# Identifying Fraud

An ML Approach

# Executive Summary

Objective - to quantify how an ML approach can help identify fraudulent transactions for the Fraud Team to analyse

Data - ~900 (0.7%) confirmed fraud cases in 120k transactions over 12 months

Key Findings

- ML approach can identify suspect fraud transactions
  - Limitations: fraud team capacity able to process 400 suspected transactions a month)
  - Best results show ability to capture about ⅔ (67%) of fraudulent transactions
  - Alternatives: At increased fraud team capacity of 1.5x → ca 77%, 2x → ca 82%
- Basic analytics show hotspots of fraud
  - 1 / 11 transactions in country code 840
  - 1 / 30 transactions in merchant code 5735

# Results - Experiment summary

The following experiments have been run

- Metric: Maximise recall under budget constraint
- Models: XGBoost, LightGBM, GBM
- Use 3 different data frames:
  - Pipe 1: One-hot encoding of top categories in data (ca 150 features)
  - Pipe 2: Adding top features based on account and lagging transactions (ca 200 features
  - Pipe 3: Like 2 with more features, top features covering ca 90% of data (ca 500 features
- Downsampling of data to balance labelling. 3 random draws of downsampling performed
- Three ratios considered between neg/pos labels: 1:1, 2:1, 3:1
- Three budget scenarios: capacity 400/600/800 per month (test data set is equivalent to 3 months of transaction data)
- 'Select' models use most relevant features based on model feature importance
- Best results achieved around (best models highlighted in red)
  - Ca 67% for 1x Budget
  - Ca 77% at 1.5x Budget
  - Ca 83% at 2x Budget
- Next steps:
  - Consider a voting model approach
  - Consider additional features while using the 'Select' approaches

| | Pipe | Pipeline 1 | | | Pipeline 2 | | | Pipeline 3 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Label Ratio | 1:1 | 2:1 | 3:1 | 1:1 | 2:1 | 3:1 | 1:1 | 2:1 | 3:1 |
| Budget | Model | | | | | | | | | |
| 1.5x budget | GBM | 0.691 | 0.707 | 0.763 | 0.690 | 0.722 | 0.751 | 0.696 | 0.745 | 0.763 |
| | LGBM | 0.696 | 0.710 | 0.739 | 0.678 | 0.729 | 0.723 | 0.681 | 0.724 | 0.729 |
| | LGBM-Select | 0.678 | 0.707 | 0.718 | 0.663 | 0.717 | 0.711 | 0.671 | 0.731 | 0.724 |
| | XGB | 0.686 | 0.720 | 0.730 | 0.686 | 0.738 | 0.732 | 0.695 | 0.734 | 0.738 |
| | XGB-Select | 0.672 | 0.714 | 0.718 | 0.681 | 0.731 | 0.729 | 0.690 | 0.734 | 0.735 |
| 1x budget | GBM | 0.556 | 0.598 | 0.647 | 0.548 | 0.635 | 0.650 | 0.577 | 0.619 | 0.637 |
| | LGBM | 0.568 | 0.632 | 0.653 | 0.543 | 0.641 | 0.660 | 0.552 | 0.643 | 0.653 |
| | LGBM-Select | 0.552 | 0.612 | 0.616 | 0.530 | 0.636 | 0.638 | 0.538 | 0.647 | 0.661 |
| | XGB | 0.562 | 0.623 | 0.668 | 0.580 | 0.646 | 0.648 | 0.594 | 0.618 | 0.647 |
| | XGB-Select | 0.559 | 0.602 | 0.635 | 0.573 | 0.628 | 0.636 | 0.590 | 0.621 | 0.648 |
| 2x budget | GBM | 0.770 | 0.793 | 0.819 | 0.758 | 0.803 | 0.795 | 0.770 | 0.806 | 0.831 |
| | LGBM | 0.761 | 0.770 | 0.795 | 0.732 | 0.775 | 0.764 | 0.729 | 0.777 | 0.767 |
| | LGBM-Select | 0.757 | 0.760 | 0.778 | 0.736 | 0.769 | 0.752 | 0.733 | 0.771 | 0.762 |
| | XGB | 0.760 | 0.778 | 0.796 | 0.750 | 0.780 | 0.766 | 0.755 | 0.785 | 0.780 |
| | XGB-Select | 0.753 | 0.771 | 0.770 | 0.737 | 0.801 | 0.771 | 0.749 | 0.786 | 0.784 |

# Appendix: Modelling approach

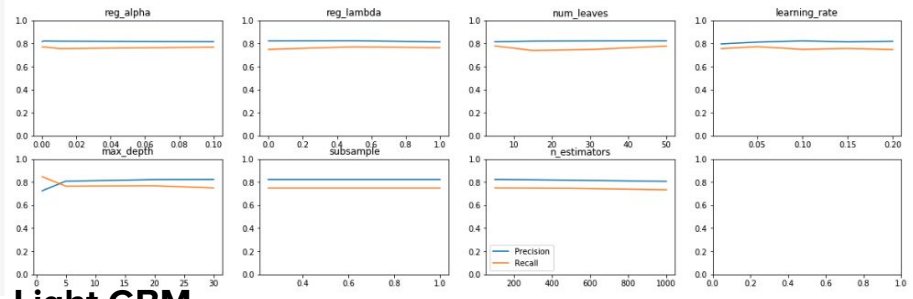
| Step | Actions/Choices | Reference (function names) |
|---|---|---|
| Data prep/Data analytics | Initial data clean up included minor replacements of missing values or '...'<br>Total % of fraud establishes a baseline and highlights the need to downsample<br>Statistics drawn on each feature, eg % fraud by country etc help find outlier cases | summary_stats<br>build_clean_df |
| Feature Engineering | In addition to one hot encoding<br>Engineering features based on:<br>- Transaction history, ie previous transaction country, merchant zip, time since previous, etc<br>- Account profile: size of typical transactions, country of typical transactions, etc | feat_cat_freq<br>account_features<br>add_prev_transaction |
| Data set splits | 25% held out as test data - this is approximately 3 months of data and can this way be used to compare against the 'budget' of 400 transactions to be reviewed per month<br>Rest downsampled to 1:1 / 2:1 / 3:1 proportion to positive labels. Results should be an average against multiple downsampling exercises | Run_big_loop in main file |
| Model choices | LightGBM, XGBoost, GBM - selected for the comparison as tend to be most performant across a large number of cases<br>Linear models considered just as baselines but not included as performance much worse<br>Deep NNs not considered as the data set of positive cases is small | Run_big_loop in main file |
| Hyperparameter tuning | See next page | Run_hyperparameters in main file |
| Feature Importance | Can be done by aggregating tree estimators or methods such as LIME, or SHAP. Here simply used SelectFromModel method from sklearn which leverages the models' build-in feature importance tooling | |
| Evaluation | Recall on the highest probability predictions under budget constraint<br>Alternative budget scenarios supplied as multiples 1.5x and 2x<br>Evaluation on entire test set (25%) equivalent to ca 3 monts (3/12) of data hence budget = 1200 | |

# Appendix: Results - Hyperparameter choice

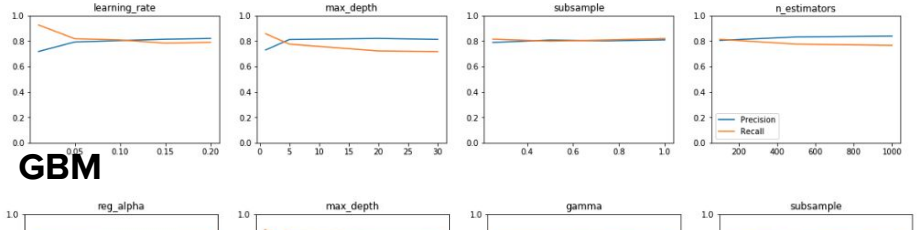
Hyperparameter tuning on single dimensions.

Aim to get best precision as number of recommendations is fixed (ie 400 per month) therefore highest precision will give the most accurate results. This is a heuristic and mostly holds assuming fairly flat effects on recall

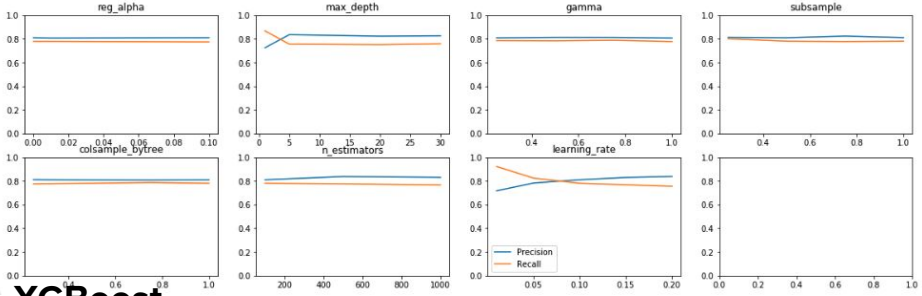
Overall results fairly flat. Slight improvement with increase of estimators and learning rate




**Light GBM**




**GBM**




**XGBoost**