Kaggle 신용카드 부정결제 검출 (Google Drive Mount)

https://www.kaggle.com/mlg-ulb/creditcardfraud

Credit Card Fraud Detection

- creditcard.csv (284,807 * 31)
- Class : '0' (정상결제), '1' (부정결제)
- 부정 검출(Fraud Detection), 이상 탐지(Anomaly Detection)

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

I. Google Drive Mount

- 'creditCardFraud.zip' 파일을 구글드라이브에 업로드 후 진행

```
from google.colab import drive

drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

- 마운트 결과 확인

```
!ls -l '/content/drive/My Drive/Colab Notebooks/datasets/creditCardFraud.zip'
```

```
-rw------- 1 root root 69155672 Mar  4 04:46 '/content/drive/My Drive/Colab Notebooks/dataset
```

II. Data Preprocessing

1) Unzip 'creditCardFraud.zip'

- Colab 파일시스템에 'creditcard.csv' 파일 생성

```
!unzip /content/drive/My₩ Drive/Colab₩ Notebooks/datasets/creditCardFraud.zip
```

```
Archive:  /content/drive/My Drive/Colab Notebooks/datasets/creditCardFraud.zip
  inflating: creditcard.csv
```

- creditcard.csv 파일 확인

```
!ls -l
```

```
total 147304
-rw-r--r-- 1 root root 150828752 Sep 20  2019 creditcard.csv
drwx------ 5 root root      4096 Mar  9 02:11 drive
drwxr-xr-x 1 root root      4096 Mar  5 14:37 sample_data
```

2) 데이터 읽어오기

- pandas DataFrame

```
%%time

import pandas as pd

DF = pd.read_csv('creditcard.csv')

DF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
CPU times: user 2.82 s, sys: 180 ms, total: 3 s
Wall time: 3.06 s
```

```
DF.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270

- '0' (정상) Class와 '1' (부정) Class 개수

```
DF.Class.value_counts()
```

```
0    284315
1       492
Name: Class, dtype: int64
```

- '0' (정상) Class와 '1' (부정) Class 비율

```
(DF.Class.value_counts() / DF.shape[0]) * 100
```

```
0    99.827251
1     0.172749
Name: Class, dtype: float64
```

3) Time 열(Column) 삭제

```
DF.drop('Time', axis = 1, inplace = True)

DF.head(1)
```

	V1	V2	V3	V4	V5	V6	V7	V8	
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.36

4) train_test_split

- X (Input), y (Output) 지정

```
X = DF.iloc[:,:-1]
y = DF.iloc[:, -1]

X.shape, y.shape
```

```
((284807, 29), (284807,))
```

(1) Without 'stratify'

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.3,
                                                    random_state = 2045)

X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((199364, 29), (199364,), (85443, 29), (85443,))
```

- Train_Data와 Test_Data의 1 (부정) 비율이 불균형

```
print('Train_Data :','₩n', (y_train.value_counts() / y_train.shape[0]) * 100)
print('Test_Data :','₩n', (y_test.value_counts() / y_test.shape[0]) * 100)
```

```
Train_Data :
 0    99.825445
1     0.174555
Name: Class, dtype: float64
Test_Data :
 0    99.831467
1     0.168533
Name: Class, dtype: float64
```

(2) With 'Stratify'

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.3,
                                                    stratify = y,
                                                    random_state = 2045)

X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((199364, 29), (199364,), (85443, 29), (85443,))
```

- Train_Data와 Test_Data의 1 (부정) 비율이 균형

```
print('Train_Data :','₩n', (y_train.value_counts() / y_train.shape[0]) * 100)
print('Test_Data :','₩n', (y_test.value_counts() / y_test.shape[0]) * 100)
```

```
Train_Data :
 0    99.827451
1     0.172549
Name: Class, dtype: float64
Test_Data :
 0    99.826785
1     0.173215
Name: Class, dtype: float64
```

III. Modeling

1) Decision Tree - Without SMOTE

```
%%time

from sklearn.tree import DecisionTreeClassifier

Model_dt = DecisionTreeClassifier()
Model_dt.fit(X_train, y_train)
```

```
CPU times: user 23.7 s, sys: 43 ms, total: 23.7 s
Wall time: 24 s
```

```
y_hat = Model_dt.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test, y_hat)
```

```
array([[85259,    36],
       [   27,   121]])
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score

print(accuracy_score(y_test, y_hat))
print(precision_score(y_test, y_hat, pos_label = 1))
print(recall_score(y_test, y_hat, pos_label = 1))
```

```
0.9992626663389628
```

```
0.7707006369426752
0.8175675675675675
```

```
from sklearn.metrics import f1_score

f1_score(y_test, y_hat, pos_label = 1)
```

```
0.7934426229508197
```

2) SMOTE

- Synthetic Minority Over-sampling TEchnique
- KNN(K-Nearst Neighbor) : K개의 이웃과 일정 값의 차이를 가지를 새로운 데이터를 생성
- imbalanced-learn Package

```
# Without SMOTE

X_train.shape, y_train.shape
```

```
((199364, 29), (199364,))
```

- imbalanced-learn Package

```
from imblearn.over_sampling import SMOTE
```

- With SMOTE

```
%%time

OS = SMOTE(random_state = 2045)

X_train_OS, y_train_OS = OS.fit_sample(X_train, y_train)

X_train_OS.shape, y_train_OS.shape
```

```
CPU times: user 1.29 s, sys: 208 ms, total: 1.5 s
Wall time: 1.51 s
```

- 0 (정상) Class와 1 (사기) Class 개수

```
pd.Series(y_train_OS).value_counts()
```

```
1    199020
0    199020
dtype: int64
```

3) Decision Tree - With SMOTE

```
%%time

from sklearn.tree import DecisionTreeClassifier

Model_dt = DecisionTreeClassifier()
Model_dt.fit(X_train_OS, y_train_OS)
```

```
CPU times: user 42.1 s, sys: 36.7 ms, total: 42.2 s
Wall time: 42.3 s
```

```
y_hat = Model_dt.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test, y_hat)
```

```
array([[85141,   154],
       [   28,   120]])
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score

print(accuracy_score(y_test, y_hat))
print(precision_score(y_test, y_hat, pos_label = 1))
print(recall_score(y_test, y_hat, pos_label = 1))
```

```
0.9978699249792259
0.43795620437956206
0.8108108108108109
```

```
from sklearn.metrics import f1_score

f1_score(y_test, y_hat, pos_label = 1)
```

```
0.5687203791469194
```

4) LightGBM - With SMOTE

- n_estimators : 모델링에 사용되는 Tree의 개수
- num_leaves : 최대 Terminal Node 개수
- boost_from_average : 불균형 데이터일 경우 'False' 지정
- learning_rate : 0~1 사이의 값
- max_depth : Tree의 최대 크기(깊이)

- min_child_samples : Terminal Node의 최소 Datapoint 개수
- 약 90초

```
%%time

from lightgbm import LGBMClassifier

Model_lgbm = LGBMClassifier(n_estimators = 1500,
                            num_leaves = 64,
                            n_jobs = -1,
                            boost_from_average = False)

Model_lgbm.fit(X_train_OS, y_train_OS)
```

```
CPU times: user 3min 22s, sys: 528 ms, total: 3min 23s
Wall time: 1min 43s
```

```
y_hat = Model_lgbm.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test, y_hat)
```

```
array([[85273,    22],
       [   19,   129]])
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score

print(accuracy_score(y_test, y_hat))
print(precision_score(y_test, y_hat, pos_label = 1))
print(recall_score(y_test, y_hat, pos_label = 1))
```

```
0.9995201479348805
0.8543046357615894
0.8716216216216216
```

```
from sklearn.metrics import f1_score

f1_score(y_test, y_hat, pos_label = 1)
```

```
0.862876254180602
```

#

#

#

The End

#

```
#
```

```
#
```