TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỀ TÀI
MÔN HỌC MACHINE LEARNING AT SCALE

Tên đề tài

BANK CUSTOMER CHURN PREDICTION

GVHD: ThS. Quách Đình Hoàng

SVTH: Nhóm 13

Lê Phương Nam	19133036
Võ Thành Đạt	19133019
Nguyễn Thị Nhả Thư	19133054
Nguyễn Phạm Duy Khiêm	19133027

Thành phố Hồ Chí Minh, ngày 17 tháng 12 năm 2022

Mục lục

1. Tóm tắt

Dự đoán sự rời đi của khách hàng giúp cho doanh nghiệp đưa ra các chiến lược, giải pháp hợp lý cho ngân hàng hạn chế sự rời đi của khách hàng, đáp ứng nhu cầu sử dụng của khách hàng. Nhóm sẽ sử dụng Pyspark để xây dựng 3 model Machine Learning đó là: Logistic Regression, Random Forest, Gradient Boosting Tree và đưa ra mô hình phù hợp nhất cho doanh nghiệp có thể áp dụng để có thể hạn chế sự rời đi của khách hàng trong ngân hàng của mình. Nhóm sử dụng 2 phương pháp để đánh giá model, phương pháp thứ nhất là sử dụng độ đo accuracy, phương pháp thứ hai là sử dụng cross-validation chọn siêu tham số cho mô hình. Kết quả đạt được sau khi thử nghiệm trên hai phương pháp thì mô hình thuật toán Random Forest cho kết quả tốt nhất cho bài toán.

2. Giới thiệu

Sự rời đi của khách hàng (còn gọi là sự tiêu hao khách hàng) xảy ra khi khách hàng ngưng sử dụng sản phẩm hoặc dịch vụ của công ty. Khách hàng rời đi sẽ ảnh hưởng đến lợi nhuận của công ty. Các nhà phân tích cho rằng việc có được khách hàng mới sẽ tốn kém gấp 4 – 5 lần so với việc giữ chân khách hàng hiện tại. Do đó việc phân tích sự rời đi của khách hàng giúp cho doanh nghiệp có thể xác định các vấn đề trong dịch vụ của mình, đưa ra các giải pháp dẫn đến sự hài lòng của khách hàng và giúp giữ chân khách hàng cao hơn.

Input của bài toán là tập hợp các thông tin của khách hàng như: mã khách hàng, điểm tín dụng, quốc gia, giới tính, tuổi, thời gian vay, số dư, số sản phẩm (dịch vụ) sử dụng, thẻ tín dụng, thành viên thường xuyên giao dịch, mức lương ước tính. Nhóm chúng em sử dụng các thuật toán Logistic Regression, Random Forest, Gradient Boosting Tree để dự đoán sự rời đi của khách hàng.

3. Dữ liệu

Tập dữ liệu của nhóm em được lấy từ Kaggle với định dạng file là csv gồm có 10000 dòng và 12 cột bao gồm các thuộc tính:

CustomerID: mã khách hàng

Credit Score: điểm tín dụng Country: quốc gia

Gender: giới tính

Age: tuổi

Tenure: thời gian vay

Balance: số dư

Products Numbers: số sản phẩm (dịch vụ) sử dụng

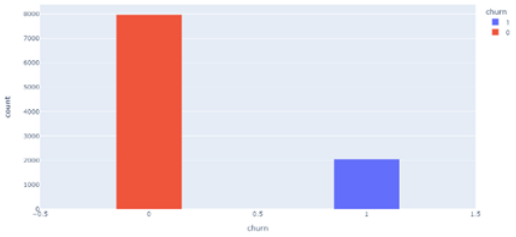
Credit Card: thẻ tín dụng

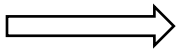
Active Member: thành viên thường xuyên giao dịch

Estimated Salary: mức lương ước tính

Churn: sự rời đi của khách hàng

Qua phần phân tích EDA, nhóm nhận thấy rằng biến Churn bị mất cân bằng giữ giá trị 0 (khách hàng không rời đi) và 1 (khách hàng rời đi), do đó trong phần tiền xử lý dữ liệu nhóm em có sử dụng phương pháp oversampling để cân bằng lại giá trị của biến Churn trong tập dữ liệu.





```
+-----+-----+
|churn|count|
+-----+-----+
|    0| 7963|
|    1| 6111|
+-----+-----+
```

4. Phương pháp

Nhóm thực hiện đánh giá mô hình theo 3 mô hình sau đây:

Logistic Regression

Random Forest

Gradient Boosting Tree

4.1. Logistic Regression

Mô hình hồi qui Logistic là sự tiếp nối ý tưởng của hồi qui tuyến tính vào các bài toán phân loại. Từ đầu ra của hàm tuyến tính chúng ta đưa vào hàm Sigmoid để tìm ra phân phối xác suất của dữ liệu. Lưu ý rằng hàm Sigmoid chỉ được sử dụng trong bài toán phân loại nhị phân. Đối với bài toán phân loại nhiều hơn hai nhãn, hàm Softmax là một dạng hàm tổng quát của Sigmoid sẽ được sử dụng. Hàm Sigmoid thực chất là một hàm biến đổi phi tuyến dựa trên công thức:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Tính chất:

– Là một hàm số liên tục và nhận giá trị trong khoảng (0;1).

– Chính vì là hàm liên tục nên hàm sigmoid sẽ có đạo hàm tại mọi điểm.

Tất cả các dạng hồi quy thực tế có cùng chung một phương trình tổng quát để giúp chỉ ra rằng biến mục tiêu y(biến phụ thuộc) sẽ thay đổi như nào dựa vào mối quan hệ của nó với tất cả các biến độc lập x mà không quan tâm đến việc mối quan hệ đó là phi tuyến hay tuyến tính:

$$y = w_n x_n + w_{n-1} x_{n-1} + \cdots + w_1 x_1 + w_0 + \epsilon$$

Trong đó:

w_0 là giá trị ước lượng của y khi tất cả các xi đều đạt giá trị 0.

ϵ là sai số thể hiện các yếu tố chưa thể nghiên cứu đến nhưng các yếu tố này vẫn ảnh hưởng tới y.

w_i là tham số ước lượng(hay ta còn gọi là trọng số), có thể hiểu nôm na là mỗi w_i sẽ chịu trách nhiệm quản lí một x_i và w_i sẽ điều chỉnh để có giá trị y mong muốn, lưu ý là giá trị tham số w_i có thể thay đổi còn giá trị x_i là cố định.

Khi thiết lập mô hình hồi quy ta quan tâm đến quan hệ giữa nhiều biến độc lập x_i với biến mục tiêu(biến phụ thuộc), phương trình tổng quát để ước lượng biến mục tiêu y:

$$E(y) = w_n x_n + w_{n-1} x_{n-1} + \cdots + w_1 x_1 + w_0 = \mathbf{w}^T \mathbf{x} + w_0$$

Lưu ý là giá trị ϵ đã được loại bỏ. E(y) trong phương trình logistic regression là xác suất để kết luận giá trị của biến y không phải giá trị thực của biến y:

$$E(y) = P(y = 0 || 1 | x_1, x_2, \ldots, x_n)$$

Áp dụng hàm sigmoid để chuyển giá trị w^Tx+w_0 thành xác suất để kết luận giá trị của biến y từ đó để xác định được nhãn của input x đây chính là mô hình của logistic regression:

$$P = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + w_0)}}$$

Hàm mất mát trên toàn bộ tập dữ liệu, chúng ta gọi đây là hàm Cross-Entropy:

$$\mathcal{L}(\mathbf{w}) = -\sum_{i=1}^{m}(y^{(i)} \log a^{(i)} + (1-y^{(i)}) \log (1-a^{(i)}))$$

Trong đó:

– $y^{(i)}$: giá trị đúng của input $x^{(i)}$, $y^{(i)}$ nhận giá trị 0 hoặc 1.

– $a^{(i)}$: giá trị mô hình dự đoán ứng với input $x^{(i)}$.

– m: là số điểm dữ liệu.

4.2. Random Forest Classifier

Mô hình random forest được huấn luyện dựa trên sự phối hợp giữa luật kết hợp (ensembling) và quá trình lấy mẫu tái lặp (bootstrapping). Cụ thể thuật toán này tạo ra nhiều cây quyết định mà mỗi cây quyết định được huấn luyện dựa trên nhiều mẫu con khác nhau và kết quả dự báo là bầu cử (voting) từ toàn bộ những cây quyết định.

Giả định dữ liệu huấn luyện mô hình random forest là một tập $D=\{(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)\}$ bao gồm N quan sát. Thuật toán random forest sẽ sử dụng phương pháp lấy mẫu tái lập để tạo thành B tập dữ liệu con Tức là chúng ta sẽ thực hiện M lượt nhặt các mẫu từ tổng thể và bỏ vào túi để tạo thành tập $B_i=\{(x_1^{(i)},y_1^{(i)}),(x_2^{(i)},y_2^{(i)}),\ldots,(x_M^{(i)},y_M^{(i)})\}$. Với mỗi tập dữ liệu B_i chúng ta xây dựng một mô hình cây quyết định và trả về kết quả dự báo là $\hat{y}_j^{(i)}=f_i(x_j)$. Trong đó $\hat{y}_j^{(i)}$ là dự báo của quan sát thứ j từ mô hình thứ (i), x_j là giá trị vectơ đầu vào, $f_i(.)$ là hàm dự báo của mô hình thứ i

- Đối với mô hình dự báo: Chúng ta tính giá trị trung bình của các dự báo từ mô hình con.

$$\hat{y}_j = \frac{1}{B}\sum_{i=1}^{B}\hat{y}_j^{(i)}$$

- Đối với mô hình phân loại: Chúng ta thực hiện bầu cử từ các mô hình con để chọn ra nhãn dự báo có tần suất lớn nhất.

$$\hat{y}_j = \arg\max_c \sum_{i=1}^{B} p(\hat{y}_j{}^{(i)} = c)$$

Như vậy phương sai của mô hình trong trường hợp đối với bài toán dự báo:

$$\begin{aligned}\sigma_{\hat{y}}^2 &= \mathrm{Var}(\frac{1}{B}\sum_{i=1}^{B}\hat{y}^{(i)}) \\ &= \frac{1}{B^2}[\sum_{i=1}^{B}\mathrm{Var}(\hat{y}^{(i)}) + 2\sum_{1\le m<n\le B}\mathrm{cov}(y^{(m)}, y^{(n)})]\end{aligned}$$

Do kết quả của mô hình con A không chịu ảnh hưởng hoặc phụ thuộc vào mô hình con B nên ta có thể giả định kết quả dự báo từ các mô hình là hoàn toàn độc lập nhau. Tức là ta có $\mathrm{cov}(y^{(m)},y^{(n)})=0$, $\forall\ 1\le m<n\le B$. Đồng thời giả định chất lượng các mô hình là đồng đều, được thể hiện qua phương sai dự báo là đồng nhất $\mathrm{Var}(\hat{y}^{(i)})=\sigma^2$, $\forall_{i=1},B$—. Từ đó suy ra:

$$\begin{aligned}\sigma_{\hat{y}}^2 &= \frac{1}{B^2}[\sum_{i=1}^{B}\mathrm{Var}(\hat{y}^{(i)})] \\ &= \frac{1}{B^2}B\sigma^2 = \frac{1}{B}\sigma^2\end{aligned}$$

4.3 Gradient Boosting Tree

Gradient Boosting là một kỹ thuật học máy được sử dụng trong các nhiệm vụ hồi quy và phân loại , trong số những nhiệm vụ khác. Nó đưa ra một mô hình dự đoán dưới dạng một tập hợp các mô hình dự đoán yếu, thường là các cây quyết định

Giả định $f^{(x)}$ là hàm dự báo từ phương pháp tăng cường được áp dụng trên một tác vụ dự báo với ma trận đầu vào X và biến mục tiêu là vector y. Tại mô hình thứ b trong chuỗi mô hình dự báo, kí hiệu là f^b, ta tìm cách khớp một giá trị phần dư r_i từ cây quyết định tiền nhiệm f^{b-1}. Các bước trong quá trình huấn luyện mô hình theo phương pháp tăng cường được tóm tắt như sau:

1. Ban đầu ta thiết lập hàm dự báo $f^{(x)}=0$ và số dư $r_0=y$ cho toàn bộ quan sát trong tập huấn luyện.

2. Lặp lại quá trình huấn luyện cây quyết định theo chuỗi tương ứng với

b=1,2,…,B. Với một lượt huấn luyện gồm các bước con sau đây:

a. Khớp một cây quyết định f^b có độ sâu là d trên tập huấn luyện (X,r_b).

b. Cập nhật $\hat{f}$ bằng cách cộng thêm vào giá trị dự báo của một cây quyết đinh, giá trị này được nhân với hệ số co λ:

$$\hat{f}(\mathbf{x}) = \hat{f}(\mathbf{x}) + \lambda \hat{f}^b(\mathbf{x})$$

c. Cập nhật phần dư cho mô hình:

$$\mathbf{r}_{b+1} := \mathbf{r}_b - \lambda \hat{f}^b(\mathbf{x})$$

Thuật toán sẽ dừng cập nhật khi số lượng cây quyết định đạt ngưỡng tối đa B hoặc toàn bộ các quan sát trên tập huấn luyện được dự báo đúng.

3.- Kết quả dự báo từ chuỗi mô hình sẽ là kết hợp của các mô hình con:

$$\hat{f}(\mathbf{x}) = \sum_{b=1}^{B} \lambda \hat{f}^b(\mathbf{x})$$

5. Thực nghiệm kết quả

Nhóm chúng em sẽ đánh giá các mô hình được xây dựng dưới đây bằng độ đo accuracy theo công thức như sau:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Trong đó:

True Positive (TP): Số các dự đoán rời đi đúng.

True Negative (TN): Số các dự đoán ở lại đúng.

False Positive (FP): Số các dự đoán rời đi sai.

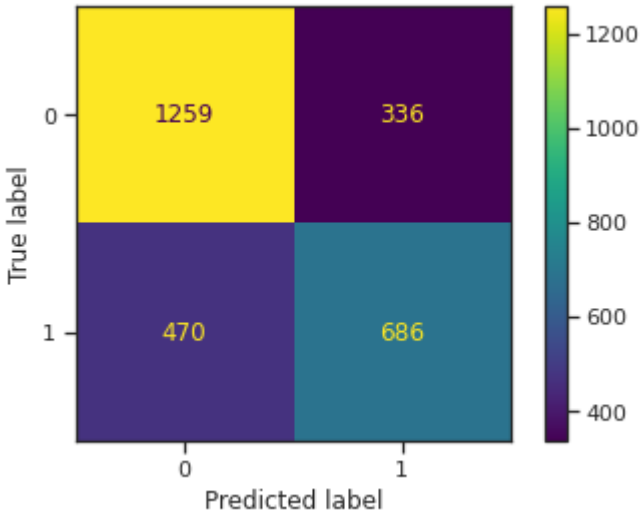
False Negative (FN): Số các dự đoán ở lại sai.

5.1. Logistic Regression

Đầu tiên ta sẽ xây dựng mô hình Logistic Regression với các tham số mặc định là RegParam=0, elasticNetParam=0 để huấn luyện mô hình trên tập train. Sau đó dự đoán trên tập test thì ta được độ đo accuracy = 0.70447.

Ta sẽ sử dụng phương pháp cross-validation để chọn các tham số tốt nhất cho mô hình. Số phần được chia để sử dụng cho phương pháp này là 3 phần, danh sách các tham số được dùng để thực hiện trong phương pháp này gồm: regParam = [0.001,

0.01, 0.1, 1.0], elasticNetParam = [0.5, 0.75, 1]. Sau khi thực hiện cross-validation ta được mô hình tốt nhất với regParam = 0.001, elasticNetParam = 0.75. Ta sẽ dùng mô hình này để dự đoán trên tập test và thu được confuse-matrix như sau:



Hình 1: Confuse-matrix của mô hình Logistic Regression (regParam = 0.001, elasticNetParam = 0.75)

Từ confuse-matrix trên ta có thể tính được accuracy = 0.70702.

5.2. Random Forest

Đầu tiên ta xây dựng mô hình Random Forest với tham số mặc định (numTrees=5, maxDepth=4) để huấn luyện mô hình trên tập train. Sau đó dự đoán trên tập test được độ đo accuracy : 0.77390

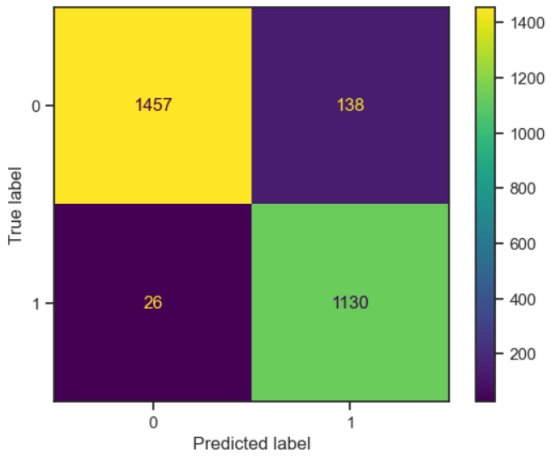
Sau đó, ta chia tập train thành 8 phần train và 2 phần validate. Tạo vòng lặp để train từng tham số mà ta nghĩ nó có khả năng là siêu tham số (num_list = [50, 100, 150, 200], depth_list = [10,20]) rồi sau đó ta dự đoán lại trên tập validate. Kết quả sau khi chạy vòng lặp như sau:

```
50, 10: 0.8246235606731621
50, 20: 0.91674047829938
100, 10: 0.8259521700620017
100, 20: 0.9154118689105403
150, 10: 0.8224092116917626
150, 20: 0.9131975199291408
200, 10: 0.8255093002657219
200, 20: 0.9154118689105403
```

Hình 2: Kết quả tìm siêu tham số của Random Forest

Từ kết quả của vòng lặp thì nhóm đã chọn được siêu tham số là numTrees=50, maxDepth=20

Sau đó ta xây dựng mô hình Random Forest với siêu tham số numTrees=50, maxDepth=20 để huấn luyện mô hình trên tập train. Ta sẽ dùng mô hình này để dự đoán trên tập test và thu được confusion-matrix như sau:



Hình 3:Confuse-matrix của mô hình Random Forest (numTrees=50, maxDepth=20).

Từ confusion matrix ta được độ đo accuracy là 0.94039

5.3 Gradient Boosting Tree

Đầu tiên ta xây dựng mô hình Gradient Boosting Tree với tham số mặc định maxIter: int = 20, maxDepth: int = 5 để huấn luyện mô hình trên tập train. Sau đó dự đoán trên tập test được độ đo accuracy : 0.79280

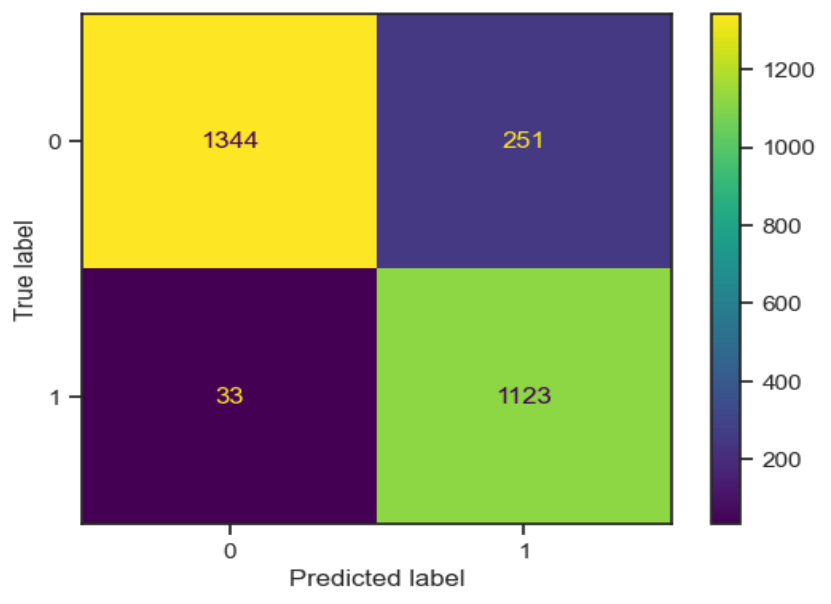
Sau đó ta tìm siêu tham số bằng cách chạy vòng lặp với mỗi tham số mà ta nghĩ nó có khả năng là siêu số (iter_list = [5,7,9,10], depth_list = [10,20]). Kết quả chạy vòng lặp như sau:

```
5, 10: 0.8259521700620017
5, 20: 0.8720106288751107
7, 10: 0.8294951284322409
7, 20: 0.8720106288751107
9, 10: 0.8396811337466785
9, 20: 0.8720106288751107
10, 10: 0.8445527015057573
10, 20: 0.8720106288751107
```

Hình 4: Kết quả chọn siêu tham số của Gradient Boosting Tree

Kết quả tìm được tham số tốt nhất là maxIter=10, maxDepth=20.

Sau đó ta xây dựng mô hình Gradient Boosting Tree với siêu tham số maxIter: int = 10, maxDepth: int = 20 để huấn luyện mô hình trên tập train. Ta sẽ dùng mô hình này để dự đoán trên tập test và thu được confusion-matrix như sau:



Hình 5: Confusion-matrix của mô hình Gradient Boosting Tree (maxIter=10, maxDepth=20)

Từ confusion matrix ta được độ đo accuracy là 0.90149

6. Kết luận

Sau khi áp dụng 3 thuật toán vào tập dữ liệu và dùng độ đo accuracy để đánh giá mức độ hiệu quả của thuật toán thì kết quả mà nhóm thu được kết quả như sau:

Dùng tham số mặc định để tính accuracy

Logistic Regression: accuracy = 0.70447 (RegParam=0, elasticNetParam=0)

Random Forest: accuracy = 0.77390 (numTrees=5, maxDepth=4)

Gradient Boosting Tree: accuracy = 0.79280 (maxIter: int = 20, maxDepth: int = 5)

Chọn tham số để tính accuracy

Logistic Regression: accuracy = 0.70702 (regParam=0.001, elasticNetParam=0.75)

Random Forest: accuracy = 0.94039 (numTrees=50,maxDepth=20)

Gradient Boosting Tree: accuracy = 0.90149 (maxIter: int = 10, maxDepth: int = 20)

Thuật toán Random Forest sẽ cho kết quả tốt nhất vì có giá trị accuracy lớn nhất trong 3 thuật toán mà nhóm sử dụng.

7. Phụ lục

8. Đóng góp

Thành viên	**Công việc**
Lê Phương Nam	Thuật toán Logistic Regression
Võ Thành Đạt	Thuật toán Gradient Boosting Tree
Nguyễn Phạm Duy Khiêm	Thuật toán Random Forest
Nguyễn Thị Nhả Thư	Phân tích EDA

9. Tham khảo

[1] Tạ Quốc Bảo, *Thư viện Matplotlib – Thư viện Python dùng để vẽ đồ thị,* 23/07/2019

https://code24h.com/thu-vien-matplotlib-8211-thu-vien-python-dung-de-ve-do-thi-d30561.htm

[2] W3School, *NumPy Tutorial*

https://www.w3schools.com/python/numpy/numpy_intro.asp

[3] Igor Radovanovic, *Sklearn – An Introduction Guide to Machine Learning*, 15/07/2022

https://algotrading101.com/learn/sklearn-guide/

[4] W3School, *Pandas Tutorial*

https://www.w3schools.com/python/pandas/default.asp

[5] GeeksForGeeks, *Python Seaborn Tutorial*, 22/11/2022

https://www.geeksforgeeks.org/python-seaborn-tutorial/

[6] *PySpark Tutorial*

https://www.tutorialspoint.com/pyspark/index.htm

[7] Deep AI KhanhBlog, *Hồi qui Logistic*

https://phamdinhkhanh.github.io/deepai-book/ch_ml/classification.html#ham-sigmoid.