HarvardX PH125.9x Data Science Capstone: Predict Backorders Project

Daoxia Ding

7/3/2022

Contents

1 Introduction **2**

2 Data Exploration and Analysis **3**

2.1 General Overview . . . 3

2.2 SKU . . . 3

2.3 Missing values . . . 4

2.4 Sales Data . . . 5

2.5 Minimum Recommended Stock . . . 6

2.6 Lead Time . . . 7

2.7 Current Inventory Level . . . 7

2.8 All Character Columns . . . 9

2.9 Correlations . . . 10

2.10 Outcome Column . . . 10

3 Model Building and Methods **12**

3.1 Class Imbalance . . . 12

3.2 kNN Model . . . 12

3.3 Random Forest Model . . . 13

4 Results **16**

5 Conclusions **18**

6 Reference **19**

1 Introduction

This project is originally from Kaggle Community Prediction Competition.
Material backorder is a very common challenge in the industries. Being able to mitigate risks of backorder help companies to enable a resilient supply chain. It not only saves millions of cost but also improve customer satisfaction and increase market share.
There are many off-the-shelf software aiming to solve the same problem. A lot of them focus on building simulation model, or simply monitoring the KPIs with threshold to notify users of upcoming risks.
With the development of computing power and advancement of machine learning technologies in the recent decades, I'd like to explore possible solutions using machine learning models.
The dataset has already been taken down on Kaggle website. However, you can still find it online, such as
https://github.com/rodrigosantis1/backorder_prediction
I have also uploaded a copy in my GitHub. Here is the link:
https://github.com/doshading/PredictBackorders/raw/main/dataset.zip
The dataset is already splitted into training set and testing set, with roughly 87% and 13% of the original data.
Below is the explanation of the data fields:

· sku -sku code
· national_inv- Current inventory level of component
· lead_time -Transit time
· in_transit_qtry - Quantity in transit
· forecast_x_month - Forecast sales for the net 3, 6, 9 months
· sales_x_month - Sales quantity for the prior 1, 3, 6, 9 months
· min_bank - Minimum recommended amount in stock
· potential_issue - Indictor variable noting potential issue with item
· pieces_past_due - Parts overdue from source
· perf_x_months_avg - Source performance in the last 6 and 12 months
· local_bo_qty - Amount of stock orders overdue
· X17-X22 - General Risk Flags
· went_on_back_order - Product went on backorder

Our target is to develop a model to predict future backorders for a specific SKU based on provided predictors. The outcome is binary.
Due to the nature of the backorder and imbalance of the classes (more details are explained in later part of this report), we will use the AUC (Area under the RoC Curve) Score to evaluate the model performance and decide the final model.
This report, following Harvard Data Science Capston course requirements, will explain the process to explore data, check data quality, clean data, develop insights, build model and eventually pick the optimal model.

2 Data Exploration and Analysis

2.1 General Overview

I start by looking at the dimensions of both data sets.
The training set includes over 1,687,000 rows and 23 columns. And the testing set includes over 242,000 rows and 23 columns.
Dimensions of the training set and testing set:

```
## [1] 1687861      23
```

```
## [1] 242076     23
```

Head of the training set:

sku	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_bank
1026827	0	NA	0	0	0	0	0	0	0	0	0
1043384	2	9	0	0	0	0	0	0	0	0	0
1043696	2	NA	0	0	0	0	0	0	0	0	0
1043852	7	8	0	0	0	0	0	0	0	0	1
1044048	8	NA	0	0	0	0	0	0	0	4	2
1044198	13	8	0	0	0	0	0	0	0	0	0

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	rev_stop	went_on_backorder
No	0	-99.00	-99.00	0	No	No	No	Yes	No	No
No	0	0.99	0.99	0	No	No	No	Yes	No	No
No	0	-99.00	-99.00	0	Yes	No	No	Yes	No	No
No	0	0.10	0.13	0	No	No	No	Yes	No	No
No	0	-99.00	-99.00	0	Yes	No	No	Yes	No	No
No	0	0.82	0.87	0	No	No	No	Yes	No	No

Head of the testing set:

sku	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_bank
1026827	0	NA	0	0	0	0	0	0	0	0	0
1043384	2	9	0	0	0	0	0	0	0	0	0
1043696	2	NA	0	0	0	0	0	0	0	0	0
1043852	7	8	0	0	0	0	0	0	0	0	1
1044048	8	NA	0	0	0	0	0	0	0	4	2
1044198	13	8	0	0	0	0	0	0	0	0	0

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	rev_stop	went_on_backorder
No	0	-99.00	-99.00	0	No	No	No	Yes	No	No
No	0	0.99	0.99	0	No	No	No	Yes	No	No
No	0	-99.00	-99.00	0	Yes	No	No	Yes	No	No
No	0	0.10	0.13	0	No	No	No	Yes	No	No
No	0	-99.00	-99.00	0	Yes	No	No	Yes	No	No
No	0	0.82	0.87	0	No	No	No	Yes	No	No

Below is the percentage of training set compared to the total set, the training set and testing set was splitted by 87% and 13%.

```
## [1] 0.8745679
```

2.2 SKU

Let's look at individual columns. First is sku. Below shows the number of unique skus in both data sets.

n_sku
1687861

n_sku
242076

I notice the number of unique sku is the same as the number of rows. Therefore I will remove sku column.

```
training_set <- training_set[,-1]
testing_set <- testing_set[,-1]
```

2.3 Missing values

Here is an overview of missing values.
Except that lead_time has a lot of missing values, all other columns have just one missing value.

```
##      national_inv         lead_time    in_transit_qty  forecast_3_month
##                 1            100894                 1                 1
##  forecast_6_month  forecast_9_month     sales_1_month     sales_3_month
##                 1                 1                 1                 1
##     sales_6_month     sales_9_month          min_bank   potential_issue
##                 1                 1                 1                 1
##   pieces_past_due  perf_6_month_avg perf_12_month_avg      local_bo_qty
##                 1                 1                 1                 1
##         deck_risk     oe_constraint         ppap_risk     stop_auto_buy
##                 1                 1                 1                 1
##          rev_stop went_on_backorder
##                 1                 1
```

```
##      national_inv         lead_time    in_transit_qty  forecast_3_month
##                 1             14725                 1                 1
##  forecast_6_month  forecast_9_month     sales_1_month     sales_3_month
##                 1                 1                 1                 1
##     sales_6_month     sales_9_month          min_bank   potential_issue
##                 1                 1                 1                 1
##   pieces_past_due  perf_6_month_avg perf_12_month_avg      local_bo_qty
##                 1                 1                 1                 1
##         deck_risk     oe_constraint         ppap_risk     stop_auto_buy
##                 1                 1                 1                 1
##          rev_stop went_on_backorder
##                 1                 1
```

It turns out the last row of the dataset was just a summary.

Therefore I can remove it.

```
training_set <- training_set[-1687861,]
testing_set <- testing_set[-242076,]
```

Now check the missing values again. It shows only lead_time now.

```
##      national_inv         lead_time    in_transit_qty  forecast_3_month
##                 0            100893                 0                 0
##  forecast_6_month  forecast_9_month     sales_1_month     sales_3_month
##                 0                 0                 0                 0
##     sales_6_month     sales_9_month          min_bank   potential_issue
##                 0                 0                 0                 0
##   pieces_past_due  perf_6_month_avg perf_12_month_avg      local_bo_qty
##                 0                 0                 0                 0
##         deck_risk     oe_constraint          ppap_risk     stop_auto_buy
##                 0                 0                  0                 0
##          rev_stop went_on_backorder
##                 0                 0
```

```
##       national_inv         lead_time    in_transit_qty  forecast_3_month
##                 0             14724                  0                 0
##  forecast_6_month  forecast_9_month      sales_1_month     sales_3_month
##                 0                 0                  0                 0
##     sales_6_month     sales_9_month           min_bank   potential_issue
##                 0                 0                  0                 0
##   pieces_past_due  perf_6_month_avg perf_12_month_avg      local_bo_qty
##                 0                 0                  0                 0
##         deck_risk     oe_constraint          ppap_risk     stop_auto_buy
##                 0                 0                  0                 0
##          rev_stop went_on_backorder
##                 0                 0
```

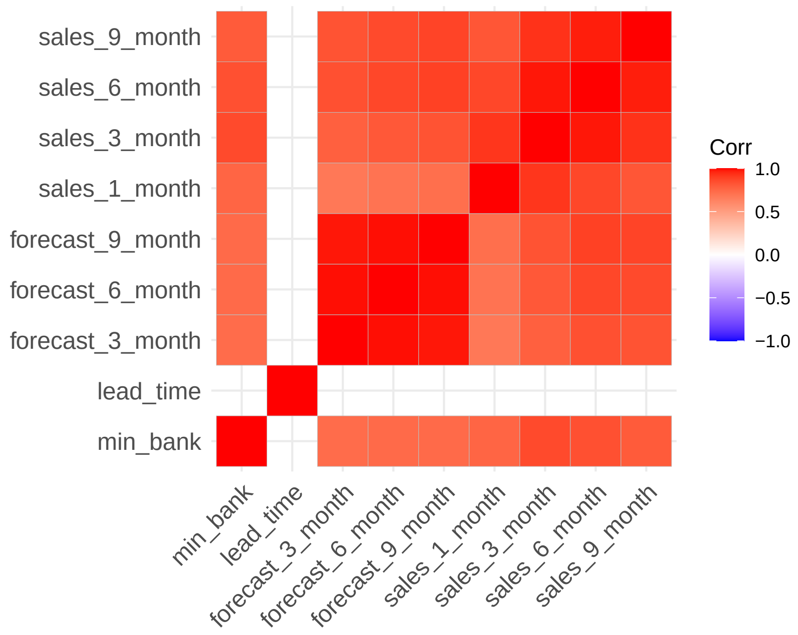
2.4 Sales Data

Here is the summary of each columns in the training dataset.
Looking at forecast sales for the next 3, 6, 9 months, I do see similar increments in between. It indicates good data quality.
I also looked at sales quantity for the past 1, 3, 6, 9 months. Similar increments can be seen in between as well. Data quality looks good.
Note both perf_6_month_avg and perf_12_month_avg (source performance in the last 6 and 12 months) have only negative values. This may be due to their IT system setup.

```
##   national_inv       lead_time      in_transit_qty     forecast_3_month
##  Min.   :  -27256   Min.   : 0.00   Min.   :     0.0   Min.   :      0.0
##  1st Qu.:       4   1st Qu.: 4.00   1st Qu.:     0.0   1st Qu.:      0.0
##  Median :      15   Median : 8.00   Median :     0.0   Median :      0.0
##  Mean   :     496   Mean   : 7.87   Mean   :    44.1   Mean   :    178.1
##  3rd Qu.:      80   3rd Qu.: 9.00   3rd Qu.:     0.0   3rd Qu.:      4.0
##  Max.   :12334404   Max.   :52.00   Max.   :489408.0   Max.   :1427612.0
##                     NA's   :100893
##  forecast_6_month  forecast_9_month  sales_1_month      sales_3_month
##  Min.   :      0   Min.   :      0   Min.   :     0.0   Min.   :      0
##  1st Qu.:      0   1st Qu.:      0   1st Qu.:     0.0   1st Qu.:      0
##  Median :      0   Median :      0   Median :     0.0   Median :      1
##  Mean   :    345   Mean   :    506   Mean   :    55.9   Mean   :    175
##  3rd Qu.:     12   3rd Qu.:     20   3rd Qu.:     4.0   3rd Qu.:     15
##  Max.   :2461360   Max.   :3777304   Max.   :741774.0   Max.   :1105478
##
##  sales_6_month       sales_9_month        min_bank          potential_issue
##  Min.   :      0.0   Min.   :      0   Min.   :     0.00   Length:1687860
##  1st Qu.:      0.0   1st Qu.:      0   1st Qu.:     0.00   Class :character
##  Median :      2.0   Median :      4   Median :     0.00   Mode  :character
##  Mean   :    341.7   Mean   :    525   Mean   :    52.77
##  3rd Qu.:     31.0   3rd Qu.:     47   3rd Qu.:     3.00
##  Max.   :2146625.0   Max.   :3205172   Max.   :313319.00
##
##  pieces_past_due     perf_6_month_avg  perf_12_month_avg  local_bo_qty
##  Min.   :     0.00   Min.   :-99.000   Min.   :-99.000   Min.   :    0.000
##  1st Qu.:     0.00   1st Qu.:  0.630   1st Qu.:  0.660   1st Qu.:    0.000
##  Median :     0.00   Median :  0.820   Median :  0.810   Median :    0.000
##  Mean   :     2.04   Mean   : -6.872   Mean   : -6.438   Mean   :    0.626
##  3rd Qu.:     0.00   3rd Qu.:  0.970   3rd Qu.:  0.950   3rd Qu.:    0.000
##  Max.   :146496.00   Max.   :  1.000   Max.   :  1.000   Max.   :12530.000
##
##   deck_risk          oe_constraint       ppap_risk          stop_auto_buy
##  Length:1687860     Length:1687860     Length:1687860     Length:1687860
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##    rev_stop         went_on_backorder
##  Length:1687860     Length:1687860
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

2.5 Minimum Recommended Stock

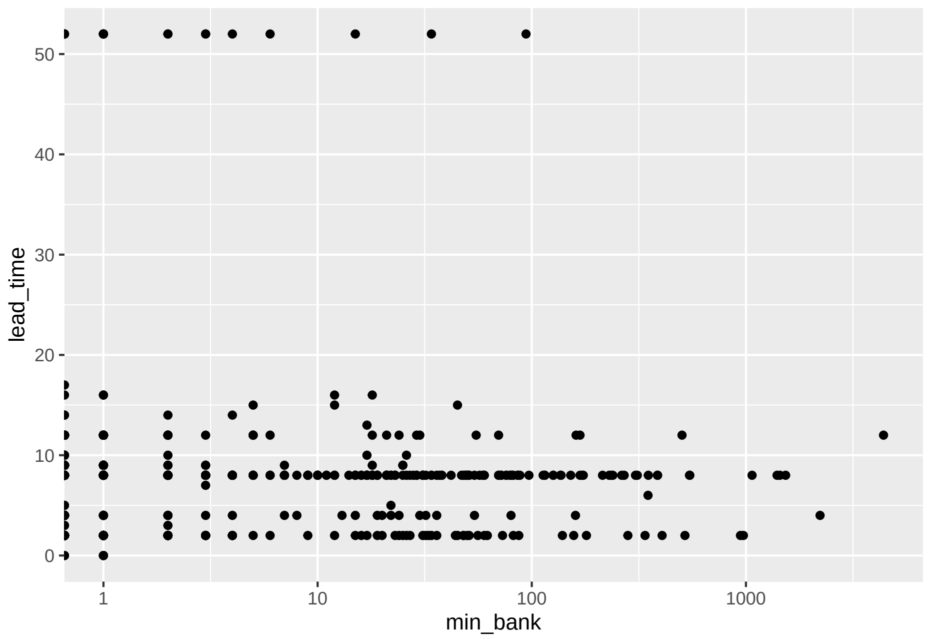
As to min_bank (Minimum recommended amount in stock), my experience in the Automotive industry tells me it has to be positively related to the lead time, sales and sales forecasting.
Below is a correlation heatmap among them. Except lead time, min_bank is correlated to all other features. This proves min_bank has good data quality.



2.6 Lead Time

Since lead_time has a lot of missing values and could not generate correlation map. I removed all missing values and tried again. Still no success.

Then I created a scatter plot (1000 sample observations) to look at the details between min_bank and lead_time. There is clearly no correlation between them. This conflicts with my industry experience.



In addition, analysis shows there are 100,893 missing values + 10,511 zero values in lead_time.

is.na(lead_time)	n()
FALSE	1586967
TRUE	100893

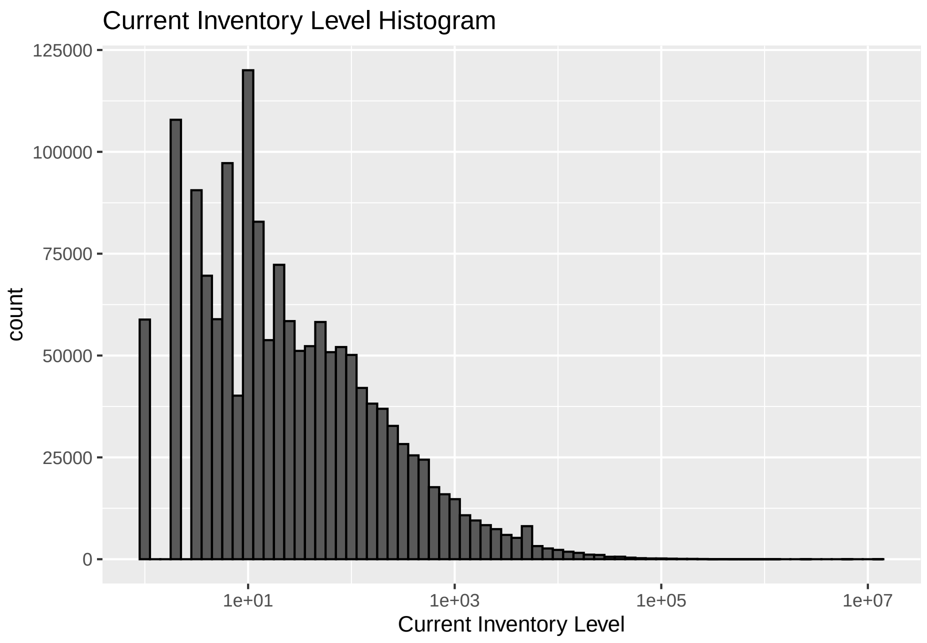
sum(lead_time == 0)
10511

At this point, I decided to remove lead_time column because the data quality does not seem good.

```r
training_set <- training_set %>% select(-lead_time)
testing_set <- testing_set %>% select(-lead_time)
```

2.7 Current Inventory Level

Here is a histogram of Current Inventory Level.



I notice national_inv has many negative values. Over 0.3% (5888) of national_inv in the training set is negative.
Negative inventory absolutely doesn't make sense. However, from my past experience of interacting with these IT systems, I recognize it could be pretty common to have negative inventory in the system.

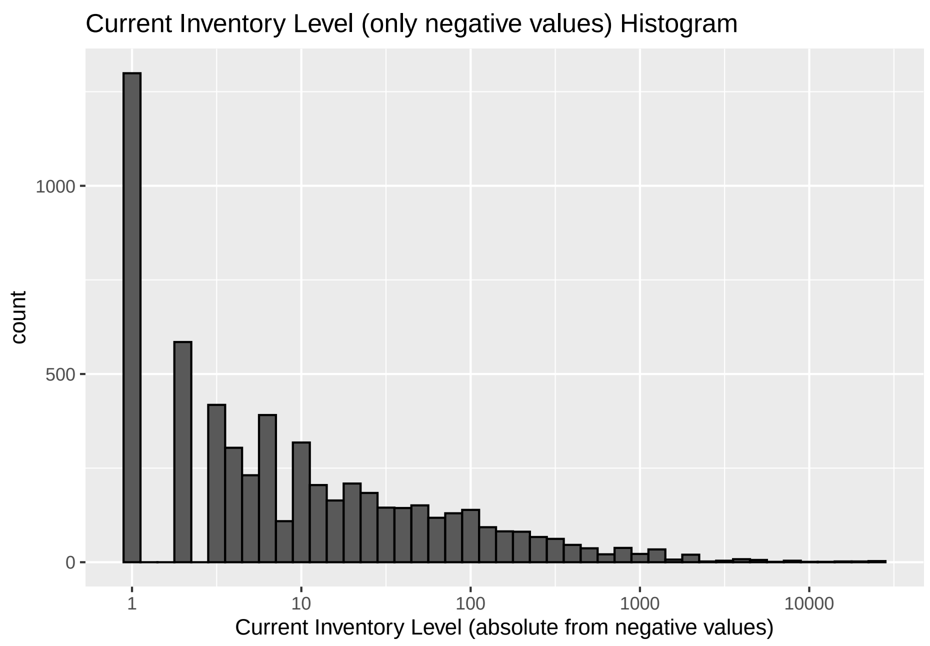
```
mean(training_set$national_inv<0)
```

```
## [1] 0.003488441
```

```
sum(training_set$national_inv<0)
```

```
## [1] 5888
```

Let's take a look only the negative values, and see how they distribute. The distribution is pretty broad.



Even though negative inventory doesn't make sense since you cannot have negative number of parts, I consider those are good indication that either the actual inventory is not properly monitored, or the whole supplier management system is not properly used. Therefore I will still include them in the model.

2.8 All Character Columns

Now let's look at all character class columns. As you can see below, all of them have only "Yes", "No" values.

```
rbind(table(training_set$potential_issue),
      table(training_set$deck_risk),
      table(training_set$oe_constraint),
      table(training_set$ppap_risk),
      table(training_set$stop_auto_buy),
      table(training_set$rev_stop),
      table(training_set$went_on_backorder))
```

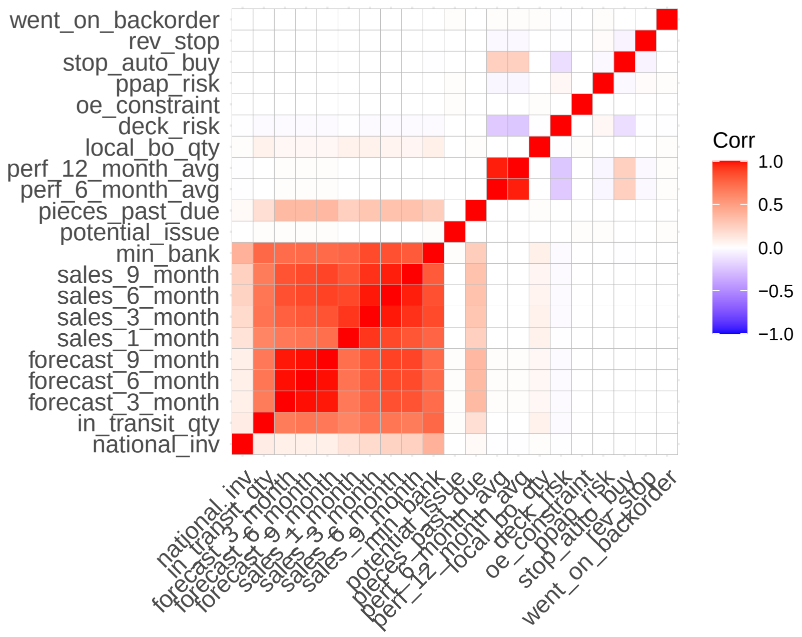
```
##           No     Yes
## [1,] 1686953     907
## [2,] 1300377  387483
## [3,] 1687615     245
## [4,] 1484026  203834
## [5,]   61086 1626774
## [6,] 1687129     731
## [7,] 1676567   11293
```

This should be converted to 1 and 0.

```
training_set <- training_set %>%
  mutate(potential_issue = ifelse(potential_issue=="Yes",1,0),
         deck_risk = ifelse(deck_risk=="Yes",1,0),
         oe_constraint = ifelse(oe_constraint=="Yes",1,0),
         ppap_risk = ifelse(ppap_risk=="Yes",1,0),
         stop_auto_buy = ifelse(stop_auto_buy=="Yes",1,0),
         rev_stop = ifelse(rev_stop=="Yes",1,0),
         went_on_backorder = ifelse(went_on_backorder=="Yes",1,0),
         )
testing_set <- testing_set %>%
  mutate(potential_issue = ifelse(potential_issue=="Yes",1,0),
         deck_risk = ifelse(deck_risk=="Yes",1,0),
         oe_constraint = ifelse(oe_constraint=="Yes",1,0),
         ppap_risk = ifelse(ppap_risk=="Yes",1,0),
         stop_auto_buy = ifelse(stop_auto_buy=="Yes",1,0),
         rev_stop = ifelse(rev_stop=="Yes",1,0),
         went_on_backorder = ifelse(went_on_backorder=="Yes",1,0),
         )
```

2.9 Correlations

Now all the data is cleaned. Let's see the overall correlation among all remaining features.



First, there is no direct correlation between went_on_backorder and any predictors. A linear model may not work well.
Second, there are clear correlations among sales forecasting, historical sales, minimum recommended stock, current inventory, quantity in transit, and parts overdue from source. This makes sense because stronger sales require more stock, more in transit, and often lead to more parts overdue from suppliers.

2.10 Outcome Column

went_on_backorder is the outcome we are trying to predict. Here I convert the outcome to factors.

```
training_set <- training_set %>%
  mutate(went_on_backorder = as.factor(went_on_backorder))
testing_set <- testing_set %>%
  mutate(went_on_backorder = as.factor(went_on_backorder))
```

A quick view shows that only 0.669% of the training data has value 1. The data is very much imbalanced.

```
##
##       0       1
## 1676567   11293
```

mean(went_on_backorder == 1)
0.0066907

3 Model Building and Methods

3.1 Class Imbalance

In order to solve the imbalance challenge, I have to either downsample or upsample the data before training the model.
Due to the fact that the training set has over 1.5 million rows, and the limited computing power of my laptop, I decided to downsample the data.

```
# set.seed(111) # if using R 3.5 or earlier
set.seed(111, sample.kind = "Rounding") # if using R 3.6 or later
training_set_down <- downSample(x=training_set[,-ncol(training_set)],
                                y=training_set$went_on_backorder,
                                list = FALSE,
                                yname = "went_on_backorder")
```

Now both classes have equal number of observations.

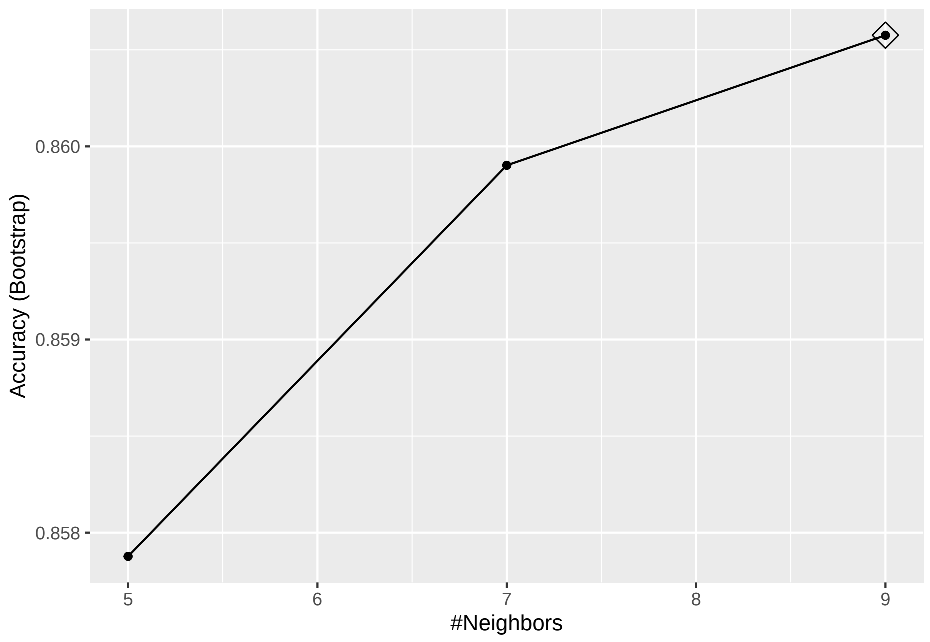
```
table(training_set_down$went_on_backorder)
```

```
##
##     0     1
## 11293 11293
```

3.2 kNN Model

Due to the large number of predictors, Generative Models such as LDA and QDA are not good options.
Classification and Regression Trees (CART) fits this situation because they are not impacted by the large number of predictors.
Before starting with Random Forest, I use kNN as the baseline model.

```
# set.seed(7) # if using R 3.5 or earlier
set.seed(7, sample.kind = "Rounding") # if using R 3.6 or later
train_knn <- train(x, y, method = "knn")
# tuning parameters chosen
ggplot(train_knn, highlight = TRUE)
```



```
train_knn$bestTune
```

	k
3	9

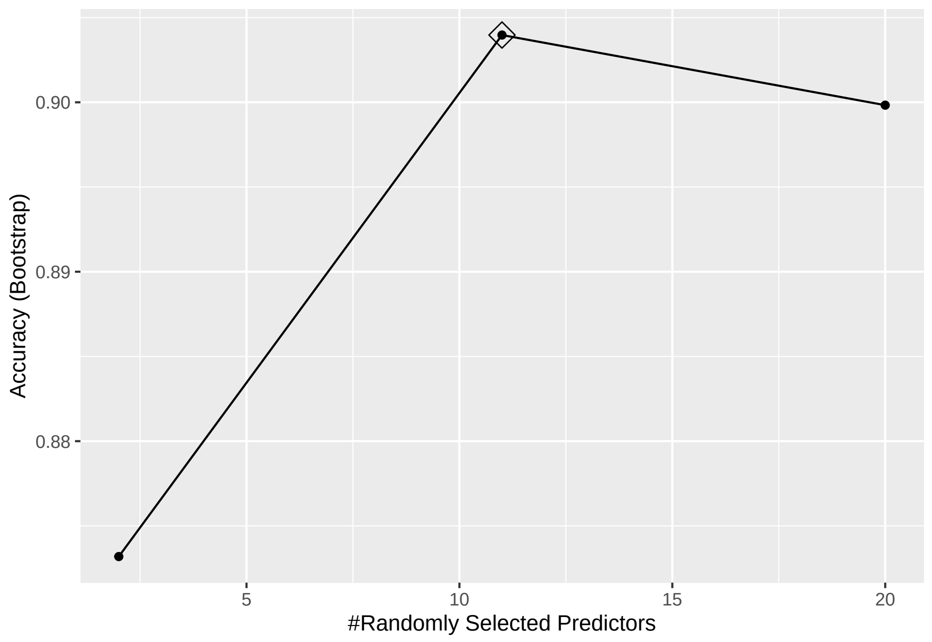
```
# Apply the fit model to the test set and see the Accuracy result.
knn_preds <- predict(train_knn, x_test)
confusionMatrix(knn_preds, y_test)$overall[["Accuracy"]]
```

```
## [1] 0.8376454
```

3.3 Random Forest Model

Let's build Random Forest Model.

```
# set.seed(7) # if using R 3.5 or earlier
set.seed(7, sample.kind = "Rounding") # if using R 3.6 or later
train_rf <- train(x, y, method = "rf")
# tuning parameters chosen
ggplot(train_rf, highlight = TRUE)
```



```
train_rf$bestTune
```

	mtry
2	11

```
# Apply the fit model to the test set and see the Accuracy result.
rf_preds <- predict(train_rf, x_test)
confusionMatrix(rf_preds, y_test)$overall[["Accuracy"]]
```

```
## [1] 0.8984736
```

Variance importance analysis shows us the current inventory level and 3-month sales forecast are the most important factors impacting backorders. They are followed up by other period sales forecasts, supplier performance and past sales.
This indeed makes perfect sense compared to real world.

```
## rf variable importance
##
##                       Overall
## national_inv        100.00000
## forecast_3_month     94.67760
## forecast_6_month     54.65739
## forecast_9_month     35.23462
## perf_12_month_avg    26.32157
## perf_6_month_avg     25.60659
## sales_9_month        19.81479
## sales_3_month        19.33887
## sales_1_month        18.70586
## in_transit_qty       17.68419
## sales_6_month        17.13949
## min_bank             12.74215
## local_bo_qty        4.07971
## deck_risk           4.06131
## pieces_past_due     3.66156
## ppap_risk           3.49835
## stop_auto_buy       1.44927
## potential_issue     0.08678
## rev_stop            0.03885
## oe_constraint       0.00000
```

4 Results

Random Forest achieved better accuracy than the kNN Model.
However, accuracy is not a good KPI here because only 0.669% of the training data is positive. It means our model can simply predict everything to be negative, and reaching 99.33% accuracy.
Instead, we need to look at Sensitivity and Specificity.
For example, the backorder will cause risks such as losing customer, market share and revenue. And the relative cost of maintaining a high inventory is low, I would pick the model with higher sensitivity.
On contrast, if the inventory and the cost of refilling a supplier order is more of a concern to the business rather than having back orders, the higher specificity model is the one to go.

```r
confusionMatrix(knn_preds, y_test)$byClass[1:2]
```

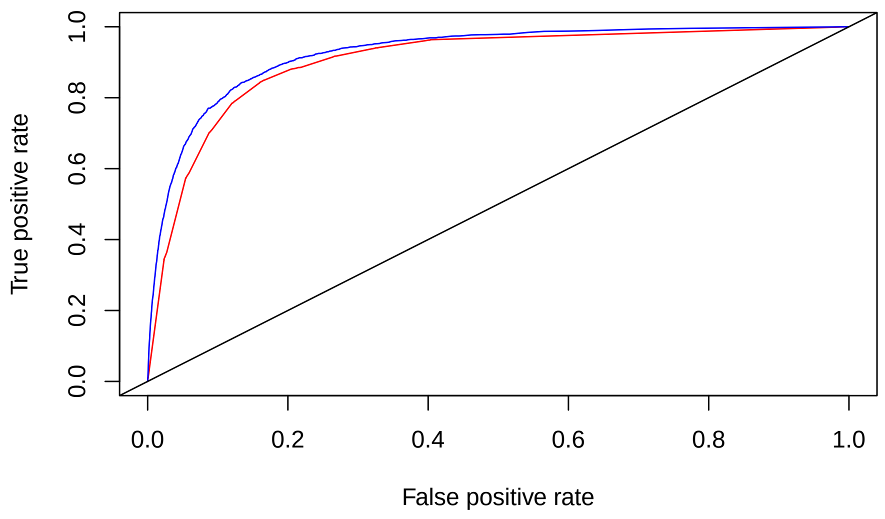
```
## Sensitivity Specificity
##   0.8375726   0.8441220
```

```r
confusionMatrix(rf_preds, y_test)$byClass[1:2]
```

```
## Sensitivity Specificity
##   0.8997105   0.7883185
```

While eventually it is up to the business to decide model based on the sensitivity and specificity, the RoC Curve can tell us how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1.
Therefore we will use AUC to decide our final model.

```r
knn_preds_prob <- predict(train_knn, x_test, type="prob")
pred_knn = prediction(knn_preds_prob[,2], as.numeric(y_test))
perf_knn = performance(pred_knn,"tpr","fpr")
rf_preds_prob <- predict(train_rf, x_test, type="prob")
pred_rf = prediction(rf_preds_prob[,2], as.numeric(y_test))
perf_rf = performance(pred_rf,"tpr","fpr")
plot(perf_knn, col="red")
plot(perf_rf, add = TRUE, col="blue")
abline(a = 0, b = 1)
```



Here is the AUC of kNN Model (Red Line).

```
## [[1]]
## [1] 0.9043403
```

Here is the AUC of Random Forest Model (Blue Line).

```
## [[1]]
## [1] 0.9264964
```

As you can see from the chart above, the Random Forest Model has larger AUC (Area under the RoC Curve) Score than kNN. Therefore for the purpose of this project, I'm choosing Random Forest Model.

5 Conclusions

After comparing the AUC Scores between the two models, I end up chose the Random Forest Model. However, it is important to note that the business eventually get to decide the model and parameters based on their unique situation - some could focus on higher sensitivity and some could weigh more on the Specificity. In addition to choosing the preference, it is also important for the the business to start using this model to drive their daily business asap in order to realize business value.
Meanwhile, there are also limitations on this model. Due to the limited computing power on my laptop, I was unable to tune additional values for the parameters except the default ones. All the values of the remaining predictors are included in the model without any data quality check due to the lack of Subject Matter Experts' input on the context of these data.
Looking forward, I would like to get input from the users of the IT system behind this data. Their knowledge would help me to further reduce the number of predictors, remove predictor overlap and improve data quality. I would also consider upsampling methods such as SMOTE and see how the model performance change. Matrix factorization would be another method to consider as well.
I hope to obtain additional data fields from their IT system such as the time stamp, location and so on. I can then integrate 3rd party data such as weather and traffic to enhance the dataset and improve model performance.

6 Reference

Kaggle Community Prediction Competition: Predict Product Backorders
https://www.kaggle.com/c/untadta/data

rodrigosantis1, 2019, Data sets link for Predict Product Backorders
https://github.com/rodrigosantis1/backorder_prediction

saaiswethasret, 2022, How to Create Correlation Heatmap in R
https://www.geeksforgeeks.org/how-to-create-correlation-heatmap-in-r/

Modeling with R, 2019, Methods for dealing with imbalanced data
https://www.r-bloggers.com/2019/04/methods-for-dealing-with-imbalanced-data/

Joseph Rickert, 2019, Some R Packages for ROC Curves
https://rviews.rstudio.com/2019/03/01/some-r-packages-for-roc-curves/

DataTechNotes, 2019, How to create a ROC curve in R
https://www.datatechnotes.com/2019/03/how-to-create-roc-curve-in-r.html

Sarang Narkhede, 2018, Understanding AUC - ROC Curve
https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5