Capstone Project 1

Fraud Detection in Mobile Payment Data
PaySim

Springboard Data Science Career Track
Eric Cruz

Introduction

Problem Statement

Online fraud is increasing and spreading rapidly across geographies and industries, and Mobile Payments represent a significant portion of the growth in both overall transaction and fraud rates. While attempting to detect and prevent fraud, the accuracy of the prediction models can have a significant impact on the ability to strike the right balance between true detection and false positives. The customer experience can be severely compromised by security measures enacted on the basis of a false positive. The fraud rate has a measurable impact on revenue, and new types and methods of fraud are evolving in response to successful detection and prevention efforts.

Dataset

Due to privacy concerns, there is little if any publicly available data for real transactions. **PaySim.csv** is a simulation of mobile money transactions with the objective to generate a synthetic transactional data set that can be used for research into fraud detection.
The dataset for this project is from the following link on Kaggle: https://www.kaggle.com/ntnu-testimon/paysim1

Clients

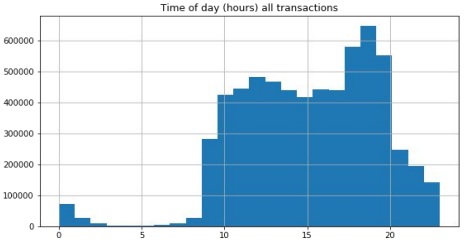
The intended clients are financial institutions and merchants who use mobile payments, and will incorporate the findings into a Fraud Detection and Prevention program which covers various types of fraud attacks.

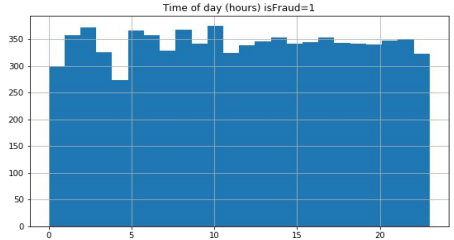
Approach

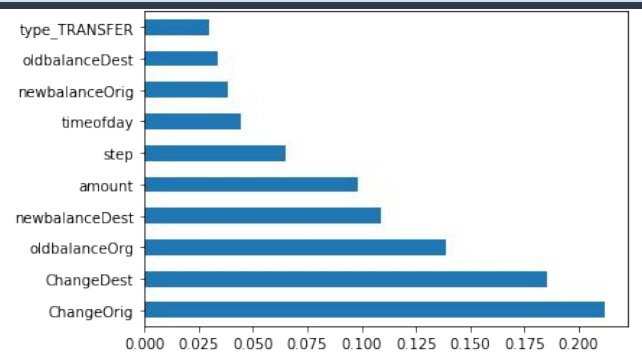
The approach is to select a few models, and determine pre-processing and feature selection upfront. As the data does not contain features other than amounts, balances, account types, and fraud flags, the explanatory features must be engineered by combining characteristics of the given fields. The PCA dimensionality reduction technique will be included in the pipeline across models, as it can be useful in classification problems and improve run time. Due to the imbalanced nature (e.g. fraud proportion is small relative to the population) of the data, SMOTE, or Synthetic Minority Oversampling Technique is also tested alone and with PCA at the same time.. Starting with 8 models from 5 categories, we compare baseline results using default parameters on a set of performance metrics. Along the way, some parameter tuning was employed but only the best model, Random Forest Classifier, is used to depict the stages of model tuning in terms of measuring performance.

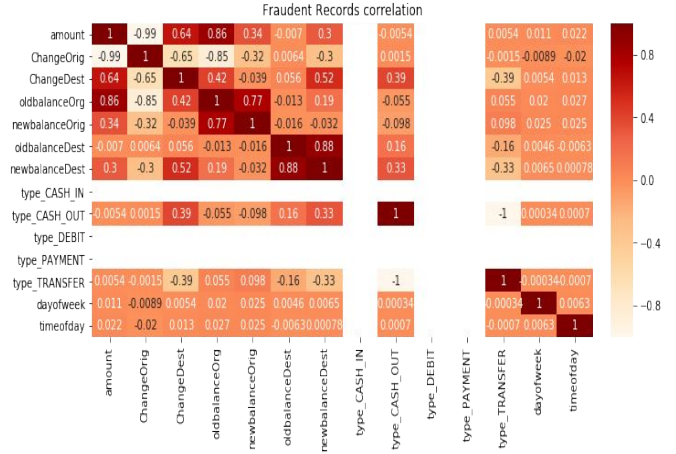
EDA, Feature Selection, Model Pipeline

- Categorical field converted to numeric boolean for 5 transaction type values
- Drop account name fields with Customer vs Merchant code because no Merchant balances
- Feature engineering to create Change Balance and Time of Day features. Visualizations show time pattern, feature importance measure, and correlation matrix
- Standard Scaler to normalize data will be added to the model pipeline.
- Train, Test, Split partitioning with test_size=0.3 (default is 0.25)
- GridSearchCV with default cv=5 will be used for cross validation









Test Parameters and Results

Supervised, Inferential, Unsupervised, Neural Network, Ensemble Bagging model categories

Metrics of Accuracy, F1, Precision, Recall, and ROC-AUC shown with confusion matrix and run time.

PCA dimensionality reduction and SMOTE balancing techniques are tested, with less than stellar overall results. Some exceptions provide major gains in run time or metrics for certain models. Balancing as a class weight parameter also proved unfruitful.

Run time limits practicality of using some models. Best result is **rfc RandomForestClassifier(), and the next step is to maximize parameter tuning on the selected model**

Default	Supervised				Inferential	Unsupervised	Neural Network	Ensemble Bagging		
Confusion Matrix	[[1906208 143] [1183 1252]]	[[1906292 59] [905 1530]]	[[1906326 25] [998 1437]]	[[1906306 45] [1482 953]]	[[1078399 827952] [0 2435]]	[[1485870 420481] [2433 2]]	[[1906248 103] [529 1906]]	[[1906325 26] [439 1996]]	[[1886432 19919] [19 2416]]	[[1905852 499] [1988 447]]
Model	lr	knn	svm	lsvm	gnb	kmeans	mlp	rfc	brfc	xgb
Accuracy score	0.9993	0.9995	0.9995	0.9992	0.5662	0.7784	0.9997	0.9998	0.9896	0.9987
F1 score	0.6538	0.7604	0.7375	0.5552	0.0058	0.0000	0.8578	0.8957	0.1951	0.2644
Precision score	0.8975	0.9629	0.9829	0.9549	0.0029	0.0000	0.9487	0.9871	0.1082	0.4725
Recall score	0.5142	0.6283	0.5901	0.3914	1.0000	0.0008	0.7828	0.8197	0.9922	0.1836
ROC-AUC score	0.7570	0.8142	0.7951		0.7828	0.3901	0.8913	0.9098	0.9909	0.5917
Wall time	Wall time: 51.9 s	Wall time: 5h 35min 2s	Wall time: 4h 26min 49s	Wall time: 14min 22s	Wall time: 14.5 s	Wall time: 55.6 s	Wall time: 15min 54s	Wall time: 7min	Wall time: 4min 12s	Wall time: 20min 57s

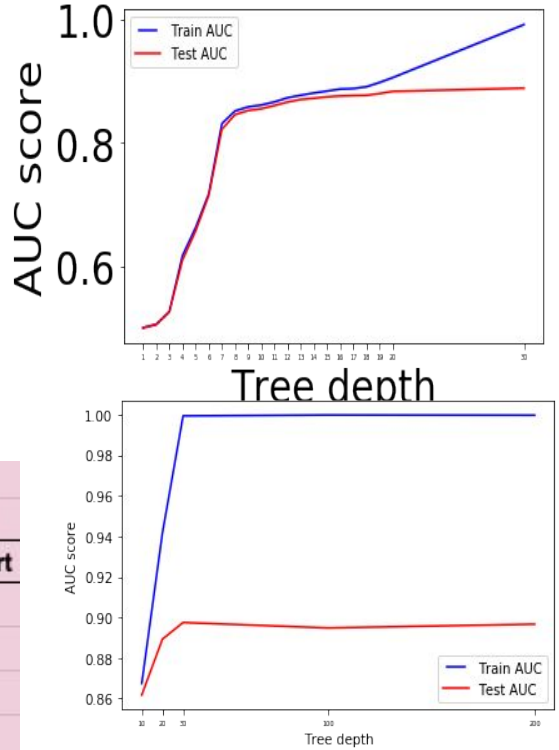
PCA	Supervised				Inferential	Unsupervised	Neural Network	Ensemble Bagging		
Confusion Matrix	[[1906208 143] [1183 1252]]	[[1906278 73] [891 1544]]	[[1906326 25] [998 1437]]	[[1906305 46] [1482 953]]	[[1872416 33935] [1396 1039]]	[[420476 1485875] [2 2433]]	[[1906271 80] [633 1802]]	[[1906300 51] [751 1684]]	[[1843533 62818] [80 2355]]	[[1906278 73] [2275 160]]
Model	lr	knn	svm	lsvm	gnb	kmeans	mlp	rfc	brfc	xgb
Accuracy score	0.9993	0.9995	0.9995	0.9992	0.9815	0.2216	0.9996	0.9996	0.9670	0.9988
F1 score	0.6538	0.7621	0.7375	0.5550	0.0555	0.0033	0.8348	0.8077	0.0697	0.1199
Precision score	0.8975	0.9549	0.9829	0.9540	0.0297	0.0016	0.9575	0.9706	0.0361	0.6867
Recall score	0.5142	0.6341	0.5901	0.3914	0.4267	0.9992	0.7400	0.6916	0.9671	0.0657
ROC-AUC score	0.7570	0.8170	0.7951		0.7044	0.6099	0.8700	0.8458	0.9671	0.5328
Wall time	Wall time: 59.3 s	Wall time: 3min 14s	Wall time: 5h 11min 3s	Wall time: 13min 59s	Wall time: 17.4 s	Wall time: 1min 5s	Wall time: 8min 6s	Wall time: 7min 22s	Wall time: 4min 30s	Wall time: 32min 6s

SMOTE	Supervised		*not SMOTE, rbf		Inferential	Unsupervised	Neural Network	Ensemble Bagging		
Confusion Matrix	[[1823974 82377] [128 2307]]	[[1902026 4325] [456 1979]]	[[21860 1884491] [8 2427]]	[[1820376 85975] [153 2282]]	[[1081677 824674] [0 2435]]	[[1906342 9] [2180 255]]	[[1901094 5257] [52 2383]]	[[1905515 836] [144 2291]]	[[1905338 1013] [97 2338]]	[[1881366 24985] [44 2391]]
Model	lr	knn	svm(linear, max1k)	lsvm	gnb	kmeans	mlp	rfc	brfc	xgb
Accuracy score	0.95678	0.99750	0.01272	0.95488	0.56796	0.99885	0.99722	0.99949	0.99942	0.98689
F1 score	0.05296	0.45291	0.00257	0.05032	0.00587	0.18896	0.47305	0.82380	0.80816	0.16041
Precision score	0.02724	0.31393	0.00129	0.02586	0.00294	0.96591	0.31191	0.73265	0.69770	0.08734
Recall score	0.94743	0.81273	0.99671	0.93717	1.00000	0.10472	0.97864	0.94086	0.96016	0.98193
ROC-AUC score	0.95211	0.90523			0.78370	0.55236	0.98794	0.97021	0.97982	0.98441
Wall time	Wall time: 2min 13s	Wall time: 18h 34min 20s	Wall time: 3min 21s	Wall time: 33min 38s	Wall time: 39.6 s	Wall time: 2min 10s	Wall time: 3h 43min 30s	Wall time: 5min 42s	Wall time: 1h 1min 53s	Wall time: 41min 48s

Model Validation, Tuning, Final Result

- The incremental gains from adding new feature sets to the data, and tuning the model parameters with Gridsearch Cross Validation are shown in the table, recording an improvement of .05 in the Recall Score, or 125 fewer False Negatives in a test sample of about 1.9M.
- The max_depth AUC score is plotted on a limited and expanded scale, to visualize the train vs test score convergence patterns. Although the measured score differential between 20 and 30 is small, the sharp increase in slope to a plateau suggests we can measure the tradeoff for more train accuracy at the risk of overfitting in this range.
- In addition to max_depth, the other changes from the default RandomForestClassifier parameters are criterion='entropy' over default 'gini', and n_esitmators=100 over the default 10 which already throws a future warning message about the switch.

Confusion Matrix	[[1906300 51]	[[1906288 63]	[[1906322 29]	[[1906325 26]	-25 FP diff
	[546 1889]]	[487 1948]]	[445 1990]]	[421 2014]]	-125 FN diff
Model	start features	plus Change balance	plus Time conversion	rfc best_params	Diff best - start
Accuracy score	0.9997	0.9997	0.9998	0.9998	0.0001
F1 score	0.8635	0.8763	0.8936	0.9001	0.0366
Precision score	0.9737	0.9687	0.9856	0.9873	0.0135
Recall score	0.7758	0.8000	0.8172	0.8271	0.0513



Future Work

- ➔ Parameter tuning was applied to LR and KNN models with minor gains in metrics. It made sense to review key parameters for overfit/underfit by comparing train vs test results over a range of values. The same could be applied to the remainder of the models, some with a great deal of complexity. In particular, tuning a Neural Network such as MLP and variations of XG Boost have worked out for some Kagglers.
- ➔ Train, Test, Split test size was held constant, and default parameters were used for SMOTE and PCA. Parameter tuning could be applied to each to ensure optimal selection.
- ➔ Alternate criteria and ensemble combinations could be interesting to pursue. For example, it might be plausible to focus on minimizing False Negatives by prioritizing Recall at the expense of Precision, or combine models.