# Examining the Indicators of Product Backorders


**Group 33:**
**Timothy Hill**
**Suraj Shah**
**Avinash Sooriyarachchi**

**STAT 571/701 Final Project**

**December 13, 2017**

# Table of Contents

## Abstract

The goal of this project was to leverage our knowledge of data science and machine learning to predict retail product backorders based on a number of criteria. We analyzed a dataset with over 240,000 different observations and 22 variables which was obtained from Kaggle, a website well known in data mining circles for the rich variety of data sets it offers. This consists of historical data collected over an eight week period, with the variable we attempt to predict being whether or not the item went on backorder, which is given as a categorical variable. The key independent variables were of types categorical, numeric and integer.

The initial step was to perform exploratory data analysis on the data set to achieve our goal of predicting the backorder status. The data set was cleaned and restructured such that each variable was of a data type compliant with the machine learning approaches resorted to gain insight from the data. In particular, the 'SKU' variable which is a unique identifier for each item in the data set was omitted from the analysis as no insight could be gained from this. Due to computational limitations, a random data set consisting of 10,000 samples was extracted from the cleaned data set for further analysis.

The data set obtained following exploratory data analysis was subjected to a number of supervised learning techniques such as logistic regression, LASSO, Random Forests, Neural Net and SVM (Support Vector Machines), as well as unsupervised learning techniques, namely Principal Component Analysis (PCA). These shed invaluable insight into the several variables playing a key role in predicting the backorder status, namely lead time, minimum recommended amount of stock, and current inventory level of the particular product.

# Retail Industry Overview

## *The Advent of E-Commerce*

The retail industry is one of the largest and most vital consumer-facing industries in the world. In 2015, worldwide retail sales amounted to over $22 trillion.1 In recent years, the industry has undergone significant transformation with the advent of e-commerce, proliferated by a variety of firms, but most notably Amazon.com. E-commerce accounted for over $2 trillion of retail sales in 2015, and is projected to account for $4 trillion in sales by 2020.2

The rise of e-commerce has drastically changed the competitive dynamics of the retail industry. Rather than traditional brick-and-mortar stores competing with each other in terms of price, brand name, and quality, these retailers are now forced to compete with e-commerce retailers as well, which poses major difficulties with regards to cost and margins. Amazon in particular has mastered the use of direct sourcing and economies of scale to keep their costs low and ship millions of products directly to consumers within a couple of days. The ability to order items instantaneously from a mobile device has altered the consumer mindset and greatly raised their expectations of what retailers must provide. Additionally, the consumers of today value convenience and price more than brand loyalty.[3]

Due to this rise in customer expectation levels and the increase in the ease of availability of most products, retailers must re-evaluate their business model to ensure they are operating efficiently by keeping costs low. Retailers are beginning to understand the need to operate efficiently and many firms have begun to analyze data to determine where improvements can be made to their business model. In many cases, inventory management is a key area where retailers can make improvements to their cost model. For example, one leading footwear company has linked its inventory systems to determine which pairs of a particular shoe are least likely to sell at full price and then determine whether it is more cost-effective to ship that pair of shoes to a consumer or let them be sold at their current store at an expected markdown price.[4]

---

[1] Farfan, B.

[2] Lazar, M.

[3] Bain & Company

[4] MacKenzie, Ian, et al.

## *Supply Chain Effectiveness*

It is important for retailers to continue to gather data and generate data-driven insights to improve their operations, particularly in the area of inventory management. Inventory management is crucially important and is considered one of the three key ingredients of supply chain effectiveness, along with a flexible supply chain (i.e. adaptability) and accurate forecasting, as summarized in Figure 1 below[5]:

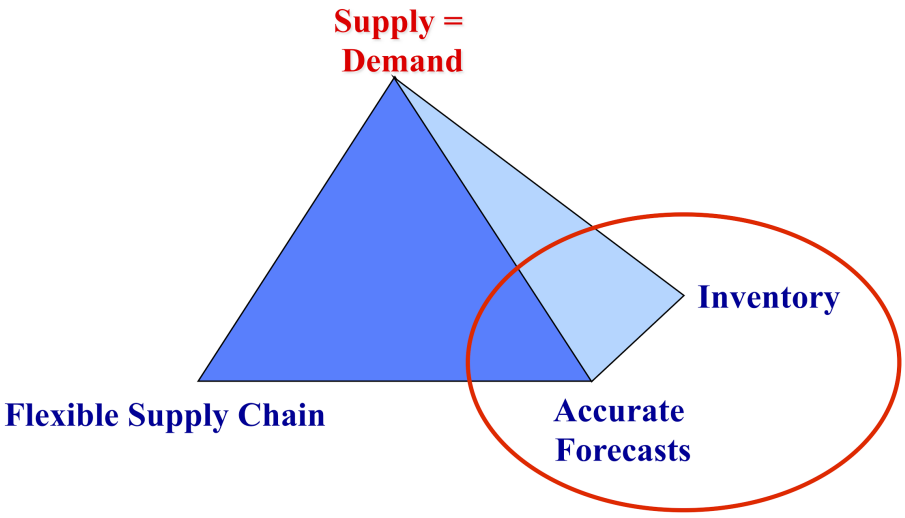



*Figure 1: The three ingredients of supply chain effectiveness*

For the purposes of this project, we are focusing on the latter two of these ingredients: inventory and forecasting, which are closely related.

Inventory management has been a key point of emphasis in the retail industry for decades, since it impacts the supply chain at multiple levels, from the manufacturing plant to the distribution center(s), to each individual store. Figure 2 below illustrates this concept[6]:

[5] Fisher, M.

[6] Fisher, M.

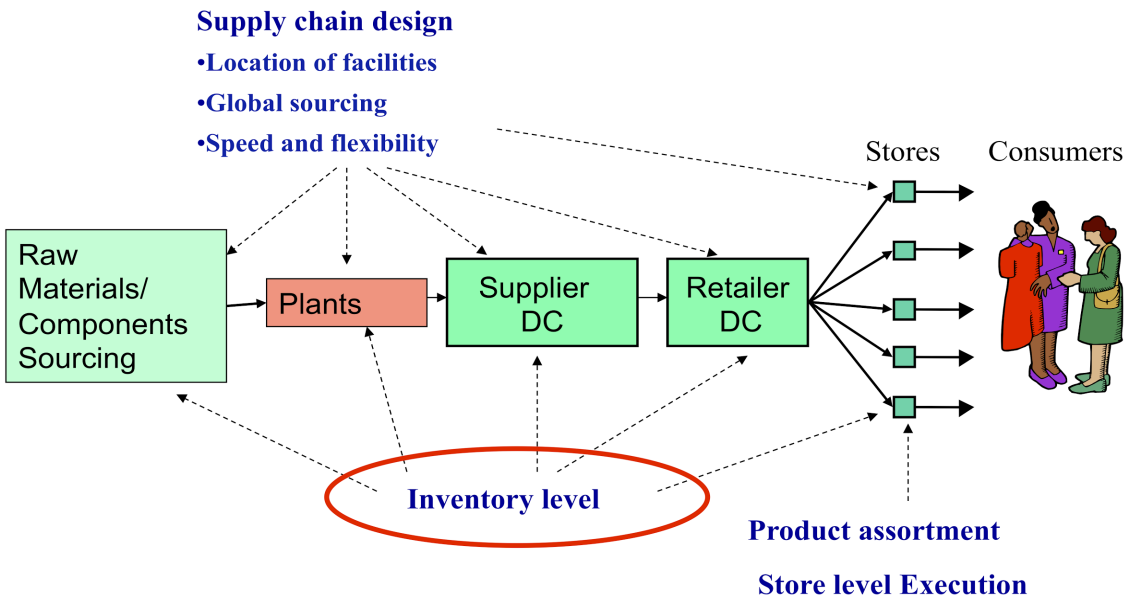



*Figure 2: Retail supply chain levels and the impact of inventory management*

We see that inventory must be effectively managed at each step of the supply chain to ensure that the supply chain remains optimal since suboptimal inventory management at one step of the supply chain can be detrimental to inventory at other steps. Suboptimal inventory management generally comes in two forms - overstocking and understocking. Overstocking refers to retailers ordering too large of a quantity of a particular product, leading to an excess supply left over at stores and/or distribution centers after demand has subsided. In this instance, retailers must sell this product at a markdown in regular retail locations or at factory outlet stores, greatly reducing the profit margins for the product. While this is certainly an interesting topic, for the purposes of this paper we are focusing solely on the other form of suboptimal inventory management, understocking.

## *Backorders and Inventory Forecasting*

Understocking refers to the situation in which retailers order too small of a quantity of a particular product, leading to excess demand and a limited supply. Understocking often leads to backorders, which is when inventory is exhausted for a particular product while consumer demand remains high. When an item goes on backorder, consumers must wait days or even weeks for their orders to be fulfilled. With the advent of e-commerce, backorders have become even more costly to retailers. Since consumers greatly value convenience and price over specific brand loyalty and substitute products are readily available from numerous competitors, it is likely that consumers will purchase products from a competitor instead of waiting days or weeks for their order to be fulfilled in the

event of a backorder, which causes the original retailer to lose all the potential revenue from that consumer.

Thus, it is clear that backorders are potentially very costly events for retailers and must be minimized wherever possible. Minimizing backorders can be achieved through accurate inventory forecasting methodologies. If forecasting is performed correctly, retailers can predict ahead of time when a backorder is likely to occur and reorder a product to replenish stock before running out of the product.

Inventory forecasting is a complicated process that must be undertaken multiple times throughout a product's lifecycle. Retail chains must first perform initial forecasting at both the company (chain) level and at the individual store level before placing the initial order for a new product. This initial forecast is based primarily upon experts' forecasts and historical data for similar products. After the produce is introduced in stores, retailers must update their forecasts based on initial sales and reorder the item if necessary. This reordering process must be done ahead of time as sufficient lead time is necessary to ensure that the reorder is received by stores before a backorder situation occurs. This cycle of reforecasting and reordering continues until a product nears its end of life, at which point it is marked down and sold at a discount to clear out any remaining inventory. This process is summarized in Figure 3 below[7]:

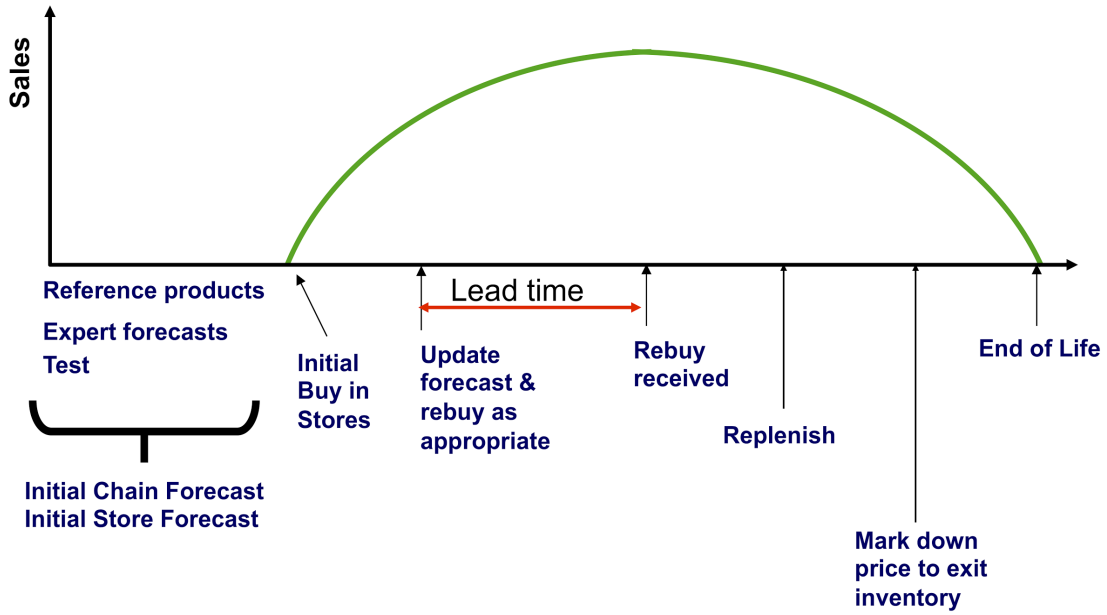



*Figure 3: Product life cycle and inventory forecasting stages*

[7] Fisher, M.

While forecasting follows the general cycle outlined above, forecasting methodologies vary significantly by firm. There are seemingly endless proposed methodologies to generate the optimal forecasting methodology. Various mathematical models have been proposed in the past to identify the best times to place reorders by assigning penalties to backorders and examining the inventory management problem through the lens of game theory. Dr. Gérard Cachon (now a professor at the Wharton School) explored this problem in a paper he wrote while teaching at Duke University's Fuqua Business School in 1998 modeling the inventory management problem as a two-stage supply chain game with an optimal Nash Equilibrium.[8]

While our project also focuses on the same issue of inventory management and forecasting to minimize backorders, we are approaching the issue from a different angle by examining data to understand what factors may be key predictors backorders to occur. By examining the underlying causes of backorders, retailers can track these key metrics to generate insights, optimize their inventory management systems, and ultimately reduce the incidence of backorders in their supply chain.

## Dataset Overview

For the purposes of this project, we are using a dataset obtained from Kaggle.[9] This dataset includes historical data that can be used to predict backorders and examine key indicators of backorders. The data is aggregated from various retailers but retailer and product names have been removed to protect anonymity. Despite the fact that this dataset is generic, the outcomes of our analysis could be useful to retailers with access to their full data. The dataset includes the past eight weeks of sales data and attempts to predict sales and backorder probabilities for the upcoming week. The full list of variable names in the dataset is as follows:

*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

[8] Cachon, G. & Zipkin, P.
[9] Tiredgeek

*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.

## EDA and Data Visualization

We began by performing some simple exploratory data analysis (EDA) on this dataset to ensure that it was cleaned and ready for us to begin developing a model. We found that dataset consisted of 242,076 observations with 23 variables. Of these, our main variable of interest is "went_on_backorder." We deem an item on backorder as a failure of the supply chain. The goal of the project is to create a classification model to help identify the factors that best predict when an item will likely go on backorder.

With regards to the dataset, there aren't too many problems with the data set, however, if we were to move forward and include the sku variable we would be unable to run any sort of analysis on the data. The sku variable is a factor with 240,000 levels and does not add any value to the data (it simply defines each individual inventory item). Since the dataset does not include any information on how to decipher the sku variable, we have no way of understanding which product a sku corresponds to. Including the sku variables greatly increases computational expense and thus we will omit it.

Another potential problem in the data is that the lead_time variable has N/A values. There are 14,725 observations with N/A lead times. Given that lead time is likely an important factor on whether a part is backordered it may be best to remove these values from the dataset and preserve the variable itself. We will leave them in for the time being. There are a total of 14,739 N/A values in the data set, and with 14,725 of them occurring in the lead time variable it shows that almost all N/A values occur here.

The rest of the N/A values are spread out amongst many other variables. There is not a significant amount.

As previously mentioned, our column of main interest is went_on_backorder. This factor has 3 values, "No" and "Yes." 1.1% of all sku's went on backorder. The factor is also flawed in that there is a blank value. The blank value occurs in the last row in the data set. We will remove that row from the data set and refactor the variable. Now we eliminate the NA values from the dataset for easier analysis. Since so few items were NA, and there seems to be no specific reason for the NA values, we will choose to ignore them. The data is now prepared for more exploration, however to better be able to maneuver through the data we will limit its size. With 240k rows of data it is simply too unwieldy in its current state. As such we are taking a sample of 10,000 rows to use going forward.

We find that the correlation between the variables sales_6_month and sales_1_month is 0.949, which is not to be unexpected as they are similar variables. We will also generate a correlation heatmap to get a better view of all variables in the sample. The heatmap can be found in Figure 4 below:

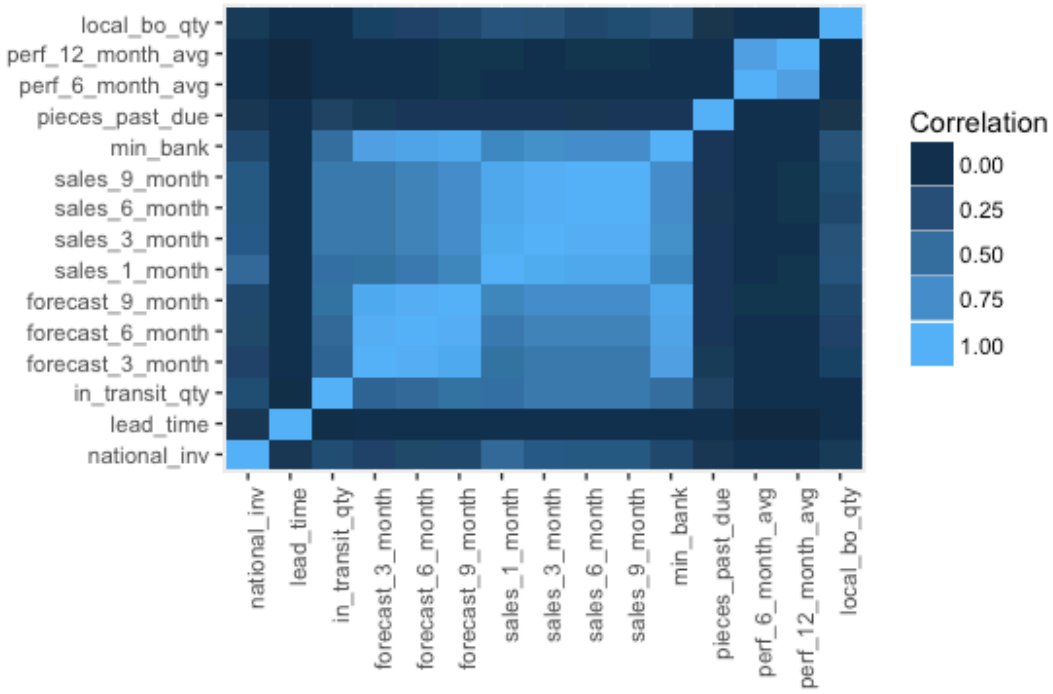



*Figure 4: Correlation heatmap of all variables*

Again we see that sales and forecast are highly correlated in their monthly increments. We will keep this of note for our final models. Now that we have performed EDA and examined our variables, we can move on to developing our statistical models. Source code for our EDA and data visualization can be found in Appendix B.

## Model Building

### *Logistic Regression*

We begin with a simple logistic regression with the lead time variable and generate a receiver operating characteristic (ROC) curve in Figure 5 below for the lead time variable using the following code:

```
fit1 <- glm(went_on_backorder~lead_time, data = Kaggle_sample, family=
binomial(logit))
summary(fit1)

fit1 <- glm(went_on_backorder~lead_time, data = Kaggle_sample, family=
binomial(logit))
fit1.roc <- roc(Kaggle_sample$went_on_backorder, fit1$fitted, plot=T,
col="blue")

pROC::auc(fit1.roc)
```

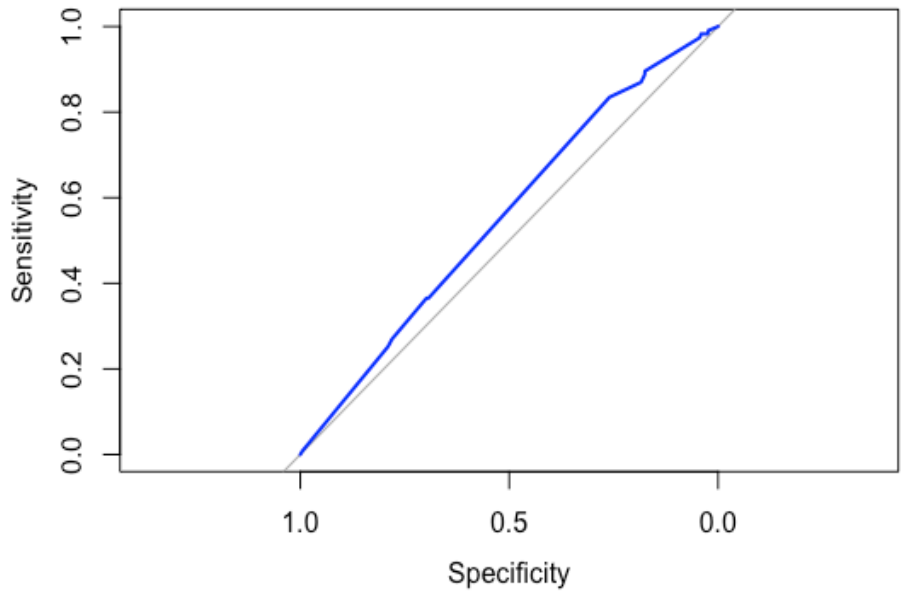



*Figure 5: ROC curve for lead time variable*

The ROC curve above has an AUC (Area Under Curve) of 0.5556, which means that this single classifier is not good enough on its own to help define backorders. However, it is a good start to the model.

### *LASSO Model and Backward Selection*

Next, we move on to the LASSO (Least Absolute Shrinkage and Selection Operator) model. We set up the LASSO model using all the variables in the dataset with went_on_backorder as our response variable. This generated the following LASSO plot in Figure 6 below:

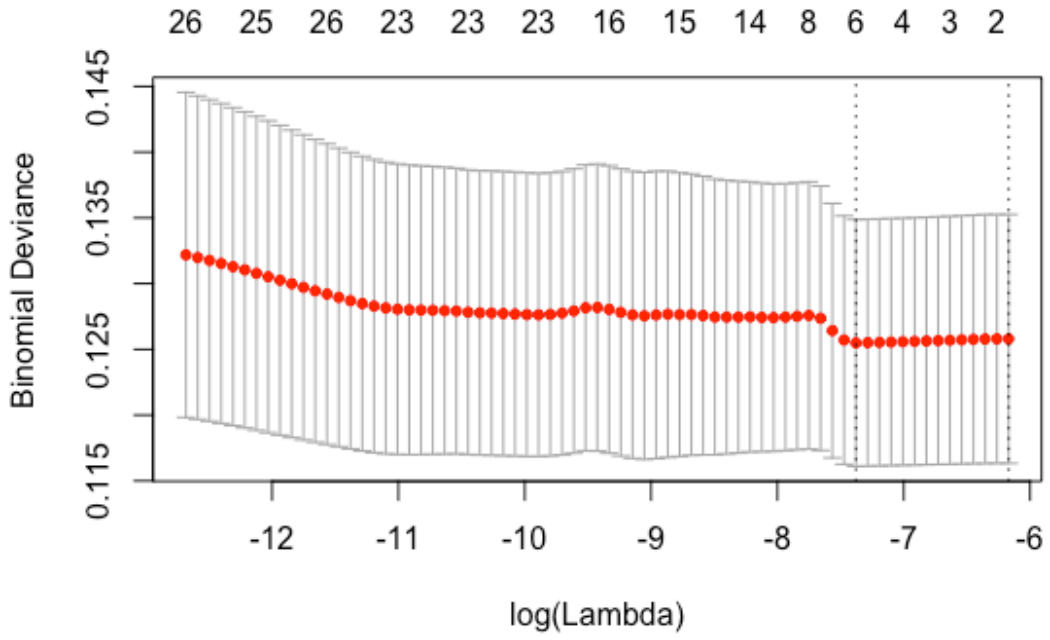



*Figure 6: LASSO plot for all variables*

We then examined the nonzero coefficients from this model using the lambda.min criterion. The following five variables were found to have nonzero coefficients:

- national_inv
- lead_time
- in_transit_qty
- perf_6_month_avg
- deck_risk

We then fit these five variables in a logistic regression and used the backward selection methodology to kick out the variable which was least significant, continuing until all variables are significant at the 0.05 level (controlling for all other variables by using the Anova function). Through this backward selection methodology, we kicked out in_transit_qty, then perf_6_month_avg, then finally deck_risk. This left us with a final model that only included the national_inv variable. The ROC curve for this model is displayed in Figure 7 below:

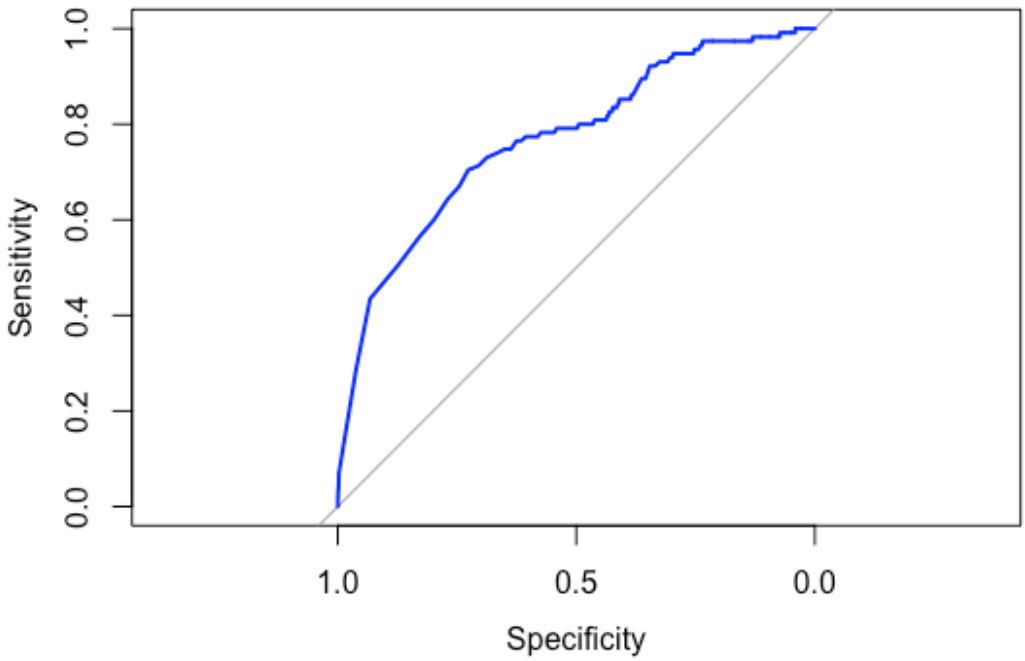



*Figure 7: ROC curve for backward selection model (national_inv variable)*

The AUC for this ROC curve was quite good, with a value of 0.7696. While we are somewhat disappointed with a model that only has one significant variable, the results at least match our intuition, since it demonstrates that current inventory levels are significant as a predictor for potential backorders.

## *Random Forest*

We then generated a random forest model trying 21 variables at each split with 500 trees in total. The confusion matrix from the random forest model was not acceptable since there was a very high misclassification error rate of over 93% despite a respectable out of bag estimate of error of just 1.2%. The confusion matrix is as follows:

```
Confusion matrix:
      No Yes    class.error
No  9872  13 0.001315123925
Yes  107   8 0.930434782609
```

The fifteen most significant variables from the random forest model are outputted in Figure 8 below. However, controlling for the sales and forecasting variables, which are highly correlated, the most significant variables are national_inv, min_bank, lead_time, in_transit_qty, pieces_past_due, and ppap_risk.

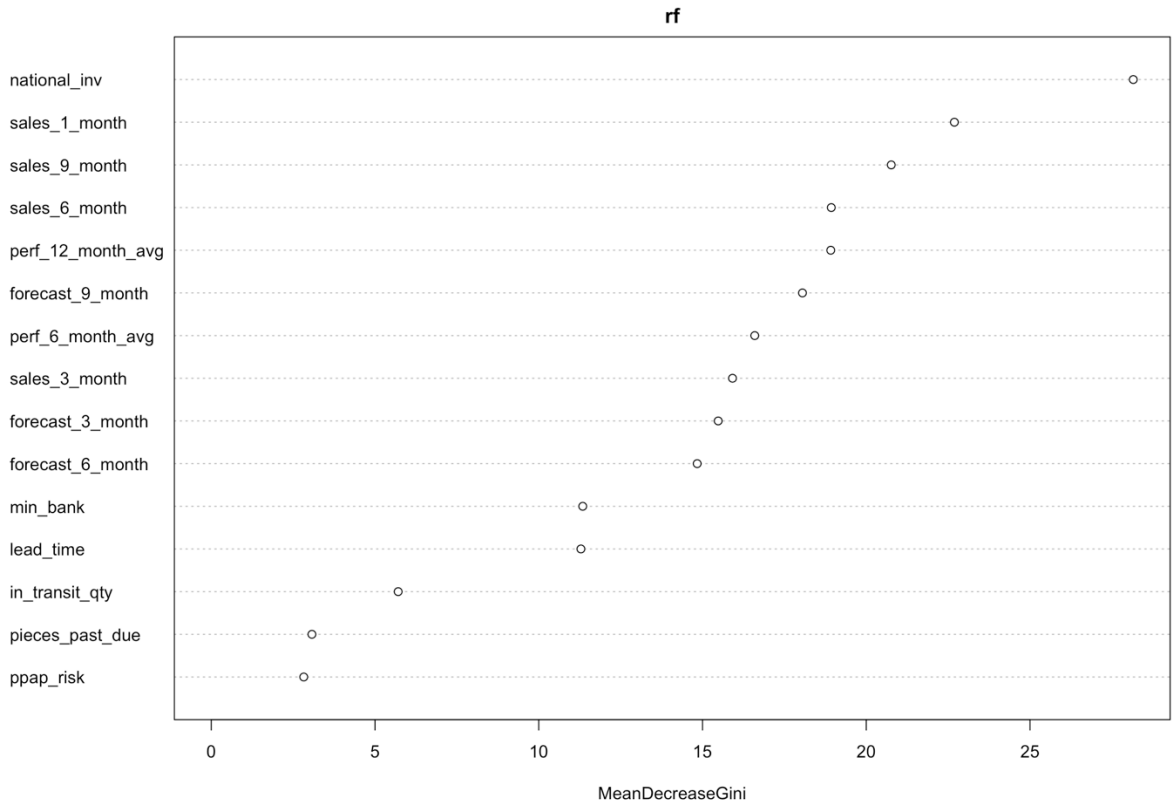



*Figure 8: Top 15 most significant variables from random forest model*

## *Neural Network*

Next, we generated a neural network for the dataset using the neural network package in R. Before we could create this model, we had to typecast all of our integer variables to numeric to get the model to run. Our final neural network model is displayed in Figure 9 below and consists of 21 predictors and one hidden layer with three neurons. These predict the outcome (i.e. went on backorder in a manner akin to that of the

human brain). However, please note that the error accrued following this method was 57.5% which is rather high, so while the insights from this model must be taken with a grain of salt, it was still an interesting exercise to run.

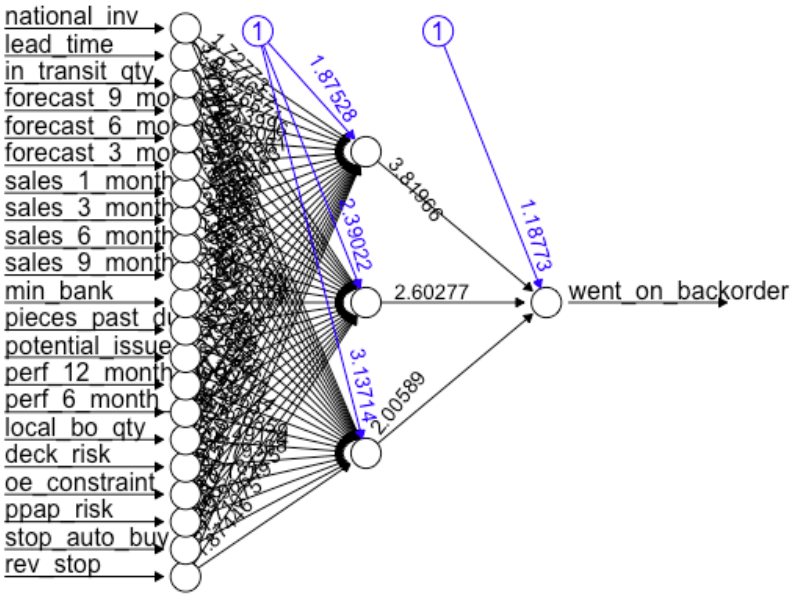



*Figure 9: Neural Network model of all variables*

## *Principal Component Analysis (PCA)*

Next, we ran principal component analysis (PCA) on our entire dataset. After scaling and centering the variables, we plotted the proportion of variance explained (pve) of each loading. This is displayed in Figures 10 below:

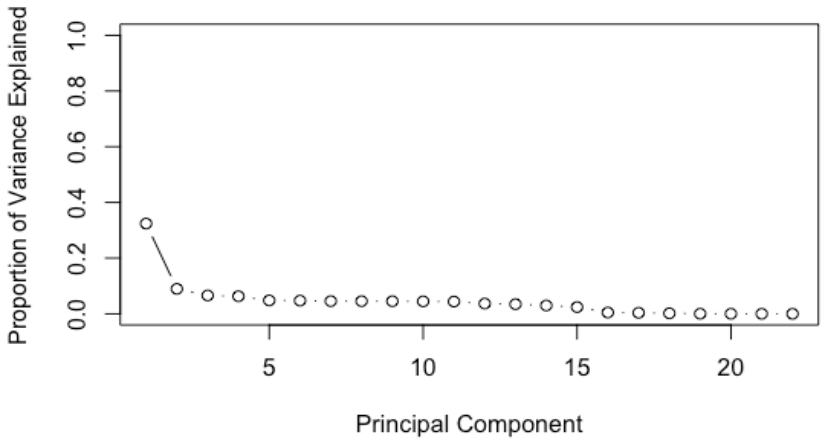



*Figure 10: Proportion of Variance explained (pve) for each principal component loading*

The 'elbow' following the second principal component in Figure 10 implies that the first two principal components explain much of the variance in the model. Specifically, the first principal component explains 32.4% of the variance and the second principal component explains 8.96% of the variance.

We also computed the correlations between the components and found that all principal components are orthogonal to one another (i.e. correlation value of 1) while the rest of the correlations are 0, meaning that each of the principal components are not correlated with other principal components. Note that without truncating these values, R internally computes these as infinitesimally small decimal values due to the nature of floating point operations, which can be seen in the full R source code in Appendix B.

Finally, Figure 11 below shows the first two principal components, indicated by the vertical and horizontal axes of the above plot. It can be clearly seen that the observations vary strongly along these two axes. This explains the variability observed in the screen plot given in Figure 10:

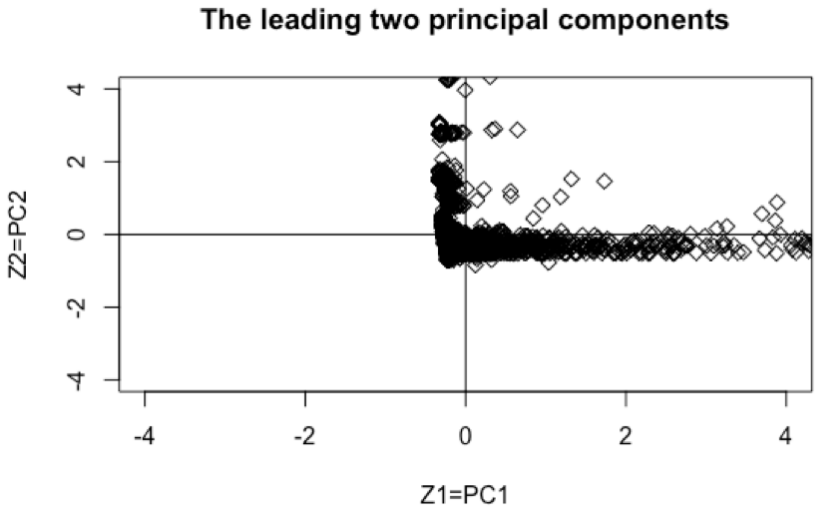



*Figure 11: Plot of two leading principal components*

## *Support Vector Machines (SVM)*

The final algorithm we implemented was the support vector machine (SVM). The performance of this algorithm was suboptimal as the confusion matrix given below indicates a high misclassification error rate.

```
  Actual
Predicted    1    2
        1 9885  113
        2    0    2
```

If higher computational power was at our disposal, perhaps a better model could be obtained. The performance of the SVM could be visualized as given below as a function of cost and epsilon, parameters which are native to the algorithm. This visualization is given below in Figure 12:

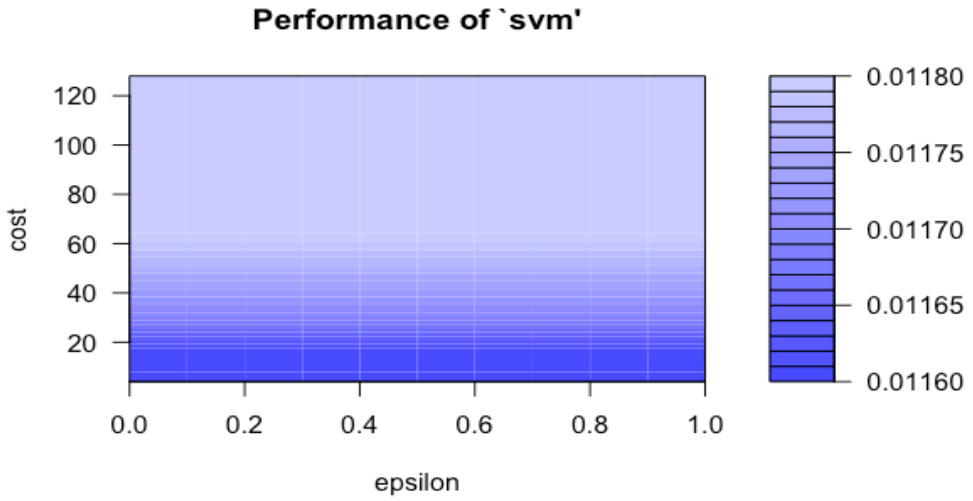



*Figure 12: Visualization of SVM algorithm performance*

## Conclusion

Extensive analysis was performed on this dataset using both supervised and unsupervised machine learning algorithms to predict the backorder status of goods. As expected, current inventory levels i.e. national inventory, is a key factor in deciding whether an item will be backordered or not. In addition to this, several other variables, namely minimum recommended amount of stock, lead time, in transit quantity, pieces past due and ppap error were also found to have noticeable predictive prowess.

Although the analysis was extensive, a more robust and complete analysis could have been performed if the data set had information with regard to the type of the product to provide some context. This is primarily because the supply chain machinations and trends in different industries vary widely, and a generalized 'one size fits all' approach to predicting backorders is insufficient. However, our exploratory data analysis and statistical learning algorithm implementation provides a structured and powerful framework to analyze if a more complete dataset is provided.

# Appendices

## *Appendix A: References*

Bain & Company. The Elements of Value. (2016, October 12). Retrieved December 13, 2017, from http://www.bain.com/publications/articles/elements-of-value-interactive.aspx

Farfan, B. (2017, July 16). 2016 Retail Industry Snapshot: Overview Stats, Facts, Research & Data. Retrieved December 13, 2017, from https://www.thebalance.com/us-retail-industry-overview-2892699

Fisher, M. (2017, Spring). Forecasting, Inventory Optimization and Pricing [PowerPoint slides]. Retrieved from https://upenn.instructure.com (OIDD 697)

Gérard P. Cachon, Paul H. Zipkin, (1999) Competitive and Cooperative Inventory Policies in a Two-Stage Supply Chain. Management Science 45(7):936-953. https://doi.org/10.1287/mnsc.45.7.936

Lazar, M. (2017, March 3). E-commerce Statistics & Technology Trendsetters for 2017 (Ecommerce CRM Software). Retrieved December 13, 2017, from https://www.ibm.com/developerworks/community/blogs/d27b1c65-986e-4a4f-a491-5e8eb23980be/entry/Ecommerce_Statistics_Technology_Trendsetters_for_20171?lang=en

MacKenzie, Ian, et al. “How retailers can keep up with consumers.” McKinsey & Company, Oct. 2013, www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers.

Tiredgeek. (2017, April 27). Retrieved December 13, 2017, from https://www.kaggle.com/tiredgeek/predict-bo-trial

## *Appendix B: R Source Code*

Presented below is the full source code for this project, knitted into Word from our original R Markdown (.rmd) file:

```
library(pROC)

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

library(glmnet)

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

## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-13

##
## Attaching package: 'glmnet'

## The following object is masked from 'package:pROC':
##
##     auc

library(neuralnet)
library(GGally)
library(reshape2)

## Warning: package 'reshape2' was built under R version 3.4.3
```

## *EDA*

In order to best understand the data before attempting to create a classification model we will explore it.

To begin an overview:

```
Kaggle_Test_Dataset <- read.csv("Kaggle_Test_Dataset_v2.csv")
dim(Kaggle_Test_Dataset)
```

There are 242,076 observations with 23 variables.

The variables of the dataset are as follows:

```
names(Kaggle_Test_Dataset)
```

Our main variable of interest is "went_on_backorder." We deem an item on backorder as a failure of the supply chain. The goal of the project is to create a classification model to help identify the factors that best predict when an item will likely go on backorder.

Here is a quick look of how the data in each variable looks:

```
str(Kaggle_Test_Dataset)
```

There aren't too many problems with the data set, however, if we were to move forward with sku we would be unable to run any sort of analysis on the data. It is a factor with 240,000 levels and does not add any value to the data (it simply defines each individual inventory item). It greatly increases computational expense and thusly we will omit it.

```
Kaggle_Test_Dataset<-Kaggle_Test_Dataset[,-1]
```

Another potential problem in the data is that the lead_time variable has NA values.

```
length(which(is.na(Kaggle_Test_Dataset$lead_time)))
```

There are 14,725 observations with NA lead times. Given that lead time is likely an important factor on whether a part is backordered it may be best to remove these values from the dataset. We will leave them in for the time being.

```
sum(is.na(Kaggle_Test_Dataset))
```

There are a total of 14739 NA values in the data set, and with 14725 of them occurring in the lead time variable it shows that almost all NA values occur here.

```
apply(Kaggle_Test_Dataset, 2, function(x) any(is.na(x)))
```

The rest of the NA values are spread out amongst many other variables. There is not a significant amount.

As previously mentioned, our column of main interest is went_on_backorder.

```
summary(Kaggle_Test_Dataset$went_on_backorder)
```

This factor has 3 values, "No" and "Yes." 1.1% of all sku's went on backorder. The factor is also flawed in that there is a blank value.

```
which(Kaggle_Test_Dataset$went_on_backorder == "")
```

It is the last row in the data set. We will remove that row from the data set and refactor the variable.

```
Kaggle_Test_Dataset <- Kaggle_Test_Dataset[-242076,]

Kaggle_Test_Dataset$went_on_backorder<-factor(Kaggle_Test_Dataset$went
```

```
_on_backorder)

###Test Summary
summary(Kaggle_Test_Dataset$went_on_backorder)
```

Now we eliminate the NA values from the dataset for easier analysis. Since so few items were NA, and there seems to be no specific reason for the NA values, we will choose to ignore them.

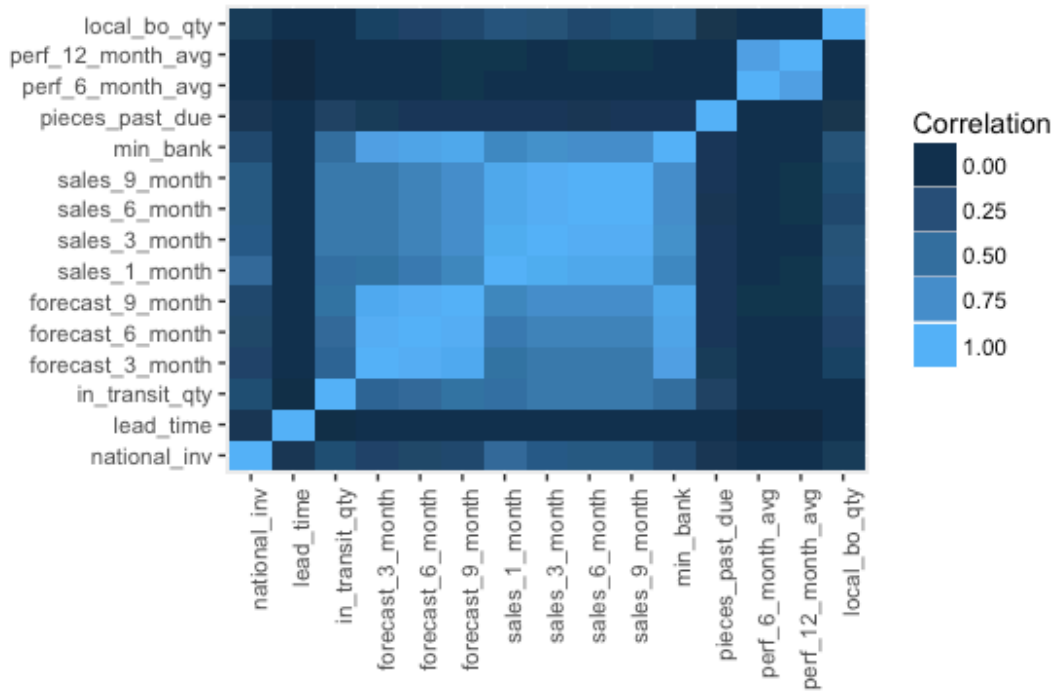
```
Kaggle_clean<-na.omit(Kaggle_Test_Dataset)
```

The data is now prepared for more exploration, however to better be able to maneuver through the data we will limit its size. With 240k rows of data it is simply too unwieldy in its current state. As such we are taking a sample of 10,000 rows to use going forward.

```
set.seed(123)
n=nrow(Kaggle_clean)
test.index <- sample(n, 10000)
length(test.index)
Kaggle_sample <- Kaggle_clean[test.index,]
names(Kaggle_sample)
dim(Kaggle_sample)
```

At 0.949 sales_6_month and sales_1_month are highly correlated. This is not to be unexpected as they are similar variables. We will also run a correlation heatmap to get a better view of all variables in the sample.

```
Kaggle_sample %>%
  select_if(is.numeric) %>%
  qplot(x = Var1,
        y = Var2,
        data = melt(cor(
          Kaggle_sample %>%
          select_if(is.numeric))),
        fill = value,
        geom = "tile") +

    xlab("") +
    ylab("") +
    guides(fill = guide_legend(title = "Correlation")) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

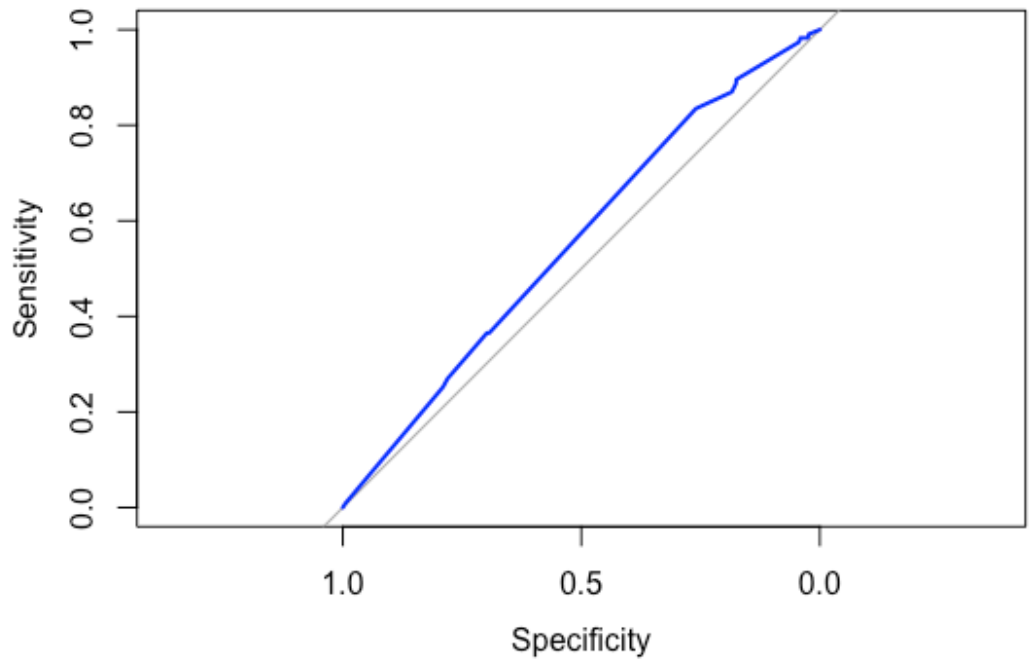



Again we see that sales and forecast are highly correlated in their monthly increments. We will keep this of note for our final models. Now we will run a simple logistic regression on the data using lead time.

```
fit1 <- glm(went_on_backorder~lead_time, data = Kaggle_sample, family=
binomial(logit))
summary(fit1)
```

Roc curve of the single variable (lead_time) classifier.

```
fit1 <- glm(went_on_backorder~lead_time, data = Kaggle_sample, family=
binomial(logit))
fit1.roc <- roc(Kaggle_sample$went_on_backorder, fit1$fitted, plot=T,
col="blue")
```



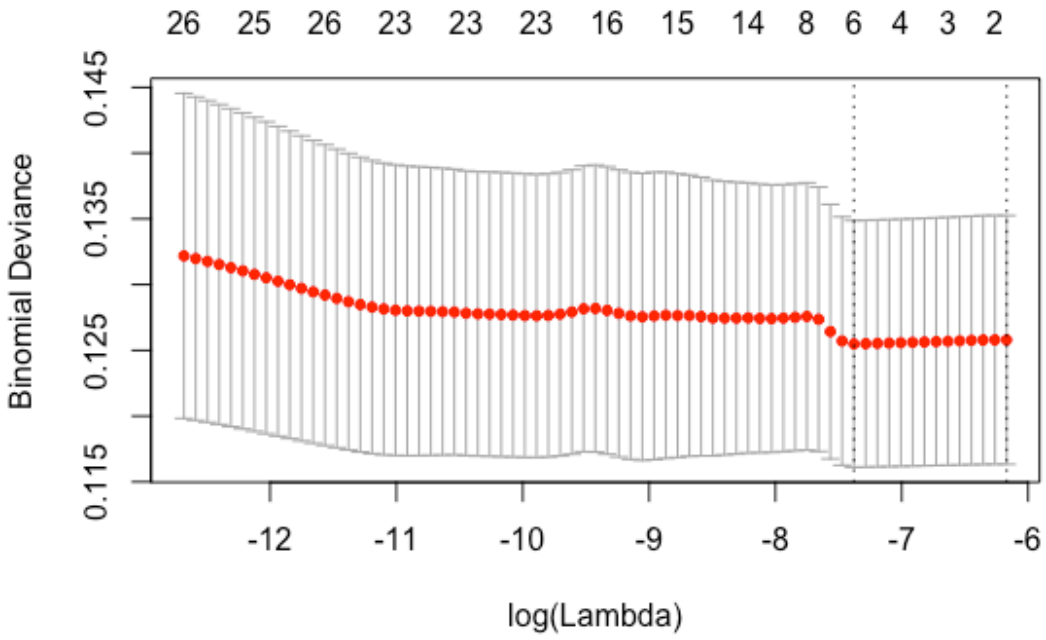

```
pROC::auc(fit1.roc)
```

With an AUC of 0.5556 this single classifier is not good enough on its own to help define backorders, however, it is a good start.

## *LASSO*

A categorical variable is coded with indicator functions.

```
X <- model.matrix(went_on_backorder~., data=Kaggle_sample)[,-1]
dim(X)

Y <- Kaggle_sample[, 22]

summary(Kaggle_sample$went_on_backorder)

set.seed(10)
fit1.cv <- cv.glmnet(X, Y, alpha=1, family="binomial", nfolds = 10, ty
pe.measure = "deviance")

plot(fit1.cv)
```

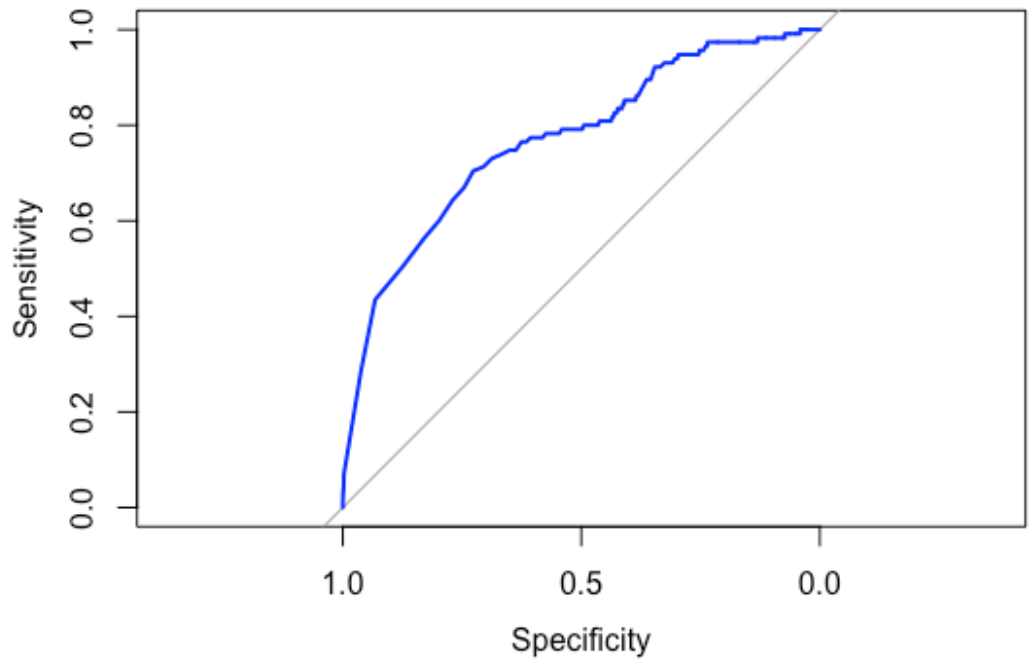



```
coef.min <- coef(fit1.cv, s="lambda.min")
coef.min <- coef.min[which(coef.min !=0),]
coef.min

fit.logit.1<-
  glm(went_on_backorder~national_inv+lead_time+in_transit_qty+perf_6_m
onth_avg+deck_risk, family=binomial, data=Kaggle_sample)
Anova(fit.logit.1)
```

### *Backward selection*

```
fit.logit.1<-
  glm(went_on_backorder~national_inv+lead_time+perf_6_month_avg+deck_r
isk, family=binomial, data=Kaggle_sample)
Anova(fit.logit.1)

fit.logit.1<-
  glm(went_on_backorder~national_inv+lead_time+deck_risk, family=binom
ial, data=Kaggle_sample)
Anova(fit.logit.1)

fit.logit.1<-glm(went_on_backorder~national_inv+deck_risk, family=bino
mial, data=Kaggle_sample)
Anova(fit.logit.1)
```

```
fit2.roc <- roc(Kaggle_sample$went_on_backorder, fit.logit.1$fitted, plot=T, col="blue")
```




```
pROC::auc(fit2.roc)
```

### *Random Forest*

Generate Random Forest

```
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
rf<-randomForest(went_on_backorder~.,data=Kaggle_sample, mtry=21, ntree=500)

print(rf)

varImpPlot(rf, sort = T, n.var = 15)
```

### *OLS backward selection*

```
linfit1<-glm(went_on_backorder~.,family=binomial, data=Kaggle_sample)
summary(linfit1)

names(Kaggle_sample)

str(Kaggle_sample)

Kaggle_sample$national_inv <-
  (Kaggle_sample$national_inv-min(Kaggle_sample$national_inv))/
  (max(Kaggle_sample$national_inv)-min(Kaggle_sample$national_inv))

Kaggle_sample$lead_time <-
  (Kaggle_sample$lead_time-min(Kaggle_sample$lead_time))/
  (max(Kaggle_sample$lead_time)-min(Kaggle_sample$lead_time))
Kaggle_sample$in_transit_qty <-
  (Kaggle_sample$in_transit_qty-min(Kaggle_sample$in_transit_qty))/
  (max(Kaggle_sample$in_transit_qty)-min(Kaggle_sample$in_transit_qty)
)
Kaggle_sample$forecast_3_month <-
  (Kaggle_sample$forecast_3_month-min(Kaggle_sample$forecast_3_month))
/
  (max(Kaggle_sample$forecast_3_month)-min(Kaggle_sample$forecast_3_mo
nth))
Kaggle_sample$forecast_6_month <-
  (Kaggle_sample$forecast_6_month-min(Kaggle_sample$forecast_6_month))
/
  (max(Kaggle_sample$forecast_6_month)-min(Kaggle_sample$forecast_6_mo
nth))
Kaggle_sample$forecast_9_month <-
  (Kaggle_sample$forecast_9_month-min(Kaggle_sample$forecast_9_month))
/
  (max(Kaggle_sample$forecast_9_month)-min(Kaggle_sample$forecast_9_mo
nth))
Kaggle_sample$sales_1_month <-
  (Kaggle_sample$sales_1_month-min(Kaggle_sample$sales_1_month))/
  (max(Kaggle_sample$sales_1_month)-min(Kaggle_sample$sales_1_month))
Kaggle_sample$sales_3_month <-
  (Kaggle_sample$sales_3_month-min(Kaggle_sample$sales_3_month))/
```

```
  (max(Kaggle_sample$sales_3_month)-min(Kaggle_sample$sales_3_month))
Kaggle_sample$sales_6_month <-
  (Kaggle_sample$sales_6_month-min(Kaggle_sample$sales_6_month))/
  (max(Kaggle_sample$sales_6_month)-min(Kaggle_sample$sales_6_month))
Kaggle_sample$sales_9_month <-
  (Kaggle_sample$sales_9_month-min(Kaggle_sample$sales_9_month))/
  (max(Kaggle_sample$sales_9_month)-min(Kaggle_sample$sales_9_month))
Kaggle_sample$min_bank <-
  (Kaggle_sample$min_bank-min(Kaggle_sample$min_bank))/
  (max(Kaggle_sample$min_bank)-min(Kaggle_sample$min_bank))
Kaggle_sample$pieces_past_due <-
  (Kaggle_sample$pieces_past_due-min(Kaggle_sample$pieces_past_due))/
  (max(Kaggle_sample$pieces_past_due)-min(Kaggle_sample$pieces_past_du
e))
Kaggle_sample$perf_6_month_avg <-
  (Kaggle_sample$perf_6_month_avg-min(Kaggle_sample$perf_6_month_avg))
/
  (max(Kaggle_sample$perf_6_month_avg)-min(Kaggle_sample$perf_6_month_
avg))
Kaggle_sample$perf_12_month_avg <-
  (Kaggle_sample$perf_12_month_avg-min(Kaggle_sample$perf_12_month_avg
))/
  (max(Kaggle_sample$perf_12_month_avg)-min(Kaggle_sample$perf_12_mont
h_avg))
Kaggle_sample$local_bo_qty <-
  (Kaggle_sample$local_bo_qty-min(Kaggle_sample$local_bo_qty))/
  (max(Kaggle_sample$local_bo_qty)-min(Kaggle_sample$local_bo_qty))
```

### *Neural Network*

```
Kaggle_sample$national_inv <- as.numeric(Kaggle_sample$national_inv)
Kaggle_sample$lead_time <- as.numeric(Kaggle_sample$lead_time )
Kaggle_sample$in_transit_qty <- as.numeric(Kaggle_sample$in_transit_qt
y)
Kaggle_sample$forecast_3_month <- as.numeric(Kaggle_sample$forecast_3_
month)
Kaggle_sample$forecast_6_month <- as.numeric(Kaggle_sample$forecast_6_
month)
Kaggle_sample$forecast_9_month <- as.numeric(Kaggle_sample$forecast_9_
month)
Kaggle_sample$sales_1_month <- as.numeric(Kaggle_sample$sales_1_month)
Kaggle_sample$sales_3_month <- as.numeric(Kaggle_sample$sales_3_month)
Kaggle_sample$sales_6_month <- as.numeric(Kaggle_sample$sales_6_month)
Kaggle_sample$sales_9_month <- as.numeric(Kaggle_sample$sales_9_month)
Kaggle_sample$min_bank <- as.numeric(Kaggle_sample$min_bank)
Kaggle_sample$pieces_past_due <- as.numeric(Kaggle_sample$pieces_past_
```

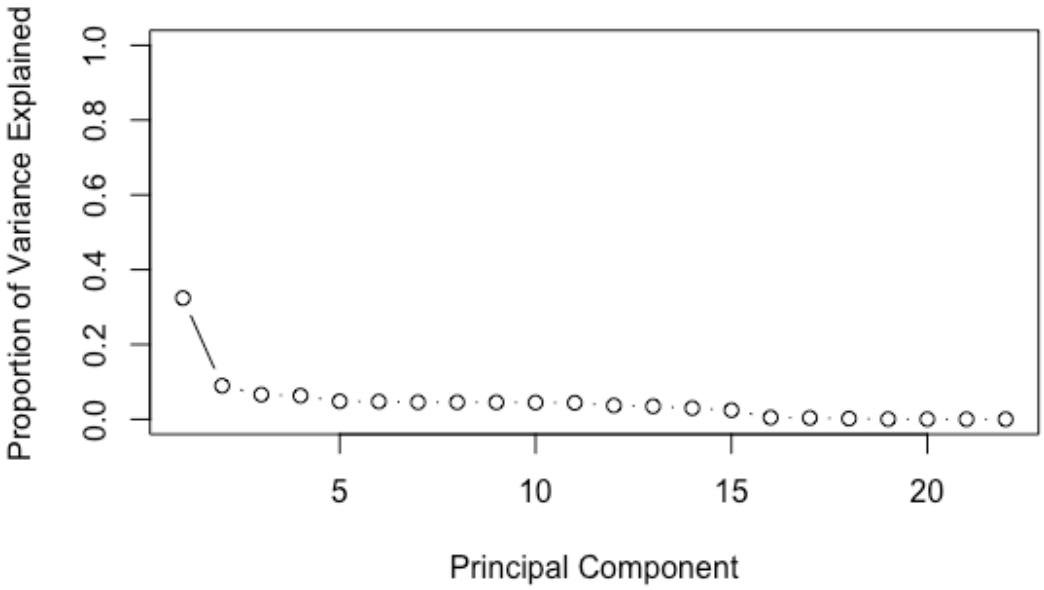
```
due)
Kaggle_sample$perf_6_month_avg <- as.numeric(Kaggle_sample$perf_6_mont
h_avg)
Kaggle_sample$perf_12_month_avg <- as.numeric(Kaggle_sample$perf_12_mo
nth_avg)
Kaggle_sample$local_bo_qty <- as.numeric(Kaggle_sample$local_bo_qty)
Kaggle_sample$potential_issue <- as.numeric(Kaggle_sample$potential_is
sue)
Kaggle_sample$deck_risk <- as.numeric(Kaggle_sample$deck_risk)
Kaggle_sample$oe_constraint <- as.numeric(Kaggle_sample$oe_constraint)
Kaggle_sample$ppap_risk <- as.numeric(Kaggle_sample$ppap_risk)
Kaggle_sample$stop_auto_buy <- as.numeric(Kaggle_sample$stop_auto_buy)
Kaggle_sample$rev_stop <- as.numeric(Kaggle_sample$rev_stop)


Kaggle_sample$went_on_backorder <- as.numeric(Kaggle_sample$went_on_ba
ckorder)

str(Kaggle_sample)

n <- neuralnet(went_on_backorder~national_inv+lead_time+in_transit_qty
+forecast_9_month+forecast_6_month+forecast_3_month+sales_1_month+sale
s_3_month+sales_6_month+sales_9_month+min_bank+pieces_past_due+potenti
al_issue+perf_12_month_avg+perf_6_month_avg+local_bo_qty+deck_risk+oe_
constraint+ppap_risk+stop_auto_buy+rev_stop, data=Kaggle_sample, hidde
n = 3, err.fct = "sse", linear.output = FALSE)


plot(n)
summary(n)
```

The neural network consisting of 21 predictors and one hidden layer with three neurons. These predict the outcome i.e. went on backorder in a manner akin to that of the human brain. However, please note that the error accrued following this method was 57.5% which is rather high.

```
n$result.matrix
```

### *PCA*

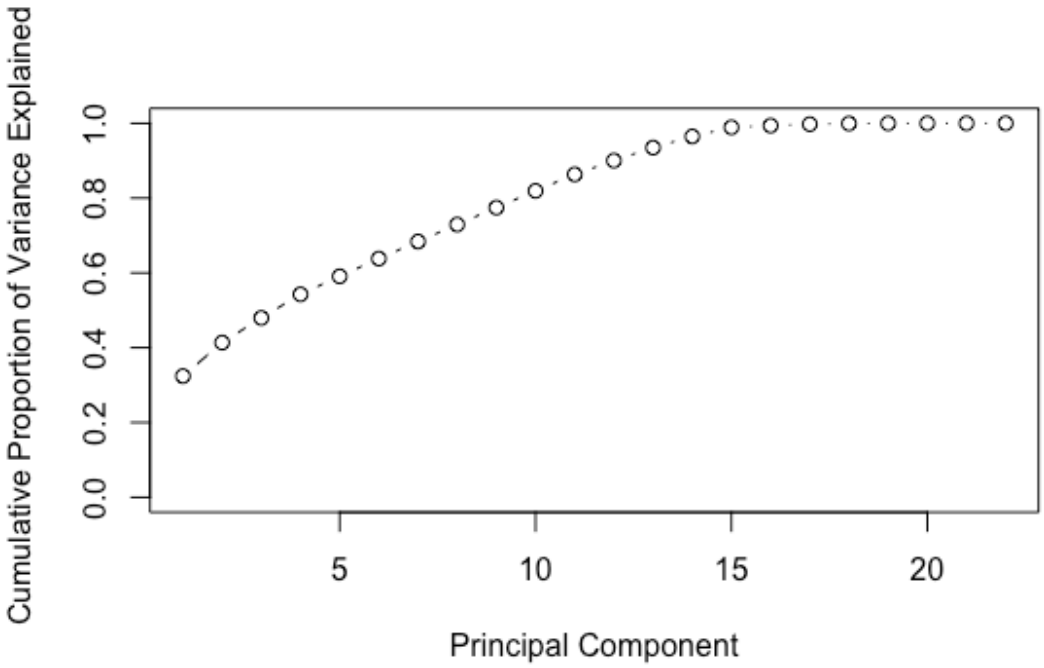
```
data.scale <- scale(Kaggle_sample, center=TRUE, scale =TRUE)
pca.all <- prcomp(Kaggle_sample , scale =TRUE)
pr.var <- pca.all$sdev^2
pve <- pr.var/sum(pr.var)
plot(pve , xlab=" Principal Component ",
ylab=" Proportion of Variance Explained ", ylim=c(0,1) ,type='b')
```




```
pve <- pr.var/sum(pr.var)
pve
```

The 'elbow' following the second principal component implies that the first 2 principal components explain much of the variance in the model.Specifically, the first principal component explains 32.4% of the variance and the second principal component explains 8.96% of the variance.

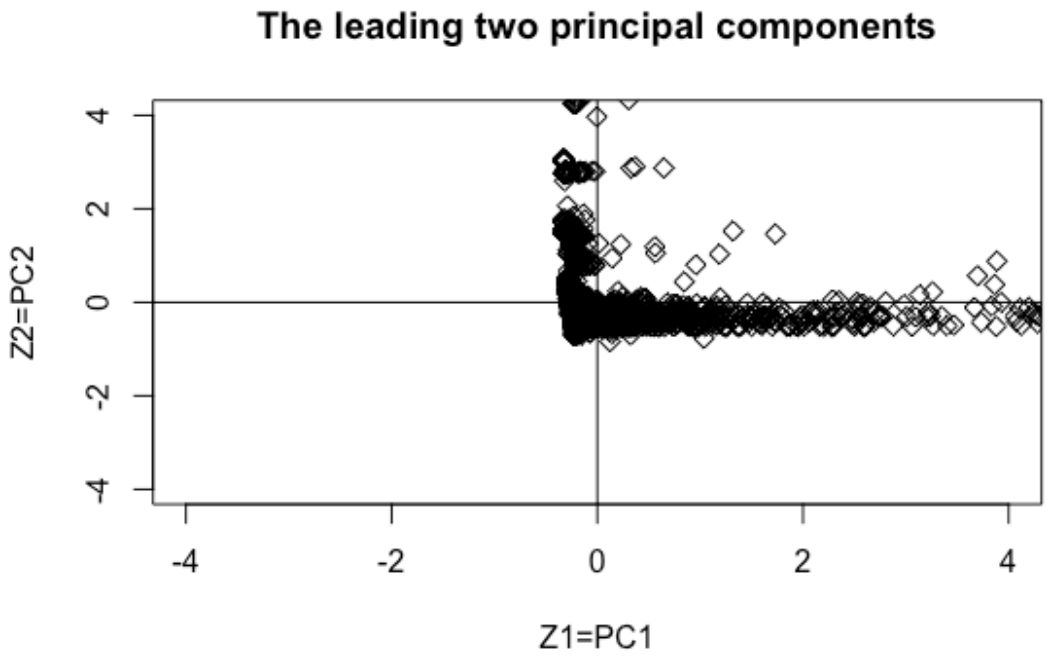
```
plot(cumsum (pve), xlab=" Principal Component ",
ylab ="Cumulative Proportion of Variance Explained ", ylim=c(0,1) ,typ
e='b')
```




```
PCAresult <- cor(pca.all$x)
trunc(PCAresult)
```

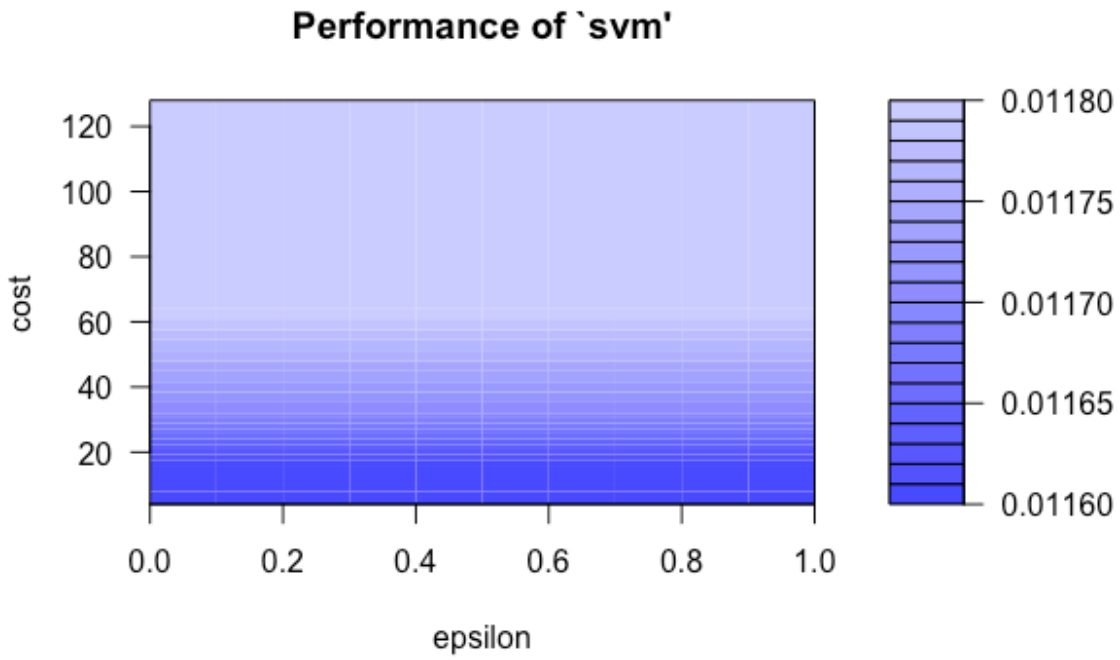
The above correlations indicate that all principal components are orthogonal to one another. In the above matrix representation of correlations, the diagonals are 1 . i.e. correlations of each Principal component with itslef, as expected. The rest are 0. Note that without truncating these values, R internally computes these as infinitesimally small decimal values due to the nature of floating point operations.

```
plot(pca.all$x[, 1], pca.all$x[,2 ], pch=5,
     xlim=c(-4, 4),
     ylim=c(-4, 4),
     main="The leading two principal components",
     xlab="Z1=PC1",
     ylab="Z2=PC2"
     )
abline(h=0, v=0)
```




The two principal components are indicated by the vertical and horizontal axes of the above plot. It can be clearly seen that the observations vary strongly along these two axes. This explains the variability observed in the screen plot given before.

```
library(e1071)
Kaggle_sample$went_on_backorder <- as.factor(Kaggle_sample$went_on_bac
korder)
mymodel <- svm(went_on_backorder~.,data=Kaggle_sample)
summary(mymodel)

# Support Vector Machines
pred <- predict(mymodel, Kaggle_sample)
tab <- table(Predicted= pred, Actual= Kaggle_sample$went_on_backorder)
tab

set.seed(123)
tmodel <-
tune(svm,went_on_backorder~.,data=Kaggle_sample, ranges= list(epsilon
= seq(0,1,0.1), cost= 2^(2:7)) )

plot(tmodel)
```




```
summary(tmodel)

mymodel <- tmodel$best.model
pred <- predict(mymodel, Kaggle_sample)
tab <- table(Predicted= pred, Actual= Kaggle_sample$went_on_backorder)
tab
```