Predicting Potential Churners Based on Personal and Behavioral Data

CIH Bank Case

Essalhi Douae, Riany Aya, Safiri Fatima Ezzahra

Al Akhawayn University

Bankathon, 1st edition

Ifrane, Morocco

Abstract

The goal of this project is to create a comprehensive predictive analytics model for identifying potential churners among CIH Bank's customer base. Churner customers, or those who are considering terminating their banking connection, provide a substantial issue to financial organizations. Understanding and anticipating churn drivers is critical for developing proactive retention tactics. Using a comprehensive dataset comprising customer transaction history, engagement patterns, and demographic information, the study focuses on identifying key variables that are useful in predicting and forecasting customer turnover. To assess and model these features, advanced machine learning methods such as logistic regression and decision trees will be used, delivering actionable insights into client behavior. The proactive management of customer connections adds value to X Bank. The bank can personalize retention measures to specific pain areas and concerns of at-risk customers by precisely identifying possible churners. This not only protects the current client base, but it also adds to increased customer pleasure and loyalty. In addition, the predictive analytics model is a useful decision-making tool for the bank. Based on the findings of the investigation, X Bank can improve resource allocation, marketing strategies, and product offerings. This preemptive approach allows the bank to properly manage resources, reducing the impact of customer attrition on revenue and profitability. Finally, this project not only tackles X Bank's severe issue of customer attrition, but it also lays the groundwork for employing predictive analytics in the financial sector. The findings add to the expanding landscape of customer relationship management by enabling banks to make data-driven decisions that support long-term customer loyalty and financial stability.

Methodology

1. *Features' Selection:*

As you may know our data is so large. Thus, taking into consideration the performance of the quality, we need to make our problem less complex and more optimal. That's why we thought of selecting the features that are more useful for us. We performed that through two steps: First selection of features based on their correlation with the target feature (churner). We selected all the features that have correlation bigger than 0.01 to keep more choices for us in the next step. We narrowed down our features from 410 to 153. Then we selected the best features by using random forest algorithm and selected the best 100 features from the 153 features we got in the previous step. Also, we noticed that when we maximize the number of estimators and minimize the maximum depth the performance of our model increased by making less complex and more solid since it does more trials to look for the best features. Then we did an intersection between the correlated features and the best features to select our best correlated features and work with them.

2. *Sampling the data:*

Now, that we selected the features we would like to work on. We trained our model using random forest to predict whether our client is churn or not and we found a very high accuracy but very insignificant confusion matrix resulting in a 0 in the f1 score which takes into consideration both the recall and the precision. This made us conclude that the accuracy is not the only measure for our model performance but, the precision and the recall are also important which push us to think about the f1 score.

In order to solve the issue, we had to do sampling of our data since our data was so imbalanced; it contained more data about none churn clients than churn clients which made our model focus more and give more priority to predict none churners than predicting churn

clients. We searched on the different algorithms to tackle the issue to select the one that =’s going to fit best our problem. First we did an under sampling by narrowing down our majority data. This made the model work faster, do very good job with the training data, but very bad in the testing data. In other words, we faced the over fitting phenomenon. It actually did badly with all the algorithms used to predict our data.

That’s why we thought of changing the sampling method and we used the SMOTE method. It took us ages to run the prediction because it actually oversample our data which didn’t make a lot of sense since our data is already large and previous steps were done aiming to narrow down our data.

Next, we tried the ADASYN method which resulted in the best performance four model using the decision tree classifier resulting in high accuracy and high f1 score in both the training and the testing data. While both ADASYN and SMOTE aim to address class imbalance through oversampling, ADASYN's adaptive nature makes it more suitable for datasets with varying densities and challenges posed by noisy regions, especially in high-dimensional spaces.

3. *Choice of Algorithm:*

We used 6 different algorithms with our method to be able to select the best one: "LogisticRegression", "DecisionTreeClassifier", "RandomForestClassifier", "GradientBoostingClassifier", "AdaBoostClassifier", and "KNeighborsClassifier".

This is an overview for each algorithm:

1. **Logistic Regression:**

    - *Type:* Supervised Learning (Classification)

- *Description:* Logistic Regression is a linear model for binary classification that predicts the probability of an instance belonging to a particular class. It uses the logistic function to model the probability and makes predictions based on a decision boundary.

2. **Decision Tree Classifier:**
    - *Type:* Supervised Learning (Classification)
    - *Description:* Decision Tree is a tree-like model where each node represents a decision based on a feature, leading to a final decision or prediction at the leaf nodes. It's intuitive, interpretable, and can handle both categorical and numerical data.
3. **Random Forest Classifier:**
    - *Type:* Ensemble Learning (Bagging)
    - *Description:* Random Forest is an ensemble of decision trees. It builds multiple trees and merges their predictions to improve accuracy and reduce overfitting. Each tree is trained on a random subset of the data and features, introducing diversity.
4. **Gradient Boosting Classifier:**
    - *Type:* Ensemble Learning (Boosting)
    - *Description:* Gradient Boosting builds an ensemble of weak learners (usually decision trees) sequentially. Each tree corrects the errors of the previous ones, optimizing a loss function. It's powerful and often yields high accuracy.
5. **AdaBoost Classifier:**

- *Type:* Ensemble Learning (Boosting)
- *Description:* AdaBoost (Adaptive Boosting) is a boosting algorithm that combines weak learners to create a strong classifier. It assigns different weights to instances, emphasizing the misclassified ones in each iteration, and adjusts the model accordingly.

6. **KNeighbors Classifier:**

    - *Type:* Supervised Learning (Classification)
    - *Description:* K-Nearest Neighbors is a simple, instance-based learning algorithm. It classifies instances based on the majority class among their k-nearest neighbors in the feature space. It's non-parametric and lazy, meaning it doesn't build a separate training model but relies on the entire dataset during prediction.

We started from the basic one to more complex ones to see which one fits best our data. Both Decision Tree classifier and random forest classifier gave very promising results. We chose random forest classifier since it gave higher f1 than decision tree classifier.

The output of our model:

Logistic Regression:

```
LogisticRegression
Model performance for Training set
- accuracy score of train: 0.6282
- f1_score: 0.7287
- roc_auc_score : 0.6288
- confusion_matrix :
[[ 35676 102741]
 [    17 137976]]
- classification report :
              precision    recall  f1-score   support

           0       1.00      0.26      0.41    138417
           1       0.57      1.00      0.73    137993

    accuracy                           0.63    276410
   macro avg       0.79      0.63      0.57    276410
weighted avg       0.79      0.63      0.57    276410

---------------------------------------------


Model performance for Testing set
- accuracy score of train: 0.2668
- f1_score: 0.0321
- roc_auc_score : 0.6287
- confusion_matrix :
[[ 44605 128410]
 [     1   2129]]
- classification report :
              precision    recall  f1-score   support

           0       1.00      0.26      0.41    173015
           1       0.02      1.00      0.03      2130

    accuracy                           0.27    175145
   macro avg       0.51      0.63      0.22    175145
weighted avg       0.99      0.27      0.41    175145
```

Decision Tree Classifier

```
DecisionTreeClassifier
Model performance for Training set
- accuracy score of train: 1.0000
- f1_score: 1.0000
- roc_auc_score : 1.0000
- confusion_matrix :
[[138417      0]
 [     0 137993]]
- classification report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    138417
           1       1.00      1.00      1.00    137993

    accuracy                           1.00    276410
   macro avg       1.00      1.00      1.00    276410
weighted avg       1.00      1.00      1.00    276410

-----------------------------------------------

-----------------------------------------------
Model performance for Testing set
- accuracy score of train: 0.9937
- f1_score: 0.7566
- roc_auc_score : 0.9029
- confusion_matrix :
[[172310    705]
 [   405   1725]]
- classification report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    173015
           1       0.71      0.81      0.76      2130

    accuracy                           0.99    175145
   macro avg       0.85      0.90      0.88    175145
weighted avg       0.99      0.99      0.99    175145
```

Random Forest Classifier:

```
RandomForestClassifier
Model performance for Training set
- accuracy score of train: 1.0000
- f1_score: 1.0000
- roc_auc_score : 1.0000
- confusion_matrix :
[[138417      0]
 [     1 137992]]
- classification report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    138417
           1       1.00      1.00      1.00    137993

    accuracy                           1.00    276410
   macro avg       1.00      1.00      1.00    276410
weighted avg       1.00      1.00      1.00    276410

-----------------------------------------------
-----------------------------------------------
Model performance for Testing set
- accuracy score of train: 0.9975
- f1_score: 0.8869
- roc_auc_score : 0.8986
- confusion_matrix :
[[173014      1]
 [   432   1698]]
- classification report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    173015
           1       1.00      0.80      0.89      2130

    accuracy                           1.00    175145
   macro avg       1.00      0.90      0.94    175145
weighted avg       1.00      1.00      1.00    175145
```

Gradient Boosting Classifier

```
GradientBoostingClassifier
Model performance for Training set
- accuracy score of train: 0.9872
- f1_score: 0.9870
- roc_auc_score : 0.9871
- confusion_matrix :
[[138119    298]
 [  3253 134740]]
- classification report :
              precision    recall  f1-score   support

           0       0.98      1.00      0.99    138417
           1       1.00      0.98      0.99    137993

    accuracy                           0.99    276410
   macro avg       0.99      0.99      0.99    276410
weighted avg       0.99      0.99      0.99    276410


Model performance for Testing set
- accuracy score of train: 0.9858
- f1_score: 0.0166
- roc_auc_score : 0.5038
- confusion_matrix :
[[172630    385]
 [  2109     21]]
- classification report :
              precision    recall  f1-score   support

           0       0.99      1.00      0.99    173015
           1       0.05      0.01      0.02      2130

    accuracy                           0.99    175145
   macro avg       0.52      0.50      0.50    175145
weighted avg       0.98      0.99      0.98    175145
```

AdaBoost Classifier

```
AdaBoostClassifier
Model performance for Training set
- accuracy score of train: 0.9800
- f1_score: 0.9798
- roc_auc_score : 0.9800
- confusion_matrix :
[[136990   1427]
 [  4101 133892]]
- classification report :
              precision    recall  f1-score   support

           0       0.97      0.99      0.98    138417
           1       0.99      0.97      0.98    137993

    accuracy                           0.98    276410
   macro avg       0.98      0.98      0.98    276410
weighted avg       0.98      0.98      0.98    276410


Model performance for Testing set
- accuracy score of train: 0.9783
- f1_score: 0.0519
- roc_auc_score : 0.5193
- confusion_matrix :
[[171245   1770]
 [  2026    104]]
- classification report :
              precision    recall  f1-score   support

           0       0.99      0.99      0.99    173015
           1       0.06      0.05      0.05      2130

    accuracy                           0.98    175145
   macro avg       0.52      0.52      0.52    175145
weighted avg       0.98      0.98      0.98    175145
```

Conclusion:

We decided that studying the correlation and selecting the best correlated features then sampling the data using the ADASYN to balance our data then train our model on random forest classifier leads to the best results with an accuracy and f1 score of 1 (perfect score) on the training data. And an accuracy of 0.9975 and f1 of 0.6689 on the testing data.