Credit Card Fraud Prediction

In [5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, RobustScaler
from sklearn.metrics import accuracy_score, confusion_matrix, classification_repo
import warnings
warnings.filterwarnings("ignore")
```

In [6]:
```
dataset = pd.read_csv('creditcard.csv')
```

In [7]:
```
dataset.head()
```

Out[7]:

	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	0.098698	0.3
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.2
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.5
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.3
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.8

5 rows × 31 columns

In [8]:
```
print(dataset.shape)
```

```
(284806, 31)
```

In [9]:
```
print("Total Null values in the dataset:",dataset.isnull().sum().sum())
```

```
Total Null values in the dataset: 0
```

In [10]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284806 entries, 0 to 284805
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284806 non-null  float64
 1   V1      284806 non-null  float64
 2   V2      284806 non-null  float64
 3   V3      284806 non-null  float64
 4   V4      284806 non-null  float64
 5   V5      284806 non-null  float64
 6   V6      284806 non-null  float64
 7   V7      284806 non-null  float64
 8   V8      284806 non-null  float64
 9   V9      284806 non-null  float64
 10  V10     284806 non-null  float64
 11  V11     284806 non-null  float64
 12  V12     284806 non-null  float64
 13  V13     284806 non-null  float64
 14  V14     284806 non-null  float64
 15  V15     284806 non-null  float64
 16  V16     284806 non-null  float64
 17  V17     284806 non-null  float64
 18  V18     284806 non-null  float64
 19  V19     284806 non-null  float64
 20  V20     284806 non-null  float64
 21  V21     284806 non-null  float64
 22  V22     284806 non-null  float64
 23  V23     284806 non-null  float64
 24  V24     284806 non-null  float64
 25  V25     284806 non-null  float64
 26  V26     284806 non-null  float64
 27  V27     284806 non-null  float64
 28  V28     284806 non-null  float64
 29  Amount  284806 non-null  float64
 30  Class   284806 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

In [11]:
```
duplicate_rows = dataset[dataset.duplicated()]
print("Number of duplicate rows: ", duplicate_rows.shape )
```

```
Number of duplicate rows:  (1081, 31)
```

In [12]:
```
dataset = dataset.drop_duplicates()
print(dataset.shape)
```

```
(283725, 31)
```

In [13]:
```
dataset.describe()
```

Out[13]:

	Time	V1	V2	V3	V4	V5
count	283725.000000	283725.000000	283725.000000	283725.000000	283725.000000	283725.000000
mean	94810.802753	0.005919	-0.004134	0.001611	-0.002965	0.001828
std	47480.905865	1.948029	1.646706	1.508684	1.414186	1.377011
min	0.000000	-56.407510	-72.715728	-48.325589	-5.683171	-113.743307
25%	54204.000000	-0.915954	-0.600324	-0.889684	-0.850137	-0.689836
50%	84692.000000	0.020386	0.063952	0.179958	-0.022240	-0.053469
75%	139298.000000	1.316069	0.800283	1.026962	0.739654	0.612223
max	172788.000000	2.454930	22.057729	9.382558	16.875344	34.801666

8 rows × 31 columns

In [14]:
```
print(dataset['Class'].value_counts())
```

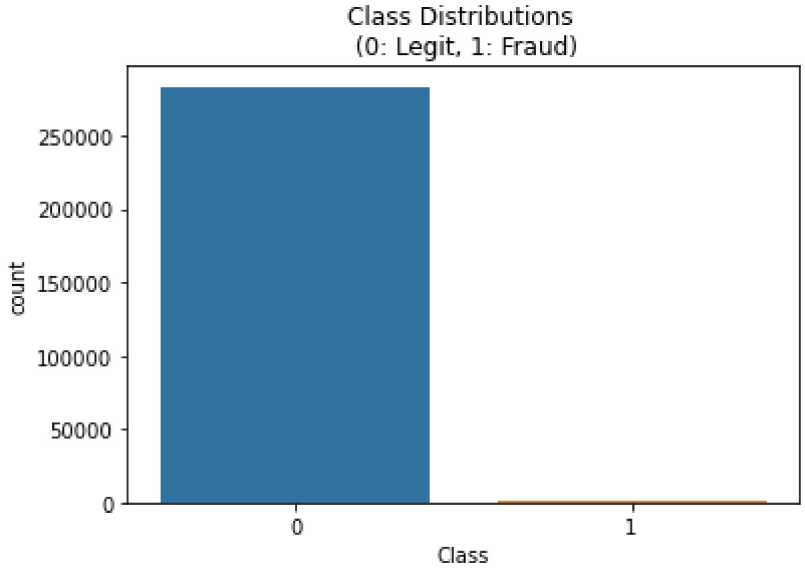
```
0    283252
1       473
Name: Class, dtype: int64
```

In [15]:
```
print('Legit Transactions:', round(dataset['Class'].value_counts()[0]/len(dataset
print('Frauds Transactions:', round(dataset['Class'].value_counts()[1]/len(datase
```

```
Legit Transactions: 99.83 %
Frauds Transactions: 0.17 %
```

In [16]:
```
sns.countplot('Class', data=dataset)
plt.title('Class Distributions \n (0: Legit, 1: Fraud)')
```

Out[16]:
```
Text(0.5, 1.0, 'Class Distributions \n (0: Legit, 1: Fraud)')
```



In [17]:
```
std_scaler = StandardScaler()
rob_scaler = RobustScaler()

dataset['scaled_amount'] = rob_scaler.fit_transform(dataset['Amount'].values.resh
dataset.drop(['Amount'], axis=1, inplace=True)
```

In [18]:
```
scaled_amount = dataset['scaled_amount']

dataset.drop(['scaled_amount'], axis=1, inplace=True)
dataset.insert(0, 'scaled_amount', scaled_amount)

dataset.head()
```

Out[18]:

	scaled_amount	Time	V1	V2	V3	V4	V5	V6	V7
0	1.774718	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599
1	-0.268530	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803
2	4.959811	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461
3	1.411487	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609
4	0.667362	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941

5 rows × 31 columns

In [19]:
```
x = dataset.drop(['Time','Class'], axis=1)
y = dataset['Class']
```

In [20]:
```
dataset = dataset.sample(frac=1)

fraud_dataset = dataset.loc[dataset['Class'] == 1]                          # L
legit_dataset = dataset.loc[dataset['Class'] == 0][:473]                    # A

normal_distributed_dataset = pd.concat([fraud_dataset, legit_dataset])      # C

new_dataset = normal_distributed_dataset.sample(frac=1, random_state=42)    # S

new_dataset.head()
```

Out[20]:

	scaled_amount	Time	V1	V2	V3	V4	V5	V
151006	-0.292032	94362.0	-26.457745	16.497472	-30.177317	8.904157	-17.892600	-1.2279
149600	3.326658	91554.0	-5.100256	3.633442	-3.843919	0.183208	-1.183997	1.6021
198030	-0.278404	132292.0	0.252563	0.622609	0.177797	-0.247886	0.447171	-0.9936
39183	10.496871	39729.0	-0.964567	-1.643541	-0.187727	1.158253	-2.458336	0.8522
93486	-0.305938	64443.0	1.079524	0.872988	-0.303850	2.755369	0.301688	-0.3502

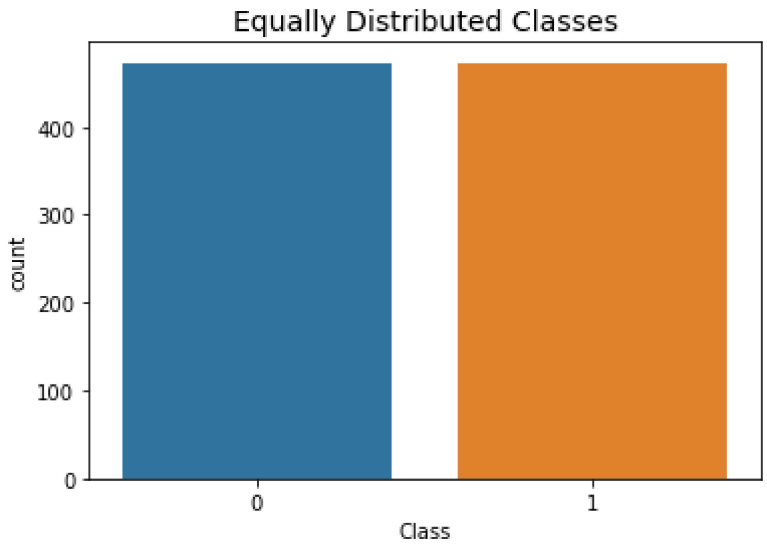
5 rows × 31 columns

In [21]:
```
new_dataset.shape
```

Out[21]:
```
(946, 31)
```

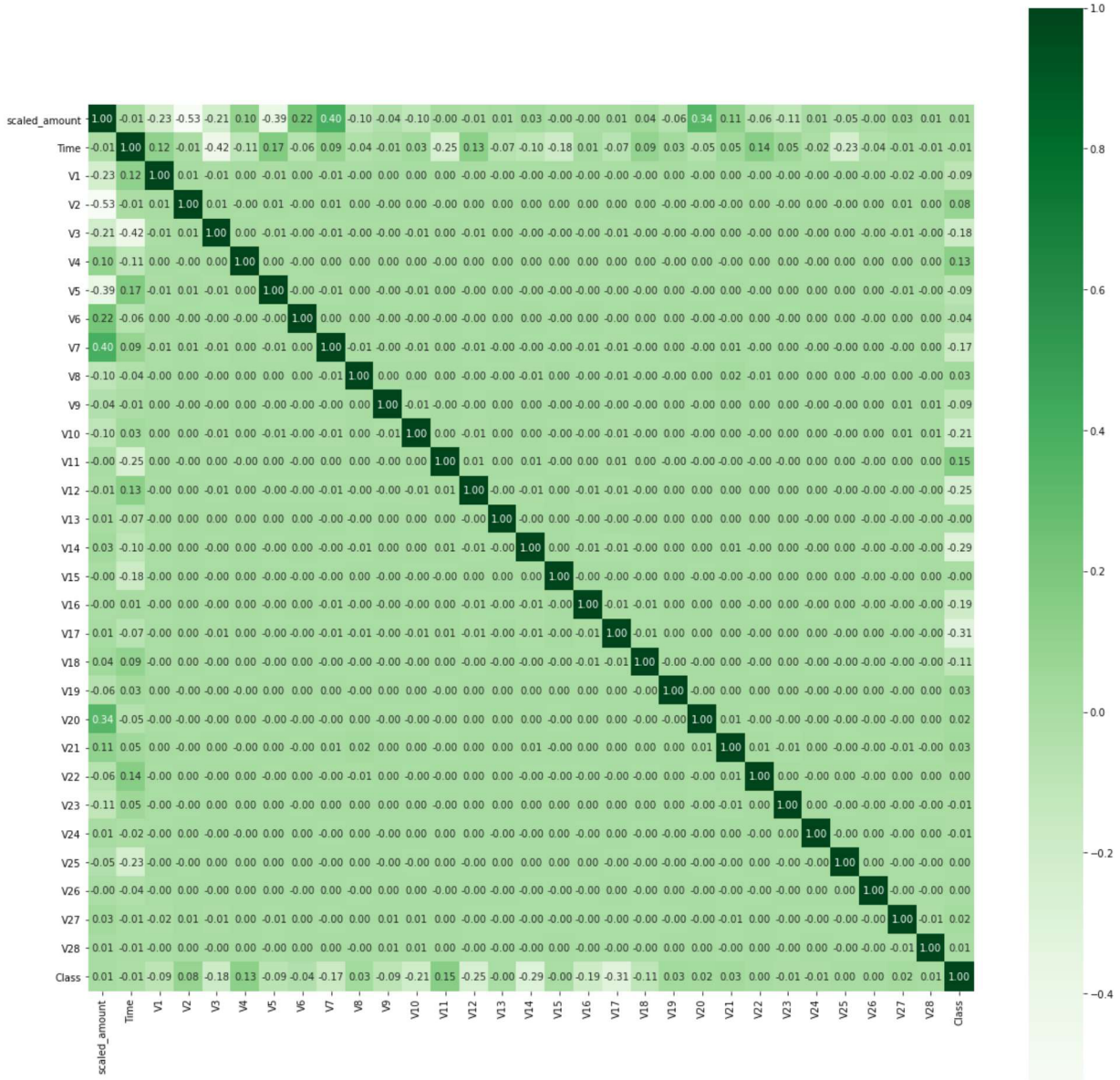
In [22]:
```
print('Distribution of the Classes in the subsample dataset')
print(new_dataset['Class'].value_counts()/len(new_dataset))

sns.countplot('Class', data=new_dataset)
plt.title('Equally Distributed Classes', fontsize=14)
plt.show()
```

```
Distribution of the Classes in the subsample dataset
0    0.5
1    0.5
Name: Class, dtype: float64
```



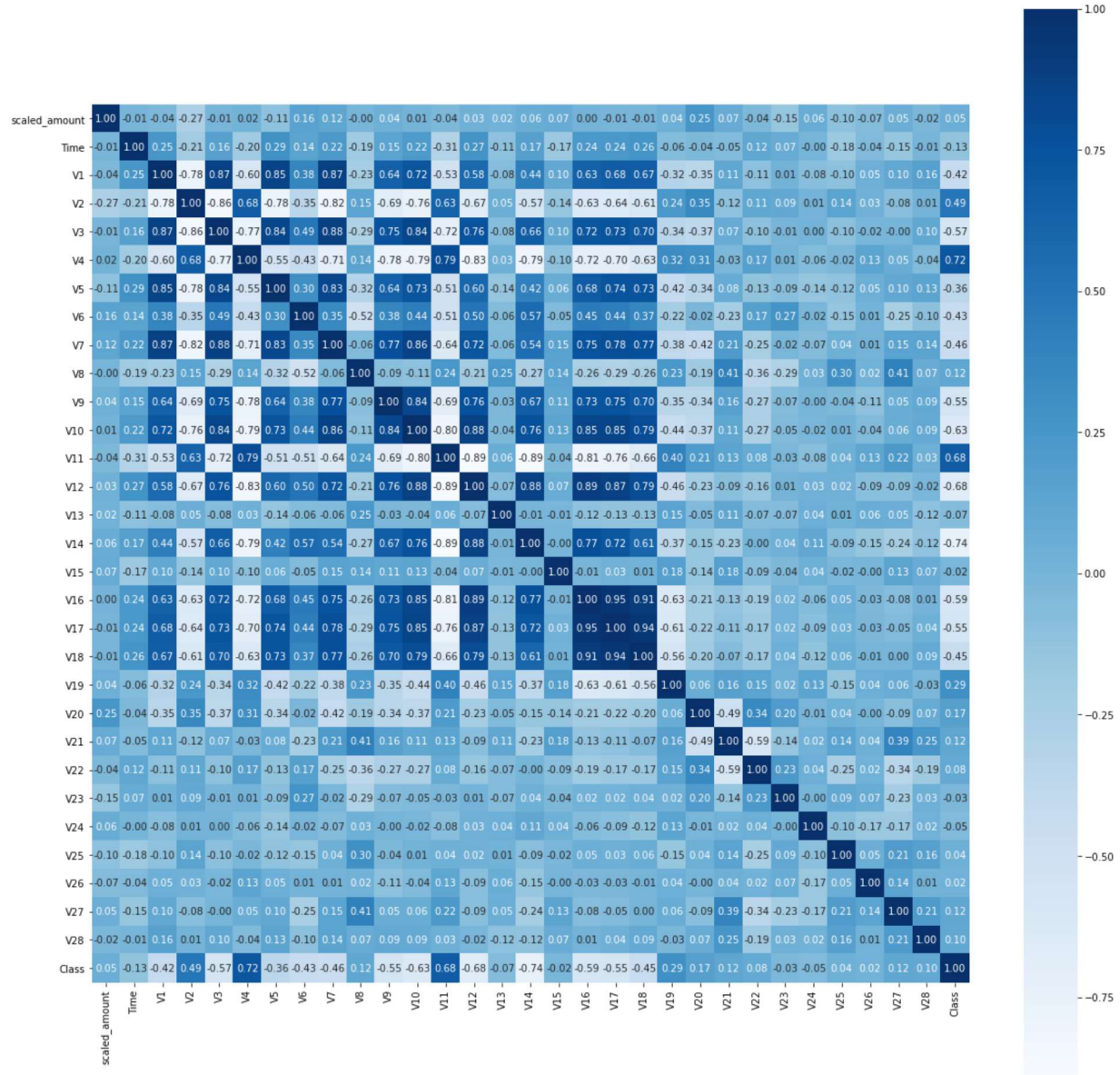
In [23]:
```
plt.subplots(figsize=(20,20))
sns.heatmap(dataset.corr(), cbar=True, square=True, fmt='.2f', annot=True, annot_
```

Out[23]: <AxesSubplot:>



In [25]:
```
plt.subplots(figsize=(20,20))
sns.heatmap(new_dataset.corr(), cbar=True, square=True, fmt='.2f', annot=True, ar
```

Out[25]: <AxesSubplot:>



In [26]:
```
x = new_dataset.drop(['Time', 'Class'], axis=1)
y = new_dataset['Class']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_s
print(x_train.shape, x_test.shape)
```

```
(756, 29) (190, 29)
```

In [27]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train, y_train)
```

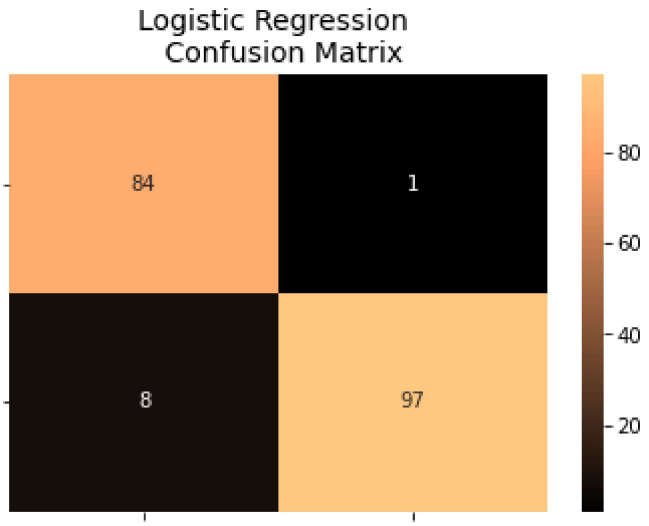
Out[27]:
```
LogisticRegression()
```

In [28]:
```
y_pred = model.predict(x_test)
print("Accuracy score:",(round(accuracy_score(y_test, y_pred)*100, 2)),'%')
```

```
Accuracy score: 95.26 %
```

In [29]:
```
log_reg_cf = confusion_matrix(y_test, y_pred)
fig, ax = plt.subplots(figsize=(6,4))
sns.heatmap(log_reg_cf, annot=True, cmap=plt.cm.copper)
ax.set_title("Logistic Regression \n Confusion Matrix", fontsize=14)
ax.set_xticklabels(['', ''], fontsize=14, rotation=90)
ax.set_yticklabels(['', ''], fontsize=14, rotation=360)
```

Out[29]:
```
[Text(0, 0.5, ''), Text(0, 1.5, '')]
```



In [30]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.99      0.95        85
           1       0.99      0.92      0.96       105

    accuracy                           0.95       190
   macro avg       0.95      0.96      0.95       190
weighted avg       0.96      0.95      0.95       190
```

Create a test case and generate a predicted result from the system

In [31]:
```
test_case_list = [239.93, -2.303349568, 1.75924746, -0.359744743, 2.330243051, -0

test_case_df = pd.DataFrame(test_case_list)              # converting the list t
test_case = test_case_df.T                               # the shape of current
test_case_prediction = model.predict(test_case)          # predicting the qualit
if(test_case_prediction == 0):                           # generating the result
    print('The Transaction was Legit')
else:
    print('The Transaction was Fraud.')
```

```
The Transaction was Fraud.
```

In [ ]: