

ML | Credit Card Fraud Detection

Divyank Singh

Overview

The data set has 31 features, 28 of which have been anonymized and are labeled V1 through V28. The remaining three features are the time and the amount of the transaction as well as whether that transaction was fraudulent or not. Before it was uploaded to Kaggle, the anonymized variables had been modified in the form of a PCA (Principal Component Analysis). Furthermore, there were no missing values in the data set. Equipped with this basic description of the data, let's jump into some exploratory data analysis.

Milestone Report

1. As the data was obtained from kaggle therefore it was pretty clean data.

```
data = pd.read_csv('creditcard.csv')
```

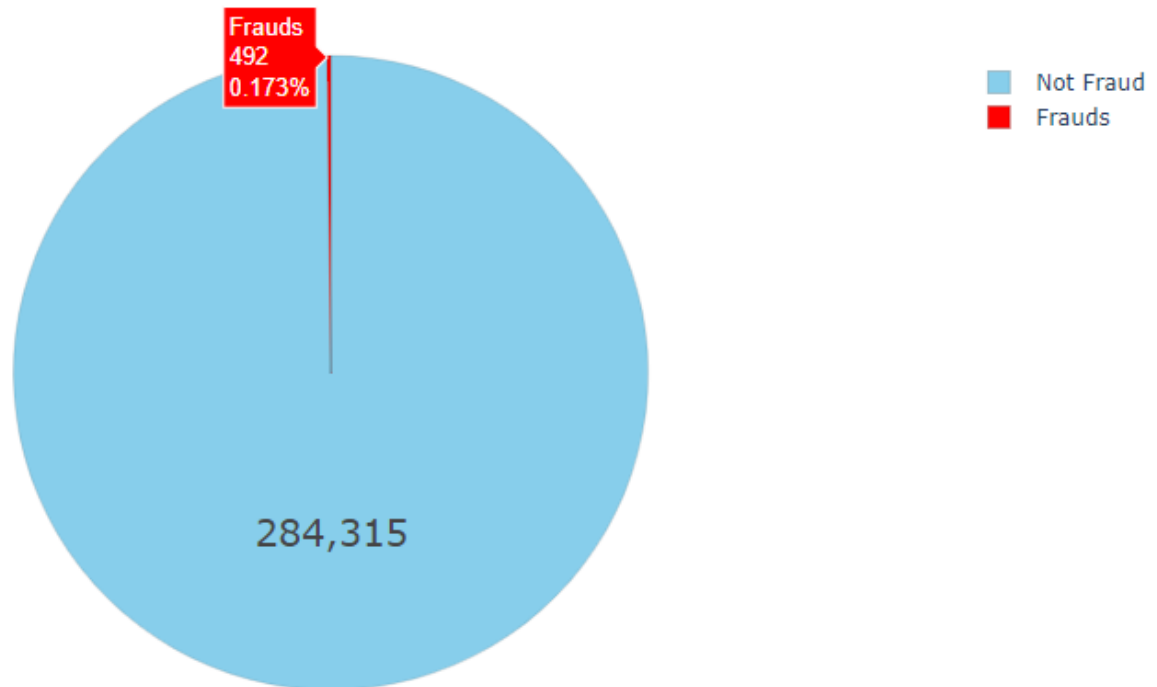
```
data.shape
```

```
(284807, 31)
```

```
data.head().style.set_properties(**{"background-color":"black","color":"lawngreen"})
```

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 | V11 | V12 | V13 | |
|---|----------|-----------|-----------|----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|--------|
| 0 | 0.000000 | -1.359807 | -0.072781 | 2.536347 | 1.378155 | -0.338321 | 0.462388 | 0.239599 | 0.098698 | 0.363787 | 0.090794 | -0.551600 | -0.617801 | -0.991390 | -0.311 |
| 1 | 0.000000 | 1.191857 | 0.266151 | 0.166480 | 0.448154 | 0.060018 | -0.082361 | -0.078803 | 0.085102 | -0.255425 | -0.166974 | 1.612727 | 1.065235 | 0.489095 | -0.143 |
| 2 | 1.000000 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 | 0.791461 | 0.247676 | -1.514654 | 0.207643 | 0.624501 | 0.066084 | 0.717293 | -0.165 |
| 3 | 1.000000 | -0.966272 | -0.185226 | 1.792993 | -0.863291 | -0.010309 | 1.247203 | 0.237609 | 0.377436 | -1.387024 | -0.054952 | -0.226487 | 0.178228 | 0.507757 | -0.287 |
| 4 | 2.000000 | -1.158233 | 0.877737 | 1.548718 | 0.403034 | -0.407193 | 0.095921 | 0.592941 | -0.270533 | 0.817739 | 0.753074 | -0.822843 | 0.538196 | 1.345852 | -1.119 |

2. The dataset is **highly imbalanced** as only **492 transactions** are fraud transaction out of **284807 transactions**.



3. Out of **492 fraud transactions** the mean amount turned out to be 122.21 and the maximum fraud transaction amount turned out to be 2125.87.

```
fraud.Amount.describe()
count      492.000000
mean       122.211321
std        256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%        105.890000
max        2125.870000
Name: Amount, dtype: float64
```

4. Out of **284315 valid transactions** the mean amount turned out to be 88.29 and the maximum valid transaction amount turned out to be 25691.16.

```
normal.Amount.describe()
```

```
count    284315.000000
mean       88.291022
std       250.105092
min         0.000000
25%        5.650000
50%       22.000000
75%       77.050000
max      25691.160000
Name: Amount, dtype: float64
```

5. It did **not contain** neither null value nor missing value.

```
data.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
```

6. The dataset does **contain** some extreme outlier, this was **verified by plotting** various graphs such as box plot graphs.
7. I also checked on whether the fraud transaction is more during specific period of time frame by plotting a scatter plot between “Time of transaction vs Amount by class” but that did not yield any clear result .

Class and Amount vs Time

```
# We Will check Do fraudulent transactions occur more often during certain time frame ? Let us find out
```

```
f, (ax1, ax2) = plt.subplots(2, 1, sharex=True)
f.suptitle('Time of transaction vs Amount by class')
ax1.scatter(fraud.Time, fraud.Amount)
ax1.set_title('Fraud')
ax2.scatter(normal.Time, normal.Amount)
ax2.set_title('Normal')
plt.xlabel('Time (in Seconds)')
plt.ylabel('Amount')
plt.show()
```

