Machine Learning Engineer Nanodegree

Capstone Proposal

Adarsha Badarinath

March, 2017

Proposal

Based on historical data predict backorder risk for products

Domain Background

Part backorders is a common supply chain problem. Working to identify parts at risk of backorder before the event occurs so the business has time to react.

What is backorder?

Popular items may sell out quickly and temporarily be on backorder. This means that the items are currently out of stock but that there are shipments on their way to re-stock our warehouses.

Why is it helpful to predict it??

Whenever a backorder situation occurs it is very easy for a competitor to woo the customer to their product and even sell it at a premium taking advantage of the scarcity. Also sometimes businesses have to give discounts to retain the customer. It affects the bottomline of any business.

Dataset I use is part of the recently released kagel dataset.

There are many existing theories about backorder prediction few are listed below.

- Backorder prediction can make use of inventory Forecasting and Aggregate Planning. I have liked to a material from UT Dallas.
- A paper from Indiana University where they conclude that it was possible to predict backorders statistically.

Problem Statement

Based on historical data predict backorder risk for products. The data is taken from kagel dataset. Using the dataset we predict 'went_on_backorder' column value given other input parameters. Explore different methods and different techniques to solve the problems. We want to answer the following questions on the dataset in the end

1. Given all the columns in the dataset predict 'went_on_backorder' column. Which technique is the best to do this?
2. Find out which are the top 5 features which has the most impact on our predictions

Inputs:

Input data format is in the form of a CSV

- sku - Random ID for the product - Integer
- national_inv - Current inventory level for the part - Integer
- lead_time - Transit time for product (if available) - Integer
- in_transit_qty - Amount of product in transit from source - Integer
- forecast_3_month - Forecast sales for the next 3 months - Integer
- forecast_6_month - Forecast sales for the next 6 months - Integer
- forecast_9_month - Forecast sales for the next 9 months - Integer
- sales_1_month - Sales quantity for the prior 1 month time period - Integer
- sales_3_month - Sales quantity for the prior 3 month time period - Integer
- sales_6_month - Sales quantity for the prior 6 month time period - Integer
- sales_9_month - Sales quantity for the prior 9 month time period - Integer
- min_bank - Minimum recommend amount to stock - Integer
- potential_issue - Source issue for part identified - Boolean
- pieces_past_due - Parts overdue from source - Integer
- perf_6_month_avg - Source performance for prior 6 month period - Integer
- perf_12_month_avg - Source performance for prior 12 month period - Integer
- local_bo_qty - Amount of stock orders overdue - Integer
- deck_risk - Part risk flag - Boolean
- oe_constraint - Part risk flag - Boolean
- ppap_risk - Part risk flag - Boolean
- stop_auto_buy - Part risk flag - Boolean
- rev_stop - Part risk flag - Boolean

Output:

Yes - Product went on back order

No - Product did not go to backorder

Machine learning task:

- The main task is binary classification problem between Yes/No of predicting if an item went on backorder

Datasets and Inputs

The data set is taken from Kaggle dataset. The dataset is already divided into test and training set. It has the following columns/features.

https://www.kaggle.com/tiredgeek/predict-bo-trial

- sku - Random ID for the product
- national_inv - Current inventory level for the part
- lead_time - Transit time for product (if available)
- in_transit_qty - Amount of product in transit from source
- forecast_3_month - Forecast sales for the next 3 months
- forecast_6_month - Forecast sales for the next 6 months
- forecast_9_month - Forecast sales for the next 9 months
- sales_1_month - Sales quantity for the prior 1 month time period
- sales_3_month - Sales quantity for the prior 3 month time period
- sales_6_month - Sales quantity for the prior 6 month time period
- sales_9_month - Sales quantity for the prior 9 month time period
- min_bank - Minimum recommend amount to stock
- potential_issue - Source issue for part identified
- pieces_past_due - Parts overdue from source
- perf_6_month_avg - Source performance for prior 6 month period
- perf_12_month_avg - Source performance for prior 12 month period
- local_bo_qty - Amount of stock orders overdue
- deck_risk - Part risk flag
- oe_constraint - Part risk flag
- ppap_risk - Part risk flag
- stop_auto_buy - Part risk flag
- rev_stop - Part risk flag
- **went_on_backorder - Product actually went on backorder. This is the target value.**

Class label distribution:

- went_on_backorder = Yes = 10914
  went_on_backorder = No = 1682136
  % of input which havewent_on_backorder = % of Yes/no = 0.645%

  We have a very skewed distribution of input towards orders NOT going on backorder.

Solution Statement

Our solution is to predict *using supervised machine learning* with highest possible accuracy if an order can go into backorder using the given features. We want to build a model which will handle this prediction and also be general enough to predict similar inputs in the future with same accuracy.

Data analysis

Analysis of data is made by looking at many aspects such as

- Size of data set
- What is the distribution of classification class in the training data
- Which of the classes can be combined without lose of information
- Analysis of any transformations that can be made on the data such as converting to other units
- Graphs to visualize the above statements

Preprocessing

- Convert any missing NaN and blank values to proper form of input
- Normalize the data for any such algorithms which require them like a neural network
- Load the data in python generators if its too big to load in memory

Evaluation

- I want to apply supervised learning models of the below
  a. Decision Trees - because they are ideal for classification
  b. Support Vector Machine - because they are ideal for classification
- I also want to try to apply a small Neural network to test how they compare with above classification. Neural networks are good at regression but we can compare their output and derive a classification based on output (e.g - Yes < .5 and No > .5)

Benchmark Model

NOTE: You can find all the model and data exploration in this notebook.

First model I considered:

A simple benchmark model predicting Backorder required = Yes(i.e we need to backorder) IF ‘stop_auto_buy’ column is true results in. Upon initial analysis I found this Part risk flag seems to be a good indicator.

Total size = 1693050

True positive = 10434

True negative = 65233

False positive = 1616903

False negative = 480

Precision = 0.64%

Recall = 95.6%

Accuracy = 4.4%

The above model suffers from a high recall but low precision.

Second model I considered is a “RandomForestClassifier”.

	precision	recall	f1-score	support
0	1.00	0.89	0.94	225210
1	0.07	0.82	0.13	2280
avg / total	0.99	0.89	0.93	227490

This has lower recall but higher precision and f-1 score.

Evaluation Metrics

I am using the evaluation metric of Precision as my primary evaluation metric. Along with this I will consider the F-1 score.

$$\text{Precision} = \frac{tp}{tp + fp}$$

Tp - True positive. | fp - False positive

$$\text{Recall} = \frac{tp}{tp + fn}$$

Fn - false negative

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Why?

In my usecase predicting a backorder places a realworld order to be dispatched to the wearhouse/shop. This means we will produce a product and have revenue impact. We should not do this unless absolutely required. Hence precision determines this. We select f1-score as secondary as it captures the overall picture including the recall score.

Project Design

My intended workflow

1. Import and explore the data set.
   a. Catch for any missing data
   b. Make sure it is not skewed in anyway so that learning algorithms get biased
   c. Compare it to common sense in real world
   d. Visualize different aspects of the data during exploration
2. feature selection: We can use a simple random forest algorithm to weight the "relevance" of features. Notice that this approach is biased towards what random forests assume as relevant, but it is often useful to get intuition overall.
3. Build multiple models of prediction - when all features are present
   a. Pipeline to prepare data for input into different models to do things such as normalization or adding new derived features (to reduce number of features).
   b. Explore multiple models of supervised learning and build models
   c. Predict using the models on the test data and pick the best model
      i. Model 1 - Decision Trees

    ii. Model 2 - Support Vector Machines
    iii. Model 3 - Neural network

4. Using the above models, build a more robust model when multiple features might be not not available in real world where data is messy.
    a. RANDOMLY drop many values in the training set
    b. Pipeline to prepare data for input into different models to do things such as normalization or adding new derived features (to reduce number of features).
    c. Explore multiple models of supervised learning and build models
    d. Predict using the models on the test data and pick the best model
5. Explore and answers questions on the data such as what is our prediction %. Also explore what a realworld company can do using this given information and also knowing that the model may not be 100% accurate.