UBS wealth management branches selection based on Adaptive Synthetic (ADASYN) minority over-sampling approach and Extreme Gradient Boosting (XGBoost)

Chen Yi, Wei Wang, Annan Dong, Yuwei Liu

New Jersey Institute of Technology

November 16, 2018

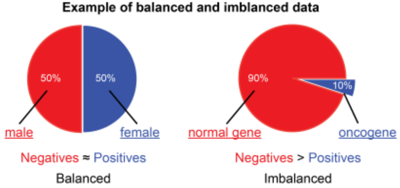
Problem definition

- Binary classification problem

  All places(zip codes) in U.S. can be divided into 2 classes: class "1" are with UBS wealth management branches and class "0" are without branches, which is determined by various features of this place, such as population, house prices, medium annual income, etc..
- Find places for next new branches

  Find 3 most likely places to be opened with new branches.
- Imbalanced data set

  The number of places in class "1" is much less than that in class "0".



Problem solutions

1. Tackle imbalanced data with Adaptive Synthetic (ADASYN) minority over-sampling approach
2. Features selection and model training by XGBoost
   - Relatively insensitive to the correlated features.
   - Features selection inherently.
   - Tackle non-linear classification problem well.
   - Relatively insensitive to the missing data.
   - Relatively insensitive to the imbalanced data problem.
   - Overfitting problem is relatively fewer.
   - Better predicting performance than other models.
3. Rank the predictions and find 3 most likely places to be opened with new branches.

Data preparation

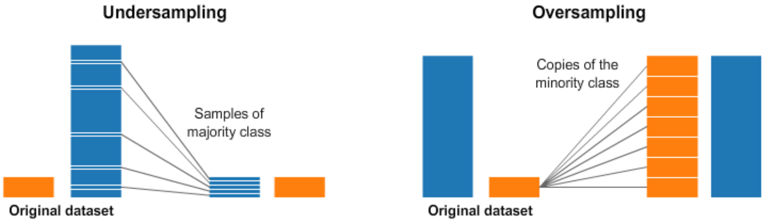
- Data cleaning
  Remove invalid and null data.
- Zip codes
  The minimum units/elements of data set are zip codes, our data contains 30398 valid zip codes.
- Possible features
  Population, house prices, medium family annual income, ect.
- Features selection
  Features importance and weights will be determined and applied by XGBoost automatically.
- Labels
  There are 259 valid zip codes which are already opened with UBS branches, where 10 zip codes contain more than one branch, no branches for the remaining zip codes. Some zip code contains more than one branch.

Data preparation

- Over-sampling minority data set
  Over sampling the class “1” in order to overcome imbalanced data set problem.
- Categorical data encoding
  Encode the categorical data with LabelEncoder and OneHotEncoder.
- Data splitting
  Split 80% of data set into training set and 20% into test set.
- Cross validation
  In the process of model training and hyper parameters tuning, 5-folder cross validation is employed to overcome over fitting.

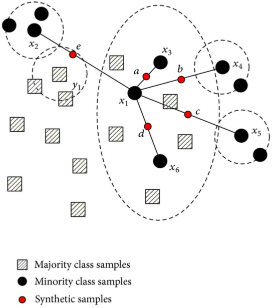
Tackle imbalanced data sets

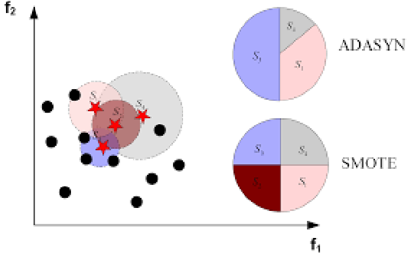
1. Under sampling
   Cause the useful data missing in majority data set
2. Over sampling
   - Synthetic Minority Over-sampling Technique (SMOTE)
   - Adaptive Synthetic (ADASYN) minority over-sampling approach



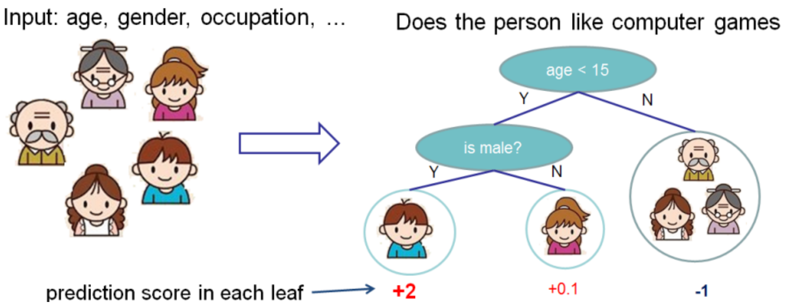
SMOTE vs. ADASYN

- SMOTE assigns the equal number of synthetic data samples to be generated for each class
- ADASYN uses a biased distribution to determine the number (not necessarily equal) of synthetic data samples for each class

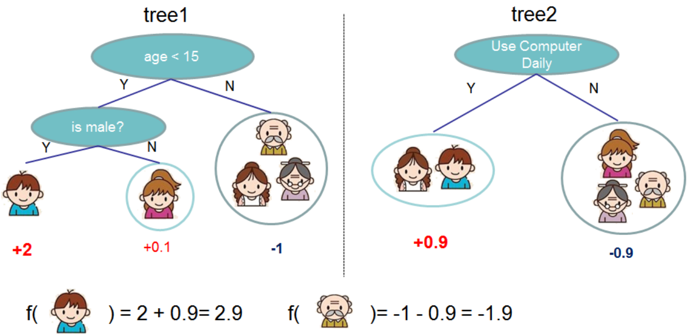




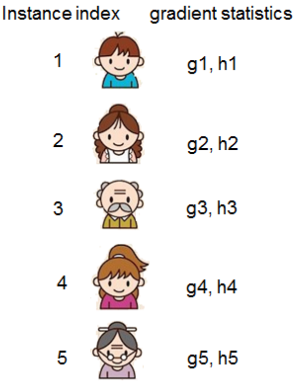
XGBoost

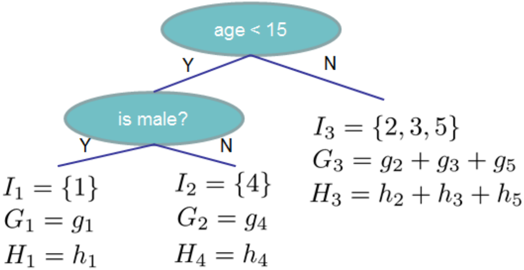

Input: age, gender, occupation, ...
Does the person like computer games
age < 15
Y
N
is male?
Y
N
prediction score in each leaf
+2
+0.1
-1

XGBoost


tree1
age < 15
Y
N
is male?
Y
N
+2
+0.1
-1
tree2
Use Computer Daily
Y
N
+0.9
-0.9
f( ) = 2 + 0.9= 2.9
f( )= -1 - 0.9 = -1.9

XGBoost



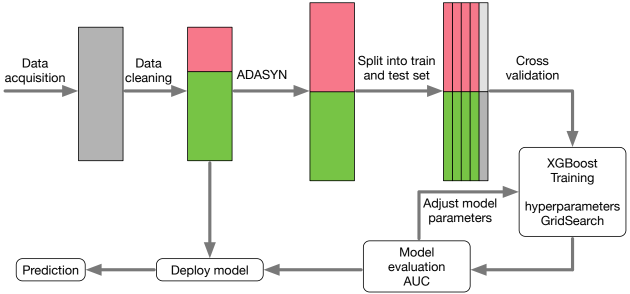


$$Obj = -\sum_j \frac{G_j^2}{H_j + \lambda} + 3\gamma$$

The smaller the score is, the better the structure is

System model and XGBoost hyper parameters tuning

System model



Hyper parameters tuning based on "GridSearchCV"

Search the best hyperparameters via cross validation based on ROC_AUC scoring.

Tune "max_depth" and "min_child_weight"

```
param_test1= {
    'max_depth': range(3,15,2),
    'min_child_weight': range(1,11,2)
}
gsearch1= GridSearchCV(estimator = XGBClassifier( learning_rate =0.1,
                                    n_estimators=1200, max_depth=3,
                                    min_child_weight=1, gamma=0.3,
                                    subsample=0.8, colsample_bytree=0.8,
                                    objective= 'binary:logistic', nthread=4,
                                    scale_pos_weight=1,seed=27),

                                    param_grid = param_test1,
                                    scoring='roc_auc', verbose=10,
                                    n_jobs=1, iid=False, cv=5)

gsearch1.fit(X_train,y_train)
gsearch1.best_params_, gsearch1.best_score_
```

max_depth=9, min_child_weight=1

Hyper parameters tuning (contd.)

Tune "gamma"

```
param_test2= {
    'gamma':[i/10.0 for i in range(0,5)]
}
gsearch2 = GridSearchCV(estimator = XGBClassifier( learning_rate =0.1,
                                    n_estimators=1200, max_depth=9,
                                    min_child_weight=1, gamma=0.3,
                                    subsample=0.8, colsample_bytree=0.8,
                                    objective= 'binary:logistic', nthread=4,
                                    scale_pos_weight=1,seed=27),

                                    param_grid = param_test2,
                                    scoring='roc_auc', verbose=10,
                                    n_jobs=1, iid=False, cv=5)

gsearch2.fit(X_train,y_train)
gsearch2.best_params_, gsearch2.best_score_
```

gamma=0.1

Hyper parameters tuning (contd.)

Tune “subsample” and “colsample_bytree”

```
param_test3= {
   'subsample': [i/10.0 for i in range(2,10,2)],
   'colsample_bytree': [i/10.0 for i in range(2,10,2)
}
gsearch3= GridSearchCV(estimator = XGBClassifier( learning_rate =0.1,
                                 n_estimators=1200, max_depth=9,
                                 min_child_weight=1, gamma=0.1,
                                 subsample=0.8, colsample_bytree=0.8,
                                 objective= 'binary:logistic', nthread=4,
                                 scale_pos_weight=1,seed=27),

                                 param_grid = param_test3,
                                 scoring='roc_auc', verbose=10,
                                 n_jobs=1, iid=False, cv=5)

gsearch3.fit(X_train,y_train)
gsearch3.best_params_, gsearch3.best_score_
```

subsample=0.8, colsample_bytree=0.4

Hyper parameters tuning (contd.)

Reducing Learning Rate in order to reduce overfitting

```
param_test4= {
   'learning_rate ':[i/1000.0 for i in range(5,20,2)]
}
gsearch4 = GridSearchCV(estimator = XGBClassifier( learning_rate =0.1,
                                        n_estimators=1200, max_depth=9,
                                        min_child_weight=1, gamma=0.1,
                                        subsample=0.8, colsample_bytree=0.4,
                                        objective= 'binary:logistic', nthread=4,
                                        scale_pos_weight=1,seed=27),

                                        param_grid = param_test4,
                                        scoring='roc_auc', verbose=10,
                                        n_jobs=1, iid=False, cv=5)

gsearch4.fit(X_train,y_train)
gsearch4.best_params_, gsearch4.best_score_
```

learning_rate=0.019

Model evaluation

AUC score for train and test sets.

```
xgb = XGBClassifier(learning_rate =0.019,
   n_estimators=1200, max_depth=9,
  min_child_weight=1, gamma=0.1,
  subsample=0.8, colsample_bytree=0.4,
   objective= 'binary:logistic', nthread=4,
  scale_pos_weight=1, seed=27
  )

xgb.fit(X_extended_train, y_extended_train)

# Performance of the train set
auc_train = roc_auc_score(y_extended_train, xgb.predict(X_extended_train))
0.9999791805463025
# Performance of the test set
auc_test = roc_auc_score(y_extended_test, xgb.predict(X_extended_test))
0.9952537671799967
```

Model evaluation

Confusion matrix.

$$ConfusionMatrix = \begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix} = \begin{bmatrix} 30089 & 50 \\ 7 & 252 \end{bmatrix}, \tag{1}$$

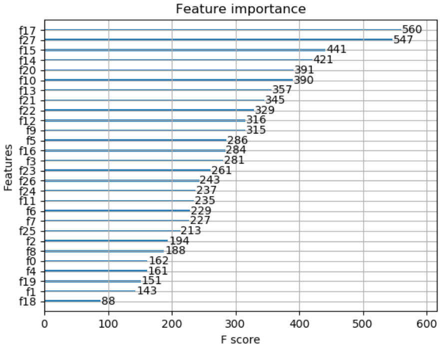
$$Recall = \frac{TP}{TP + FN} = 0.972972972972973, \tag{2}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = 0.998124876636621, \tag{3}$$

$$Precision = \frac{TP}{TP + FP} = 0.834437086092715, \tag{4}$$

$$Specificity = \frac{TN}{TN + FP} = 0.998341019940940 \tag{5}$$

Features importance



7 most important features: Core Based Statistical Areas population, health insurance market rating area ID, business first quarter payroll, number of employees, delivery business, average house value, number of businesses.

Predicting probability weighting

$$w_0 = \frac{259}{30398}, w_1 = \frac{10}{30398} \tag{6}$$

where w_1 or w_0 are a priori probability that the zip code is already opened or not opened with branches

$$\mathbf{P} = \mathbf{P}_1 \cdot w_1 + \mathbf{P}_0 \cdot w_0, \tag{7}$$

where $\mathbf{P}_1$ or $\mathbf{P}_0$ are predicted probability when zip code is in class “1” or class “0”, respectively.

Conclusion

Rank **P** of all zip codes, we can find 3 most likely zip codes that will be opened with new UBS wealth management branches: "01803" in Massachusetts, "45069" in Ohio, "54303" in Wisconsin.

Zip code	01803	45069	54303
Core Based Statistical Areas population	4552402	2130151	306241
Health insurance market rating area ID	4	4	16
Business first quarter payroll	917618	335246	157424
Number of employees	40005	32343	16929
Delivery business	1456	1903	1027
Average house value	439400	202700	112800
Number of businesses	1474	1602	846