Supply Chain and Manufacturing System Management Repot

Shiyu You

Objective

The objective of this project is to address real-world supply chain management problems that goes beyond textbook level curriculum, based on practical problems encountered by two companies: *Arconic Fastening Systems and Rings*, as well as *VTI of Texas.* The goal is to apply inventory management and forecasting concepts acquired from the course and translate the results of the methods into useful information for the companies to apply.

PART I: ARCONIC

Background Information

Arconic, previously known as Alcoa, is a mechanical and industrial engineering company specialized in manufacturing, and lightweight metals engineering, creating metals like aluminum, titanium, and nickel to be used across a variety of industries in aerospace, automotive, construction and many others. In this project, the team is tasked to solve a forecasting issue pertaining to their manufacturing site at Waco, TX.

Problem Statement

There is a current gap in knowledge within the current high level market forecast state of Arconic, specifically with truck build rates, upper management use, lack of correlation to products, and various customer demands. Our team wishes to develop 2 optimal short-term, and 2 optimal long-term forecasting models to upgrade the company to an ideal state which relates product part number to group level, has explicit inputs, a demand planner use, and an ability to react to unpredictable market swings.

Methods and Approaches

1.1 Data Preprocessing and Preparation

As a team, the decision was made to use the request date instead of the order date, promised shipment date, and actual shipment dates; since the request date is dictated by the demand of our customers, it is a more effective indicator of the demand rate of each product group. After a spreadsheet perusal, the project team noticed, and corrected apparent mistakes in the entries. For example, sales orders 259689 and 259512 had request dates that were at least 5 years before the order date, so the years for those request dates were manually adjusted to match the order shipment dates promised. After all the data preprocessing, the project team graphed the total cumulative sales from 2014-2017 to visualize possible

trends in the sales by item group. From **Figure 1** below, the project team saw that the pareto 80/20 would apply here, as approximately 80% of the total sales in quantity (not dollar value) comes from the top 20% of the item groups. Given the limited time and resources, the project team decided to tackle the top 20% item groups, which amounts to 16 item groups out of 79 total item groups. The Grand Total Cumulative Sales (on the right) is compared to the remaining 63 item groups (11.4%) and 16 item groups (88.6%). In the **Table 1**, the project team also shows the list of the top 16 item groups along with its sales from 2014 to 2017.

Table 1

Item Group (Product Family)	2014	2015	2016	2017	Grand Total
12300	2936877	63293903	44768500	52291720	163291000
12600	257250	61402350	37163836	46325737	145149173
11286	1052552	40861421	29969409	35463887	107347269
14388	390081	41925962	29180495	35643682	107140220
12610	948598	36110339	27496007	33932821	98487765
12310	1448367	35207632	25536387	30523947	92716333
12354	90000	24812875	14150718	16801474	55855067
11280	216000	17490187	17536800	17610947	52853934
14403	405890	15848417	10173520	13384450	39812277
14386	437940	8885817	6594516	8806924	24725197
14407		9220989	5658130	6270285	21149404
11281	19000	6165163	5322180	6899802	18406145
12630	268650	7058444	4327780	5689402	17344276
12353	10500	5461556	3104750	4188750	12765556
11285	337844	4838850	2629000	3867800	11673494
14405	57090	4702109	2843565	3742809	11345573

1.2 Modeling and Decision Criteria

The team decided to produce 2 short term models and 2 long term models using the following methods taught in the course. The reasoning being that different prediction models have different prediction power based on demand pattern and type. Each individual model was used to predict the demand on weekly, monthly and quarterly buckets in order to identify the best performing model within each time bucket. The historical data was then split between training data and test data. The training data was any data before 2018 while the test data was any data record between 1/1/2018 and 6/30/2018. The best weekly model was then chosen to predict a 6 week forecast, the best monthly model was used to produce a 3 month forecast, while the best quarterly model was used to predict a 2 year quarterly forecast and a 3 year yearly forecast. The criteria to judge models is based on mean average percentage error (MAPE) obtained from the prediction on our test data. The model with the smallest MAPE within their respective time buckets will be selected as the best model of the time bucket.

1.2.1 Exponential Smoothing

Exponential smoothing method was tested because it is memory based and has a tuning parameter that assigns a weight (lambda) to the most recent demand. When product group demands have high volatility, the exponential smoothing method comes in handy as the tuning parameter can assign a greater weight to more recent demand so that the procurement team can better respond to the recent change in demand. To optimize our exponential smoothing model, the GRG Nonlinear Simplex function on Excel was used to determine the lambda value that minimizes the MAPE within the product group of each time bucket. This is seen in **Table 2.**

Table 2

Optimized Lambda Using GRG												
Item	Monthly				Quarterly				Weekly			
Group	Lambda	MAPE	Max E	Min E	Lambda	MAPE	Max E	Min E	Lambda	MAPE	Max E	Min E
12300	0.3028622	26.5%	49.7%	0.0%	0.639957	34.3%	34.4%	34.2%	0.00065	92.0%	370.6%	0.0%
12600	0.1681275	24.1%	42.7%	2.5%	0.5071975	28.5%	28.9%	28.1%	0.022713	196.1%	2961.5%	6.9%
11286	0.2106593	18.1%	34.4%	10.5%	0.7191241	9.8%	19.5%	0.0%	0.003415	36.3%	77.7%	0.0%
14388	0.3602058	21.2%	57.9%	0.0%	1.3729688	26.8%	53.6%	0.0%	0.052766	28.6%	61.9%	0.0%
12610	0.0253363	27.7%	59.5%	0.0%	0.0867651	15.0%	30.1%	0.0%	0	42.9%	90.9%	0.1%
12310	0.1433703	19.6%	54.1%	1.1%	0.7353826	13.9%	27.7%	0.0%	0.001447	46.8%	105.6%	0.0%
12354	0.1069364	61.9%	175.4%	0.9%	0.7158198	25.6%	51.3%	0.0%	1.508475	151.5%	478.3%	0.0%
11280	0.2887112	27.5%	53.1%	0.0%	0.275359	38.9%	42.7%	35.0%	0.001736	77.3%	224.8%	0.0%
14403	0.212236	28.1%	37.3%	6.5%	1.3707703	18.0%	36.1%	0.0%	0.006165	46.0%	246.4%	13.8%
14386	0.1159211	38.9%	65.7%	0.0%	1.1043149	25.3%	50.7%	0.0%	1.02553	167.0%	2039.6%	0.0%
14407	1.4451464	73.4%	159.7%	0.0%	0	16.8%	23.5%	10.2%	0	267.0%	1186.5%	12.7%
11281	0.0102895	61.1%	80.1%	30.3%	0.5167902	22.9%	45.9%	0.0%	1.15865	263.6%	1719.6%	0.0%
12630	0.0333275	26.5%	68.6%	0.0%	0.2266803	17.7%	22.8%	12.5%	0	77.6%	100.0%	0.0%
12353	0.0322404	55.1%	148.7%	5.9%	0.0663563	21.9%	43.8%	0.0%	0.881576	172.1%	844.4%	0.0%
11285	0.837511	32.5%	56.0%	0.0%	2.1402133	16.8%	33.6%	0.0%	0.008944	65.6%	161.1%	0.0%
14405	0.0248215	20.7%	43.8%	0.0%	0	16.4%	26.9%	6.0%	0.006551	44.2%	188.0%	3.2%
Average	0.2698564	35.19%	74.17%	3.61%	0.654856	21.8%	35.7%	7.9%	0.292414	110.9%	678.6%	2.3%

1.2.2 One Step Ahead Moving Averages

One step moving average method was another memory based prediction model that we tested because it provided equal weight to recent demands. In product groups where volatility is high but relatively centered around the average, the moving average model performed the best. The tuning parameter in the moving average model was the number of data points taken into consideration when performing the model. The higher the number of points taken into consideration, the lower the weight assigned to each data point, hence the model's decrease in sensitivity to recent more recent demand changes. To test the optimum number of points needed for the moving average model, we compared the MAPE obtained from models using 3, 5 , 10 and 20 data points and the best model is shown as seen in **Table 3**

Table 3

Item Group	MA=15 Monthly			MA=5 Quarterly			MA=20 Weekly		
	MAPE	Max E	Min E	MAPE	Max E	Min E	MAPE	Max E	Min E
12300	31.4%	48.0%	1.8%	36.9%	40.7%	33.1%	209.8%	3298.4%	4.3%
12600	24.4%	45.5%	5.0%	28.3%	32.1%	24.5%	189.2%	2717.5%	5.1%
11286	18.3%	37.6%	3.6%	18.4%	25.0%	11.7%	54.5%	170.5%	3.5%
14388	30.3%	48.9%	12.4%	36.0%	36.6%	35.5%	29.4%	79.8%	2.3%
12610	39.8%	50.6%	20.4%	20.1%	26.4%	13.7%	70.3%	236.0%	2.2%
12310	19.0%	53.9%	1.0%	22.1%	30.0%	14.2%	60.5%	238.0%	2.8%
12354	61.8%	176.6%	3.7%	39.2%	51.6%	26.8%	504.6%	7860.4%	0.2%
11280	34.5%	55.4%	9.6%	35.0%	38.5%	31.4%	131.3%	1008.4%	7.8%
14403	28.2%	41.1%	10.0%	29.7%	30.1%	29.3%	51.1%	267.1%	0.2%
14386	38.5%	63.7%	1.1%	39.0%	43.4%	34.7%	293.3%	3983.6%	1.6%
14407	57.0%	115.1%	19.5%	21.1%	33.5%	8.8%	464.1%	2237.0%	0.7%
11281	64.6%	223.2%	10.2%	28.3%	43.5%	13.0%	233.2%	1061.0%	0.3%
12630	47.8%	67.3%	30.9%	10.1%	14.6%	5.6%	146.5%	728.6%	4.2%
12353	59.4%	171.5%	6.0%	19.3%	38.3%	0.3%	205.7%	1481.8%	2.6%
11285	41.7%	64.2%	20.7%	50.5%	54.8%	46.1%	77.2%	507.3%	1.5%
14405	21.6%	36.6%	4.4%	21.5%	26.6%	16.4%	53.8%	286.9%	2.1%
Average	38.64%	81.19%	10.02%	28.5%	35.4%	21.6%	173.4%	1635.1%	2.6%

1.2.4 Regression Analysis Using a Seasonal Factor

Regression analysis is effective especially when used for predictions involving large time buckets, as the increase in stability of the training data allows for the model to generate a linear relationship between the quantity and time. It's less suitable when used for short term predictions, however, as greater noise and a larger variance are introduced, negatively affecting the predictability of the model. Because our model showed annual seasonal trends, the addition of seasonal factors to our regression analysis allowed for our model to better predict the future demand annually.. Working with time series, the data had to be adjusted seasonally to uncover the underlying dynamics in the development of the investigated phenomena and allow for a direct comparison of their development in different seasons. A trend exists, as there was seen to be a long-term increase over the course of many periods, and a subsequent decrease in the data for different items. A seasonal pattern occurs when a time series is affected by seasonal factors. In this case, we used the quarter feature as the seasonal factor, which can be coded using corresponding dummy variables. For quarterly data, we needed to use 3 dummy variables, d1, d2, d3. The coefficient of the dummy variables associated with Quarter 1 will measure the effect of this quarter on the forecast variable, and will be compared to the effect of that of the dummy variables associated with Quarter 4.

1.2.5 ARIMA Models

Autoregressive Integrated Moving Average (ARIMA) models provide another approach to time series forecasting, which aim to describe autocorrelations present within the data. In an ARIMA model, 3 parameters exist which serve as a guide in modelling major aspects of a time series, ie. seasonality, trend and noise; these parameters are labeled p,d, and q, respectively. We predicted the demand on a weekly, monthly and quarterly basis, and set d = 1 and q = 1, which meant that the next demand would be influenced by the last week, month or quarter, assuming no noise. The tuning parameter in our ARIMA model was p, which was decided by the partial autocorrelation function plot with the first 50 lags and lowest MAPE. The final result with MAPE can be seen below in **Table 4**.

Table 4

	MAPE(%)								
	Weekly			Monthly			Quarterly		
Item Group	P=10	p=11	p=12	p=4	p=5	p=6	p=1	p=2	p=3
12300	69	68.71	69.53	48.98	44.71	41.74	45.59	45.53	55.1
12600	571.38	566.83	548.09	66.82	69.19	63.69	50.74	51.95	31.85
11286	30.24	31.34	33.61	23.05	23.14	20.42	20.56	19.53	23.33
14388	61.47	62.79	55.42	61.64	68.55	67.16	48.42	49.98	55.94
12610	56.1	53.9	50.02	44.49	35.71	46.64	13.32	14.83	18.13
12310	101.7	102.28	101.08	41.47	41.64	41.03	27.88	28.9	43.57
12354	72.76	76.63	89.95	64.8	76.45	76.7	59.31	54.37	35.87
11280	188.76	204.47	202.79	67.01	66.53	45.22	71.16	27.7	22.28
14403	73.87	73.61	73.76	57.41	57.77	55.69	39.02	34.67	35.61
14386	72.03	69.9	72.84	57.6	59.68	52.78	53.42	55.12	53.99
14407	542.24	504.74	493.33	44.35	45.11	44.37	35.19	34.99	32.82
11281	115.49	110.47	86.49	58.04	55.45	57.58	49.35	37.25	43.9
12630	56.5	48.43	48.36	62.33	70.3	77.78	78.89	82.49	108.31
12353	497.18	488	433.55	70.4	68.95	80.69	50.02	49.68	56.05
11285	48.79	43.82	40.95	53.15	51.18	50.68	61.69	61.09	75.43
14405	173.6	164.84	164.69	22.11	22.66	20.5	11.82	14.21	16.42
average	170.6944	166.9225	160.2788	52.72813	53.56375	52.66688	44.77375	41.39313	44.2875

1.3 Model Comparison and Selection

Table 5 - Average Test MAPE for forecast between 1/1/2018 and 6/30/2018

Method\Time Bucket	**Weekly**	**Monthly**	**Quarterly**
Exponential Smoothing	110.9%	35.19%	21.8%
Moving Average	173.4%	38.64%	28.5%
Regression w/ seasonal factors	-	35.39%	35.0%
ARIMA w/ seasonal factors	160.2%	52.7%	41.4%

Based on the results obtained from the table above, exponential smoothing method will be used for the short term 6 week forecast and 3 month forecast while the regression analysis with a seasonal factor model will be used quarterly for the 2 and the 3 year forecasts. Even though regression w/ seasonal factors has higher error rate than exponential and moving average, it's not a one step prediction model and therefore is better for long term forecasts.

Results - Forecast Models

Table 6 - [6 Week Forecast (Week Ending)]

Product Group	1/5/2018	1/12/2018	1/19/2018	1/26/2018	1//2018	2/5/2018
12300	195305	195319	196021	198696	199474	199628
12600	1110955	1088709	1115306	1126781	1120199	1095420
11286	542936	543809	543645	545298	550746	550594
14388	794260	802493	845894	835739	865348	865440
12610	394400	394400	394400	394400	394400	394400
12310	439552	439265	442450	443744	443933	444038
12354	86658	295343	170075	2272399	0	159898
11280	98022	97996	98470	99147	99128	98991
14403	276075	275634	278694	278162	279205	277942
14386	71527	0	944765	285739	352898	301270
14407	85105	85105	85105	85105	85105	85105
11281	0	300423	118025	13718	48804	293506
12630	14000	14000	14000	14000	14000	14000
12353	58066	31208	137255	98902	56673	12001
11285	85136	85072	85957	85868	85902	85134
14405	73655	73442	73436	73684	73540	73446

Table 7 - [3 Month Forecast]

Product Group	January	February	March
12300	5259998	4462713	3421408
12600	4881477	4303164	3794971
11286	3817539	3887680	3505106
14388	3398777	2839310	2207116

12610	2274654	2261633	2236275
12310	3006433	2922700	2650683
12354	1818910	1710304	1553610
11280	1787587	1626375	1498772
14403	1397915	1381817	1275970
14386	750791	682004	603006
14407	906964	193909	0
11281	439909	449845	457239
12630	366145	370440	362227
12353	303409	303266	303412
11285	580557	569622	581245
14405	305106	304834	303074

Table 8 - [2 Year Forecast by Quarter]

PG\Q	Q1-19	Q2-19	Q3-19	Q4-19	Q1-20	Q2-20	Q3-20	Q4-20
12300	18916795	18589441	17619920	12939473	19875337	19547983	18578461	13898015
12600	15006925	15704312	14817520	11818465	15497211	16194598	15307806	12308751
11286	10515995	11599279	11374097	10339653	11078334	12161618	11936437	10901992
14388	12221110	11903363	9654643	9214009	12698313	12380566	10131846	9691212
12610	9630877	8415009	9498968	8520610	9870089	8654221	9738179	8759822
12310	9652524	8747658	8852862	8245032	9974634	9069768	9174973	8567142
12354	6010228	6204973	5205824	4399532	6152500	6347245	5348097	4541804
11280	5724957	7816566	6397860	4173723	6219001	8310610	6891904	4667767
14403	4067803	4557675	4011924	3660667	4274199	4764071	4218320	3867063
14386	3119665	3011662	2334857	2270262	3316731	3208728	2531922	2467328
14407	2130720	1974088	1695972	1636594	2122904	1966272	1688156	1628778
11281	3286480	3447971	4075839	3385465	3900666	4062157	4690025	3999651
12630	1527841	1383730	1295006	972032	1472233	1328122	1239398	916424
12353	733325	725748	516214	610523	592474	584897	375363	469672

11285	1754473	2156275	2230984	1635540	2042239	2444041	2518750	1923306
14405	903777	1103323	913684	912983	897106	1096652	907013	906312

Table 9 - [3 Year Forecast]

Product Group\Year	2019	2020	2021
12300	68065629	71899796	75733963
12600	57347222	59308365	61269509
11286	43829023	46078382	48327740
14388	42993126	44901936	46810746
12610	36065464	37022311	37979157
12310	35498076	36786517	38074959
12354	21820557	22389647	22958737
11280	24113106	26089282	28065459
14403	16298070	17123653	17949237
14386	10736446	11524709	12312972
14407	7437374	7406111	7374847
11281	14195755	16652498	19109242
12630	5178609	4956177	4733745
12353	2585809	2022406	1459003
11285	7777272	8928336	10079400
14405	3833767	3807084	3780401

Recommendations

Recommendation 1:

It would be ideal for Arconic to incorporate a cost component associated with each forecast, ie. a cost of backordering and a variable cost of manufacturing. As the cost component assigns a weight to the model, the model could over forecast, to minimize the total cost incurred depending on the tradeoff between cost of backordering and the cost of manufacturing.

Recommendation 2:

Arconic should also look into obtaining forecasts from their key customers. Key customers usually place large orders and since such orders would amount to more than 50% of the total order quantity, it could

provide Arconic with an insight of its product demand in the future market. This insight would give Arconic the ability to save more by purchasing raw materials in larger lot sizes and producing larger lot quantity, granted storage cost is negligible. It's recommended that Arconic periodically reviews its customers' forecast as their forecasts change over time, as this would allow ample time for Arconic to react to market swings, since one of Arconic's goal is to be able to react to market swings.

Recommendation 3:

The exponential smoothing model assigned a greater weight to the most recent demands, so it would react better to market swings without needing updates or modifications. However, it would best predict one step ahead moving averages, so it is recommended that the forecast be generated every period for maximum accuracy. To produce the the short term forecast, Arconic would have to update the range in the "Weekly Demand" macro or the "Monthly Demand" macro in the provided excel sheet to select the weeks & months they wish to forecast. Along with that, to achieve a more accurate long term forecast, we recommend updating the long term forecast models once every six months using demands from the past 12 months. The python code for long term forecast is provided in the submitted package.

PART 2: VTI

Background Information

VTI of Texas is a company founded by Roger Clausen in 1956, that became an automated production line, fabricating wood doors, laminate & stone countertops, and varnished surfaces.

Problem Statement

The company faces is that there were 27 different board types to inventory, while the storage capacities, lead times, and shipment quantities all varied. The VTI company decided it needed to determine a safety stock, reorder point and reorder quantity to update automatically, which would be based on usage with maximizing trucks for each order.

Methods and Approaches

Based on the requirements, the project team decided to establish an ***optimal (Q,R) policy*** with the historical data offered by the company with the minimization of cost and standardization of quantities

with a periodic inventory level review. The project team recommended setting up a ***(S,s) model*** to perform a ***periodic-check system*** to determine the reorder limit and the reorder quantity.

2.1 Data Preprocessing and Preparation

Ideally, optimal (Q,R) policy can only be determined by iterative procedure, the project team first needed to determine if the penalty (or stock-out) cost was given. It seemed as if the company had a difficult time determining an exact value of the stock-out cost and the intangible loss-of-goodwill cost, so the project team decided to select a service level in the computation of the optimal (Q,R) system. After sorting the data given, the project team decided to use the data of *" >% Orders Filled Without Safety Stock"* from the *"Board Inventory Count Sheet VTI Of Texas"* from the *"Constants"* tab as the type 1 service level. The Type 1 service level is the percentage of orders fulfilled so it is appropriate to assume that the 99% value of "% orders filled without safety stock" equivalates to the type 1 service, α. Since the project is based upon fulfilling 99% Type 1 service level, the optimal quantity to purchase is equivalent to Economic Order Quantity (EOQ). The parameters needed to compute the EOQ are setup cost, demand rate, and holding cost. The reorder point is determined using the average demand during lead time, the service level and the standard error of demand during lead time.

The project team wasn't able to locate any information regarding setup cost and holding cost, so these two parameters are approximated using information obtained from online researches. In order to obtain recent demand rates, the team attempted to use the information on the *"Data Log"* tab to calculate weekly demand rate by finding the difference between inventory between weeks but there were a lot of discrepancies so it wasn't used. Eventually, the project team decided to use the data from *"Average Weekly Use and Std. Weekly Use"* in the *"Board Inventory Count Sheet VTI Of Texas"* sheet. The mean **λ** and standard deviation during lead time is calculated on the basis that each week contains 5 working days. Last but not the least, for the (S, s) periodic-check model, the project team decided to use the data from the *"Board Inventory Count Sheet VTI Of Texas"* in the *"Data Log"* sheet as the starting inventory u in a selected week. It was easy to understand that if summing up the *"In Storage"*, *"On Order"* and *"New Order"*, the data was the new starting inventory of this period. In this way it was easier for the company to update the starting inventory by just updating the inventory levels in the data log sheet. The reorder point is denoted as 's' and the reorder up to quantity is the summation of EOQ and reorder point and is denoted as 'S'.

For the consideration of maximizing trucks for each order, the project team decided to mix different products in a same truck in order to use the space in the truck more efficiently. The Solver function on

excel is used to maximize truck utilization by adjusting the "buffer percentages". For example, in cases where the optimal purchase for the week amount to only 2.5 trucks (the third truck only has 50% utilization) the solver function would adjust the buffer percentages so that the third truck is fully stocked by purchasing additional materials.Also, the project team doesn't take the difference of the vendors into account because the team assumes that VTI is an intermediate of those different vendors and the trucks could be shared among the products from different vendors.

Results - Inventory Control Model

The project team determined to use MS Excel to set up the computation for the (Q, R) policy and the (S, s) system for each product. Details of computation are seen in the attached excel file **"VTI QR Model".** VTI is free to update the demand rates, trucking cost, annual interest rate, the order cost and the unit cost per product to in order to produce more accurate purchasing quantity and reorder points.

To use the model made, VTI should follow the steps below:-

1. *Select date of last recorded inventory in data log sheet*
2. *Click on the "Click to Maximize Truck Utilization" button. (Optional)*
3. *Use Final Order Quantity for purchase.*

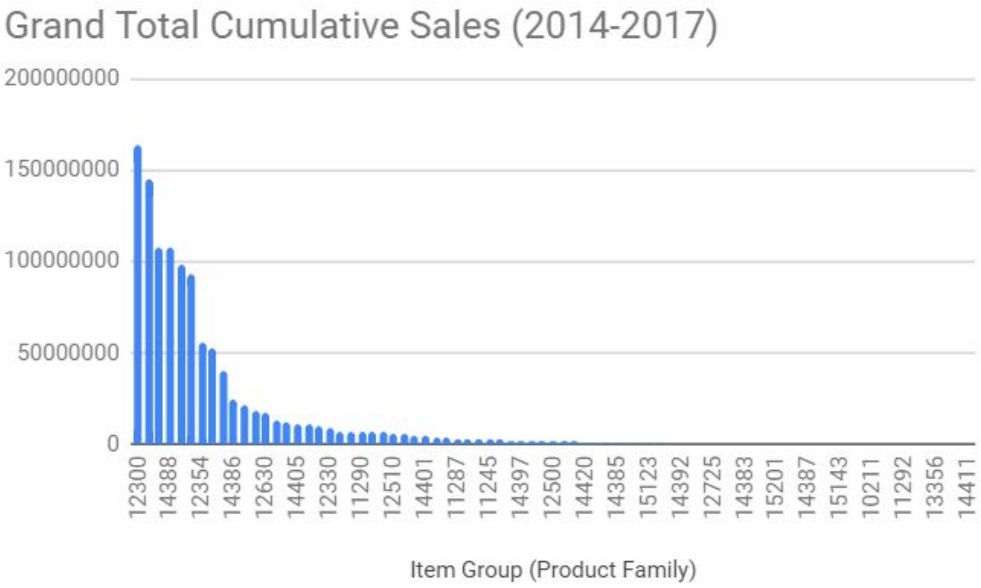
Recommendation

We recommend that VTI uses the most recent demand rates or forecast and update the trucking cost, annual interest rate, order cost and unit cost to increase the accuracy of the model. We recommend periodic updates to the model once every 3 months to ensure these parameters reflect the reality of the company so the likeliness of a stockout or overbuying could be reduced.

Conclusion

Arconic's forecasting along with VTI's inventory management system provides our team an insight on supply chain problems faced by the industry. The methods acquired in the class are good references to solve industry supply chain problems but often require tweeks to reflect the company's supply chain situation and the nature of the business.

Appendix


Grand Total Cumulative Sales (2014-2017)
200000000
150000000
100000000
50000000
0
12300
14388
12354
14386
12630
14405
12330
11290
12510
14401
11287
11245
14397
12500
14420
14385
15123
14392
12725
14383
15201
14387
15143
10211
11292
13356
14411
Item Group (Product Family)

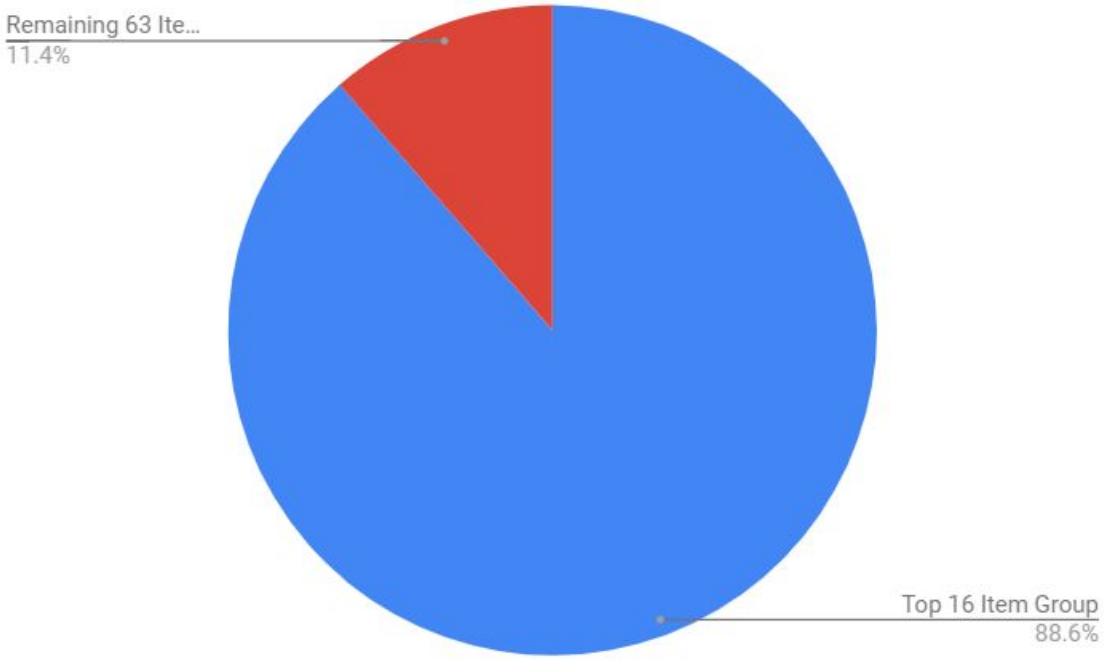

Remaining 63 Ite...
11.4%
Top 16 Item Group
88.6%

Figure 1 - Top 16 Product Group

```
Public Sub ExpLamCal()

Dim stuff As Range
Set stuff = Range("R8:R23")

Dim currentrow As Integer

For Each Item In stuff.Cells
    Range("B1") = Item.Value

    SolverOk SetCell:="$E$1", MaxMinVal:=2, ValueOf:=0, ByChange:="$B$2", Engine:=2 _
        , EngineDesc:="GRG Nonlinear"
        SolverSolve
        SendKeys ("{Enter}")
    currentrow = Item.row
    ActiveSheet.Cells(currentrow, "S") = Range("B2").Value
    ActiveSheet.Cells(currentrow, "T") = Range("E1").Value
    ActiveSheet.Cells(currentrow, "U") = Range("E2").Value
    ActiveSheet.Cells(currentrow, "V") = Range("E3").Value

    SolverOk SetCell:="$J$1", MaxMinVal:=2, ValueOf:=0, ByChange:="$B$2", Engine:=2 _
        , EngineDesc:="GRG Nonlinear"
        SolverSolve
        SendKeys ("{Enter}")
    ActiveSheet.Cells(currentrow, "W") = Range("B2").Value
    ActiveSheet.Cells(currentrow, "X") = Range("J1").Value
    ActiveSheet.Cells(currentrow, "Y") = Range("J2").Value
    ActiveSheet.Cells(currentrow, "Z") = Range("J3").Value

    SolverOk SetCell:="$O$1", MaxMinVal:=2, ValueOf:=0, ByChange:="$B$2", Engine:=2 _
        , EngineDesc:="GRG Nonlinear"
        SolverSolve
        SendKeys ("{Enter}")
    ActiveSheet.Cells(currentrow, "AA") = Range("B2").Value
    ActiveSheet.Cells(currentrow, "AB") = Range("O1").Value
    ActiveSheet.Cells(currentrow, "AC") = Range("O2").Value
    ActiveSheet.Cells(currentrow, "AD") = Range("O3").Value

Next Item

End Sub
```

Figure - VBA code to calculate lambda for exponential smoothing

https://www.plywoodcompany.com/MainSite/Store1/Content/SiteContent/1/Home/home.aspx

Resource used to estimate cost of product for VTI

```
from matplotlib import pyplot
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.tsa.arima_model import ARIMA

#Training data load
df_arima = pd.read_csv('C:/yoyo/course/615/project/arconic/data/Arconic.csv')
alist=[12300,12600,11286,14388,12610,12310,12354,11280,14403,14386,14407,11281,12630,12353,11285,14405]
df_arima = df_arima[df_arima['Item Group (Product Family)'].isin(alist)]
df_arima = df_arima[['Item Group (Product Family)','Quantity (PCS)','Request Date','Year','Week','Month','Quarter']]
df_arima.rename(columns={'Item Group (Product Family)': 'Item','Quantity (PCS)': 'Quantity',
                  'Request Date': 'Date'}, inplace=True)
df_arima = df_arima[(df_arima['Year'] >= 2015) & (df_arima['Year'] <= 2017)]
df_arima.head()


#Test data load
df_pre = pd.read_csv('C:/yoyo/course/615/project/arconic/data/Arconic.csv')
alist=[12300,12600,11286,14388,12610,12310,12354,11280,14403,14386,14407,11281,12630,12353,11285,14405]
df_pre = df_pre[df_pre['Item Group (Product Family)'].isin(alist)]
df_pre = df_pre[['Item Group (Product Family)','Quantity (PCS)','Request Date','Year','Week','Month','Quarter']]
df_pre.rename(columns={'Item Group (Product Family)': 'Item','Quantity (PCS)': 'Quantity',
                  'Request Date': 'Date'}, inplace=True)

#ARIMA Model Function define
def arima(df,item,p):
    train_week=df.loc[item]
    train_week=pd.pivot_table(train_week, index='Date', values='Quantity')
    model = ARIMA(train_week, order=(p,1,0))
    model_fit = model.fit(disp=0)
    return model_fit
```

```
##MAPE define
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
```

Figure - ARIMA Python Code to calculate regression

```
##weekly ARIMA
df_pre_week = df_pre[(df_pre['Year'] == 2018) & (df_pre['Week'] <= 6)]
df_pre_week=pd.pivot_table(df_pre_week, values=['Quantity', 'Date'], index=['Item', 'Year','Week'],
                aggfunc={'Quantity': sum,
                         'Date': max})
df_pre_week['Date'] = df_pre_week['Date'].apply(lambda x: pd.to_datetime(x))
df_test_week = df_pre_week
df_train_week =pd.pivot_table(df_arima, values=['Quantity', 'Date'], index=['Item', 'Year','Week'],
                aggfunc={'Quantity': sum,
                         'Date': max})
df_train_week['Date'] = df_train_week['Date'].apply(lambda x: pd.to_datetime(x))
print('--Weekly predict MAPE by first 6 weeks in 2018--')
print('item',' MAPE',' Forecast')
mape = 0
for item in alist:
    model_fit = arima(df_train_week,item,12)
    test_week=df_test_week.loc[item]
    test = pd.pivot_table(test_week,index = 'Date',values ='Quantity')
    y_test = test['Quantity']
    y_pred = model_fit.forecast(len(y_test))[0]
    item_MAPE = round(mean_absolute_percentage_error(y_test, y_pred),2)
    mape = mape + item_MAPE
    print(item,item_MAPE)

##Monthly ARIMA
df_pre_month = df_pre[(df_pre['Year'] == 2018) & (df_pre['Month'] <= 6)]
df_pre_month=pd.pivot_table(df_pre_month, values=['Quantity', 'Date'], index=['Item', 'Year','Month'],
                aggfunc={'Quantity': sum,
                         'Date': max})
df_pre_month['Date'] = df_pre_month['Date'].apply(lambda x: pd.to_datetime(x))
df_test_month = df_pre_month
df_train_month =pd.pivot_table(df_arima, values=['Quantity', 'Date'], index=['Item', 'Year','Month'],
                aggfunc={'Quantity': sum,
                         'Date': max})
df_train_month['Date'] = df_train_month['Date'].apply(lambda x: pd.to_datetime(x))
print('--Monthly predict MAPE by first 6 months in 2018--')
print('item',' MAPE')
mape = 0
for item in alist:
    model_fit = arima(df_train_month,item,6)
    test_month=df_test_month.loc[item]
    test = pd.pivot_table(test_month,index = 'Date',values ='Quantity')
    y_test = test['Quantity']
    y_pred = model_fit.forecast(len(y_test))[0]
    item_MAPE = round(mean_absolute_percentage_error(y_test, y_pred),2)
    mape = mape + item_MAPE
    print(item_MAPE)

##Quarterly ARIMA
df_pre_qr = df_pre[(df_pre['Year'] == 2018) & (df_pre['Quarter'] <= 2)]
df_pre_qr=pd.pivot_table(df_pre_qr, values=['Quantity', 'Date'], index=['Item', 'Year','Quarter'],
                aggfunc={'Quantity': sum,
                         'Date': max})
df_pre_qr['Date'] = df_pre_qr['Date'].apply(lambda x: pd.to_datetime(x))
df_test_qr = df_pre_qr
df_train_qr =pd.pivot_table(df_arima, values=['Quantity', 'Date'], index=['Item', 'Year','Quarter'],
                aggfunc={'Quantity': sum,
                         'Date': max})
df_train_qr['Date'] = df_train_qr['Date'].apply(lambda x: pd.to_datetime(x))
print('--Quaterly predict MAPE by quarter in 2018--')
print('item',' MAPE')
mape = 0
for item in alist:
    model_fit = arima(df_train_qr,item,1)
    test_qr=df_test_qr.loc[item]
    test = pd.pivot_table(test_qr,index = 'Date',values ='Quantity')
    y_test = test['Quantity']
    y_pred = model_fit.forecast(len(y_test))[0]
    item_MAPE = round(mean_absolute_percentage_error(y_test, y_pred),2)
    mape = mape + item_MAPE
    print(item,item_MAPE)
```

Figure - ARIMA Python Code to calculate regression

```python
# Define the plot function
def plot_point(x, y, y_pred):
    plt.figure()
    plt.scatter(x, y, color='black')
    plt.plot(x, y_pred, color='blue', linewidth=3)
    plt.xlabel('Time')
    plt.ylabel('Quantity')
    #设置坐标轴刻度
    plt.show()


# Regression Model with Seasonal Factor
def regression_with_seasonal_factor(df, df_new, item, time_unit):
    df_item = df.loc[item]
    df_item_group = df_item.groupby(['Year', time_unit]).sum()

    # Data Preprocessing
    nominal_data = [[str(df_item_group.index[i][1])] for i in range(df_item_group.shape[0])]
    onehot = preprocessing.OneHotEncoder()
    hot_array = onehot.fit_transform(nominal_data).toarray()

    quantity_data = [[df_item_group['Quantity (PCS)'].values[x]] for x in range(df_item_group.shape[0])]
    data_array = np.hstack((quantity_data, hot_array))
    df_encoded = pd.DataFrame(data_array)
    df_encoded['timeline'] = [x for x in range(1, data_array.shape[0] + 1)]
    tr = len(df_item_group.loc[2015, 'Quantity (PCS)'])

    df_encoded.drop([tr], axis = 1, inplace = True)

    ## Training Dataset
    X_train = df_encoded.iloc[: 3 * tr, 1:]
    y_train = np.ravel(df_encoded.iloc[:3 * tr, 0])

    ## Trainging the model
    lr = LinearRegression()
    lr.fit(X_train, y_train)

    # Validate dataset
    X_validate = df_encoded.iloc[3 * tr:int(3.5 * tr), 1:]
    y_validate = np.ravel(df_encoded.iloc[3 * tr:int(3.5 * tr), 0])

    y_train_pred = lr.pred(X_train)
    y_validate_pred = lr.predict(X_validate)
    # calculate the model parameter
    mse = math.sqrt(mean_squared_error(y_validate, y_validate_pred))
    mape = abs(y_validate - y_validate_pred) / (y_validate)
    r = r2_score(y_train, y_train_pred)

    # Plot the Trend line
    plot_point(X_train.iloc[:,-1], y_train, y_train_pred)
    return item, np.mean(mape), r

# Get the parameter of the model
for item in item_group:
    time_unit_1 = 'Quarter'
    data = regression_with_seasonal_factor(df_top, item, time_unit_1)
    print(data)
```

Figure: Regression Python Code model with seasonal factor