Table of Contents

Risk and Fraud Individual Assignment

Author: Marang Mutloatse

Date: 26 May 2020

Summary of Activities

This notebook is an overview of strategies, subsequent results and analyses for Risk and Fraud game to generate a model with a KS2 at least 0.4219 and optimization of the GINI coefficient (+- 0.53).

My intial strategy was to follow a classical Machine learning Best practice process:

- Data Description
- Impute missing
- Deal with skewness
- Deal with scaling
- Deal with outliers
- Choose evaluation metric and develop cross validation strategy
- Baseline
- Feature Engineering
- Hyperparameter Tuning
- Feature Importance & Subsequent Selection

But During Phase 1, listening to the Professors comments and researching about nature of fraud modelling, one has to develop a better methodology.

So my fraud modelling strategy is as follows:

- Data Description
- Impute Missing data
- Baseline -> Comparing classification models first using accuracy then comparing the GINI and KS using the functions provided by you.
- Hyperparameter Optimization of chosen model.
- Apply Population Stability Index for Feature selection and Determine Feature importance
- Apply Scaling techniques and deal with skewness
- Ensemble modelling with a combination **manual** hyperparameter optimization
- Feature Engineering
- Advanced methods - Neural Networks

Note: Cross-validation metric is both accuracy and roc_auc.

Logbook & Analysis

Phase 1: First Submission, Baseline Model and Hyperparameter Optimization

1. Logistic Regression with the four given features:
2. Logistic Regression with all but the numeric features (all categorical variables):
3. Random Forest Classifier with hyperparameter tuning whereby best parameters are (class_weight='balanced_subsample',min_samples_split=2,n_estimators=500,max_depth=15,max_features='log2'):
4. ExtraTreeClassifier, BaggingClassifier & XGBoostClassifier:
5. DecisionTreeClassifier+RandomizedSearchCV:

Iteration	KS2	GINI
1	0.125484061775	0.118134178689
2	0.234742718283	0.277442234421
3	0.314123714253	0.367214251189
4	0.304586970801	0.323794921814
5	0.307829785858	0.321114335622

Observations

- The best model was the RandomForestClassifier, even though I grid searched it to find the best params, it was still heavily overfitting. But this is now the baseline model which be used to improve the KS2 and GINI.

Phase 2: Population Stability Index, Feature Importance and Missing value imputation

1. RandomForestClassifier where features with PSI=0 were removed.
2. RandomForestClassifier+PSI_removed_features and further removal of features below chosen threshold of 0.002
3. RandomForestClassifier: KNNImputer = (3,4,5) in Out-of-Time Sample
4. RandomForestClassifer: SimpleImputer('strategy'='most_frequent' & 'median' & mean)

Iteration	KS2	GINI
6	0.361105463082	0.507598268006
7	0.365516317156	0.494392277193
8	0.36887933345	0.511397979649
9	0.374885627409	0.509637334911

Observations

- Population stability index (PSI) is a metric to measure how much a variable has shifted in distribution between two samples or over time. By removing for values. In my case, I removed for features PSI = 0. A better way could va been to group them into insignificant < 0.1, 0 - 0.25 as some minor change and 0.25 < as some major changes.
- First KNNImputer improved the score from the fillna(0) but SimpleImputer(strategy 'mean') was the best.

So the new model was a RandomForestClassifer with removed features(PSI and RF feature importance) with the simple imputer.

Phase 3: Feature Scaling

1. Random Forest Classifier + MinMaxScaler on numerical variables:
2. Random Forest Classifier + RobustScaler on numerical variables:
3. Random Forest Classifier + MinMaxAbsScaler on numerical variables:
4. Random Forest Classifier + StandardScaler on numerical variables:
5. Random Forest Classifier + fix skewness (for numerical columns where skewness > 0.75 and kurtosis < 3) using boxcox:
6. Random Forest Classifier + Skewed + StandardScaler on numerical variables:

Iteration	KS2	GINI
10	0.348310751486	0.465875377776
11	0.372493045222	0.479741032727
12	0.377971376815	0.499309143338
13	0.396093078494	0.521812817124
14	0.343835201804	0.4828556641
15	0.38508211721	0.49274253921

Observations

- Best perfomring scaler was StandardScaler. StandardScaler assumes your data is normally distributed within each feature and will scale them such that the distribution is now centred around 0, with a standard deviation of 1. MinMaxScaler is sensitve to outliers so that is why it performed so poorly.

In the next step, my Random Forest + StandardScaler will be used.

Phase 4: Voting and Stacking Ensemble models

1. Logistic Regression + RandomForestClassifier: VotingClassifier
2. Logistic Regression + RandomForestClassifier: GridSearchCV - VotingClassifier
3. Logistic Regression + RandomForestClassifier: StackingClassifier
4. Increased weighting RandomForest to 18. + Logistic Regression + RandomForestClassifier: VotingClassifier

Iteration	KS2	GINI
16	0.401829037237	0.518718980767
17	0.400318909692	0.495938132281
18	0.390337480014	0.505056654867
19	0.405289096942	0.531975381515

Observations

- The soft voting classifier dramatically improved both the KS2 and the Gini-coefficient as it took the both of best worlds. The stacking classifier performed worse then the various voting classifier because regardless of the base estiimator (I switched between RF and LR), it loses information. So weighting each boosted sample is more effective. As a result, in the next phase I used the parameters from Iteration 19 with engineered features.

Phase 5: Feature Engineering

1. One-hot encoding nominal categories of icn_var_22 and icn_var_24:
2. Binary encoding to high cardinal ordinal variables ico_var_33, ico_var_34, ico_var_35, ico_var_36:
3. Quantile Binning - Ordinal and K-means:
4. Genetic Programming:
5. Polynomial Features(interaction_only='True',degree=3):

Iteration	KS2	GINI
20	0.341745302637	0.462277839906
21	0.329513211767	0.440242053993
22	0.372473405484	0.498438063198
23	0.38040092792	0.495737021599
24	0.343835201804	0.4828556641

Observations

- none these methods worked effectively in improving the score. I thought Genetic Programming would work but I needed to geenrate an automate way of optimizing the population. So none of these iterations bore any great fruit, I needed a method which could identify interactions of the features, get the relationships and generalize on OOT. So I implemented an Neural Network.

Phase 6: Final modelling: Ensembling to improve feature interactions

1. Logistic Regression + RandomForestClassifier + MLPClassifier + OneVsRestClassifier: VotingClassifier:
2. Logistic Regression + RandomForestClassifier + MLPClassifier (base estimators were MLP and RF) StackingClassifier:
3. Logistic Regression + RandomForestClassifier + MLPClassifier + KNN: Weighted Soft Voting Classifier:

Iteration	KS2	GINI
25	0.420223384689	0.510998253219
26	0.38803154373	0.495806338321
27	0.421843085426	0.507927522436

Observations

- Multi-Layer Perceptron generates the relationships which i seek, decreases the GINI but incrreases the KS2 dramatically by 0.2. I optimized it and the tanh activation function provided the best score.
- By adding the KNN, we are able to increase the KS2 and decrease GINI slightly. This distance based method is simple but effective at classfication and compensates for the other models pitfalls.

Final Model

Final Model Hyperparameters for **Voting Classifier** with **soft voting** and **weights** = {19,1,1,1} are:

for all models **random_state**=0

Random Forest:

- **n_estimators:** 102
- **min_samples_split:** 4
- **n_jobs:** -1
- **max_depth:** 12
- **class_weight:** 'balanced_subsample'
- **max_features:** 'log2'

KNN:

- **n_neighbors:** 5
- **weights:** uniform
- **n_jobs:** -1

Logistic Regression:

- **C:** 12.0
- **solver:** 'lbfgs'
- **penalty:** 'l2'
- **class_weight:** None

Multi-Layer Perceptron Classifier:

- **hidden_layer_sizes:** 102
- **solver:** 'lbfgs'
- **learning_rate:** 'adaptive'
- **activation:** 'tanh'

Analysis of Model and Conclusions

Random Forest

A pruned Random Forest has great prediction power. This Random Forest can be stored to produce forecasts or new input data and offer information about proximities between pairs, which are important for clustering either under a supervised or an unsupervised setup. Moreover Its attractiveness is increased by its capability to handle missing data or unbalanced data and its flexibility to adapt nicely in sparsity. Random Forests increased efficiency in modeling outliers due to the random subspaces process and its ability to recognize non-linear relationships in the development sample was also a reason I weighted it the most in the voting classifier.

KNN

KNN simple, but effectively generates the realtionship due to distance of the features.

Logistic Regression

Logistic regression's robustness is effective but limited.

Multi-Layer Perceptron Classifier

MLP if well tuned (n_estimators=number/size of hidden layers),is able to identify the relationships of the variables/features and generalize better on an OOT.

Final Submission results

KS: 0.421843085426

Gini: 0.507927522436

Grade: 9.99600

id: 519

Future Work

- Explore different sampling techniques using the imblearn library. Because of the nature of our problem, primarily exploring the under_sampling methods incuding CondensedNearestNeighbours, InstanceThresholds, NearMiss, Tomeklinks and other methods might have provided for some improvement in the KS
- Explore the use of Deep Feature Synthesis to perform feature engineering on relational and temporal data. The algorithms' ability to stack primitive to generate more complex features where each time we increase the depth of a feature. Moreover, we can create a feature matrix for any entity in the dataset. When not having an idea about what each column means, this technique could prove useful
- Use the Weight of Evidence encoding within the logistic regression pipeline but not necessarily the whole pipeline. Having read some literature on these fraud problems, WOE significantly overfits and careful choice of the feature and greater