DS Recruitment challenge

1. **Maths & Statistics**: Modelling, understanding and predicting:
2. **Business understanding & value creation.**
3. **Data engineering, deployment & usability**

**Data analysis.**

There are many more non – returned items than returns. This is generally a good thing but can cause imbalance in dataset when training model.

A graph of a number of numbers

Description automatically generated

1. **Modelling the data**

3 models tested: Hist Gradient Boost/Gradient Boost, XGBoost and Random Forest

Hist Gradient boost does not run. Requires dense matrix and computer crashes. Gradient Boost gives low performance. Now focusing on XGBoost: see Table 1

TODO: Add plot here that compares the different model performance

Score 0.9519374996687355

classification report XGBoost

precision recall f1-score support

0 0.95 1.00 0.98 179480

1 0.60 0.04 0.08 9191

accuracy 0.95 188671

macro avg 0.78 0.52 0.53 188671

weighted avg 0.94 0.95 0.93 188671

Table 1

The low recall is likely due to data imbalance. Let’s oversample the minority class (Table 2). This improves recall but decreases precision.

Score 0.786119753433225

classification report XGBoost with resampling

precision recall f1-score support

0 0.98 0.79 0.88 179480

1 0.15 0.75 0.25 9191

accuracy 0.79 188671

macro avg 0.57 0.77 0.56 188671

weighted avg 0.94 0.79 0.84 188671

F1 score 0.2535654168439356

Table 2

Random forest performance is low on recall & precision. It also is very time-consuming to run. For a simple model training with only n\_estimators = 10, it took +-90 minutes. Ideally we can try multiple conditions but given time constraints and computational resources this is not the best way forward.

Score 0.9412787338806706

classification report RF

precision recall f1-score support

0 0.96 0.98 0.97 179480

1 0.24 0.10 0.14 9191

accuracy 0.94 188671

macro avg 0.60 0.54 0.55 188671

weighted avg 0.92 0.94 0.93 188671

F1 score Random Forest 0.1377539108101798

Table 3

In conclusion, from the models tested XGBoost with resampling performs best. Like shown above the recall is ok (0.75) but poor precision (0.15). In order to improve precision it would be useful to have more data that can somewhat compensate for the data imbalance.

Other options that can still be investigated will be to fine-tune the hyper parameters of the model.

* Hyper parameters to test can be regularization, increasing regularization to increase the precision (this can help to reduce the complexity of the model and prevent overfitting)

1. **Business understanding and value creation.**

How can we implement this model in a useful way? Ideally, we want to reduce the amount of returns in all companies. This has multiple reasons:

* **Costs**: It costs the company more money. They must pay for the return label.
* **Sustainability**: More returns, more resources needed, also it is more difficult to resell used items.
* **Labour intensive:** somewhat related to costs, but more people and space needed to process returns.

Having a model that can predict if an item is likely to be returned with >=70% accurary will be useful.

TODO: check if model can return probability rate of return

Implementation ideas:  
Can be implemented as an API. Every time new product comes in the catalogus model **predicts return probability.** If probability is higher than a certain threshold, let’s assume for now that this is 70% than item get flagged.

Flagged items can be further investigated: Are there certain companies that sell these products? In what kind of categories do these items fall?

This can lead to actions towards the companies or types of items.