Combination Sampling

Implement the SMOTEENN technique with the credit card default data. Then estimate a logistic regression model and report the classification evaluation metrics.

ln_balance_limit is the log of the maximum balance they can have on the card; 1 is female, 0 male for sex; the education is denoted: 1 = graduate school; 2 = university; 3 = high school; 4 = others; 1 is married and 0 single for marriage; default_next_month is whether the person defaults in the following month (1 yes, 0 no).

In [1]:
```
import pandas as pd
from path import Path
from collections import Counter
```

In [7]:
```
data = Path('../Resources/cc_default.csv')
df = pd.read_csv(data)
df.head()
```

Out[7]:

	ID	ln_balance_limit	sex	education	marriage	age	default_next_month
0	1	9.903488	1	2	0	24	1
1	2	11.695247	1	2	1	26	1
2	3	11.407565	1	2	1	34	0
3	4	10.819778	1	2	0	37	0
4	5	10.819778	0	2	0	57	0

In [8]:
```
x_cols = [i for i in df.columns if i not in ('ID', 'default_next_month')]
X = df[x_cols]
y = df['default_next_month']
```

In [9]:
```
x_cols
```

Out[9]:
```
['ln_balance_limit', 'sex', 'education', 'marriage', 'age']
```

In [10]:
```
Counter(y)
```

Out[10]:
```
Counter({1: 6636, 0: 23364})
```

In [11]:
```
# Normal train-test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
```

Combination Sampling with SMOTEENN

In [12]:
```
# Use the SMOTEENN technique to perform combination sampling on the data
# Count the resampled classes
from imblearn.combine import SMOTEENN

smote_enn = SMOTEENN(random_state=0)
X_resampled, y_resampled = smote_enn.fit_resample(X, y)
Counter(y_resampled)
```

Out[12]:
```
Counter({0: 10148, 1: 7645})
```

Logistic Regression

In [13]:
```
# Fit a Logistic regression model using random undersampled data
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(solver='lbfgs', random_state=1)
model.fit(X_resampled, y_resampled)
```

Out[13]:
```
LogisticRegression(random_state=1)
```

Evaluation Metrics

In [14]:
```
# Display the confusion matrix
from sklearn.metrics import confusion_matrix

y_pred = model.predict(X_test)
confusion_matrix(y_test, y_pred)
```

Out[14]:
```
array([[4179, 1653],
       [ 880,  788]])
```

In [15]:
```
# Calculate the Balanced Accuracy Score
from sklearn.metrics import balanced_accuracy_score

balanced_accuracy_score(y_test, y_pred)
```

Out[15]:
```
0.5944929241791752
```

In [16]:
```
# Print the imbalanced classification report
from imblearn.metrics import classification_report_imbalanced

print(classification_report_imbalanced(y_test, y_pred))
```

```
                   pre       rec       spe        f1       geo       iba
sup

          0       0.83      0.72      0.47      0.77      0.58      0.35
5832
          1       0.32      0.47      0.72      0.38      0.58      0.33
1668

avg / total       0.71      0.66      0.53      0.68      0.58      0.34
7500
```

In [ ]: