Chapter 7 - Ex1: Hacide - Full

- Cho 2 tập tin là hacide_train.csv và hacide_test.csv. Dữ liệu này được dùng để xây dựng model dự đoán và kiểm tra một mẫu là hiếm hay phổ biến.
- Đọc dữ liệu hacide_train.csv, xem xét tính cân bằng giữa hai loại mẫu hiếm và phổ biến. Trực quan hóa. Nhận xét.
- Nếu 2 loại mẫu này không cân bằng, hãy chọn một phương pháp cân bằng dữ liệu và thực hiện. Giải thích lý do. Trực quan hóa kết quả.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
train_data = pd.read_csv("hacide_train.csv")
```

In [3]:
```
train_data.info()
```

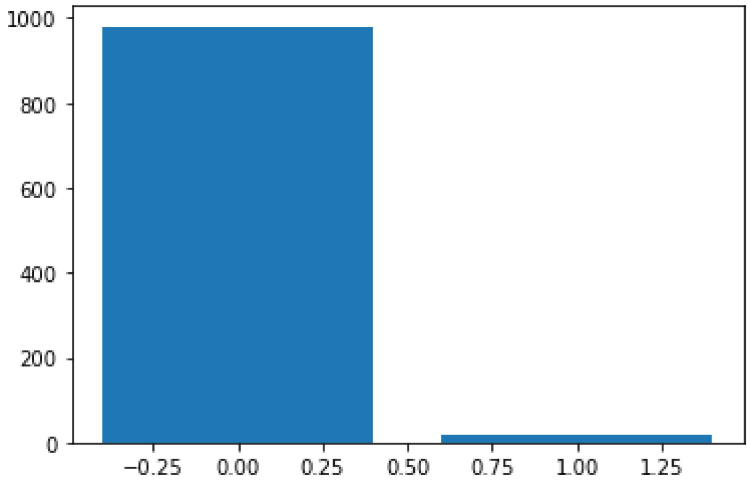
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 4 columns):
Unnamed: 0    1000 non-null int64
cls           1000 non-null int64
x1            1000 non-null float64
x2            1000 non-null float64
dtypes: float64(2), int64(2)
memory usage: 31.4 KB
```

In [4]:
```
# Đếm theo loại: hiếm, phổ biến
occ = train_data.cls.value_counts()
occ
```

Out[4]:
```
0    980
1     20
Name: cls, dtype: int64
```

In [5]:
```
plt.bar(occ.index.values, occ.values)
```

Out[5]:
```
<BarContainer object of 2 artists>
```

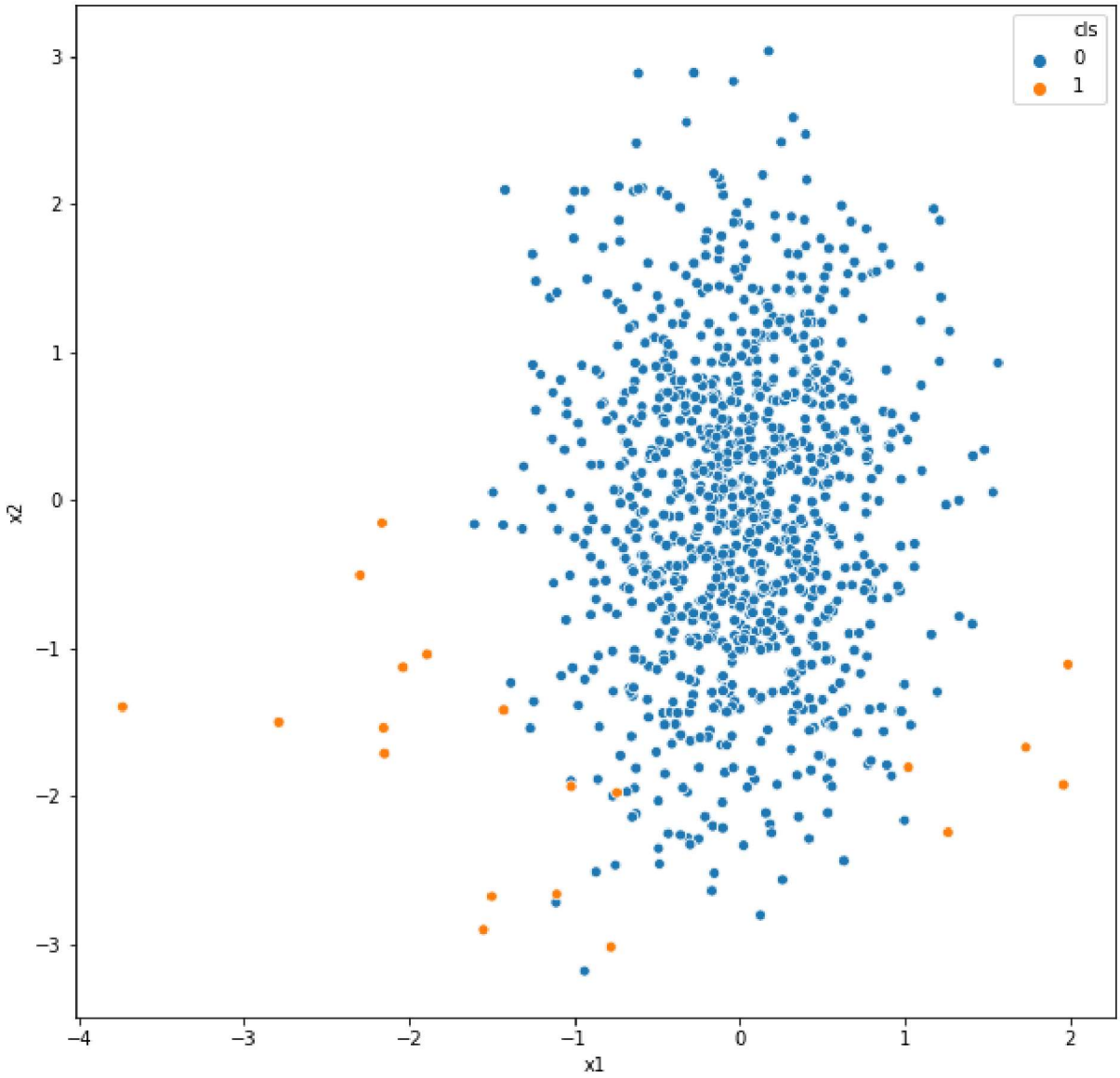


In [6]:
```
# Print the ratio of fraud cases
print(occ / len(train_data.index))
```

```
0    0.98
1    0.02
Name: cls, dtype: float64
```

In [7]:
```
# Vì lượng dữ liệu class 1 rất ít => do đó ta sẽ áp dụng Oversampling để nâng số mẫu của nhóm hiếm
```

In [8]:
```
plt.figure(figsize=(10,10))
sns.scatterplot(data=train_data, x="x1", y="x2", hue="cls")
plt.show()
```



In [9]:
```
# chia ra 2 tập: X (input), y (output)
X = train_data[["x1", "x2"]]
X.head()
```

Out[9]:

	x1	x2
0	0.200798	0.678038
1	0.016620	1.576558
2	0.228725	-0.559534
3	0.126379	-0.093814
4	0.600821	-0.298395

In [10]:
```
y = train_data["cls"]
y.head()
```

Out[10]:
```
0    0
1    0
2    0
3    0
```

```
4    0
Name: cls, dtype: int64
```

Áp dụng thuật toán với dữ liệu gốc

In [19]:
```
from sklearn.linear_model import LogisticRegression
```

In [20]:
```
model = LogisticRegression()
```

In [21]:
```
model.fit(X, y)
```

Out[21]:
```
LogisticRegression()
```

In [22]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
```

In [23]:
```
y_pred = model.predict(X)
```

In [24]:
```
accuracy_score(y, y_pred)
```

Out[24]:
```
0.983
```

In [25]:
```
cm = confusion_matrix(y, y_pred)
```

In [26]:
```
cm
```

Out[26]:
```
array([[979,   1],
       [ 16,   4]], dtype=int64)
```

In [33]:
```
# Đánh giá model
from sklearn. metrics import classification_report, roc_auc_score, roc_curve
```

In [34]:
```
print(classification_report(y, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       980
           1       0.80      0.20      0.32        20

    accuracy                           0.98      1000
   macro avg       0.89      0.60      0.66      1000
weighted avg       0.98      0.98      0.98      1000
```

In [35]:
```
y_prob = model.predict_proba(X)
y_prob
```

Out[35]:
```
array([[9.99086926e-01, 9.13073899e-04],
       [9.99680383e-01, 3.19617276e-04],
       [9.94613861e-01, 5.38613886e-03],
       ...,
```

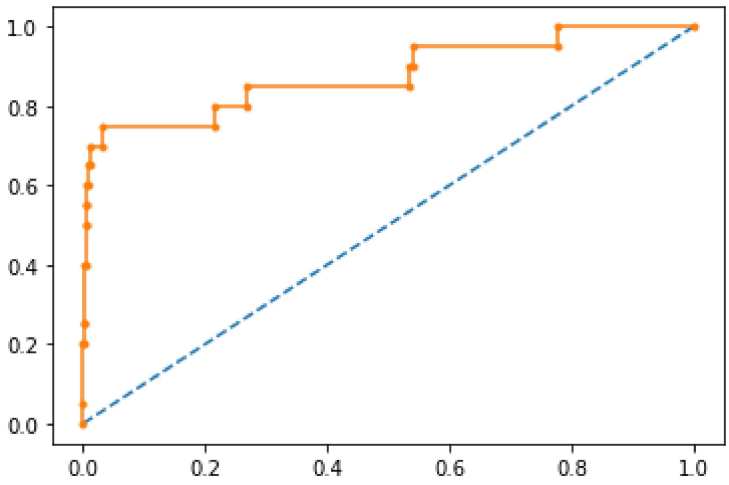
```
       [3.00672836e-01, 6.99327164e-01],
       [9.96889354e-01, 3.11064642e-03],
       [9.97002184e-01, 2.99781558e-03]])
```

In [36]:
```
roc_auc_score(y, y_prob[:, 1])
```

Out[36]:
```
0.879234693877551
```

In [37]:
```
import matplotlib.pyplot as plt
```

In [38]:
```
# calculate roc curve
fpr, tpr, thresholds = roc_curve(y, y_prob[:, 1])
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr, tpr, marker='.')
plt.show()
```



Kết luận

- ROC_AUC cao
- precision class 1 cao nhưng recall thấp

Oversampling

In [16]:
```
from imblearn.over_sampling import SMOTE
```

In [17]:
```
X_S, y_S = SMOTE().fit_resample(X, y)
```

```
c:\program files\python36\lib\site-packages\sklearn\utils\deprecation.py:86: FutureWarning: Functi
on safe_indexing is deprecated; safe_indexing is deprecated in version 0.22 and will be removed in
version 0.24.
  warnings.warn(msg, category=FutureWarning)
```

In [13]:
```
from collections import Counter
sorted(Counter(y_S).items())
```

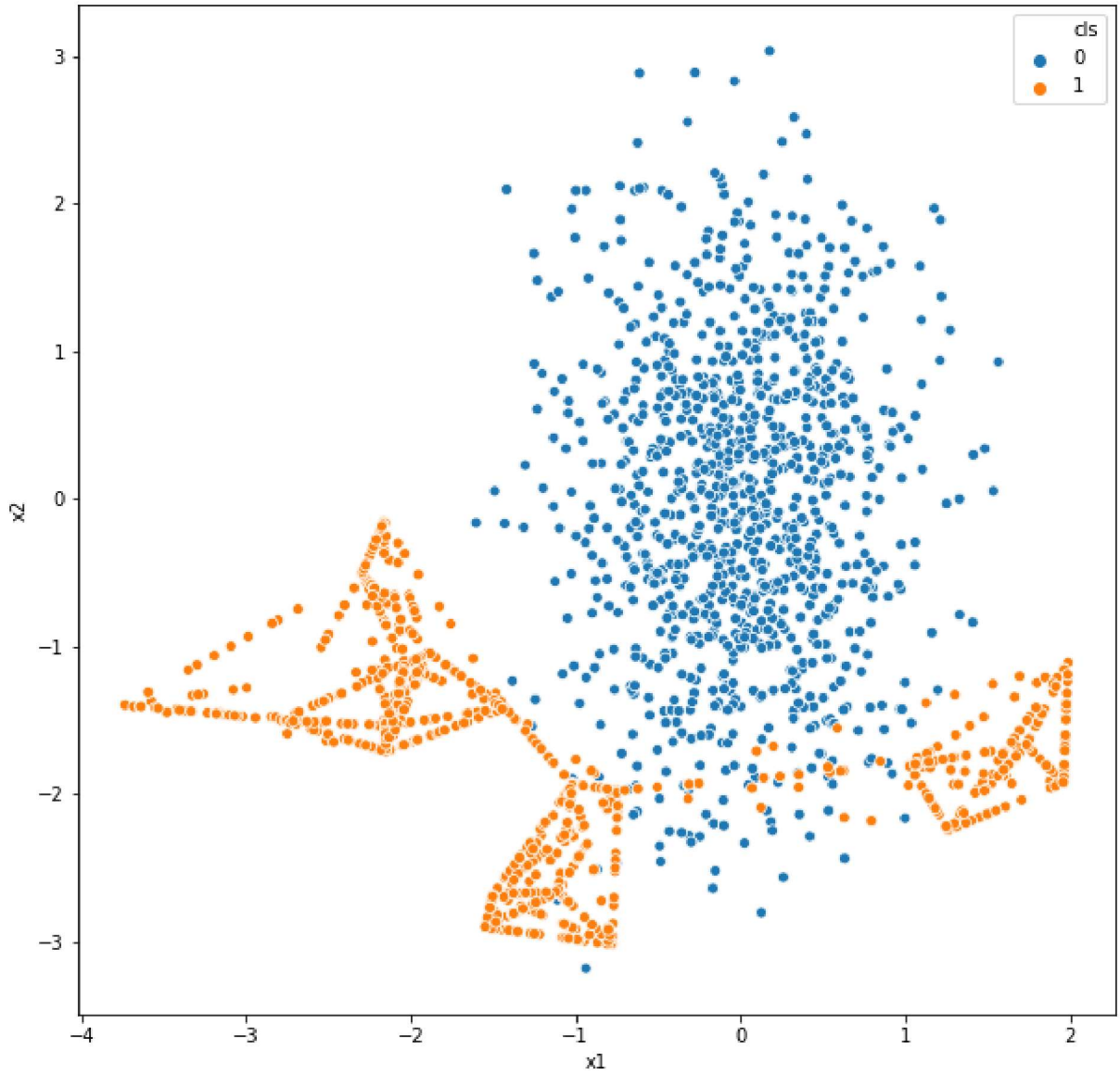
Out[13]:
```
[(0, 980), (1, 980)]
```

In [14]:
```
data_S= pd.DataFrame(X_S)
data_S.columns = ["x1", "x2"]
data_S['cls'] = y_S
data_S.head()
```

Out[14]:

	x1	x2	cls
0	0.200798	0.678038	0
1	0.016620	1.576558	0
2	0.228725	-0.559534	0
3	0.126379	-0.093814	0
4	0.600821	-0.298395	0

In [15]:
```
plt.figure(figsize=(10,10))
sns.scatterplot(data=data_S, x="x1", y="x2", hue="cls")
plt.show()
```



Áp dụng thuật toán với dữ liệu Oversampling

In [27]:
```
model_o = LogisticRegression()
```

In [28]:
```
model_o.fit(X_S, y_S)
```

Out[28]:
```
LogisticRegression()
```

In [29]:
```
y_pred_o = model.predict(X_S)
```

In [30]:
```
accuracy_score(y_S, y_pred_o)
```

Out[30]:
```
0.6030612244897959
```

In [31]:
```
cm = confusion_matrix(y_S, y_pred_o)
```

In [32]:
```
cm
```

Out[32]:
```
array([[979,   1],
       [777, 203]], dtype=int64)
```

In [39]:
```
print(classification_report(y_S, y_pred_o))
```

```
              precision    recall  f1-score   support

           0       0.56      1.00      0.72       980
           1       1.00      0.21      0.34       980

    accuracy                           0.60      1960
   macro avg       0.78      0.60      0.53      1960
weighted avg       0.78      0.60      0.53      1960
```

In [40]:
```
y_prob_o = model.predict_proba(X_S)
y_prob_o
```

Out[40]:
```
array([[9.99086926e-01, 9.13073899e-04],
       [9.99680383e-01, 3.19617276e-04],
       [9.94613861e-01, 5.38613886e-03],
       ...,
       [5.31296998e-01, 4.68703002e-01],
       [3.99010701e-01, 6.00989299e-01],
       [7.65971678e-01, 2.34028322e-01]])
```

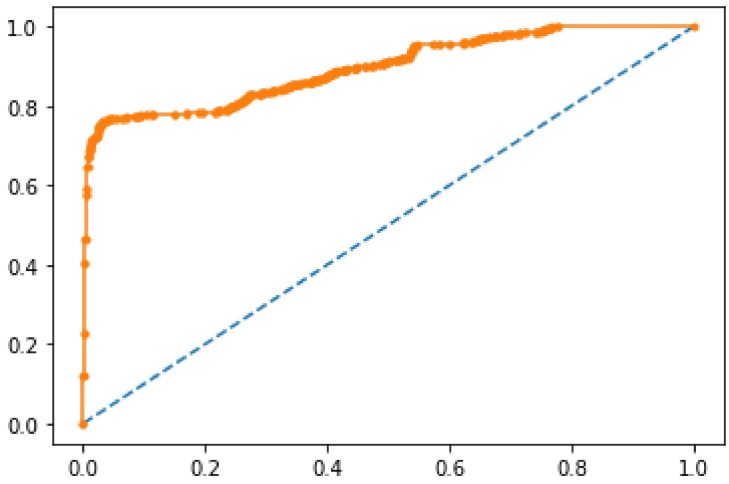
In [41]:
```
roc_auc_score(y_S, y_prob_o[:, 1])
```

Out[41]:
```
0.8968929612661392
```

In [37]:
```
import matplotlib.pyplot as plt
```

In [42]:
```
# calculate roc curve
fpr, tpr, thresholds = roc_curve(y_S, y_prob_o[:, 1])
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
```

```
plt.plot(fpr, tpr, marker='.')
plt.show()
```



Kết luận

- ROC_AUC cao hơn một chút so với dữ liệu gốc
- precision class 1 cao nhưng recall vẫn thấp
- Thêm vào đó class 0 precsion giảm đi rất nhiều #### Việc áp dụng không đạt được kết quả như mong muốn. Không áp dụng #### => Xem xét việc lựa chọn 1 thuật toán khác

Chọn thuật toán khác và làm việc với dữ liệu gốc

In [43]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [44]:
```
model_n = DecisionTreeClassifier()
```

In [45]:
```
model_n.fit(X, y)
```

Out[45]:
```
DecisionTreeClassifier()
```

In [46]:
```
y_pred_n = model_n.predict(X)
```

In [47]:
```
accuracy_score(y, y_pred_n)
```

Out[47]:
```
1.0
```

In [48]:
```
cm = confusion_matrix(y, y_pred_n)
```

In [49]:
```
cm
```

Out[49]:
```
array([[980,   0],
       [  0,  20]], dtype=int64)
```

In [50]:

```
print(classification_report(y, y_pred_n))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       980
           1       1.00      1.00      1.00        20

    accuracy                           1.00      1000
   macro avg       1.00      1.00      1.00      1000
weighted avg       1.00      1.00      1.00      1000
```

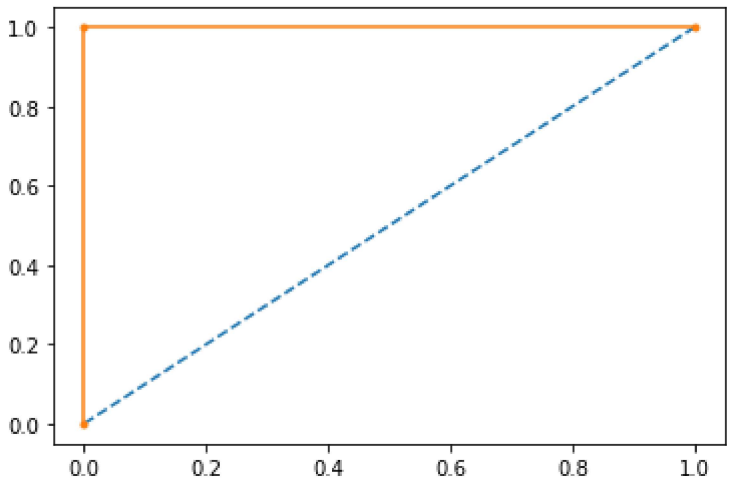
In [55]:
```
y_prob_n = model_n.predict_proba(X)
y_prob_n
```

Out[55]:
```
array([[1., 0.],
       [1., 0.],
       [1., 0.],
       ...,
       [0., 1.],
       [0., 1.],
       [0., 1.]])
```

In [56]:
```
roc_auc_score(y, y_prob_n[:, 1])
```

Out[56]:
```
1.0
```

In [57]:
```
# calculate roc curve
fpr, tpr, thresholds = roc_curve(y, y_prob_n[:, 1])
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr, tpr, marker='.')
plt.show()
```



Kết luận

- Với việc áp dụng DecisionTree model, chúng ta không cần resample data mà kết quả tốt hơn nhiều.
- Có thể sử dụng model DecisionTree cho bộ dữ liệu này thay cho LogisticRegression

In [ ]: