

```
In [256]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [257]: d=pd.read_csv('https://github.com/YBI-Foundation/Dataset/raw/main/Bank%20Churn%20Dataset.csv')
```

```
In [258]: d.head()
```

Out[258]:

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	Num Of Products	Has Credit Card
0	15634602	Hargrave	619	France	Female	42	2	0.00	1	0
1	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0
2	15619304	Onio	502	France	Female	42	8	159660.80	3	0
3	15701354	Boni	699	France	Female	39	1	0.00	2	0
4	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	0

```
In [259]: d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   CustomerId      10000 non-null   int64  
 1   Surname          10000 non-null   object  
 2   CreditScore     10000 non-null   int64  
 3   Geography        10000 non-null   object  
 4   Gender           10000 non-null   object  
 5   Age              10000 non-null   int64  
 6   Tenure           10000 non-null   int64  
 7   Balance          10000 non-null   float64 
 8   Num Of Products 10000 non-null   int64  
 9   Has Credit Card 10000 non-null   int64  
 10  Is Active Member 10000 non-null   int64  
 11  Estimated Salary 10000 non-null   float64 
 12  Churn            10000 non-null   int64  
dtypes: float64(2), int64(8), object(3)
memory usage: 1015.8+ KB
```

```
In [260]: d.duplicated('CustomerId').sum()
```

Out[260]: 0

```
In [261]: d=d.set_index('CustomerId')
```

In [262]: `d.info()`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10000 entries, 15634602 to 15628319
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Surname          10000 non-null   object  
 1   CreditScore      10000 non-null   int64  
 2   Geography         10000 non-null   object  
 3   Gender            10000 non-null   object  
 4   Age               10000 non-null   int64  
 5   Tenure            10000 non-null   int64  
 6   Balance           10000 non-null   float64 
 7   Num Of Products  10000 non-null   int64  
 8   Has Credit Card  10000 non-null   int64  
 9   Is Active Member 10000 non-null   int64  
 10  Estimated Salary 10000 non-null   float64 
 11  Churn             10000 non-null   int64  
dtypes: float64(2), int64(7), object(3)
memory usage: 1015.6+ KB
```

Encoding

In [263]: `d['Geography'].value_counts()`

Out[263]:

France	5014
Germany	2509
Spain	2477

Name: Geography, dtype: int64

In [264]: `d.replace({'Geography': {'France':2, 'Germany':1, 'Spain':0}}, inplace=True)`

In [265]: `d['Gender'].value_counts()`

Out[265]:

Male	5457
Female	4543

Name: Gender, dtype: int64

In [266]: `d.replace({'Gender': {'Male':0, 'Female':1}}, inplace=True)`

In [267]: `d['Num Of Products'].value_counts()`

Out[267]:

1	5084
2	4590
3	266
4	60

Name: Num Of Products, dtype: int64

```
In [268]: d.replace({'Num Of Products': {1:0,2:1,3:1,4:1}},inplace=True)
```

```
In [269]: d['Has Credit Card'].value_counts()
```

```
Out[269]: 1    7055  
0    2945  
Name: Has Credit Card, dtype: int64
```

```
In [270]: d['Is Active Member'].value_counts()
```

```
Out[270]: 1    5151  
0    4849  
Name: Is Active Member, dtype: int64
```

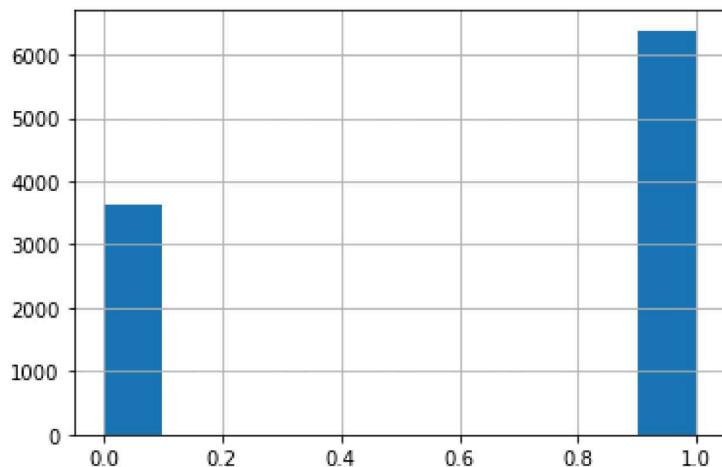
```
In [271]: d.loc[(d['Balance']==0),'Churn'].value_counts()
```

```
Out[271]: 0    3117  
1    500  
Name: Churn, dtype: int64
```

```
In [272]: d['Zero Balance']=np.where(d['Balance']>0,1,0)
```

```
In [273]: d['Zero Balance'].hist()
```

```
Out[273]: <AxesSubplot:>
```



In [274]: `d.groupby(['Churn', 'Geography']).count()`

Out[274]:

		Surname	CreditScore	Gender	Age	Tenure	Balance	Num Of Products	Has Credit Card	Act Mem
Churn	Geography									
0	0	2064	2064	2064	2064	2064	2064	2064	2064	2
		1695	1695	1695	1695	1695	1695	1695	1695	1
		4204	4204	4204	4204	4204	4204	4204	4204	4
	1	413	413	413	413	413	413	413	413	.
		814	814	814	814	814	814	814	814	814
		810	810	810	810	810	810	810	810	810

In [275]: `d.columns`

Out[275]: `Index(['Surname', 'CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance', 'Num Of Products', 'Has Credit Card', 'Is Active Member', 'Estimated Salary', 'Churn', 'Zero Balance'], dtype='object')`

In [276]: `x=d.drop(['Surname', 'Churn'], axis=1)`

In [277]: `y=d['Churn']`

In [278]: `x.shape, y.shape`

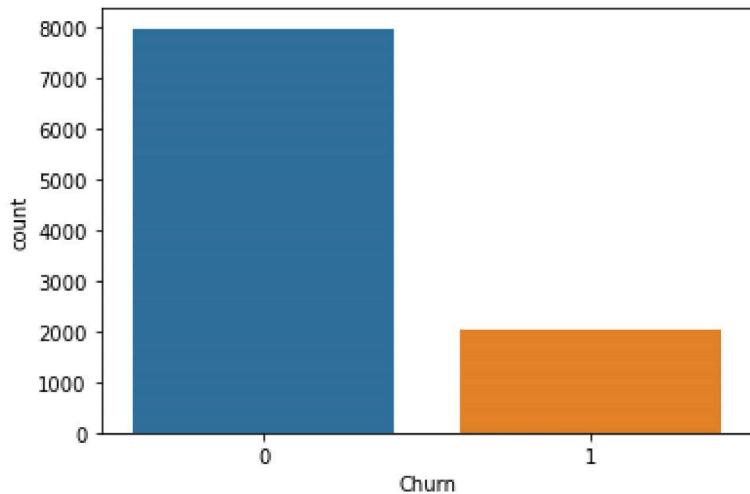
Out[278]: `((10000, 11), (10000,))`

In [279]: `d['Churn'].value_counts()`

Out[279]: `0 7963
1 2037
Name: Churn, dtype: int64`

```
In [280]: sns.countplot(x='Churn', data=d)
```

```
Out[280]: <AxesSubplot:xlabel='Churn', ylabel='count'>
```



```
In [281]: x.shape,y.shape
```

```
Out[281]: ((10000, 11), (10000,))
```

Random Under sampling

```
In [282]: !pip install imblearn
```

```
Requirement already satisfied: imblearn in c:\users\g.ravi prakash reddy\anaconda3\lib\site-packages (0.0)
Requirement already satisfied: imbalanced-learn in c:\users\g.ravi prakash reddy\anaconda3\lib\site-packages (from imblearn) (0.9.1)
Requirement already satisfied: joblib>=1.0.0 in c:\users\g.ravi prakash reddy\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.1.0)
Requirement already satisfied: scikit-learn>=1.1.0 in c:\users\g.ravi prakash reddy\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.1.0)
Requirement already satisfied: scipy>=1.3.2 in c:\users\g.ravi prakash reddy\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.7.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\g.ravi prakash reddy\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (2.2.0)
Requirement already satisfied: numpy>=1.17.3 in c:\users\g.ravi prakash reddy\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.20.3)
```

```
In [283]: from imblearn.under_sampling import RandomUnderSampler
```

```
In [284]: r=RandomUnderSampler(random_state=2529)
```

```
In [285]: x_r,y_r=r.fit_resample(x,y)
```

```
In [286]: x_r.shape,y_r.shape,x.shape,y.shape
```

```
Out[286]: ((4074, 11), (4074,), (10000, 11), (10000,))
```

```
In [287]: y.value_counts()
```

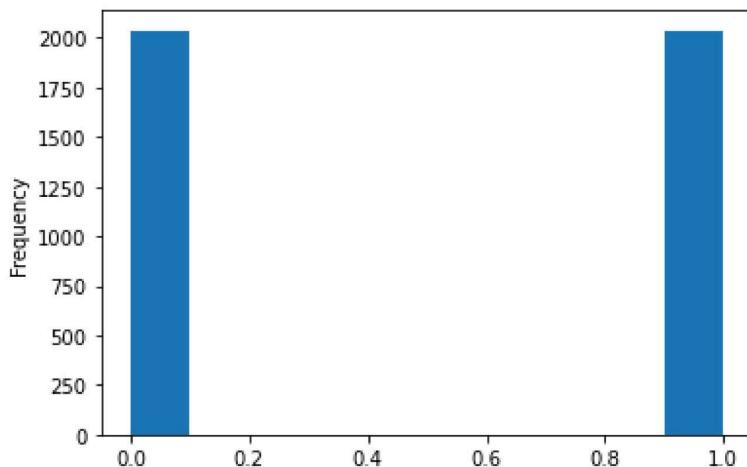
```
Out[287]: 0    7963  
1    2037  
Name: Churn, dtype: int64
```

```
In [288]: y_r.value_counts()
```

```
Out[288]: 0    2037  
1    2037  
Name: Churn, dtype: int64
```

```
In [289]: y_r.plot(kind='hist')
```

```
Out[289]: <AxesSubplot:ylabel='Frequency'>
```



```
In [290]: #RANDOM OVER SAMPLING  
from imblearn.over_sampling import RandomOverSampler
```

```
In [291]: ros=RandomOverSampler(random_state=2529)
```

```
In [292]: x_ros,y_ros=ros.fit_resample(x,y)
```

```
In [293]: x_ros.shape,y_ros.shape,x.shape,y.shape
```

```
Out[293]: ((15926, 11), (15926,), (10000, 11), (10000,))
```

```
In [294]: y.value_counts()
```

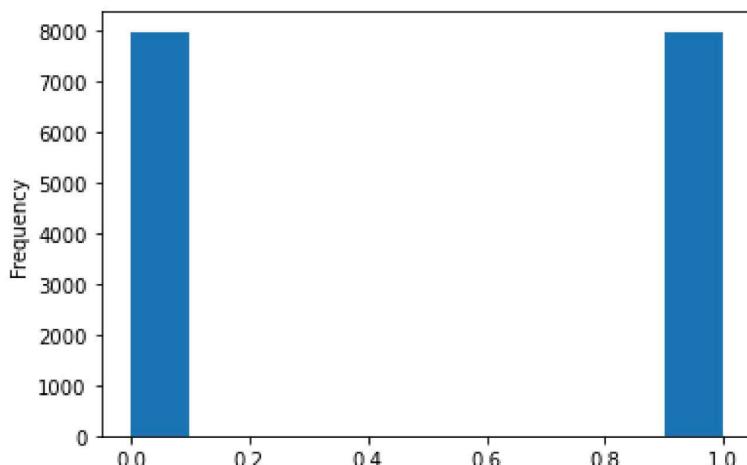
```
Out[294]: 0    7963  
1    2037  
Name: Churn, dtype: int64
```

```
In [295]: y_ros.value_counts()
```

```
Out[295]: 1    7963  
0    7963  
Name: Churn, dtype: int64
```

```
In [296]: y_ros.plot(kind='hist')
```

```
Out[296]: <AxesSubplot:ylabel='Frequency'>
```



Train Test Split

```
In [297]: from sklearn.model_selection import train_test_split
```

```
In [298]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=25)
```

```
In [299]: x_train_r,x_test_r,y_train_r,y_test_r=train_test_split(x_r,y_r,test_size=0.3,rand
```

```
In [300]: x_train_ros,x_test_ros,y_train_ros,y_test_ros=train_test_split(x_ros,y_ros,test_s
```

```
In [301]: from sklearn.preprocessing import StandardScaler
```

```
In [302]: sc=StandardScaler()
```

```
In [303]: x_train[['CreditScore','Age','Tenure','Balance','Estimated Salary']] = sc.fit_transform(x_train)
x_test[['CreditScore','Age','Tenure','Balance','Estimated Salary']] = sc.fit_transform(x_test)
```

```
In [304]: x_train_r[['CreditScore','Age','Tenure','Balance','Estimated Salary']] = sc.fit_transform(x_train)
x_test_r[['CreditScore','Age','Tenure','Balance','Estimated Salary']] = sc.fit_transform(x_test)
```

```
In [305]: x_train_rus[['CreditScore','Age','Tenure','Balance','Estimated Salary']] = sc.fit_transform(x_train)
x_test_rus[['CreditScore','Age','Tenure','Balance','Estimated Salary']] = sc.fit_transform(x_test)
```

```
In [306]: from sklearn.svm import SVC
svc=SVC()
```

```
In [307]: svc.fit(x_train,y_train)
```

```
Out[307]:
```



```
In [308]: y_pred=svc.predict(x_test)
```

Model accuracy

```
In [309]: from sklearn.metrics import confusion_matrix,classification_report
confusion_matrix(y_test,y_pred)
```

```
Out[309]: array([[2374,    45],
       [ 421,  160]], dtype=int64)
```

```
In [310]: print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.85	0.98	0.91	2419
1	0.78	0.28	0.41	581
accuracy			0.84	3000
macro avg	0.81	0.63	0.66	3000
weighted avg	0.84	0.84	0.81	3000

Hyperparameter Tuning

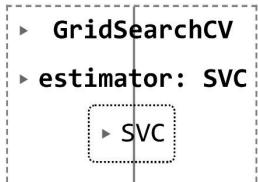
```
In [311]: from sklearn.model_selection import GridSearchCV
```

```
In [312]: param_grid={ 'C':[0.1,1,10],  
                    'gamma':[1,0.1,0.01],  
                    'kernel':['rbf'],  
                    'class_weight':['balanced']}
```

```
In [313]: grid=GridSearchCV(SVC(),param_grid,refit=True,verbose=2,cv=2)  
grid.fit(x_train,y_train)
```

```
Fitting 2 folds for each of 9 candidates, totalling 18 fits  
[CV] END ..C=0.1, class_weight=balanced, gamma=1, kernel=rbf; total time= 1.8  
s  
[CV] END ..C=0.1, class_weight=balanced, gamma=1, kernel=rbf; total time= 1.8  
s  
[CV] END C=0.1, class_weight=balanced, gamma=0.1, kernel=rbf; total time= 1.2  
s  
[CV] END C=0.1, class_weight=balanced, gamma=0.1, kernel=rbf; total time= 1.2  
s  
[CV] END C=0.1, class_weight=balanced, gamma=0.01, kernel=rbf; total time= 1.  
3s  
[CV] END C=0.1, class_weight=balanced, gamma=0.01, kernel=rbf; total time= 1.  
3s  
[CV] END ....C=1, class_weight=balanced, gamma=1, kernel=rbf; total time= 1.4  
s  
[CV] END ....C=1, class_weight=balanced, gamma=1, kernel=rbf; total time= 1.4  
s  
[CV] END ..C=1, class_weight=balanced, gamma=0.1, kernel=rbf; total time= 1.0  
s  
[CV] END ..C=1, class_weight=balanced, gamma=0.1, kernel=rbf; total time= 1.0  
s  
[CV] END .C=1, class_weight=balanced, gamma=0.01, kernel=rbf; total time= 1.1  
s  
[CV] END .C=1, class_weight=balanced, gamma=0.01, kernel=rbf; total time= 1.1  
s  
[CV] END ...C=10, class_weight=balanced, gamma=1, kernel=rbf; total time= 1.3  
s  
[CV] END ...C=10, class_weight=balanced, gamma=1, kernel=rbf; total time= 1.4  
s  
[CV] END .C=10, class_weight=balanced, gamma=0.1, kernel=rbf; total time= 1.0  
s  
[CV] END .C=10, class_weight=balanced, gamma=0.1, kernel=rbf; total time= 1.0  
s  
[CV] END C=10, class_weight=balanced, gamma=0.01, kernel=rbf; total time= 1.1  
s  
[CV] END C=10, class_weight=balanced, gamma=0.01, kernel=rbf; total time= 1.1  
s
```

```
Out[313]:
```



```
In [314]: print(grid.best_estimator_)

SVC(C=10, class_weight='balanced', gamma=1)

In [315]: grid_predictions=grid.predict(x_test)

In [316]: confusion_matrix(y_test,grid_predictions)

Out[316]: array([[2166,  253],
   [ 365,  216]], dtype=int64)

In [317]: print(classification_report(y_test,grid_predictions))

          precision    recall  f1-score   support

           0       0.86      0.90      0.88     2419
           1       0.46      0.37      0.41      581

    accuracy                           0.79     3000
   macro avg       0.66      0.63      0.64     3000
weighted avg       0.78      0.79      0.79     3000
```

Model with Random Under Sampling

```
In [318]: svc_r=SVC()

In [319]: svc_r.fit(x_train_r,y_train_r)

Out[319]: SVC()
SVC()
```

```
In [320]: y_pred_r=svc_r.predict(x_test_r)
```

Model Accuracy

```
In [321]: confusion_matrix(y_test_r,y_pred_r)

Out[321]: array([[483, 120],
   [172, 448]], dtype=int64)
```

```
In [322]: print(classification_report(y_test_r,y_pred_r))
```

	precision	recall	f1-score	support
0	0.74	0.80	0.77	603
1	0.79	0.72	0.75	620
accuracy			0.76	1223
macro avg	0.76	0.76	0.76	1223
weighted avg	0.76	0.76	0.76	1223

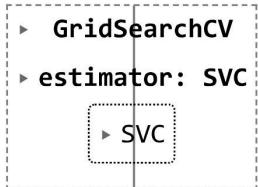
```
In [323]: param_grid=[{'C':[0.1,1,10],  
'gamma':[1,0.1,0.01],  
'kernel':['rbf'],  
'class_weight':['balanced']}]
```

In [324]:

```
grid_r=GridSearchCV(SVC(),param_grid,refit=True,verbose=2,cv=2)
grid_r.fit(x_train_r,y_train_r)
```

Fitting 2 folds for each of 9 candidates, totalling 18 fits
[CV] END ..C=0.1, class_weight=balanced, gamma=1, kernel=rbf; total time= 0.2 s
[CV] END ..C=0.1, class_weight=balanced, gamma=1, kernel=rbf; total time= 0.2 s
[CV] END C=0.1, class_weight=balanced, gamma=0.1, kernel=rbf; total time= 0.1 s
[CV] END C=0.1, class_weight=balanced, gamma=0.1, kernel=rbf; total time= 0.2 s
[CV] END C=0.1, class_weight=balanced, gamma=0.01, kernel=rbf; total time= 0.2 s
[CV] END C=0.1, class_weight=balanced, gamma=0.01, kernel=rbf; total time= 0.2 s
[CV] ENDC=1, class_weight=balanced, gamma=1, kernel=rbf; total time= 0.2 s
[CV] ENDC=1, class_weight=balanced, gamma=1, kernel=rbf; total time= 0.2 s
[CV] END ..C=1, class_weight=balanced, gamma=0.1, kernel=rbf; total time= 0.1 s
[CV] END ..C=1, class_weight=balanced, gamma=0.1, kernel=rbf; total time= 0.1 s
[CV] END .C=1, class_weight=balanced, gamma=0.01, kernel=rbf; total time= 0.1 s
[CV] END .C=1, class_weight=balanced, gamma=0.01, kernel=rbf; total time= 0.1 s
[CV] END ...C=10, class_weight=balanced, gamma=1, kernel=rbf; total time= 0.2 s
[CV] END ...C=10, class_weight=balanced, gamma=1, kernel=rbf; total time= 0.2 s
[CV] END .C=10, class_weight=balanced, gamma=0.1, kernel=rbf; total time= 0.1 s
[CV] END .C=10, class_weight=balanced, gamma=0.1, kernel=rbf; total time= 0.1 s
[CV] END C=10, class_weight=balanced, gamma=0.01, kernel=rbf; total time= 0.1 s
[CV] END C=10, class_weight=balanced, gamma=0.01, kernel=rbf; total time= 0.1 s

Out[324]:



In [325]:

```
print(grid_r.best_estimator_)
```

SVC(C=10, class_weight='balanced', gamma=0.01)

In [326]:

```
grid_predictions_r=grid_r.predict(x_test_r)
```

```
In [327]: confusion_matrix(y_test_r,grid_predictions_r)
```

```
Out[327]: array([[456, 147],
 [161, 459]], dtype=int64)
```

```
In [328]: print(classification_report(y_test_r,grid_predictions_r))
```

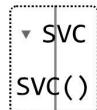
	precision	recall	f1-score	support
0	0.74	0.76	0.75	603
1	0.76	0.74	0.75	620
accuracy			0.75	1223
macro avg	0.75	0.75	0.75	1223
weighted avg	0.75	0.75	0.75	1223

Model with Random Over Sampling

```
In [329]: svc_ros=SVC()
```

```
In [330]: svc_ros.fit(x_train_ros,y_train_ros)
```

```
Out[330]:
```



```
In [331]: y_pred_ros=svc_ros.predict(x_test_ros)
```

Model Accuracy

```
In [332]: from sklearn.metrics import confusion_matrix,classification_report
confusion_matrix(y_test_ros,y_pred_ros)
```

```
Out[332]: array([[ 969, 1457],
 [ 562, 1790]], dtype=int64)
```

```
In [333]: print(classification_report(y_test_ros,y_pred_ros))
```

	precision	recall	f1-score	support
0	0.63	0.40	0.49	2426
1	0.55	0.76	0.64	2352
accuracy			0.58	4778
macro avg	0.59	0.58	0.56	4778
weighted avg	0.59	0.58	0.56	4778

```
In [334]: param_grid=[{'C':[0.1,1,10],
                     'gamma':[1,0.1,0.01],
                     'kernel':['rbf'],
                     'class_weight':['balanced']}]
```

```
In [335]: grid_ros=GridSearchCV(SVC(),param_grid,refit=True,verbose=2,cv=2)
grid_ros.fit(x_train_ros,y_train_ros)
```

Fitting 2 folds for each of 9 candidates, totalling 18 fits

[CV] END ..C=0.1, class_weight=balanced, gamma=1, kernel=rbf; total time= 8.0 s

[CV] END ..C=0.1, class_weight=balanced, gamma=1, kernel=rbf; total time= 8.0 s

[CV] END C=0.1, class_weight=balanced, gamma=0.1, kernel=rbf; total time= 7.9 s

[CV] END C=0.1, class_weight=balanced, gamma=0.1, kernel=rbf; total time= 8.3 s

[CV] END C=0.1, class_weight=balanced, gamma=0.01, kernel=rbf; total time= 8.2 s

[CV] END C=0.1, class_weight=balanced, gamma=0.01, kernel=rbf; total time= 8.3 s

[CV] ENDC=1, class_weight=balanced, gamma=1, kernel=rbf; total time= 9.9 s

[CV] ENDC=1, class_weight=balanced, gamma=1, kernel=rbf; total time= 10.1 s

[CV] END ..C=1, class_weight=balanced, gamma=0.1, kernel=rbf; total time= 8.4 s

[CV] END ..C=1, class_weight=balanced, gamma=0.1, kernel=rbf; total time= 8.2 s

[CV] END .C=1, class_weight=balanced, gamma=0.01, kernel=rbf; total time= 8.9 s

[CV] END .C=1, class_weight=balanced, gamma=0.01, kernel=rbf; total time= 7.9 s

[CV] END ...C=10, class_weight=balanced, gamma=1, kernel=rbf; total time= 6.7 s

[CV] END ...C=10, class_weight=balanced, gamma=1, kernel=rbf; total time= 6.6 s

[CV] END .C=10, class_weight=balanced, gamma=0.1, kernel=rbf; total time= 6.8 s

[CV] END .C=10, class_weight=balanced, gamma=0.1, kernel=rbf; total time= 6.4 s

[CV] END C=10, class_weight=balanced, gamma=0.01, kernel=rbf; total time= 6.7 s

[CV] END C=10, class_weight=balanced, gamma=0.01, kernel=rbf; total time= 6.3 s

```
Out[335]:
```

```

    ▶ GridSearchCV
    ▶ estimator: SVC
        ▶ SVC
  
```

```
In [336]: print(grid_ros.best_estimator_)
```

```
SVC(C=1, class_weight='balanced', gamma=1)
```

```
In [337]: grid_predictions_ros=grid_ros.predict(x_test_ros)
```

```
In [338]: confusion_matrix(y_test_ros,grid_predictions_ros)
```

```
Out[338]: array([[2426,    0],
   [ 151, 2201]], dtype=int64)
```

```
In [339]: print(classification_report(y_test_ros,grid_predictions_ros))
```

	precision	recall	f1-score	support
0	0.94	1.00	0.97	2426
1	1.00	0.94	0.97	2352
accuracy			0.97	4778
macro avg	0.97	0.97	0.97	4778
weighted avg	0.97	0.97	0.97	4778

```
In [340]: print(classification_report(y_test_ros,grid_predictions_ros))
```

	precision	recall	f1-score	support
0	0.94	1.00	0.97	2426
1	1.00	0.94	0.97	2352
accuracy			0.97	4778
macro avg	0.97	0.97	0.97	4778
weighted avg	0.97	0.97	0.97	4778

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```