1000,
colsample_bytree= 0.7,
subsample= 0.7,
eta= 0.3,
seed= 0
```

XGBoost is an extremely efficient algorithm which delivers accuracy coupled with high performance in comparison to other algorithms. XGBoost, also known as regularized version of GBM (Gradient Boosting Machine). has to offer the following advantages [32]:

1. Regularization: XGBoost has in-built Lasso Regression and Ridge Regression regularization which prevents the model from overfitting. That the reason why XGBoost is also called regularized form of GBM.

2. Parallel Processing: XGBoost uses the power of parallel processing and this is the reason why it is much faster than Gradient Boosting Machine. It utilized many CPU cores to execute the prediction.

3. Missing Values handling: The algorithm contains a built-in capability to handle missing values. The moment XGBoost encounters a missing value at a tree node, it attempts both a left and a right hand split and learns the way leading to higher loss for each node.

4. Cross Validation: XGBoost is capable to run at each iteration, a cross-validation of the boosting process. This enables the user to get the optimum number of boosting iterations.

5. Tree Pruning: While the GBM algorithms stops splitting a tree node when it encounters a negative loss in the split, XGBoost on the other hand make splits taken into consideration the "max_depth" parameter and then and then it starts pruning the tree backwards where there is no positive gain.

The model's maximum depth of the tree was set to 8, because increasing even more makes the model very complex and the consumption of memory aggressively rises with the increase of the maximum depth. Along with the complexity the danger of overfitting the model to the training data arises with a big number of tree depth. The learning rate of the model, defined by the parameter "eta", was set to .3 which acts as the step size shrinkage in order to prevent overfitting. The feature weights are extracted after each step of boosting. The "eta" parameter shrinks the weights to prevent overfitting. "min_child_weight" parameter s set to 1.000, generally the larger value that this parameter is set to the more conservative the model is going to be. "Colsample_bytree", is the ratio of columns when constructing the tree. Every time a tree is constructed the subsampling occurs. Subsample set to 0.7 means the algorithm will randomly sample 30% of the training data before growing the trees. Again this subsample which occurs in every boosting turn, is used to prevent the overfitting of the model.

Lead Time Prediction

The algorithms used for predicting the delivery days (lead time) in this research are:

- Linear Regression,
- Decision Trees,
- Ensemble techniques, gradient boosting methods on top of regression and decision trees,
- Neural Networks,
- Deep Neural Networks.

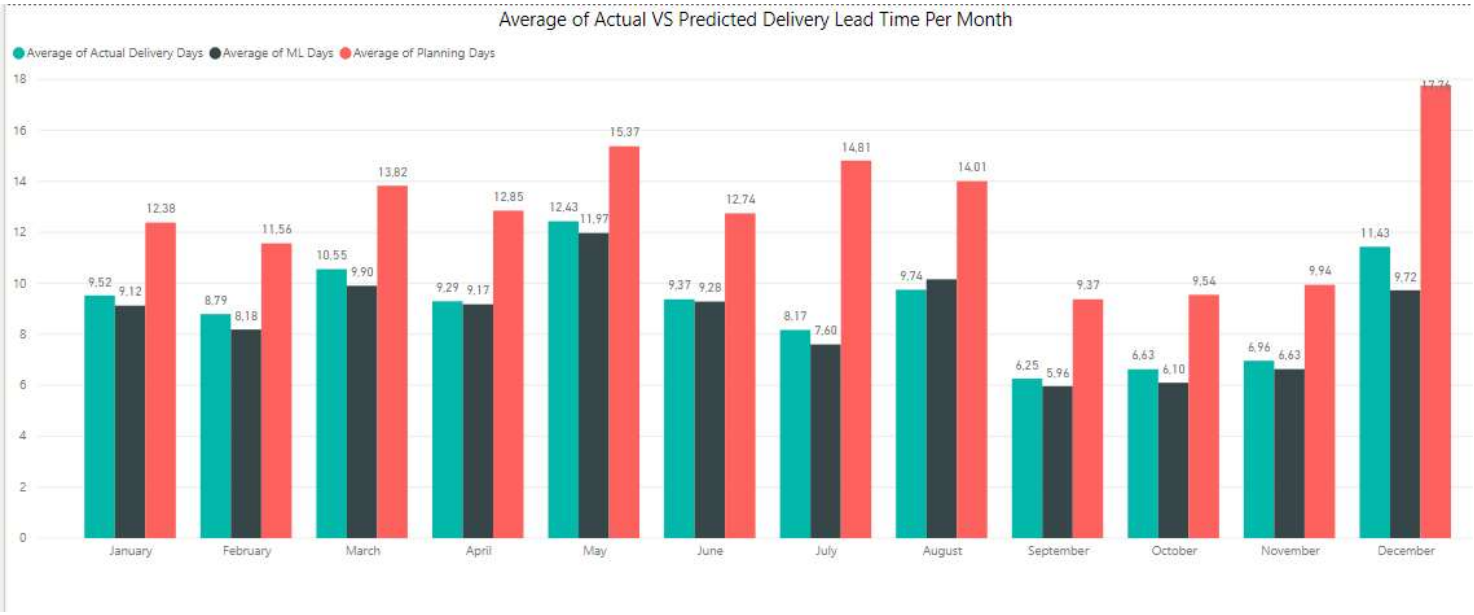
A sample of 38.500 orders of architectural aluminum profiles have been used in order to examine and compare the performance between the existing capacity planning software, and the best performing algorithm so far (a Sequential neural network model with two densely connected hidden layers, and an output layer that returns a single, continuous value using Keras). The orders are from 2018 and the first 4 months of 2019. The best – performing algorithm was implemented as follows: The number of neurons in the input layer equals the number of input variables (features) in the data being processed. In this case, 90 input features

require 90 input neurons. Then, a dense hidden layer with 64 neurons is added to the model. The layer manages its own weight matrix, containing all the connection weights between the neurons and their inputs. It also manages a vector of bias terms (one per neuron). Next, a second dense hidden layer with another 64 neurons is added, also using the ReLU activation function. Finally, a dense output layer with only 1 neuron without an activation function is needed (because the outcome is a continuous variable).

Performance so far:

For the three topics that this thesis is dealing with, the machine learning algorithms that were applied in the company's data performed much better than the techniques that supply chain's division was utilizing so far. In a glance, the performance of the algorithms is displayed in the following table:

Topic	Performance	Remarks
Lead time prediction for aluminum accessories	Average Absolute Error of 9 Days	Industrial orders, whose lead time is affected by contracts (>2 month) are included. Overall a very good performance.
Lead Time Prediction for customer orders of aluminum architectural profiles	Average Absolute Error of 3,4 Days	Very Good performance.
Demand forecasting of next month for aluminum accessories.	Total error of 3%	Decent performance overall, a deeper analysis is required for commercial groups with bigger error.



Picture 15: Comparison of performance between existing system (planning) and ML Algorithm VS Actuals

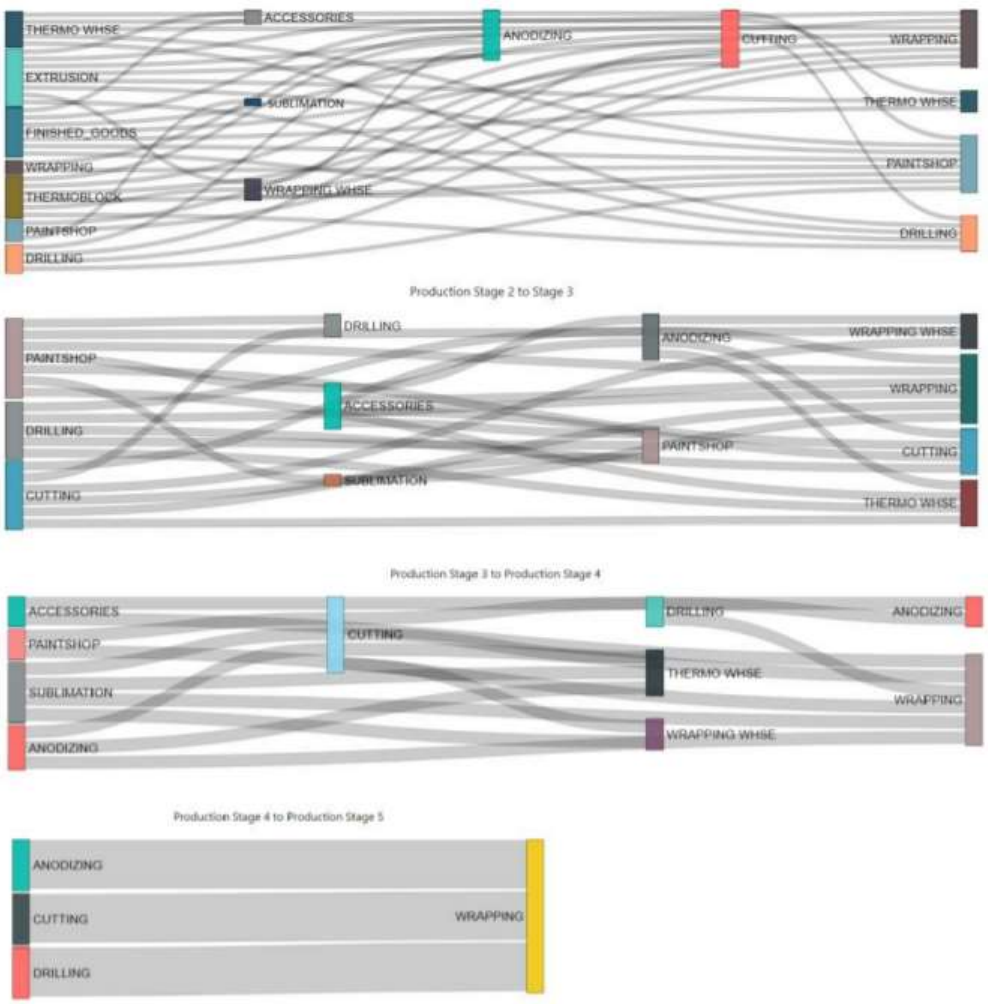
The main conclusions that have been drown so far are non-technical. When dealing with predictive modeling with machine learning in the business area, it has been deduced that two main factors of success are quality data collection and business knowledge. Poor performance in the algorithms has been improved drastically by acquiring more data (from previous not currently operating ERP) and by having meetings and conversations with supply chain professionals who have the business know how and indicated some pitfalls, exceptions and

outliers that otherwise it would be extremely difficult to realize and detect.

7 Future work and challenges

A very important factor for the determination and prediction of production lead times are the production routing that each product has to follow. When ordered, the product may be ready in the warehouse and only needs to be wrapped, while other products needs to be extruded, painted or anodized, cut or drilled etc. For future research concerning the enhancement of the lead time prediction's accuracy, waiting times between stages should be taken into account since this "dead time" affects the total cycle time.

The following Sankey chart depicts the possible routing that each order line may follow. Containing extra details about the capacity of each production stage and the resource that is being used (for example there are four extrusion presses each one with different capacity) is going to provide valuable insights to the machine learning model, thus explain the outcome's variance better.



Picture 16: Sankey Chart: Possible production routings that each ordered item may follow.

Another challenge that has to be taken into account, is the dynamic changes that the production environment is being subject to. While the machine learning algorithms are excellent at detecting the environment's changes when being retrained major changes in the systems like the adoption of new planning methods and systems are not easy to be realized by the algorithms without sufficient data.

8 Bibliography

[1] "Corporate profile," [Online]. Available: https://www.alumil.com/greece/en/corporate/about-us/corporate-profile.

[2] "Linear Regression — Understanding the Theory - Towards Data Science," [Online]. Available: https://towardsdatascience.com/linear-regression-understanding-the-theory-7e53ac2831b5.

[3] "Can neural networks solve any problem? - Towards Data Science," [Online]. Available: https://towardsdatascience.com/can-neural-networks-really-learn-any-function-65e106617fc6.

[4] "Jeff Bezos: Amazon succeeds through focus on customer over competitor - Business Insider," [Online]. Available: https://www.businessinsider.com/amazon-jeff-bezos-success-customer-obsession-2018-9.

[5] "55% of shoppers abandon carts due to shipping costs: infographic – Econsultancy," [Online]. Available: https://econsultancy.com/55-of-shoppers-abandon-carts-due-to-shipping-costs-infographic/.

[6] "Seven reasons why you need to forecast in supply chain | Top 10 | Supply Chain Digital," [Online]. Available: https://www.supplychaindigital.com/top-10/seven-reasons-why-you-need-forecast-supply-chain.

[7] D. Simchi-Levi and M. A. Trick, "Introduction to "little's law as viewed on its 50th anniversary"," *Operations Research,* vol. 59, no. 3, p. 535, 5 2011.

[8] W. J.K., " A simulation study of forecastable due-dates, Management science, Vol. 25, No.4," 1979,, pp. pp. 363-373.

[9] M. M. Vig and K. J. Dooley, "Dynamic rules for due-date assignment," *International Journal of Production Research,* vol. 29, no. 7, pp. 1361-1377, 1991.

[10] S. T. Enns, "Job shop leadtime requirements under conditions of controlled delivery performance," *European Journal of Operational Research,* vol. 77, no. 3, pp. 429-439, 29 9 1994.

[11] S. R. Lawrence, "Negotiating due-dates between customers and producers," *International Journal of Production Economics,* vol. 37, no. 1, pp. 127-138, 1994.

[12] I. P. Tatsiopoulos and B. G. Kingsman, *Lead time management,* vol. 14, 1983, pp. 351-358.

[13] B. G. Kingsman, I. P. Tatsiopoulos and L. C. Hendry, "A structural methodology for managing manufacturing lead times in make-to-order companies," *European Journal of Operational Research,* vol. 40, no. 2, pp. 196-209, 25 5 1989.

[14] "Modelling input-output workload control for dynamic capacity planning in production planning systems - Research Portal | Lancaster University," [Online]. Available: http://www.research.lancs.ac.uk/portal/en/publications/modelling-inputoutput-workload-control-for-dynamic-capacity-planning-in-production-planning-systems(a7736d91-fd7e-4e8c-8719-281f554f70dd)/export.html.

[15] H. P. Van Ooijen and J. W. Bertrand, "Economic due-date setting in job-shops based on routing and workload dependent flow time distribution functions," *International Journal of Production Economics,* vol. 74, no. 1-3, pp. 261-268, 12 2001.

[16] A. Öztürk, S. Kayaligil and N. E. Özdemirel, "Manufacturing lead time estimation using data mining," *European Journal of Operational Research,* vol. 173, no. 2, pp. 683-700, 1 9 2006.

[17] A. Simeone and P. C. Priarone, Envisaging the Future Manufacturing, Design, Technologies and Systems in Innovation Era, 2018.

[18] D. Gyulai, A. Pfeiffer, G. Nick, V. Gallina, W. Sihn and L. Monostori, "Lead time prediction in a flow-shop environment with analytical and machine learning approaches," *IFAC-PapersOnLine,* vol. 51, no. 11, pp. 1029-1034, 1 1 2018.

[19] Y. Meidan, B. Lerner, G. Rabinowitz and M. Hassoun, "Cycle-time key factor identification and prediction in semiconductor manufacturing using machine learning and data mining," in *IEEE Transactions on Semiconductor Manufacturing*, 2011.

[20] A. Alenezi, S. A. Moses and T. B. Trafalis, "Real-time prediction of order flowtimes using support vector regression," *Computers and Operations Research,* vol. 35, no. 11, pp. 3489-3503, 11 2008.

[21] J. Mori and V. Mahalec, "Planning and scheduling of steel plates production. Part I: Estimation of production times via hybrid Bayesian networks for large domain of discrete variables," *Computers and Chemical Engineering,* vol. 79, pp. 113-134, 4 8 2015.

[22] F. J. de Cos Juez, P. J. García Nieto, J. Martínez Torres and J. Taboada Castro, "Analysis of lead times of metallic components in the aerospace industry through a supported vector machine model," *Mathematical and Computer Modelling,* vol. 52, no. 7-8, pp. 1177-1184, 10 2010.

[23] S. Susanto, P. I. Tanaya and A. S. Soembagijo, "Formulating standard product lead time at a textile factory using artificial neural networks," in *Proceeding of 2012 International Conference on Uncertainty Reasoning and Knowledge Engineering, URKE 2012*, 2012.

[24] B. M. Williams, M. Asce, L. A. Hoel and F. Asce, "Modeling and Forecasting Vehicular Traffic Flow as a Seasonal ARIMA Process: Theoretical Basis and Empirical Results".

[25] C. Lim and M. McAleer, "Time series forecasts of international travel demand for Australia," *Tourism Management,* vol. 23, no. 4, pp. 389-396, 2002.

[26] &. G. P. Z. Ching-Wu Chu, "A comparative study of linear and nonlinear models for aggregate retail sales forecasting.," *International Journal of Production Economics,* pp. 217-231.

[27] K. Mupparaju, A. Soni, P. Gujela and M. A. Lanham, "A Comparative Study of Machine Learning Frameworks for Demand Forecasting".

[28] A. Graves, "Generating Sequences With Recurrent Neural Networks".

[29] L. Yue, L. Zhenjiang, Y. Yafeng, T. Zaixia, G. Junjun and Z. Bofeng, "Selective and Heterogeneous SVM Ensemble for demand forecasting," in *Proceedings - 10th IEEE International Conference on Computer and Information Technology, CIT-2010, 7th IEEE International Conference on Embedded Software and Systems, ICESS-2010, ScalCom-2010*, 2010.

[30] I. Watanabe, Tooru and Y. Tateki Imaoka, "AI-Based Demand Forecasting for Both Reliable Forecasting and Efficient Operation: Dynamic Ensemble Forecasting," 2019.

[31] "The Future of Data Integration is No ETL," [Online]. Available: https://www.splicemachine.com/future-data-integration-is-no-etl/.

[32] "The Professionals Point: Advantages of XGBoost Algorithm in Machine Learning," [Online]. Available: http://theprofessionalspoint.blogspot.com/2019/03/advantages-of-xgboost-algorithm-in.html.

[33] G. P. Zhang, "A Comparative Study of Linear and Nonlinear Models for Aggregate Retail Sales Forecasting," in *Proceedings - Annual Meeting of the Decision Sciences Institute*, 2002.

[34] N. Slack and S. Chambers, Operations management, Prentice Hall/Financial Times, 2007, p. 728.

[35] L. Lingitz, V. Gallina, F. Ansari, D. Gyulai, A. Pfeiffer and W. Sihn, "Lead time prediction using machine learning algorithms: A case study by a semiconductor manufacturer," in *Procedia CIRP*, 2018.

[36] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye and T.-Y. Liu, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree".

[37] D. Gyulai, A. Pfeiffer, J. Bergmann and V. Gallina, "Online lead time prediction supporting situation-aware production control," in *Procedia CIRP*, 2018.

[38] T. Berlec and M. Starbek, "Forecasting of Production Order Lead Time in Sme's," in *Products and*

Services; from R&D to Final Solutions, Sciyo, 2010.

[39] "A Solution to Forecast Demand Using Long Short-Term Memory Recurrent Neural Networks for Time Series Forecasting - Purdue Krannert," [Online]. Available: https://krannert.purdue.edu/masters/business-analytics-and-information-management/student-experiences/experiential-learning/2018/a-solution-to-forecast-demand-using-long-short-term-memory-recurrent-neural-networks-for-time-series-forecasting.php.

9 Appendix

In the following appendix is displayed the SQL view that reads the tables from the tables in the data warehouse and the source code from the predictive modeling in python:

9.1 SQL View in company's Data Warehouse

```
- The view that reads from the data warehouse and
--it is used as the dataset in python's predictive modeling:

CREATE OR REPLACE FORCE VIEW "ML_ACCESS_DEMAND_V"
("ORDER_DATE", "DAY_OF_MONTH", "DAY_NAME", "CAL_MONTH", "YEARMONTH", "DATE_BLOCK_NUM", "
MONTH", "YEAR", "ITEM_CODE",  "ITEM_CATEGORY",    "LINE_OF_BUSINESS", "COMMERCIAL_CATEGO
RY", "COMMERCIAL_GROUP", "UNIT_PRICE", "VALUE", "TRANSACTIONS") AS
select
a.order_date, d.day_of_month, d.day_name, d.cal_month,
case
when d.cal_month > 9 then to_number(d.year || d.cal_month) else to_number(d.year
|| '0'|| d.cal_month) end yearmonth,
dense_rank() over( order by
case
when d.cal_month > 9 then to_number(d.year || d.cal_month) else to_number(d.year
|| '0' || d.cal_month) end
) date_block_num, d.month,
d.year, b.item_code, b.item_category, b.line_of_business,
b.commercial_descr, b.commercial_group_descr, a.unit_price, a.line_amount,
1 transactions from
w_order_line_f a
join w_product_d b on a.item_wid = b.row_wid
join w_day_d d on to_char(a.order_date,'rrrrmmdd') = d.row_wid where
item_category = 'ΕΞΑΡΤΗΜΑΤΑ' and d.year >= 2016
;
```

9.2 Code for Accessories lead time prediction

```
import pandas as pd
path = r'C:\Users\v.manasas\Documents\Subject Areas\Thesis\Lead Time Predict
ion'
train = pd.read_excel(path+r'\Training Data - Lead Time Prediction ( Accesso
```

```
ries).xlsx')
actual_LT = pd.read_excel(path+r'\Training Data - Lead Time Actual Delivery
Dates.xlsx')

train = pd.merge(train, actual_LT, on = ['ORDER_NUMBER','LINE_NUMBER'])
train.dropna(subset=['LAST_LINE_SHIP_DT'],inplace = True)
train['ORDERED_DATE'] = pd.to_datetime(train['ORDERED_DATE'],
       format='%d-%m-%Y %H:%M:%S', utc=True)
train['LAST_LINE_SHIP_DT'] = pd.to_datetime(train['LAST_LINE_SHIP_DT'], form
at='%d-%m-%Y %H:%M:%S', utc=True)
train['ORDERED_DATE'] = pd.to_datetime(train['ORDERED_DATE'] )
train['LAST_LINE_SHIP_DT'] = pd.to_datetime(train['LAST_LINE_SHIP_DT'] )
train['ACTUAL_LT'] = (train['LAST_LINE_SHIP_DT'] -
train['ORDERED_DATE']).dt.days
train['DAY'] = pd.DatetimeIndex(train['ORDERED_DATE']).day
train['YEAR'] = pd.DatetimeIndex(train['ORDERED_DATE']).year
train['MONTH'] = pd.DatetimeIndex(train['ORDERED_DATE']).month
train['WEEKDAY'] = pd.DatetimeIndex(train['ORDERED_DATE']).weekday_name
train = train.dropna(subset=['ORDERED_DATE', 'LAST_LINE_SHIP_DT'])
train.drop( columns = ['ORDER_NUMBER','ORDERED_DATE','LINE_NUMBER','ITEM_COD
E','COMMERCIAL_DESCR','COMMERCIAL_GROUP_DESCR','SCHEDULE_SHIP_DATE_ESTIMATED
','LAST_LINE_SHIP_DT'],axis = 1, inplace = True)
train = train.fillna(0)
x = train.drop(['ACTUAL_LT'], axis=1)
y = train['ACTUAL_LT']
categorical_cols = ['LOG_XAR', 'COMMERCIAL_CODE','LINE_OF_BUSINESS',
       'COMMERCIAL_GROUP_CODE', 'PROFORMA_HOLD_FLAG', 'CREDIT_HOLD_FLAG','MR
P_MTO_STATUS',
       'MRP_STATUS','DAY', 'YEAR', 'MONTH', 'WEEKDAY']
x = pd.get_dummies(x,columns = categorical_cols, prefix = categorical_cols)
from sklearn import preprocessing
import numpy as np
X_scaled = preprocessing.scale(x)
from sklearn.model_selection import train_test_split
xTrain, xTest, yTrain, yTest = train_test_split(X_scaled, y, test_size = 1/3
, random_state = 0)
# from sklearn import ensemble
# params = {'n_estimators': 500, 'max_depth': 4, 'min_samples_split': 2,
#           'learning_rate': 0.01, 'loss': 'ls'}
# clf = ensemble.GradientBoostingRegressor(**params)
# clf.fit(xTrain, yTrain)
# prediction = clf.predict(xTest)
# prediction_ens = pd.DataFrame(prediction)
# real_ens = pd.DataFrame(yTest)
# prediction_ens.to_excel(r'C:\Users\v.manasas\Desktop\prediction.xlsx',inde
x= False)
# real_ens.to_excel(r'C:\Users\v.manasas\Desktop\actual.xlsx',index= False)
xTrain = pd.DataFrame(xTrain)
from__future__import absolute_import, division, print_function
```

```
import pathlib
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
def huber_loss(y_true, y_pred, clip_delta=1.0):
  error = y_true - y_pred
  cond  = tf.keras.backend.abs(error) < clip_delta

  squared_loss = 0.5 * tf.keras.backend.square(error)
  linear_loss  = clip_delta * (tf.keras.backend.abs(error) - 0.5 * clip_delt
a)

  return tf.where(cond, squared_loss, linear_loss)
def build_model():
  model = keras.Sequential([
    layers.Dense(64, activation=tf.nn.relu, input_shape=[len(xTrain.keys())]
),
    layers.Dense(64, activation=tf.nn.relu),
    layers.Dense(1)
  ])
  optimizer = tf.keras.optimizers.RMSprop(0.001)
  model.compile(loss=huber_loss,
                optimizer=optimizer,
                metrics=['mean_absolute_error', 'mean_squared_error'])
  return model
model = build_model()
# Display training progress by printing a single dot for each completed epoc
h
class PrintDot(keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs):
    if epoch % 100 == 0: print('')
    print('.', end='')

EPOCHS = 100
history = model.fit(
  xTrain, yTrain,
  epochs=EPOCHS, validation_split = 0.2, verbose=0,
  callbacks=[PrintDot()])
loss, mae, mse = model.evaluate(xTest, yTest, verbose=0)
print("Testing set Mean Abs Error: {:5.2f} days".format(mae))
```

9.3 Code for accessories demand forecasting

```
import numpy as np
import pandas as pd
path = r'C:\Users\v.manasas\Documents\Subject Areas\Thesis'
```

```
train = pd.read_excel(path+r'\train.xlsx')
items = pd.read_excel(path+r'\items.xlsx')
test = pd.read_excel(path+r'\test.xlsx')
train = train.query('UNIT_PRICE > 0')

#outliers
train= train.query('VALUE<1000000')
train= train.query('VALUE>0')
exculed_commercials = ['ΦΩΤΟΒΟΛΤΑΙΚΑ','ΛΑΣΤΙΧΑ']
train = train[~train.COMMERCIAL_CATEGORY.isin(exculed_commercials)]
train[['COMMERCIAL_CATEGORY',              'COMMERCIAL_GROUP']]              =
train[['COMMERCIAL_CATEGORY','COMMERCIAL_GROUP']].fillna(value='GOOGLE')



train['YEARMONTH'] = train['YEAR'].map(str) + train['MONTH'].map(str)
train['DATE_BLOCK_NUM'] = train.YEARMONTH.rank(method='dense').astype(int)
# Group by month in this case "date_block_num" and aggregate features.

train_monthly                                                                  =
train.sort_values('DATE_BLOCK_NUM').groupby(['DATE_BLOCK_NUM',   'ITEM_CODE',
'COMMERCIAL_CATEGORY', 'COMMERCIAL_GROUP','YEAR','MONTH'], as_index=False)
train_monthly  =  train_monthly.agg({'UNIT_PRICE':['mean'],  'VALUE':['sum',
'mean'],'TRANSACTIONS':['sum']})
train_monthly.head()

# Rename features.

train_monthly.columns        =        ['date_block_num',        'item_code',
'commercial_category',  'commercial_group','year','month',  'mean_item_price',
'value', 'mean_value', 'transactions']

#   Build   a   data   set   with   all   the   possible   combinations   of
['date_block_num','item_code'] so we won't have missing records.

# each item code corresponds to 1 commercial category and 1 commercial group
# so there is no need to include commercial category and commercial groups.
date_block_num_unique =  train_monthly['date_block_num'].unique()
item_unique = items['ITEM_CODE'].unique()
empty_df = []
for i in date_block_num_unique:
    for item in item_unique:
        empty_df.append([i, item])

empty_df = pd.DataFrame(empty_df, columns=['date_block_num','item_code'])
```

```
# Merge the train set with the complete set (missing records will be filled
with 0).

train_monthly = pd.merge(empty_df, train_monthly,
on=['date_block_num','item_code'], how='left')
train_monthly['year'] = train_monthly['date_block_num'].apply(lambda x:
((x//12) + 2013))
train_monthly['month'] = train_monthly['date_block_num'].apply(lambda x: (x%
12))
train_monthly.fillna(0, inplace=True)



#get the commercial category and commercial group for months with zero sales
train_monthly = pd.merge(train_monthly, items, left_on='item_code',
right_on='ITEM_CODE', how='left')
train_monthly.drop(['ITEM_CODE', 'commercial_category','commercial_group'],
axis = 1, inplace = True)
train_monthly.rename(columns={"COMMERCIAL_CATEGORY": "commercial_category",
"COMMERCIAL_GROUP": "commercial_group","YEAR":"year","MONTH":"month"},inplace
= True)
train_monthly['surrogate_key'] = train_monthly.item_code + '-'
+train_monthly.date_block_num.map(str)
train_monthly.head()

# for months that we have zero sales we do not know the item's unit price
# so for each combination of item and month we are going to get the nearest
mean unit price
s=train_monthly.loc[train_monthly.mean_item_price!=0]
s=pd.merge_asof(train_monthly.sort_values('date_block_num'),s.sort_values('d
ate_block_num'),on='date_block_num',by='item_code',direction='nearest')
s.fillna(0, inplace=True)
s = s[['surrogate_key_x','mean_item_price_y']]
train_monthly = pd.merge(train_monthly, s, left_on='surrogate_key',
right_on='surrogate_key_x', how='left')
train_monthly.drop(['mean_item_price','surrogate_key_x'],axis = 1, inplace =
True)
train_monthly.rename(columns={"mean_item_price_y": "mean_item_price"},inplace
= True)



train_monthly.loc[train_monthly['item_code'] == 'EX-7651382273'].tail()

# Extract time based features.train_monthly['year'] =
train_monthly['date_block_num'].apply(lambda x: ((x//12) + 2016))
```

```
train_monthly['month'] = train_monthly['date_block_num'].apply(lambda x: (x%
12))
# Creating the label value for our model

# Our label will be the "value" of the next month, as we are dealing with a
forecast problem.
train_monthly['value_month']                                                =
train_monthly.sort_values('date_block_num').groupby(['item_code',
'commercial_category','commercial_group'])['value'].shift(-1)


gp_item_price                                                               =
train_monthly.sort_values('date_block_num').groupby(['item_code'],
as_index=False).agg({'mean_item_price':[np.min, np.max]})
gp_item_price.columns       =       ['item_code',       'hist_min_item_price',
'hist_max_item_price']
train_monthly   =   pd.merge(train_monthly,   gp_item_price,   on='item_code',
how='left')


train_monthly['price_increase']    =    train_monthly['mean_item_price']    -
train_monthly['hist_min_item_price']
train_monthly['price_decrease']   =   train_monthly['hist_max_item_price']   -
train_monthly['mean_item_price']


# Rolling window based features (window = 3 months)
# Min value
f_min = lambda x: x.rolling(window=12, min_periods=1).min()
# Max value
f_max = lambda x: x.rolling(window=12, min_periods=1).max()
# Mean value
f_mean = lambda x: x.rolling(window=12, min_periods=1).mean()
# Standard deviation
f_std = lambda x: x.rolling(window=12, min_periods=1).std()

function_list = [f_min, f_max, f_mean, f_std]
function_name = ['min', 'max', 'mean', 'std']

for i in range(len(function_list)):
    train_monthly[('value_%s'          %          function_name[i])]          =
train_monthly.sort_values('date_block_num').groupby(['commercial_group',
'commercial_category', 'item_code'])['value'].apply(function_list[i])

# Fill the empty std features with 0
train_monthly['value_std'].fillna(0, inplace=True)


# lag based features
```

```
lag_list = [1, 2, 3, 6, 12, 24]

for lag in lag_list:
    ft_name = ('value_shifted_%s' % lag)
    train_monthly[ft_name] =
train_monthly.sort_values('date_block_num').groupby(['commercial_category',
'commercial_group', 'item_code'])['value'].shift(lag)
    # Fill the empty shifted features with 0
    train_monthly[ft_name].fillna(0, inplace=True)


# Item sales count trend.
train_monthly['item_trend'] = train_monthly['value']

for lag in lag_list:
    ft_name = ('value_shifted_%s' % lag)
    train_monthly['item_trend'] -= train_monthly[ft_name]

train_monthly['item_trend'] /= len(lag_list) + 1


train_monthly.fillna(0, inplace=True)


#      train_monthly      =      pd.get_dummies(train_monthly,columns      =
'commercial_category', prefix = 'cc_')


#Our train set will be the first 25 - 76 blocks, and test will be block 77 (
because we are trying to predict
# block number 78. The label value_month of block 77 represents the sales of
block 78.
#leaving the first 3 months out because we use a 3 month window to generate
features, so these first 3 month won't have really windowed useful features
train_set  =  train_monthly.loc[(train_monthly['date_block_num']  >  24)  &
(train_monthly['date_block_num'] < 77)]
test_set = train_monthly.loc[train_monthly['date_block_num'] == 77]

#mean encodings

gp_commercial_group_mean =
train_set.groupby(['commercial_group']).agg({'value': ['mean']})
gp_commercial_group_mean.columns = ['commercial_group_mean']
gp_commercial_group_mean.reset_index(inplace=True)

gp_commercial_category_mean =
train_set.groupby(['commercial_category']).agg({'value': ['mean']})
gp_commercial_category_mean.columns = ['commercial_category_mean']
gp_commercial_category_mean.reset_index(inplace=True)
```

```
gp_item_code_mean = train_set.groupby(['item_code']).agg({'value': ['mean']})
gp_item_code_mean.columns = ['item_code_meand']
gp_item_code_mean.reset_index(inplace=True)

gp_category_item_mean = train_set.groupby(['commercial_category',
'item_code']).agg({'value': ['mean']})
gp_category_item_mean.columns = ['category_item_mean']
gp_category_item_mean.reset_index(inplace=True)

gp_year_mean = train_set.groupby(['year']).agg({'value': ['mean']})
gp_year_mean.columns = ['year_mean']
gp_year_mean.reset_index(inplace=True)

gp_month_mean = train_set.groupby(['month']).agg({'value': ['mean']})
gp_month_mean.columns = ['month_mean']
gp_month_mean.reset_index(inplace=True)



# Add mean encoding features to train set.
train_set = pd.merge(train_set, gp_commercial_group_mean,
on=['commercial_group'], how='left')
train_set = pd.merge(train_set, gp_commercial_category_mean,
on=['commercial_category'], how='left')
train_set = pd.merge(train_set, gp_item_code_mean, on=['item_code'],
how='left')
train_set = pd.merge(train_set, gp_category_item_mean,
on=['item_code','commercial_category'], how='left')
train_set = pd.merge(train_set, gp_month_mean, on=['month'], how='left')
# Add mean encoding features to test set.
test_set = pd.merge(test_set, gp_commercial_group_mean,
on=['commercial_group'], how='left')
test_set = pd.merge(test_set, gp_commercial_category_mean,
on=['commercial_category'], how='left')
test_set = pd.merge(test_set, gp_item_code_mean, on=['item_code'], how='left')
test_set = pd.merge(test_set, gp_category_item_mean,
on=['item_code','commercial_category'], how='left')
test_set = pd.merge(test_set, gp_month_mean, on=['month'], how='left')


# Create train and validation sets and labels.
X_train = train_set.drop(['value_month', 'date_block_num'], axis=1)
Y_train = train_set['value_month'].astype(int)
X_test = test_set.drop(['value_month', 'date_block_num'], axis=1)
Y_test = test_set['value_month'].astype(int)


validation = test_set[['item_code','value_month']]
```

```
X_train.columns


X_test.drop(['item_code','commercial_category',           'commercial_group',
'surrogate_key','year', 'month'],inplace = True, axis = 1)
X_train.drop(['item_code','commercial_category',          'commercial_group',
'surrogate_key','year', 'month'],inplace = True, axis = 1)


import datetime
import warnings
import numpy as np
import catboost
from catboost import Pool
from catboost import CatBoostRegressor
from xgboost import XGBRegressor
from xgboost import plot_importance
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler, MinMaxScaler

get_ipython().run_line_magic('matplotlib', 'inline')
pd.set_option('display.float_format', lambda x: '%.2f' % x)
warnings.filterwarnings("ignore")
X_train[X_train.columns] = X_train[X_train.columns].astype('int32')

X_test[X_test.columns] = X_test[X_test.columns].astype('int32')
cat_features = [0, 1, 7, 8]
catboost_model = CatBoostRegressor(
    iterations=500,
    max_ctr_complexity=4,
    random_seed=0,
    od_type='Iter',
    od_wait=25,
    verbose=50,
    depth=4
)

catboost_model.fit(
    X_train, Y_train,
    cat_features=cat_features
)
catboost_test_pred = catboost_model.predict(X_test)

catboost_test_pred = pd.DataFrame(catboost_test_pred)
```

```
validation.to_excel(r'C:\Users\v.manasas\Documents\Subject
Areas\Thesis\validation.xls', index=False)
catboost_test_pred.to_excel(r'C:\Users\v.manasas\Documents\Subject
Areas\Thesis\prediction.xls', index=False)


train_monthly.head()

xgb_features = X_train.columns
xgb_train = X_train[xgb_features]
xgb_test = X_test[xgb_features]
xgb_model = XGBRegressor(max_depth=8,
                         n_estimators=500,
                         min_child_weight=1000,
                         colsample_bytree=0.7,
                         subsample=0.7,
                         eta=0.3,
                         seed=0)
xgb_model.fit(xgb_train,
              Y_train,
              verbose=20,
              )



xgboost_test_pred = xgb_model.predict(X_test)
xgboost_test_pred = pd.DataFrame(xgboost_test_pred)
validation.to_excel(r'C:\Users\v.manasas\Documents\Subject
Areas\Thesis\validation_x.xls', index=False)
xgboost_test_pred.to_excel(r'C:\Users\v.manasas\Documents\Subject
Areas\Thesis\xg_prediction.xls', index=False)
```

9.4 Lead time prediction for aluminum profiles

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler

train  =  pd.read_excel(r'C:\Users\v.manasas\Desktop\Subject  Areas\Alumil  -
IHU\ORDER LINES.xlsx')
train.columns.values.tolist()
drop_columns = ['ORDER_NUMBER',
'LINE_NUMBER',
'PROFORMA_HOLD_FLAG',
'ORDER_LINE_STATUS',
'LINE_ID',
'ORDERED_DATE',
```

```
'SCHEDULE_SHIP_DATE',
'REQUEST_DATE',
'ORDERED_QUANTITY',
'ORDERED_QUANTITY2',
'ORDERED_UOM',
'ORDERED_UOM2',
'SCHEDULE_SHIP_DATE_ESTIMATED',
'PLN_PREDICTTIME_DT',
'SHIPMENT_DT',
'SHIPMENT_NO',
'ON_HOLD_DAYS',
'IN_WAREHOUSE_DT',
'PLN_WMS_DT',
'PLN_PROMISED_DT_LAG',
'DAILY_LOAD_KGS',
'ORDERS DAY COUNT',
'PLN_ORDERREMARKS',
'PLANNING_PROMISED_DATE'
                ]
train = train.drop(drop_columns,axis =1,inplace = False)
train.info()
train.dropna(axis = 0, inplace = True)
train.info()
orders= train

orders.info()

x = orders.drop(['DELIVERY_DAYS'], axis=1)
y = orders['DELIVERY_DAYS']
categorical_cols = ['AUDIT_YEAR',
'ORDER_DAY_NAME',
'AUDIT_MONTH',
'COUNTRY',
'ABC_CODE',
'PLN_NOTPROFORMAPAYED',
'PLN_EXPRESSORDER',
'PLN_EXPRESS_LINE_FLAG',
'PLN_IS_SM_REPLENISHMENT',
'PLN_PREASON',
'PLN_PRIORITYDESCR',
'PLN_HASLOADED',
'PLN_ORDERCOMPLETE',
'HAS REMARKS',
'SUBSIDIATY',
'IS_INDUSTRIAL',
'EXPRESS_FLAG',
'RAL_CATEGORIES',
'IS_THERMO',
'SUPREME_FLAG']
```

```
x = pd.get_dummies(x,columns = categorical_cols, prefix = categorical_cols)


x.columns.values.tolist()
x.drop(['PROFIL_USE'],axis =1, inplace = True)

from sklearn.model_selection import train_test_split
xTrain, xTest, yTrain, yTest = train_test_split(x, y, test_size = 1/3,
random_state = 0)
from sklearn import preprocessing
import numpy as np
x_Train = preprocessing.scale(xTrain)
x_Test = preprocessing.scale(xTest)


from___future___import absolute_import, division, print_function

import pathlib
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers



x_Train = pd.DataFrame(x_Train)
def huber_loss(y_true, y_pred, clip_delta=1.0):
  error = y_true - y_pred
  cond  = tf.keras.backend.abs(error) < clip_delta

  squared_loss = 0.5 * tf.keras.backend.square(error)
  linear_loss = clip_delta * (tf.keras.backend.abs(error) - 0.5 * clip_delta)

  return tf.where(cond, squared_loss, linear_loss)
def build_model():
  model = keras.Sequential([
    layers.Dense(64,                                    activation=tf.nn.relu,
input_shape=[len(x_Train.keys())]),
    layers.Dense(64, activation=tf.nn.relu),
    layers.Dense(1)
  ])

  optimizer = tf.keras.optimizers.RMSprop(0.001)

  model.compile(loss=huber_loss,
                optimizer=optimizer,
                metrics=['mean_absolute_error', 'mean_squared_error'])
```

```
  return model


model = build_model()
# Display training progress by printing a single dot for each completed epoch
class PrintDot(keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs):
    if epoch % 100 == 0: print('')
    print('.', end='')

EPOCHS = 1000

history = model.fit(
  x_Train, yTrain,
  epochs=EPOCHS, validation_split = 0.2, verbose=0,
  callbacks=[PrintDot()])


loss, mae, mse = model.evaluate(x_Test, yTest, verbose=0)

print("Testing set Mean Abs Error: {:5.2f} days".format(mae))
test_predictions = model.predict(x_Test).flatten()

model.summary()

hist = pd.DataFrame(history.history)
hist['epoch'] = history.epoch
hist.tail(10)


def plot_history(history):
  hist = pd.DataFrame(history.history)
  hist['epoch'] = history.epoch

  plt.figure()
  plt.xlabel('Epoch')
  plt.ylabel('Mean Abs Error [Days]')
  plt.plot(hist['epoch'], hist['mean_absolute_error'],
           label='Train Error')
  plt.plot(hist['epoch'], hist['val_mean_absolute_error'],
           label = 'Val Error')
  plt.ylim([0,5])
  plt.legend()

  plt.figure()
  plt.xlabel('Epoch')
  plt.ylabel('Mean Square Error [$Days$]')
  plt.plot(hist['epoch'], hist['mean_squared_error'],
           label='Train Error')
```

```
  plt.plot(hist['epoch'], hist['val_mean_squared_error'],
           label = 'Val Error')
  plt.ylim([0,20])
  plt.legend()
  plt.show()


plot_history(history)
class PrintDot(keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs):
    if epoch % 100 == 0: print('')
    print('.', end='')

model = build_model()

# The patience parameter is the amount of epochs to check for improvement
early_stop = keras.callbacks.EarlyStopping(monitor='val_loss', patience=10)

history = model.fit(x_Train, yTrain, epochs=1000,
                    validation_split = 0.2, verbose=0, callbacks=[early_stop,
PrintDot()])

plot_history(history)
loss, mae, mse = model.evaluate(x_Test, yTest, verbose=0)

print("Testing set Mean Abs Error: {:5.2f} days".format(mae))
error = abs(test_predictions - yTest)
plt.hist(error, bins = 10000)
plt.xlim(xmin=0, xmax = 100)
plt.xlabel("Prediction Error [Days]")
_ = plt.ylabel("Count")

saver = tf.train.Saver()
model_json = model.to_json()
with      open(r"C:\Users\v.manasas\Desktop\Subject      Areas\Alumil      -
IHU\model_huber.json", "w") as json_file:
    json_file.write(model_json)

model.save_weights(r"C:\Users\v.manasas\Desktop\Subject    Areas\Alumil    -
IHU\model_huber.h5")
```