Backorder dataset- Report

Contents

1. Introduction

1.1. Problem Statement

Part backorders is a common supply chain problem, wherein a customer places an order for a product that is temporarily out of stock. The percentage of items backordered and the number of backorder days are important measures of the quality of a company's customer service and the effectiveness of its inventory management.

A company can manage its inventory more efficiently using a prediction on the backorder risk for the products. Goal here is to use the past data and metadata around the backorders, and provide a prediction on the potential products for backorders.

1.2. Client

Sigma Retails Ltd is a leading online store providing products ranging from clothing, home improvements to grocery. Recently, client has been finding it difficult to manage the backorders, resulting into increasing customer issues and a decline in customer satisfaction.

Client is looking for ways to improve backorders handling. 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.

1.3. Dataset

Dataset consists of the historical data around the backorders. It has 23 features and ~ 40,000 observations.

Dataset can be found at: https://www.kaggle.com/tiredgeek/predict-bo-trial

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 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.

2. Data Wrangling

Goal: Prepare the backorder dataset for EDA and Modeling

Tasks performed:

- Handling inconsistent column names and datatype
- Missing Data handling
- Removal of duplicate rows
- Handling columns with repetitive values
- Handling the outliers
- Write the clean data into a new file for further steps

Data load and description:

Dataset had 1687861 rows and 23 columns

```
RangeIndex: 1687861 entries, 0 to 1687860
Data columns (total 23 columns):
sku                  1687861 non-null object
national_inv         1687860 non-null float64
lead_time            1586967 non-null float64
in_transit_qty       1687860 non-null float64
forecast_3_month     1687860 non-null float64
forecast_6_month     1687860 non-null float64
forecast_9_month     1687860 non-null float64
sales_1_month        1687860 non-null float64
sales_3_month        1687860 non-null float64
sales_6_month        1687860 non-null float64
sales_9_month        1687860 non-null float64
min_bank             1687860 non-null float64
potential_issue      1687860 non-null object
pieces_past_due      1687860 non-null float64
perf_6_month_avg     1687860 non-null float64
perf_12_month_avg    1687860 non-null float64
local_bo_qty         1687860 non-null float64
deck_risk            1687860 non-null object
oe_constraint        1687860 non-null object
ppap_risk            1687860 non-null object
stop_auto_buy        1687860 non-null object
rev_stop             1687860 non-null object
went_on_backorder    1687860 non-null object
dtypes: float64(15), object(8)
memory usage: 296.2+ MB
```

2.1. Handling inconsistent column names and datatype

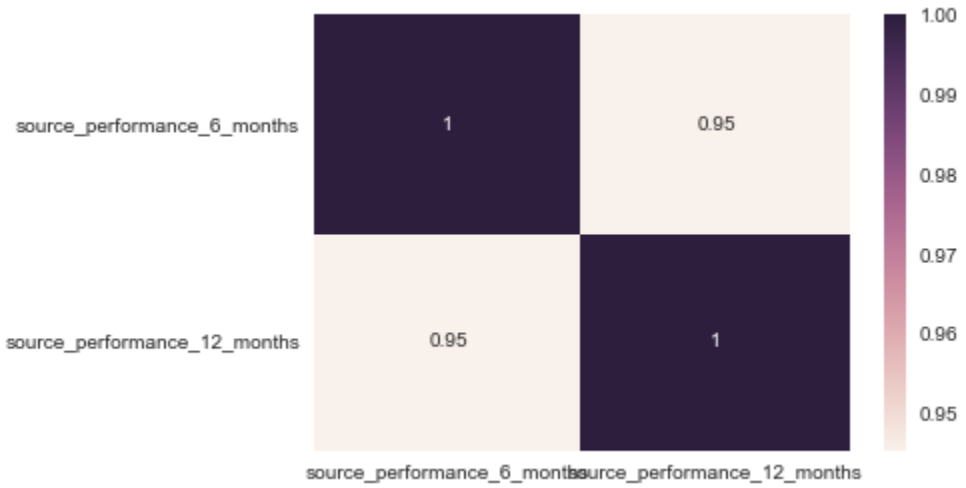
Columns were with mixed data types (was also evident from the pandas warning while loading csv). We wanted to identify and clean them up. Some of the column names had to be changed for better readability.

Many of the columns had just a single null entry. After investigation I realized that this all belonged to same row which was part of the footer .Removal of the footer solved the mixed datatype issue as well. I also changed the data type wherever applicable.

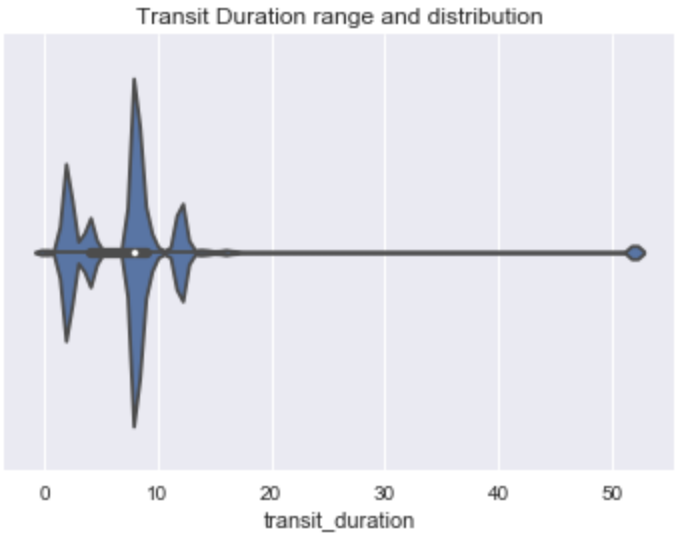
2.2. Missing Data handling

Missing values in columns source_performance_6_months and source_performance_12_months were represented as -99. I replaced -99 with NaN for the ease of processing. source_performance_6_months had 129478 and source_performance_12_months had 122050 missing values. Heatmap showed a strong correlation between source_performance_6_months and source_performance_12_months. So, linear regression is used to fill missing values. However another interesting point to note is that many observations had both source_performance_12_months

and source_performance_6_months as null, so linear regression cannot fill such values so I looked for another approach. I checked for the central tendency of the data and replace the null accordingly. It's clearly visible from violin plot that data is not distributed normally. So I picked up median to fill remaining values.

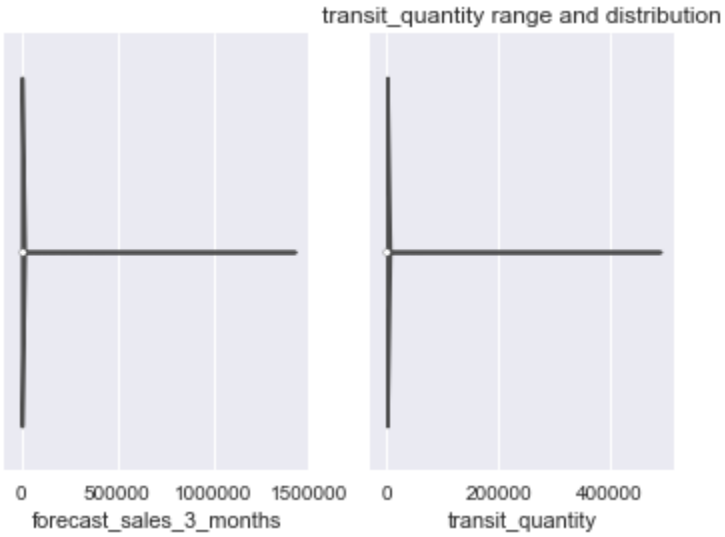


Transit Duration had 100894 null values. I didn't see correlation of this attribute with other features and also data is not distributed normally. So, again I chose median to fill the nulls.



2.3. Handling columns with repetitive values

Looking at the data set I realized there are many 0's in our dataset. So I decided to check on 0's. I took approach to drop all the columns which has more than 60% 0's.



All the below columns had more than 60% of the 0's and I removed these features from the data set.

Features	0's
forecast_sales_3_months	69.78%
forecast_sales_6_months	64.23%
forecast_sales_9_months	61.22%

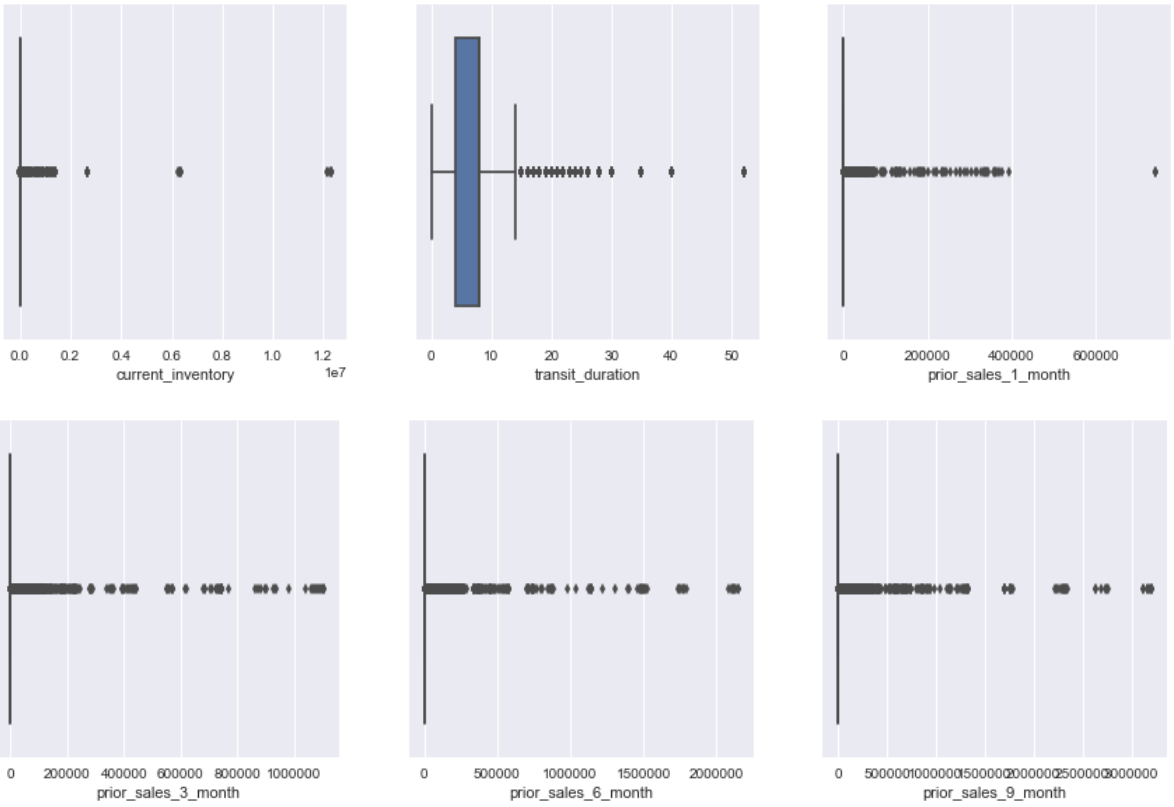
source_overdue	98.50%
stock_overdue	98.62%
transit_quantity	79.67%

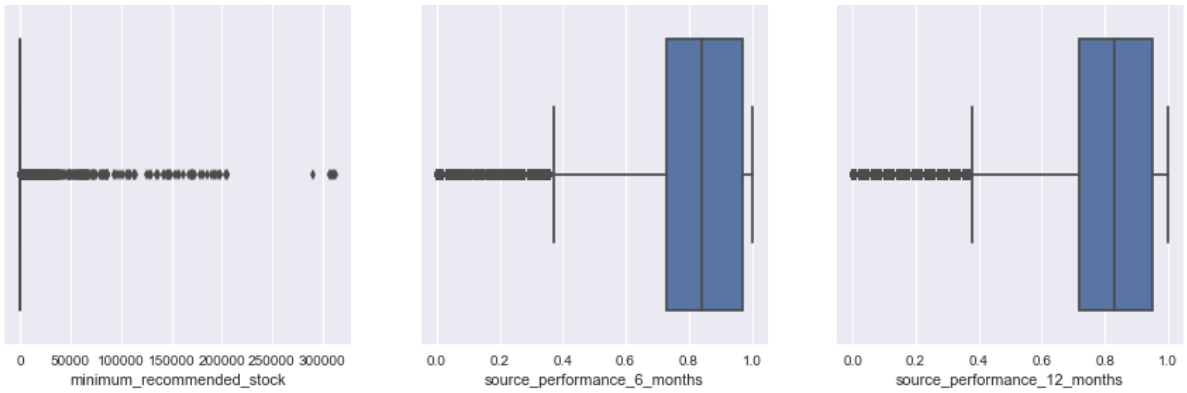
Same way we also had below categorical features which I removed-

Features	Yes	No
oe_constraint	245	1687584
rev_stop	731	1687098
source_has_issue	907	1686922

2.4. Handling the outliers

Outliers are something we need to take care at multiple phases. At this point of time I used univariate outlier detect techniques to come-up with the feature outliers. Using the Boxplot some of the outliers can be determined and handled now itself.





Below are the outlier details I handled

Feature	Condition	Outliers count	Action
prior_sales_9_month	>100,000 unit	3	Removed the observation
transit_duration	> 14 weeks	48098	Removed the observation
minimum_recommended_stock	> 210000 units	7	Removed the observation
source_performance_6_months	<.3	95591	Size is large. We will keep this as is for now and will decide during later stage
current_inventory	> 0 <211 units	225,000	Further investigation is required

2.5. Write the clean data into a new file for further steps

Finally I wrote the data to a new file Backorder_clean.csv which will be used for further analysis. Cleaned dataset has 1687829 observations and 14 features.

3. EDA and Data storytelling

Goal: To Perform EDA and storytelling on backorder dataset.

Tasks performed:

- Handling of multicollinearity
- Some interesting questions uncovered
- Relationship of categorical variables
- Inferential statistics

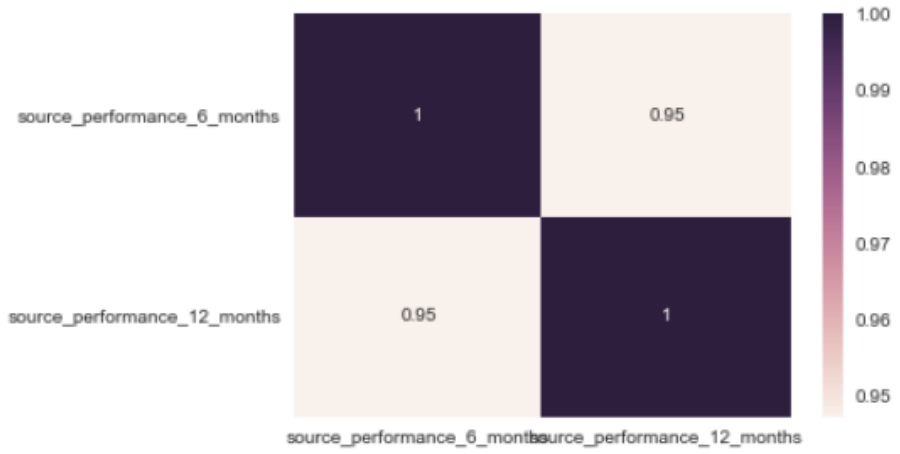
Data load and description

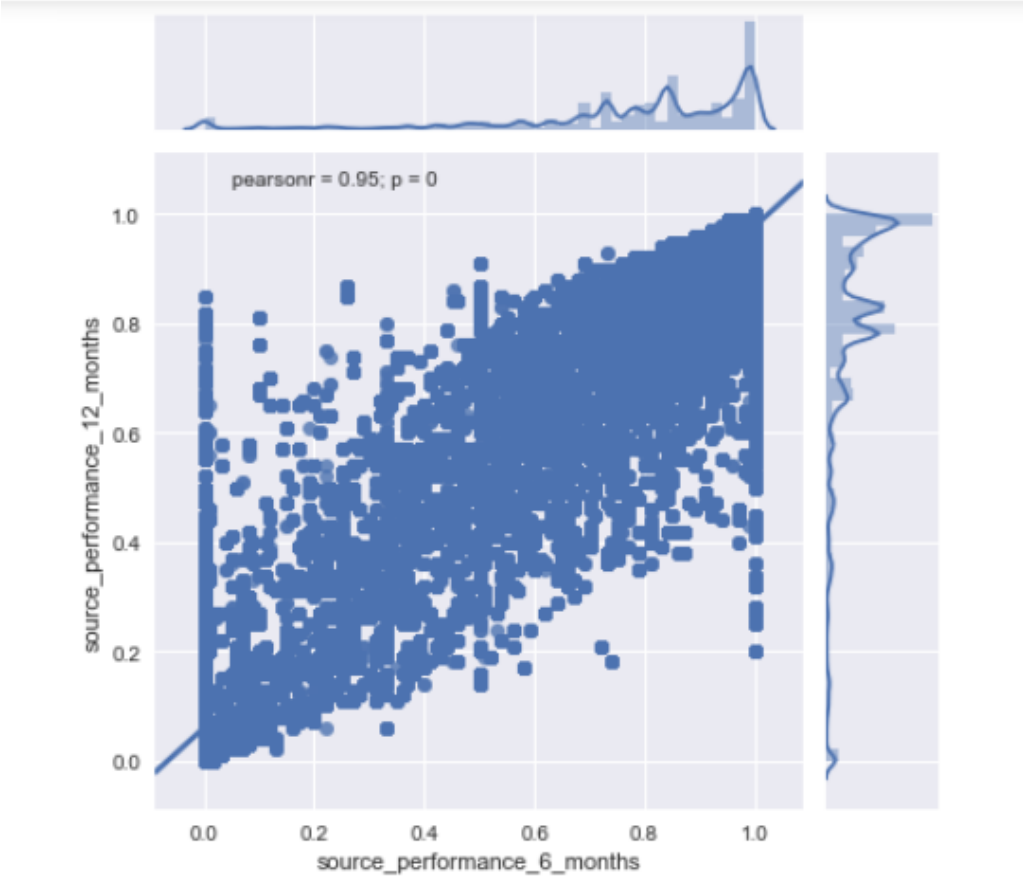
Dataset loaded here is the clean dataset which was obtained after the data wrangling process. Dataset had 1669374 rows and 14 columns

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1639734 entries, 0 to 1639733
Data columns (total 14 columns):
product_id                      1639734 non-null int64
current_inventory               1639734 non-null float64
transit_duration                1639734 non-null float64
prior_sales_1_month             1639734 non-null float64
prior_sales_3_month             1639734 non-null float64
prior_sales_6_month             1639734 non-null float64
prior_sales_9_month             1639734 non-null float64
minimum_recommended_stock       1639734 non-null float64
source_performance_6_months     1639734 non-null float64
source_performance_12_months    1639734 non-null float64
deck_risk                       1639734 non-null object
ppap_risk                       1639734 non-null object
stop_auto_buy                   1639734 non-null object
went_on_backorder               1639734 non-null object
dtypes: float64(9), int64(1), object(4)
memory usage: 175.1+ MB
```

3.1. Multicollinearity

During EDA process I found that there are many highly co-related columns. Performance over 6 and 12 month periods are rolling window and expected to be co-related. It is evident after performing the analysis as below-

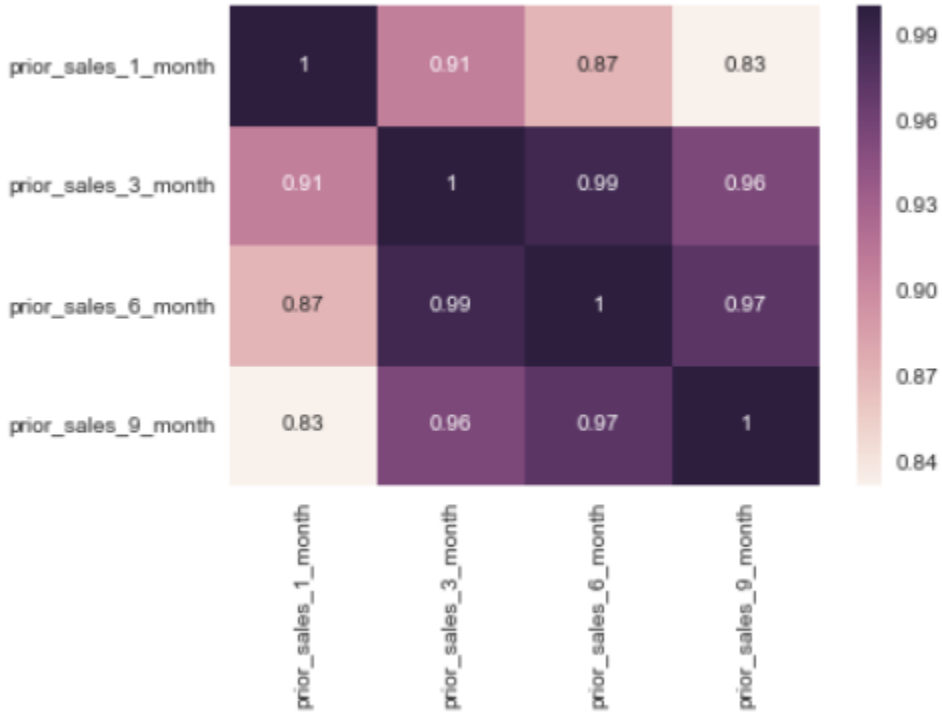


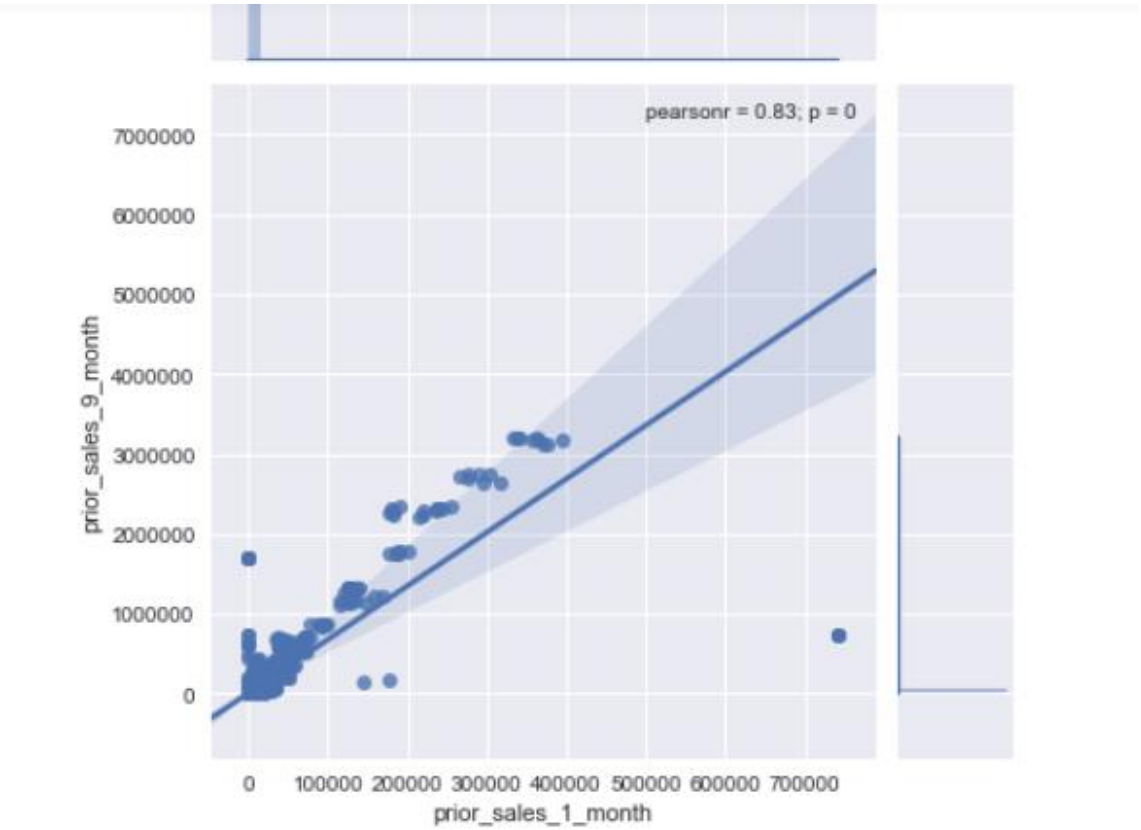


source_performance_12_months and source_performance_6_months are co-linear , so we can drop one of the feature. Since source_performance_12_months had less missing values I decided to retain this and drop source_performance_6_months column.

Same is applicable for the columns for the prior sales. These prior sales over the periods of 1, 3, 6 and 9 months are collinear. However correlation between prior sales over 1 month and 9 months is not very

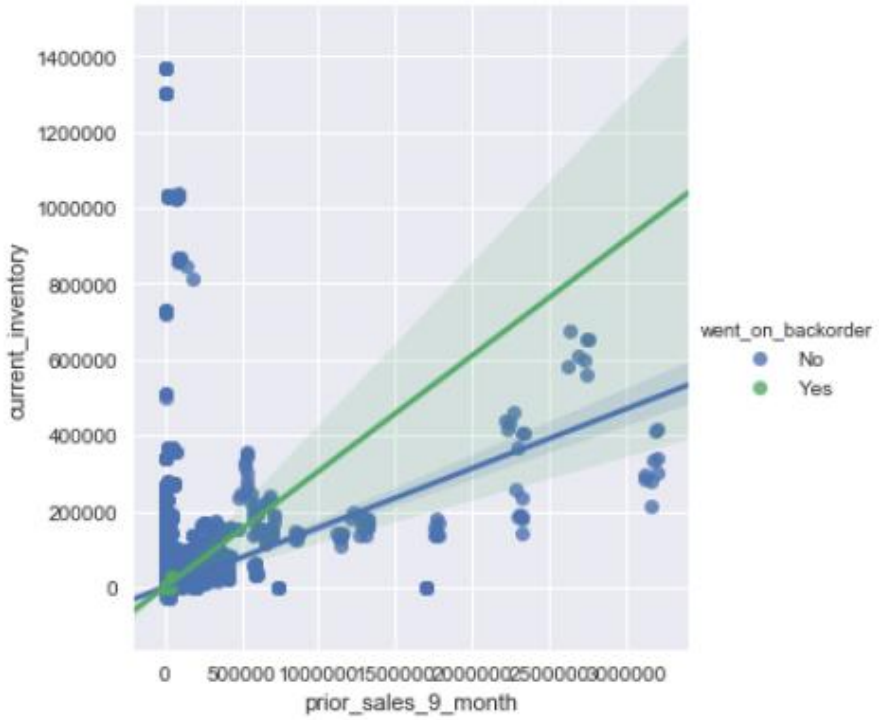
strong. So I think it would be good idea to keep these two columns and drop the columns prior sales over 3 months and 6 months.



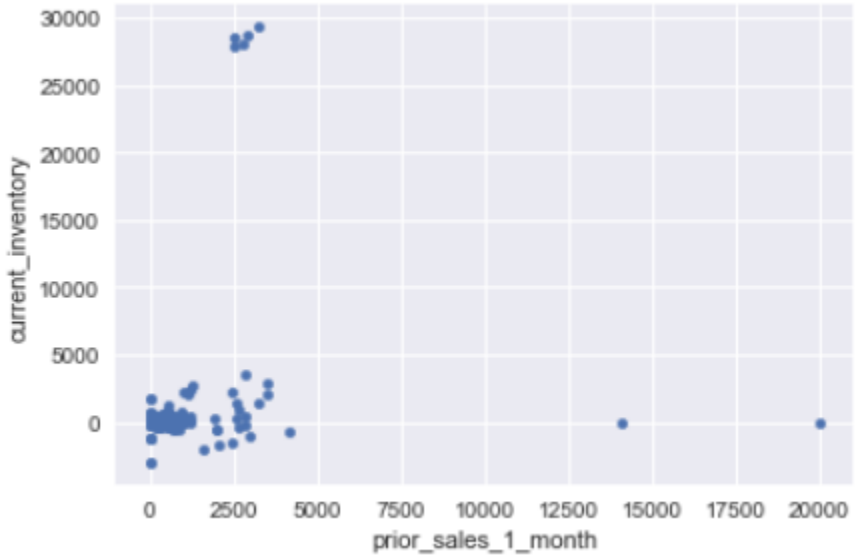


3.2. Some interesting questions

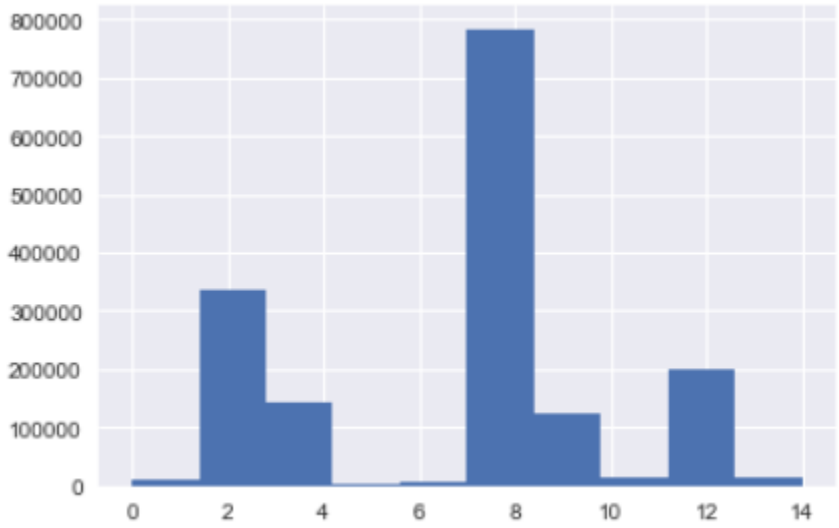
- It would be interesting to find out why the current inventory for some products is as high as 1.4 million even when there is no sale in prior 9 months. This could even be a question to client.



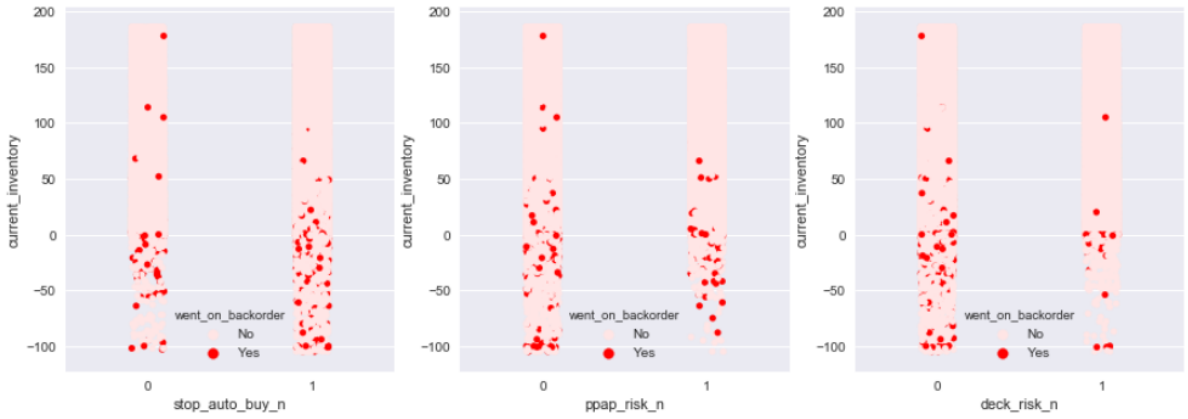
- Something unexpected - current inventory is zero for some products even the prior 1 months shows sale as high as 200,000



- Good trend for transit duration for the client. Most of the products have a short duration on transit.



3.3. Relationship of Categorical variables



I used crosstabulation and chi-square to find the relation between target variable with other categorical variables.

Contingency tables

- ppap_risk and went_on_backorder

	No	Yes
No	1217206	9291
Yes	165518	1700

- deck_risk and went_on_backorder

	No	Yes
No	1055802	9118
Yes	326922	1873

- stop_auto_buy and went_on_backorder

	No	Yes
No	35221	430
Yes	1347503	10561

Below are the calculated values

Relation	p -value	Chi2 statics	Chi2-critical
ppap_risk and went_on_backorder	3.05E-32	139.7272688	3.841458821
deck_risk and went_on_backorder	3.26E-59	263.3098726	3.841458821
stop_auto_buy and went_on_backorder	2.29E-19	80.97375064	3.841458821

All the relations have p-values less than 0.05 and we also have chi-square calculated value greater than the chi-square critical value. Based on these two evidences, I rejected the null hypothesis that variables are independent and went ahead with the alternate hypothesis.

Here we can say that went_on_backorder is related to deck_risk, ppap_risk and stop_auto_buy, so we will keep all these features for modeling.

4. Modelling

Dataset is highly imbalanced. We have only 0.6% data as 'Yes' to back order , So Accuracy cannot be a validation criteria here . Client is looking forward for **'Precision'** as validation criteria because it is important that as many of the records predicted are correct as possible so that time is not wasted working on false positives.

4.1. KNN

Since Dataset is very large, KNN model cannot be trained with the entire training dataset. I used 20% sample data to train the Model. Scaling is applied on the features. Since Target class is highly imbalance precision for this model was 0.15 only.

	precision	recall	f1-score	support
0	0.99	1.00	1.00	100645
1	0.15	0.01	0.01	625
avg / total	0.99	0.99	0.99	101270

4.2. LinearSVM

I trained the LinearSVM with 20% sample of the data. class_weight = 'balanced' is used with default features. Precession for the model was 0.01 only.

	precision	recall	f1-score	support
0	1.00	0.73	0.85	100645
1	0.01	0.59	0.03	625
avg / total	0.99	0.73	0.84	101270

4.3. SupportVectorMachine(svc.SVM)

class_weight = 'balanced' is used with default features. This Precession for the model also had 0.01 only.

	precision	recall	f1-score	support
0	1.00	0.63	0.77	100645
1	0.01	0.77	0.03	625
avg / total	0.99	0.63	0.77	101270

4.4. LogisticRegression

class_weight = 'balanced‘ is used with default features. This precession for the model also had 0.01 only.

	precision	recall	f1-score	support
0	1.00	0.63	0.77	100645
1	0.01	0.77	0.03	625
avg / total	0.99	0.63	0.77	101270

4.5. RandomForestClassifier

RandomForestClassifier had good Precession score compare to all other classifiers . Using class_weight = 'balanced‘ increased recall score but decreased precision score . So I decide to drop class weight from default RandomForestClassifier.

	precision	recall	f1-score	support
0	0.99	1.00	1.00	100645
1	0.47	0.06	0.10	625
avg / total	0.99	0.99	0.99	101270

4.6. GradientBoostingClassifier

This model also had better precision score. I chose this model to tune further to see if precision can be increased.

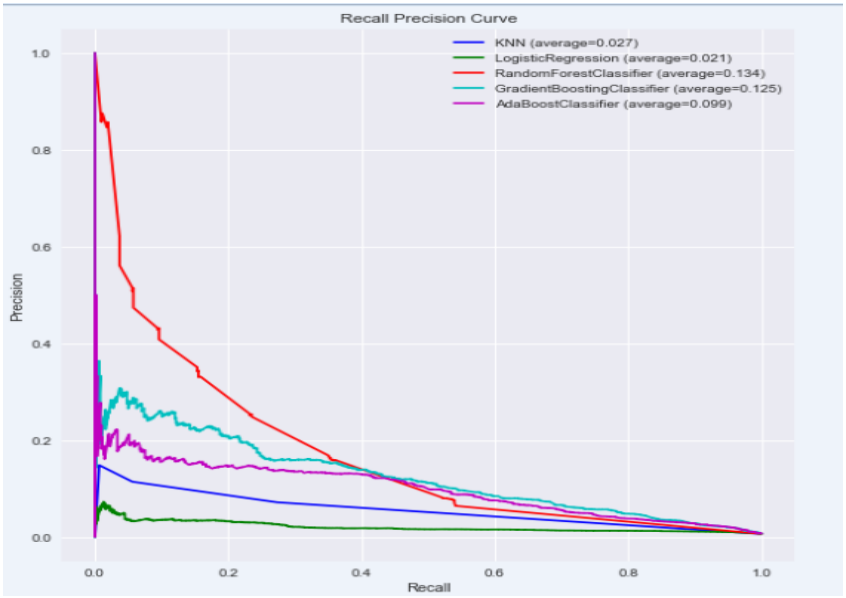
	precision	recall	f1-score	support
0	0.99	1.00	1.00	100645
1	0.26	0.02	0.03	625
avg / total	0.99	0.99	0.99	101270

4.7. AdaBoostClassifier

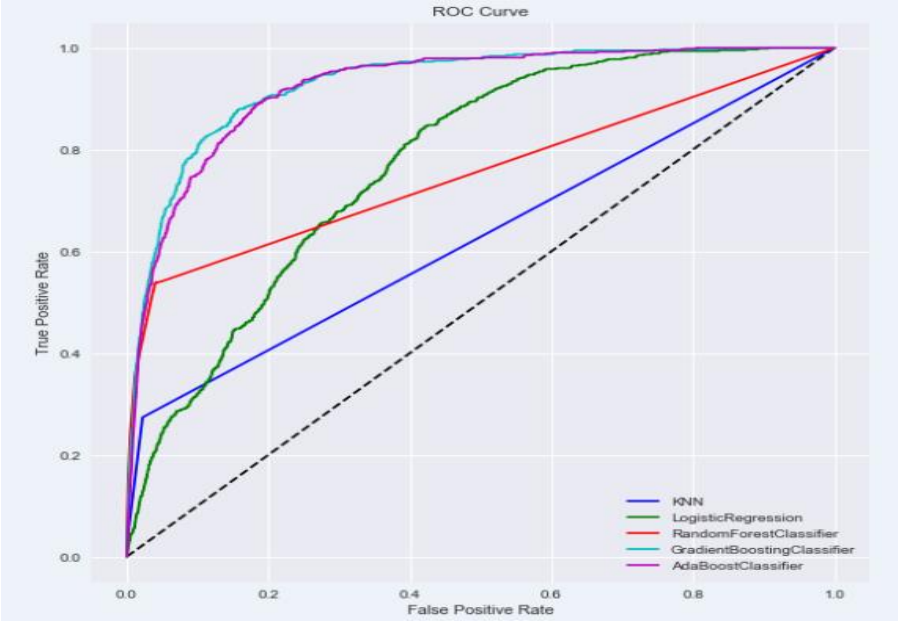
This model also had better precision score. I chose this model to tune further to see if precision can be increased.

	precision	recall	f1-score	support
0	0.99	1.00	1.00	100645
1	0.21	0.03	0.06	625
avg / total	0.99	0.99	0.99	101270

Recall Precision Curve



ROC Curve



5. Model tuning and cross validation

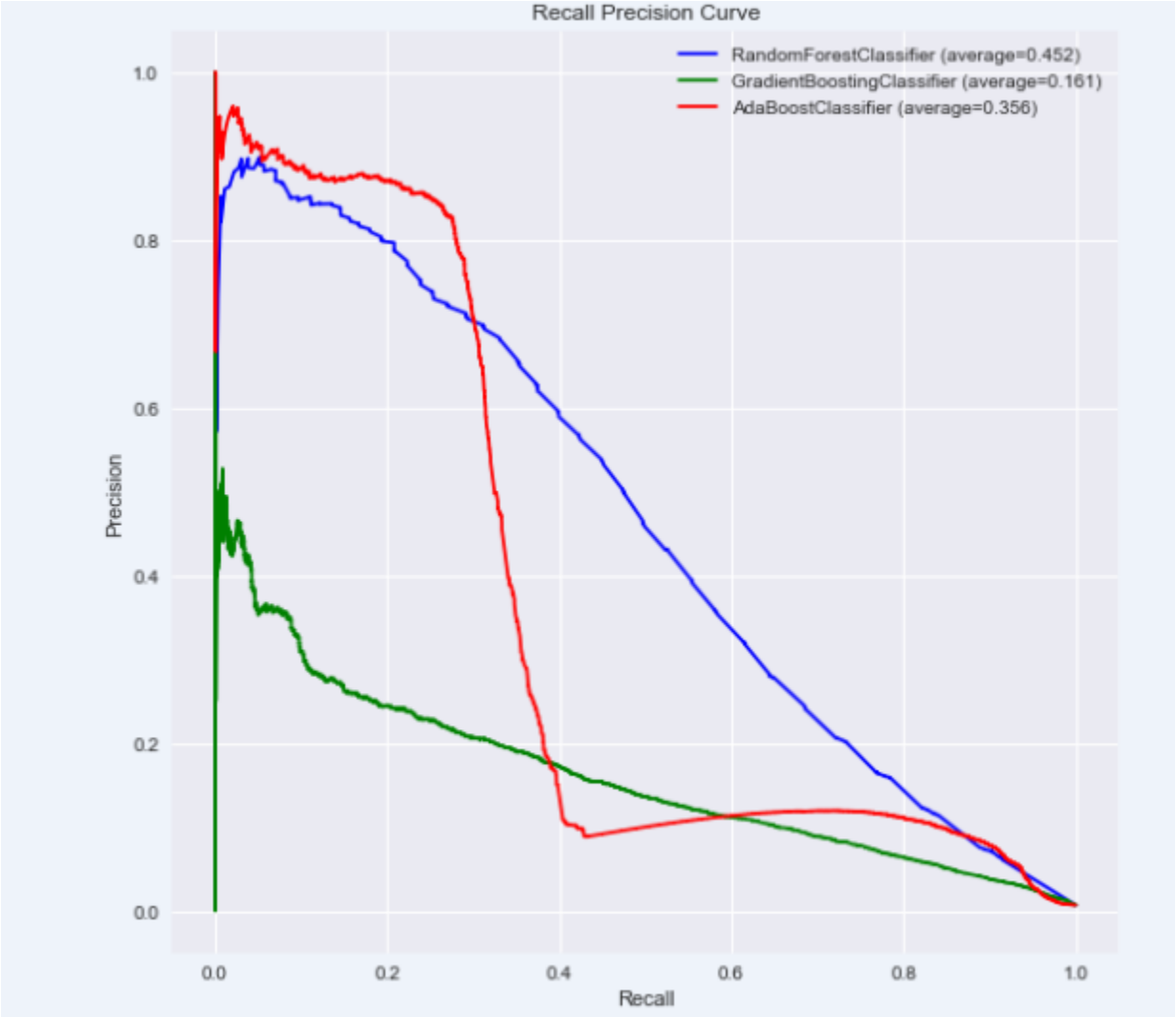
As we can see here, RandomForestClassifier, GradientBoostingClassifier, AdaBoostClassifier has better precision value. I decided to tune and validate these models

For tuning the parameters of the model, I used a mix of cross-validation and randomize search.

Model Validation: Model is trained on 70% of the data and and 30% is the test data below is the precision score on 10 folds

Estimator	Precision Score
RandomForestClassifier	0.80864198
GradientBoostingClassifier	0.567225189671
AdaBoostClassifier(randomForestClassifier)	0.812655571276

Precision-Recall curve for Tuned Classifiers



6. Conclusion

Based on the performance of all the predictive model I found Tuned AdaBoostClassifier with base classifier as tuned RandomForestClassifier is the most suitable predictive model to choose here.

I will recommend this model to client as it has precision score of .81.