BÁO CÁO LAB 1

KHAI THÁC TẬP PHỔ BIẾN VÀ LUẬT KẾT HỢP

Họ tên	Mã số sinh viên
Trần Hoàng Nam	19127481

Đánh giá độ hoàn thành : 100%

I. TIỀN XỬ LÝ DỮ LIỆU

Bộ dữ liệu ban đầu

	State	Account Length	Area Code	Phone	Int'l Plan	VMail Plan	VMail Message	Day Mins	Day Calls	Day Charge	...	Eve Calls	Eve Charge	Night Mins	Night Calls	Night Charge	Intl Mins	Intl Calls	Intl Charge	CustServ Calls	Churn?
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	99	16.78	244.7	91	11.01	10.0	3	2.70	1	False.
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	103	16.62	254.4	103	11.45	13.7	3	3.70	1	False.
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	110	10.30	162.6	104	7.32	12.2	5	3.29	0	False.

- Chúng ta sẽ tiến hành xem xét sơ qua bộ dữ liệu

```
#   Column           Non-Null Count  Dtype
---  ------           --------------  -----
0   State            3333 non-null   object
1   Account Length   3333 non-null   int64
2   Area Code        3333 non-null   int64
3   Phone            3333 non-null   object
4   Int'l Plan       3333 non-null   object
5   VMail Plan       3333 non-null   object
6   VMail Message    3333 non-null   int64
7   Day Mins         3333 non-null   float64
8   Day Calls        3333 non-null   int64
9   Day Charge       3333 non-null   float64
10  Eve Mins         3333 non-null   float64
11  Eve Calls        3333 non-null   int64
12  Eve Charge       3333 non-null   float64
13  Night Mins       3333 non-null   float64
14  Night Calls      3333 non-null   int64
15  Night Charge     3333 non-null   float64
16  Intl Mins        3333 non-null   float64
17  Intl Calls       3333 non-null   int64
18  Intl Charge      3333 non-null   float64
19  CustServ Calls   3333 non-null   int64
20  Churn?           3333 non-null   object
```

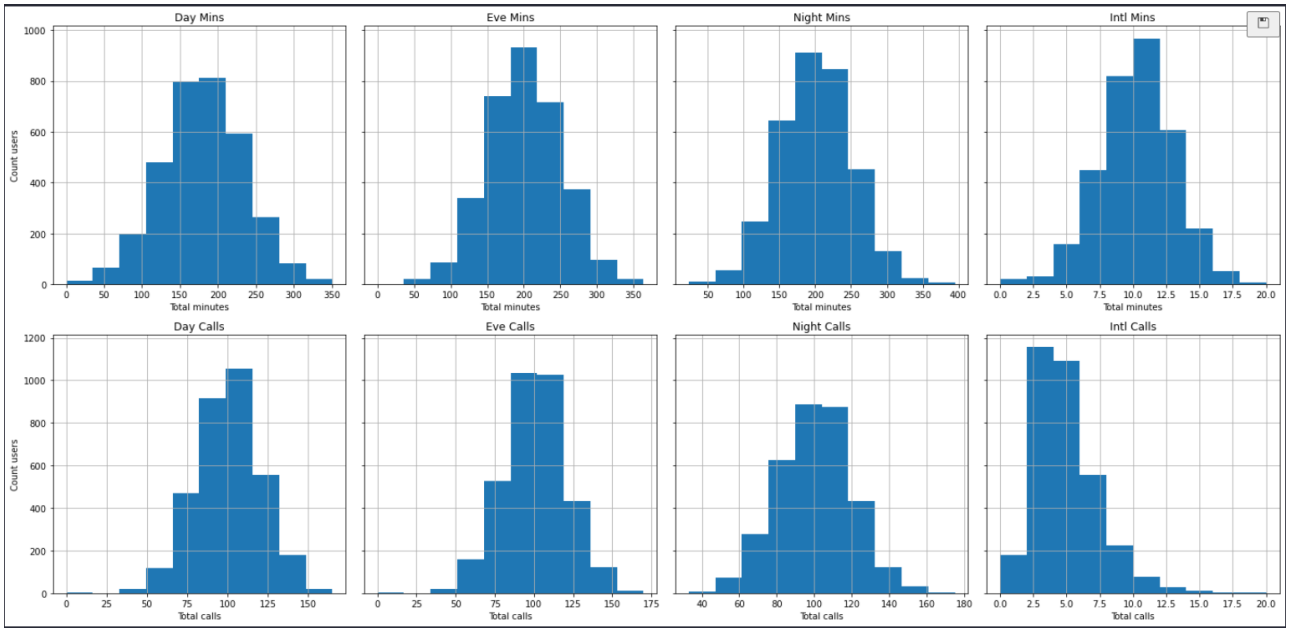
Nhận xét: Có thể thấy rằng không có dữ liệu thiếu, vì thế, chúng ta sẽ không xoá mẫu dữ liệu nào cả

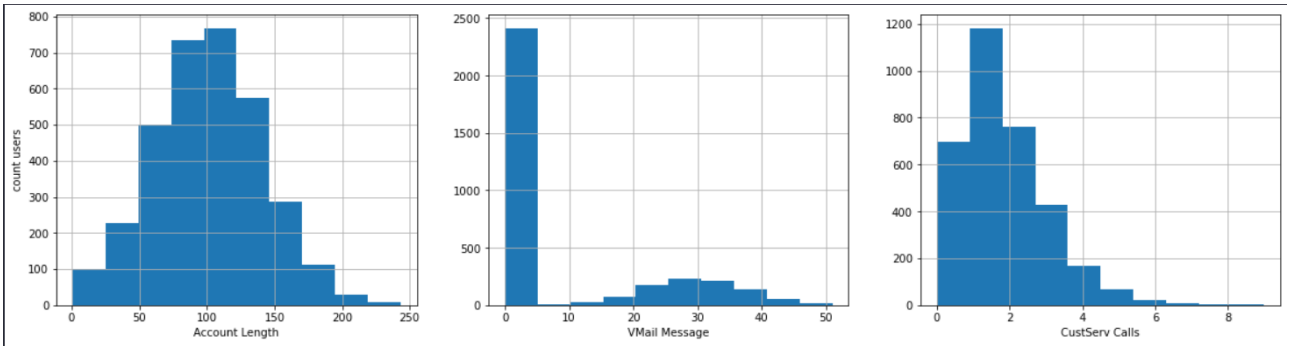
- Chúng ta sẽ phân loại các cột category và numeric:
  - Numeric gồm các cột:
    - Account Length
    - VMail Message: Số lượng VMail

        - Day Mins, Day Calls, Day charge: Tổng số min, call, charge buổi sáng
        - Eve Mins, Eve Calls, Eve charge: Tổng số min, call, charge buổi chiều
        - Night Mins, Calls, Night charge: Tổng số min, call, charge buổi tối
        - Intl Mins, Intl Calls, Intl charge: Tổng số min, call, charge quốc tế
        - CustSerCall: Số cuộc gọi hỗ trợ
    - Category gồm các cột:
        - State: tên viết tắt của bang
        - Area code: mã khu vực
        - Int'l plan: có đăng ký gói Int'l Plan hay không
        - VMail Plan: có đăng ký gói Vmail plan không
- Tiếp theo, chúng ta sẽ xem xét các giá trị của các cột thuộc dạng numeric có hợp lệ hay không

	Account Length	VMail Message	Day Mins	Day Calls	Day Charge	Eve Mins	Eve Calls	Eve Charge	Night Mins	Night Calls	Night Charge	Intl Mins	Intl Calls	Intl Charge
min	1	0	0.0	0	0.00	0.0	0	0.00	23.2	33	1.04	0.0	0	0.0
max	243	51	350.8	165	59.64	363.7	170	30.91	395.0	175	17.77	20.0	20	5.4

- Nhìn chung, các giá đều là các số dương, điều này thích hợp với tính chất của cột dữ liệu, tuy nhiên nếu chỉ để các cột này ở dạng số thì sẽ gây khó khăn trong việc khai thác dữ liệu, do đó chúng ta sẽ chuyển đổi (binning) các giá trị liên tục này về dạng category.
- Chúng ta có nhận xét rằng giá trị cột charge phụ thuộc vào tổng số mins gọi vào buổi đó, do đó, chúng ta sẽ không xét tới các cột charge





- Chúng ta nhận thấy rằng mỗi cột có khoảng giá trị phổ biến và không phổ biến, phần lớn đều có phân phối chuẩn, do đó, chúng ta sẽ phân ra làm 3 khoảng là **low, medium, high**.

- Tuỳ vào từng cột chúng ta sẽ có khoảng riêng cho chúng, phần này phụ thuộc nhiều vào kinh nghiệm và sẽ ảnh hưởng lớn đến kết quả thu được nên chúng ta cũng không thể đảm bảo rằng độ chính xác thu được có phù hợp với thực tế hay không, tạm thời chúng ta sẽ ước lượng sao cho mức medium là mức mà 80% khách hang có đặc điểm đó

	LOW	MEDIUM	HIGH
Account Length	Length < 50	50 ≤ Length ≤ 175	Length > 175
VMail Message	Messages < 0	0 ≤ Message ≤ 20	Messages > 20
Day Mins	Mins < 100	100 ≤ Mins ≤ 250	Mins > 250
Eve Mins	Mins < 150	150 ≤ Mins ≤ 250	Mins > 250
Night Mins	Mins < 150	150 ≤ Mins ≤ 250	Mins > 250
Intl Mins	Mins < 5	5 ≤ Mins ≤ 16	Mins > 16
Day Calls	Calls < 60	60 ≤ Calls ≤ 125	Calls > 125
Eve Calls	Calls < 60	60 ≤ Calls ≤ 125	Calls > 125
Night Calls	Calls < 60	60 ≤ Calls ≤ 125	Calls > 125
Intl Calls	Calls < 2	2 ≤ Calls ≤ 10	Calls > 10
CustSerCall	Calls < 0	0 ≤ Calls ≤ 3	Calls > 3

II. KHAI THÁC DỮ LIỆU

Câu hỏi:

- Câu 1: Liệu vị trí có ảnh hưởng đến số lượng cuộc gọi hay không?

- Câu 2: Các yếu tố ảnh hưởng tới việc người dung tiếp tục sử dụng dịch vụ
- Câu 3: Người gọi nhiều cuộc gọi thì có khả năng cao sẽ gọi hỗ trợ hơn

Chúng ta sẽ phân tích câu hỏi

Câu 1: Vị trí ảnh hưởng số lượng cuộc gọi => luật kết hợp State -> Calls có đáp ứng được độ tin cậy hay không

Câu 2: Yếu tố ảnh hưởng tới việc người dung sử dụng dịch vụ => Luật kết hợp (Unknow columns) -> Churn?

Câu 3: Người gọi nhiều cuộc gọi thì sẽ gọi hỗ trợ => Luật kết hợp Calls -> CustSerCall có đáp ứng độ tin cậy hay không

Để giải quyết 3 câu hỏi trên, chúng ta phải khai thác dữ liệu, ở đây, chúng ta sẽ dung thuật toán apriori để tìm mẫu phổ biến, từ mẫu phổ biến đó, chúng ta có thể dung để xác định được các luật kết hợp thì chúng ta sẽ trả lời được 3 câu hỏi trên

1. Khai thác tập phổ biến với min_supp = 0.8

- Bước 1: Chuẩn bị hàm cần thiết:
  - Ở bước này: chúng ta sẽ viết 1 hàm gọi là frequency dung để đếm số lần tập hợp xuất hiện của 1 tập hợp trong bảng dữ liệu.
- Bước 2: Khởi tạo tập phổ biến chỉ có 1 phần tử đáp ứng min_supp, ở bước này chúng ta sẽ viết hàm gọi là hàm init để khởi tạo tập phổ biến này.

```
def init(df):
    F1 = []
    n = len(df)
    for col in cols:
        status = list(np.unique(df[col]))
        for x in status:
            list_item = []
            if frequency(df,[[col,x]])/n>=min_supp:
                list_item = [[col,x]]
                F1.append(list_item)
    return sorted(F1)
```

- Bước 3: Tiến hành phát sinh tập phổ biến có 2 phần tử dựa trên tập phổ biến chỉ có 1 phần tử bằng cách bắt cặp các phần tử trong tập chỉ có 1 phần tử sao cho, tất cả các phần tử trong 2 tập này bằng nhau từ vị trí 0 đến vị trí size – 1, phần tử cuối cùng phải khác nhau, chúng ta cũng phải kiểm tra xem tất cả các tập con có kích thước size – 1 có tồn tại trong tập có kích

thước size – 1 không và thoả mãn min_supp, nếu thoả cả 2 đều trên thì them vào danh sách tập phổ biến, ngược lại thì bỏ qua

- Bước 4: Tiếp tục phát sinh tập có kích thước size + 1 dựa trên tập vừa mới khai thác được, thuật toán chỉ dung khi kích thước tập thu được <= 1

```
def generate_popular(df,F,size):
    n = len(F[size-1])
    if n <= 1:
        return
    F_size = []
    for i in range(n-1):
        for j in range(i+1,n):
            if(F[size-1][i][0:-1]==F[size-1][j][0:-1] and F[size-1][i][-1][0]!=F[size-1][j][-1][0]):
                candidate = F[size-1][i][:]
                candidate.append(F[size-1][j][-1])
                k = frequency(df,candidate)/len(df)
                if(combine_pre_check(F,candidate)==False):              # Tỉa nhánh
                    continue
                if k>=min_supp:                                         # Lọc bằng min_supp
                    F_size.append(candidate)

    F.append(F_size)
    generate_popular(df,F,size+1)
```

2. Khai thác luật kết hợp

   Bước 1: Tiến hành duyệt tập phổ biến, với mỗi transaction phổ biến, chúng ta có tiến hành generate ra tập luật có hệ quả là 1 item trong transaction thoả độ tin cậy

```
def ap_gen_rules(f_k,H,count_premise):
    if(len(f_k)>1 and len(H)!=0):
        premises, consequences = candidate_gen(f_k,H)
        for premise, consequence in zip(premises,consequences):
            if count_premise/frequency(df,consequence)<min_confidence:
                premises.remove(premise)
                consequences.remove(consequence)
            else:
                rules.append([premise,consequence])

        ap_gen_rules(f_k,consequences,count_premise)
```

✓ 0.8s

```
def candidate_gen(f_k,h):
    premise = []
    consequences = []
    origin_h = h[:]
    origin_f_k=f_k[:]
    for i in range(len(h)):
        for j in range(i+1,len(h)):
            if h[i][:-1]==h[j][:-1]:
                temp_consequences = h[i][:]
                temp_consequences.append(h[j][-1])
                last_premise = f_k[
                for consequence in temp_consequences:
                    last_premise.remove(consequence)
                if len(last_premise)>0:
                    premise.append(last_premise)
                    consequences.append(temp_consequences)
    return premise,consequences
```

(variable) temp_consequences: Any

✓ 0.6s

Bước 2: Từ tập hệ quả 1 item, chúng ta sẽ tiến hành generate ra tập hệ quả 2 phần tử từ tập hệ quả có 1 phần tử rồi tiến hành xem xét xem tập luật kết hợp hiện tại có thoả độ tin cậy không

- nếu có thì them vào kết quả và tiếp tục gọi đệ quy để tìm tập có kích thước hệ quả lớn hơn 1 item
- Nếu không thoả độ tin cậy thì bỏ qua

```
def genRules(F):
    for i in range(2,len(F)):
        for f_k in F[i]:
            origin_f_k=f_k[:]
            f_k_count = frequency(df,f_k)
            H = []
            for i in range(len(f_k)):
                h=[]
                h.append(f_k[i])
                f_k.remove(f_k[i])
                f_pre = frequency(df,h)
                if f_k_count/f_pre>=min_confidence:
                    rules.append([f_k,h])
                    H.append(h)
                f_k=origin_f_k[:]
            ap_gen_rules(f_k,H,f_k_count)
```

✓ 0.7s

III. KẾT QUẢ

Kết quả của tập phổ biến được lưu vào file popular_set.txt

Kết quả của tập luật kết hợp được lưu vào file rules.txt

Trả lời câu hỏi:

Câu 1:

```
for rule in rules:
    for item in rule[0]:
        if('State' in item):
            print(f'{rule[0]}: {rule[1]}')
            break
```

✓ 0.1s

Khi chạy code để lọc ra những rule có State nằm trong tập luật kết hợp, chúng ta không thu được gì cả, vậy nên, tạm thời chúng ta kết luận với độ trợ và độ tin cậy 80%, vị trí địa lý không ảnh hưởng đến số lượng cuộc gọi

Câu 2:

```
for rule in rules:
    for item in rule[1]:
        if('Churn?' in item):
            print(f'{rule[0]}   ->   {rule[1]}')
            break
```

✓ 0.8s

```
[['CustServ Calls', 'medium']]  ->  [['Churn?', 'False.']]
[['Intl Mins', 'medium']]  ->  [['Churn?', 'False.']]
```

Khi chạy code để lọc kết quả, chúng ta thu được như trên, có thể thấy rằng những người có số lượng cuộc gọi hỗ trợ ở mức medium hoặc thời gian gọi ở mức bình thường thì đa số không có ý định ngừng sử dụng dịch vu

Câu 3:

```
for rule in rules:
    for item in rule[1]:
        if('Day Calls' in item or 'Eve Calls' in item or 'Night Calls' in item):
            print(f'{rule[0]}   ->   {rule[1]}')
            break
```

✓ 0.1s

```
[['CustServ Calls', 'medium']]  ->  [['Day Calls', 'medium']]
[['CustServ Calls', 'medium']]  ->  [['Eve Calls', 'medium']]
[['CustServ Calls', 'medium']]  ->  [['Night Calls', 'medium']]
[['Intl Calls', 'medium']]  ->  [['Day Calls', 'medium']]
[['Intl Mins', 'medium']]  ->  [['Day Calls', 'medium']]
[['Intl Calls', 'medium']]  ->  [['Eve Calls', 'medium']]
[['Intl Mins', 'medium']]  ->  [['Eve Calls', 'medium']]
[["Int'l Plan", 'no']]  ->  [['Night Calls', 'medium']]
[['Intl Calls', 'medium']]  ->  [['Night Calls', 'medium']]
[['Intl Mins', 'medium']]  ->  [['Night Calls', 'medium']]
```

Khi chạy code để lọc ra kết quả, có vẻ như những người gọi hỗ trợ ở mức bình thường thì có số lượng cuộc gọi ở mức bình thường, nhưng ngược lại thì chưa chắc.

IV. THAM KHẢO

- Slide bài giảng