Kaggle 신용카드 부정결제 검출

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)

+ 코드 + 텍스트

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

I. wget From Github

- 'creditCardFraud.zip' 파일 다운로드

```
!wget https://raw.githubusercontent.com/rusita-ai/pyData/master/creditCardFraud.zip
```

```
--2023-07-09 02:21:06--  https://raw.githubusercontent.com/rusita-ai/pyData/master/creditCardFraud.zip
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 69155672 (66M) [application/zip]
Saving to: 'creditCardFraud.zip'

creditCardFraud.zip 100%[===================>]  65.95M   169MB/s    in 0.4s

2023-07-09 02:21:10 (169 MB/s) - 'creditCardFraud.zip' saved [69155672/69155672]
```

- 다운로드 결과 확인

```
!ls -l
```

```
total 67540
-rw-r--r-- 1 root root 69155672 Jul  9 02:21 creditCardFraud.zip
drwxr-xr-x 1 root root     4096 Jul  6 13:44 sample_data
```

II. Data Preprocessing

1) Unzip 'creditCardFraud.zip'

- Colab 파일시스템에 'creditcard.csv' 파일 생성

```
!unzip creditCardFraud.zip
```

```
Archive:  creditCardFraud.zip
  inflating: creditcard.csv
```

- creditcard.csv 파일 확인

```
!ls -l
```

```
total 214840
-rw-r--r-- 1 root root 150828752 Sep 20  2019 creditcard.csv
-rw-r--r-- 1 root root  69155672 Jul  9 02:21 creditCardFraud.zip
drwxr-xr-x 1 root root      4096 Jul  6 13:44 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 3.97 s, sys: 195 ms, total: 4.16 s
Wall time: 4.85 s
```

```
DF.head()
```

	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	0.09869
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.08510
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.24767
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.37743
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.27053

5 rows × 31 columns

3) 'Time' -> 'hours'

- 'Time': 각 거래와 첫 번째 거래 사이에 경과된 초('Seconds')

(1) 시간('hours') 정보 생성

```
timedelta = pd.to_timedelta(DF['Time'], unit = 's')

DF['Time'] = (timedelta.dt.components.hours).astype(int)
```

```
DF.head(3)
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8
0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698
1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102
2	0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676

3 rows × 31 columns

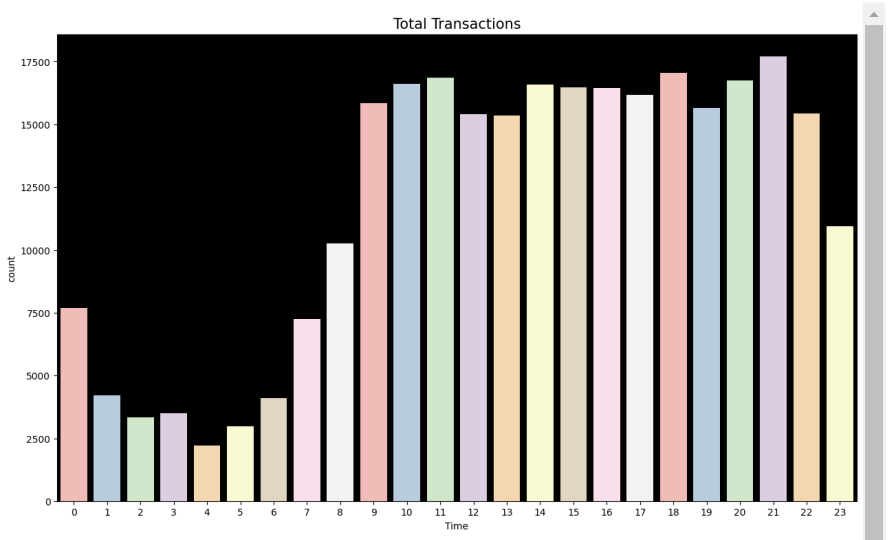
(2) 시간별 거래량

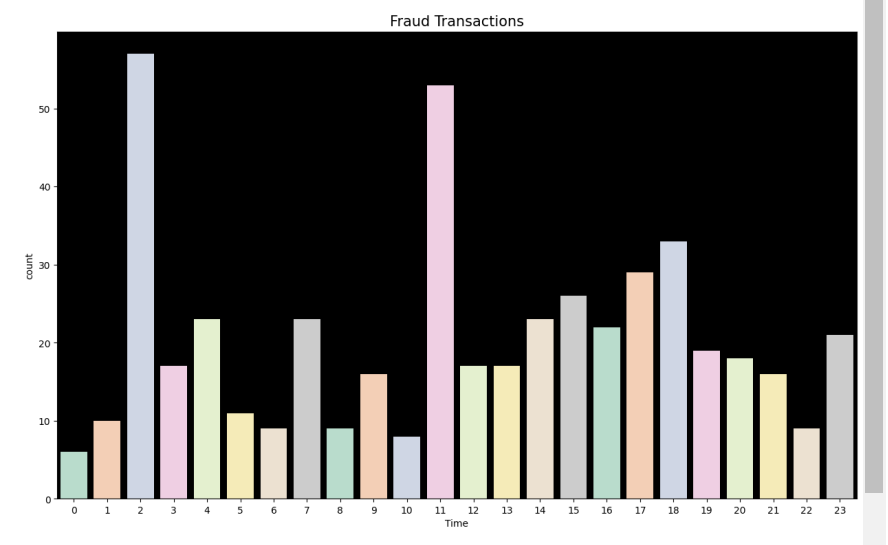
```
DF['Time'].value_counts()
```

```
21    17703
18    17039
11    16856
20    16756
10    16598
14    16570
15    16461
16    16453
17    16166
9     15838
19    15649
22    15441
12    15420
13    15365
23    10938
8     10276
0      7695
7      7243
1      4220
6      4101
3      3492
2      3328
5      2990
4      2209
Name: Time, dtype: int64
```

(3) 시간별 거래량 Visualization

```
import matplotlib.pyplot as plt
import seaborn as sns

fig, axs = plt.subplots(3, figsize = (15, 30))

sns.countplot(x = DF['Time'],
              ax = axs[0],
              palette = 'Pastel1')
axs[0].set_title('Total Transactions',
                 fontsize = 15)
axs[0].set_facecolor("black")

sns.countplot(x = DF[(DF['Class'] == 1)]['Time'],
              ax = axs[1],
              palette = 'Pastel2')
axs[1].set_title('Fraud Transactions',
                 fontsize = 15)
axs[1].set_facecolor('black')

sns.countplot(x= DF[(DF['Class'] == 0)]['Time'],
              ax = axs[2],
              palette = 'Set3')
axs[2].set_title('Normal Transactions',
                 fontsize = 15)
axs[2].set_facecolor("black")

plt.show()
```





4) Visualization

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize = (20, 50))

for index, col_names in enumerate(DF.columns[:-1]):
  ax1 = plt.subplot(10, 3, index + 1)
  sns.boxplot(data = DF,
              x = 'Class',
              y = col_names,
              order = [0, 1],
              showfliers = False,
              ax = ax1)

plt.show()
```

# # #

End Of Document

# # #