

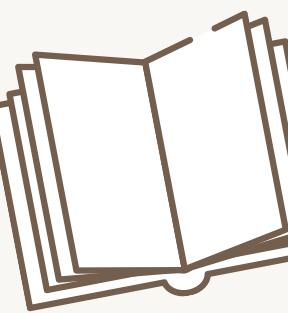
Nội dung

1. Giới thiệu đề tài

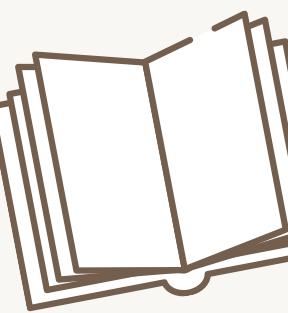
2. Mô tả dữ liệu, quy trình tiền xử lý

3. Trích chọn đặc trưng

4. Xây dựng và đánh giá mô hình



1. Giới thiệu đề tài



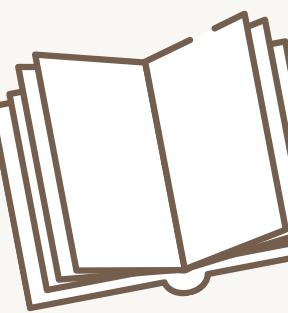
1. Giới thiệu đề tài

Trong Data Mining có nhiều mô hình phân lớp để phục vụ cho việc khai phá dữ liệu, tuy nhiên mỗi mô hình đều có những ưu nhược điểm, phù hợp với những bài toán khác nhau.

- Sử dụng bộ dữ liệu Bank Marketing (liên quan đến chiến dịch tiếp thị trực tiếp của một tổ chức Ngân hàng ở Bồ Đào Nha - chiến dịch tiếp thị dựa trên cuộc gọi điện thoại liên lạc nhiều lần với khách hàng để mời họ gửi tiền vào ngân hàng) để minh họa cho việc xây dựng, so sánh, đánh giá các mô hình phân lớp.
- Mô hình phân lớp: Logistic Regression, KNN, Naive Bayes, Random Forest, Neural Network, SVM
- Tools sử dụng: Python.
- Tiến trình thực hiện: Tìm hiểu dữ liệu → Tiền xử lý → Giải quyết mất cân bằng dữ liệu → RFE (trích chọn đặc trưng) → Xây mô hình → Đánh giá mô hình



2. Mô tả dữ liệu và Tiền xử lý



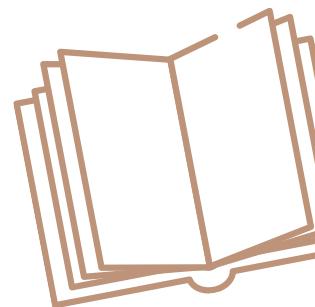
2.1 Dữ liệu trước tiên xử lý (41188 dòng x 21 cột)

Tên cột	Giá trị	Ý nghĩa	month	Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec	Tháng liên lạc gần nhất trong năm
age	17- 98	Tuổi	day_of_week	Mon, Tue, Wed, Thu, Fri	Thứ liên lạc trong tuần
job	Admin, Blue-collar, Entrepreneur, Housemaid, Management, Self-employed, Services, Student, Technician, Retired, Unemployed, Unknown	Công việc	duration	0-4918	Khoảng thời gian liên hệ với khách hàng gần đây nhất (khoảng thời gian gọi đt cho KH, tính bằng giây)
marital	Married, Divorced, Single	Tình trạng hôn nhân	campaign	1-56	Số lần liên lạc với khách hàng trong suốt chiến dịch
education	basic.4y,basic.6y ,basic.9y High school, Illiterate, Professional.course, Tốt nghiệp đại học University.degree, Unknown.	Trình độ học vấn	pdays	1-999	Số ngày kể từ khi khách hàng được liên lạc lần cuối từ chiến dịch trước đó
Default	Yes, No, Unknown	Tình trạng vỡ nợ	previous	0-7	Số lần liên lạc với khách hàng trước khi chiến dịch bắt đầu
housing	Yes, No, Unknown	Nhà ở (có đang vay tiền mua nhà hay không)	poutcome	Success, Failure, Nonexistent	Kết quả của chiến dịch
loan	Yes, No, Unknown	Khoản vay cá nhân			
Contact	Cellular, Telephone, Unknown	Phương thức liên lạc			

2.1 Dữ liệu trước tiên xử lý (41188 dòng x 21 cột)

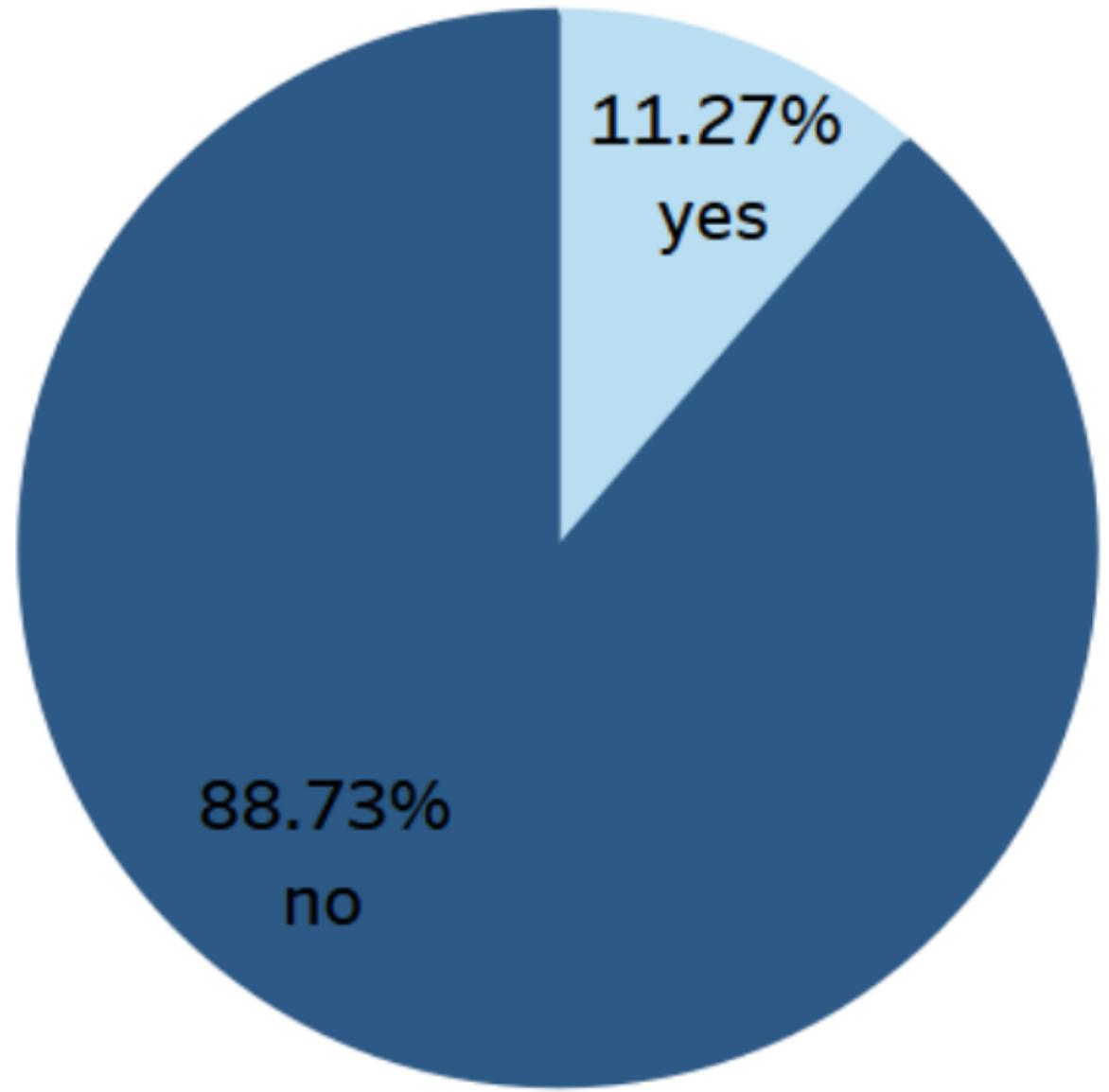


16	emp.var.rate	(-2.9) - 1.4	Tỷ lệ thay đổi việc làm của các khách hàng tham gia khảo sát
17	cons.price.idx	92.201 - 94.767	Chi số giá tiêu dùng của các khách hàng tham gia khảo sát
18	cons.conf.idx	(-50.8) - (-26.9)	Chi số niềm tin tiêu dùng của các khách hàng tham gia khảo sát
19	euribor3m	0.634 - 5.045	Xác định lãi suất 3 tháng của các khách hàng tham gia khảo sát
20	nr.employed		Số nhân viên tham gia vào chiến dịch
21	y	Yes, No	Khách hàng có đăng ký một khoản tiền gửi tiết kiệm có kỳ hạn hay không

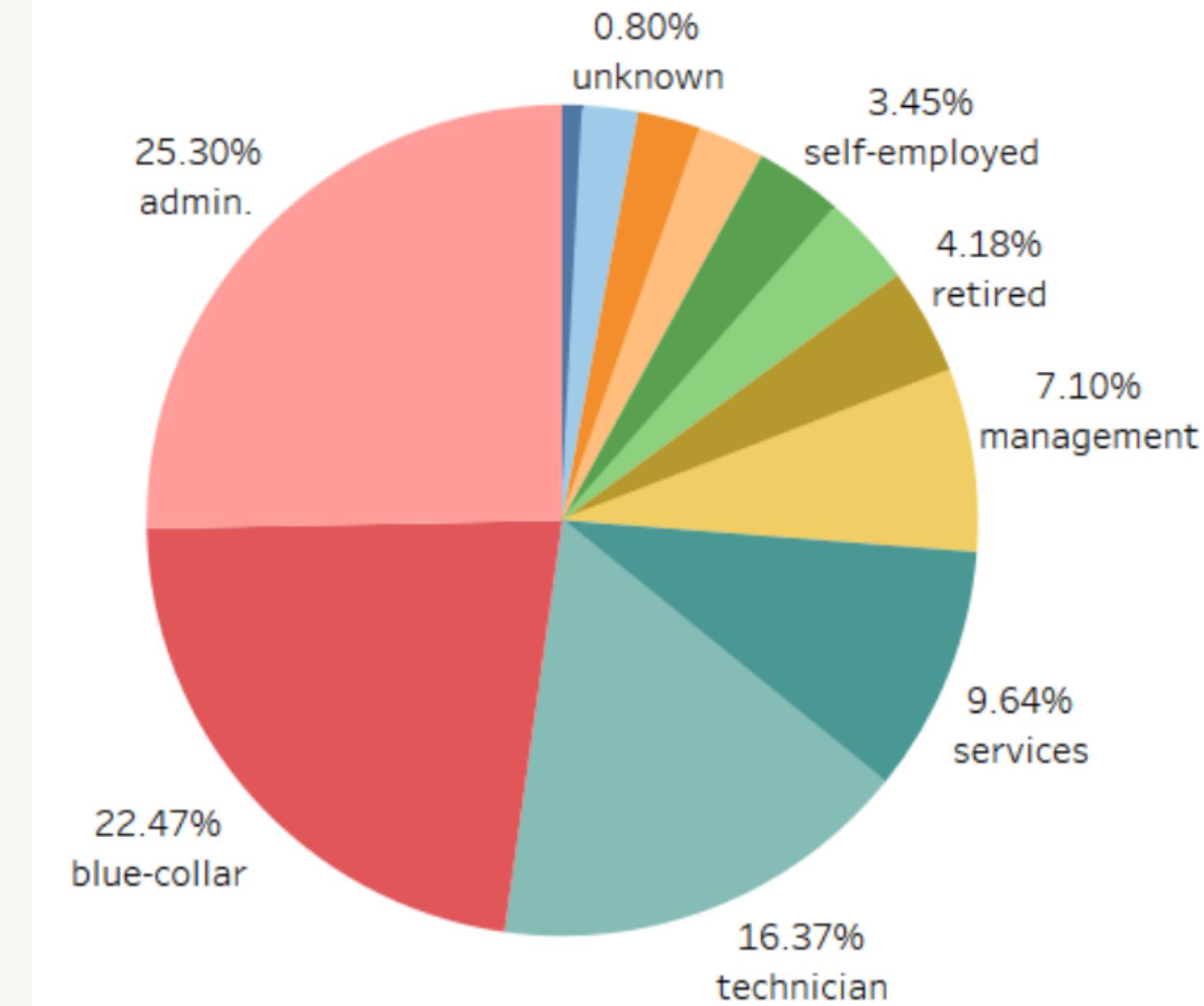


Tổng quan dữ liệu trước tiền xử lý

Tỷ lệ đồng ý khoản cho vay



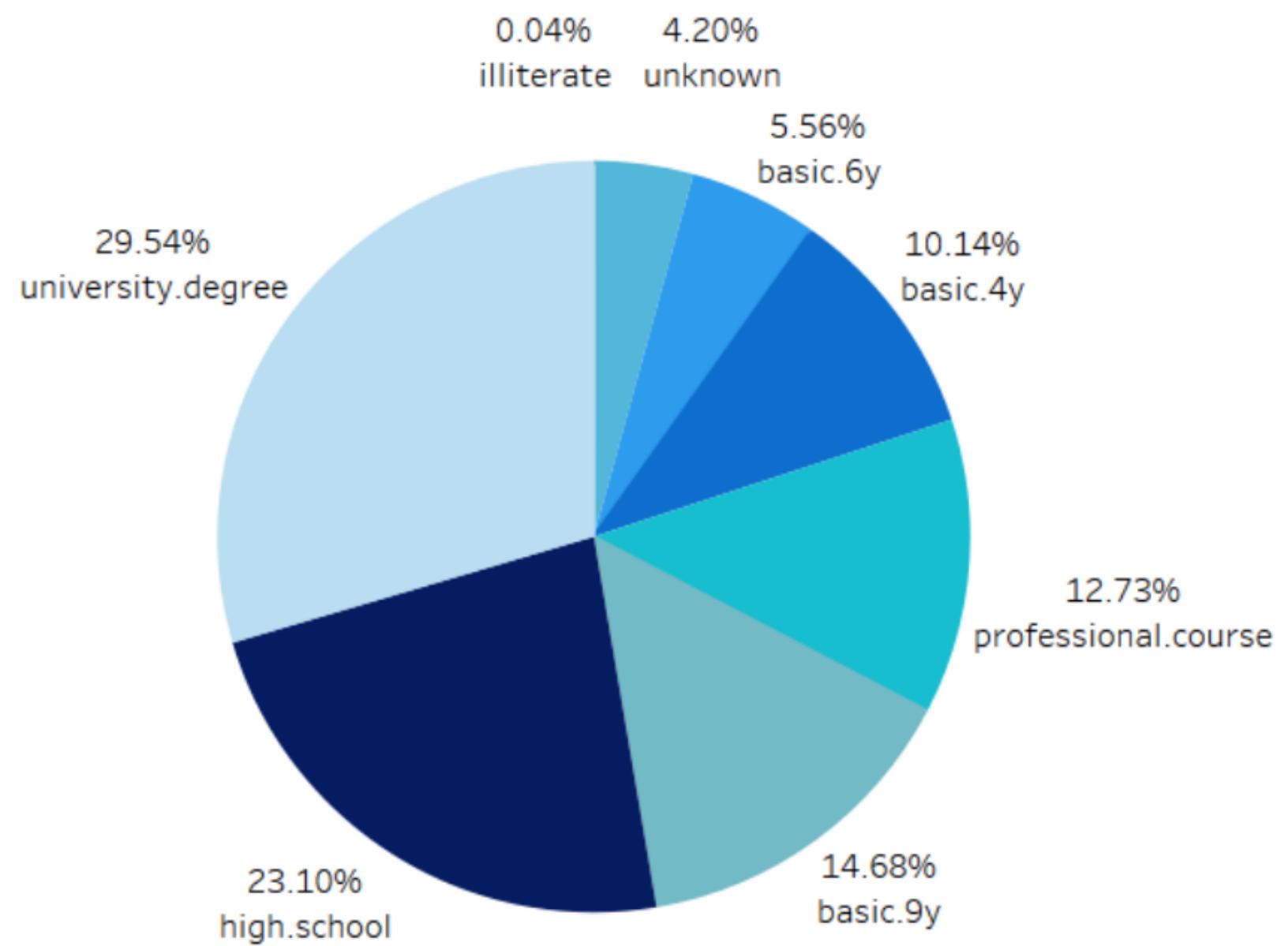
Nghề nghiệp của khách hàng



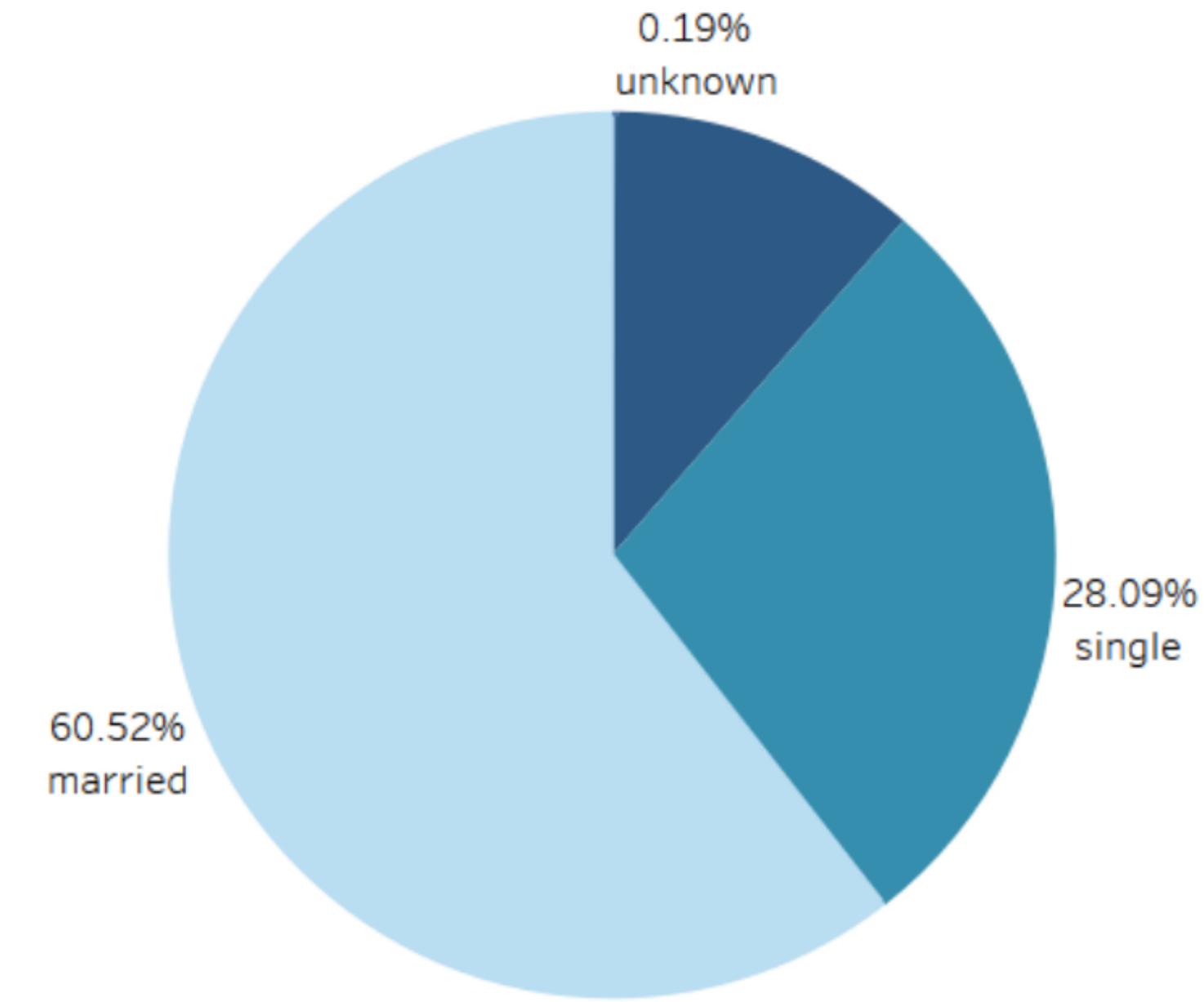
Trong 41188 khách hàng tham gia vào chiến dịch thì chỉ có 11.27% khách hàng có đồng ý gửi tiết kiệm.

Tổng quan dữ liệu trước tiền xử lý

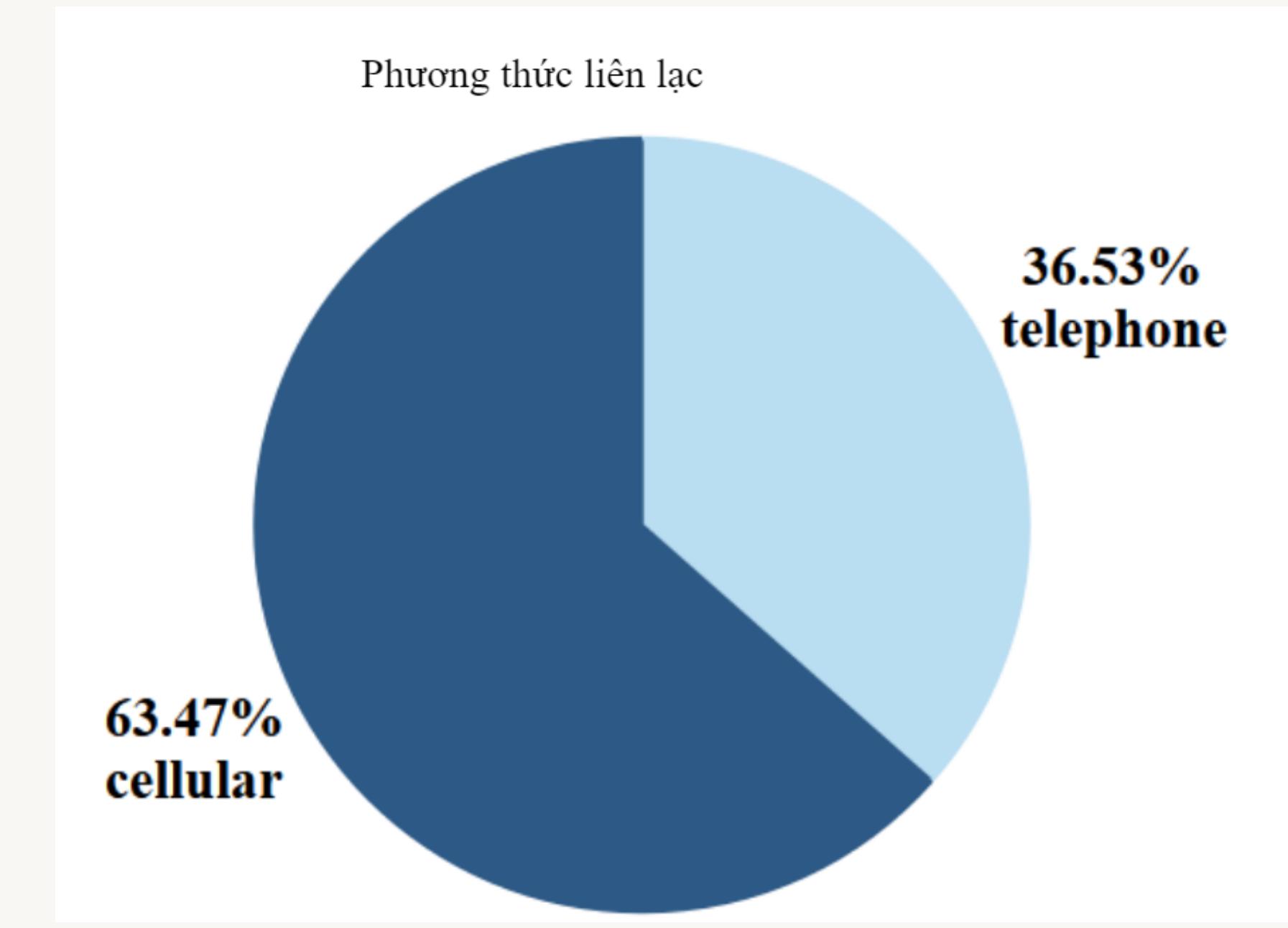
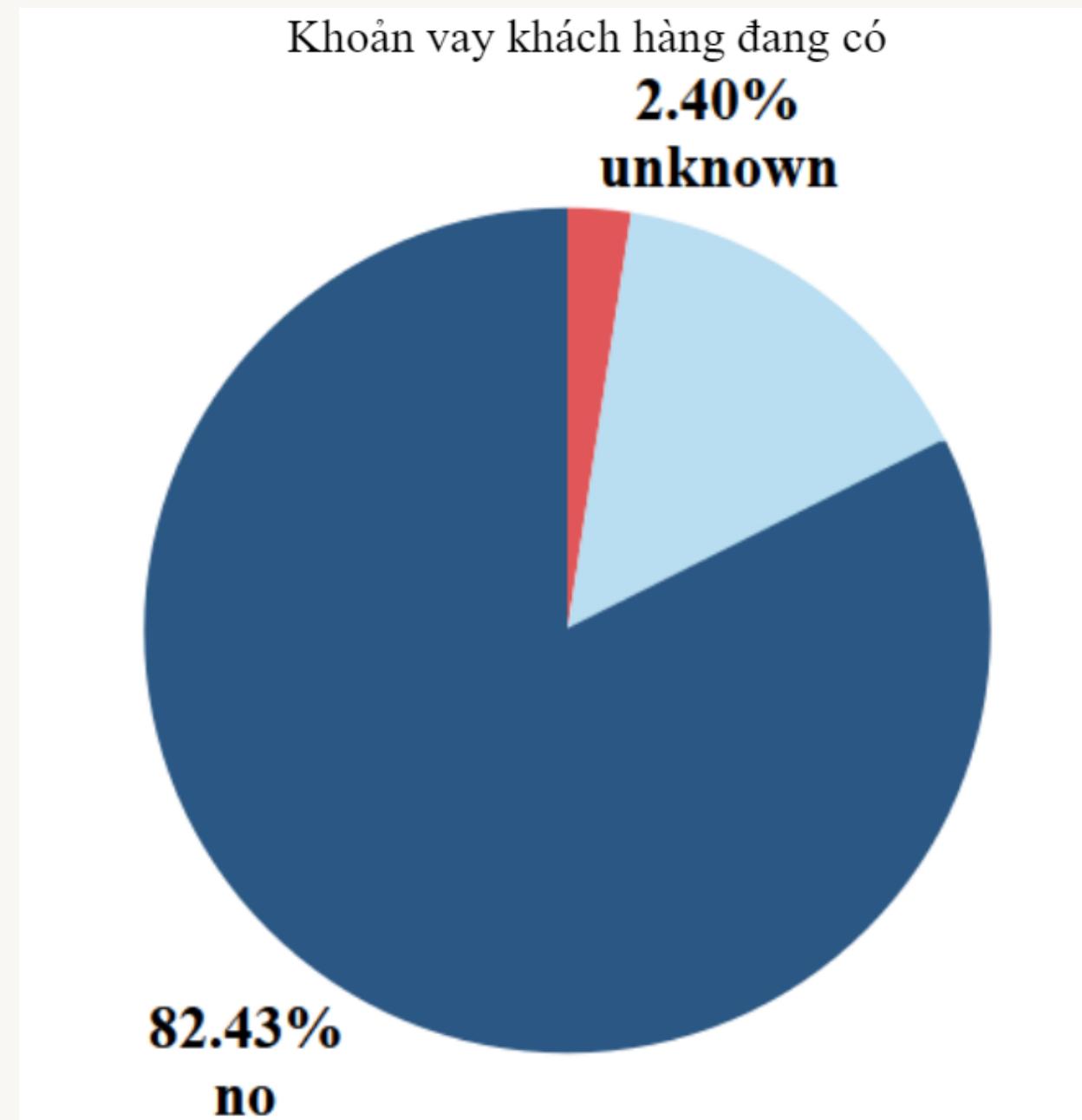
Trình độ học vấn

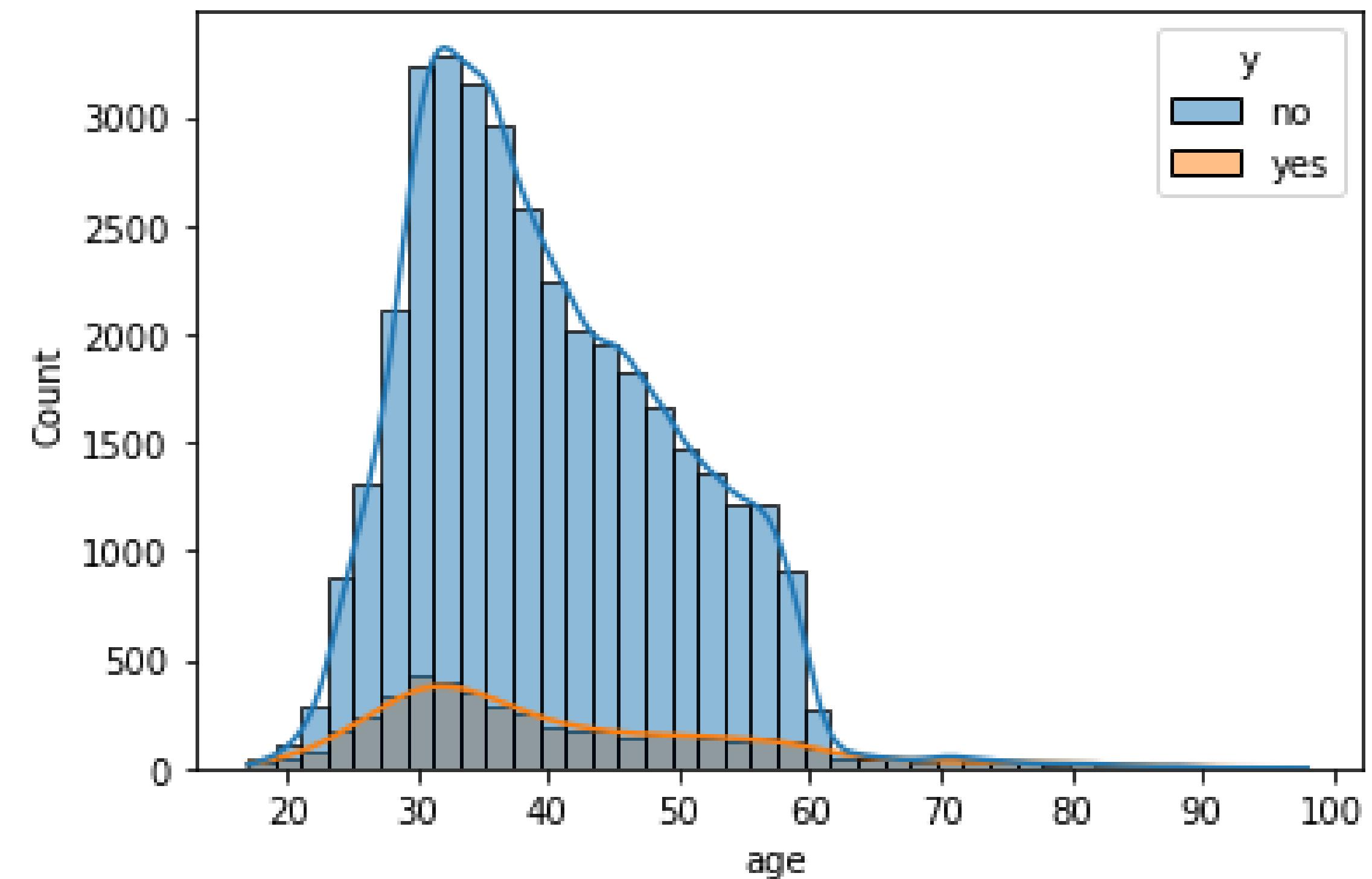


Tình trạng hôn nhân



Tổng quan dữ liệu trước tiền xử lý





- Phần lớn khách tham gia chiến dịch là từ 28-55 tuổi
- Độ tuổi 25-45 có tỉ lệ đồng ý gửi tiền cao hơn so với số còn lại

2.2 Tiền xử lý

```
data.isnull().sum()
```

```
age          0  
job          0  
marital      0  
education    0  
default       0  
housing       0  
loan          0  
contact       0  
month         0
```

```
data.duplicated().sum()
```

```
data.drop_duplicates(inplace=True)
```

2.2 Tiền xử lý

```
data = data.replace({'y': {'no': 0, 'yes': 1}})
```

```
data['pdays'] = data['pdays'].replace(999,0)
```

```
data = pd.get_dummies(data, columns=["job", "marital", "education", "default", "housing", "loan",  
"contact", "month", "day_of_week", "poutcome"], drop_first=True)
```

```
scaler = StandardScaler()  
data_sc = data.copy().drop(columns='y')  
data_y = data['y']  
data_scales = scaler.fit_transform(data_sc)  
data = pd.DataFrame(data_scales, index=data_sc.index, columns=data_sc.columns)  
data['y'] = data_y
```

2.2 Tiền xử lý

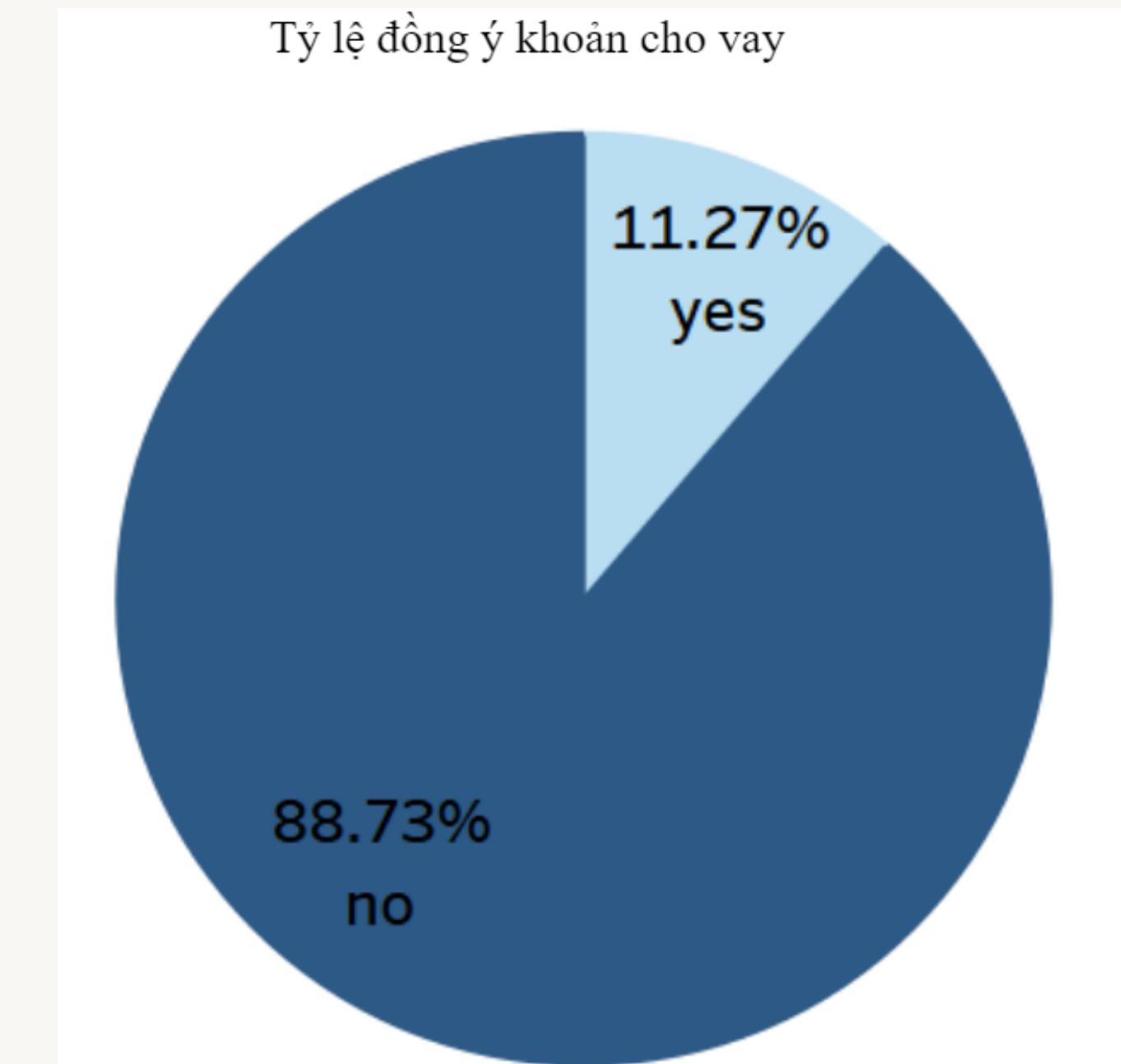
	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
0	56	261	1	0	0	1.1	93.994	-36.4	4.857	5191.0
1	57	149	1	0	0	1.1	93.994	-36.4	4.857	5191.0
2	37	226	1	0	0	1.1	93.994	-36.4	4.857	5191.0
3	40	151	1	0	0	1.1	93.994	-36.4	4.857	5191.0
4	56	307	1	0	0	1.1	93.994	-36.4	4.857	5191.0

	month_may	month_nov	month_oct	month_sep	day_of_week_mon	day_of_week_thu	day_of_week_tue
	1	0	0	0	1	0	0
	1	0	0	0	1	0	0
	1	0	0	0	1	0	0
	1	0	0	0	1	0	0

2.3 Giải quyết mất cân bằng dữ liệu

Vấn đề của imbalance

- Hầu hết các thuật toán phân lớp hoạt động tốt với balance dataset
- Khi dataset không balance:
 - Model có thiên hướng dự đoán ra lớp Majority để tăng Accuracy
 - Accuracy không còn tác dụng đánh giá



2.3 Giải quyết mất cân bằng dữ liệu

Xây dựng và đánh giá các mô hình khi dữ liệu bị mất cân bằng

Chọn biến input và output

```
X = data.drop(['duration', 'y'], axis=1)  
Y = data["y"]
```

Chia tập dữ liệu thành 2 tập train(80%) và test(20%)

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 42)
```

2.3 Giải quyết mất cân bằng dữ liệu

Xây dựng và đánh giá các mô hình khi dữ liệu bị mất cân bằng

KNN

```
error = []
# Calculating error for K values between 1 and 30
for i in range(1, 20):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train, y_train)
    pred_i = knn.predict(x_test)
    error.append(np.mean(pred_i != y_test))
k_toi_uu = error.index(min(error)) + 1
print("Minimum error:-", min(error), "at K =", k_toi_uu)
```

Minimum error:- 0.10684798445847499 at K = 14

```
knn = KNeighborsClassifier (n_neighbors = k_toi_uu)
list_model.append(knn)
```

2.3 Giải quyết mất cân bằng dữ liệu

Xây dựng và đánh giá các mô hình khi dữ liệu bị mất cân bằng
Logistic Regression

```
lr = LogisticRegression()  
list_model.append(lr)
```

Random Forest

```
rf = RandomForestClassifier()  
list_model.append(rf)
```

Naive Bayes

```
nb = GaussianNB()  
list_model.append(nb)
```

2.3 Giải quyết mất cân bằng dữ liệu

Xây dựng và đánh giá các mô hình khi dữ liệu bị mất cân bằng

Neural Network (Multi Layer Perceptron)

```
mlp = MLPClassifier()  
list_model.append(mlp)
```

Support Vector Machines

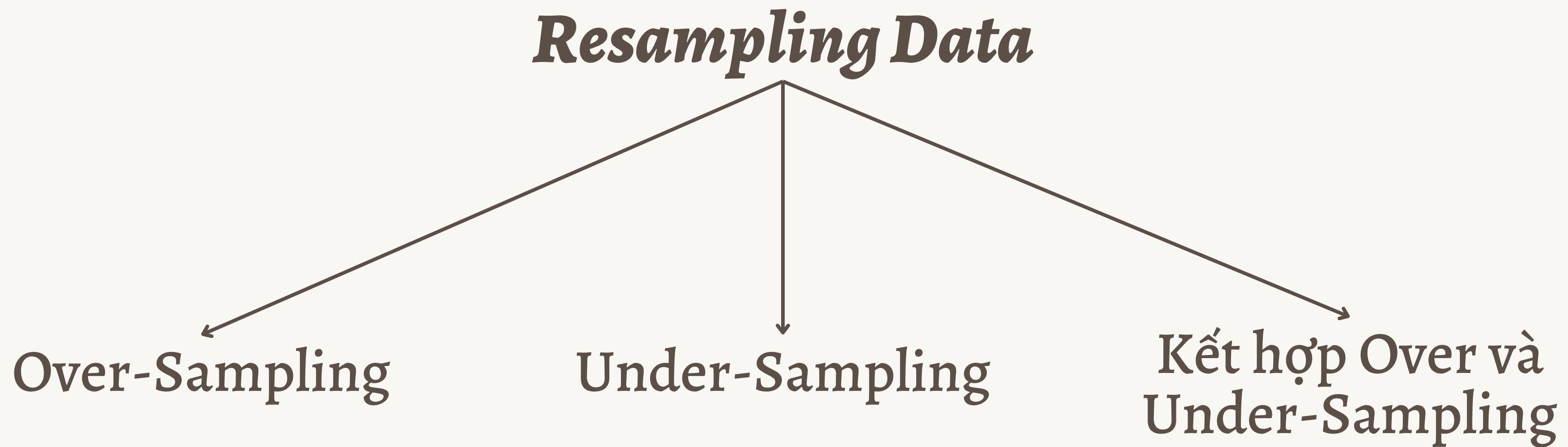
```
svm = SVC(kernel='linear')  
list_model.append(svm)
```

2.3 Giải quyết mất cân bằng dữ liệu

Xây dựng và đánh giá các mô hình khi dữ liệu bị mất cân bằng

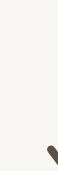
Model name	Accuracy (%)	Precision	Recall	F1-score	Running time
Logistic Regression	89.38	0.67	0.20	0.31	0:00:00.636904
SVC	89.33	0.67	0.19	0.29	0:01:29.777268
KNN	89.32	0.64	0.21	0.32	0:00:02.026124
Random Forest	88.89	0.56	0.28	0.37	0:00:06.461850
Neural Network	88.22	0.50	0.30	0.37	0:00:48.166419
Naive Bayes	72.94	0.26	0.70	0.38	0:00:00.326902

2.3 Giải quyết mất cân bằng dữ liệu

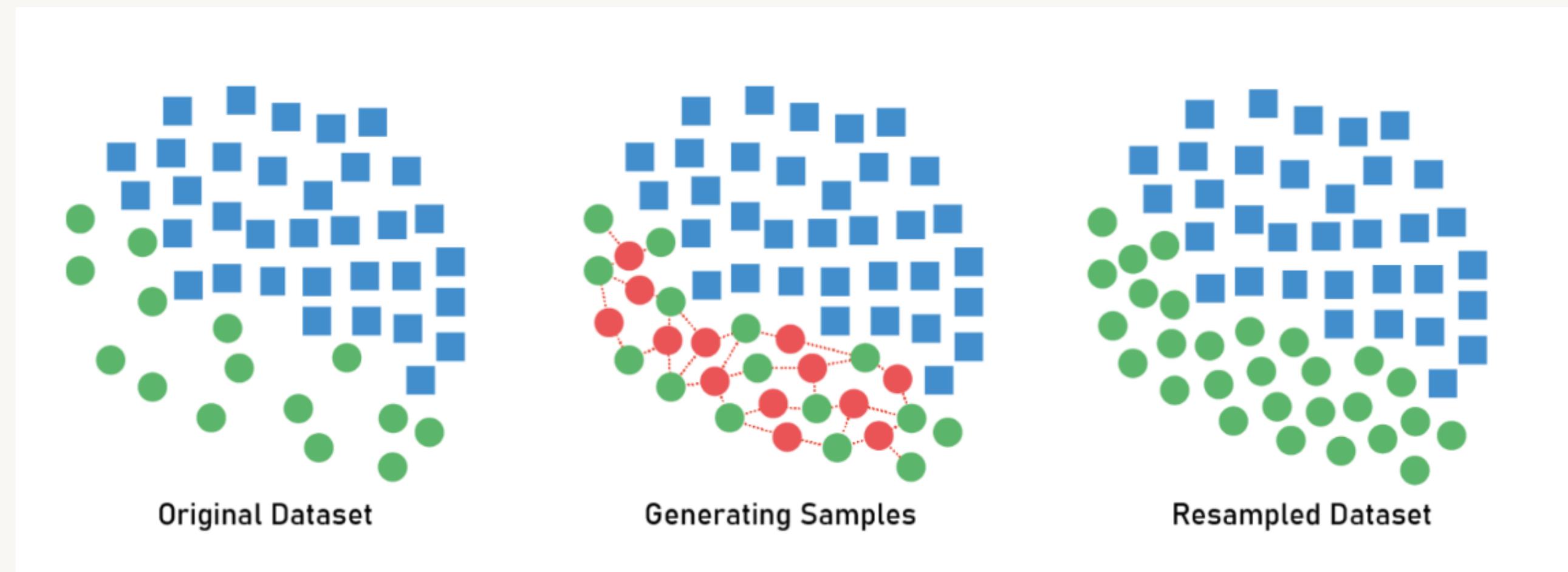


2.3 Giải quyết mất cân bằng dữ liệu

Over-Sampling

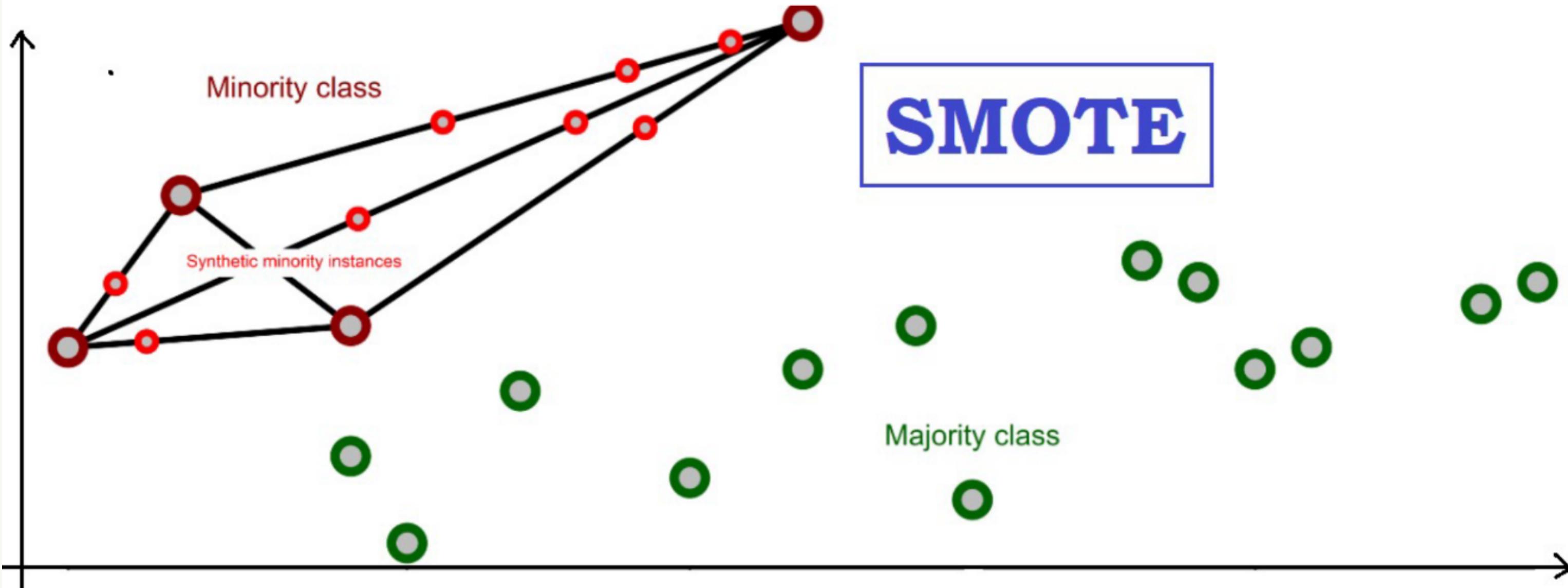


Sythetic Minority Over-sampling Technique



2.3 Giải quyết mất cân bằng dữ liệu

SMOTE



2.3 Giải quyết mất cân bằng dữ liệu

SMOTE

```
sm = SMOTE()  
  
X_sm, Y_sm = sm.fit_resample(X_res, Y_res)
```

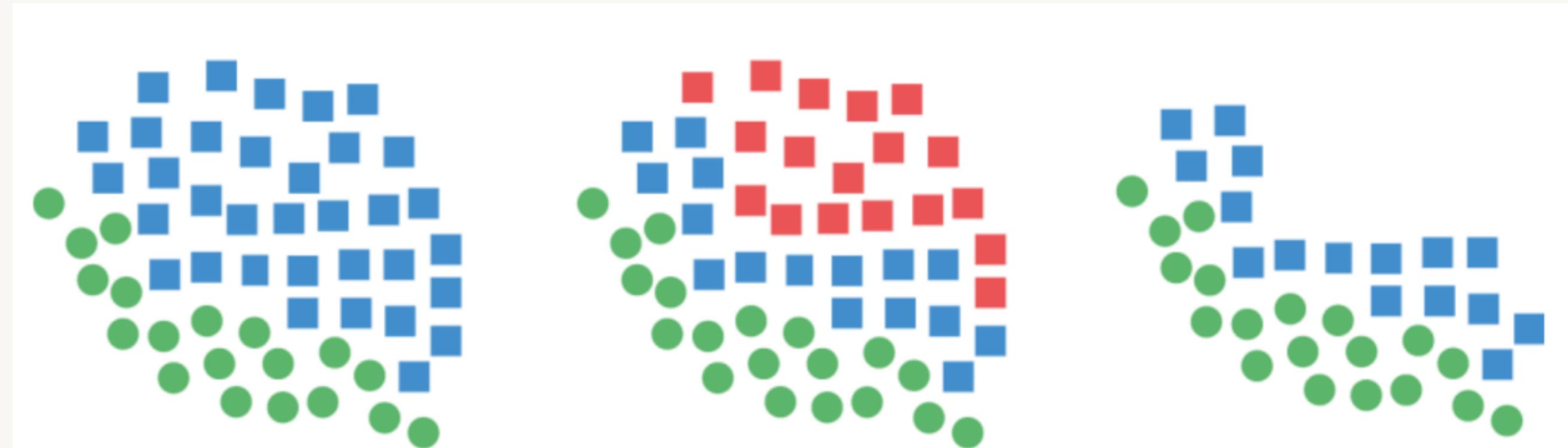
Model name	Accuracy (%)	Precision	Recall	F1-score	Running time
Random Forest	93.65	0.94	0.93	0.94	0:00:08.344488
Neural Network	88.89	0.89	0.89	0.89	0:00:57.232408
KNN	80.81	0.77	0.89	0.82	0:00:04.218817
Logistic Regression	74.77	0.80	0.65	0.72	0:00:00.685487
SVC	74.20	0.82	0.62	0.71	0:43:15.581848
Naive Bayes	69.93	0.68	0.76	0.71	0:00:00.197165

2.3 Giải quyết mất cân bằng dữ liệu

Under-Sampling

Tomek Links

Edited Nearest Neighbour



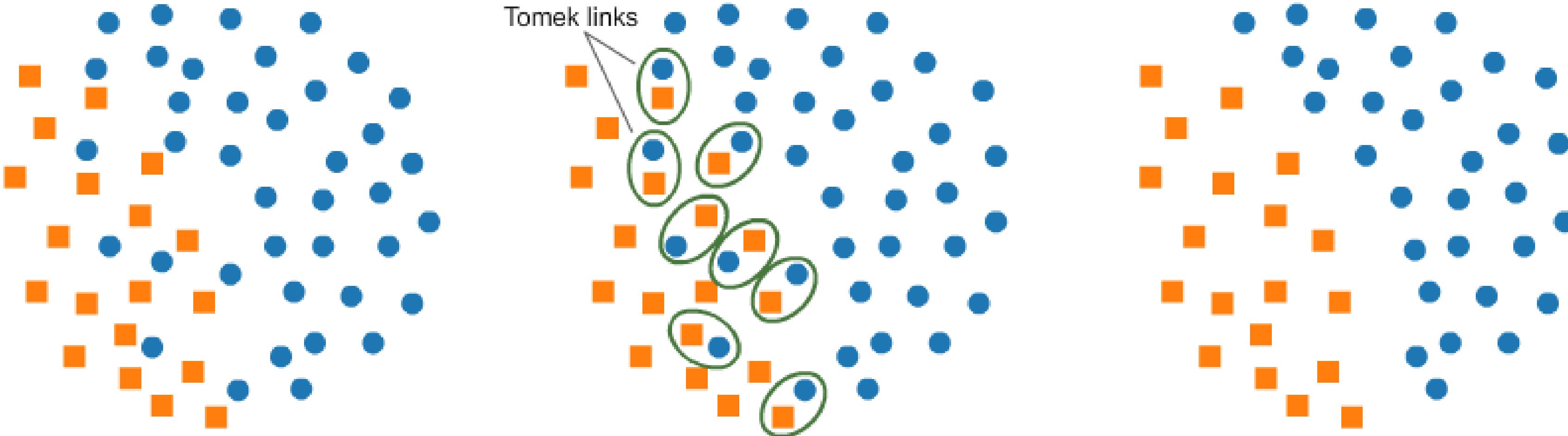
Original Dataset

Selecting Samples

Resampled Dataset

2.3 Giải quyết mất cân bằng dữ liệu

Tomek Links



2.3 Giải quyết mất cân bằng dữ liệu

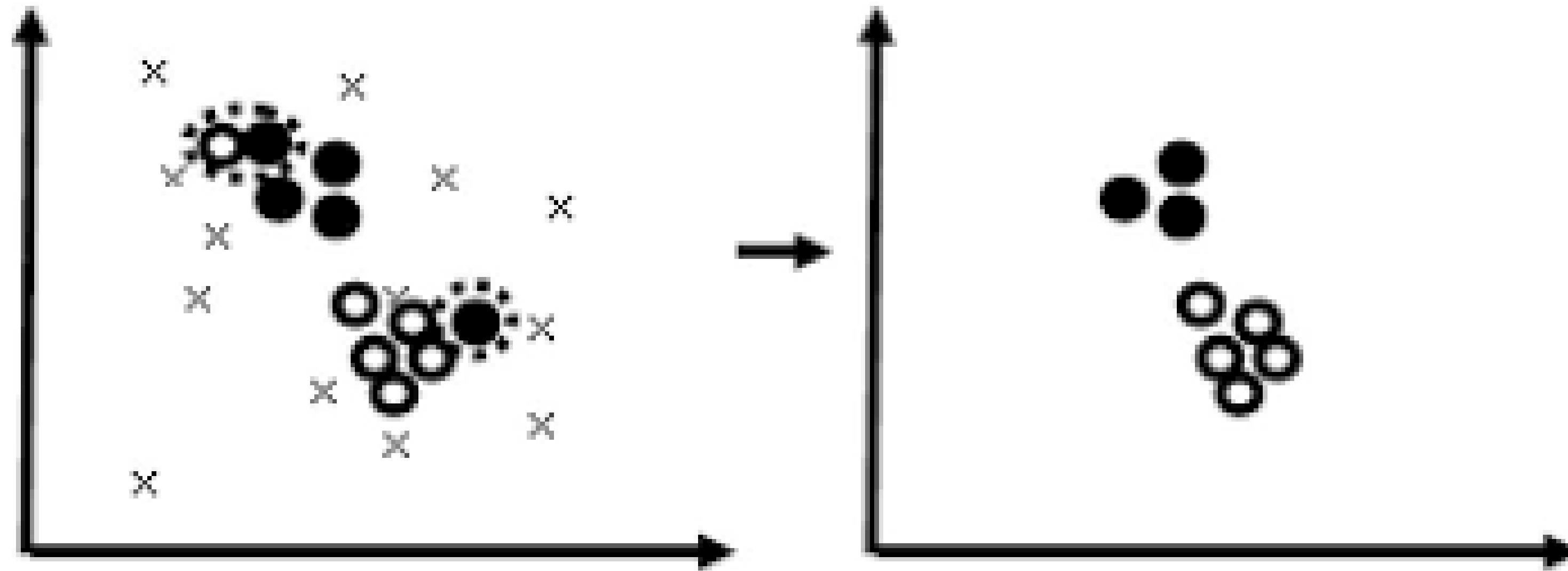
Tomek Links

```
t1 = TomekLinks()  
  
X_t1, Y_t1 = t1.fit_resample(X_res, Y_res)
```

Model name	Accuracy (%)	Precision	Recall	F1-score	Running time
Random Forest	90.28	0.64	0.37	0.47	0:00:03.935720
Logistic Regression	90.21	0.68	0.29	0.40	0:00:00.323113
KNN	90.06	0.67	0.27	0.39	0:00:01.573187
SVC	90.02	0.75	0.21	0.32	0:01:03.806668
Neural Network	88.97	0.53	0.39	0.45	0:00:30.506083
Naive Bayes	75.25	0.28	0.70	0.40	0:00:00.150103

2.3 Giải quyết mất cân bằng dữ liệu

Edited Nearest Neighbour



a. Original Dataset

b. Edited Dataset



2.3 Giải quyết mất cân bằng dữ liệu

Edited Nearest Neighbour

```
enn = EditedNearestNeighbours()  
  
X_enn, Y_enn = enn.fit_resample(X_res,Y_res)
```

Model name	Accuracy (%)	Precision	Recall	F1-score	Running time
Random Forest	92.66	0.84	0.55	0.66	0:00:04.729511
Neural Network	91.50	0.77	0.50	0.60	0:00:46.155550
KNN	91.29	0.84	0.42	0.56	0:00:01.849712
Logistic Regression	91.25	0.77	0.47	0.59	0:00:00.624700
SVC	90.98	0.72	0.51	0.60	0:54:14.068766
Naive Bayes	83.57	0.41	0.62	0.50	0:00:00.231158

2.3 Giải quyết mất cân bằng dữ liệu

Kết hợp Over và Under-Sampling

SMOTETomek

SMOTEENN



2.3 Giải quyết mất cân bằng dữ liệu

SMOTETomek

```
sme = SMOTEENN(random_state=42)
```

```
X_sme, Y_sme = sme.fit_resample(X_res, Y_res)
```

Model name	Accuracy (%)	Precision	Recall	F1-score	Running time
Random Forest	94.03	0.95	0.93	0.94	0:00:09.557598
Neural Network	89.01	0.88	0.91	0.89	0:00:58.836810
KNN	81.13	0.77	0.89	0.82	0:00:04.163559
Logistic Regression	73.88	0.79	0.64	0.71	0:00:00.486180
SVC	73.47	0.81	0.61	0.69	0:41:59.141477
Naive Bayes	69.93	0.68	0.75	0.71	0:00:00.201385



2.3 Giải quyết mất cân bằng dữ liệu

SMOTEENN

```
sme = SMOTEENN(random_state=42)
```

```
X_sme, Y_sme = sme.fit_resample(X_res, Y_res)
```

Model name	Accuracy (%)	Precision	Recall	F1-score	Running time
Random Forest	96.51	0.98	0.96	0.97	0:00:07.283087
Neural Network	92.68	0.95	0.91	0.93	0:00:53.765639
KNN	85.62	0.84	0.91	0.87	0:00:03.595982
Logistic Regression	77.55	0.88	0.68	0.77	0:00:00.604769
SVC	76.69	0.92	0.62	0.74	0:31:04.262583
Naive Bayes	75.17	0.85	0.66	0.74	0:00:00.202057

2.3 Giải quyết mất cân bằng dữ liệu

So sánh các phương pháp

Res method	AVG Accuracy	AVG Precision	AVG Recall	AVG F1-score
Imbalanced data	86.346667	0.550000	0.313333	0.340000
ENN	90.208333	0.725000	0.511667	0.585000
Tomek Links	87.465000	0.591667	0.371667	0.405000
SMOTE	80.375000	0.816667	0.790000	0.798333
SMOTEENN	84.036667	0.903333	0.790000	0.836667
SMOTETomek	80.241667	0.813333	0.788333	0.793333

2.3 Giải quyết mất cân bằng dữ liệu

So sánh các phương pháp

Res method	AVG Accuracy	AVG Precision	AVG Recall	AVG F1-score
ENN	90.208333	0.725000	0.511667	0.585000
Tomek Links	87.465000	0.591667	0.371667	0.405000
Imbalanced data	86.346667	0.550000	0.313333	0.340000
SMOTEENN	84.036667	0.903333	0.790000	0.836667
SMOTE	80.375000	0.816667	0.790000	0.798333
SMOTETomek	80.241667	0.813333	0.788333	0.793333

2.3 Giải quyết mất cân bằng dữ liệu

So sánh các phương pháp

Res method	AVG Accuracy	AVG Precision	AVG Recall	AVG F1-score
SMOTEENN	84.036667	0.903333	0.790000	0.836667
SMOTE	80.375000	0.816667	0.790000	0.798333
SMOTETomek	80.241667	0.813333	0.788333	0.793333
ENN	90.208333	0.725000	0.511667	0.585000
Tomek Links	87.465000	0.591667	0.371667	0.405000
Imbalanced data	86.346667	0.550000	0.313333	0.340000

2.3 Giải quyết mất cân bằng dữ liệu

So sánh các phương pháp

Res method	AVG Accuracy	AVG Precision	AVG Recall	AVG F1-score
SMOTE	80.375000	0.816667	0.790000	0.798333
SMOTEEENN	84.036667	0.903333	0.790000	0.836667
SMOTETomek	80.241667	0.813333	0.788333	0.793333
ENN	90.208333	0.725000	0.511667	0.585000
Tomek Links	87.465000	0.591667	0.371667	0.405000
Imbalanced data	86.346667	0.550000	0.313333	0.340000

2.3 Giải quyết mất cân bằng dữ liệu

So sánh các phương pháp

Res method	AVG Accuracy	AVG Precision	AVG Recall	AVG F1-score
SMOTEENN	84.036667	0.903333	0.790000	0.836667
SMOTE	80.375000	0.816667	0.790000	0.798333
SMOTETomek	80.241667	0.813333	0.788333	0.793333
ENN	90.208333	0.725000	0.511667	0.585000
Tomek Links	87.465000	0.591667	0.371667	0.405000
Imbalanced data	86.346667	0.550000	0.313333	0.340000

2.3 Giải quyết mất cân bằng dữ liệu

Sử dụng SMOTEENN

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	...
0	56	261	1	0	0	1.100000	93.994000	-36.400000	4.857000	5191.000000	...
1	57	149	1	0	0	1.100000	93.994000	-36.400000	4.857000	5191.000000	...
2	37	226	1	0	0	1.100000	93.994000	-36.400000	4.857000	5191.000000	...
3	40	151	1	0	0	1.100000	93.994000	-36.400000	4.857000	5191.000000	...
4	56	307	1	0	0	1.100000	93.994000	-36.400000	4.857000	5191.000000	...
...
64303	48	1985	1	0	0	0.130689	93.352639	-40.923452	4.287953	5194.877245	...
64304	34	371	1	0	0	-3.400000	92.428382	-27.046012	0.742417	5017.500000	...
64305	37	636	2	0	0	1.400000	93.918000	-42.700000	4.959953	5228.100000	...
64306	24	724	2	0	0	1.400000	93.918000	-42.700000	4.967199	5228.100000	...
64307	28	598	2	2	2	-1.100000	94.199000	-37.500000	0.879953	4963.600000	...

64308 rows × 54 columns

2.3 Giải quyết mất cân bằng dữ liệu

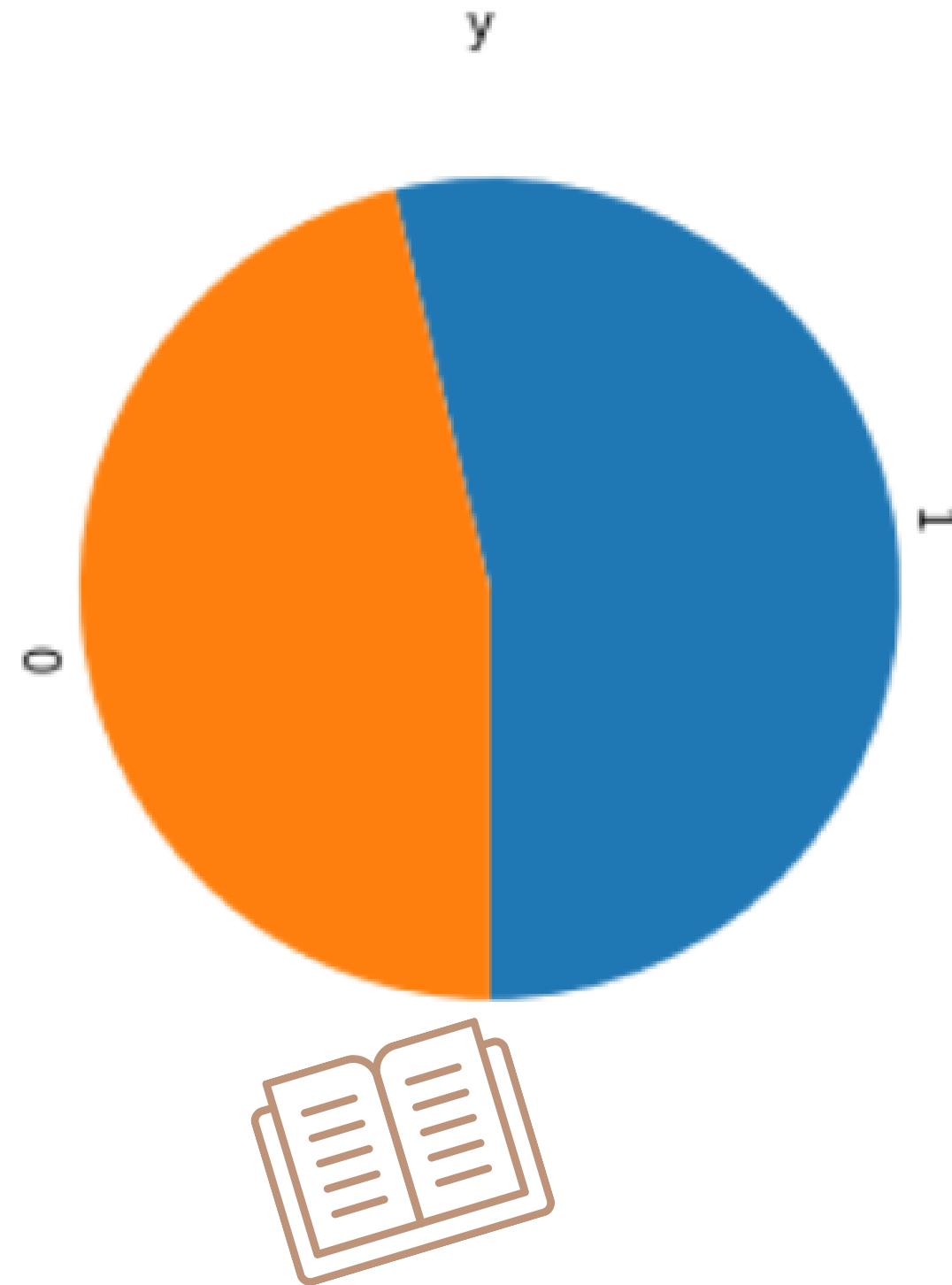
Sử dụng SMOTEENN

1

0

34693

29615



2.4 Dữ liệu sau tiền xử lý (64308 dòng x 51 cột)

Tên cột	Ý nghĩa
age	Tuổi
duration	Khoảng thời gian liên hệ với khách hàng (khoảng thời gian gọi đt cho KH, tính bằng giây)
campaign	Số lần liên lạc với khách hàng trong suốt chiến dịch
pdays	Số ngày trôi qua sau khi liên hệ lần cuối
previous	Số lần liên lạc với khách hàng trước khi chiến dịch bắt đầu
emp.var.rate	Tỷ lệ thay đổi việc làm của các khách hàng tham gia khảo sát
cons.price.idx	Chi số giá tiêu dùng của các khách hàng tham gia khảo sát
cons.conf.idx	Chi số niềm tin tiêu dùng của các khách hàng tham gia khảo sát
euribor3m	Xác định lãi suất 3 tháng của các khách hàng tham gia khảo sát
nr.employed	Chi số nhân viên tham gia vào chiến dịch
y	Khách hàng có đăng ký một khoản tiền gửi tiết kiệm có kỳ hạn hay không
job_blue-collar	Nghề nghiệp: Công nhân phổ thông
job_entrepreneur	Nghề nghiệp: Chủ doanh nghiệp
job_housemaid	Nghề nghiệp: Người giúp việc nhà

job management	Nghề nghiệp: Nhà quản lý
job_retired	Nghề nghiệp: Đã nghỉ hưu
job_self-employed	Nghề nghiệp: Kinh doanh tự do
job_services	Nghề nghiệp: Dịch vụ
job_student	Nghề nghiệp: Sinh viên
job_technician	Nghề nghiệp: Kỹ thuật viên
job_unemployed	Nghề nghiệp: Thất nghiệp
job_unknown	Nghề nghiệp: Chưa xác định
marital_married	Tình trạng hôn nhân: Đã kết hôn
marital_single	Tình trạng hôn nhân: Độc thân
marital_unknown	Tình trạng hôn nhân: Đã ly hôn
education_basic.6y	Trình độ học vấn: 6 năm tiêu học
education_basic.9y	Trình độ học vấn: Tốt nghiệp trung học
education_high.school	Trình độ học vấn: Tốt nghiệp trung học phổ thông
education_illiterate	Trình độ học vấn: Thất học
education_professional.course	Trình độ học vấn: Đào tạo nghiệp vụ
education_university.degree	Trình độ học vấn: Tốt nghiệp đại học
education_unknown	Trình độ học vấn: Chưa xác định
default_unknown	Tình trạng tín dụng vỡ nợ: Chưa xác định

housing_unknown	Khoản vay nhà ở: Chưa xác định
housing_yes	Khoản vay nhà ở: Có
loan_unknown	Tình trạng vay nợ: Chưa xác định
loan_yes	Tình trạng vay nợ: Có
contact_telephone	Phương thức liên lạc: telephone
month_aug	Tháng liên lạc gần nhất trong năm: 8
month_dec	Tháng liên lạc gần nhất trong năm: 12
month_jul	Tháng liên lạc gần nhất trong năm: 7
month_jun	Tháng liên lạc gần nhất trong năm: 6
month_mar	Tháng liên lạc gần nhất trong năm: 3
month_may	Tháng liên lạc gần nhất trong năm: 5
month_nov	Tháng liên lạc gần nhất trong năm: 11
month_oct	Tháng liên lạc gần nhất trong năm: 10
month_sep	Tháng liên lạc gần nhất trong năm: 9
day_of_week_mon	Thứ liên lạc gần nhất trong tuần: 2
day_of_week_thu	Thứ liên lạc gần nhất trong tuần: 5
day_of_week_tue	Thứ liên lạc gần nhất trong tuần: 3
day_of_week_wed	Thứ liên lạc gần nhất trong tuần: 4
poutcome_nonexistent	Kết quả chiến dịch tiếp thị trước đó: Không tồn tại
outcome_success	Kết quả chiến dịch tiếp thị trước đó: Thành công

2. Trích chọn đặc trưng

Sử dụng phương pháp RFE (Recursive Feature Elimination)

Feature Selection

Full Feature Set



Identify Useful Features



Selected Feature Set



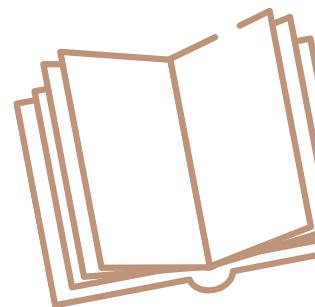
3.Trích chọn đặc trưng

Trích chọn đặc trưng là gì?

Trích chọn đặc trưng (feature selection) là một bước quan trọng trong việc xây dựng mô hình học máy, giúp đánh giá được mức độ tác động của các thuộc tính để từ đó chọn ra được tập thuộc tính tốt nhất để xây dựng mô hình, làm tăng hiệu suất và rút ngắn thời gian huấn luyện mô hình.

Cách thức hoạt động của RFE

Dựa trên các trọng số được gán cho các biến của một mô hình (coef_, feature_importances) để xếp hạng cho các thuộc tính. Mục tiêu của RFE là chọn ra tập thuộc tính tốt nhất để xây dựng mô hình bằng việc đệ quy, qua mỗi vòng lặp thuộc tính được cho là ít quan trọng nhất sẽ được loại bỏ và quá trình này tiếp tục đến khi hàm tìm được một tập thuộc tính tối ưu nhất cho mô hình.



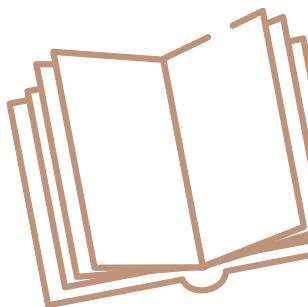
3. Trích chọn đặc trưng



```
X_rfe = data.drop(['duration', 'y'], axis=1)
Y_rfe = data['y']
```

```
feature_selection = data.columns
X_train, X_test, y_train, y_test = train_test_split(X_rfe, Y_rfe, test_size=0.2)
DT = DecisionTreeClassifier()
RFE_DT = RFE(DT)
RFE_DT.fit(X_train, y_train)
col_DT = X_train.columns[RFE_DT.support_]

LG = LogisticRegression()
RFE_LG = RFE(LG)
RFE_LG.fit(X_train, y_train)
col_LG = X_train.columns[RFE_LG.support_]
```



3. Trích chọn đặc trưng

```
RF = RandomForestClassifier()
RFE_RF = RFE(RF)
RFE_RF.fit(X_train, y_train)
col_RF = X_train.columns[RFE_RF.support_]

SVM = SVC(kernel="linear", max_iter=1000)
RFE_SVM = RFE(SVM)
RFE_SVM.fit(X_train, y_train)
col_SVM = X_train.columns[RFE_SVM.support_]

feature_selection = list(set.intersection(*map(set, [col_DT, col_LG, col_RF, col_SVM])))
```

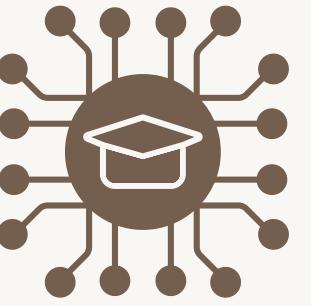


3. Trích chọn đặc trưng

feature_selection

```
[ 'marital_single',
  'education_professional.course',
  'emp.var.rate',
  'day_of_week_mon',
  'previous',
  'contact_telephone',
  'day_of_week_wed',
  'month_may',
  'poutcome_nonexistent',
  'euribor3m',
  'education_basic.9y',
  'default_unknown',
  'day_of_week_thu']
```





4 Xây dựng và đánh giá mô hình



4.1 Xây dựng mô hình

- K Nearest Neighbor (KNN)
 - Logistic Regression
 - Random Forest
 - Naive Bayes
 - Neural Network
 - SVM

4.1 Xây dựng mô hình

Tạo các biến input và output

```
X = data[feature_selection]  
Y = data["y"]
```

Chia 2 phần train(80%) và test(20%)

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 42)
```

4.1 Xây dựng mô hình

KNN

```
error = []

for i in range(1, 20):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train, y_train)
    pred_i = knn.predict(x_test)
    error.append(np.mean(pred_i != y_test))

k_toi_uu = error.index(min(error)) + 1
```

Minimum error:- 0.08218006530866118 at K = 5

```
knn = KNeighborsClassifier (n_neighbors = k_toi_uu)
list_model.append(knn)
```

4.1 Xây dựng mô hình

Logistic Regression

```
lr = LogisticRegression()  
list_model.append(lr)
```

Random Forest

```
rf = RandomForestClassifier()  
list_model.append(rf)
```

Naive Bayes

```
nb = GaussianNB()  
list_model.append(nb)
```

4.1 Xây dựng mô hình

Neural Network (Multi Layer Perceptron)

```
mlp = MLPClassifier()  
list_model.append(mlp)
```

Support Vector Machines

```
svm = SVC(kernel='linear')  
list_model.append(svm)
```

4.1 Xây dựng mô hình

```
model_eval = []
model_name = ['KNN', 'Logistic Regression', 'Random Forest', 'Naive Bayes', "Neural Network", "SVC"]
i = 0

for model in list_model:
    start_time = datetime.now()

    eval_dict = {}
    eval_dict['Model name'] = model_name[i]
    i += 1

    # Xây dựng mô hình
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)

    # Model accuracy
    acc_score = accuracy_score(y_test, y_pred)
    eval_dict['Accuracy (%)'] = round(acc_score, 4)*100

    # Model precision
    precision = precision_score(y_test, y_pred)
    eval_dict['Precision'] = round(precision, 2)
```

4.1 Xây dựng mô hình

```
# Model recall
recall = recall_score(y_test, y_pred)
eval_dict['Recall'] = round(recall, 2)

# Model F1 score
f1_score = metrics.f1_score(y_test, y_pred)
eval_dict['F1-score'] = round(f1_score, 2)

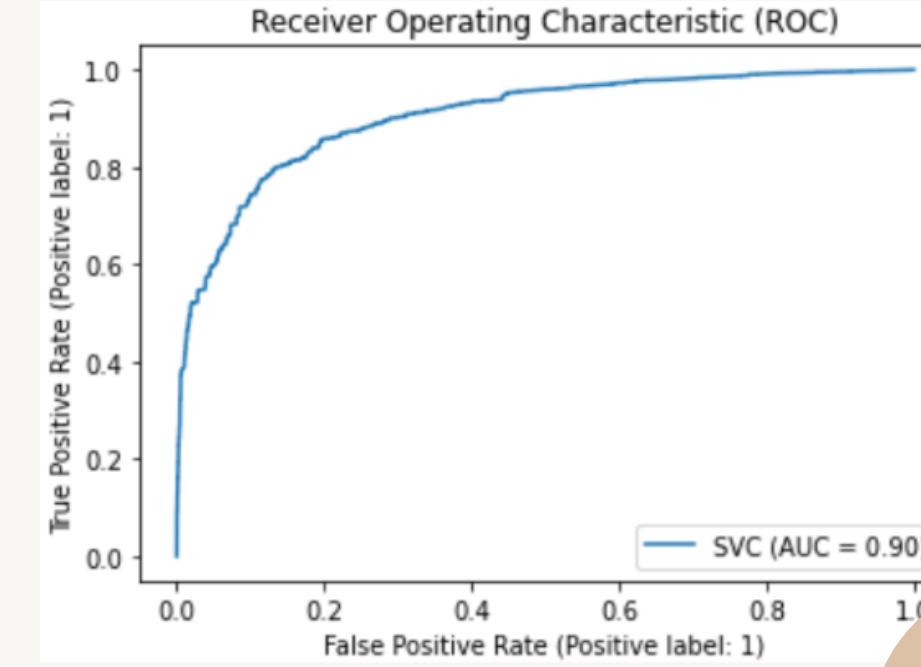
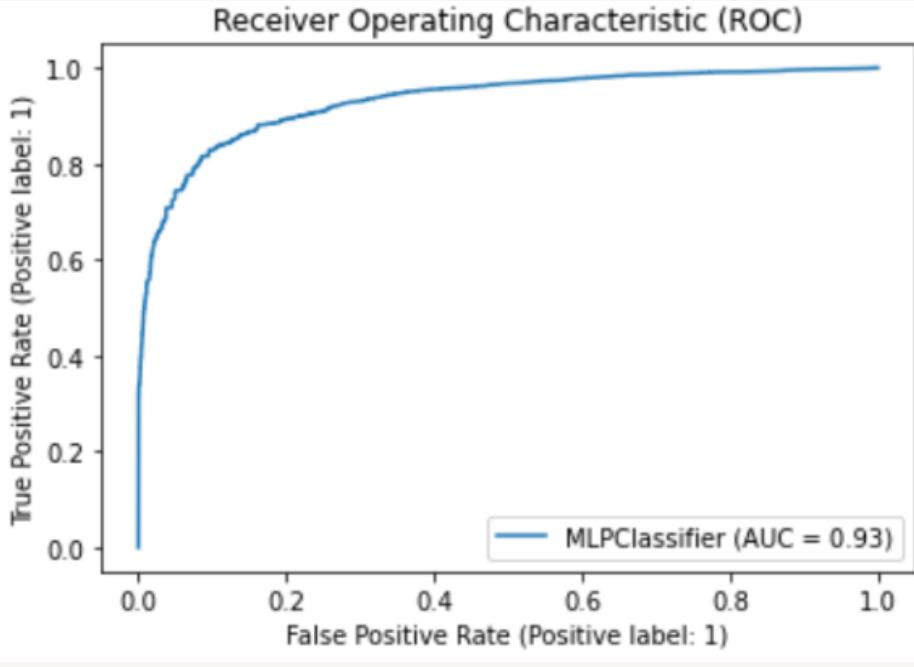
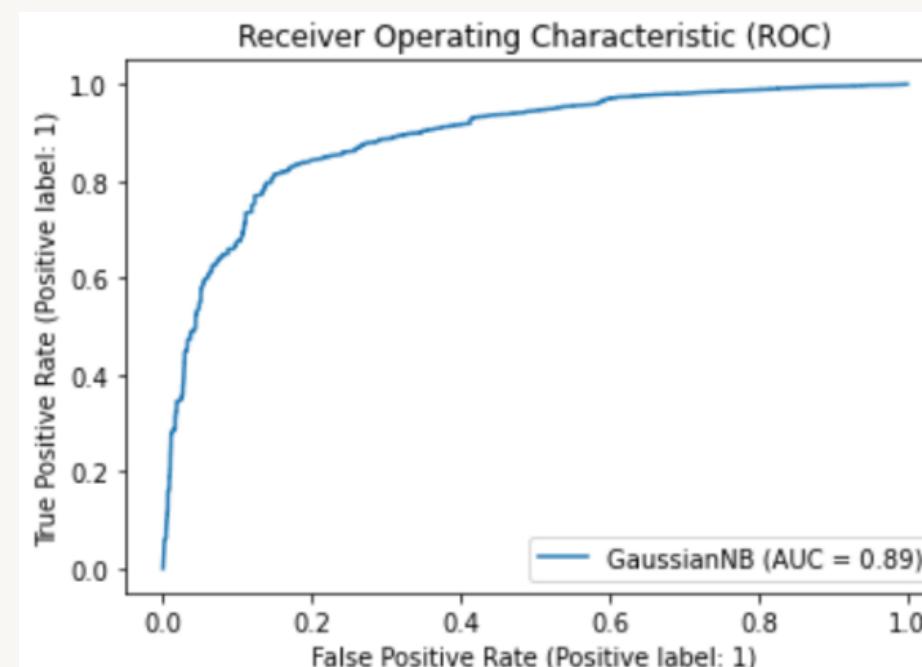
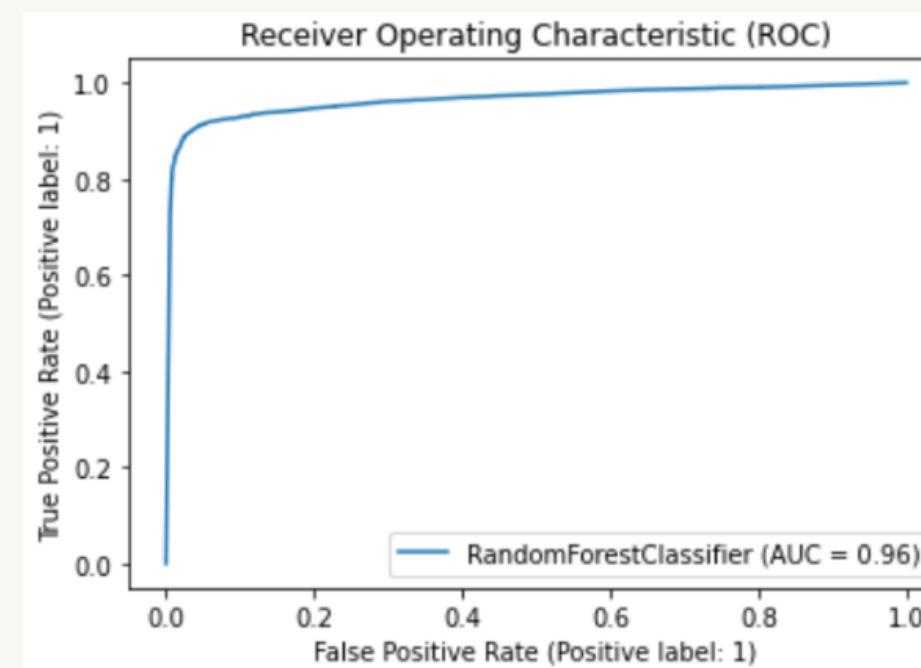
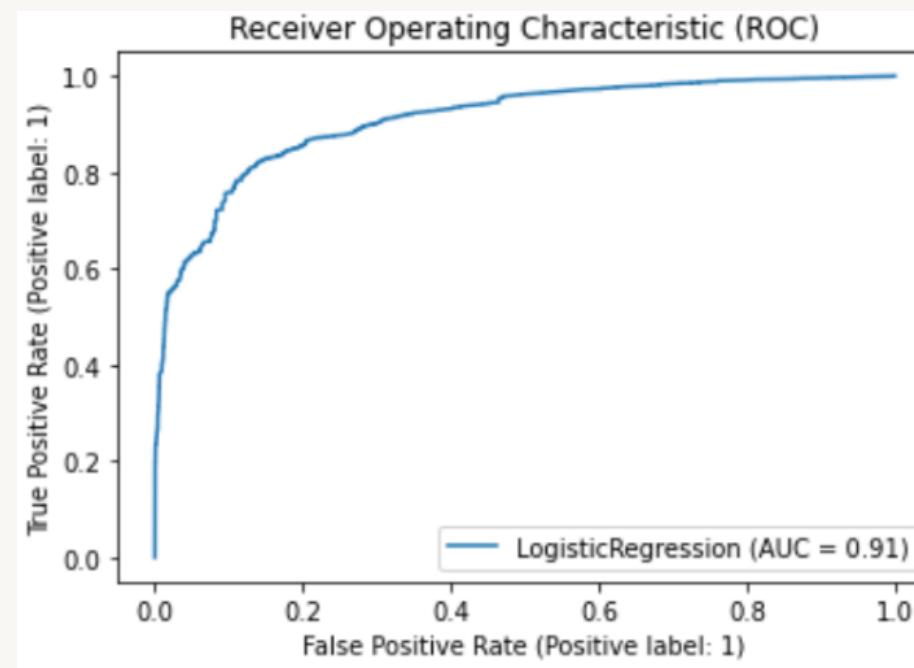
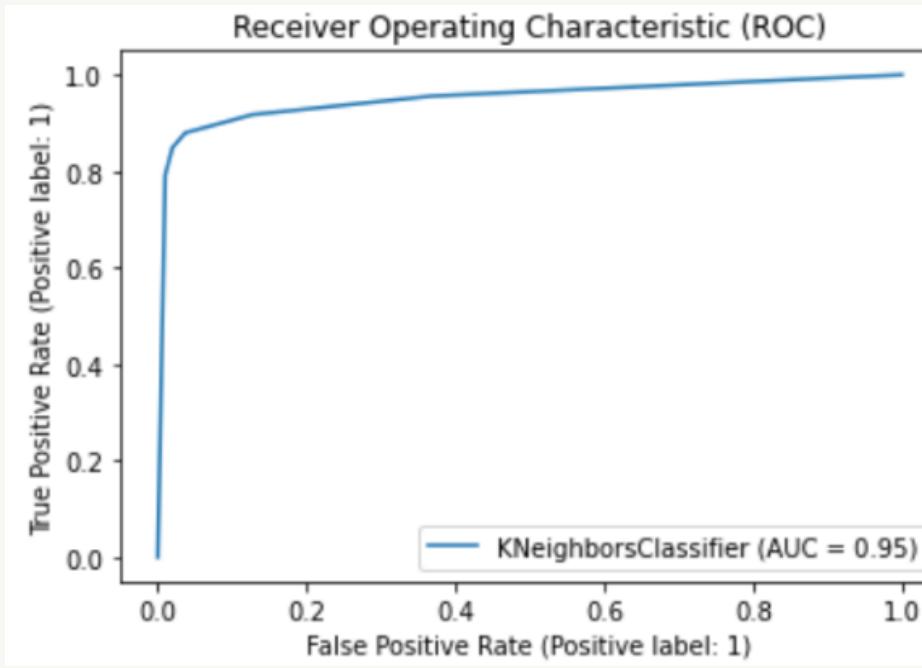
# Model ROC and AUC
RocCurveDisplay.from_estimator(model, x_test, y_test)
plt.title("Receiver Operating Characteristic (ROC)")
plt.show()

# Running time
end_time = datetime.now()
eval_dict['Running time'] = '{}'.format(end_time - start_time)

model_eval.append(eval_dict)

model_eval = pd.DataFrame(model_eval)
```

4.2 Đánh giá mô hình



4.2 Đánh giá mô hình

Model name	Accuracy (%)	Precision	Recall	F1-score	Running time
Random Forest	92.87	0.98	0.89	0.93	0:00:07.579594
KNN	91.78	0.97	0.88	0.92	0:00:20.452834
Neural Network	86.09	0.91	0.83	0.87	0:01:01.350964
Logistic Regression	83.04	0.83	0.86	0.85	0:00:00.435305
SVC	83.03	0.84	0.85	0.85	0:04:00.560216
Naive Bayes	82.79	0.85	0.83	0.84	0:00:00.369362

4.3 Đánh giá mô hình với K fold

K fold cross validation là một phương pháp mà nó tận dụng tối đa dữ liệu trong dataset cho việc huấn luyện mô hình. K fold đặc biệt hữu ích cho việc đánh giá hiệu quả của mô hình khi nó đưa ra nhiều chỉ số đánh giá khác nhau dựa trên nhiều tập train, test khác nhau của dataset, ngoài ra K fold còn giúp cho mô hình tránh trường hợp overfitting và đưa ra kết quả đánh giá khách quan hơn.



4.3 Đánh giá mô hình với K fold

Quy trình thực hiện của K fold:

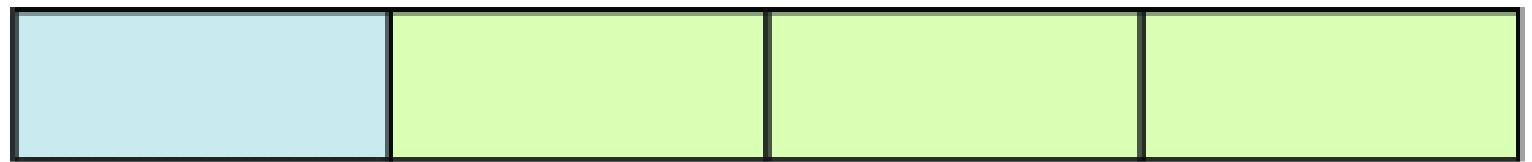
- Đầu tiên dữ liệu sẽ được chia thành K tập dữ liệu có kích thước bằng nhau/tương đối bằng nhau (K folds).
- Sau đó sẽ có K vòng lặp được chạy tương ứng với K mô hình được huấn luyện ở mỗi vòng lặp.
- Trong mỗi vòng lặp, một phần dữ liệu (1 fold) được lấy ra để sử dụng cho việc kiểm tra mô hình, $K-1$ phần ($K-1$ fold) dữ liệu còn lại sẽ được đưa vào mô hình để huấn luyện. Quá trình này lặp lại cho đến hết K vòng lặp.

Các chỉ số đánh giá của K fold (accuracy, precision, recall, ...) sẽ là trung bình K chỉ số của K mô hình.

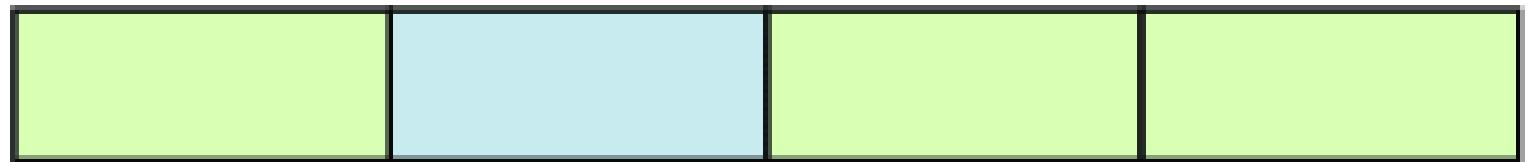
VD: $K = 10$ thì accuracy của 10 fold sẽ là trung bình accuracy của 10 mô hình.

4.3 Đánh giá mô hình với K fold

Fold 1 Fold 2 Fold 3 Fold 4



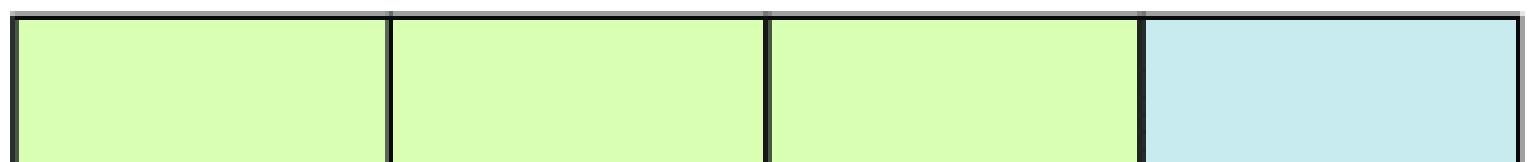
Run 1



Run 2



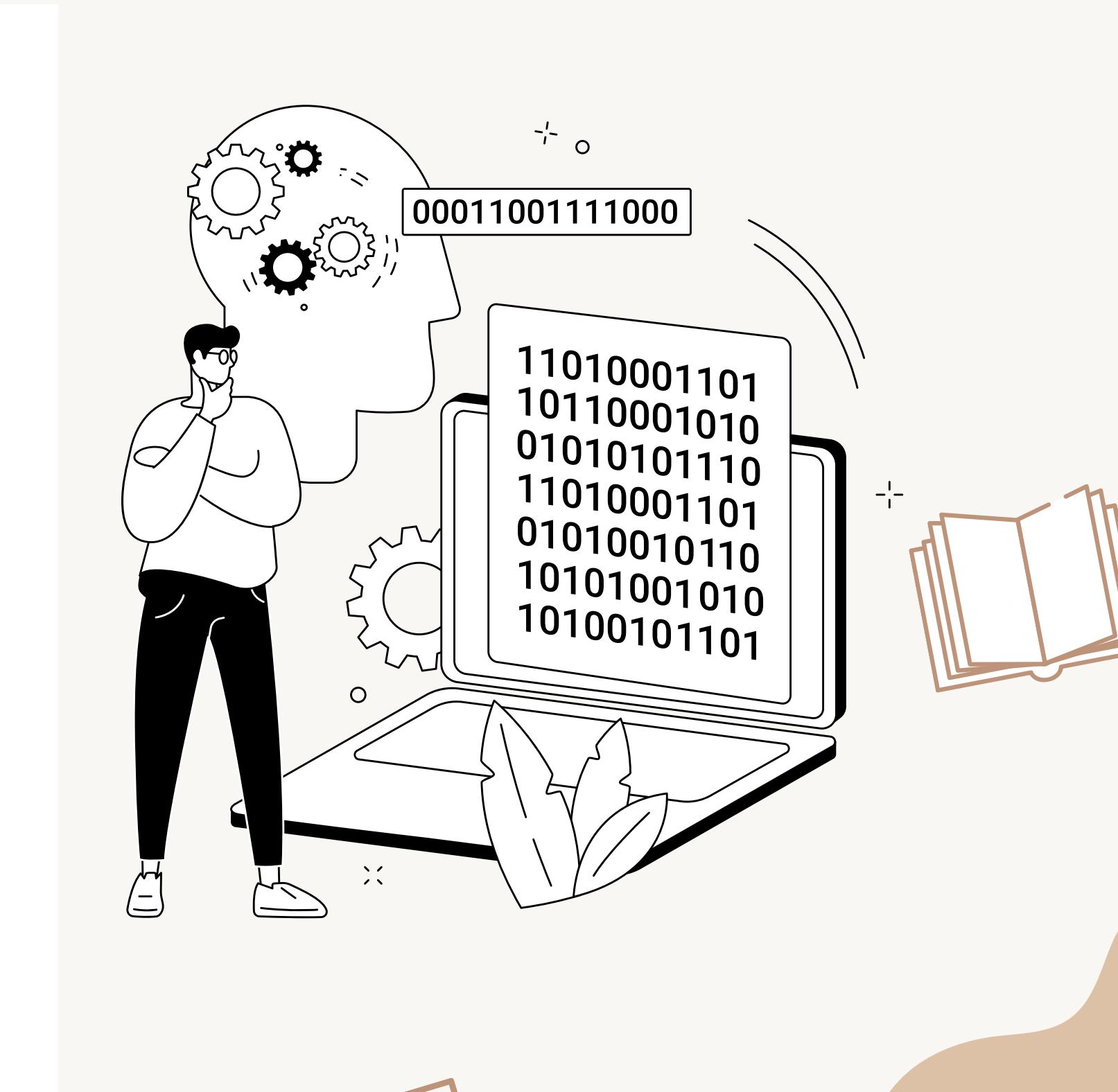
Run 3



Run 4

training set

test set



4.3 Đánh giá mô hình với K fold

```
KF = KFold(n_splits = 10, shuffle=True, random_state=1)
```

```
model_eval_k = []
model_name = ['KNN', 'Logistic Regression', 'Random Forest', 'Naive Bayes', "Neural Network", "SVC"]
i = 0

scoring = ['accuracy', 'precision', 'recall', 'f1']

for model in list_model:
    eval_dict = {}
    eval_dict['Model name (kFold)'] = model_name[i]
    i += 1

    # Xây dựng mô hình
    results = cross_validate(estimator=model, X=X, y=Y, cv=KF, scoring=scoring)

    # Model accuracy
    eval_dict['Accuracy (%)'] = round(results['test_accuracy'].mean(), 4)*100

    # Model precision
    eval_dict['Precision'] = round(results['test_precision'].mean(), 2)
```

4.3 Đánh giá mô hình với K fold

```
# Model recall
eval_dict['Recall'] = round(results['test_recall'].mean(), 2)

# Model F1 score
eval_dict['F1-score'] = round(results['test_f1'].mean(), 2)

# Running time
end_time = datetime.now()
eval_dict['Running time'] = '{}'.format(end_time - start_time)

model_eval_k.append(eval_dict)

model_eval_k = pd.DataFrame(model_eval_k)
```

4.3 Đánh giá mô hình với K fold

Model name (kFold)	Accuracy (%)	Precision	Recall	F1-score	Running time
Random Forest	93.04	0.97	0.90	0.93	0:00:46.820851
KNN	91.78	0.98	0.87	0.92	0:00:33.171073
Neural Network	85.89	0.89	0.84	0.87	0:07:57.673765
SVC	83.25	0.83	0.86	0.85	0:28:11.029780
Logistic Regression	83.20	0.83	0.86	0.85	0:00:01.177424
Naive Bayes	82.88	0.85	0.84	0.84	0:00:00.505614

4.3 So sánh K fold với train test split thông thường

Model name	Dif_Accuracy	Dif_Precision	Dif_Recall	Dif_F1-score
KNN	0.000000	0.010000	-0.010000	0.000000
Logistic Regression	0.160000	0.000000	0.000000	0.000000
Random Forest	0.170000	-0.010000	0.010000	0.000000
Naive Bayes	0.090000	0.000000	0.010000	0.000000
Neural Network	-0.200000	-0.020000	0.010000	0.000000
SVC	0.220000	-0.010000	0.010000	0.000000

4.3 So sánh K fold với train test split thông thường

Ta sẽ lựa chọn dựa trên kết quả đánh giá khi chạy K fold

Model name (kFold)	Accuracy (%)	Precision	Recall	F1-score	Running time
Random Forest	93.04	0.97	0.90	0.93	0:00:46.820851
KNN	91.78	0.98	0.87	0.92	0:00:33.171073
Neural Network	85.89	0.89	0.84	0.87	0:07:57.673765
SVC	83.25	0.83	0.86	0.85	0:28:11.029780
Logistic Regression	83.20	0.83	0.86	0.85	0:00:01.177424
Naive Bayes	82.88	0.85	0.84	0.84	0:00:00.505614

4.4 Lựa chọn mô hình tốt nhất

Best model

- Xét theo Accuracy, Random Forest cao nhất (93.04%) và KNN cao thứ 2 (91.78%)
- Xét theo Precision, KNN cao nhất (0.98) và Random Forest cao thứ 2 (0.97)
- Xét theo Recall, Random Forest cao nhất (0.90) và KNN cao thứ 2 (0.87)
- Xét theo F1-score, Random Forest cao nhất (0.93) và KNN cao thứ 2 (0.92)
- Xét theo thời gian cho ra kết quả, ngoại trừ Neural Network và SVC các mô hình còn lại đều có thời gian chạy dưới 1 phút

