In [458…

```
import pandas as pd
import numpy as np
from numpy.random import seed
```

In [459…

```
seed(100)
```

In [460…

```
import warnings
warnings.filterwarnings("ignore")
```

In [461…

```
df = pd.read_excel("./DATA/default of credit card clients.xls", skiprows=1)
df.head()
```

Out[461…

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	…	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	de pay… m
0	1	20000	2	2	1	24	2	2	-1	-1	…	0	0	0	0	689	0	0	0	0	
1	2	120000	2	2	2	26	-1	2	0	0	…	3272	3455	3261	0	1000	1000	1000	0	2000	
2	3	90000	2	2	2	34	0	0	0	0	…	14331	14948	15549	1518	1500	1000	1000	1000	5000	
3	4	50000	2	2	1	37	0	0	0	0	…	28314	28959	29547	2000	2019	1200	1100	1069	1000	
4	5	50000	1	2	1	57	-1	0	-1	0	…	20940	19146	19131	2000	36681	10000	9000	689	679	

5 rows × 25 columns

In [462…

```
cols = df.columns.tolist()
cols
```

Out[462…

```
['ID',
 'LIMIT_BAL',
 'SEX',
 'EDUCATION',
 'MARRIAGE',
 'AGE',
 'PAY_0',
 'PAY_2',
 'PAY_3',
 'PAY_4',
 'PAY_5',
 'PAY_6',
 'BILL_AMT1',
 'BILL_AMT2',
 'BILL_AMT3',
 'BILL_AMT4',
 'BILL_AMT5',
 'BILL_AMT6',
 'PAY_AMT1',
 'PAY_AMT2',
 'PAY_AMT3',
 'PAY_AMT4',
 'PAY_AMT5',
 'PAY_AMT6',
 'default payment next month']
```

In [463…

```
idx1 = df[ df['default payment next month']==0 ].ID.values
idx2 = df[ df['default payment next month']==1 ].ID.values
idx1.shape, idx2.shape
```

Out[463…

```
((23364,), (6636,))
```

In [464…

```
n1 = np.random.choice(idx1, 15000)
n2 = np.random.choice(idx2, 5000)
np.shape(n1), np.shape(n2)
```

Out[464…

```
((15000,), (5000,))
```

In [465…

```
n11 = np.unique(n1)
n22 = np.unique(n2)
np.shape(n11), np.shape(n22)
```

Out[465…

```
((11025,), (3553,))
```

In [466…

```
type(n11), type(n22)
```

Out[466…

```
(numpy.ndarray, numpy.ndarray)
```

In [467…

```
def ret_idx(m1, m2):
    idx = list(n11[0:m1]) + list(n22[0:m2])
    return idx
```

m1:m2 = 9:1

In [468…

```
idx = ret_idx(950,50)
np.shape(idx)
```

Out[468…

```
(1000,)
```

In [469…

```
from collections import Counter
```

In [470…

```
Counter(idx).most_common(5)
```

Out[470…

```
[(3, 1), (12, 1), (13, 1), (19, 1), (20, 1)]
```

In [471…

```
df1 = df.iloc[idx,:]
df1.shape
```

Out[471…

```
(1000, 25)
```

In [472…

```
df1.head()
```

Out[472…

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	…	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	d pa…
3	4	50000	2	2	1	37	0	0	0	0	…	28314	28959	29547	2000	2019	1200	1100	1069	1000	
12	13	630000	2	2	2	41	-1	0	-1	-1	…	6500	6500	2870	1000	6500	6500	6500	2870	0	
13	14	70000	1	2	2	30	1	2	2	0	…	66782	36137	36894	3200	0	3000	3000	1500	0	
19	20	180000	2	1	2	29	1	-2	-2	-2	…	0	0	0	0	0	0	0	0	0	
20	21	130000	2	3	2	39	0	0	0	0	…	20616	11802	930	3000	1537	1000	2000	930	33764	

5 rows × 25 columns

ML

In [473…

```
del cols[0]
y = df1['default payment next month'].values
del cols[-1]
cols
```

Out[473…

```
['LIMIT_BAL',
 'SEX',
 'EDUCATION',
 'MARRIAGE',
 'AGE',
 'PAY_0',
 'PAY_2',
 'PAY_3',
 'PAY_4',
 'PAY_5',
 'PAY_6',
 'BILL_AMT1',
 'BILL_AMT2',
 'BILL_AMT3',
 'BILL_AMT4',
 'BILL_AMT5',
 'BILL_AMT6',
 'PAY_AMT1',
 'PAY_AMT2',
 'PAY_AMT3',
 'PAY_AMT4',
 'PAY_AMT5',
 'PAY_AMT6']
```

In [474…

```
data = df1.values
data.shape
```

Out[474…

```
(1000, 25)
```

In [475…

```
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
```

In [476…

```
X = preprocessing.StandardScaler().fit(data).transform(data)
X.shape, y.shape
```

Out[476…

```
((1000, 25), (1000,))
```

In [477…

```
x_tr, x_t, y_tr, y_t = train_test_split(X, y, test_size=0.10, random_state=100)
x_tr.shape, x_t.shape, y_tr.shape, y_t.shape
```

Out[477…

```
((900, 25), (100, 25), (900,), (100,))
```

In [478…

```
from sklearn import svm
from sklearn.svm import SVC
```

In [479…

```
clf = svm.SVC()
clf.fit(x_tr, y_tr)
y_p = clf.predict(x_t)
```

In [480…

```
print(classification_report(y_t, y_p))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99        76
           1       1.00      0.96      0.98        24

    accuracy                           0.99       100
   macro avg       0.99      0.98      0.99       100
weighted avg       0.99      0.99      0.99       100
```

In [481…

```
accuracy_score(y_p, y_t)
```

Out[481…

```
0.99
```

7:3 => 99.3 7.5:2.5 => 99.7 8:2 => 99.3 8.5:1.5 => 99.7 9:1 => 99.3 9.5:0.5 => 99.3Test2: vary test_size with 9.5:0.5 0.99 0.99 0.99 Now decreasing =>> 0.995 0.993 0.99

No clear indication yet !!

In [ ]:

In [ ]:

In [ ]:

In [ ]: