In [3]:
```
import pandas as pd
df=pd.read_csv('creditcard.csv')
```

In [4]:
```
df.head()
```

Out[4]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.3
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.2
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.5
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.3
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.8

5 rows × 31 columns

In [5]:
```
df.Class.value_counts()
```

Out[5]:
```
Class
0    284315
1       492
Name: count, dtype: int64
```

In [6]:
```
## Independent and dependent features
X=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

Cross Validation: KFold and Hyperparameter Technique: GridSearchCV

In [7]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_re
from sklearn.model_selection import KFold
import numpy as np
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
```

In [8]:
```
log_class = LogisticRegression()
grid = {'C': 10.0 ** np.arange(-2, 3), 'penalty': ['l1', 'l2']}
cv = KFold(n_splits=5, shuffle=False, random_state=None)
```

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7)
```

In [10]:
```
clf = GridSearchCV(log_class, grid, cv=cv, n_jobs=-1, scoring='f1_macro')
clf.fit(X_train, y_train)
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\model_selection\_validati
on.py:547: FitFailedWarning:
25 fits failed out of a total of 50.
The score on these train-test partitions for these parameters will be set to
nan.
If these failures are not expected, you can try to debug them by setting erro
r_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
---
25 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\model_selection\_
validation.py", line 895, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\base.py", line 14
74, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\linear_model\_log
istic.py", line 1172, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\linear_model\_log
istic.py", line 67, in _check_solver
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or None penalties, got l1 penalt
y.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\model_selection\_search.p
y:1051: UserWarning: One or more of the test scores are non-finite: [       n
an 0.77737632        nan 0.79331461        nan 0.79586192
        nan 0.78327248        nan 0.7837017 ]
  warnings.warn(
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.p
y:469: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

Out[10]:

GridSearchCV (https://scikit-learn.org/1.4/modules/generated/sklearn.model_selection.G

estimator: LogisticRegression

LogisticRegression (https://scikit-learn.org/1.4/modules/generated/sklearn.linear_model.LogisticR

In [14]:

```
prediction=clf.predict(X_test)
print(confusion_matrix(y_test,prediction))
print(accuracy_score(y_test,prediction))
print(classification_report(y_test,prediction))
```

```
[[199001     12]
 [    87    265]]
0.9995034233691972
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    199013
           1       0.96      0.75      0.84       352

    accuracy                           1.00    199365
   macro avg       0.98      0.88      0.92    199365
weighted avg       1.00      1.00      1.00    199365
```

In [13]:

```
class_weight = dict({0: 1, 1: 100})
```

In [12]:

```
from sklearn.ensemble import RandomForestClassifier
clf=RandomForestClassifier(class_weight=class_weight)
clf.fit(X_train,y_train)
```

Out[12]:

RandomForestClassifier (https://scikit-learn.org/1.4/modules/generated/skl

RandomForestClassifier(class_weight={0: 1, 1: 100})

In [15]:
```
prediction = clf.predict(X_test)
print(confusion_matrix(y_test, prediction))
print(accuracy_score(y_test, prediction))
print(classification_report(y_test, prediction))
```

```
[[199001     12]
 [    87    265]]
0.9995034233691972
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    199013
           1       0.96      0.75      0.84       352

    accuracy                           1.00    199365
   macro avg       0.98      0.88      0.92    199365
weighted avg       1.00      1.00      1.00    199365
```

Under Sampling

In [16]:
```
from imblearn.under_sampling import NearMiss
from collections import Counter
```

In [17]:
```
ns = NearMiss(sampling_strategy=0.8)
X_train_ns, y_train_ns = ns.fit_resample(X_train, y_train)
print("The number of classes before fit {}".format(Counter(y_train)))
print("The number of classes after fit {}".format(Counter(y_train_ns)))
```

```
The number of classes before fit Counter({0: 85302, 1: 140})
The number of classes after fit Counter({0: 175, 1: 140})
```

In [18]:
```
y_train.value_counts()
```

Out[18]:
```
Class
0    85302
1      140
Name: count, dtype: int64
```

In [19]:
```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier()
clf.fit(X_train_ns, y_train_ns)
```

Out[19]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [20]:
```
prediction = clf.predict(X_test)
print(confusion_matrix(y_test, prediction))
print(accuracy_score(y_test, prediction))
print(classification_report(y_test, prediction))
```

```
[[105921  93092]
 [    22    330]]
0.5329471070649312
              precision    recall  f1-score   support

           0       1.00      0.53      0.69    199013
           1       0.00      0.94      0.01       352

    accuracy                           0.53    199365
   macro avg       0.50      0.73      0.35    199365
weighted avg       1.00      0.53      0.69    199365
```

Over Sampling

In [21]:
```
from imblearn.over_sampling import RandomOverSampler
```

In [22]:
```
os = RandomOverSampler(sampling_strategy=0.5)
X_train_os, y_train_os = os.fit_resample(X_train, y_train)
print("The number of classes before fit {}".format(Counter(y_train)))
print("The number of classes after fit {}".format(Counter(y_train_os)))
```

```
The number of classes before fit Counter({0: 85302, 1: 140})
The number of classes after fit Counter({0: 85302, 1: 42651})
```

In [24]:
```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier()
clf.fit(X_train_os, y_train_os)
```

Out[24]:

RandomForestClassifier (https://scikit-learn.org/1.4/modules/generated/sklearn.ensemble.RandomFore

```
RandomForestClassifier()
```

In [25]:
```
prediction = clf.predict(X_test)
print(confusion_matrix(y_test, prediction))
print(accuracy_score(y_test, prediction))
print(classification_report(y_test, prediction))
```

```
[[198992     21]
 [    77    275]]
0.9995084392947609
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    199013
           1       0.93      0.78      0.85       352

    accuracy                           1.00    199365
   macro avg       0.96      0.89      0.92    199365
weighted avg       1.00      1.00      1.00    199365
```

SMOTETomek

In [26]:
```
from imblearn.combine import SMOTETomek
```

In [29]:
```
os = SMOTETomek(sampling_strategy=0.5)
X_train_os, y_train_os = os.fit_resample(X_train, y_train)
print("The number of classes before fit {}".format(Counter(y_train)))
print("The number of classes after fit {}".format(Counter(y_train_os)))
```

```
The number of classes before fit Counter({0: 85302, 1: 140})
The number of classes after fit Counter({0: 84321, 1: 41670})
```

In [31]:
```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier()
clf.fit(X_train_os, y_train_os)
```

Out[31]:
```
RandomForestClassifier(https://scikit-learn.org/1.4/modules/generated/sklearn.ensemble.RandomFore
RandomForestClassifier()
```

In [32]:
```
prediction = clf.predict(X_test)
print(confusion_matrix(y_test, prediction))
print(accuracy_score(y_test, prediction))
print(classification_report(y_test, prediction))
```

```
[[198965     48]
 [    66    286]]
0.9994281844857422
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    199013
           1       0.86      0.81      0.83       352

    accuracy                           1.00    199365
   macro avg       0.93      0.91      0.92    199365
weighted avg       1.00      1.00      1.00    199365
```

In [ ]:

In [ ]: