```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import pandas as pd
```

```
data = pd.read_csv("/content/Int_dataset.csv")
data.head()
```

| | 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** | 1 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 | 0.791461 | 0.247676 |
| **3** | 1 | -0.966272 | -0.185226 | 1.792993 | -0.863291 | -0.010309 | 1.247203 | 0.237609 | 0.377436 |
| **4** | 2 | -1.158233 | 0.877737 | 1.548718 | 0.403034 | -0.407193 | 0.095921 | 0.592941 | -0.270533 |

5 rows × 31 columns

```
data["Class"].value_counts()
```

```
0.0    34854
1.0      103
Name: Class, dtype: int64
```

```
normal = data[data.Class == 0]
scam = data[data.Class == 1]
```

```
scam.Class.value_counts()
```

```
1.0    103
Name: Class, dtype: int64
```

```
new_data = normal.sample(n=492)
scam
```

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | |
|---|---|---|---|---|---|---|---|---|---|
| **541** | 406 | -2.312227 | 1.951992 | -1.609851 | 3.997906 | -0.522188 | -1.426545 | -2.537387 | 1.3 |
| **623** | 472 | -3.043541 | -3.157307 | 1.088463 | 2.288644 | 1.359805 | -1.064823 | 0.325574 | -0.0 |
| **4920** | 4462 | -2.303350 | 1.759247 | -0.359745 | 2.330243 | -0.821628 | -0.075788 | 0.562320 | -0.3 |
| **6108** | 6986 | -4.397974 | 1.358367 | -2.592844 | 2.679787 | -1.128131 | -1.706536 | -3.496197 | -0.2 |
| **6329** | 7519 | 1.234235 | 3.019740 | -4.304597 | 4.732795 | 3.624201 | -1.357746 | 1.713445 | -0.4 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **30442** | 35926 | -3.896583 | 4.518355 | -4.454027 | 5.547453 | -4.121459 | -1.163407 | -6.805053 | 2.9 |
| **30473** | 35942 | -4.194074 | 4.382897 | -5.118363 | 4.455230 | -4.812621 | -1.224645 | -7.281328 | 3.3 |
| **30496** | 35953 | -4.844372 | 5.649439 | -6.730396 | 5.252842 | -4.409566 | -1.740767 | -6.311699 | 3.4 |
| **31002** | 36170 | -5.685013 | 5.776516 | -7.064977 | 5.902715 | -4.715564 | -1.755633 | -6.958679 | 3.8 |
| **33276** | 37167 | -7.923891 | -5.198360 | -3.000024 | 4.420666 | 2.272194 | -3.394483 | -5.283435 | 0.1 |

103 rows × 31 columns

```
new_data = pd.concat([new_data,scam],axis=0)
```

```
new_data.head()
```

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | |
|---|---|---|---|---|---|---|---|---|---|
| **12826** | 22518 | 1.185932 | -0.304126 | 1.016854 | -0.564526 | -0.941222 | -0.299163 | -0.714790 | 0.0 |
| **4948** | 4496 | -0.943128 | 1.355472 | 1.424962 | 1.550441 | 0.062901 | 0.100756 | 0.203065 | 0.3 |
| **7819** | 10893 | 1.067726 | 0.617049 | 0.300988 | 2.764698 | 0.194083 | -0.515162 | 0.425758 | -0.2 |
| **31846** | 36540 | 1.213307 | 0.277917 | 0.318310 | 0.580329 | -0.344206 | -0.755309 | -0.040387 | -0.0 |
| **26102** | 33908 | -0.948376 | 0.018859 | 1.467157 | 0.332388 | 1.123956 | 1.534841 | -0.095735 | 0.5 |

5 rows × 31 columns

```
new_data.describe()
```

| | Time | V1 | V2 | V3 | V4 | V5 | V6 |
|---|---|---|---|---|---|---|---|
| **count** | 595.000000 | 595.000000 | 595.000000 | 595.000000 | 595.000000 | 595.000000 | 595.00000 |
| **mean** | 23797.462185 | -1.580652 | 0.995949 | -1.273854 | 1.107867 | -1.103884 | -0.30296 |
| **std** | 12061.790860 | 5.129083 | 3.242786 | 6.083743 | 2.772186 | 3.691124 | 1.63940 |
| **min** | 11.000000 | -30.552380 | -11.173511 | -31.103685 | -3.542486 | -22.105532 | -4.97769 |
| **25%** | 12418.500000 | -1.599538 | -0.441326 | -0.362924 | -0.579894 | -1.143213 | -0.98098 |
| **50%** | 28658.000000 | -0.480487 | 0.287722 | 0.588477 | 0.456498 | -0.391977 | -0.27738 |
| **75%** | 34144.500000 | 1.113977 | 1.264830 | 1.324030 | 1.747282 | 0.319346 | 0.29101 |
| **max** | 37887.000000 | 1.582499 | 16.713389 | 3.200586 | 11.927512 | 5.252943 | 4.76242 |

8 rows × 31 columns

```
new_data.groupby("Class").mean()
X = new_data.drop("Class",axis=1)
Y = new_data["Class"]
```

```
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.2,random_state=2)
```

```
model = LogisticRegression()
model.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Converger
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
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
▾ LogisticRegression
LogisticRegression()
```

```
pred = model.predict(x_test)
```

```
acc = accuracy_score(pred,y_test)
```

```
print(acc)
```

```
     0.9747899159663865
```

```
#pred = model.predict(x_test)
#acc = accuracy_score(pred,y_test)
#new_data.corr()
```

```
def pred(data):
    data = np.asarray(data).reshape(1,-1)
    predd = model.predict(data)
    if predd == 0:
        print("it's a normal card")
    else:
        print("it's a scam card")
```

```
pred([4,1.22965763450793,0.141003507049326,0.0453707735899449,1.20261273673594,0.191880988597
```

```
     it's a scam card
     /usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
       warnings.warn(
```

```
pred([4462,-2.30334956758553,1.759247460267,-0.359744743330052,2.33024305053917,-0.82162832
```

```
     it's a scam card
     /usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
       warnings.warn(
```