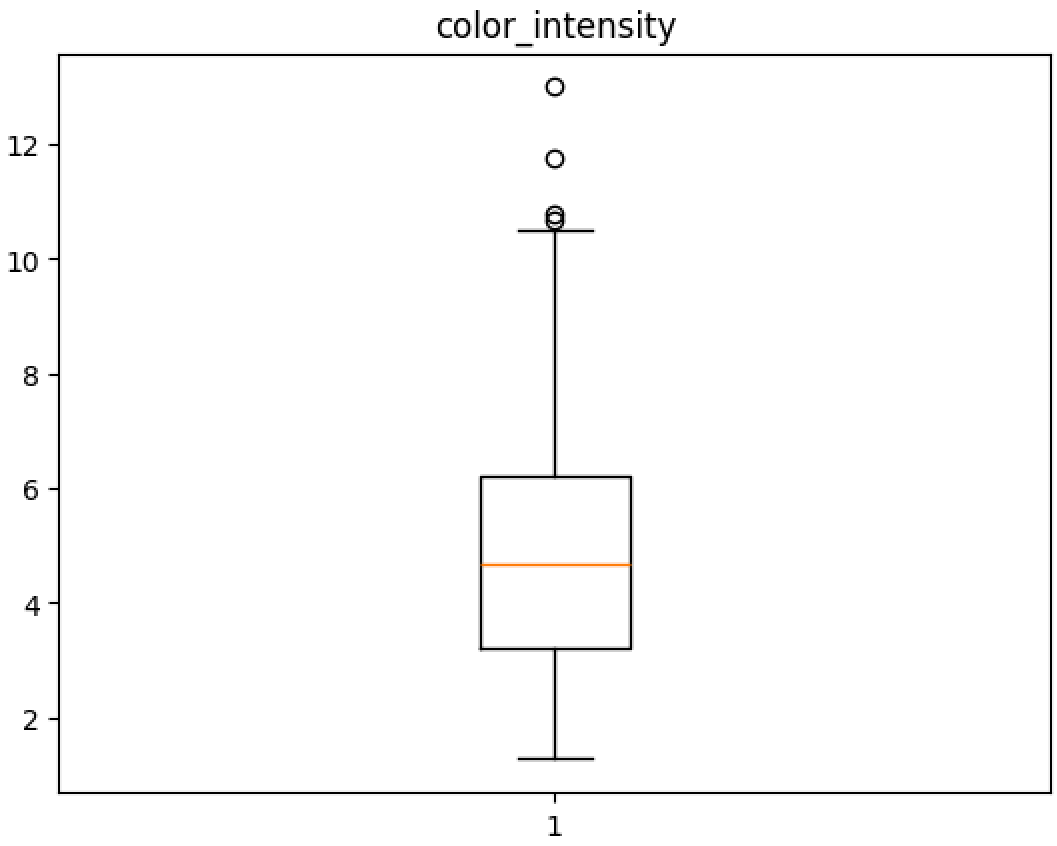
In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_wine
wine_load=load_wine()
wine=pd.DataFrame(wine_load.data,columns=wine_load.feature_names)
wine['class']=wine_load.target
wine['class']=wine['class'].map({0:'class0',1:'class1',2:'class2'})
plt.boxplot(wine['color_intensity'],whis=1.5)
plt.title('color_intensity')
plt.show()
```



In [2]:

```
import numpy as np
def outlier_iqr(df,col):
    quartile_1,quartile_3= np.percentile(df[col],[25,75])
    iqr=quartile_3-quartile_1
    lower_whis=quartile_1-1.5*iqr
    upper_whis=quartile_3+1.5*iqr
    outliers=df[(df[col]>upper_whis)|(df[col]<lower_whis)]
    return outliers[[col]]


outliers=outlier_iqr(wine,'color_intensity')

print(outliers)
```

```
     color_intensity
151            10.80
158            13.00
159            11.75
166            10.68
```

In [3]:

```
#이상치 제거 ,

drop_outliers=wine.drop(index=outliers.index)
print('Original',wine.shape)
print('Drop Outlier',drop_outliers.shape)
```

```
Original (178, 14)
Drop Outlier (174, 14)
```

In [4]:

```
#이상치 대체
wine.loc[outliers.index,'color_intensity']=np.NaN
wine['color_intensity'].fillna(wine['color_intensity'].mean(),inplace=True)
print(wine.loc[outliers.index,['color_intensity']])
```

```
     color_intensity
151         4.908678
158         4.908678
159         4.908678
166         4.908678
```

```
/var/folders/hv/lqp1gn9n1ll0lbh2pfzn9pww0000gn/T/ipykernel_4821/3601741771.py:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  wine['color_intensity'].fillna(wine['color_intensity'].mean(),inplace=True)
```

In [5]:

```
import pandas as pd
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
```

```
iris=load_iris()
iris=pd.DataFrame(iris.data,columns=iris.feature_names)
iris['class']=load_iris().target
iris['class']=iris['class'].map({0:'Setosa',1:'Versicolour',2:'Virginica'})

from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test = train_test_split(iris.drop(columns='class'),iris['class'],test_size=0.2,random_state=1004)
print(X_train.shape,X_test.shape,Y_train.shape,Y_test.shape)
```

```
(120, 4) (30, 4) (120,) (30,)
```

In [6]:

```
X_train
```

Out[6]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
87	6.3	2.3	4.4	1.3
67	5.8	2.7	4.1	1.0
131	7.9	3.8	6.4	2.0
74	6.4	2.9	4.3	1.3
63	6.1	2.9	4.7	1.4
...	...	...	...	...
14	5.8	4.0	1.2	0.2
69	5.6	2.5	3.9	1.1
31	5.4	3.4	1.5	0.4
11	4.8	3.4	1.6	0.2
2	4.7	3.2	1.3	0.2

120 rows × 4 columns

In [7]:

```
Y_train.value_counts()
```

Out[7]:

```
class
Versicolour    41
Setosa         40
Virginica      39
Name: count, dtype: int64
```

In [8]:

```
X_train,X_test,Y_train,Y_test = train_test_split(iris.drop(columns='class'),iris['class'],test_size=0.2,random_state=1004,strat

Y_train.value_counts()
```

Out[8]:

```
class
Versicolour    40
Virginica      40
Setosa         40
Name: count, dtype: int64
```

In [ ]:

```
!pip install imbalanced-learn
!pip uninstall pandas-profiling -y
!pip install --upgrade pip
!pip install scikit-learn==1.6.1
```

In [39]:

```
import numpy as np
import pandas as pd
from sklearn.datasets import make_classification
from collections import Counter
from imblearn.under_sampling import RandomUnderSampler

x,y=make_classification(n_samples=2000, n_features=6, weights=[0.95], flip_y=0)
print(Counter(y))
```

```
Counter({0: 1900, 1: 100})
```

In [40]:

```
undersample=RandomUnderSampler(sampling_strategy='majority')
x_under,y_under=undersample.fit_resample(x,y)
print(Counter(y_under))
undersample=RandomUnderSampler(sampling_strategy=0.5)
x_under,y_under=undersample.fit_resample(x,y)
print(Counter(y_under))
```

```
Counter({0: 100, 1: 100})
Counter({0: 200, 1: 100})
```

In [41]:

```
from imblearn.over_sampling import RandomOverSampler

oversample = RandomOverSampler(sampling_strategy=0.5)
x_over,y_over= oversample.fit_resample(x,y)
print(Counter(y_over))

oversample = RandomOverSampler(sampling_strategy='minority')
x_over,y_over= oversample.fit_resample(x,y)
print(Counter(y_over))
```

```
Counter({0: 1900, 1: 950})
Counter({0: 1900, 1: 1900})
```

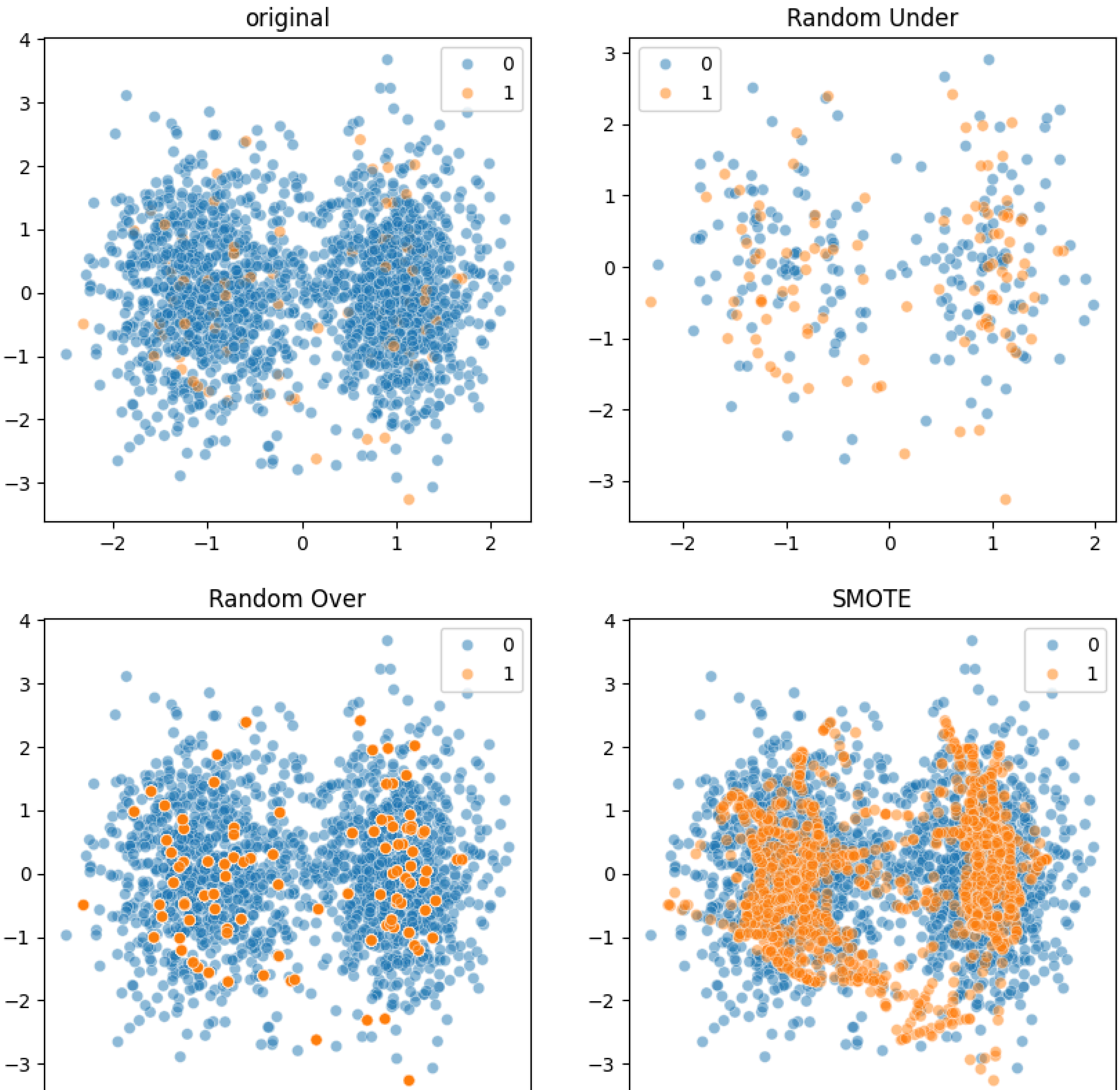
In [42]:

```python
from imblearn.over_sampling import SMOTE

smote=SMOTE(sampling_strategy='minority')
x_sm,y_sm= smote.fit_resample(x,y)
print(Counter(y_sm))
```

```
Counter({0: 1900, 1: 1900})
```

In [45]:

```python
import matplotlib.pyplot as plt
import seaborn as sns

fig,ax= plt.subplots(nrows=2,ncols=2,figsize=(10,10))
sns.scatterplot(x=x[:,1],y=x[:,2],hue=y,ax=ax[0][0],alpha=0.5)
sns.scatterplot(x=x_under[:,1],y=x_under[:,2],hue=y_under,ax=ax[0][1],alpha=0.5)
sns.scatterplot(x=x_over[:,1],y=x_over[:,2],hue=y_over,ax=ax[1][0],alpha=0.5)
sns.scatterplot(x=x_sm[:,1],y=x_sm[:,2],hue=y_sm,ax=ax[1][1],alpha=0.5)
ax[0][0].set_title('original')
ax[0][1].set_title('Random Under')
ax[1][0].set_title('Random Over')
ax[1][1].set_title('SMOTE')
plt.show()
```


original
Random Under
Random Over
SMOTE
0
1
−3
−2
−1
0
1
2
3
4
−2
−1
0
1
2

In [ ]:

```
```