Capstone_project_Report

December 5, 2017

I. Introduction

1. Description of the problem
2. Explanation of previous work that's relevant to the problem
3. Preview of your analysis

II. Data

1. Data description
   a. Nature / size of the data
   b. Data dictionary (if the feature set is large and appendix is more appropriate)
   c. Data wrangling discussion (how missing values, outliers, etc. were handled)
2. Exploratory data analysis

III. Training

1. Objective (target and loss function)
2. Features (and feature selection when appropriate)
3. Model selection

IV. Evaluation

1. Analysis of champion model (this is specific to the intended application and goals of the model)

V. Conclusions

1. Summary of main results
2. Problems / issues with analysis or results
3. Any description of intended or potential future work

1 I. Introduction

1.1 1.Description of the Problem

Part backorders is a typical supply chain problem. Backordering happens when a customer places an order for a product that is temporarily out of stock with the vendor ultimately, and order cannot fulfill. It is a dream for any business, but it is also a massive problem if you do not know how to handle it. In this project, the goal is based on the collection of data to identify parts at risk of backorder before the event occurs, so the business has time to react.

1.2 2. Explanation of previous work that's relevant to the problem

1.3 3. Preview of your analysis

With the help of this analysis, a reasonable prediction on the products that can go on backorder is expected. Such a prediction could immensely help client to plan for a more effective stocking and backorder handling.

Goals of this project are: - Provide an overall insight from data using exploratory data analysis - Identifying what features have the main effects of product backorder. - Predict Product is going to backorder or not?

2 II. Data

2.1 1. Data description

In this project, I used the a dataset available on the Kaggle website.Training data file contains the historical data for the eight weeks before the week we are trying to predict. The data was taken as weekly snapshots at the start of each week.

2.1.1 a. Nature of the data

Dataset can be found at: https://www.kaggle.com/tiredgeek/predict-bo-trial.Dataset consists of the historical data around the backorders and it has 23 columns and 1687860 entries.

2.1.2 b. Data dictionary

Dataset columns are defined as follows: Feature details: 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 three months
forecast_6_month - Forecast sales for the next six months
forecast_9_month - Forecast sales for the next nine months
sales_1_month - Sales quantity for the prior one month period
sales_3_month - Sales quantity for the prior three month period
sales_6_month - Sales quantity for the prior six month period
sales_9_month - Sales quantity for the prior nine month period
min_bank - Minimum recommended amount of stock
potential_issue - Source issue for part identified

pieces_past_due - Parts overdue from source
perf_6_month_avg - Source performance for prior six 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 went on backorder (target value).

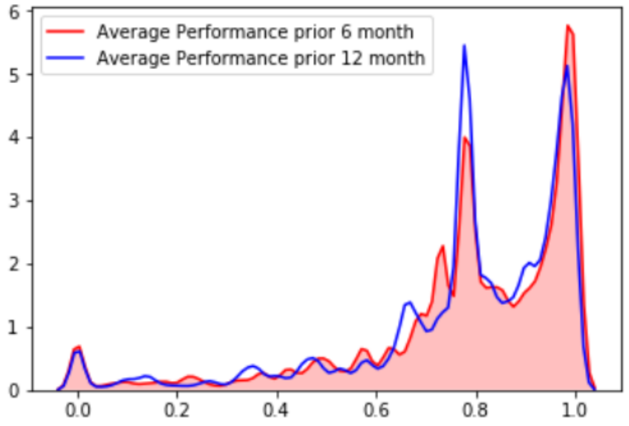
2.1.3 c. Data wrangling discussion (how missing values, outliers, etc. were handled)

Goal: Prepare the backorder dataset for EDA and Modeling Tasks performed: - Handling missing Data - convert to binary - Handling the outliers - How common are backorders? - Write the clean data into a new data frame further steps Data load and description: The data come in the form of MS Excel spreadsheets, which are easily loaded into pandas dataframes.

In this dataset, Every Categorical Feature includes only two values: ’Yes’ and ’No.’for reducing memory usage binaries were converted from strings (’Yes’ and ’No’) to 1 and 0.

Missing data:

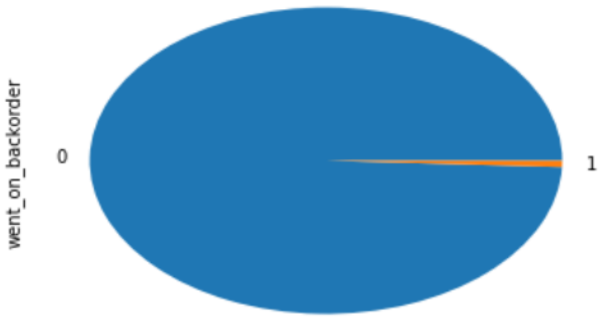
Missing values in all columns of the footer in the dataset were represented as NaN, so I dropped the footer row.columns Source_Performance_12_month and source_performance_6_month have missing valu as -99.There is a strong correlation between avg_performance_6_months and avg_performance_12_months. So, linear regression would use to filing missing values. However another interesting point to note here is that many observations have both avg_performance_12_months and avg_performance_6_months as null, so linear regression cannot fill such values, and we need to see another approach there. Probably we would like to check for the central tendency of the data and replace the null accordingly. It is visible from the seaborne plot that data did not distribute normally. Therefore picking median to fill remaining values is a good choice.



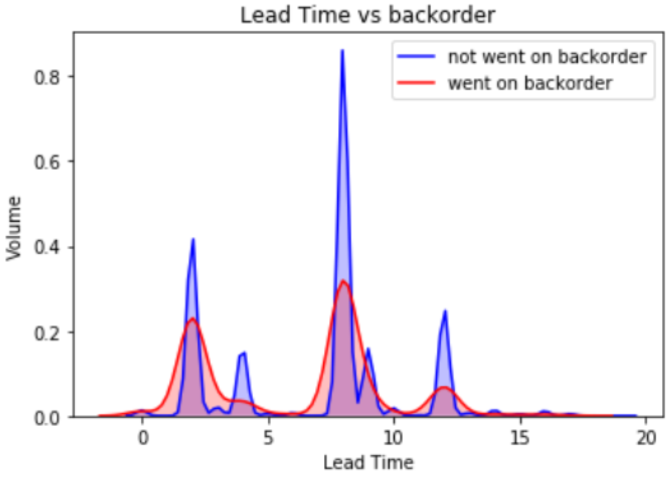
title

Lead_time column had 100893 missing values, and it was not clear if it was missing or not. It is quite likely that when lead time is missing, it is missing for a reason and not at random, which means a mean/median imputation strategy may not be appropriate. I preferred to decide by looking at data with calculating the proportion of backordered products vs. without a missing value in lead time.

The calculation below shows how to handle missing data in lead time: 1.Proportion of orders that "went_on_backorder" for missing lead_time records. 2.Proportion of orders that "went_on_backorder" for non-null lead_time records. Went on backorder ratio for orders that they went on backorder is 0.66%.



Based on the above calculations the proportion of backordered products with missing lead time is 50% less than those without missing lead time. The proportion of backordered products with missing lead time is half of the products with no missing values. The amount is significant enough that I decided not to replace the missing data in lead time and to drop them. The plot below shows the density of products for a given lead time that went on backorder and did not go on backorder.



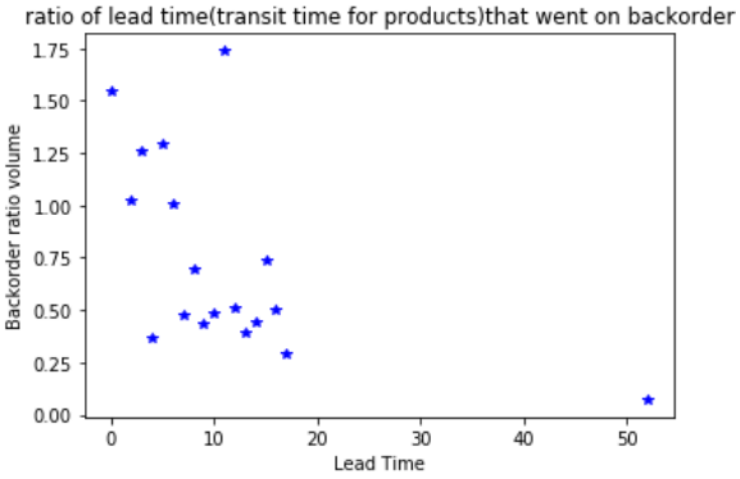
title

that plot shows that lead time of backordered orders looks exactly like the plot of lead time with not backorder data. It means that most of the products are not going on backorder and if we choose the random sample of data it is the same distribution. Therefore we are going to see if lead time and went on backorder are dependent or independent to/from each other.

Handling the outliers

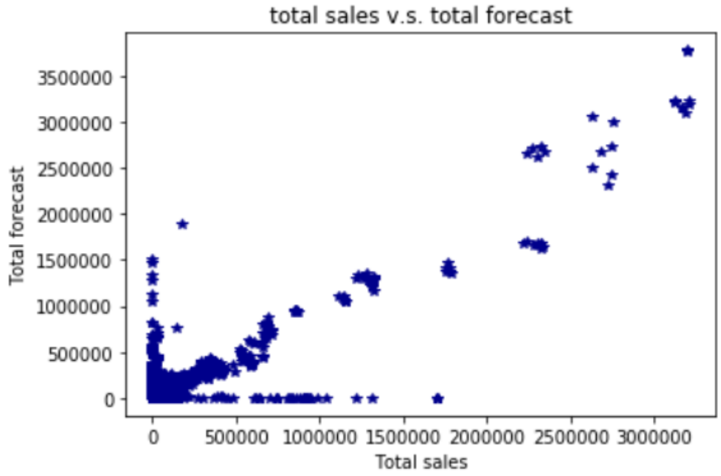
Next step is looking at the relationship between lead time and a fraction of products that went on backorder. Let's look at the lead time and how it changes the probability of went to backorder.

Products that went on backorder with lead time eight weeks and then two weeks have the highest order volumes. The plot below shows the relation between lead time and the fraction of backorder. The following plot shows with longer lead time; backorder proportion goes down. In the below plot, two outliers noticed. One is at lead time=11 and one at lead time 52. For the point on 52, I believe there were not enough records to show the rest of point between 17 to 52. The point at lead time 11 should be given particular attention to its cause is known. For this reason, I am going to calculate the binomial probability distribution. As you see from the above calculations, the standard deviation of the binomial distribution is 3.23 standard deviation from the mean, so I am going to ignore this point for now.



Sales versus Forecast

The plot below shows the strong relationship between total prior sales and total prior sales forecast. Order backorder can happen because of the wrong forecast. For example when for any reason .forecast is less than sales of the month. From 11293 backordered products, 4274 orders sales were more than sales forecast. Therefore other reasons might affect back ordering.

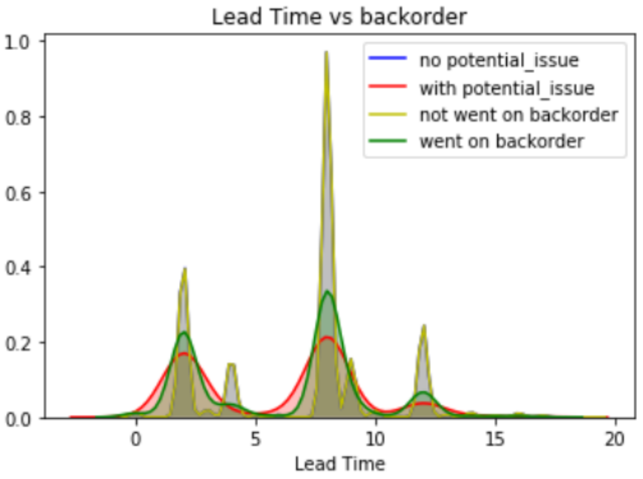


Data reduction

Since the dataset was massive, I decided to reduce data by capturing data from the total sales volume which is a significant reduction in data for not much loss of fidelity. How I captured the total sales values is I used the cumulative sum of total sales volume.For data reduction, I captured 60% total sales volume, which is data was reduced to 7397 rows. Using data reduction may save some computing time and also presenting a cleaner dataset for the predictive model. After Data reduction I looked at the relationship between sales and went on backorder. There is no relation between total sales and went on backorder now; the reason is high sales products they do not go on backorder. Backorder ratio is higher when we drop the NaN values in lead time that's because most of the orders were not backordered. There were no significant differences in the result of data reduction when I dropped missing values of lead time. In my opinion, missing values in lead time do not affect the result of volume reductions.

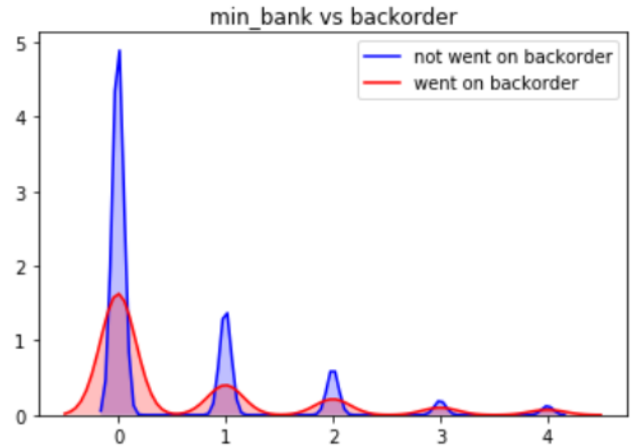
Therefore other reasons might affect back ordering. In the following exploratory I answer these questions: - Given that, how likely are backorders based on the part risk flags? - How prevalent are they? - What is the relationship between "potential_issue" and "pieces_past_due" are each - What is the relationship between "potential_issue" and "pieces_past_due" are each represented by part risk flags or are they unrelated concepts? - Based on the answers to these questions you could recommend: What aspects of the supply chain present the most prominent risks? - Based on the risks, what would you recommend improving first?

The flag columns in the dataset are: - potential_issue - Source issue for part identified - pieces_past_due - Parts overdue from source - local_bo_qty - Amount of stock orders overdue - deck_risk - Part risk flag - oe_constraint - Part risk flag - ppap_risk - Part risk flag Below plot shows that went on backorder and potential issues have the same relationship with lead time. It means when products with specific lead time did not have the potential issue the products did not go on backorder. Same thing with parts overdue; there are no parts overdue from the source the products do not go backorder.The probability of products without any of risk that did not go on backorder is almost 98%. If the product did not have parts overdue, it is doubtful it went on backorder.On the other hand The probability of product had any of risks and went on backorder is very low but the intersting part is probability of product had pieces past due,local_bo_qty,potential_issue and went on backorder is 96%. It means the combination of these flags affects the going on backorder.



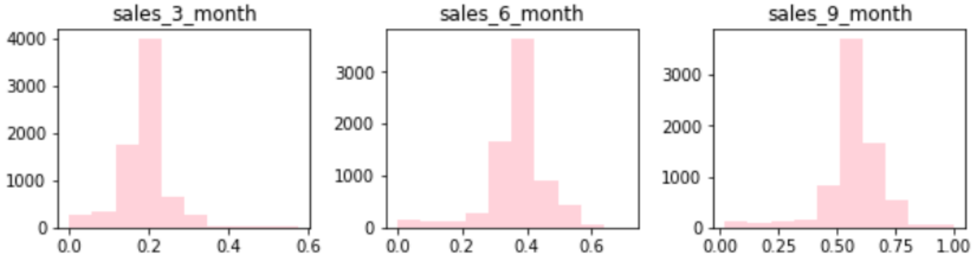
Minimum recommended amount of stock versus went on backorder

With more minimum recommended amount of stock the order volumes decrease.The proportion of orders with the minimum recommended an amount to stock that went_on_backorder: 0.66.



Normalization

Normalization is keep some valuable information about the part (for example, if inventory is lower than 0, it can correct that in preprocessing,or get misleading the models). From below plot, we can see the normal distribution for sales columns.



3 III. Training

Imbalanced classification is a supervised learning problem where one class outnumbers other class by a large proportion.This problem is faced more frequently in binary classification problems than multi-level classification problems.The reasons which leads to reduction in accuracy of ML algorithms on imbalanced data sets: - ML algorithms struggle with accuracy because of the unequal distribution in dependent variable. - This causes the performance of existing classifiers to get biased towards majority class. - The algorithms are accuracy driven i.e. they aim to minimize the overall error to which the minority class contributes very little. - ML algorithms assume that the data set has balanced class distributions. - They also assume that errors obtained from different classes have same cost. because the data set is very imbalanced so I add up the data that went on backorder to this sample.

3.1 1. Objective (target and loss function)

In this data set target value is went on backorder.

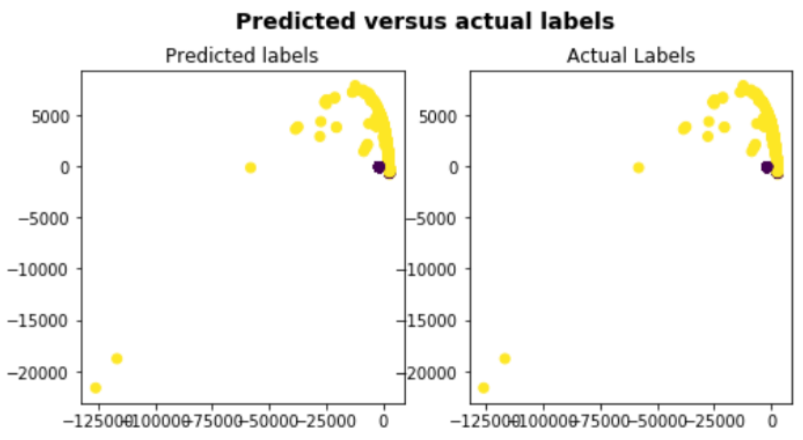
3.2 2. Features

for training data set the features I used are: {national_inv,lead_time,in_transit_qty,forecast_3_month,forecast_6_m
forecast_9_month,sales_1_month,sales_3_month,sales_6_month,sales_9_month,

min_bank,potential_issue,pieces_past_due,perf_6_month_avg,perf_12_month_avg, local_bo_qty,deck_risk,oe_constraint,ppap_risk,stop_auto_buy}

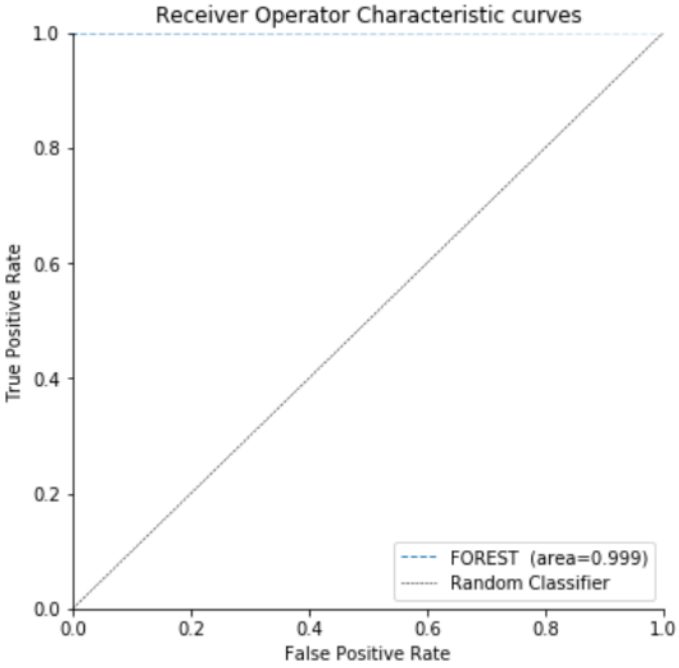
3.3 3. Model selection

Because data set is very imbalanced so I add up the data that went on backorder to this sample. I used supervised learning to predict "went on backorder" product according to what they have reordered. Result shows the accuracy of 99%. Data trained in two supervised models logistic regression, and random forest(Bagging-based ensemble).Comparing these two models in logestic regression model first because data was imbalance it showed high accuracy. That reason is, Logistic regression produces an estimated probability that a particular instance is from the positive class. It caused the classifier to over-predict positive instances. For some classifiers, it is not a significant problem, but I expect that logistic regression might be more sensitive to this mismatch between training distribution and test distribution. After balancing the data set, I used regularization with my logistic regression model and used cross-validation to select the regularization hyper-paramete to find a suitable threshold that maximizes the F1 score (or some other metric). A logistic regression model is searching for a single linear decision boundary in the feature space, whereas a decision tree is essentially partitioning the feature space into half-spaces using axis-aligned linear decision boundaries. The net effect is that it is a non-linear decision boundary, possibly more than one. This is nice when a single hyperplane does not readily separate the data points, but on the other hand, decisions trees are so flexible that they can be prone to overfitting. To combat this, I used the random forest. Logistic regression tends to be less susceptible (but not immune!) to overfitting. I used ROC AUC score since it gives the probability of an estimator ranking a positive example higher than a negative example. This way it can evaluate the models before selecting a threshold for the decision function. F1-score, for example, or other composed metrics such as geometric mean, or F2-score can be adopted. The ideal, though, would be we use the cost associate with False-positive and False-negative in the inventory system.



title

Precision-recall curves show how Precision and Recall metrics compete depending on the threshold defined for the decision function of the model. Following is the ROC curve for the case in hand.


Receiver Operator Characteristic curves
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
FOREST (area=0.999)
Random Classifier

title