INVENTORY MANAGEMENT AND FORECASTING SYSTEM USING ML

A Final Report for the BDM Capstone Project

Submitted by

Name: Himanshu Savargaonkar

Roll number: 22DS3000140

IITM Online BS Degree Program,

Indian Institute of Technology, Madras, Chennai

Tamil Nadu, India, 600036

Contents

1. Executive Summary and Title 3
2. Detailed Explanation of Analysis Process/ Method 3
2.1 Data Pre-Processing and Analysis 3
2.1.1 Raw Data: 3
2.1.2 Metadata 4
2.1.3 Preprocessing 4
2.2 Analysis and Implementation of Machine Learning 5
2.2.1 Choosing Regression Algorithm 6
2.2.2 Implementing Regression Algorithms 7
2.2.3 Dealing with Variation and Fluctuations in the Data 8
2.3 Deliverables and Application 8
2.4 (FUTURE SCOPE) AutoRegressive Integrated Moving Average 9
2.4.1 Introduction to ARIMA and SARIMA Models 9
2.4.2 ARIMA Equation and Tuning 9
2.4.3 Implementation and Results 10
3. Results 11
3.1 Dead Inventory and Zero Inventory Reports 11
3.2 ML Analysis Results 13
3.2.1 Train-Validate Data Sets 13
3.2.2 ML Algorithm Scoring Metrics 13
3.2.3 Observed Results 14
3.2.4 Algorithm Optimization 16
3.3 Algorithm Finalization and Implementation 16
3.4 Deliverable Application 17
4. Recommendations and Conclusions 18
4.1 Types of Inventory Items 18
4.2 Dead Inventory and Zero Inventory 19
4.3 Using ML in the inventory management. 19
5. Conclusion 19

Declaration Statement

I am working on a Project titled “Inventory Management and Forecasting System Using Machine Learning (ML)”. I extend my appreciation to Nasan Medical, for providing the necessary resources that enabled me to conduct my project.

I hereby assert that the data presented and assessed in this project report is genuine and precise to the utmost extent of my knowledge and capabilities. The data has been gathered from primary sources and carefully analyzed to assure its reliability.

Additionally, I affirm that all procedures employed for the purpose of data collection and analysis have been duly explained in this report. The outcomes and inferences derived from the data are an accurate depiction of the findings acquired through thorough analytical procedures.

I am dedicated to adhering to the principles of academic honesty and integrity, and I am receptive to any additional examination or validation of the data contained in this project report.

I understand that the execution of this project is intended for individual completion and is not to be undertaken collectively. I thus affirm that I am not engaged in any form of collaboration with other individuals, and that all the work undertaken has been solely conducted by me. In the event that plagiarism is detected in the report at any stage of the project's completion, I am fully aware and prepared to accept disciplinary measures imposed by the relevant authority.

I understand that all recommendations made in this project report are within the context of the academic project taken up towards course fulfillment in the BS Degree Program offered by IIT Madras. The institution does not endorse any of the claims or comments.

Signature of Candidate: 

Name: Himanshu Savargaonkar

Date: 02/10/2023

1. Executive Summary and Title

The primary objective of this project was to develop an intelligent solution to enhance inventory management for Nassan Medical Equipment. Machine Learning techniques were employed to achieve this goal. The project was divided into three key stages.

In the initial stage, data received from the company was subjected to comprehensive preprocessing. This phase yielded valuable reports, including dead-inventory and zero-inventory reports. These reports were instrumental in optimizing the existing inventory management system.

The second stage involved the utilization of the pre-processed data to train various regression algorithms. After rigorous statistical analysis, Lasso regression emerged as the most effective algorithm. The regression model was tuned and then applied to the sorted inventory items, providing valuable insights for the company.

The third and final stage focused on user-friendliness. A graphical user interface (GUI) was developed, integrating the ML model on the backend. This user-friendly interface empowered the company to use the ML algorithm more efficiently. It facilitated accurate predictions regarding the future usage of all inventory items.

2. Detailed Explanation of Analysis Process/ Method

2.1 Data Pre-Processing and Analysis

2.1.1 Raw Data:

The data was collected in the form of an Excel workbook. The workbook was subdivided into each item's transactions. Below in Figure 1, we see the transactions for the first item in the provided data.

Trxn Date	Trxn 12	Final Qty	Type	Unnamed:	Qty	Value	Qty.1	Value.1	Qty.2	Value.2
Item Group : 10 - Raw Materials										
100101001			alluminium clip for to-		220 pack.		UOM	NO	269	707.47
30-Aug-2019	TOR20001329		0032 0001	milind use	0	0	1	2.63	268	704.84
16-Sep-2019	TOW20001636		0007 0001		0	0	6	15.78	262	689.06
24-Dec-2019	TOW20002513		0007 0002		21	55.23	0	0	283	744.29
24-Dec-2019	TOW20002512		0007 0001		0	0	21	55.23	262	689.06
24-Dec-2019	TOW20002514		0007 0001		0	0	20	52.6	242	636.46
16-Mar-2020	TOW20003391		0042 0001		0	0	15	39.45	227	597.01
17-Mar-2020	TOW20003400		0042 0001		0	0	15	39.45	212	557.56
18-Mar-2020	TOW20003421		0042 0001		0	0	15	39.45	197	518.11
19-Mar-2020	TOW20003446		0042 0001		0	0	15	39.45	182	478.66
21-Mar-2020	TOW20003459		0042 0001		0	0	90	236.7	92	241.96
29-Apr-2020	TOW21000065		0007 0001		0	0	8	21.04	84	220.92
24-Aug-2020	TOW21000858		0007 0001		0	0	25	65.75	59	155.17
11-Sep-2020	GRN21000581		0003 0001	S B Enterp	50	125.5	0	0	109	280.67
24-Sep-2020	TOW21001137		0007 0001		0	0	11	28.32	98	252.35
22-Oct-2020	TOW21001425		0007 0001		0	0	10	25.75	88	226.6
27-Oct-2020	GRN21000717		0001 0001	S B Enterp	100	250	0	0	188	476.6
27-Oct-2020	TOW21001485		0007 0001		0	0	10	25.35	178	451.25
05-Nov-2020	TOW21001573		0007 0001		0	0	20	50.7	158	400.55
19-Nov-2020	TOR21001296		0001 0002	wrong mat	100	253.51	0	0	258	654.06
19-Nov-2020	TOR21001295		0001 0001	wrong mat	0	0	100	253.51	158	400.55
19-Nov-2020	TOL21000049		0001 0001	wrong mat	0	0	100	253.51	58	147.04
25-Nov-2020	TOW21001703		0007 0001		0	0	20	50.7	38	96.34
07-Dec-2020	TOW21001810		0007 0001		0	0	20	50.71	18	45.63

Figure 1: Raw Database

As you can see in Figure 1, we see the transaction data for the “aluminium clip” item. In the first row of the entry, we can note the product or item identification number (ID), the item name and the initial stock quantity, and the price of the current stock present in the inventory from when the report was generated.

Following are the transactions that took place in the provided time span of the report. The data presented in each transaction is:

- Date
- Part Footprint
- Transaction ID
- Comments Added by The Inventory Controller
- Incoming Stock Quantity
- Incoming Stock Price
- Outgoing Stock Quantity
- Outgoing Stock Price
- Available Stock Quantity
- Available Stock Price

In the report, there are 2 main types of transactions, incoming stock, and outgoing stock. These transactions are not classified in the report provided but are simply represented by having the necessary fields populated in the above list and others set to 0. For example, in the case of an outgoing stock transaction, the Incoming stock quantity and price fields are set to 0 and the outgoing stock quantity and price fields represent the actual transaction.

2.1.2 Metadata

The provided data has the following meta-data:

Time span – 3 years (July 2019- June 2023)

Number of items tracked – 3,119 items.

Total number of transactions – 1,90,648 transactions.

Final stock quantity – 13,49,587 components.

2.1.3 Preprocessing

The data that was received was in a raw format. It was generated by the internal inventory management program and thus had many discrepancies that needed to be handled manually.

Duplicate data was one of the issues identified in the database. A few of the items had duplicate item numbers and were represented twice in the database. They needed to be merged into a single item for further processing. Another point to be resolved was based on the comments added to the items, few items had comments like, “Do not use” or “Duplicate”. These also needed to be addressed and resolved before the analytical processing could be started.

Once the data was refined from these mistakes, I segregated the data to help in the further analysis. The segregation was conducted on two main axes, the number of transactions and the stock

quantity present at the end. These two factors were selected as they contribute the most from a business point of view. The number of transactions shows the dynamicity of the stock and helps identify products that most commonly used. While the stock quantity helps us identify the items which are proposed to have a higher demand in the long term.

The data was segregated into 5 sections on both the axes, their relative findings are displayed below:

Number of transactions:

- Very Low (0-5): 982 items
- Low (6-100): 1608 items
- Medium (101-500): 488 items
- High (501-1000): 34 items
- Very High (>1000): 7 items

Final Stock Quantity:

- Very Low (0-5): 1320 items
- Low (6-100): 894 items
- Medium (101-500): 486 items
- High (501-1000): 125 items
- Very High (>1000): 294 items

After the segregation it was identified that many items with low transaction numbers had not been used for over a year. Upon discussion with the company, it was decided to generate a dead-inventory report. Alongside this a separate report was requested for dead inventory items with zero inventory called a zero-inventory report. These reports were generated based on the preliminary analysis of the data.

After preprocessing it was identified that most of the items do not have many transactions over the year. These items are easy to manage from an inventory management point of view for a human as the amount of data is small and a trend of usage can be easily spotted.

It was decided after a discussion with the team at Nasan that a ML algorithm be only implemented on the high-frequency items as that would be a valuable contribution to the inventory management system already in place. Only the items in the high and very high categories in both in the segregations were combined into a separate list for this implementation.

2.2 Analysis and Implementation of Machine Learning

The data is time-based sequential data. There are 2 main types of machine learning algorithms. Classification problems and Regression problems.

Classification problems are used when we have a fixed number of categories and the inputted data needs to be sorted into one or more categories by the algorithm. This algorithm is normally used when a data sorting problem statement is presented. For example, rating if a movie was good or bad based on reviews from people.

Regression algorithms are algorithms that predict the output value provided the input features. These algorithms are used to map a direct or indirect relationship between the input features and the output features. This relationship helps us predict the output variables with the input variables accurately and with high degree of precision.

This project will be completed using a Regression algorithm. In this project we have time and transactions as the input features and the stock quantity as the output variable. The goal of this project is to predict the ideal stock quantity provided the time and the transactions in the past. There are many different algorithms in regression, for this project I explored and implemented 3 most commonly used regression algorithms in the industry.

2.2.1 Choosing Regression Algorithm

There are many different types of regression algorithms. In this use-case I experimented with 3 regression algorithms. The algorithm is:

- Linear Regression
- Ridge Regression
- Lasso Regression

These were chosen because these are used widely in the industry and have a stable analytical output that can be measured easily. Below is the working of each of these algorithms with their ideal use-cases to highlight their unique features and application

2.2.1.1 Linear Regression

Linear Regression is a basic supervised learning algorithm that aims to establish a linear relationship between input features and the target variable. It is used for predicting a continuous numerical output by minimizing the sum of squared differences between predicted and actual values.

Linear regression derives a linear equation between the input features and the output variable. Below is the generic formulae used by linear regression to establish a relationship.

$$Y = \beta + \alpha X + \varepsilon$$

In the above formulae <Y> is the output variable, <X> is the input feature. In this project we only have one input feature of time so it is a single linear regression formulae mentioned above. The <β> is the intercept and <α> is the coefficient for the feature <X>. Finally <ε> represents the residual error in the algorithm.

This algorithm is best implied when the relation between the independent variables and the dependent variables is linear and can be presented with minimum error.

2.2.1.1 Ridge Regression

Ridge Regression is a regularization technique applied to Linear Regression to prevent overfitting. It adds a penalty term proportional to the sum of squared coefficients to the loss function, constraining their magnitude. This results in a more stable model by reducing the impact of irrelevant features.

$$SSE_{L_2} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{P} {\beta_j}^2$$

The above equation represents the Ridge regression algorithm. In this algorithm, the left-hand term is the standard regression equation. But the right-hand term is added to counter over-fitting as mentioned above. This term takes <β> squared and adds it after which it is multiplied by <λ> as the standardizing term. By adding this extra correction to the standard regression equation we ensure a better overall result than a standard regression algorithm.

2.2.1.1 Lasso Regression

Lasso Regression is similar to Ridge Regression but employs L1 regularization. It adds a penalty proportional to the absolute value of coefficients and performs feature selection by pushing some coefficients to exactly zero. Lasso Regression not only helps prevent overfitting but also reduces the impact of irrelevant features.

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 + \lambda \sum_{j=1}^{p}\left|\hat{\beta}_j\right|$$

As we can see the formulae looks very similar to Ridge regression in implementation. The only difference between Ridge regression and Lasso regression is the standardization procedure used. Ridge regression uses the L2 method to standardize while Lasso employes the L1 standardization.

Each of these algorithms caters to specific tasks and addresses various challenges, making them valuable tools in the field of machine learning. These 3 algorithms were used to find the most optimum algorithm to be used in this test case.

2.2.2 Implementing Regression Algorithms

2.2.2.1 Train Test Split

Before implementation, the data was split into the train and test data sets. A **70%-30%** split was taken for the train test split. Normally the train-test split is done using a random state order, but since the data is time-series data the first 70% of the data was taken as the training dataset and the last 30% of the data was taken as the validation data set. This helps the algorithm learn the overall trend in the data and lets us judge the algorithm in a more accurate manner.

2.2.2.2 Implementation

The algorithms were implemented in Python programming language using the **SKLEARN library** for the models and for the statistics of the trained models.

Linear Regression model was trained on the vanilla model provided by SKLEARN and no parameters were set in this particular training. For Ridge regression and Lasso Regression, the cross-validation function in the Sklearn library was used to find the optimum Lambda value to be used in the algorithms. Once the ideal lambda value was identified they were trained on the training data set.

2.2.2.3 Validation and Scoring

Once the models were optimized and trained they were used to predict the validation data set and statistics were used to judge the efficiency of the trained model.

For validation, 5 items were taken at random from the Very high-frequency items list. Each algorithm was trained and optimized on the 5 data sets independently. 5 epochs/iterations were conducted for each model on a single data set. This iteration was used to average out any badly optimized models.

The **R-squared score, Root Mean Squared Error, and Mean Absolute Error** were used as verification factors for the trained models. The data was averaged over the 5 datasets and the final output was used for the selection of the best algorithm out of 3 tested for this project.

2.2.3 Dealing with Variation and Fluctuations in the Data

As we can see there is a lot of variation and sudden spikes in the data. These spikes are caused because the purchase of stock is performed in bulk by the company. Unluckyly there is a no direct method to deal with these spikes as they are one of the key features of the data. Thus the spikes and variation is left to the ML algorithm to deal with. The aim is to generate a trend line for the ideal stock price thus we ignore these high spikes and concentrate on generating a more linear graph predicting the future useage of that particular item.

2.3 Deliverables and Application

Once the analysis it was finalized, the team at Nasan and I had a discussion on the format in which they would like to have this system deployed.

All the analysis and processing were completed in a python code for this project. The code uses multiple libraries and version specific commands. Because of this sharing the python code is not an optimum or a desired deliverable for this project. Upon discussion with the company, it was decided to make a simplistic GUI application.

The application had the following requirements:

- Data generated by the current inventory management system should be accepted as input without any modifications to the file.
- A button to generate the dead inventory reports.
- A button to generate zero inventory reports.
- The application will display a list of all the items it has run the ML algorithm on for prediction.
- Once the item is selected the user can enter a desired date and the ideal stock quantity for the selected item at the entered date be calculated and displayed to the user.
- A button to generate a report for ideal stock quantities for all the items at a given date.
- Simple to use and understandable UI.

An application with the above specification was built using python. The python code was converted to an executable file (*.exe) so that any windows system can execute it without worrying about the python installation.

2.4 (FUTURE SCOPE) AutoRegressive Integrated Moving Average

In a previous TA note, it was mentioned that I explore the ARIMA and SARIMA models since they are used in the industry for time-series data prediction. To this I experimented with the implementation of ARIMA and tuning it for this data.

Due to the deadline set by the company for the completion of the project, I was not able to complete the tuning and result generation from the ARIMA model in time. I will present my findings and work done below but this has not been considered in the final deliverable to the company.

2.4.1 Introduction to ARIMA and SARIMA Models

ARIMA (AutoRegressive Integrated Moving Average) is a time series forecasting model that captures time-dependent patterns by combining AutoRegressive (AR), Integrated (I), and Moving Average (MA) components. It's effective for predicting data with no clear seasonality. In contrast, SARIMA extends ARIMA by introducing Seasonal AutoRegressive (SAR) and Seasonal Moving Average (SMA) terms, making it suitable for time series data with distinct seasonal fluctuations. SARIMA is the choice for datasets with regular seasonality, while ARIMA is preferred when seasonality is less pronounced or removed through differencing.

The data for this project is not seasonal data. The data does display sudden peaks in the stock quantity due to the large purchase orders but these orders are executed on a need basis and not on a seasonal basis. Thus I think ARIMA is the ideal model for this project when compared with SARIMA.

2.4.2 ARIMA Equation and Tuning

$$\mathbf{Y(t) = c + \phi_1 Y(t-1) + \phi_2 Y(t-2) + \ldots + \phi_p Y(t-p) - \theta_1 \varepsilon(t-1) - \theta_2 \varepsilon(t-2) - \ldots - \theta q \varepsilon(t-q) + \varepsilon(t)}$$

Where:

- **Y(t)** is the value of the time series at time t.
- **c** is a constant (the intercept term).
- **p** represents the order of the AutoRegressive (AR) component, which specifies how many lagged values of the time series are used in the model.
- **d** represents the degree of differencing applied to make the time series stationary (i.e., differencing order).
- **q** represents the order of the Moving Average (MA) component, which specifies how many past error terms (ε) are used in the model.
- $\phi_1, \phi_2, \ldots, \phi_p$ are the AR coefficients.
- $\theta_1, \theta_2, \ldots, \theta q$ are the MA coefficients.
- **ε(t)** is the white noise error term at time t.

ARIMA model tuning is done by finding the optimum p,d,q values for the data. The tuning is performed similar to the λ optimization in regression models. A grid of p,d,q values is generated and a fix scoring metric is decided. Using functions like GridSearchCV we can parse the generated matrix

and evaluate all the combinations against the scoring metric. We find the optimum combination of the 3 variables and use it as the final solution.

This optimization process is extremely computationally intensive since a 3d matrix is required to get a more completely optimized solution.

2.4.3 Implementation and Results

ARIMA was implemented in Python using the "statsmodels.tsa" library. Since the parameters were not optimized a random set of parameters was set as (p = 5, d = 1, q = 0). Please note that these parameters were taken at random and do not denote a logical selection process.

Similar to the regression models a 70-30 train test split was used on the data. The model was trained on the training data and used for prediction on the validation data set. Below are the scoring metrics findings from the implementation.

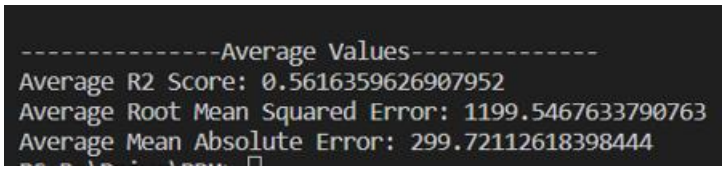


Figure 2: Results from ARIMA algorithm

Below is the produced graph when the model predicted the values:

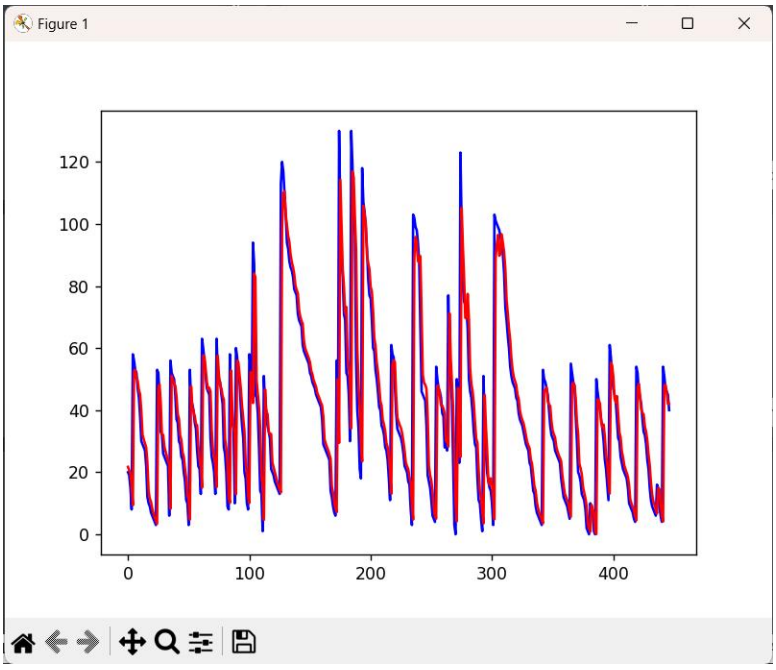


Figure 3: Prediction Graph generated by ARIMA algorithm

From the above metrics and graph we can note that the algorithm has been over-fitted for this data and will provide highly volatile and inaccurate predictions in a larger time-frame. We can conclude that the optimization is required to make the algorithm more useable in the real world. But this

experiment shows us that ARIMA when implemented correctly shows great promise for these kind of problem statements.

3. Results

3.1 Dead Inventory and Zero Inventory Reports

One of the first analytical observations was that many items had their last transactions from a couple of years or a year old. An item which has not been used for a pre-defined set of time is called dead inventory.

Dead inventory, also known as obsolete or stagnant inventory, is detrimental to businesses for several reasons. Not only does it occupy precious storage space, tie up capital, and demand management attention, but it also has a negative impact on cash flow, profitability, and operational efficiency. The presence of dead inventory results in increased carrying costs, write-offs, and a need for clearance sales, all of which can significantly erode profit margins. Furthermore, dead inventory indicates missed market trends, poor demand forecasting, and inefficiencies in inventory management practices. To maintain a healthy supply chain, optimize resources, and remain competitive, it is crucial to address and minimize dead inventory.

Along side dead inventory there was one more observation, there were items in the dead inventory that had a stock quantity of '0'. Such items are classified as zero inventory. If an item is a zero-inventory item, it means that the item has not been used for the pre-define time and the stock quantity has not been more than 0. This leads us to the conclusion that, this item is not required and since it has not stock, there is no need for that item to tracked by the inventory management software. It is important to generate a zero-inventory report periodically so that such items can be removed from the inventory system and the over-head can be reduced on the system. Such items are known to cause unnecessary burden on the inventory management system and often skew the results in an inventory audit, which is not desired.

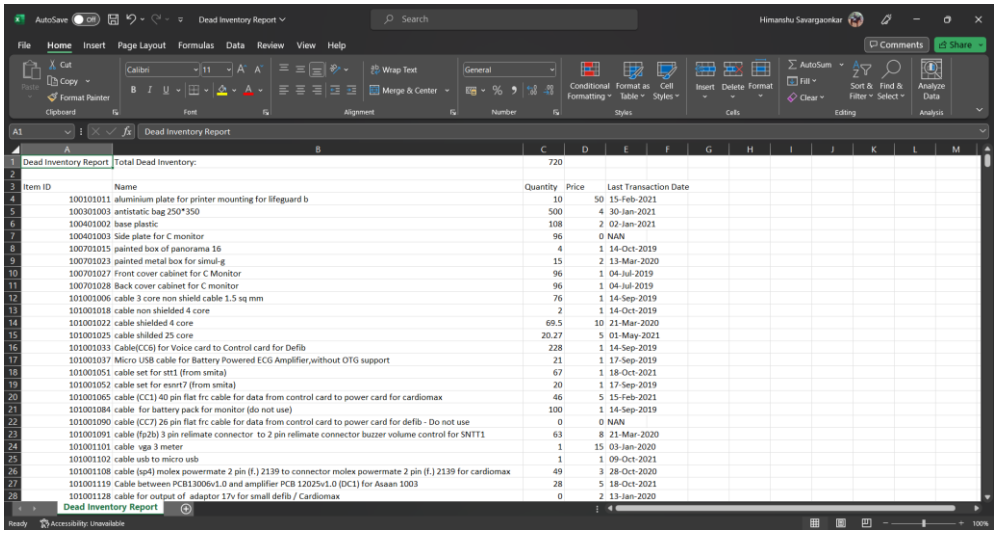
After discussion with the team, it was finalized that an item be considered dead inventory if there have not been any transactions in the item in the past 1 year. It was also decided that dead inventory and zero inventory reports be delivered as csv files. The data format followed for these reports are:

- Item ID
- Name
- Quantity
- Price
- Last Transaction Date

For the zero inventory report the following data format was followed:

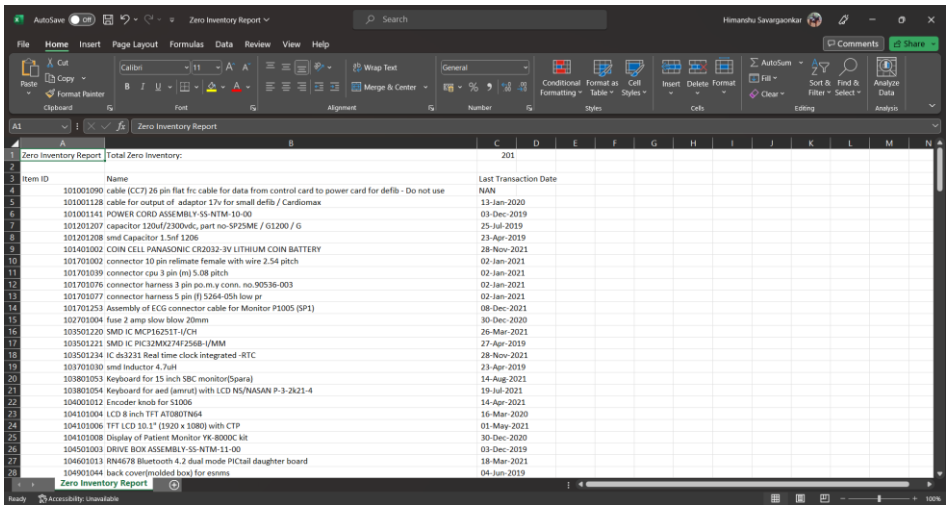
- Item ID
- Name
- Last Transaction Date

A excel format was selected as the company is used to handling data in that format. Additionally, by adding filters to the generated report the dead inventory duration can be easily modified in the report. This feature future-proofs these reports.



Dead Inventory Report	Total Dead Inventory:	720		
Item ID	Name	Quantity	Price	Last Transaction Date
100101011	aluminium plate for printer mounting for lifeguard b	10	50	15-Feb-2021
100301003	antistatic bag 250*350	500	4	30-Jan-2021
100401002	base plastic	108	2	02-Jan-2021
100401003	Side plate for C monitor	96	0	NAN
100701015	painted box of panorama 16	4	1	14-Oct-2019
100701023	painted metal box for simul-g	15	2	13-Mar-2020
100701027	Front cover cabinet for C Monitor	96	1	04-Jul-2019
100701028	Back cover cabinet for C monitor	96	1	04-Jul-2019
101001006	cable 3 core non shield cable 1.5 sq mm	76	1	14-Sep-2019
101001018	cable non shielded 4 core	2	1	14-Oct-2019
101001022	cable shielded 4 core	69.5	10	21-Mar-2020
101001025	cable shilded 25 core	20.27	5	01-May-2021
101001033	Cable(CC6) for Voice card to Control card for Defib	228	1	14-Sep-2019
101001037	Micro USB cable for Battery Powered ECG Amplifier,without OTG support	21	1	17-Sep-2019
101001051	cable set for stt1 (from smita)	67	1	18-Oct-2021
101001052	cable set for esnrt7 (from smita)	20	1	17-Sep-2019
101001065	cable (CC1) 40 pin flat frc cable for data from control card to power card for cardiomax	46	5	15-Feb-2021
101001084	cable for battery pack for monitor (do not use)	100	1	14-Sep-2019
101001090	cable (CC7) 26 pin flat frc cable for data from control card to power card for defib - Do not use	0	0	NAN
101001091	cable (fp2b) 3 pin relimate connector to 2 pin relimate connector buzzer volume control for SNTT1	63	8	21-Mar-2020
101001101	cable vga 3 meter	1	15	03-Jan-2020
101001102	cable usb to micro usb	1	1	09-Oct-2021
101001108	cable (sp4) molex powermate 2 pin (f.) 2139 to connector molex powermate 2 pin (f.) 2139 for cardiomax	49	3	28-Oct-2020
101001119	Cable between PCB13006v1.0 and amplifier PCB 12025v1.0 (DC1) for Asaan 1003	28	5	18-Oct-2021
101001128	cable for output of adaptor 17v for small defib / Cardiomax	0	2	13-Jan-2020

Figure 4: Dead Inventory Report



Zero Inventory Report	Total Zero Inventory:	201
Item ID	Name	Last Transaction Date
101001090	cable (CC7) 26 pin flat frc cable for data from control card to power card for defib - Do not use	NAN
101001128	cable for output of adaptor 17v for small defib / Cardiomax	13-Jan-2020
101001141	POWER CORD ASSEMBLY-SS-NTM-10-00	03-Dec-2019
101201207	capacitor 120uf/2300vdc, part no-SP25ME / G1200 / G	25-Jul-2019
101201208	smd Capacitor 1.5nf 1206	23-Apr-2019
101401002	COIN CELL PANASONIC CR2032-3V LITHIUM COIN BATTERY	28-Nov-2021
101701002	connector 10 pin relimate female with wire 2.54 pitch	02-Jan-2021
101701039	connector cpu 3 pin (m) 5.08 pitch	02-Jan-2021
101701076	connector harness 3 pin po.m.y conn. no.90536-003	02-Jan-2021
101701077	connector harness 5 pin (f) 5264-05h low pr	02-Jan-2021
101701253	Assembly of ECG connector cable for Monitor P1005 (SP1)	08-Dec-2021
102701004	fuse 2 amp slow blow 20mm	30-Dec-2020
103501220	SMD IC MCP16251T-I/CH	26-Mar-2021
103501221	SMD IC PIC32MX274F256B-I/MM	27-Apr-2019
103501234	IC ds3231 Real time clock integrated -RTC	28-Nov-2021
103701030	smd Inductor 4.7uH	23-Apr-2019
103801053	Keyboard for 15 inch SBC monitor(Spara)	14-Aug-2021
103801054	Keyboard for aed (amrut) with LCD NS/NASAN P-3-2k21-4	19-Jul-2021
104001012	Encoder knob for S1006	14-Apr-2021
104101004	LCD 8 inch TFT AT080TN64	16-Mar-2020
104101006	TFT LCD 10.1" (1920 x 1080) with CTP	01-May-2021
104101008	Display of Patient Monitor YK-8000C kit	30-Dec-2020
104501003	DRIVE BOX ASSEMBLY-SS-NTM-11-00	03-Dec-2019
104601013	RN4678 Bluetooth 4.2 dual mode PICtail daughter board	18-Mar-2021
104901044	back cover(molded box) for esnms	04-Jun-2019

Figure 5: Zero Inventory Report

We can note from the above reports that there are a total of 720 items that can be classified as dead inventory. Additionally, out of the 720 items 201 items are classified as zero inventory. These items need to be scanned by the company's inventory team and appropriate action needs to be taken on them to minimize overhead stock and prices in the inventory system.

3.2 ML Analysis Results

3.2.1 Train-Validate Data Sets

The first task of any ML algorithm application is to perform the train test split. As mentioned in the method section rather than performing a random state split I choose to have the older 70% data be the training set and the latest 30% data be the validation set. This was done to check the prediction capabilities of the algorithms. Below is an example plot of the training and validation data sets used to train the models.

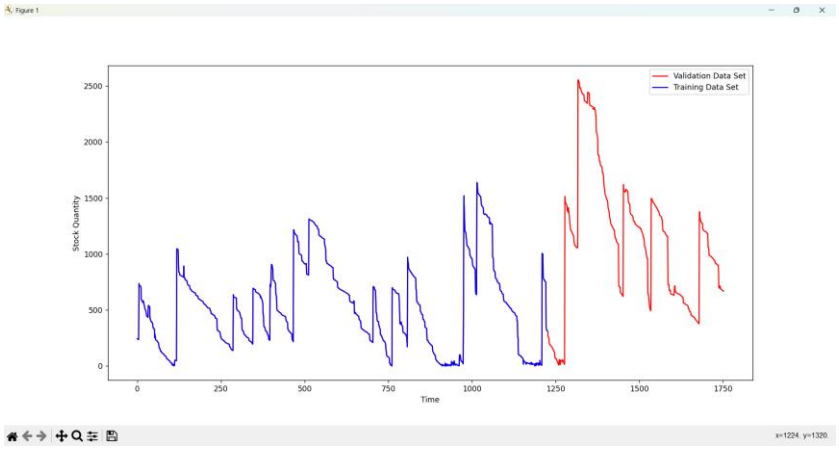


Figure 6: Train - Validation Data Sets Graphichal Representation

3.2.2 ML Algorithm Scoring Metrics

To measure the success and accuracy of the models trained it is important to use proper metrics to judge them. For this project 3 metrics were chosen to judge the efficiency of the algorithms. The metrics used are explained and justified below in brief.

3.2.2.1 R- Squared(R2) Test

The R-squared (R2) test in regression is a statistical measure used to assess the goodness of fit of a regression model. It quantifies the proportion of the variance in the dependent variable that can be explained by the independent variables included in the model. R2 is a value between 0 and 1, with higher values indicating that a larger percentage of the variability in the dependent variable is accounted for by the independent variables. In essence, it tells us how well the model fits the observed data points. R2 is widely used in regression analysis because it provides valuable insights into the model's predictive power and helps researchers and analysts determine whether the chosen independent variables are effective in explaining the variation in the dependent variable. A high R2 suggests that the model is a good fit for the data, while a low R2 suggests that the model may need refinement or that other variables should be considered.

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

3.2.2.2 Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE) is a critical metric in regression analysis that assesses the accuracy of a predictive model. It quantifies the average magnitude of errors between the predicted values and the actual values in a dataset. RMSE provides a clear and interpretable measure of how well the model's predictions align with the observed data. Lower RMSE values indicate that the model's predictions are closer to the actual data points, reflecting higher prediction accuracy. Conversely, higher RMSE values indicate larger prediction errors and suggest that the model may need improvement. RMSE is widely used in various fields, such as finance, engineering, and machine learning, to evaluate the quality of predictive models and guide model selection and fine-tuning.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

3.2.2.3 Mean Absolute Error (MAE)

Mean Absolute Error (MAE) is another important metric in regression analysis that evaluates the accuracy of a predictive model. It calculates the average absolute differences between the predicted values and the actual values in a dataset. Unlike RMSE, which squares the errors, MAE treats all errors equally, making it less sensitive to outliers.

MAE provides a straightforward and easily interpretable measure of the model's performance. A lower MAE indicates that the model's predictions are closer to the actual data points, implying better accuracy. Conversely, a higher MAE suggests that the model's predictions have larger absolute errors and may require refinement. MAE is commonly used in fields like economics, statistics, and machine learning to assess model accuracy and compare different models or algorithms.

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|$$

3.2.3 Observed Results

As mentioned in the method section the models were trained on 5 different datasets. This was done to ensure that the models didn’t favour a particular dataset and were not over-fitted for that data set.

All the 3 metrics were calculated for the 5 data sets. The average was used to make the final selection between the 3 regression algorithms explored in this project. Below we can see the output of the benchmarking code.

```
--------------Linear Regression------------
Average R2 Score: 0.4452502589277152
Average Root Mean Squared Error: 994.1037338300455
Average Mean Absolute Error: 788.4802033144247


--------------Ridge Regression-------------
Average R2 Score: 0.41900946329203687
Average Root Mean Squared Error: 994.1037338300455
Average Mean Absolute Error: 808.4588187132207


--------------Lasso Regression-------------
Average R2 Score: 0.6525041605659274
Average Root Mean Squared Error: 994.1037338300455
Average Mean Absolute Error: 803.2882961443607
```

Figure 7: Regression Algorithms Scoring Metrics

From the data presented, we can not that Lasso Regression is the only algorithm to provide a value of the R-squared test greater than (0.5). When comparing the RMSE all the algorithms provide similar numbers, the numbers may seem high for the error but since the data is processing nodes in the thousands these errors are expected. For the MAE we can again note that Lasso Regression provides the least error though with not a very big margin.

From the discussion above we can conclude that Lasso Regression is the ideal solution for this project and that will be employed in the final version of deliverables

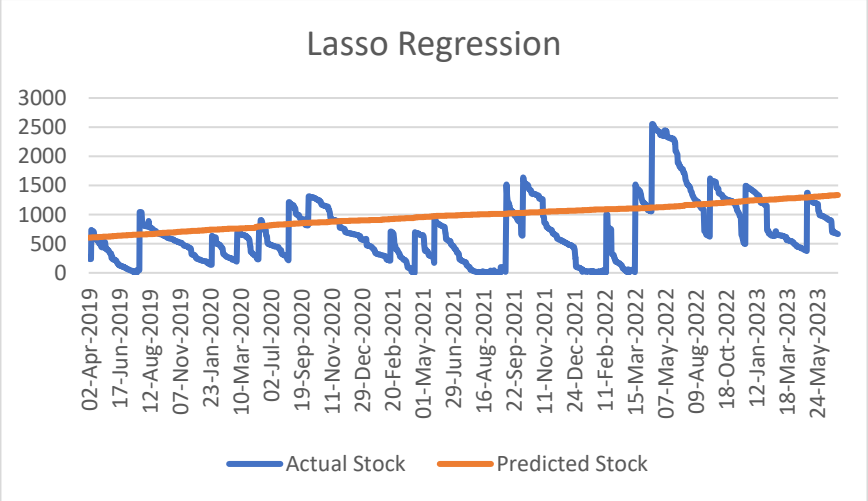


Figure 8: Lasso Regression Prediction

3.2.4 Algorithm Optimization

Both the algorithms Ridge regression and Lasso regression have a standardization variable λ. When training a model using these algorithms it is very important to tune this variable to attain the most optimum result.

For this project λ values of [2,1,0.1,0.01,0.001] were tested. Out of these 5 values the best λ was chosen by using the Grid Search function in Sklearn library. Since all the models trained had their own optimized λ value a output result cannot be displayed in this section but a screen-grab of the code section is attached below to show the validity of the optimization process undertaken for this project.

```
130  lasso = Lasso()
131
132  param_grid = {'alpha': [2,1,0.1,0.01,0.001]}
133
134  # Using GridSearchCV to find the best alpha value through cross-validation
135  grid_search = GridSearchCV(lasso, param_grid, cv=5)
136  grid_search.fit(X_train, y_train)
137  best_alpha = grid_search.best_params_['alpha']
138
139  lasso_optimized = Lasso(alpha=best_alpha)
140
141  # Fit the model to the entire training data
142  lasso_optimized.fit(X_train, y_train)
```

Figure 9: Code of λ tuning for Lasso Regression

3.3 Algorithm Finalization and Implementation

In Section 5.2, I have presented the results from the various algorithms that I tested on the provided data. It was noted that Lasso regression provides the most accurate results out of all the algorithms.

As the final step of the project, Lasso regression was applied to all the High and Very High-frequency items. A few of the graphs obtained from other items have been presented below:

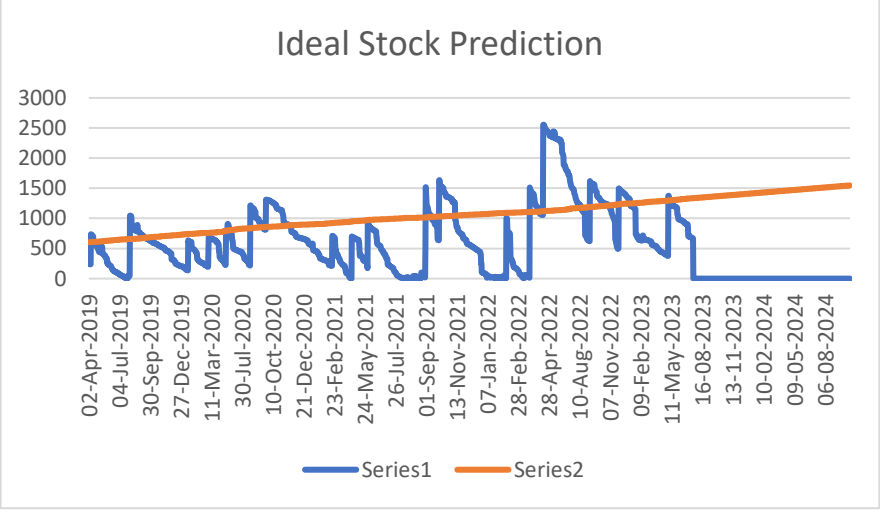
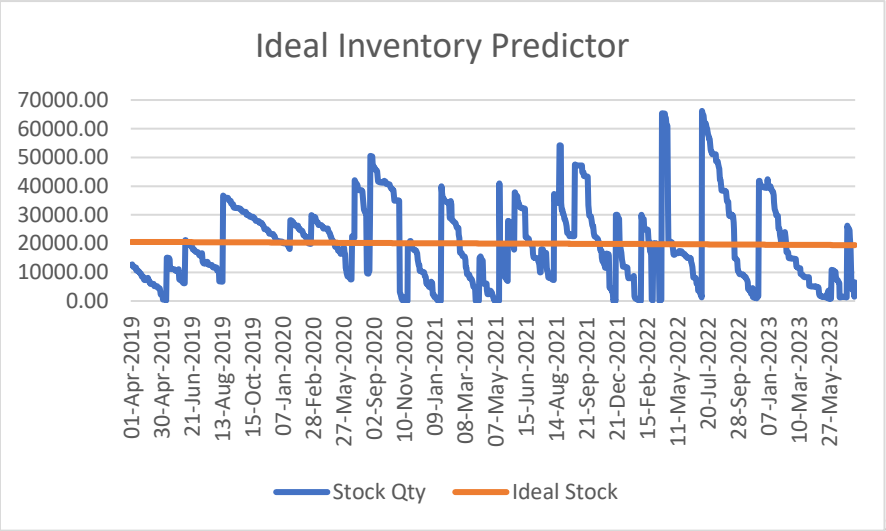


Figure 10: Ideal Inventory Example 1



s

Figure 11: Ideal Inventory Example 2

The above 2 examples were taken to display how the algorithm copes with the different types of data. The data on which the algorithm was tuned has a very clear graph and is human-readable, but these examples have such a high number of transactions that it is not possible for a human to comprehend this data.

Additionally, we can also observe that the algorithm shows the trend of the ideal stock quantity. In the first example, we can note that the ideal stock quantity will remain the same for a long period of time with a slight predicted increase in the future. But in the second graph, we note that there is a decrease in usage of this particular item. This also means that the ideal stock quantity is decreasing, and we can predict that in due course or in 5-10 years this product will be obsolete and will not be needed for production at the company.

3.4 Deliverable Application

As discussed in Section 5, after discussion with the team it was decided that the most productive and practical implementation of this project can be achieved through a GUI which has all the functionality built into it.

Based on it, I designed a GUI using Tkinter library in python. The workflow for the application is as follows:

- Insert the File path and click “Load Data”.
- We can generate the dead inventory and zero inventory reports.
- Select the item ID from the drop-down menu.
- Enter Date for which you want ideal stock value and press “Ideal Stock”.
- If you want the ideal stock value for all items on a given date, enter the desired date and press, “Generate All Stock”.

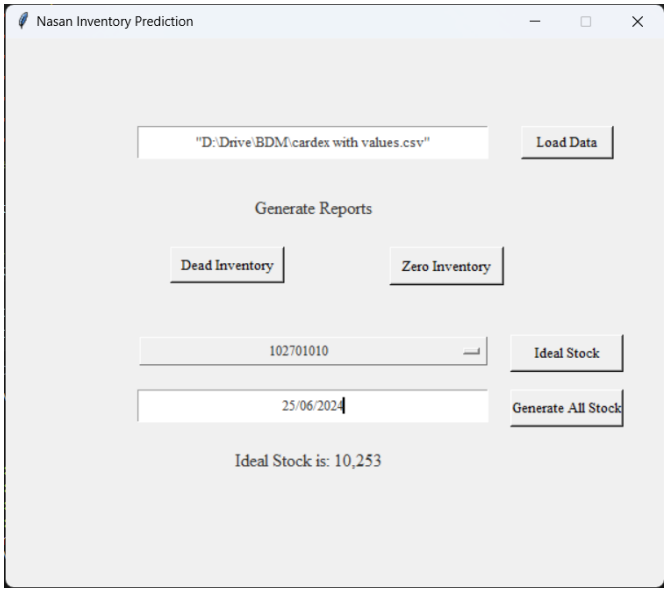


Figure 11: Application with Prediction

4. Recommendations and Conclusions

This project has granted me with a deep insight into inventory management for a medium scale manufacturing unit. While doing the analysis I have made some observations and reached some conclusions. These findings are presented in this section, the aim is that these findings will be beneficial to the company to make a positive step towards an improved inventory management system.

4.1 Types of Inventory Items

In the preliminary analysis it was observed that majority of the items are located in the low transaction number pools while only a few were present in the high frequency sections. I believe that it is important to understand the difference between these categories and that they need to be treated differently from an inventory management point of view. The low transaction items need an independent analysis based on their usage and if these items can be replaced by other pre-existing items. Items with a high frequency need to be monitored from a purely data point of view, since the number of transactions is high it is not possible for a human to get an idea about the trends by looking at the transactions.

4.2 Dead Inventory and Zero Inventory

I would strongly suggest deletion of all the zero inventory from the current inventory management system. At the same time, I would recommend maintaining a separate database of discontinued inventory items for future reference from a price and vendor point of view.

With regard to the present dead inventory, the number of items is extremely high to be ignored. There needs to be an internal inventory audit to figure out how the items are linked to the products and if selling/disposing them is a right decision. Unluckily this has to completed manually iterating through each of the dead inventory items due to the complexity of the manufacturing process and bill of materials. If a decision is made to get rid of a few items, they need to be removed from the inventory management system, along with physically segregating them in the inventory room for easier disposal. The activity should be carried out independent from the inventory management.

4.3 Using ML in the inventory management.

The main goal of this project was to deliver a machine learning-based solution to improve the present inventory management system. The main use of the program delivered as part of this project is to be used in planning meetings. When a business plan has been developed, the predicted numbers can be compared with the output from the program to cross-validate the predictions being established.

I would also suggest that the tool delivered as part of this project be used periodically and compared with the current stock present for all the items. This exercise will help identify over-stocking and more importantly, under-stocked items that need to be purchased on a high priority. This tool presents a new way to process the data that is readily available, the company should deploy this tool to the inventory management department and let the team experiment and help integrate the tool into the inventory management toolchain. Once completely integrated, the tool will serve as a valuable resource to cross-check assumptions about the future but also to ensure the day-to-day functioning of the inventory department by raising red flags before a crisis arises.

5. Conclusion

The project conducted at Nasan Medical Electronics Pvt. Ltd. has been a very interesting project in improving inventory management through the implementation of advanced machine learning techniques. By effectively tackling the challenges associated with dead and zero inventory, the project has effectively demonstrated its ability to optimize resources. The project employed a two-stage approach, which involved collecting and preprocessing data, followed by the implementation of item-specific prediction algorithms. The integration of this model into a user-friendly application further solidified the project's practicality, providing the staff with timely and accurate insights to ensure optimal inventory levels. This project not only exemplifies the power of data-driven decision-making but also signifies the potential for innovation in the realm of inventory management, particularly within the dynamic medical equipment manufacturing sector.