In [1]:

```python
import seaborn as sns
import numpy as np
import pandas as pd
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```python
credit_card = pd.read_csv('FRAUD DETECTION CREDIT CARD.csv')
credit_card.head(10)
```

Out[2]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422
5	2.0	-0.425966	0.960523	1.141109	-0.168252	0.420987	-0.029728	0.476201	0.260314	-0.568671	...	-0.208254	-0.559825	-0.026398	-0.371427	-0.232794	0.105915	0.253844
6	4.0	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.005159	0.081213	0.464960	...	-0.167716	-0.270710	-0.154104	-0.780055	0.750137	-0.257237	0.034507
7	7.0	-0.644269	1.417964	1.074380	-0.492199	0.948934	0.428118	1.120631	-3.807864	0.615375	...	1.943465	-1.015455	0.057504	-0.649709	-0.415267	-0.051634	-1.206921
8	7.0	-0.894286	0.286157	-0.113192	-0.271526	2.669599	3.721818	0.370145	0.851084	-0.392048	...	-0.073425	-0.268092	-0.204233	1.011592	0.373205	-0.384157	0.011747
9	9.0	-0.338262	1.119593	1.044367	-0.222187	0.499361	-0.246761	0.651583	0.069539	-0.736727	...	-0.246914	-0.633753	-0.120794	-0.385050	-0.069733	0.094199	0.246219

10 rows × 31 columns

In [3]:

```python
credit_card.tail(10)
```

Out[3]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26
284797	172782.0	-0.241923	0.712247	0.399806	-0.463406	0.244531	-1.343668	0.929369	-0.206210	0.106234	...	-0.228876	-0.514376	0.279598	0.371441	-0.559238	0.113144
284798	172782.0	0.219529	0.881246	-0.635891	0.960928	-0.152971	-1.014307	0.427126	0.121340	-0.285670	...	0.099936	0.337120	0.251791	0.057688	-1.508368	0.144023
284799	172783.0	-1.775135	-0.004235	1.189786	0.331096	1.196063	5.519980	-1.518185	2.080825	1.159498	...	0.103302	0.654850	-0.348929	0.745323	0.704545	-0.127579
284800	172784.0	2.039560	-0.175233	-1.196825	0.234580	-0.008713	-0.726571	0.017050	-0.118228	0.435402	...	-0.268048	-0.717211	0.297930	-0.359769	-0.315610	0.201114
284801	172785.0	0.120316	0.931005	-0.546012	-0.745097	1.130314	-0.235973	0.812722	0.115093	-0.204064	...	-0.314205	-0.808520	0.050343	0.102800	-0.435870	0.124079
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864	1.014480	-0.509348	1.436807	0.250034
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.012463	-1.016226	-0.606624	-0.395255
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.037501	0.640134	0.265745	-0.087371
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.163298	0.123205	-0.569159	0.546668
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.376777	0.008797	-0.473649	-0.818267

10 rows × 31 columns

In [4]:

```python
credit_card.shape
```

Out[4]: (284807, 31)

In [5]:

```python
credit_card.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
```

In [6]:

```python
#check missing values each columns
credit_card.isnull().sum()
```

Out[6]:

```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

In [7]:

```python
# Distribution on transation # 0 Normal Transaction and 1 Fraudulent Transaction # Inbalance dataset
credit_card['Class'].value_counts()
```

Out[7]:

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

In [8]:

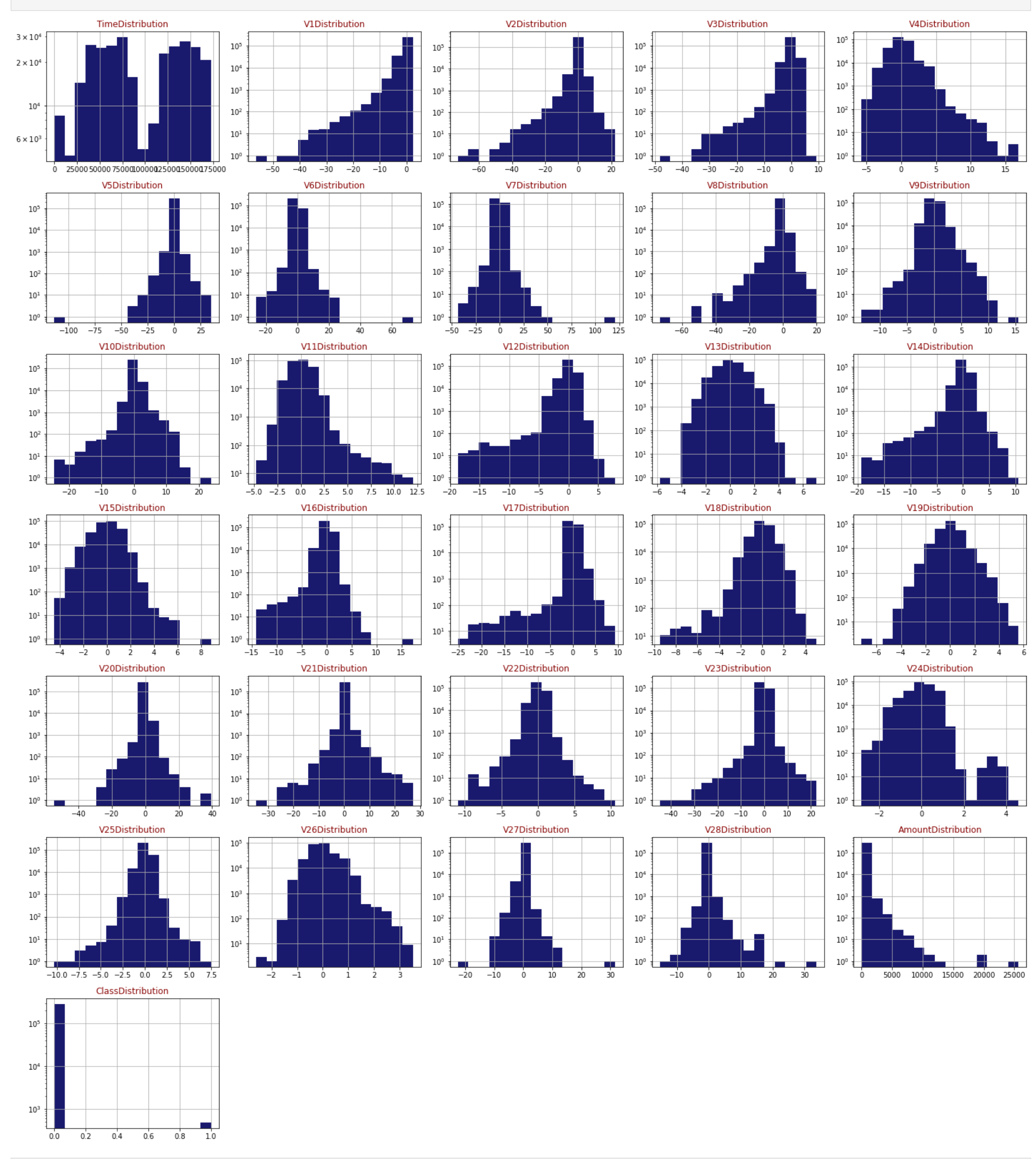
```python
#statistical measures of date
credit_card.Amount.describe()
```

Out[8]:

```
count    284807.000000
mean         88.349619
std         250.120109
min           0.000000
25%           5.600000
50%          22.000000
75%          77.165000
max       25691.160000
Name: Amount, dtype: float64
```

In [9]:

```python
import matplotlib.pyplot as plt
def draw_histograms(dataframe, features, rows, cols):
    fig=plt.figure(figsize=(20,25))
    for i, feature in enumerate(features):
        ax=fig.add_subplot(rows,cols,i+1)
        dataframe[feature].hist(bins=15,ax=ax,facecolor='midnightblue')
        ax.set_title(feature+'Distribution', color='DarkRed')
        ax.set_yscale('log')
    fig.tight_layout()
    plt.show()
draw_histograms(credit_card,credit_card.columns,8,5)
```



In [10]:

```python
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
```

Training Model in Logistic Regression Data

In [11]:

```python
x=credit_card[['V3']]
y=credit_card[['Class']]
```

In [12]:

```python
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.4,random_state=20)
print(x_train)
print(y_train)
```

```
              V3
213718 -2.207794
212769  0.288451
183893 -0.972378
125908  0.596302
196626 -0.858383
...          ...
124308  1.807926
178569 -0.501296
31962   2.141873
220060 -1.604193
37135   0.051301

[170884 rows x 1 columns]
        Class
213718      0
212769      0
183893      0
125908      0
196626      0
...       ...
124308      0
178569      0
31962       0
220060      0
37135       0

[170884 rows x 1 columns]
```

In [13]:

```python
x_train.shape
```

Out[13]: (170884, 1)

In [14]:

```python
x_test.shape
```

Out[14]: (113923, 1)

In [16]:

```python
log = LogisticRegression()
log.fit(x_train,y_train)
```

Out[16]:

```
▾ LogisticRegression
LogisticRegression()
```

Testing of Data

In [17]:

```python
log.score(x_test,y_test)
```

Out[17]: 0.9985077640160459

In [18]:

```python
y_pred=log.predict(x_test)
```

In [19]:

```python
accuracy_score(y_test,y_pred)
```

Out[19]: 0.9985077640160459

In [20]:

```python
precision_score(y_test,y_pred)
```

Out[20]: 0.5294117647058824

In [21]:

```python
recall_score(y_test,y_pred)
```

Out[21]: 0.15606936416184972

In [22]:

```python
f1_score(y_test,y_pred)
```

Out[22]: 0.24107142857142858

In [23]:

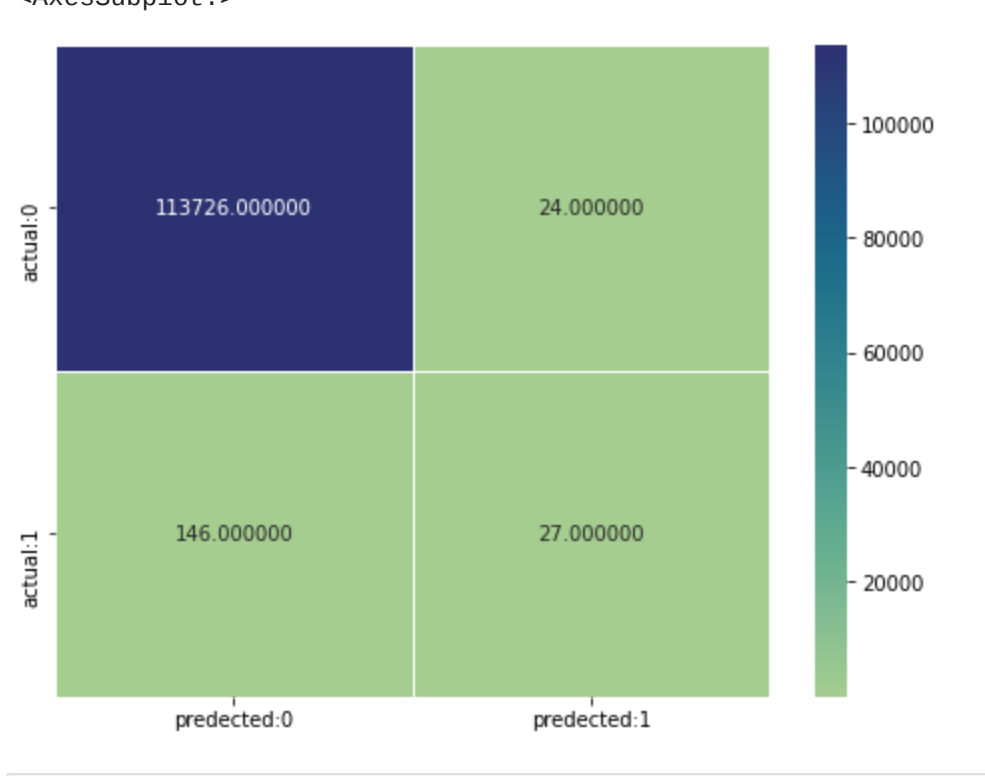
```python
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    113750
           1       0.53      0.16      0.24       173

    accuracy                           1.00    113923
   macro avg       0.76      0.58      0.62    113923
weighted avg       1.00      1.00      1.00    113923
```

In [24]:

```python
# Confusion Matrix
cm = confusion_matrix(y_test,y_pred)
conf_matrix = pd.DataFrame(data = cm,columns =['predected:0','predected:1'], index=['actual:0','actual:1'])
plt.figure(figsize=(8,6))
sns.heatmap(conf_matrix, annot = True, fmt = '1f', cmap = "crest", linewidth=.5)
```

Out[24]: <AxesSubplot:>



In [25]:

```python
from sklearn.ensemble import RandomForestClassifier
classfier = RandomForestClassifier(criterion = 'gini', max_depth =10, min_samples_split = 5, min_samples_leaf = 1)
classfier. fit(x_train,y_train)
```

Out[25]:

```
▾ RandomForestClassifier
RandomForestClassifier(max_depth=10, min_samples_split=5)
```

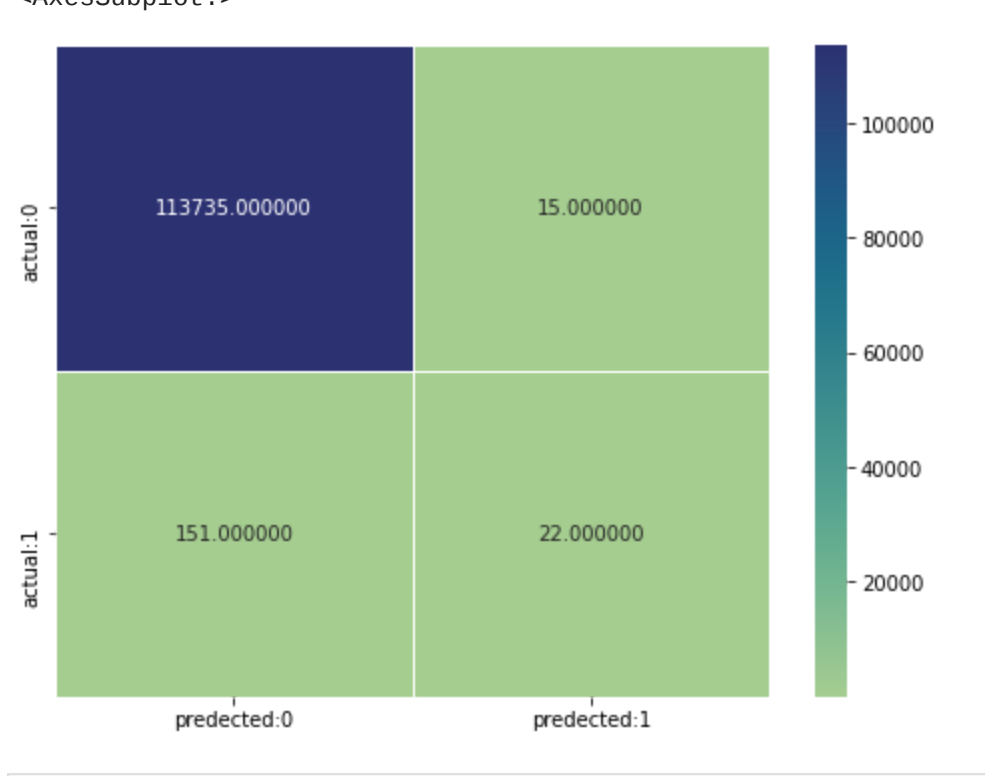
In [26]:

```python
y_pred=classfier.predict(x_test)
```

In [27]:

```python
#Confusion Matrix
cm = confusion_matrix(y_test,y_pred)
conf_matrix = pd.DataFrame(data = cm,columns =['predected:0','predected:1'], index=['actual:0','actual:1'])
plt.figure(figsize=(8,6))
sns.heatmap(conf_matrix, annot = True, fmt = '1f', cmap = "crest", linewidth=.6)
```

Out[27]: <AxesSubplot:>



In [28]:

```python
print(accuracy_score(y_test,y_pred))
```

0.9985428754509625

In [29]:

```python
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    113750
           1       0.59      0.13      0.21       173

    accuracy                           1.00    113923
   macro avg       0.80      0.56      0.60    113923
weighted avg       1.00      1.00      1.00    113923
```

HANDLING INBALANCED DATA SET

In [30]:

```python
!python -m pip install imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in c:\users\majid aslam\anaconda3\lib\site-packages (0.9.1)
Requirement already satisfied: scikit-learn>=1.1.0 in c:\users\majid aslam\anaconda3\lib\site-packages (from imbalanced-learn) (1.3.2)
Requirement already satisfied: numpy>=1.17.3 in c:\users\majid aslam\anaconda3\lib\site-packages (from imbalanced-learn) (1.22.4)
Requirement already satisfied: joblib>=1.0.0 in c:\users\majid aslam\anaconda3\lib\site-packages (from imbalanced-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\majid aslam\anaconda3\lib\site-packages (from imbalanced-learn) (2.1.0)
Requirement already satisfied: scipy>=1.3.2 in c:\users\majid aslam\anaconda3\lib\site-packages (from imbalanced-learn) (1.6.2)
```

In [34]:

```python
#Sampling of data
y_train.value_counts()
```

Out[34]:

```
Class
0        170565
1           319
dtype: int64
```

In [43]:

```python
from imblearn.over_sampling import RandomOverSampler
roc=RandomOverSampler(random_state=1)
```

In [44]:

```python
x_train_rs,y_train_rs=roc.fit_resample(x_train,y_train)
```

In [102…

```python
classifier = RandomForestClassifier(n_estimators=100,max_depth=10)
classifier.fit(x_train_rs,y_train_rs)
```

Out[102…

```
▾ RandomForestClassifier
RandomForestClassifier(max_depth=10)
```

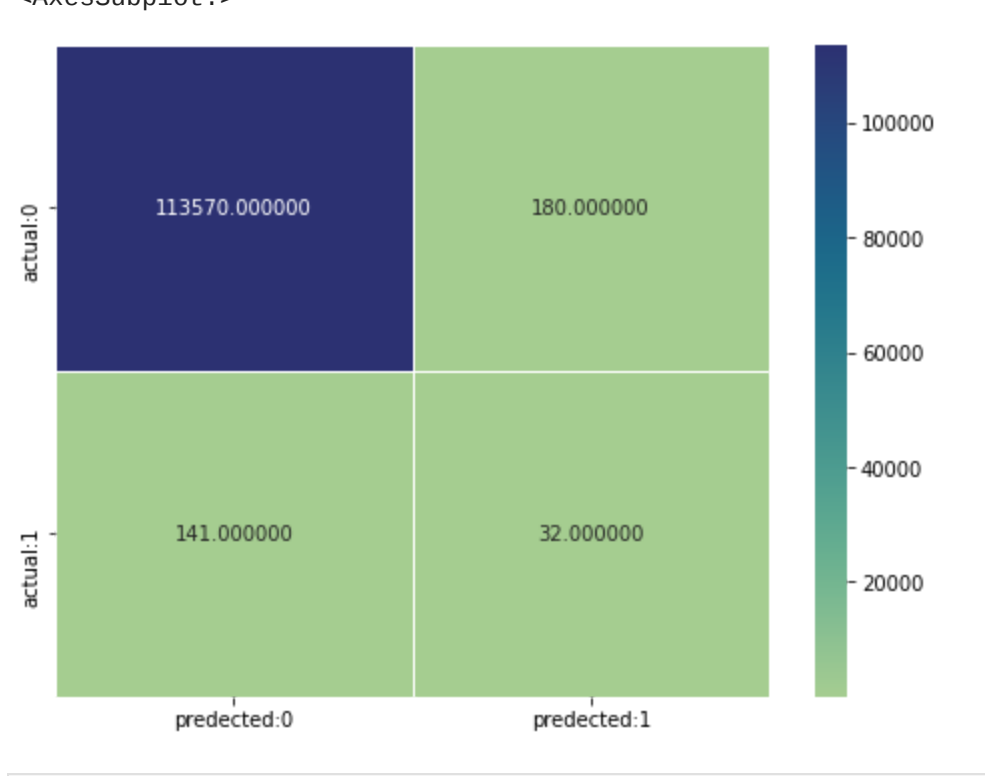
In [49]:

```python
y_predicd = classifier.predict(x_test)
```

In [50]:

```python
cm = confusion_matrix(y_test,y_predicd)
conf_matrix = pd.DataFrame(data = cm,columns =['predected:0','predected:1'], index=['actual:0','actual:1'])
plt.figure(figsize=(8,6))
sns.heatmap(conf_matrix, annot = True, fmt = '1f', cmap = "crest", linewidth=.6)
```

Out[50]: <AxesSubplot:>



In [52]:

```python
print(accuracy_score(y_test,y_predicd))
```

0.9971823073479456

In [53]:

```python
print(classification_report(y_test,y_predicd))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    113750
           1       0.15      0.18      0.17       173

    accuracy                           1.00    113923
   macro avg       0.57      0.59      0.58    113923
weighted avg       1.00      1.00      1.00    113923
```

In [76]:

```python
!pip install ada-boost
```

```
Collecting ada-boost
  Downloading ada_boost-0.0.1-py3-none-any.whl (1.7 kB)
Requirement already satisfied: pytz in c:\users\majid aslam\anaconda3\lib\site-packages (from ada-boost) (2021.1)
Installing collected packages: ada-boost
Successfully installed ada-boost-0.0.1
```

In [79]:

```python
RANDOM_STATE = 2020
NUM_ESTIMATIORS = 100
target = 'Class'
predectiors = ['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10', 'V11', 'V12', 'V13', 'V14', 'V15',
               'V16', 'V17', 'V18', 'V19', 'V20', 'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount']
```

In [80]:

```python
from sklearn.ensemble import AdaBoostClassifier
clf = AdaBoostClassifier(random_state=RANDOM_STATE,algorithm= 'SAMME.R',learning_rate=0.08,n_estimators=NUM_ESTIMATIORS)
```

In [81]:

```python
clf.fit(credit_card[predectiors],credit_card['Class'].values)
```

Out[81]:

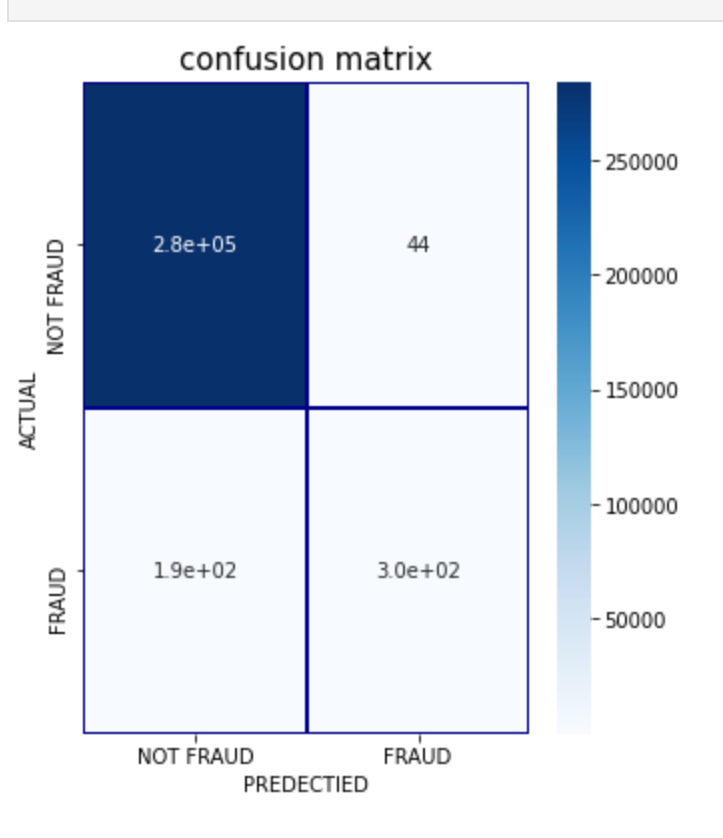
```
▾ AdaBoostClassifier
AdaBoostClassifier(learning_rate=0.08, n_estimators=100, random_state=2020)
```

In [82]:

```python
y_prede = clf.predict(credit_card[predectiors])
```

In [88]:

```python
cl = pd.crosstab(credit_card[target].values, y_prede, rownames= ['ACTUAL'], colnames =['PREDECTIED'])
fig, (ax1) = plt.subplots(ncols=1, figsize=(5,6))
sns.heatmap(cl, xticklabels= ['NOT FRAUD', 'FRAUD'], yticklabels= ['NOT FRAUD', 'FRAUD'],
            annot = True, ax = ax1, linewidths= .2, linecolor= 'Darkblue', cmap= 'Blues')
plt.title('confusion matrix', fontsize=15)
plt.show()
```



SMOTE ANALYSIS

In [97]:

```python
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state =2)
smote_x_train, smote_y_train = smote.fit_resample(x_train,y_train)
print("before sampling class distribuation", Counter(y_train))
print("after sampling class distribuation", Counter(smote_y_train))
```

```
before sampling class distribuation Counter({'Class': 1})
after sampling class distribuation Counter({'Class': 1})
```

In [103…

```python
classifier = RandomForestClassifier(n_estimators=100,max_depth=10)
classifier.fit(smote_x_train,smote_y_train)
```

Out[103…

```
▾ RandomForestClassifier
RandomForestClassifier(max_depth=10)
```

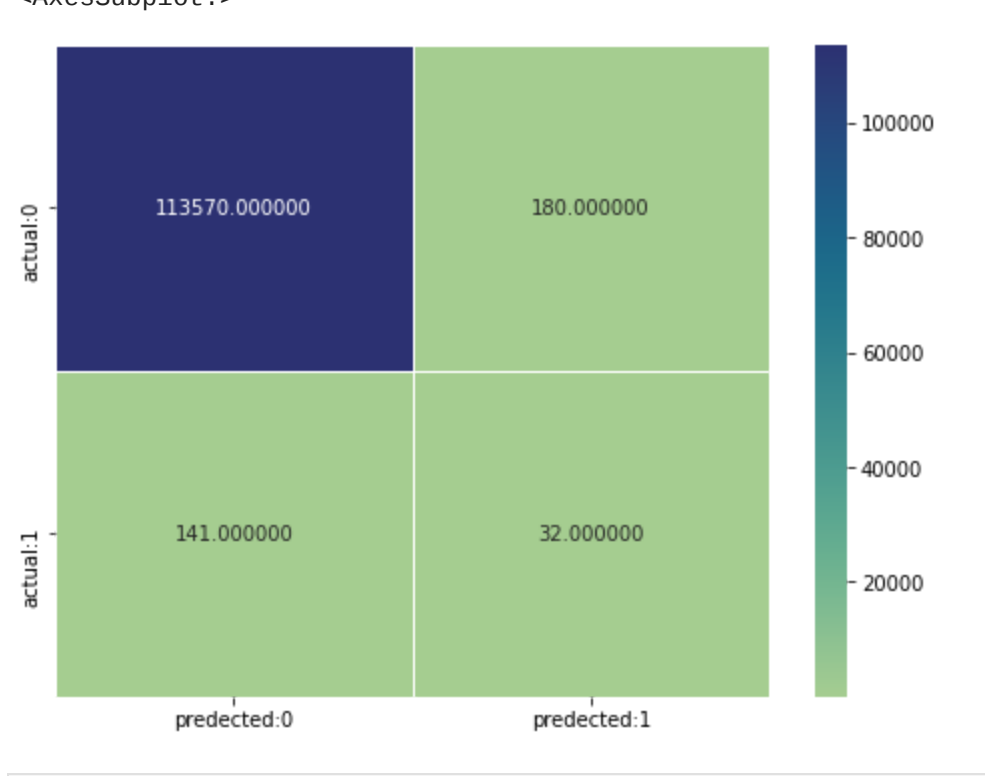
In [104…

```python
predictions = classifier.predict(x_test)
```

In [105…

```python
cm = confusion_matrix(y_test,y_predicd)
conf_matrix = pd.DataFrame(data = cm,columns =['predected:0','predected:1'], index=['actual:0','actual:1'])
plt.figure(figsize=(8,6))
sns.heatmap(conf_matrix, annot = True, fmt = '1f', cmap = "crest", linewidth=.6)
```

Out[105…: <AxesSubplot:>



In [108…

```python
print(accuracy_score(y_test,predictions))
```

0.9140823187591619

In [109…

```python
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       1.00      0.91      0.96    113750
           1       0.01      0.75      0.03       173

    accuracy                           0.91    113923
   macro avg       0.51      0.83      0.49    113923
weighted avg       1.00      0.91      0.95    113923
```

In [ ]: