Churn Prediction

May 2023

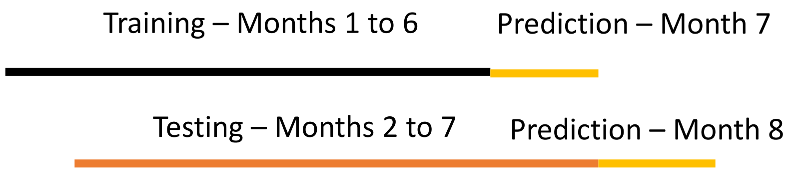
Background – A Credit Union in Central Florida

- Founded in 1937
- >160,000 customers
- 22 branches in Central Florida
- Asset Size: $2 billion
- Annual churn rate: 8%
- Retention Team
- Existing churn prediction model: built 3 years ago

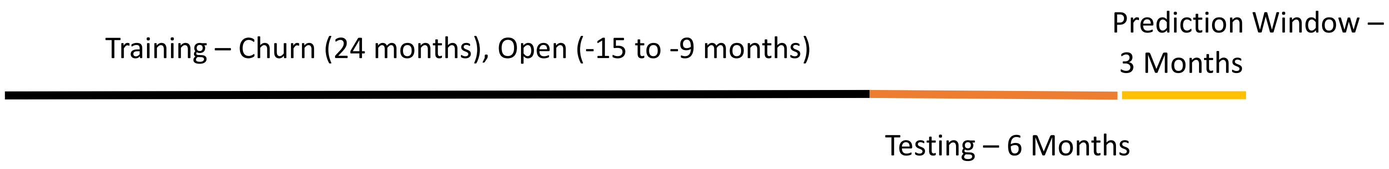
Comparison

	Existing Model	New Model
Level	Account Level Prediction	Customer Level Prediction
Manpower	Involve 4 BI Team members update the data every month	Automate the entire process (SSIS, SQL, Python, Power BI, DevOps)
Overlapping Data	Yes	No
Data Source	Savings Transactions only	Add credit card, loan transactions
Prediction Window	1 month	3 months
Modeling Approaches	Random Forest	Multiple approaches – Random Forest
Sampling Strategy	Oversampling	Multiple approaches - SMOTE
Performance	<10 % of churn cases predicted	72% of churn cases predicted

Existing Model



New Model



Data Mining Process

Data Extraction

- 40 SQL Queries
- 10 Train Churn
- 10 Train Open
- 10 Test Mixed
- 10 For Prediction

Data Preparation

- Data Consolidation
  - Train Set
  - Test Set
- Features Engineering
  - Aggregates (Mean, SD)
  - Trend Factor
  - Zip Code (long. & lat.)
- Feature Selection
  - Correlation
- Normalization
  - 0 to 1
- Sample Balancing
  - Oversampling
  - SMOTE
  - ADYSYN

Modeling

- Logistic Regression
- Random Forest
- Gradient Boosting
- Light GBM

Evaluation Metrics

- F1 Score
- Recall / Sensitivity
- Precision
- Specificity

Deployment

- New Data
- Continual Monitoring
- Retraining Model

SQL Queries

- TrainChurnAccountDetail.sql
- TrainChurnBasic.sql
- TrainChurnDirectDeposit.sql
- TrainChurnExternalLoan.sql
- TrainChurnExtLoanTrack.sql
- TrainChurnList.sql
- TrainChurnLoanDetail.sql
- TrainChurnLoanTran.sql
- TrainChurnSavingsTran.sql
- TrainChurnTblTran.sql

'OpenSharesCount',
'OpenShareBalance',
'ShareChargeOffCount',
'ShareChargeOffAmt',
'OpenCertCount',
'OpenCertBalance',
'OpenLoanCount',
'LoanChargeOffCount',
'DirectDepositAmt',
'CreditLimit',
'PlasticsCount',
'DisputeCount',
'TimesOverLimit',
'DaysDelq',

'CashAdvanceBal',
'LastPaymentAmt',
'LastStatementBal',
'PastDueAmt',
'LoanChargeOffAmt',
'LoanBalance',
'LoanDelqDays',
'LoanPayment',
'LoanInterest',
'LoanLateChargeYTD',
'LoanCreditLimit',
'OnlineLoanPaymentAmt',
'PhoneLoanPaymentAmt',
'PhoneLoanPaymentCount'
'ACHLoanPaymentAmt',
'ACHLoanPaymentCount',
'CheckLoanPaymentCount',

'DraftLoanPaymentAmt',
'DraftLoanPaymentCount',
'CashLoanPaymentAmt',
'CashLoanPaymentCount',
'LoanFeeAmt',
'AllLoanPaymentAmt',
'AllLoanPaymentCount',
'ATMAmt',
'BillPaymentAmt',
'FeeAmt',
'FeeCount',
'DebitCardAmt',
'CheckCount',
'OnlineAmt',
'OnlineCount',
'CheckAmt',
'DividendAmt',
'DividendCount',
'BranchCount',
'EmbFeeAmt',
'EmbFeeCount',
'CCTranAmt'

Trend Factor

- Capture the trend for 6-month transaction
- Recent data carries more weight.
- No change = 1
- Declining trend < 0 - 1
- Increasing trend > 1 - 2

```
TotalWeight = 21  #1+2+3+4+5+6
NofMonths = 6  #Data in 6 month chunks
TrainTrend = pd.DataFrame()
length = int(len(TrainCombined)/6)
```

```
for feature in Features1:
   TF = []
   j = TrainCombined.columns.get_loc(feature)
   for i in range (length):
      a = TrainCombined.iloc[i*6, j]
      b = TrainCombined.iloc[i * 6 + 1, j]
      c = TrainCombined.iloc[i * 6 + 2, j]
      d = TrainCombined.iloc[i * 6 + 3, j]
      e = TrainCombined.iloc[i * 6 + 4, j]
      f = TrainCombined.iloc[i * 6 + 5, j]
      MOU = a+b+c+d+e+f
      if MOU == 0:
         tf = 1
      else:
         WeightMOU = a*1+b*2+c*3+d*4+e*5+f*6
         Numberator = 
(WeightMOU*NofMonths)/TotalWeight
         tf = Numberator/MOU
      TF.append(tf)
```

Balancing Sample – 3 Approaches

- Libraries – Imblearn (RandomOverSampler, SMOTE, ADYSYN)
- Use Logistic Regression to test
- Evaluation Criteria
  - Precision: TP/(TP + FP)
  - Recall: TP / (TP + FN)
  - F1: 2*(Precision*Recall)/(Precision + Recall)
  - Specificity: TN / (TN + FP)

	Precision	Recall	F1	Specificity
Original	0.09	0.16	0.12	0.99
Oversampling	0.02	0.85	0.03	0.42
SMOTE	0.03	0.55	0.05	0.77
Adasyn	0.03	0.55	0.05	0.76

Modeling

- Logistic Regression with / without Grid Search 5-fold Cross Validation
- Random Forest with / without Grid Search & 5-fold Cross Validation
- Gradient Boosting
- Light GBM
- Libraries: sklearn, lightgbm
- Limitation: Modeling on laptop

	Precision	Recall	F1	Specificity
Logistic Regression Without Grid Search	0.03	0.55	0.05	0.77
Logistic Regression With Grid Search	0.03	0.46	0.06	0.84
Random Forest without Grid Search	0.02	0.57	0.03	0.60
Random Forest with Grid Search	0.02	0.72	0.04	0.63
Gradient Boosting	0.01	0.98	0.02	0.11
Light GBM	0.01	0.09	0.02	0.92

Use pickle to save the best model

Improvement

- Categorize customers in terms of their value to the bank. Set different threshold for different categories.
- Use a server or virtual machine to do modeling.
- Try more models with grid search and cross validation.
- Use variable importance in random forest to remove unimportant variables, avoid overfitting.