Importing Necessary Libraries

In [57]:
```
import sys
import numpy
import pandas
import matplotlib
import seaborn
import scipy

print('Python: {}'.format(sys.version))
print('Numpy: {}'.format(numpy.__version__))
print('Pandas: {}'.format(pandas.__version__))
print('Matplotlib: {}'.format(matplotlib.__version__))
print('Seaborn: {}'.format(seaborn.__version__))
print('Scipy: {}'.format(scipy.__version__))
```

```
Python: 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit
(AMD64)]
Numpy: 1.21.6
Pandas: 1.4.4
Matplotlib: 3.7.2
Seaborn: 0.11.2
Scipy: 1.9.1
```

import the necessary packages

In [58]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

loading the data set

In [59]:
```

data = pd.read_csv("C:/Users/sykam/OneDrive/Desktop/final.csv")
```

In [60]:
```
print(data.columns)
```

```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V
9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V
19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'A
mount',
       'Class'],
      dtype='object')
```

In [61]:
```
data = data.sample(frac=0.1, random_state = 1)
print(data.shape)
print(data.describe())
```

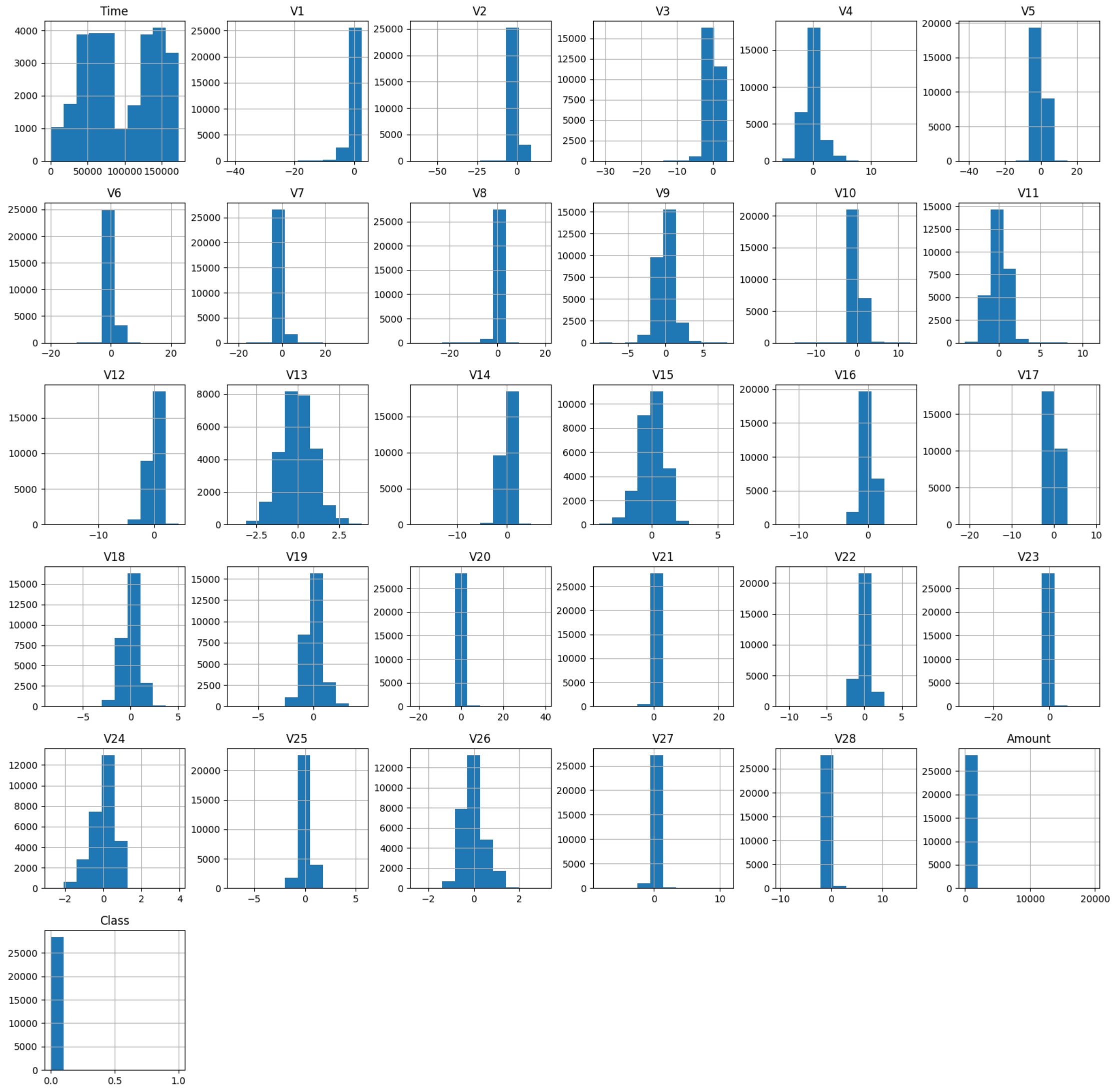
```
(28481, 31)
                Time            V1            V2            V3            
V4  \
count   28481.000000  28481.000000  28481.000000  28481.000000  2
8481.000000
mean    94705.035216     -0.001143     -0.018290      0.000795      
0.000350
std     47584.727034      1.994661      1.709050      1.522313      
1.420003
min         0.000000    -40.470142    -63.344698    -31.813586      
-5.266509
25%     53924.000000     -0.908809     -0.610322     -0.892884      
-0.847370
50%     84551.000000      0.031139      0.051775      0.178943      
-0.017692
75%    139392.000000      1.320048      0.792685      1.035197      
0.737312
max    172784.000000      2.411499     17.418649      4.069865     
16.715537

                 V5            V6            V7            V8            
V9  \
count  28481.000000  28481.000000  28481.000000  28481.000000  28
481.000000
mean      -0.015666      0.003634     -0.008523     -0.003040      
0.014536
std        1.395552      1.334985      1.237249      1.204102      
1.098006
min      -42.147898    -19.996349    -22.291962    -33.785407      
-8.739670
25%       -0.703986     -0.765807     -0.562033     -0.208445      
-0.632488
50%       -0.068037     -0.269071      0.028378      0.024696      
-0.037100
75%        0.603574      0.398839      0.559428      0.326057      
0.621093
max       28.762671     22.529298     36.677268     19.587773      
8.141560

       ...           V21           V22           V23           V2
4  \
count  ...  28481.000000  28481.000000  28481.000000  28481.00000
0
mean   ...      0.004740      0.006719     -0.000494     -0.00262
6
std    ...      0.744743      0.728209      0.645945      0.60396
8
min    ...    -16.640785    -10.933144    -30.269720     -2.75226
3
25%    ...     -0.224842     -0.535877     -0.163047     -0.36058
2
50%    ...     -0.029075      0.014337     -0.012678      0.03838
3
75%    ...      0.189068      0.533936      0.148065      0.43485
1
```

```
max     ...      22.588989      6.090514     15.626067      3.94452
0

                  V25          V26          V27          V28
Amount  \
count  28481.000000  28481.000000  28481.000000  28481.000000  28
481.000000
mean      -0.000917      0.004762     -0.001689     -0.004154
89.957884
std        0.520679      0.488171      0.418304      0.321646
270.894630
min       -7.025783     -2.534330     -8.260909     -9.617915
0.000000
25%       -0.319611     -0.328476     -0.071712     -0.053379
5.980000
50%        0.015231     -0.049750      0.000914      0.010753
22.350000
75%        0.351466      0.253580      0.090329      0.076267
78.930000
max        5.541598      3.118588     11.135740     15.373170  19
656.530000

                Class
count  28481.000000
mean       0.001720
std        0.041443
min        0.000000
25%        0.000000
50%        0.000000
75%        0.000000
max        1.000000

[8 rows x 31 columns]
```

Plot histograms of each parameter

In [62]:

```
data.hist(figsize = (20, 20))
plt.show()
```



Determine number of fraud cases in dataset

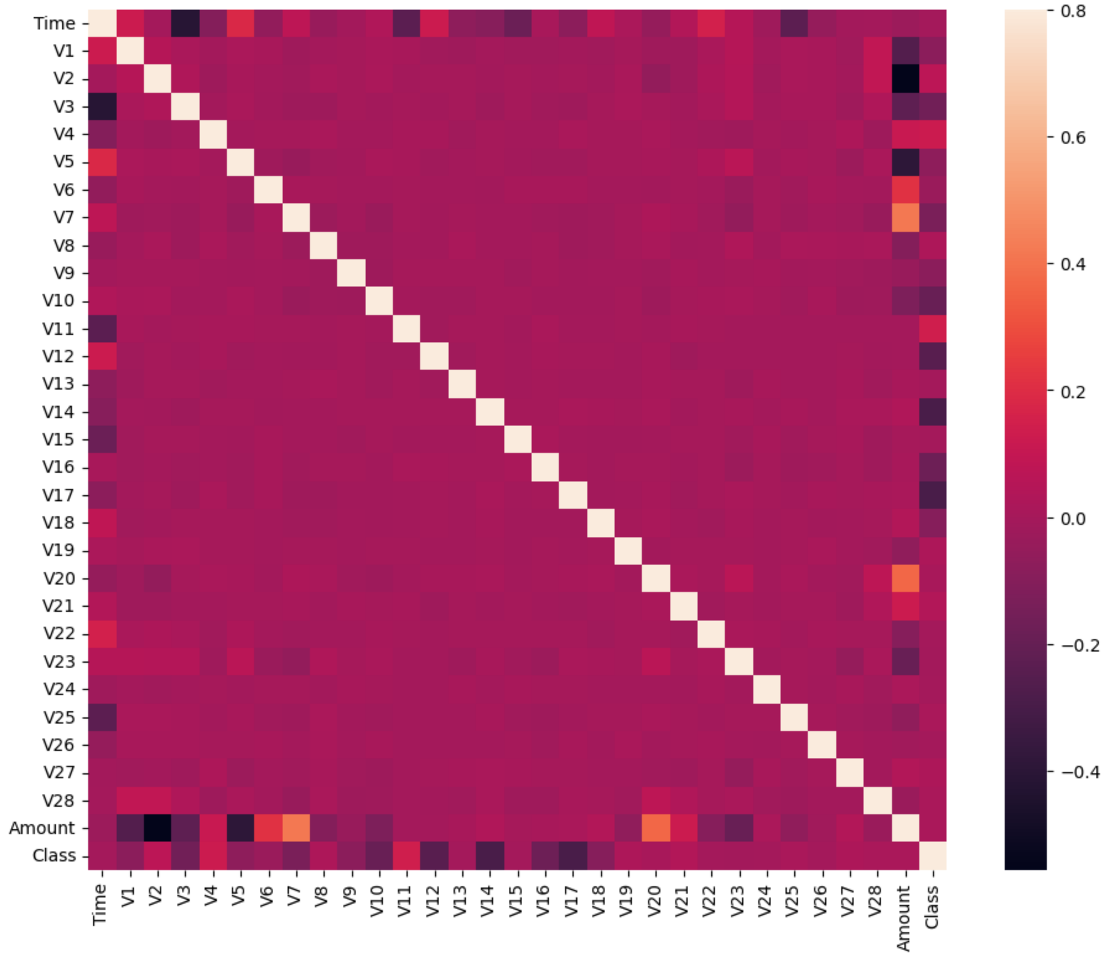
In [63]:

```
Fraud = data[data['Class'] == 1]
Valid = data[data['Class'] == 0]

outlier_fraction = len(Fraud)/float(len(Valid))
print(outlier_fraction)

print('Fraud Cases: {}'.format(len(data[data['Class'] == 1])))
print('Valid Transactions: {}'.format(len(data[data['Class'] == 0])
```

```
0.0017234102419808666
Fraud Cases: 49
Valid Transactions: 28432
```

Correlation matrix

In [64]:
```
corrmat = data.corr()
fig = plt.figure(figsize = (12, 9))

sns.heatmap(corrmat, vmax = .8, square = True)
plt.show()
```



In [65]:
```
columns = data.columns.tolist()


columns = [c for c in columns if c not in ["Class"]]


target = "Class"

X = data[columns]
Y = data[target]


print(X.shape)
print(Y.shape)
```

```
(28481, 30)
(28481,)
```

In [66]:
```
from sklearn.metrics import classification_report, accuracy_score
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
state = 1

classifiers = {
    "Isolation Forest": IsolationForest(max_samples=len(X),
                                        contamination=outlier_fract
                                        random_state=state),
    "Local Outlier Factor": LocalOutlierFactor(
        n_neighbors=20,
        contamination=outlier_fraction)}
```

fitting the model

In [67]:

```python
plt.figure(figsize=(9, 7))
n_outliers = len(Fraud)


for i, (clf_name, clf) in enumerate(classifiers.items()):


    if clf_name == "Local Outlier Factor":
        y_pred = clf.fit_predict(X)
        scores_pred = clf.negative_outlier_factor_
    else:
        clf.fit(X)
        scores_pred = clf.decision_function(X)
        y_pred = clf.predict(X)


    y_pred[y_pred == 1] = 0
    y_pred[y_pred == -1] = 1

    n_errors = (y_pred != Y).sum()


    print('{}: {}'.format(clf_name, n_errors))
    print(accuracy_score(Y, y_pred))
    print(classification_report(Y, y_pred))
```

```
M:\Anaconda\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but IsolationForest was fitted
with feature names
  warnings.warn(

Isolation Forest: 71
0.99750711000316
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28432
           1       0.28      0.29      0.28        49

    accuracy                           1.00     28481
   macro avg       0.64      0.64      0.64     28481
weighted avg       1.00      1.00      1.00     28481

Local Outlier Factor: 97
0.9965942207085425
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28432
           1       0.02      0.02      0.02        49

    accuracy                           1.00     28481
   macro avg       0.51      0.51      0.51     28481
weighted avg       1.00      1.00      1.00     28481
```

```
<Figure size 900x700 with 0 Axes>
```

In [89]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df=pd.read_csv("C:/Users/sykam/OneDrive/Desktop/final.csv")
df
```

Out[89]:

	Time	V1	V2	V3	V4	V5	V6
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921
...	...	...	...	...	...	...	...
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617

284807 rows × 31 columns

In [90]:
```python
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
trans=['V1','V2','V3','V4','V5','V6']
for i in trans:
    df[i]=le.fit_transform(df[i])
df
```

Out[90]:

ıe	V1	V2	V3	V4	V5	V6	V7	V8	V9
.0	44927	120597	270845	243075	105666	211699	0.239599	0.098698	0.363787
.0	190302	160387	134831	182517	151838	161419	-0.078803	0.085102	-0.255425
.0	44981	28959	251912	175551	87587	253504	0.791461	0.247676	-1.514654
.0	67133	107604	252745	68888	143697	244159	0.237609	0.377436	-1.387024
.0	55017	212076	240739	177853	97920	181128	0.592941	-0.270533	0.817739
...	...	...	...	...	...	...	...	...	...
.0	581	275585	270	16342	836	1621	-4.918215	7.305334	1.914428
.0	86023	122769	260716	79677	225505	238989	0.024330	0.294869	0.584800
.0	236103	96081	5048	97733	268447	261576	-0.296827	0.708417	0.432454
.0	128555	182902	180278	202316	101183	221235	-0.686180	0.679145	0.392087
.0	103428	107100	180354	102691	143454	84662	1.577006	-0.414650	0.486180

31 columns

In [91]:
```python
xdata=data.iloc[:,1:7]
ydata=data.iloc[:,30]
```

In [92]:
```
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  int64  
 2   V2      284807 non-null  int64  
 3   V3      284807 non-null  int64  
 4   V4      284807 non-null  int64  
 5   V5      284807 non-null  int64  
 6   V6      284807 non-null  int64  
 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(24), int64(7)
memory usage: 67.4 MB
```

In [93]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(xdata, ydata,te
from sklearn.linear_model import LinearRegression
clf = LinearRegression()
```

In [94]:
```
clf.fit(x_train,y_train)
```

Out[94]:
```
LinearRegression()
```

In [95]:
```
y_pred=clf.predict(x_test)
y_pred
```

Out[95]:
```
array([-0.00492341, -0.01526546,  0.00358697, ...,  0.00278612,
        0.00568916, -0.00406908])
```

In [96]:
```
k=y_pred
```

In [97]:
```
y_pred=[]
for i in range(len(k)):
    y_pred.append(int(k[i]))
y_pred
```

```
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
```

In [98]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_pred,y_test)
```

Out[98]:
```
0.998404085541015
```

In [99]:
```
clf.predict([[44981,28959,251912,175551,87587,253504]])
```

```
M:\Anaconda\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but LinearRegression was fitte
d with feature names
  warnings.warn(
```

Out[99]:
```
array([-964.54548725])
```

In [ ]:
```

```

In [100]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(x_train,y_train)
```

Out[100]:
```
KNeighborsClassifier(n_neighbors=10)
```

In [101]:
```
knn.score(x_test,y_test)
k=knn.predict(x_test)
k
```

```
M:\Anaconda\lib\site-packages\sklearn\neighbors\_classification.p
y:228: FutureWarning: Unlike other reduction functions (e.g. `ske
w`, `kurtosis`), the default behavior of `mode` typically preserv
es the axis it acts along. In SciPy 1.11.0, this behavior will ch
ange: the default value of `keepdims` will become False, the `axi
s` over which the statistic is taken will be eliminated, and the
value None will no longer be accepted. Set `keepdims` to True or
False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
M:\Anaconda\lib\site-packages\sklearn\neighbors\_classification.p
y:228: FutureWarning: Unlike other reduction functions (e.g. `ske
w`, `kurtosis`), the default behavior of `mode` typically preserv
es the axis it acts along. In SciPy 1.11.0, this behavior will ch
ange: the default value of `keepdims` will become False, the `axi
s` over which the statistic is taken will be eliminated, and the
value None will no longer be accepted. Set `keepdims` to True or
False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

Out[101]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [102]:
```
from sklearn.metrics import accuracy_score
accuracy_score(k,y_test)
```

Out[102]:
```
0.998723268432812
```

In [116]:
```
knn.predict([[-0.5894,1.5000,-0.50008,-0.90008,-0.500000,0.51210]])
```

```
M:\Anaconda\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but KNeighborsClassifier was f
itted with feature names
  warnings.warn(
M:\Anaconda\lib\site-packages\sklearn\neighbors\_classification.p
y:228: FutureWarning: Unlike other reduction functions (e.g. `ske
w`, `kurtosis`), the default behavior of `mode` typically preserv
es the axis it acts along. In SciPy 1.11.0, this behavior will ch
ange: the default value of `keepdims` will become False, the `axi
s` over which the statistic is taken will be eliminated, and the
value None will no longer be accepted. Set `keepdims` to True or
False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

Out[116]:
```
array([0], dtype=int64)
```

In [119]:

```
import pickle
path="C:/Users/sykam/OneDrive/Desktop/New folder/ssai.sav"
bo=pickle.dump(knn,open(path,'wb'))
```

In [ ]: