Back Protector - Minimizing Backorder using ML Algorithms

Machine Learning II Final Project
Professor Panos
Team 4
Frank Fan, Carl Xi, Yaping Zhang, Jie Zhu

Preface

For as long as production has been a part of human civilization, inventory has been a pain point for most businesses. Until the development of better demand forecasting and prediction methods in recent decades, overstocking and understocking has cost businesses trillions of dollars around the world each year. Toyota was one of the first ever companies to implement Just-in-Time (JIT) methods, where parts arrive just in time for production, saving the company a fortune in minimizing part storage. While leaner supply chains around the world have effectively reduced the costs associated with overstocking, the problem of understocking still remains. Unexpected surges in demand in both parts and consumer products continue to plague the global economy. Our project is largely inspired by the current COVID-19 crisis, where social distancing has severely impacted production around the world. The effects of COVID-19 on China's manufacturing industry is still felt around the world, as countless industries scramble to catch up the backorder that has accumulated over the past few months.

Business Understanding

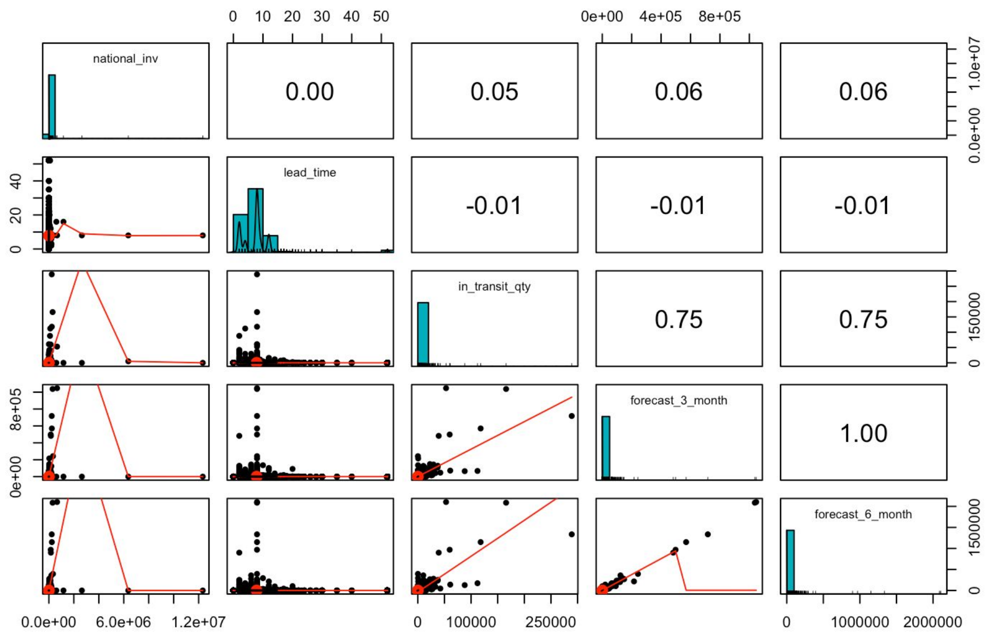
When products become out of stock, the business has no choice but to backorder the item with the supplier one-up in the supply chain. This is a lose-lose situation, as the business is placing more orders based on a promise of future revenue (the customer can churn), the customer receives the product much later than when the need first arises, and the supplier is under pressure to produce overtime to make up for the increase in demand. This is even a greater problem in manufacturing chains, where the delay of a single part can have a ripple effect along the whole production line. Just like how the sudden stop of a single car can cause a traffic jam along an entire highway, backorders can severely disrupt the delicate production/supply streams in today's society. Thus, being able to predict when and by how much backorders can occur for each product can significantly reduce the risk of running out of inventory and the associated financial consequences.

Problem Statement

In a nutshell, our problem statement is: "Can we build a cost-aware model that accurately predicts when backorders are most likely to happen?" As mentioned above, we wanted to explore this topic as it is an extremely current and relevant problem that industries around the world are facing today. To tackle this problem, we will be examining a dataset from the garment industry. The clothing industry is notoriously known to have immense backorder problems due to the countless SKU, sizing and color variations for each product. Even the most well managed clothing companies still suffer from product shortage in some of their distributors and retailers. Thus, the data from the clothing industry is rich with backorder data and perfect for our modeling purposes.

Data Understanding

Our source data comes from datahub and is labeled "Global Garment Supply Chain Data". This dataset covers the performance of various clothing suppliers' fulfillment history. The dataset consists of 1 target variable column and 22 attributes, of which 6 are binary, 15 are numeric, and 1 is a notation. The attributes range from simpler variables like quantity to more informative variables like product lead time (weeks prior to delivery). Despite choosing one of the most backorder-prevalent industries in the world, modern supply chain efficiency means that backorder data entries only account for about 1% of our data. This is very problematic, as class imbalance can severely impact the performance of classification models. We have made steps to tackle this problem, as we will explain in the data cleaning section below. The summary statistics table is also included in the data preparation step below for your reference. We also made sure to look at the distribution of our variables and the correlations between them for any potential moticollearinity. The chart below is an example of how our ultimate pairplot looks like, but with just 5 variables instead of 22. We can immediately see that some variables (e.g. the forcast_x_months) are heavily correlated with each other, while other variables (e.g. lead_time) are heavily skewed. As such, data transformation or equivalent steps must be taken prior to modeling to ensure good performance. Normalization and Standardization details will be covered in Data Preparation, though we also wanted to make a note that some of the modeling processes we have chosen does not require data transformation or automatically conducts them.



An example of a pairplot built on a subset of our data.

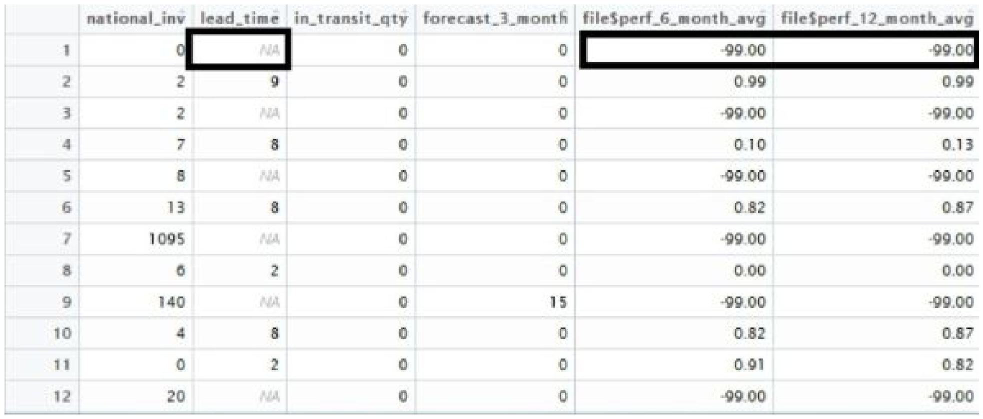
Data Preparation

We divided the data preparation process into three steps: data cleaning, dealing with imbalanced dataset and feature engineering, in order to produce the format required for data mining:

1. Data Cleaning

For the original data set, we have 22 features and 1 target variable. Out of the 22 features, we have 1 notation feature, 6 binary features and 15 numerical features.

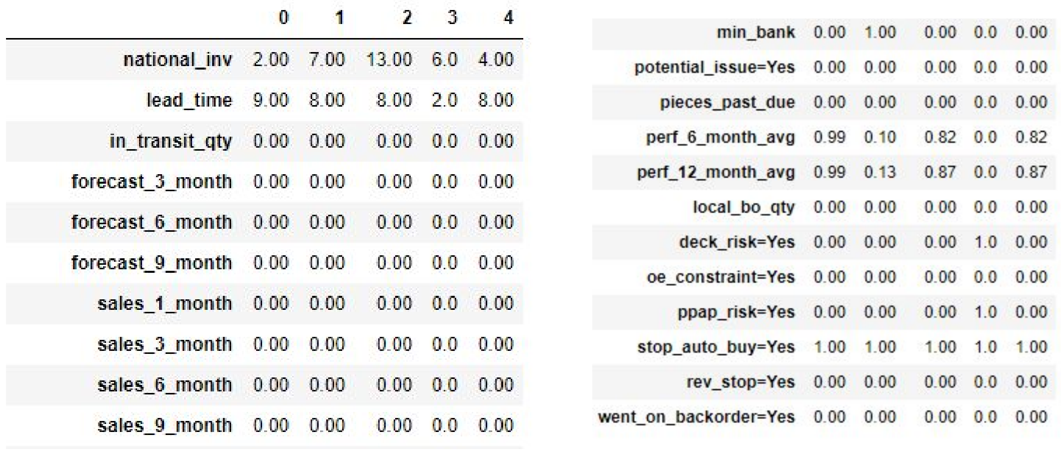
The features are a representative of different product suppliers' historical logistic performance, and some of the values are not reasonable, for example they have values of -99 or they are marked with NA. For the obvious missing data, we will delete the rows. For the notation column, since it will not be useful in the model building part, we will also delete it.



	national_inv	lead_time	in_transit_qty	forecast_3_month	file$perf_6_month_avg	file$perf_12_month_avg
1	0	NA	0	0	-99.00	-99.00
2	2	9	0	0	0.99	0.99
3	2	NA	0	0	-99.00	-99.00
4	7	8	0	0	0.10	0.13
5	8	NA	0	0	-99.00	-99.00
6	13	8	0	0	0.82	0.87
7	1095	NA	0	0	-99.00	-99.00
8	6	2	0	0	0.00	0.00
9	140	NA	0	15	-99.00	-99.00
10	4	8	0	0	0.82	0.87
11	0	2	0	0	0.91	0.82
12	20	NA	0	0	-99.00	-99.00

Table: Outliers and Missing Values

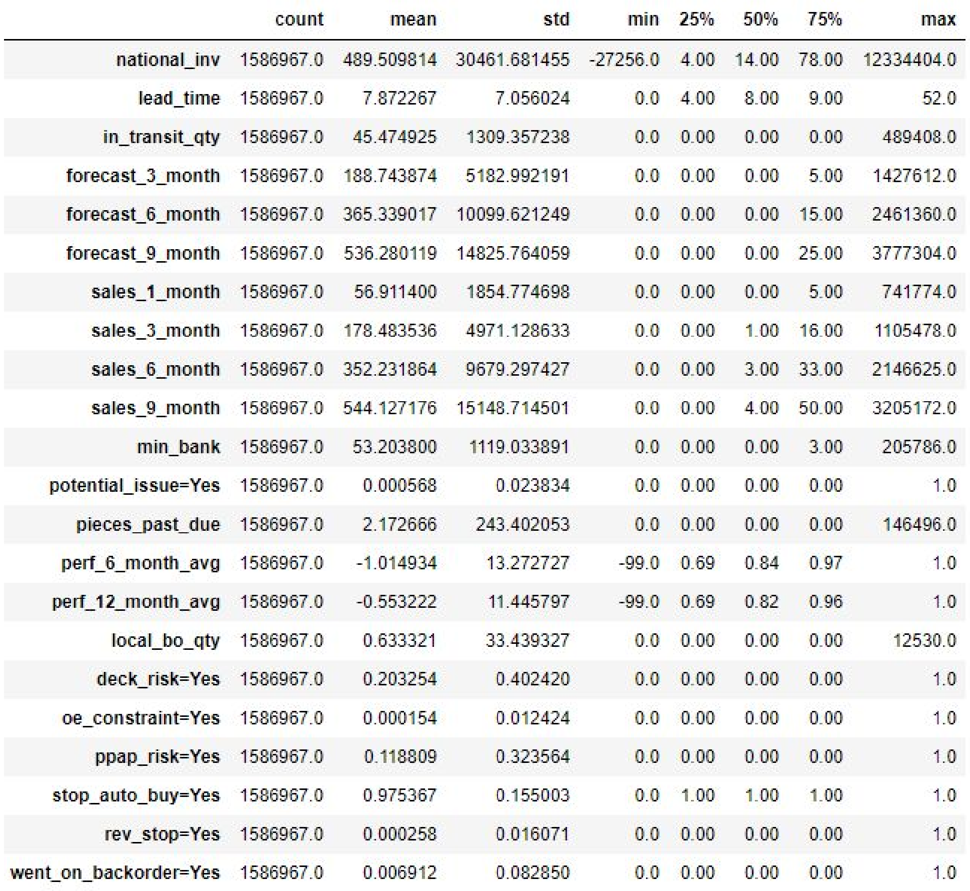
After initial cleaning, the data we have right now are shaped with 1586967 rows and 22 columns (21 features and 1 target variable). We can take a look at the first 5 rows of the dataset.



	0	1	2	3	4
national_inv	2.00	7.00	13.00	6.0	4.00
lead_time	9.00	8.00	8.00	2.0	8.00
in_transit_qty	0.00	0.00	0.00	0.0	0.00
forecast_3_month	0.00	0.00	0.00	0.0	0.00
forecast_6_month	0.00	0.00	0.00	0.0	0.00
forecast_9_month	0.00	0.00	0.00	0.0	0.00
sales_1_month	0.00	0.00	0.00	0.0	0.00
sales_3_month	0.00	0.00	0.00	0.0	0.00
sales_6_month	0.00	0.00	0.00	0.0	0.00
sales_9_month	0.00	0.00	0.00	0.0	0.00
min_bank	0.00	1.00	0.00	0.0	0.00
potential_issue=Yes	0.00	0.00	0.00	0.0	0.00
pieces_past_due	0.00	0.00	0.00	0.0	0.00
perf_6_month_avg	0.99	0.10	0.82	0.0	0.82
perf_12_month_avg	0.99	0.13	0.87	0.0	0.87
local_bo_qty	0.00	0.00	0.00	0.0	0.00
deck_risk=Yes	0.00	0.00	0.00	1.0	0.00
oe_constraint=Yes	0.00	0.00	0.00	0.0	0.00
ppap_risk=Yes	0.00	0.00	0.00	1.0	0.00
stop_auto_buy=Yes	1.00	1.00	1.00	1.0	1.00
rev_stop=Yes	0.00	0.00	0.00	0.0	0.00
went_on_backorder=Yes	0.00	0.00	0.00	0.0	0.00

Table: Head of Dataset

And we can also take a look at the basic descriptive statistics of the dataset to see if there is something interesting to look at or something to be aware of.



	count	mean	std	min	25%	50%	75%	max
national_inv	1586967.0	489.509814	30461.681455	-27256.0	4.00	14.00	78.00	12334404.0
lead_time	1586967.0	7.872267	7.056024	0.0	4.00	8.00	9.00	52.0
in_transit_qty	1586967.0	45.474925	1309.357238	0.0	0.00	0.00	0.00	489408.0
forecast_3_month	1586967.0	188.743874	5182.992191	0.0	0.00	0.00	5.00	1427612.0
forecast_6_month	1586967.0	365.339017	10099.621249	0.0	0.00	0.00	15.00	2461360.0
forecast_9_month	1586967.0	536.280119	14825.764059	0.0	0.00	0.00	25.00	3777304.0
sales_1_month	1586967.0	56.911400	1854.774698	0.0	0.00	0.00	5.00	741774.0
sales_3_month	1586967.0	178.483536	4971.128633	0.0	0.00	1.00	16.00	1105478.0
sales_6_month	1586967.0	352.231864	9679.297427	0.0	0.00	3.00	33.00	2146625.0
sales_9_month	1586967.0	544.127176	15148.714501	0.0	0.00	4.00	50.00	3205172.0
min_bank	1586967.0	53.203800	1119.033891	0.0	0.00	0.00	3.00	205786.0
potential_issue=Yes	1586967.0	0.000568	0.023834	0.0	0.00	0.00	0.00	1.0
pieces_past_due	1586967.0	2.172666	243.402053	0.0	0.00	0.00	0.00	146496.0
perf_6_month_avg	1586967.0	-1.014934	13.272727	-99.0	0.69	0.84	0.97	1.0
perf_12_month_avg	1586967.0	-0.553222	11.445797	-99.0	0.69	0.82	0.96	1.0
local_bo_qty	1586967.0	0.633321	33.439327	0.0	0.00	0.00	0.00	12530.0
deck_risk=Yes	1586967.0	0.203254	0.402420	0.0	0.00	0.00	0.00	1.0
oe_constraint=Yes	1586967.0	0.000154	0.012424	0.0	0.00	0.00	0.00	1.0
ppap_risk=Yes	1586967.0	0.118809	0.323564	0.0	0.00	0.00	0.00	1.0
stop_auto_buy=Yes	1586967.0	0.975367	0.155003	0.0	1.00	1.00	1.00	1.0
rev_stop=Yes	1586967.0	0.000258	0.016071	0.0	0.00	0.00	0.00	1.0
went_on_backorder=Yes	1586967.0	0.006912	0.082850	0.0	0.00	0.00	0.00	1.0

Table: Descriptive Statistics

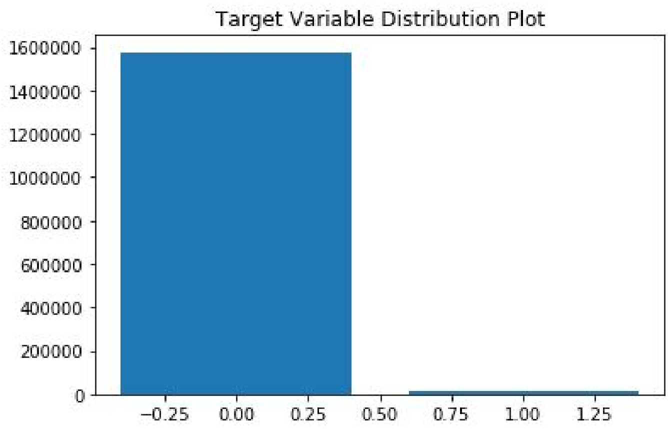
Based on the descriptive statistics above, we can see that our target variable may have the problem of imbalance because the mean (0.0069) is very close to the min data (0) and far away from the max (1). What's more, for the 25%, 50% and 75% of the target variable, the values are all 0. Therefore, we decided to take a closer look at the target variable and address the imbalanced problem.

2. Dealing with Imbalanced Data

We have explored different techniques to deal with imbalanced data:

a. Up-sampling

When we take a closer look at the target variable 'went_on_backorder=Yes' which describes whether or not a product's supply is on backorder (1) or not (0), we can see that it's very imbalanced: 1579988 '0' values and 10969 '1' values, which means we have about 1% of the products that went on backorder, causing a serious imbalance on the target variable classes.



Graph: Imbalanced Class Distribution

With such a severely imbalanced class distribution, our further predictive models can get high overall accuracy but with a very low recall for the minority class. And those models can't generate any good insights.

To solve this problem, we will use the up-sampling method to up-sample minority class by randomly duplicating observations from the minority class in order to reinforce its signal. Specifically, we will simply resample the target variable's minority class with replacement to make the number of it even with the majority class.

The specific steps we take are:

- Firstly, we'll separate observations from each target variable class into two DataFrames.
- Next, we'll resample the minority class with replacement to make the number of minority samples finally match that of the majority class.
- Finally, combine the up-sampled minority class data with the original majority class data to make up the final data.

After those steps, we can get a balanced dataset. However, this approach involves duplicating examples in the minority class, although these examples don't add any new information to the model. Therefore, we decide to try more resampling approaches.

b. Synthetic Minority Oversampling Technique (SMOTE)

SMOTE is a new up-sampling approach. It solves the imbalance issue by resampling from the minority class while slightly perturbing feature values, thereby creating "new" samples. And the below is a detailed explanation of this method from 'Imbalanced Learning: Foundations, Algorithms, and Applications, 2013'.

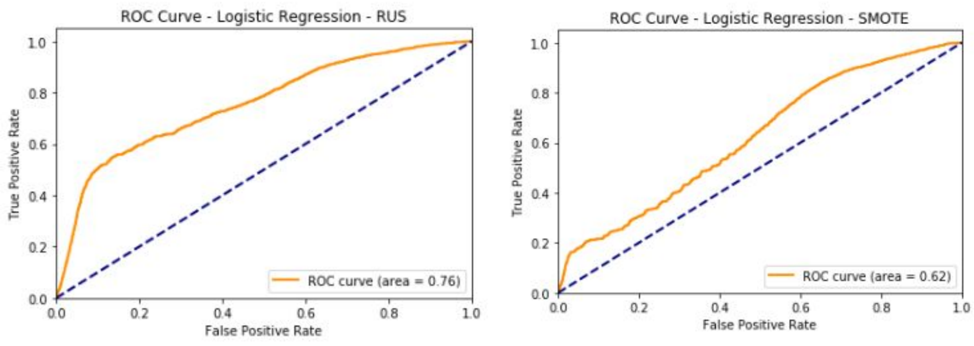
"SMOTE first selects a minority class instance at random and finds its k nearest minority class neighbors. The synthetic instance is then created by choosing one of the k nearest neighbors b at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances a and b.”

c. Down-sampling

Down-sampling is the process of randomly removing observations from the majority class to prevent its signal from dominating the learning algorithm.

The specific steps need to take are:

- First, we'll separate observations from each class into different DataFrames.
- Next, we'll resample the majority class without replacement, setting the number of samples to match that of the minority class.
- Finally, we'll combine the down-sampled majority class DataFrame with the original minority class DataFrame.



Graphs: ROC Curves with Different Resampling Techniques

After trying out all resampling approaches using our logistic regression base model, the results show that the under-sampling technique works the best for our data (as shown in the above ROC graphs). Thus, we decided to use undersampling to transform our dataset.

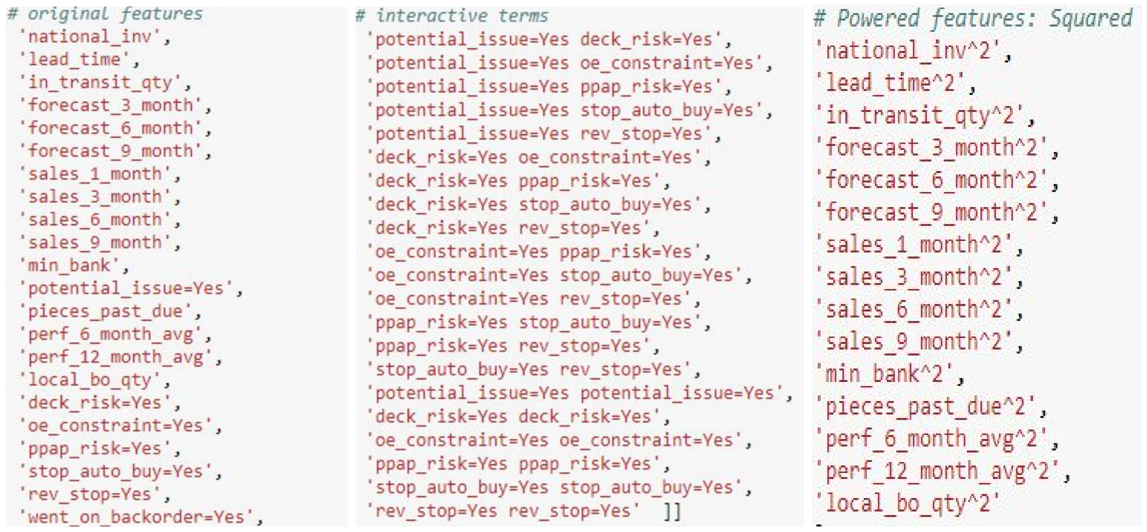
3. Feature Engineering

After data cleaning and resampling, we moved onto feature engineering. To quickly introduce the features, we have 21 features and 1 target variable whose meanings are listed below:

- X1 = Current inventory level of a component: 'national_inv';
- X2 = Registered transit time for the product: 'lead_time';
- X3 = In transit quantity: 'in_transit_qty';
- X4,5,6 = Forecast for next 3,6 and 9 months: 'forecast_3_month', 'forecast_6_month', 'forecast_9_month';
- X7,8,9,10 = Sales quantity for the prior 1,3,6,9 months: 'sales_1_month', 'sales_3_month', 'sales_6_month', 'sales_9_month';
- X11 = Minimum recommended amount to stock: 'min_bank';
- X12 = Parts overdue from source: 'pieces_past_due';
- X13,14 = Source performance in last 6 and 12 months: 'perf_6_month_avg', 'perf_12_month_avg';
- X15 = Amount of stock orders overdue: 'local_bo_qty';
- X16-21 = General risk flags: 'potential_issue=Yes', 'deck_risk=Yes', 'oe_constraint=Yes', 'ppap_risk=Yes', 'stop_auto_buy=Yes', 'rev_stop=Yes';
- Y= Product went on backorder: 'went_on_backorder=Yes'.

And for feature engineering, we have two approaches. The first one is a mathematical approach: we will raise the power of all continuous variables and create interactive terms for all binary features.

And as a result, we have more features listed below:



```
# original features
 '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',
 'potential_issue=Yes',
 'pieces_past_due',
 'perf_6_month_avg',
 'perf_12_month_avg',
 'local_bo_qty',
 'deck_risk=Yes',
 'oe_constraint=Yes',
 'ppap_risk=Yes',
 'stop_auto_buy=Yes',
 'rev_stop=Yes',
 'went_on_backorder=Yes',
```

```
# interactive terms
 'potential_issue=Yes deck_risk=Yes',
 'potential_issue=Yes oe_constraint=Yes',
 'potential_issue=Yes ppap_risk=Yes',
 'potential_issue=Yes stop_auto_buy=Yes',
 'potential_issue=Yes rev_stop=Yes',
 'deck_risk=Yes oe_constraint=Yes',
 'deck_risk=Yes ppap_risk=Yes',
 'deck_risk=Yes stop_auto_buy=Yes',
 'deck_risk=Yes rev_stop=Yes',
 'oe_constraint=Yes ppap_risk=Yes',
 'oe_constraint=Yes stop_auto_buy=Yes',
 'oe_constraint=Yes rev_stop=Yes',
 'ppap_risk=Yes stop_auto_buy=Yes',
 'ppap_risk=Yes rev_stop=Yes',
 'stop_auto_buy=Yes rev_stop=Yes',
 'potential_issue=Yes potential_issue=Yes',
 'deck_risk=Yes deck_risk=Yes',
 'oe_constraint=Yes oe_constraint=Yes',
 'ppap_risk=Yes ppap_risk=Yes',
 'stop_auto_buy=Yes stop_auto_buy=Yes',
 'rev_stop=Yes rev_stop=Yes'  ]]
```

```
# Powered features: Squared
'national_inv^2',
'lead_time^2',
'in_transit_qty^2',
'forecast_3_month^2',
'forecast_6_month^2',
'forecast_9_month^2',
'sales_1_month^2',
'sales_3_month^2',
'sales_6_month^2',
'sales_9_month^2',
'min_bank^2',
'pieces_past_due^2',
'perf_6_month_avg^2',
'perf_12_month_avg^2',
'local_bo_qty^2'
```

Table: Feature Engineering Process

The dataset has 1586967 rows and 37 columns after adding interaction terms and 1586967 rows and 52 columns after adding polynomial features.

The second approach is business understanding based. Based on domain knowledge, we created two features named 'forecast_mean' and 'sales_mean'. And they are the monthly mean value of the Forecast for the next 3,6 and 9 months and Sales quantity for the prior 1,3,6,9 months.

The final date set we have after feature engineering is shaped with 1586967 rows and 54 columns.

4. Data Standardization:

Since different models have different requirements for data, we will selectively perform data standardization. Specifically, the 4 models standardization conditions are listed below:

   a. Logistic Regression (NO Data Standardization)

b. SVM (With Data Standardization)

c. Neural Network (With Data Standardization)

d. Gradient Boosting (NO Data Standardization)

Modeling

To begin with, we define our task as a supervised learning classification problem. The target variable is a binary variable called "went_on_backorder=Yes". And we have 53 features and one target variable after feature engineering. In the following steps, we will choose among various algorithms, choose the optimal feature set for each algorithm, finely tune the parameters for each model and thus decide the best model for our task based on validation results.

1. Choices for Data Mining Algorithm

   a. Logistic Regression

   To solve this supervised learning classification problem, we started with a logistic regression model. Logistic Regression is a popular and simple approach for binary classification. In logistic regression, the response variable is a log function of odds of an event happening. This log function can take any value between -∞ and +∞. The shape of the end function obtained is no longer a linear difference between logistic and linear regression, it is in the choice of model parameters and the error function. Contrary to least square, the approach used in the logistic regression is minimizing log-likelihood function. Logistic shows improved sensitivity to class imbalance problems because of this.

We chose to start with logistic regression because it is simple and fast. It has the pros of being cost efficient and very suitable for this particular problem. We mainly used the validation result of logistic regression as a base line, to be compared with other more advanced algorithms that we have learnt from the Machine Learning II class.

When we need to choose the appropriate algorithm for data mining, there is always a tradeoff between accuracy and model comprehensibility. Generally, algorithms such as SVM, gradient boosting and Neural Network are more difficult to understand, and they may take more time to process, but they have the advantages of better generalization performances. For our particular data mining problem, because inaccurate predictions of backorder can induce large amounts of financial loss, we have decided that we care more about accuracy than speed and model comprehensibility. So, we then started to explore more advanced algorithms.

b. Support Vector Machine

If the data points are linearly inseparable, there can exist infinite many hyperplanes to separate the data. Support vector classifier tries to find the optimal hyperplane which is the farthest from the nearest training observations of each class. However, if the data points are not linearly separable, nonlinear class boundaries need to be incorporated. Support Vector Machines try to address this problem by enlarging the feature space using quadratic, cubic or higher-order polynomial functions of the predictors with the help of kernels.

As for the pros of supporting vector machine algorithms, it is suited for extreme case binary classification and it is really effective in the higher dimension. Even though for larger dataset, it requires a fair amount of time to process, we still considered SVM as a candidate for our final model.

c. Gradient Boosting

Ensemble modelling refers to building multiple diverse models to predict an outcome and boosting is one of the ensemble modelling techniques that we have learnt in class. It is a popular ensemble technique in which many weak classifiers are combined to create a stronger learner. Boosting sequentially builds estimators that try to get higher accuracy by focusing or placing more weight on the samples which were incorrectly classified by the previous learner.

As for Gradient boosting, it is a variant of boosting ensemble techniques. Gradient tree boosting uses decision trees as weak learners and tries to minimize the loss function using gradient descent algorithms. Each iteration tries to minimize the loss incurred in the previous iteration. Gradient tree boosting is robust to outliers and hence efficiently deals with the class imbalance problem.

Even though gradient boosting is a very powerful algorithm, we need to use it carefully because it is prone to overfitting. In the next steps, we have incorporated cross validation techniques and added appropriate parameters (depth of tree, etc.) to prevent overfitting.

d. Neural Networks

Neural networks are powerful nonlinear regression and classification techniques inspired by theories about how the brain functions. In Neural networks, outcome is modeled by an intermediary set of hidden units and each hidden unit has some weight parameter associated with it. Each hidden unit is a linear combination of some or all of the predictor variables. To introduce non-linearity, this linear combination is passed through a nonlinear function such as the logistic function. Back-propagation algorithm is used to find the optimal value of the parameters. In this algorithm, error is calculated at the output layer in each iteration and the weight parameters are optimized by taking derivatives of the error loss function with respect to the parameters. As a result, neural networks are particularly good at capturing non-linearity in the data and solving complicated problems.

However, neural networks are usually more computationally expensive than traditional machine learning algorithms. And NN is a "black box" algorithm, from which we can hardly gain business insights. But just as we have mentioned before, we care more about accuracy than speed and model comprehensibility for our particular task, so NN is a strong candidate for our final model.

2. Feature Selection

After feature engineering, we have 53 features in total. It is very important to perform feature selection because of three reasons. First, feature selection can enable machine learning algorithms to train faster. It is even more important for our models because we are using many advanced algorithms. Second, feature selection can improve the accuracy of a model if we have chosen the correct subset. Third it serves to prevent overfitting.

We performed feature selection in two steps:

a. Select features based on correlation

Correlation refers to how close two features are to having a perfect linear relationship with each other. Because highly-correlated features will induce almost the same effects on the target variable, so we should only keep one feature among highly-correlated features to prevent multicollinearity. Also, we have used correlation to remove some low-correlated features to our target.

After this step, we still have 33 features in total.

b. Forward Selection

We then used forward selection to select the best feature-set for each algorithm.

Algorithm	Selected Features	F1-Score after feature selection
SVM	'local_bo_qty', 'perf_12_month_avg', 'local_bo_qty^2', 'forecast_mean', 'national_inv'	0.73
NN	'potential_issue=Yes', 'oe_constraint=Yes, stop_auto_buy=Yes', 'potential_issue=Yes, ‘deck_risk=Yes', 'local_bo_qty^2', 'forecast_9_month', 'national_inv', 'sales_1_month', 'sales_3_month', 'sales_6_month', 'sales_9_month', 'in_transit_qty'	0.87

Gradient Boosting	'deck_risk=Yes, ppap_risk=Yes', 'perf_12_month_avg', 'forecast_mean', 'forecast_9_month', 'national_inv', 'sales_1_month', 'sales_3_month', 'stop_auto_buy=Yes', 'in_transit_qty', 'perf_12_month_avg^2', 'lead_time^2'	0.88

Table: Validation Results after Feature Selection

Forward selection is an iterative approach, where we start with zero feature and add one feature at a time that improves our model's performance the most. To be more particular, we have used forward selection in combination with stratified k-fold cross validation and F1-score. We will talk more about the validation techniques later in the report. In each iteration of forward selection, the feature that maximizes our average F1-score across five folds is chosen and added to the model. This process is repeated until the F1-score stops improving any more. As a result, we have found the optimal feature set for each algorithm.

We are also able to observe what features have the most predicting power to backorder. For example, 'national_inv' is chosen for all three models. National inventory is obviously an important feature when we try to predict the occurrence of backorder.

3. Hyperparameter Tuning

We finely tuned the parameters for each model by applying cross-validated grid-search over a parameter grid. More specifically, the first thing we did was to use "Kfold" function to create both an inner layer of our dataset and an outer layer of our dataset. The “Kfold” function with n_splits=5 can split data into five parts for cross validation purposes. We used the inner cross validation to search for the best parameters that maximize F1-score and used the outer cross validation to test the generalization performances of models after hyperparameter tuning.

For each model we tuned different parameters:

Algorithm	**Parameters**	**Meaning**	**Chosen Value**
Gradient Boosting	n_estimator	The number of boosting stages to perform.	200
	max_depth	The maximum depth of the individual regression estimators.	12
	min_sample_leaf	The minimum number of samples required to be at a leaf node.	9
Support Vector Machine	kernel	The kernel type to be used in the algorithm. For example, rbf refers to a radial basis function kernel.	rbf
	C	Regularization parameter. The strength of the regularization is inversely proportional to C.	1
Neural Network	activation	Activation function for the hidden layer. For example, relu refers to the rectified linear unit function.	relu
	alpha	L2 penalty parameter.	0.0001

Table: Parameters Adopted for Classifiers

4. Validation Approaches and Best Model

We have incorporated cross validation techniques in the previous processes of feature selection and hyperparameter tuning. To be more specific, we utilized stratified k-fold cross-validation. Stratified k-fold cross validation is the most appropriate technique to use when the data is unbalanced, therefore it is suitable for our particular problem. In stratified k-fold cross validation, each fold approximately has the same mean response value i.e. each fold has the same proportion of observations from each class. The main advantage of this technique of validation is that the model is fitted on a training set that is not biased towards a specific class.

The metric we chose to maximize along the way is F1-score, again because we have an imbalanced classification task. F1 is the harmonic mean of Precision and Recall and thus a better measurement compared to accuracy when we have uneven class distribution.

Based on the five-fold stratified cross validation results, we have chosen our best model for predicting backorder:

Algorithm	**Selected Parameters**	**Selected Features**	**F1 Score**
Gradient Boosting	n_estimator=200	'deck_risk=Yes, ppap_risk=Yes', 'perf_12_month_avg', 'forecast_mean', 'forecast_9_month',	0.903

	max_depth=12	'national_inv', 'sales_1_month', 'sales_3_month', 'stop_auto_buy=Yes', 'in_transit_qty', 'perf_12_month_avg^2', 'lead_time^2'	
	min_sample_leaf=9		

Table: Final Model

This model can accurately predict the occurrence of backorder with F1-score = 0.903, which will improve the company's ability to forecast backorder and react to the problem timely. By building and optimizing an accurate predictive model, we have "solved" the business problem. In the next part, we will talk about how we evaluate the model's financial impact.

Evaluation

To evaluate the actual effect of the gradient boosting model on Walmart's garment business, we need to calculate the cost matrix first. For false positives, the cost comes from the increase of warehousing cost, because if we predict a backorder when there is actually not one, it is likely that there will be an overstock, and will thus increase the warehousing cost. For false negatives, the cost comes from primarily two aspects, customer loss and potential profit loss. When customers fail to find what they want to buy, some of them will stop buying this kind of product from Walmart and switch to other stores for clothing goods in future. Thus, Walmart will lose potential revenue generated by this group of customers for a certain period of time due to this effect. In addition, because of backorder, Walmart cannot fulfill customers' demands and will lose profit from direct sales loss.

For the calculation of the costs, we use following facts:

- Walmart has total revenues of $514,405M and cost of sales of $385,301M in 2019, having a gross margin of (514,405 – 385,301) / 514,405 = 25.10%.
- According to a report by McKinsey & Company, the warehousing cost of retail industry is about 2.5% of total sales.
- According to a research by Statista, total sales from the clothing department for Walmart is expected to be $22.1B in 2019 and $22.4B in 2020.

For the calculation of the costs, we use following assumptions:

- Walmart's clothing department has similar gross margin as the rest of the corporation.
- Walmart's clothing department has a similar warehousing cost rate as the retail industry average.
- On average, each backorder causes 0.1% of the customer base to switch to other stores.
- The effect of customer base loss lasts for 2 years on average.
- The increase of warehousing cost due to overstock is 20% on average.
- There are 1586967 orders (total observations in the dataset) each year.

The calculated costs for each condition are:

- True Negative: Cost = 0
- True Positive: Cost = 0
- False Negative:
    - Customer Loss Cost = (Clothing Sales 2019 + Clothing Sales 2020) * Sales Loss Rate / Total Orders = (22.1B + 22.4B) * 0.1% / 1586967 = $28.04

    - Potential Sales Loss Cost = Gross Margin * Clothing Sales 2019 / Total Orders = 25.10% * 22.1B / 1586967 = $3495.09
    - Total Cost = Customer Loss Cost + Potential Sales Loss Cost = $28.04 + $3495.09 = $3523.13
- False Positive: Cost = Warehousing Cost Rate * Clothing Sales 2019 * Increase Rate of Cost / Total Orders = 2.5% * 22.1B * 20% / 1586967 = $69.63

The final cost matrix is:

	Predicted N	Predicted Y
Actual N	0	69.62968
Actual Y	3523.135	0

The final profit matrix is:

	Predicted N	Predicted Y
Actual N	0	-69.62967
Actual Y	0	3523.1351

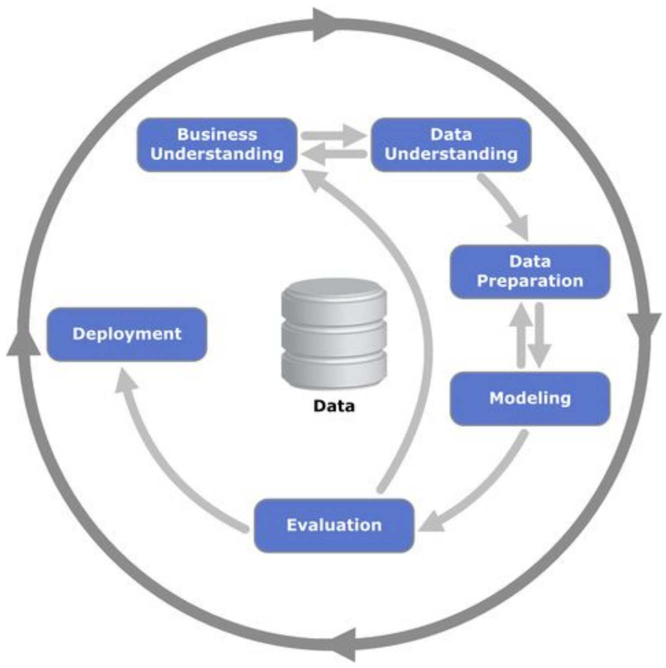
Confusion matrix of the best model:

	Predicted N	Predicted Y
Actual N	77.99%	21.32%
Actual Y	0.06%	0.63%

Thus, expected dollar amount impact on Walmart is 21.32% * 1586967 * -$69.63 + 0.63% * 1586967 * $3,523.14 = $11,494,262. In other words, the adoption of this predictive model in Walmart's clothing department is expected to increase Walmart's gross profit by $11.5M per year.

Business Recommendations & Deployment

For deployment, we recommend using the Cross-Industry Standard Process for Data Mining (CRISP-DM) framework as guidance in order to keep the deployment process iterative. This means potentially conducting trial runs on samples of the data. As well, we also wanted to remind businesses that overstocking, understocking and backorder costs differ business to business. Target's warehousing costs can be very different from Walmarts for example. As such, the cost function must be adjusted and must be re-evaluated. In a similar fashion, these costs can also differ industry to industry. The costs of overstocking in the food industry can be much more than the raw materials industry for example, thus again the cost function must be revised, along with the entire modeling process and weights on variables. Next, we also wanted to warn businesses that preventing backorder using our model should not come at the cost of ethics reasons. The best example of this is how businesses on Amazon overstocked on necessity goods and resell them at absurd prices. While our model can predict these surges in demand, profit from stocking up on hand sanitizer should not come at the cost of people losing lives. Lastly, industry and business differences can induce risks that must be counteracted. The details of counteracting risk are mentioned below.



Our model evaluation is based on many factors, and the accuracy of these factors can be low or change with time. Thus, this difference between expected benefit and actual benefit brings us potential risk, in which the application of this model may not be as beneficial as expected. In order to mitigate this risk, following steps should be taken:

1. Apply this model to only a portion of Walmart's clothing department
2. Collect data for the whole clothing department to see the actual effect of application and the difference from the control group
3. Include new data and re-run the model to improve the model performance
4. Adjust other factors, such as gross margin and customer loss rate, to improve calculation accuracy of profit matrix
5. Apply the new model to the whole clothing department
6. Repeat step 2 to 4 and apply to more departments
7. Repeat step 2 to 4 and apply to the whole market place

Work Cited/Appendix

Data Sources:

https://old.datahub.io/dataset/global-garment-supply-chain-data

Lean and mean: How does your supply chain shape up? *McKinsey & Company*.

https://www.mckinsey.com/~/media/mckinsey/dotcom/client_service/operations/pdfs/lean_and_mean-how_does_your_supply_chain_shape_up.ashx

Walmart Form 10-K, 2019. *SEC*.

https://www.sec.gov/Archives/edgar/data/104169/000010416919000016/wmtform10-kx1312019.htm#s25D406F0AEA257A487237EDC74B45CFC

Projected apparel and accessories gross merchandise sales volume of Walmart in the United States from 2011 to 2020. *Statista*.

https://www.statista.com/statistics/524829/walmart-fashion-sales-gmv/