***Disclaimer: This completed capstone report is intended for interpretation for business members, not as much for data scientists. As such the level of detail in terms of the methods applied and code specifics will be interpreted at a high level. The capstone report was a project wrapup from the author (Joe Marco), who had limited R & data science knowledge prior to this initiative. See the end of the report for more information about the author***

1. Executive Overview The all too frequent feeling of searching for hours online for the perfect item, only to click the "add to cart" option or order button, to find out the product is on "backorder". Sometimes we do not even have this option when searching in a retail store, if a product is sold out, we simply move on 9 times out of 10. But what exactly is backorder?

"Backorders represent purchase orders made to the supplier for products that are already out of stock from a given location being served. Backordering is the process of selling inventory the company doesn’t have on hand. Backordering takes place only when the demand is captured in a formal manner: for example, in a retail store, most customers would simply move on when facing an out-of-shelf situation, without reporting the missing product to the store. Backorders represent specific challenges in terms of inventory optimization, as backordered units are typically associated with a degree of urgency coming from the client." 1

Now that we understand what a backorder is, why should we care? As end consumers this is typically less of a concern, but for the business who is actually struggling with product backorders, this is of extreme concern. Some companies may think "What's the big deal? The customer wants my product and has no problem waiting for it to be in stock." Some direct business impacts can be seen in the below reference:

Stock-outs cause a projected $25 billion in losses for individual businesses every year. To a struggling business, simply being out of merchandise doesn’t just put an immediate dent in finances, it can also mean adverse long-term impact such as loss of market share due to customer dissatisfaction, loss of patronage, and negative word-of-mouth. 2

If you are a business owner, or handle directly the shipping of products, you are probably now thinking "OH [$%#@!](mailto:$%#@!), I had no idea I could be losing that much" .... But what can I do to avoid these issues, and ensure proper branding of my business? Do you have an answer Joe?

--------- LEVERAGE YOUR DATA!! ---------

With this capstone report, I will be attempting to show you exactly the steps you can take (from a data point of view) to evaluate your supply chain information (inventory, lead time, sales predictions, etc.) and build predictive models that show how AT RISK your company may be for product backorder issues, ultimately contributing to a negative impact on your business. The methods included in this report will encompass Data Science fundamentals (data analysis, data cleansing, statistics, machine learning, easy to understand interpretations of output). The end deliverable as part of this initiative, will be to take product inventory information, and provide you a leverageable model that can be applied to existing data you should have on your product information, so you can better assess your companies risk for product backorders. More importantly, this will allow you to decide what factors you may be able to focus on to avoid this issue in the future.

### 2. The Plan

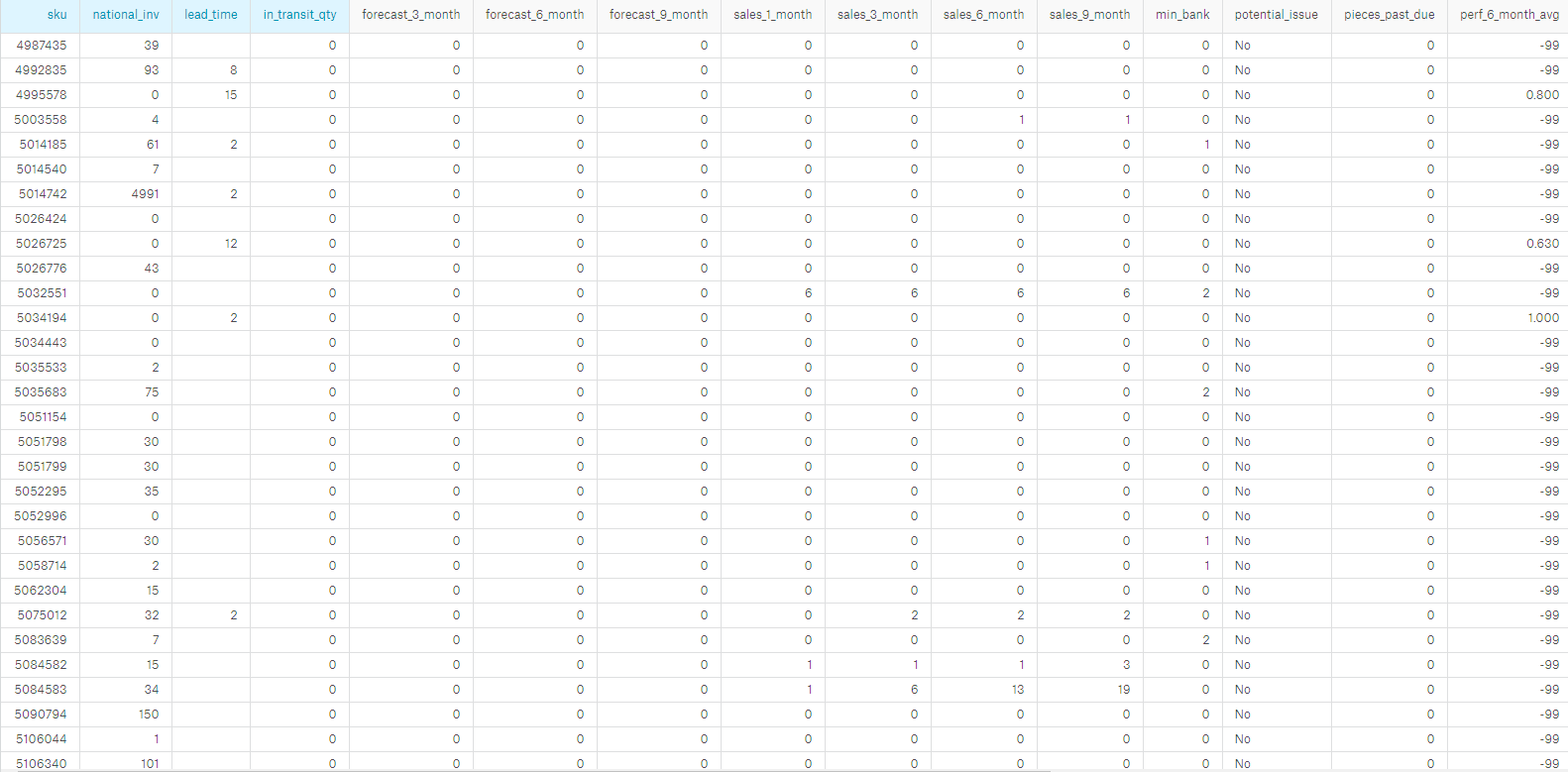
Joe, how do you plan to do this?

Plan of action focus areas:

* What can I predict based of a reference data set?
* What can I measure based off my output?
* What actions should be taken by a business based off my output?
* We will attempt to predict back orders based off historical data
* We will Utilize key pieces of data like inventory levels and lead time to analyze predictions of backorders
* We will focus on products that have sales in recent months, and have a high consumer need so that are model is optimized for "in demand products"

### 3. The Data

* I will be leveraging existing product backorder data that highlights a variety of information around products, and will usually exist in business systems. A snapshot of the data can be seen below. For information refer to references. 3



Screenshot of some of the data

#### Data Definitions:

Moving forward we will specifically focus on the items below highlighted red above as these will contribute to our final model of backorder predictions.

* sku - Random ID for the product
* national\_inv - Current inventory level for the part
* lead\_time - Transit time for product (if available)
* in\_transit\_qty - Amount of product in transit from source
* forecast\_3\_month - Forecast sales for the next 3 months
* forecast\_6\_month - Forecast sales for the next 6 months
* forecast\_9\_month - Forecast sales for the next 9 months
* sales\_1\_month - Sales quantity for the prior 1 month time period
* sales\_3\_month - Sales quantity for the prior 3 month time period
* sales\_6\_month - Sales quantity for the prior 6 month time period
* sales\_9\_month - Sales quantity for the prior 9 month time period
* min\_bank - Minimum recommend amount to stock
* potential\_issue - Source issue for part identified
* pieces\_past\_due - Parts overdue from source
* perf\_6\_month\_avg - Source performance for prior 6 month period
* perf\_12\_month\_avg - Source performance for prior 12 month period
* local\_bo\_qty - Amount of stock orders overdue
* deck\_risk - Part risk flag
* oe\_constraint - Part risk flag
* ppap\_risk - Part risk flag
* stop\_auto\_buy - Part risk flag
* rev\_stop - Part risk flag
* went\_on\_backorder - Product actually went on backorder. This is the target value.

### 3.1 Data Limitations

* Now that we have an idea of the data attributes most important to the business case at hand, we can talk about some items that we CANNOT do with the data set given:
  + We do not have dollar sales figures, meaning no way to guage potential impact of revenue/profit based off backorder issues
  + We do not have "Source (supplier)" information to be able to determine which suppliers by name are contributing to backorder issues, although we could likely find some commonalities to assume which skus come from the same source

### 4. Data Wrangling

One of the most time consuming steps in any data analysis is cleaning the data and getting it into a format amenable for analysis. Data wrangling, also known as Data Munging, is the process of converting data from a raw form into another format that allows for more convenient analysis of the data with the help of semi-automated tools.

Load tidyverse package which includes: \* ggplot2, for data visualisation. \* dplyr, for data manipulation. \* tidyr, for data tidying. \* readr, for data import. \* purrr, for functional programming. \* tibble, for tibbles, a modern re-imagining of data frames

#suppressmessages so we do not get a ton of additional run on loading criteria  
suppressMessages(library(tidyverse))

#### 4.1 Read in the data

Next we will read in our data (ensure you are in the directory where your file is saved)

#assign data to a dataframe, in this case "capstonedata"  
setwd("~/DataScienceFoundationsCourse/Capstone")  
  
capstonedata <- read.csv('productbackorder.csv')  
  
# you can view your datafram with the code below if you are interested in initial analysis  
# View(capstonedata)

#### 4.2 Rename fields to common business ontologies

Rename some columns if needed, to common ontologies for you business, so your specific groups can understand which field values refer to what business process (in our case the definitions are above, but we will rename a value for sake of showing how it is done with R in just one line of code). I am renaming min\_bank from aboce to min\_stock:

capstonedata <- rename(capstonedata, min\_stock = min\_bank)

#### 4.3 Check for null values in the data set (all fields)

Let's analyze our data a bit further by checking for missing values and decide what to do about them across every column in the data frame:

If you run the below code you will be able to see for each column if a Null/NA value exists when you run the code. Each line will generate either a TRUE (null exists) or a FALSE (no null). We will not run this code now, as it will give our report unnecessary length while each line generates a true or false.

any(is.na(capstonedata$sku))  
any(is.na(capstonedata$national\_inv))  
any(is.na(capstonedata$lead\_time))  
any(is.na(capstonedata$in\_transit\_qty))  
any(is.na(capstonedata$forecast\_3\_month))  
any(is.na(capstonedata$forecast\_6\_month))  
any(is.na(capstonedata$forecast\_9\_month))  
any(is.na(capstonedata$sales\_1\_month))  
any(is.na(capstonedata$sales\_3\_month))  
any(is.na(capstonedata$sales\_6\_month))  
any(is.na(capstonedata$sales\_9\_month))  
any(is.na(capstonedata$min\_stock))  
any(is.na(capstonedata$potential\_issue))  
any(is.na(capstonedata$pieces\_past\_due))  
any(is.na(capstonedata$perf\_6\_month\_avg))  
any(is.na(capstonedata$perf\_12\_month\_avg))  
any(is.na(capstonedata$local\_bo\_qty))  
any(is.na(capstonedata$deck\_risk))  
any(is.na(capstonedata$oe\_constraint))  
any(is.na(capstonedata$ppap\_risk))  
any(is.na(capstonedata$stop\_auto\_buy))  
any(is.na(capstonedata$rev\_stop))  
any(is.na(capstonedata$went\_on\_backorder))

#### 4.4 Replace null values

BAsed off the above we can see the below columns have some NA or Null values:

* **national\_inv**
* **lead\_time**
* **in\_transit\_qty**
* **forecast\_3\_month**
* **forecast\_6\_month**
* **forecast\_9\_month**
* **sales\_1\_month**
* **sales\_3\_month**
* **sales\_6\_month**
* **sales\_9\_month**
* **min\_stock**
* **pieces\_past\_due**
* **perf\_6\_month\_avg**
* **perf\_12\_month\_avg**
* **local\_bo\_qty**

All these columns are fields that have numbers within the data rows. Thus we will remove the fields with null values and replace them with the average of said fields/columns:

Assign containers to the mean of the associated columns with null values (your mean needs to ignore "!" any NA columns so that the mean does not pull back NA). We will use these next to overwrite the null values

national\_inv\_mean <- mean(capstonedata$national\_inv[!is.na(capstonedata$national\_inv)])  
lead\_time\_mean <- mean(capstonedata$lead\_time[!is.na(capstonedata$lead\_time)])  
in\_transit\_qty\_mean <- mean(capstonedata$in\_transit\_qty[!is.na(capstonedata$in\_transit\_qty)])  
forecast\_3\_month\_mean <- mean(capstonedata$forecast\_3\_month[!is.na(capstonedata$forecast\_3\_month)])  
forecast\_6\_month\_mean <- mean(capstonedata$forecast\_6\_month[!is.na(capstonedata$forecast\_6\_month)])  
forecast\_9\_month\_mean <- mean(capstonedata$forecast\_9\_month[!is.na(capstonedata$forecast\_9\_month)])  
sales\_1\_month\_mean <- mean(capstonedata$sales\_1\_month[!is.na(capstonedata$sales\_1\_month)])  
sales\_3\_month\_mean <- mean(capstonedata$sales\_3\_month[!is.na(capstonedata$sales\_3\_month)])  
sales\_6\_month\_mean <- mean(capstonedata$sales\_6\_month[!is.na(capstonedata$sales\_6\_month)])  
sales\_9\_month\_mean <- mean(capstonedata$sales\_9\_month[!is.na(capstonedata$sales\_9\_month)])  
min\_stock\_mean <- mean(capstonedata$min\_stock[!is.na(capstonedata$min\_stock)])  
pieces\_past\_due\_mean <- mean(capstonedata$pieces\_past\_due[!is.na(capstonedata$pieces\_past\_due)])  
perf\_6\_month\_avg\_mean <- mean(capstonedata$perf\_6\_month\_avg[!is.na(capstonedata$perf\_6\_month\_avg)])  
perf\_12\_month\_avg\_mean <- mean(capstonedata$perf\_12\_month\_avg[!is.na(capstonedata$perf\_12\_month\_avg)])  
local\_bo\_qty\_mean <- mean(capstonedata$local\_bo\_qty[!is.na(capstonedata$local\_bo\_qty)])

overwrite the existing columns NA values with the mean (average) of all other values in that column:

capstonedata$national\_inv[is.na(capstonedata$national\_inv)] <- national\_inv\_mean  
capstonedata$lead\_time[is.na(capstonedata$lead\_time)] <- lead\_time\_mean  
capstonedata$in\_transit\_qty[is.na(capstonedata$in\_transit\_qty)] <- in\_transit\_qty\_mean  
capstonedata$forecast\_3\_month[is.na(capstonedata$forecast\_3\_month)] <- forecast\_3\_month\_mean  
capstonedata$forecast\_6\_month[is.na(capstonedata$forecast\_6\_month)] <- forecast\_6\_month\_mean  
capstonedata$forecast\_9\_month[is.na(capstonedata$forecast\_9\_month)] <- forecast\_9\_month\_mean  
capstonedata$sales\_1\_month[is.na(capstonedata$sales\_1\_month)] <- sales\_1\_month\_mean  
capstonedata$sales\_3\_month[is.na(capstonedata$sales\_3\_month)] <- sales\_3\_month\_mean  
capstonedata$sales\_6\_month[is.na(capstonedata$sales\_6\_month)] <- sales\_6\_month\_mean  
capstonedata$sales\_9\_month[is.na(capstonedata$sales\_9\_month)] <- sales\_9\_month\_mean  
capstonedata$min\_stock[is.na(capstonedata$min\_stock)] <- min\_stock\_mean  
capstonedata$pieces\_past\_due[is.na(capstonedata$pieces\_past\_due)] <- pieces\_past\_due\_mean  
capstonedata$perf\_6\_month\_avg[is.na(capstonedata$perf\_6\_month\_avg)] <- perf\_6\_month\_avg\_mean  
capstonedata$perf\_12\_month\_avg[is.na(capstonedata$perf\_12\_month\_avg)] <- perf\_12\_month\_avg\_mean  
capstonedata$local\_bo\_qty[is.na(capstonedata$local\_bo\_qty)] <- local\_bo\_qty\_mean

#### 4.5 Confirm null values are in fact replaced

If you would like to double check that now Null values are gone, rerun the code below to check for NA values, we will skip this step. All should be FALSE now:

any(is.na(capstonedata$sku))  
any(is.na(capstonedata$national\_inv))  
any(is.na(capstonedata$lead\_time))  
any(is.na(capstonedata$in\_transit\_qty))  
any(is.na(capstonedata$forecast\_3\_month))  
any(is.na(capstonedata$forecast\_6\_month))  
any(is.na(capstonedata$forecast\_9\_month))  
any(is.na(capstonedata$sales\_1\_month))  
any(is.na(capstonedata$sales\_3\_month))  
any(is.na(capstonedata$sales\_6\_month))  
any(is.na(capstonedata$sales\_9\_month))  
any(is.na(capstonedata$min\_stock))  
any(is.na(capstonedata$potential\_issue))  
any(is.na(capstonedata$pieces\_past\_due))  
any(is.na(capstonedata$perf\_6\_month\_avg))  
any(is.na(capstonedata$perf\_12\_month\_avg))  
any(is.na(capstonedata$local\_bo\_qty))  
any(is.na(capstonedata$deck\_risk))  
any(is.na(capstonedata$oe\_constraint))  
any(is.na(capstonedata$ppap\_risk))  
any(is.na(capstonedata$stop\_auto\_buy))  
any(is.na(capstonedata$rev\_stop))  
any(is.na(capstonedata$went\_on\_backorder))

#### 4.6 Save your cleansed output

Output the data to a new cleaned file

write.csv(capstonedata, file = "capstonedata\_clean.csv")

### 5. Statistics, trends, probability, & visual analysis

Now that we have our cleansed data set, we will proceed to apply some foundational statistics methods as commonly used in data science initiatives. We will also utilize ggplot (a visualization package in R) to visually analyze our information at a high level

#### 5.1 Unique counting

We know that we replaced the null values in our data set earlier. But lets assume someone just hands you a data set and tells you it is clean, we want to do a quick spot check to confirm they are correct. Let's start by double checking we have no "NA" values across the entire data set and count unique skus (products):

capstonedata <- read.csv('capstonedata\_clean.csv')  
  
any(is.na(capstonedata$sku))

## [1] FALSE

nrow(distinct(capstonedata))

## [1] 1687861

#### 5.2 Focus on backordered products

We want to explore the total products to find out how many specifically went on backorder vs did not go on backorder. The field "went\_on\_backorder" is what we will explore further

table(capstonedata$went\_on\_backorder)

##   
## No Yes   
## 1 1676567 11293

Interesting... now we can see a small percentage of our total products went on backorder, we may want to focus specifically on the products that went on backorder, so lets create a dataframe for those 7602 products

productbackorders <- capstonedata[capstonedata$went\_on\_backorder == "Yes",]

#### 5.3 Calculate probability

I mentioned a small percentage of our total list of products went on backorder, but lets show the EXACT probability of backorder based off the data we have

# We can show the probability of backorder based off our data, to do this we will want to divide products that went on backorer over the total amount of products  
Probability\_backorder <- nrow(productbackorders) / nrow(capstonedata)  
Probability\_backorder

## [1] 0.006690717

# Great - we can see our output, but in decimal format, let's throw a function in there to convet the decimal to percent  
percent <- function(x, digits = 2, format = "f", ...) {  
 paste0(formatC(100 \* x, format = format, digits = digits, ...), "%")  
}  
# now we simply call our function "percent" on the output of our probability  
percent(Probability\_backorder)

## [1] "0.67%"

#### 5.4 Calculate Averages

Let's take a look at some averages around ALL Products (inventory and lead time). Let's not forget to add in standard deviation calculations as well so we can understand how spread out our averages are. Recall the output of standard deviation: A low standard deviation means that most of the numbers are very close to the average. A high standard deviation means that the numbers are spread out

mean\_inventory <- mean(capstonedata$national\_inv[!is.na(capstonedata$national\_inv)])  
mean\_leadtime <- mean(capstonedata$lead\_time[!is.na(capstonedata$lead\_time)])  
  
sd\_inventory <- sd(capstonedata$national\_inv[!is.na(capstonedata$national\_inv)])  
sd\_leadtime <- sd(capstonedata$lead\_time[!is.na(capstonedata$lead\_time)])  
  
mean\_inventory

## [1] 496.1118

sd\_inventory

## [1] 29615.23

mean\_leadtime

## [1] 7.872267

sd\_leadtime

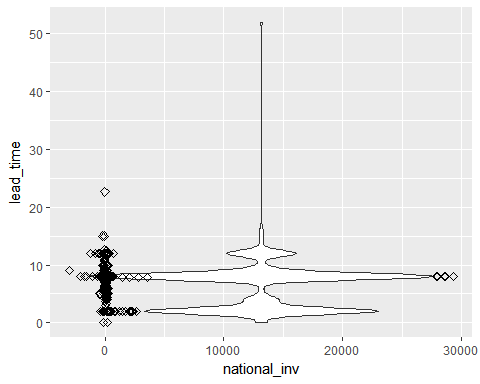
## [1] 6.841883

We can see our variance for inventory is HUGE - which makes sense (some products have very low inventory, some have very high), while the variance for leadtime is roughly close to the average

#### 5.5 Trends (high, low, increase, decrease, anomalies)

Let's try a violin plot. We will compare national inventory with lead time.

ggplot(productbackorders, aes(x = national\_inv, y = lead\_time))+  
 geom\_violin() +  
 stat\_summary(fun.y=mean, geom="point", shape=23, size=2)

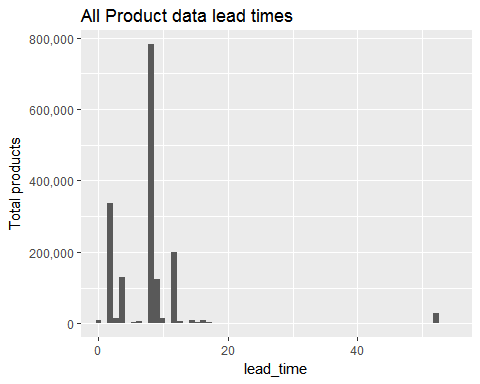


The above plot tells us that we see some groupings for products that went on backorder with lead time in the following groups (0-3, 7-10, 12-13). We then plot the poins on our averages and what we can see is that we have a average at or close to 0 for many of our products which basically means, I have 0 or negative inventory for products that went on backorder, which makes sense and means I may need to increase my inventory for those items.

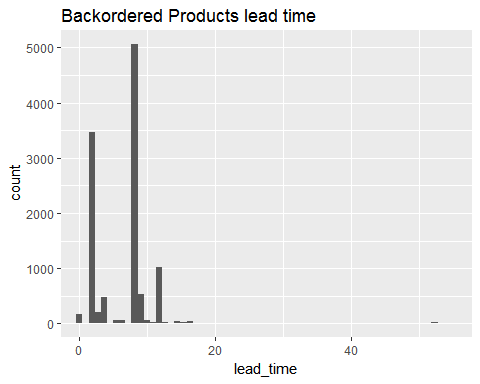
#### 5.6 Data Visualization

Now let's make a histogram looking at ALL products vs Backordered products and their associated lead times. We want to measure if their is some association with how long it takes products to be delivered and if that plays into backorders. With the below graphs that longer lead time does not necessarily == backorder. We will use ggplot to visualize.

suppressMessages(library(scales))  
  
ggplot (capstonedata, aes(lead\_time)) +   
 geom\_histogram(binwidth = 0.9)+  
 coord\_cartesian(xlim = c(0, 55))+  
 ggtitle("All Product data lead times")+  
 scale\_y\_continuous(name="Total products", labels = comma)



ggplot (productbackorders, aes(lead\_time)) +   
 geom\_histogram(binwidth = 0.9)+  
 coord\_cartesian(xlim = c(0, 55)) +  
 ggtitle("Backordered Products lead time")

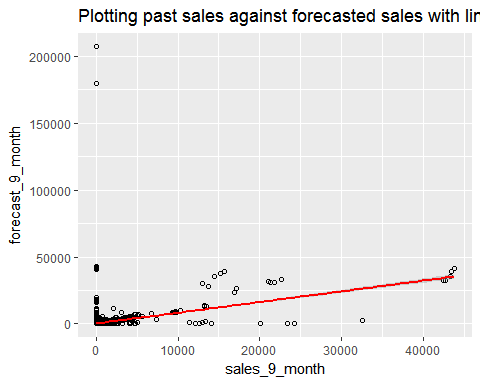


scale\_y\_continuous(name="Total backordered products", labels = comma)

## <ScaleContinuousPosition>  
## Range:   
## Limits: 0 -- 1

Lets make a scatter plot looking at SALES VS FORECAST specifically for products that went on backorder. What we can gather from this is that there is some density close to 0, meaning our actual sales is close to 0 dollars and our forecast of sales is close to 0 dollars --- raises the question, WHY are we even making and producing those products?? Additionally but we can see a few data points where we are forcasting $175,000 or more in sales, we NEED to make sure we stock up these products to avoid backorder, otherwise we have a major impact to business revenue. Additionally, there are a few points in which our sales were 40,000 or more over the last 9 months, these appear to be the major revenue making products over the past 9 months that went on backorder, so we should focus on upping our inventory for these products as well

ggplot(productbackorders, aes(x = sales\_9\_month, y = forecast\_9\_month)) +   
 geom\_point(shape = 1)+  
 geom\_smooth(method=lm , color="red", se=TRUE)+  
 ggtitle("Plotting past sales against forecasted sales with linear pattern")



# Smoothed conditional means aids the eye in seeing patterns in the presence of overplotting. geom\_smooth and stat\_smooth are effectively aliases: they both use the same arguments.

Let's cut up some of our data, specifically looking at lead time and inventory, and plotting some baselines for "low", "medium", "high" to see if we can gather some insight.

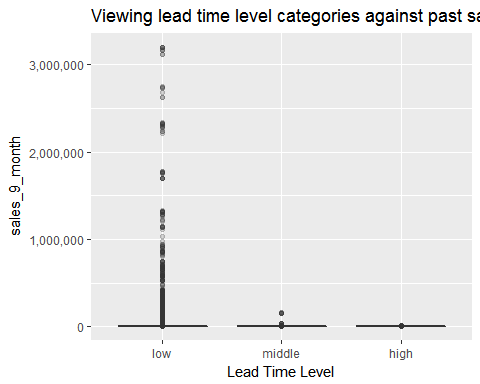
#cut the data  
cutinv <- cut(capstonedata$national\_inv,3,labels=c("low","middle","high"))  
  
#check out counts  
table(cutinv)

## cutinv  
## low middle high   
## 1687847 7 7

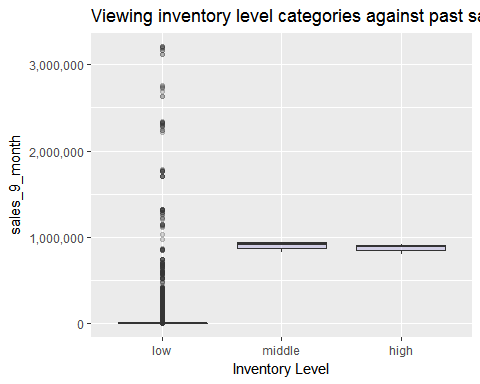
#add to dataset  
capstonedata$invlevel<- cut(capstonedata$national\_inv,3,labels=c("low","middle","high"))  
  
#cut the data on lead time  
cutlead <- cut(capstonedata$lead\_time,3,labels=c("low","middle","high"))  
  
#check out counts  
table(cutlead)

## cutlead  
## low middle high   
## 1656181 1484 30196

#add to data set  
capstonedata$leadlevel<- cut(capstonedata$lead\_time,3,labels=c("low","middle","high"))  
  
#plot lead level category against past sales to understand if there is any correlation. We can see that alot of products we sold had a low lead time, a few had middle level lead time, and less than that had high lead time  
ggplot(capstonedata, aes(x=as.factor(leadlevel), y=sales\_9\_month)) +   
 geom\_boxplot(fill="slateblue", alpha=0.2) +   
 xlab("Lead Time Level")+  
 scale\_y\_continuous(labels = scales::comma)+  
 ggtitle("Viewing lead time level categories against past sales")



#plot inventory levels on data and see if there is correlation on pas sales, and amount of inventory business has available. We can see we have low inventory for alot of things we sold!  
ggplot(capstonedata, aes(x=as.factor(invlevel), y=sales\_9\_month)) +   
 geom\_boxplot(fill="slateblue", alpha=0.2) +   
 xlab("Inventory Level")+  
 scale\_y\_continuous(labels = scales::comma)+  
 ggtitle("Viewing inventory level categories against past sales")



### 6. Machine Learning and Predictions

At this point, I have collected my data, cleaned it up, wrangled it into shape and explored it. Now it's time to perform some in-depth data analysis using machine learning.

#### 6.1 Logistic Regression

Preferred option I will be choosing to utilize supervised machine learning (specifically logistic regression). The reason why I choose to focus on logistic regression is due to the fact that logistic regression will lead to the outcome (dependent variable) having only a limited number of possible values.

In our case, will the product go on backorder -- (Yes, No) OR (1, 0) speaking in binary. Wheras another regression type is linear regression in which the outcome (dependent variable) is continuous. It can have any one of an infinite number of possible values, which is less relevant to our use case.

Predictors (Independent Variables of Choice)

* Lead Time
* Inventory Levels
* forecast\_3\_month (we will focus on values greater than 0, as a predicted forecast of less than 0 will mess up our predictions)
* sales\_9\_month (we will focus on values greater than 0, as a past sales 9 month of less than 0 or 0 will mess up our predictions)
* How will you evaluate the success of your machine learning technique? What metric will you use?

I will plan to see how well my logistic regression model can classify product backorders by producing a ROC (Reciever Operating Characteristic) curve. ROC Curves are used to see how well a classifier can separate positive and negative examples and to identify the best threshold for separating them. This will help me to understand how well my classifiers seperate (will go on backorder vs will not go on backorder) predictions. A perfect classifier would be EXACTLY 1, so we will attempt to get as close to 1 as we can. Out of general interest for clustering, I plan to attempt some cluster exercises against the data to see what kind of clusters I can unravel in the data. This should not be considered in the above steps, and is just an added personal preference to play with an additional method.

Start by reading in cleansed data, then convert yes & nos to binary values (1 or 0)

#Read in cleansed file from earlier  
capstonedata <- read.csv('capstonedata\_clean.csv')  
  
#double check for nulls  
any(is.na(capstonedata))

## [1] FALSE

#convert binary values (1 or 0) for "Yes" & "No" options  
indx = sapply(capstonedata,is.factor)  
capstonedata[indx]= lapply(capstonedata[indx],as.character)  
capstonedata[capstonedata == "Yes"] = 1  
capstonedata[capstonedata == "No"] = 0  
capstonedata[indx]= lapply(capstonedata[indx],as.numeric)

## Warning in lapply(capstonedata[indx], as.numeric): NAs introduced by  
## coercion

#lets check the structure now, we should see no more yes or no values  
str(capstonedata)

## 'data.frame': 1687861 obs. of 24 variables:  
## $ X : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ sku : num 1026827 1043384 1043696 1043852 1044048 ...  
## $ national\_inv : num 0 2 2 7 8 ...  
## $ lead\_time : num 7.87 9 7.87 8 7.87 ...  
## $ in\_transit\_qty : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ forecast\_3\_month : num 0 0 0 0 0 0 0 0 15 0 ...  
## $ forecast\_6\_month : num 0 0 0 0 0 0 0 0 114 0 ...  
## $ forecast\_9\_month : num 0 0 0 0 0 0 0 0 152 0 ...  
## $ sales\_1\_month : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ sales\_3\_month : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ sales\_6\_month : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ sales\_9\_month : num 0 0 0 0 4 0 0 0 0 0 ...  
## $ min\_stock : num 0 0 0 1 2 0 4 0 0 0 ...  
## $ potential\_issue : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ pieces\_past\_due : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ perf\_6\_month\_avg : num -99 0.99 -99 0.1 -99 0.82 -99 0 -99 0.82 ...  
## $ perf\_12\_month\_avg: num -99 0.99 -99 0.13 -99 0.87 -99 0 -99 0.87 ...  
## $ local\_bo\_qty : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ deck\_risk : num 0 0 1 0 1 0 1 1 0 0 ...  
## $ oe\_constraint : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ ppap\_risk : num 0 0 0 0 0 0 0 1 0 0 ...  
## $ stop\_auto\_buy : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ rev\_stop : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ went\_on\_backorder: num 0 0 0 0 0 0 0 0 0 0 ...

#### 6.2 Build your Logistic Regression Model(s)

# Let's create a training set with the data --   
# Revisit your backorder numbers on the data set if you forget. 0 = no backorder, 1 = backorder  
table(capstonedata$went\_on\_backorder)

##   
## 0 1   
## 1676567 11293

#load the caret library and caTools  
library(caret)

## Warning: package 'caret' was built under R version 3.4.2

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

library(caTools)  
  
#set a specified seed  
set.seed(150)  
  
#utilize the subset function. SplitRatio = percentage of the original data you want in your training set  
  
split = sample.split(capstonedata$went\_on\_backorder, SplitRatio = .7)  
table(split)

## split  
## FALSE TRUE   
## 506359 1181502

#Define train and test sets  
BackorderTrain = subset(capstonedata, split == TRUE)  
BackorderTest = subset(capstonedata, split == FALSE)  
  
#eliminate NAs from train set, which may have been created with converting values to binary  
BackorderTrain <- na.omit(BackorderTrain)  
  
#count rows of train and test  
nrow(BackorderTrain)

## [1] 1181502

nrow(BackorderTest)

## [1] 506359

#use glm function with two different models. First we will start with 4 predictors  
model\_more\_predictors <- glm(went\_on\_backorder ~ lead\_time + national\_inv + forecast\_3\_month + sales\_9\_month, data = BackorderTrain, family = binomial)

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

#create another model with less predictors (2 this time)  
Backorderlog <- glm(went\_on\_backorder ~ lead\_time + national\_inv, data = BackorderTrain, family = binomial)

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

#Evaluate the 2 models against each other:  
anova(model\_more\_predictors, Backorderlog, test = "LRT")

## Analysis of Deviance Table  
##   
## Model 1: went\_on\_backorder ~ lead\_time + national\_inv + forecast\_3\_month +   
## sales\_9\_month  
## Model 2: went\_on\_backorder ~ lead\_time + national\_inv  
## Resid. Df Resid. Dev Df Deviance Pr(>Chi)  
## 1 1181497 569922   
## 2 1181499 92602 -2 477320

A logistic regression is said to provide a better fit to the data if it demonstrates an improvement over a model with fewer predictors. This is performed using the likelihood ratio test, which compares the likelihood of the data under the full model against the likelihood of the data under a model with fewer predictors. Removing predictor variables from a model will almost always make the model fit less well (i.e. a model will have a lower log likelihood), but it is necessary to test whether the observed difference in model fit is statistically significant. Given that (H\_0) holds that the reduced model is true, a p-value for the overall model fit statistic that is less than (0.05) would compel us to reject the null hypothesis. It would provide evidence against the reduced model in favor of the current model. The likelihood ratio test can be performed in R using the lrtest() function from the lmtest package or using the anova() function in base.4

We will move forward with the model with less predictors as this will give us a better model.

#view summary of glm function  
summary(Backorderlog)

##   
## Call:  
## glm(formula = went\_on\_backorder ~ lead\_time + national\_inv, family = binomial,   
## data = BackorderTrain)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -8.4904 -0.1229 -0.1187 -0.1066 8.4904   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -4.3993202 0.0230556 -190.81 <2e-16 \*\*\*  
## lead\_time -0.0613449 0.0030678 -20.00 <2e-16 \*\*\*  
## national\_inv -0.0024775 0.0001018 -24.33 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 94918 on 1181501 degrees of freedom  
## Residual deviance: 92602 on 1181499 degrees of freedom  
## AIC: 92608  
##   
## Number of Fisher Scoring iterations: 14

#make predictions  
predictTrain <- predict(Backorderlog, type = "response")  
summary(predictTrain)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000000 0.005708 0.007024 0.006691 0.007523 1.000000

#based off our predictions we can view the head of our predicted set (likelihood of backorder, and we see for the first few records a VERY LOW percentage of backorder .7, .6, etc..)  
head(predictTrain)

## 1 2 3 5 6   
## 0.0075229619 0.0069891709 0.0074860563 0.0073764142 0.0072298052   
## 7   
## 0.0005026574

#we can revist the actual measures of our capstone data set for the first few values and see that the head of our data did not go on backorder, so our predictions were accurate in such that the prediction likelihood was correct.   
head(capstonedata$went\_on\_backorder)

## [1] 0 0 0 0 0 0

#double check rows in train set and predict set are exactly the same, (THESE NEED TO BE THE SAME FOR YOUR GLM FUNCTION TO WORK)  
nrow(BackorderTrain)

## [1] 1181502

length(predictTrain)

## [1] 1181502

tapply(predictTrain, BackorderTrain$went\_on\_backorder, mean)

## 0 1   
## 0.006678495 0.008493181

table(BackorderTrain$went\_on\_backorder, predictTrain > .2)

##   
## FALSE TRUE  
## 0 1173558 39  
## 1 7902 3

#### 6.3 Model assessment

#overall Accuracy  
1173561/1181502

## [1] 0.9932789

#Overall Error Rate  
7941/1181502

## [1] 0.006721106

#calculate sensitivity  
3/7905

## [1] 0.0003795066

#Calculate specificity  
1173558/1173597

## [1] 0.9999668

#### 6.4 ROC Curve

Let's place our values on a Reciever Operator Characteristic Curve (ROC)

library(ROCR)

## Warning: package 'ROCR' was built under R version 3.4.2

## Loading required package: gplots

## Warning: package 'gplots' was built under R version 3.4.2

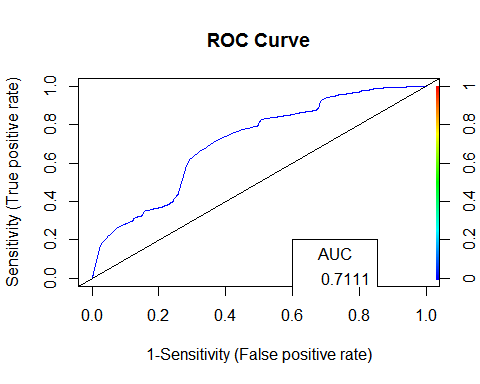
##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

ROCRpred <- prediction(predictTrain, BackorderTrain$went\_on\_backorder)  
ROCRperf <- performance(ROCRpred, "tpr", "fpr")  
plot(ROCRperf, colorize = TRUE, main = "ROC Curve", ylab = "Sensitivity (True positive rate)", xlab = "1-Sensitivity (False positive rate)")  
abline(a=0, b=1)  
  
# Calculate the AUC (area under the curver)  
auc <-performance(ROCRpred, "auc")  
auc <- unlist(slot(auc, "y.values"))  
  
#run AUC to see the number  
auc

## [1] 0.7111262

#round to less decimals  
auc <- round(auc, 4)  
  
legend(.6, .2, auc, title = "AUC")



Interpretations of model based on AUC measures, can be seen below. Although my AUC did not hit the upper quantile of excellent or good, for a rookie and amateur .7 is not so bad for my first try! With more time, and more experience/expertise, I would continue to play with different variables and combinations to try to improve this overall AUC measure, which equates to a more efficient model.

* 0.9 < AUC < 1.0 = excellent
* 0.8 < AUC < 0.9 = good
* 0.7 < AUC < 0.8 = okay
* 0.6 < AUC 0.7 = not good
* 0.5 < AUC < 0.6 = failed

#### 6.5 Out of sample metrics

Now we want to make predictions on a test set to compute out-of-sample metrics

predictTest <- predict(Backorderlog, type="response", newdata = BackorderTest)  
summary(predictTest)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000000 0.005708 0.007029 0.006706 0.007523 1.000000

tapply(predictTest, BackorderTest$went\_on\_backorder, mean)

## 0 1   
## 0.006690727 0.008966272

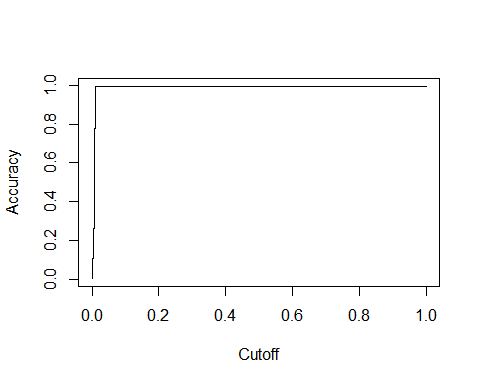
table(BackorderTest$went\_on\_backorder, predictTest > .2)

##   
## FALSE TRUE  
## 0 502953 17  
## 1 3386 2

#out of sample accuracy:  
502955/506358

## [1] 0.9932795

#lets plot the accuracy on a graph  
acc.perf <- performance(ROCRpred, measure = "acc")  
plot(acc.perf)



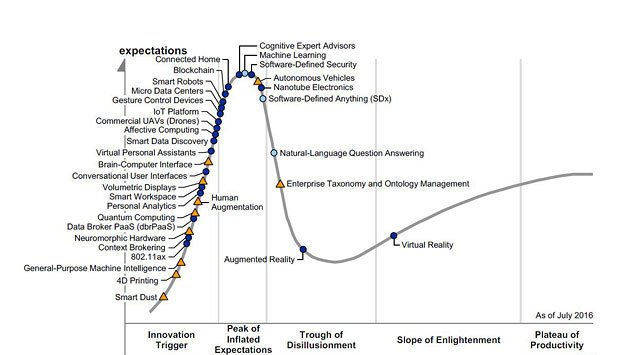
#### 7. Conclusions and Recommendations

* A trained model can accurately identify likelihood of product backorders based off inventory levels and lead time.
* In practice, the probabilities returned from the logistic regression model can be used to prioritize how "AT RISK" your supply chain pipeline could be due to lead time & inventory issues.
* This model can be used to feed another out of sample set of data, from any big business, in which we stated a list of products, their lead time to delivery, and their inventory to get some rough predictions on likelihood of backorders.
* We can assume that if our predicted true backorders were a larger number, then we would have an issue that we could act on and increase/replenish inventory, or look into product transport and see how we can cut down lead time.

#### Where do we go from here?

The best recommendation I could give off the analysis and interpretations we have done is to place heavy focus on inventory and lead time around your product data. Using existing models and those outlined in this project, we can continuously measure our business performance. When we know we have a forecast of certain items, we can focus on either upping our inventory for those products, or cut down lead time for said products. We want to keep this effort in mind whenever dealing with any type of business data that touches supply chain or inventory levels of products. Utilizing the methodologies mentioned here, we can directly take data from existing business systems, feed our models we outlined, continuously tweak and alter our methods to get the best measures and predictions, and ultimately save our business from the "Billion dollar issue".

Let's keep in mind a very important concept. DATA SCIENCE and MACHINE LEARNING are not a foolproof solution. Any initiatives around Machine learning and data science should ALWAYS be cross verified with other solutions to ensure proper business decisions. While this is a great set of tools, to help unravel insights not readily thought of before, we need to ensure a proper understanding that Data Science is not the answer to all business problems. I leave you with a parting graph from gartner which highlights the "overhype" of some methodologies like machine learning. While it is a great tool to compliment any business initiatives, it should not be used as the only srategy when making critical business decisions.



References:

1 <https://www.lokad.com/backorders-definition>

2 <http://oregonstate.edu/ua/ncs/archives/2011/apr/managing-online-retail-stock-outs-critical-business-success>

3 <https://www.kaggle.com/tiredgeek/predict-bo-trial>

4 <https://gist.githubusercontent.com/MartinMacharia/b23c5450c72693c54b6f/raw/0a1a52625d14f228ff2e541b4998b64cbb505f64/Evaluating%2520Logistic%2520Regression%2520Models%2520in%2520R>

5 <https://www.gartner.com/smarterwithgartner/7-technologies-underpin-the-hype-cycle-for-the-internet-of-things-2016/>

About the author:



Joe Marco is an ccomplished IT healthcare leader offering over 5 years of experience driving innovation through technology in partnership with multiple business functions. Creative and dynamic thinker with proven expertise executing technology solutions to solve business problems leading to efficiency and profitability gains. Adept at working effectively to achieve goals both as a cross-functional global team member and individual contributor.

LinkedIn GitHub