NOVA IMS
Information Management School

Building a machine learning model to predict product back-orders

EDSA Bootcamp 2018

André Águas	M20170973@novaims.unl.pt
António Correia	M20170975@novaims.unl.pt
João Januário	M20170985@novaims.unl.pt
Valter Bento	M20170999@novaims.unl.pt

AGENDA

- Introduction
- Exploratory data analysis
- Data pre-processing
- Feature analysis
- Model pipeline and evaluation metrics
- Model results and proof of concept
- Conclusions

INTRODUCTION

What?

- BI4all
- Inventory Management
- Predicting Back Orders

Why?

- "Stock-Out"
- Signals fast growthing companies
- Reputational risk
- Risk to lose clients
- Inventory Risk (spoilage, theft, damage)

INTRODUCTION

How can we increase profit?

Excess inventory (increased storage costs)	Accurate predictions of stock needs and back-orders occurences	Unable to predict "stock-out" (increased cost of lost sales)

EXPLORATORY DATA ANALYSIS

- 8-weeks historical data
- Predict next week

Data types

- Current inventory
- Forecast
- Sales
- Performance

```
In [11]: df.shape

Out[11]: (1687861, 23)
```

```
In [9]: df.dtypes

Out[9]: sku                   object
        national_inv         float64
        lead_time            float64
        in_transit_qty       float64
        forecast_3_month     float64
        forecast_6_month     float64
        forecast_9_month     float64
        sales_1_month        float64
        sales_3_month        float64
        sales_6_month        float64
        sales_9_month        float64
        min_bank             float64
        potential_issue       object
        pieces_past_due      float64
        perf_6_month_avg     float64
        perf_12_month_avg    float64
        local_bo_qty         float64
        deck_risk             object
        oe_constraint         object
        ppap_risk             object
        stop_auto_buy         object
        rev_stop              object
        went_on_backorder     object
        dtype: object
```

EXPLORATORY DATA ANALYSIS

```
In [10]: df.isnull().sum()/len(df)*100

Out[10]: sku                     0.000000
         national_inv            0.000059
         lead_time               5.977625
         in_transit_qty          0.000059
         forecast_3_month        0.000059
         forecast_6_month        0.000059
         forecast_9_month        0.000059
         sales_1_month           0.000059
         sales_3_month           0.000059
         sales_6_month           0.000059
         sales_9_month           0.000059
         min_bank                0.000059
         potential_issue         0.000059
         pieces_past_due         0.000059
         perf_6_month_avg        0.000059
         perf_12_month_avg       0.000059
         local_bo_qty            0.000059
         deck_risk               0.000059
         oe_constraint           0.000059
         ppap_risk               0.000059
         stop_auto_buy           0.000059
         rev_stop                0.000059
         went_on_backorder       0.000059
         dtype: float64
```

```
In [14]: df.describe().T

Out[14]:
```

	count	mean	std	min	25%	50%	75%	max
national_inv	1687860.0	496.111782	29615.233831	-27256.0	4.00	15.00	80.00	12334404.0
lead_time	1586967.0	7.872267	7.056024	0.0	4.00	8.00	9.00	52.0
in_transit_qty	1687860.0	44.052022	1342.741731	0.0	0.00	0.00	0.00	489408.0
forecast_3_month	1687860.0	178.119284	5026.553102	0.0	0.00	0.00	4.00	1427612.0
forecast_6_month	1687860.0	344.986664	9795.151861	0.0	0.00	0.00	12.00	2461360.0
forecast_9_month	1687860.0	506.364431	14378.923562	0.0	0.00	0.00	20.00	3777304.0
sales_1_month	1687860.0	55.926069	1928.195879	0.0	0.00	0.00	4.00	741774.0
sales_3_month	1687860.0	175.025930	5192.377625	0.0	0.00	1.00	15.00	1105478.0
sales_6_month	1687860.0	341.728839	9613.167104	0.0	0.00	2.00	31.00	2146625.0
sales_9_month	1687860.0	525.269701	14838.613523	0.0	0.00	4.00	47.00	3205172.0
min_bank	1687860.0	52.772303	1254.983089	0.0	0.00	0.00	3.00	313319.0
pieces_past_due	1687860.0	2.043724	236.016500	0.0	0.00	0.00	0.00	146496.0
perf_6_month_avg	1687860.0	-6.872059	26.556357	-99.0	0.63	0.82	0.97	1.0
perf_12_month_avg	1687860.0	-6.437947	25.843331	-99.0	0.66	0.81	0.95	1.0
local_bo_qty	1687860.0	0.626451	33.722242	0.0	0.00	0.00	0.00	12530.0

DATA PRE-PROCESSING

- Data Conversion [String → Numeric] (“Yes/No” → “1/0”)
- Drop lines with NULL values (↓ 6% train data)
- Replace performance “NULL values” (-99) with median:

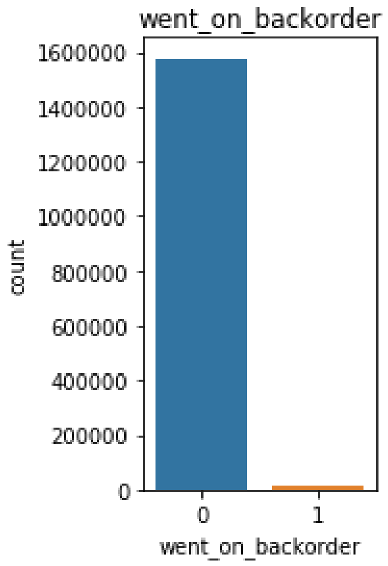
  perf_6_month_avg (≈ 2% train data)
  perf_12_month_avg (≈ 1% train data)
- Normalization

FEATURE ANALYSIS

Target variable - went_on_backorder

Despite having more than 1.5M observations on our set, we only have 10k observations **classified as 1, less than 1%.**

The small number of observations classified as 1's **might be a challenge** during training process of the models



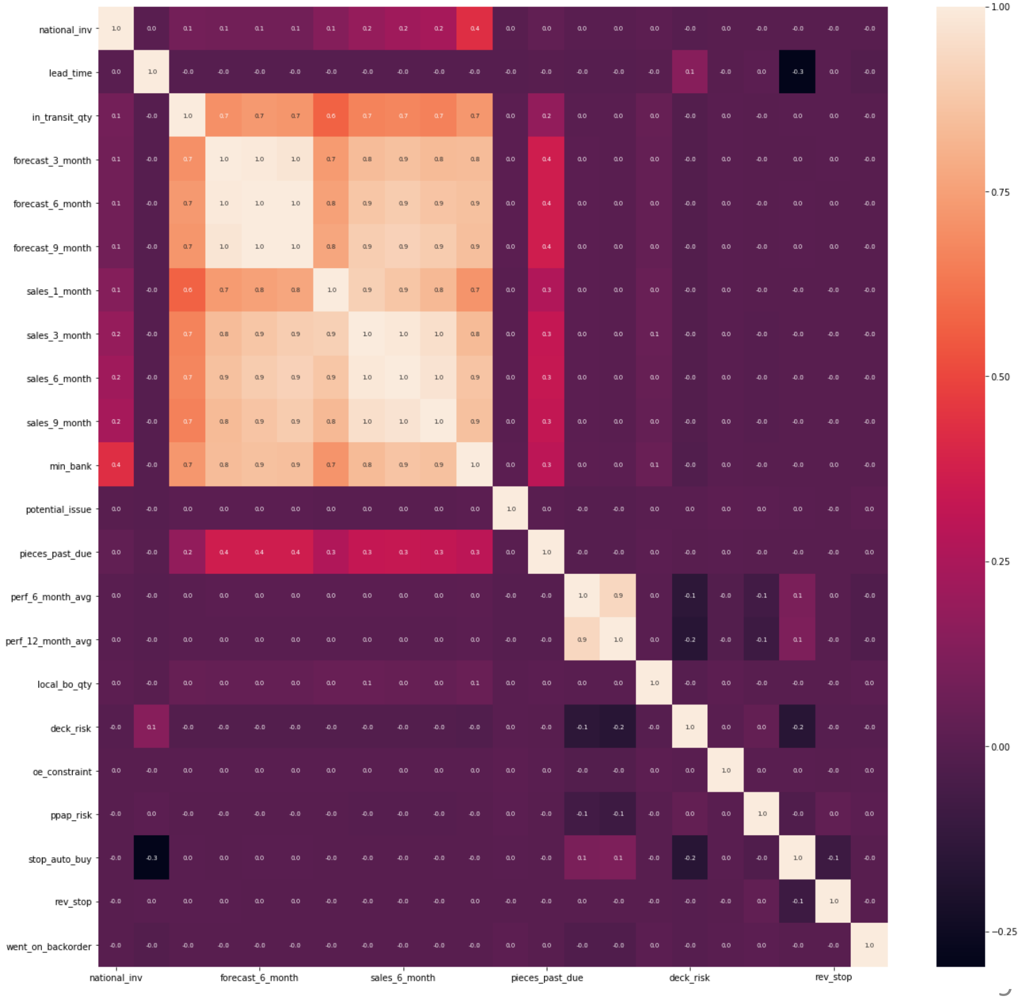
went_on_backorder		
0	1,575,998	99.3%
1	10,969	0.7%

FEATURE ANALYSIS

Relation between independent variables

Applying a correlation to all independent variables we determine a strong relation between three groups

- Forecast_3_month, forecast_6_month and forecast_9_month
- Sales_1_month, Sales_3_month, sales_6_month and sales_9_month
- Perf_6_month_avg and Perf_12_month_avg



FEATURE ANALYSIS

Relation between independent variables

Beside correlation between variables we performed an Random Forest model to assess the importance of a variable to the model accuracy and we compared the Mean and Std Deviation when went_on_backorder is 0 and 1

variables	RF Model importance	Mean		Std Dev	
		0	1	0	1
national_inv	3.54	499	21	29,715	608
forecast_3_month	1.43	178	157	5,042	1,635
forecast_6_month	1.08	346	245	9,826	2,457
forecast_9_month	0.58	508	326	14,425	3,145
in_transit_qty	0.50	44	4	1,347	47
sales_1_month	0.50	56	29	1,935	273
sales_3_month	0.41	176	79	5,210	509
stop_auto_buy	0.30	1	1	0	0
local_bo_qty	0.28	1	5	33	62
sales_6_month	0.24	343	139	9,645	901
sales_9_month	0.21	527	206	14,888	1,375
lead_time	0.19	6,006	2,875	23,744	16,693
perf_6_month_avg	0.19	-7	-3	27	19
perf_12_month_avg	0.18	-6	-3	26	18
min_bank	0.15	53	24	1,259	145
pieces_past_due	0.09	2	4	237	34
ppap_risk	0.08	0	0	0	0
deck_risk	0.07	0	0	0	0
potential_issue	0.00	0	0	0	0
oe_constraint	0.00	0	0	0	0
rev_stop	0.00	0	0	0	0

FEATURE ANALYSIS

Selecting best variables for our model

We have selected 11 of 21 variables as relevant features for use in model construction.

We have taken into account the correlation between variables, the importance according to the Random Forest and Mean and Std Deviation analysis

Initial Variables	Final Variables
national_inv	national_inv
forecast_3_month	forecast_3_month
forecast_6_month	
forecast_9_month	
in_transit_qty	in_transit_qty
sales_1_month	sales_1_month
sales_3_month	
stop_auto_buy	stop_auto_buy
local_bo_qty	local_bo_qty
sales_6_month	
sales_9_month	
lead_time	lead_time
perf_6_month_avg	perf_6_month_avg
perf_12_month_avg	
min_bank	min_bank
pieces_past_due	pieces_past_due
ppap_risk	
deck_risk	
potential_issue	potential_issue
oe_constraint	
rev_stop	

MODEL PIPELINE AND EVALUATION

Selecting and adjusting the best models: avoiding model overfitting

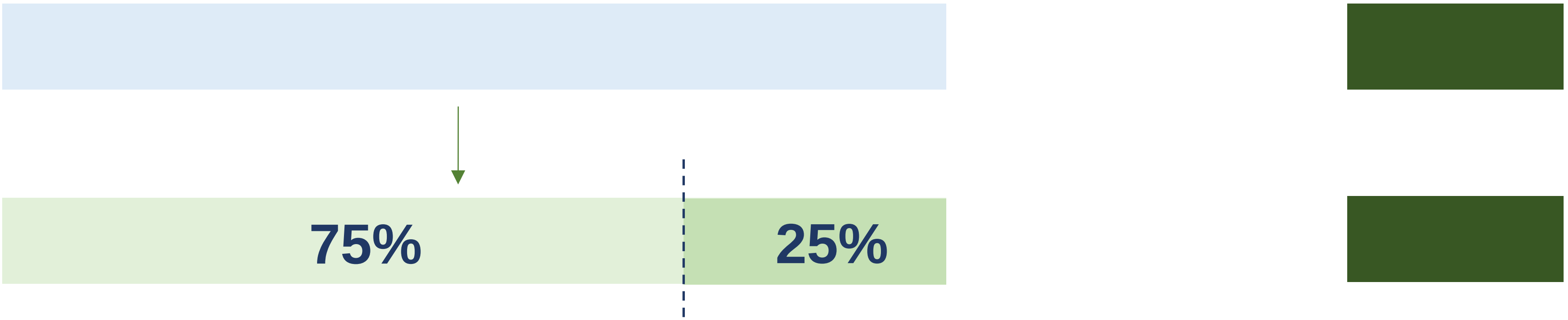
- The ‘Test’ dataset will be used to create a proof of concept, hence we do not want to adjust models based on it in order to avoid overfitting
- We will thus assess models and tweak their parameters by randomly splitting the ‘Train’ dataset into a Train and Validation subsets
- Then we can use the new Train subset to train different models with varying configurations and use the Validation set to check their performance

MODEL PIPELINE AND EVALUATION

Selecting and adjusting the best models: avoiding model overfitting

Original training set

Original test set



Training subset

Train several models, with different parameters each

Validation subset

Evaluate performance before deployment

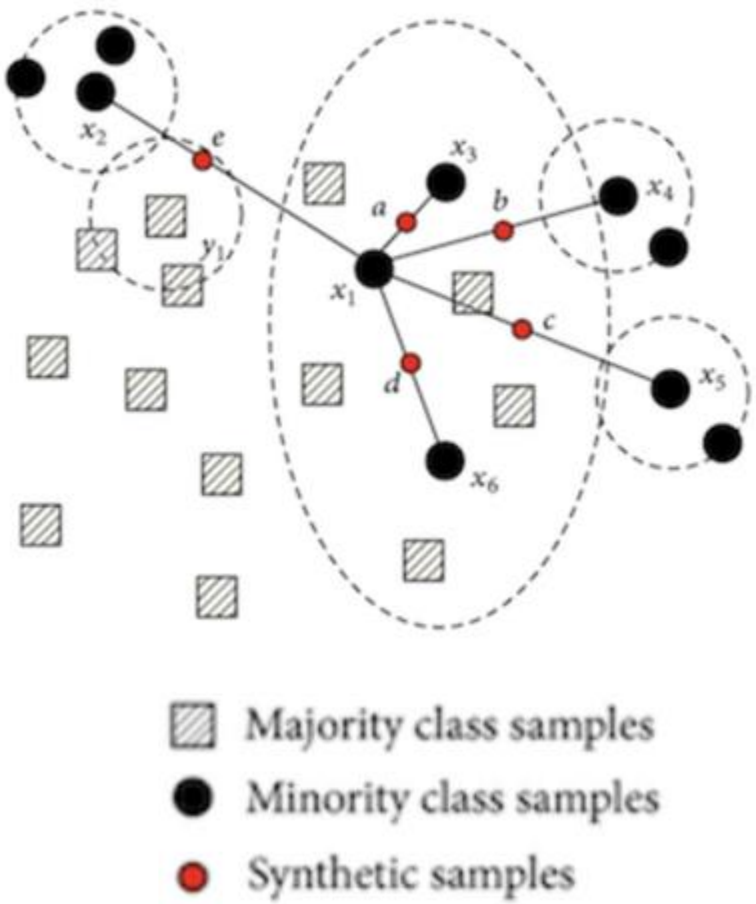
Original test set

Make predictions using the best model(s) to demonstrate potential in production

MODEL PIPELINE AND EVALUATION

Addressing data imbalance

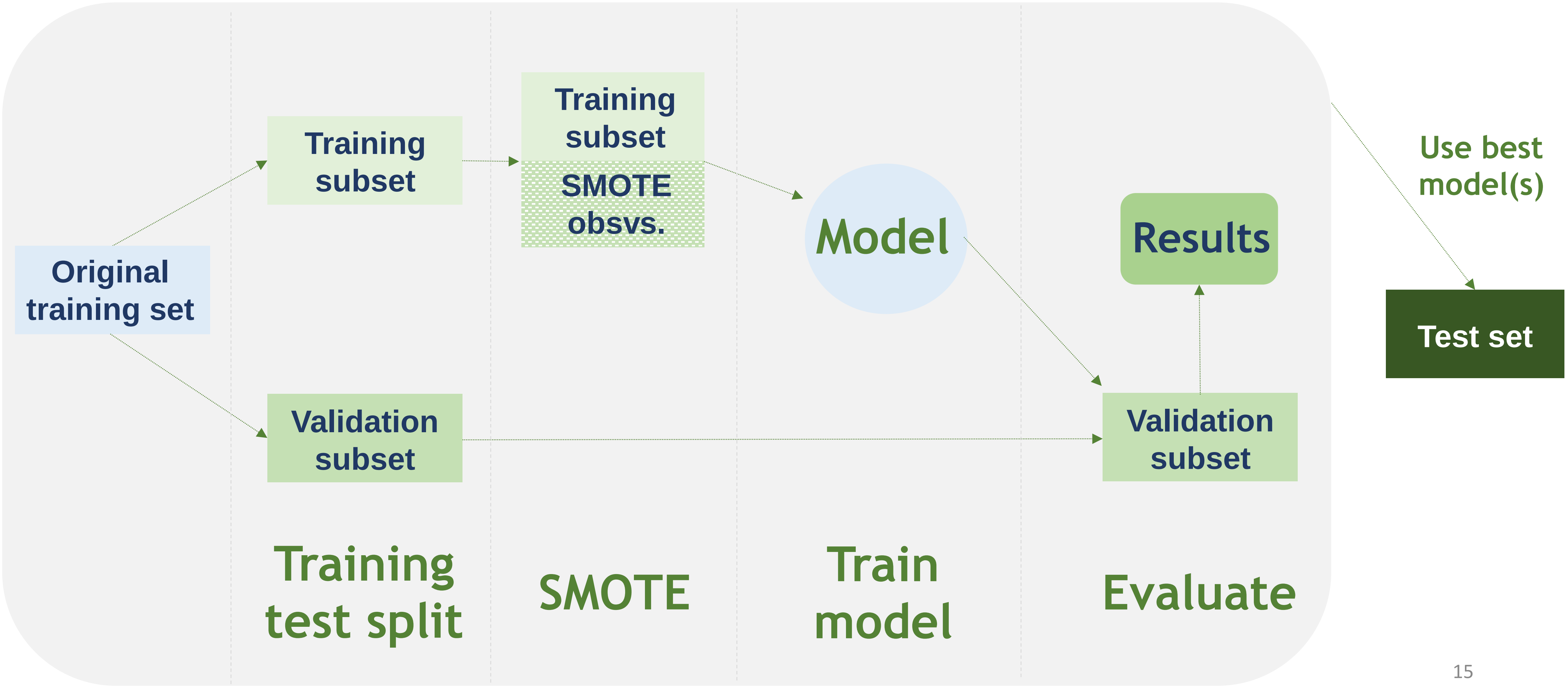
- Our dataset is extremely imbalanced, and we address this issue with an over-sampling algorithm
- SMOTE artificially creates observations based on the K nearest neighbors of the minority class observations
- To use SMOTE correctly, we should apply it only to our training subset (i.e., after splitting the original training set)



From James blogs (link)

MODEL PIPELINE AND EVALUATION

Model pipeline



MODEL PIPELINE AND EVALUATION

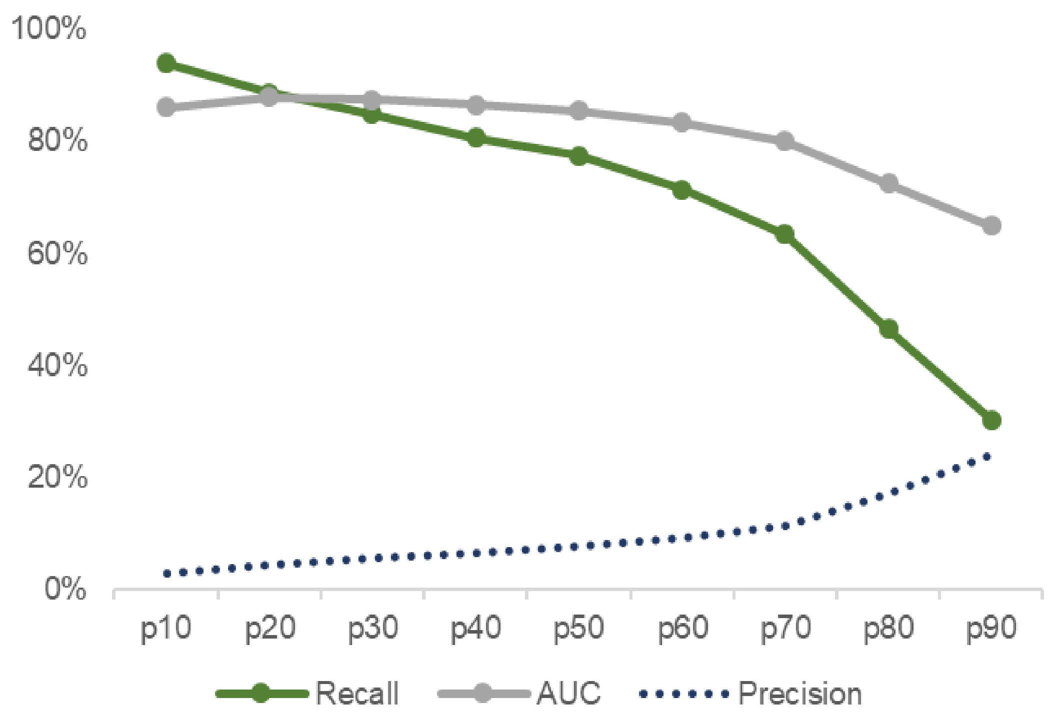
Model performance evaluation metrics: a purely statistical approach

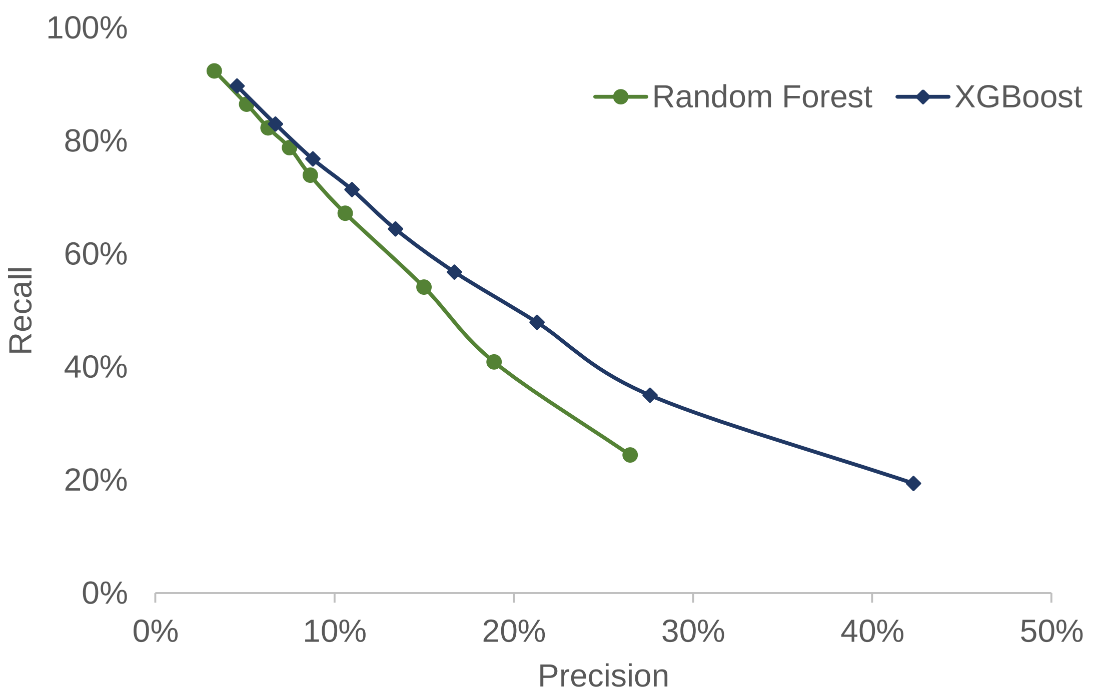
- In our opinion, accuracy is a poor measure of performance in a classification task with such a severe imbalance*
- We focus mostly on
  - Recall: percentage of back orders correctly predicted
  - Area under the ROC curve (AUC): probability that a positive record is given a higher likelihood of being a back order than a negative record
- We also monitor precision to get a sense of how many incorrectly predicted back orders our model produces

*Predicting that all records will equal 0 yields an accuracy of 99%

MODEL PIPELINE AND EVALUATION

Model performance evaluation metrics: a purely statistical approach





MODEL PIPELINE AND EVALUATION

Model performance evaluation metrics: a business cost approach

	Pred 0	Pred 1
True 0	**344,318**	49,697
True 1	295	**2,432**

Recall = 89%
Precision = 5%

- Is a model any good if it correctly predicts 89% of all true back orders, but only 5% of the model's predicted back orders become true?

- If the company faces a cost when incorrectly predicting a back order (e.g., because it is stocking up on inventory that it doesn't need), then a model with 5% precision might be highly undesirable, depending on the cost
- From the business case provided, we cannot know such costs, but we can try to rank models accordingly

MODEL PIPELINE AND EVALUATION

Model performance evaluation metrics: a business cost approach

	Pred 0	Pred 1
True 0	**344,318**	49,697
True 1	295	**2,432**

Suppose:

- The cost of a False Positive (FP) is x
- And the cost of a False Negative (FN) is y
- The model would produce a cost of $295y + 49{,}697x$

- To simplify, we assume the baseline case is that the firm does not try to predict back orders at the moment. Therefore, it faces a cost equal to $(295 + 2{,}432)y$
- The firm should adopt the model if $295y + 49{,}697x < (295 + 2{,}432)y$

MODEL PIPELINE AND EVALUATION

Model performance evaluation metrics: a business cost approach

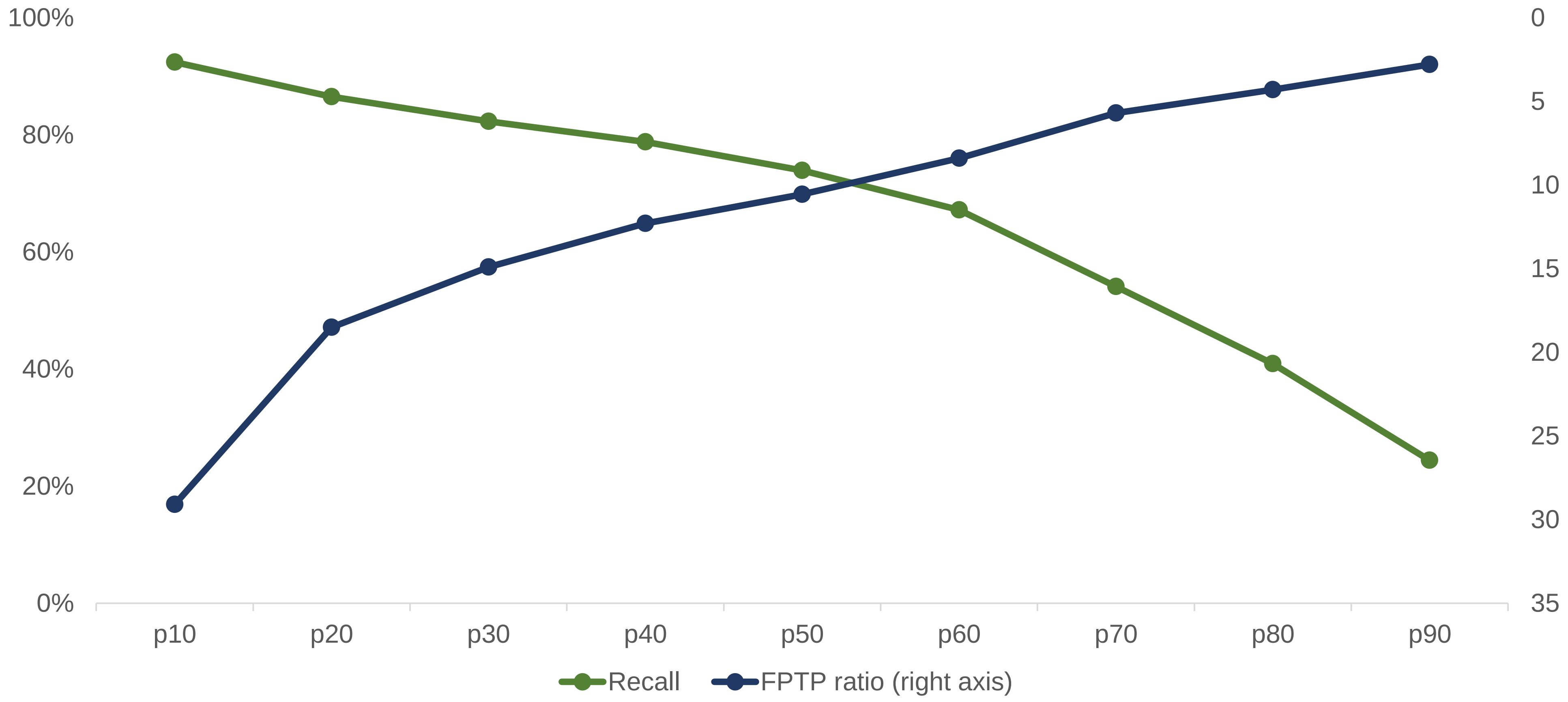
- More generally, the firm should adopt the model when $(FN + TP).y > FN.y + FP.x$, i.e., when the baseline cost is greater than the model cost
- This is the same result as

$$\frac{FP}{TP} < \frac{y}{x}$$

Which means that the firm should always adopt a model when the ratio FP/TP is smaller than the ratio of back order costs (y) to excess inventory costs (x). Therefore, in addition to recall and AUC, we will favor models with low FP / TP (let us call it FPTP ratio)

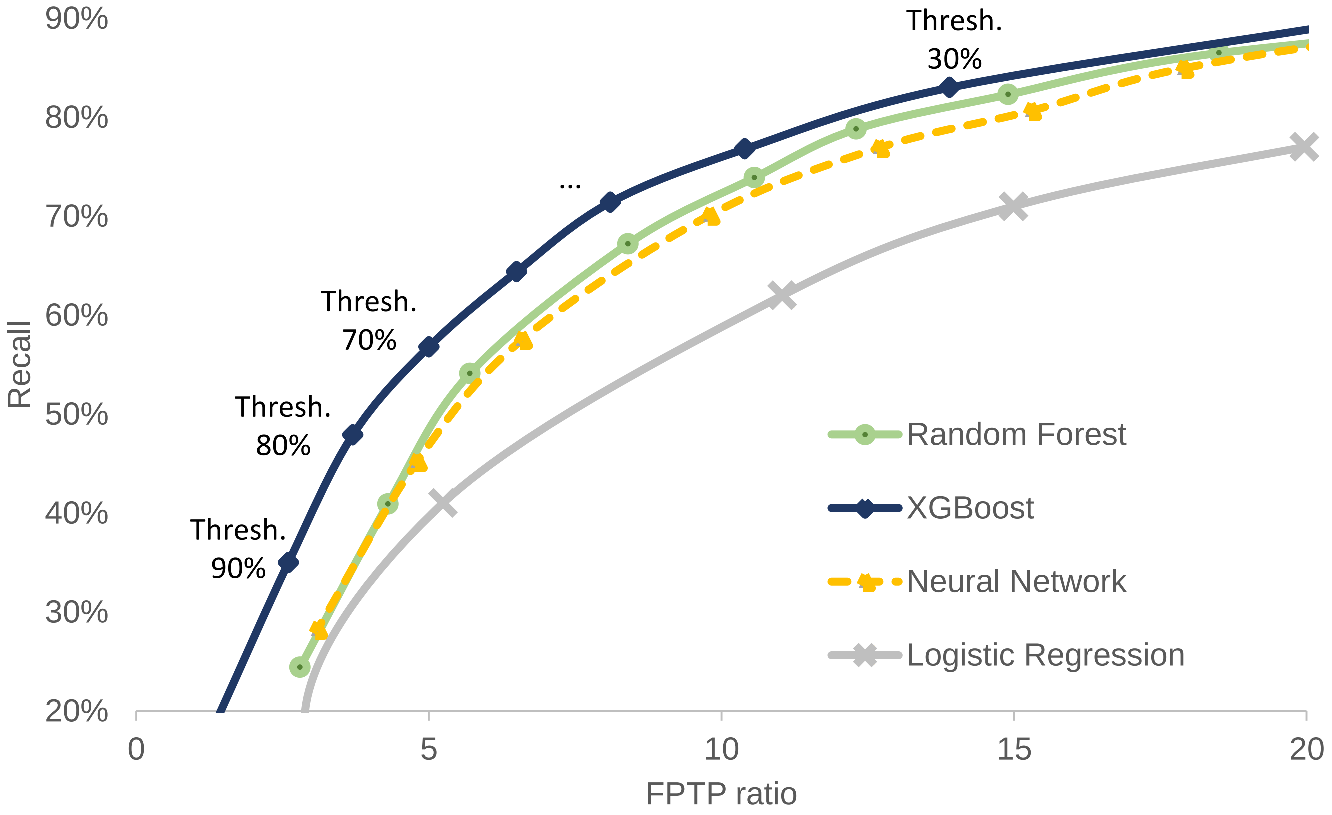
MODEL PIPELINE AND EVALUATION

Model performance evaluation metrics: a business cost approach



MODEL PIPELINE AND EVALUATION

Model performance evaluation



Model	Recall	AUC	Precision	FPTP
Random forest with 1,000 trees	74%	84%	9%	10,6
Balanced RF with 1,000 trees	89%	86%	4%	20,4
KNN with 3 neighbors	43%	76%	16%	5,1
Logistic Regression	72%	82%	6%	14,9
Neural Network	75%	94%	8%	12,1
XGBoost with 1,000 trees	64%	81%	13%	6,5

MODEL RESULTS AND PROOF OF CONCEPT 🎯

Model performance in predicting the Test set

- How would our model stack up in predicting the following week?
- Recall = 67%, FPTP ratio = 9.2, AUC = 80%
- Perhaps more importantly, **how much would the company save**?

	Pred 0	Pred 1
True 0	**208,683**	16,064
True 1	850	**1,754**

→

- For the sake of the example, suppose the expected cost of a backorder is 10€ and the cost of a False Positive is 1€ (e.g. stocking up on too much inventory
- Cost without predictions = $(850 + 1{,}754) \times 10€ = \mathbf{26,040€}$
- Costs when using the model = $850 \times 10€ + 16{,}064 \times 1€ = \mathbf{16,914€}$

- The company would have saved 9,126€ in 1 week, which in annual terms would imply a cost saving above 470,000€

CONCLUSIONS

- In a prediction problem with imbalanced data, accuracy is a poor performance measure, and a cost-benefit analysis of the outcomes of our model can be more insightful

- Taking a more statistical approach, we would be inclined to say that our Neural Network model is our best model (high AUC)

- However, taking into account potential business costs, the NN model might be too “imprecise”, and we’d propose using the XGBoost model, which could be more profitable for the firm

THANK YOU

ANNEX: ASSESSING FEATURE IMPORTANCE

- In order to use only relevant features, we calculate feature importance in a random forest model through the so called “Gini importance”, which is measured as is defined as the total decrease in node impurity (weighted by the probability of reaching that node) averaged over all trees of the ensemble.

- The Gini importance essentially evaluates the decrease in model accuracy when one randomly permutes the values for a feature. The higher the decrease, the greater the importance in that feature. A higher level of the measure indicates the feature is more relevant.

	importance
national_inv	3.54
forecast_3_month	1.43
forecast_6_month	1.08
forecast_9_month	0.58
in_transit_qty	0.50
sales_1_month	0.50
sales_3_month	0.41
stop_auto_buy	0.30
local_bo_qty	0.28
sales_6_month	0.24
sales_9_month	0.21
lead_time	0.19
perf_6_month_avg	0.19
perf_12_month_avg	0.18
min_bank	0.15
pieces_past_due	0.09
ppap_risk	0.08
deck_risk	0.07
potential_issue	0.00
oe_constraint	0.00
rev_stop	0.00

ANNEX: CHOOSING AN APPROPRIATE SMOTE RATIO

Model	Recall	AUC	Precision	FPTP ratio
Random forest with 100 trees, SMOTE = 0%	0%	50%	100%	0,0
Random forest with 100 trees, SMOTE = 3%	24%	63%	30%	2,3
Random forest with 100 trees, SMOTE = 5%	36%	67%	23%	3,3
Random forest with 100 trees, SMOTE = 10%	48%	73%	18%	4,6
Random forest with 100 trees, SMOTE = 20%	67%	82%	11%	8,4
Random forest with 100 trees, SMOTE = 30%	**74%**	**84%**	**9%**	**10,5**
Random forest with 100 trees, SMOTE = 40%	**77%**	**85%**	**8%**	**11,8**
Random forest with 100 trees, SMOTE = 100%	85%	87%	6%	17,0

- The table above shows the KPIs in a Random Forest model for different levels of the SMOTE ratio
- Given the trade-off that we believe exists between Recall and FPTP, we concluded that the most balanced performance is achieved in the 30-40% proportion of minority class samples (generated using SMOTE)
- We decided to use 30% since it achieves similar performance and is computationally “less expensive” than 40%

ANNEX: PARAMETERIZING XGBOOST

Screenshot 1

```
params = {
        'gamma': [1, 1.5, 2],
        'subsample': [0.6, 0.8, 1.0],
        'colsample_bytree': [0.6, 0.8, 1.0],
    'learning_rate': [0.01, 0.02, 0.03, 0.04],
        }
```

```
xgb = XGBClassifier(n_estimators=600, max_depth=10, min_child_weight=5, objective='binary:logistic',
                    silent=True, nthread=1)
```

```
folds = 5
param_comb = 4

skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)

random_search = RandomizedSearchCV(xgb, param_distributions=params, n_iter=param_comb,
                                   scoring='roc_auc', n_jobs=2, cv=skf.split(X_train_res, y_train_res),
                                   verbose=3, random_state=1001 )
```

We chose XGBoost's hyperparameters through cross-validated randomized search, optimizing for AUC and with the initial set of parameters shown in screenshot 1

Screenshot 2

```
random_search.best_estimator_
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=0.6, gamma=1, learning_rate=0.03, max_delta_step=0,
       max_depth=10, min_child_weight=5, missing=None, n_estimators=600,
       n_jobs=1, nthread=1, objective='binary:logistic', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=True, subsample=0.8)
```

The best parameters are shown in screenshot 2