Credit Card Fraud Detection

April 21, 2019

1 CREDIT CARD FRAUD DETECTION

In [1]:
```
import sys
import numpy as np
import pandas as pd
import seaborn as sns
import scipy

print("Python {}".format(np.__version__))
```

```
Python 1.14.3
```

In [2]:
```
data = pd.read_csv('F:\Work\project\Kaggle\CCFraudDetection\creditcard.csv')
```

In [3]:
```
print(data.columns)
print(data.shape)
```

```
Index(['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',
       'Class'],
      dtype='object')
(284807, 31)
```

In [4]:
```
print(data.describe())
```

```
                Time            V1            V2            V3            V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean    94813.859575  3.919560e-15  5.688174e-16 -8.769071e-15  2.782312e-15
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+00
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-02
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-01
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00  1.687534e+01
```

```
                 V5            V6            V7            V8            V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean  -1.552563e-15  2.010663e-15 -1.694249e-15 -1.927028e-16 -3.137024e-15
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00  1.098632e+00
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -1.343407e+01
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -6.430976e-01
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -5.142873e-02
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01  5.971390e-01
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01  1.559499e+01

           ...                V21           V22           V23           V24  \
count      ...       2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean       ...       1.537294e-16  7.959909e-16  5.367590e-16  4.458112e-15
std        ...       7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01
min        ...      -3.483038e+01 -1.093314e+01 -4.480774e+01 -2.836627e+00
25%        ...      -2.283949e-01 -5.423504e-01 -1.618463e-01 -3.545861e-01
50%        ...      -2.945017e-02  6.781943e-03 -1.119293e-02  4.097606e-02
75%        ...       1.863772e-01  5.285536e-01  1.476421e-01  4.395266e-01
max        ...       2.720284e+01  1.050309e+01  2.252841e+01  4.584549e+00

                V25           V26           V27           V28         Amount  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  284807.000000
mean   1.453003e-15  1.699104e-15 -3.660161e-16 -1.206049e-16      88.349619
std    5.212781e-01  4.822270e-01  4.036325e-01  3.300833e-01     250.120109
min   -1.029540e+01 -2.604551e+00 -2.256568e+01 -1.543008e+01       0.000000
25%   -3.171451e-01 -3.269839e-01 -7.083953e-02 -5.295979e-02       5.600000
50%    1.659350e-02 -5.213911e-02  1.342146e-03  1.124383e-02      22.000000
75%    3.507156e-01  2.409522e-01  9.104512e-02  7.827995e-02      77.165000
max    7.519589e+00  3.517346e+00  3.161220e+01  3.384781e+01   25691.160000

              Class
count  284807.000000
mean        0.001727
std         0.041527
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000

[8 rows x 31 columns]
```

Check for any missing values

```
In [5]: data.isnull().values.any()
```

```
Out[5]: False
```

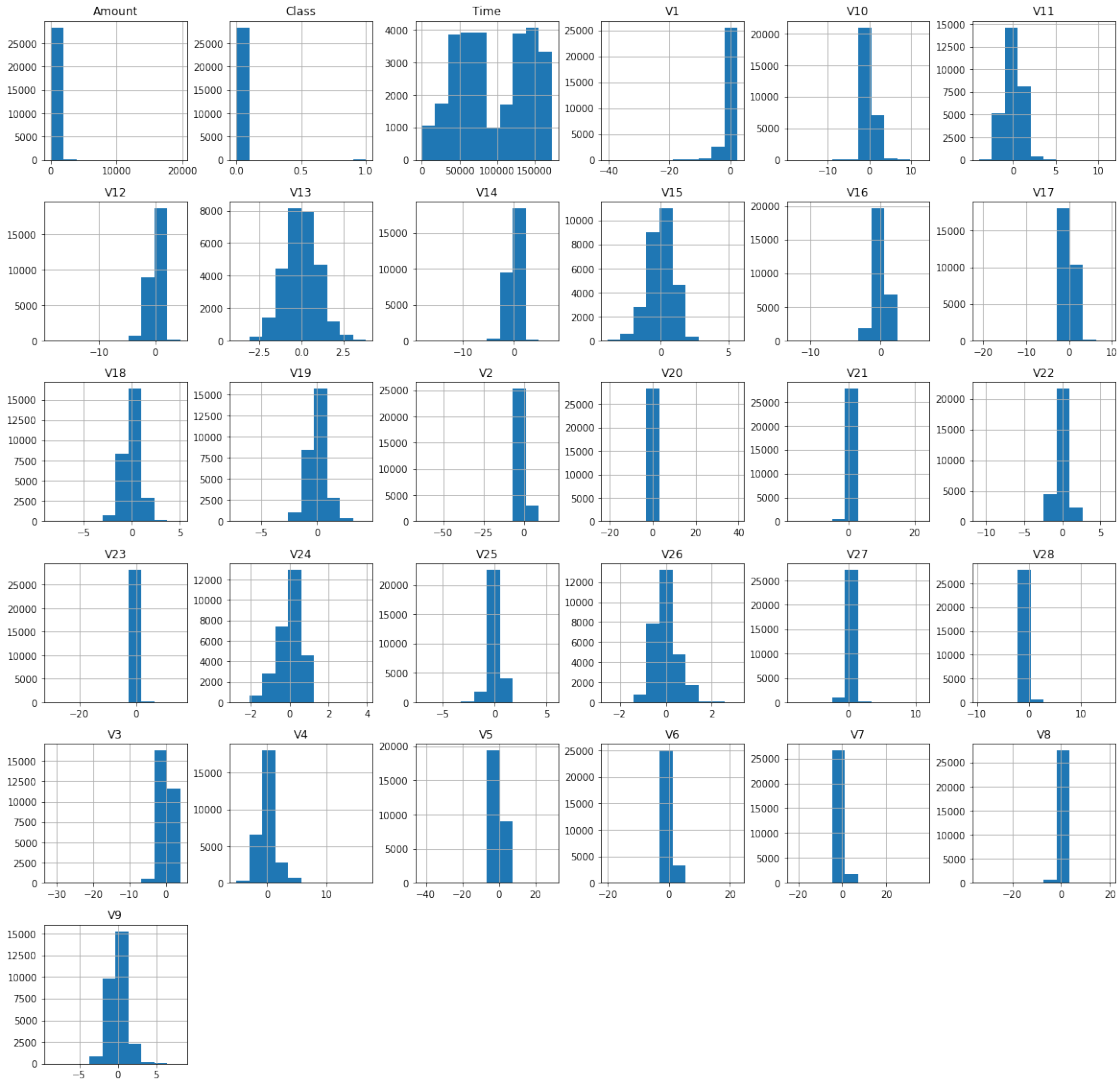
```
In [6]: data = data.sample(frac= 0.1,random_state = 1)
        print(data.shape)
```

```
(28481, 31)
```

VISUALIZING DATA

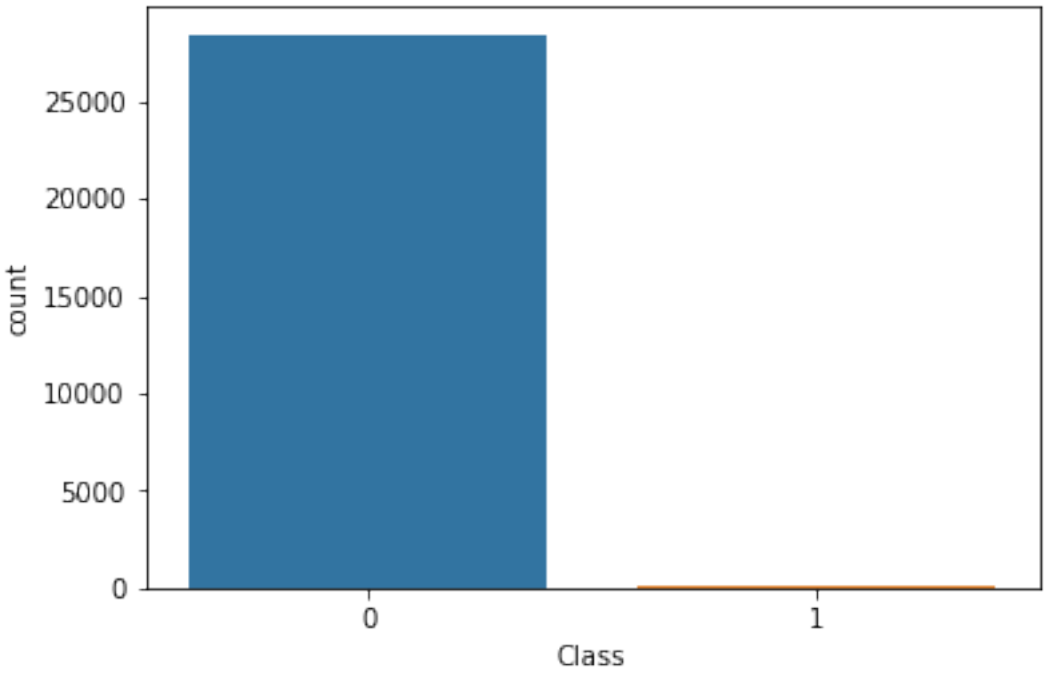
```
In [7]: #plot histogram
        import matplotlib.pyplot as plt
        data.hist(figsize=(20,20))
        plt.show
```

```
Out[7]: <function matplotlib.pyplot.show(*args, **kw)>
```



In [8]:
```
sns.countplot("Class",data=data)
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x1d1ea5c9588>



In [9]:
```
#Determining fraud cases
fraud = data[data['Class']==1]
valid = data[data['Class']==0]

outlier_fraction = float(len(fraud))/float(len(valid))
print("Number of fraud cases:{} \nNumber of valid cases:{} \nThe outlier_fraction is:{}
```

```
Number of fraud cases:49
Number of valid cases:28432
The outlier_fraction is:0.0017234102419808666
```

In [10]:
```
#Correlation Matrix
corrmat = data.corr()
print(corrmat)
```

```
            Time        V1        V2        V3        V4        V5        V6  \
Time    1.000000  0.126475 -0.001584 -0.413547 -0.104527  0.182205 -0.060483
V1      0.126475  1.000000  0.048796  0.015452 -0.010592  0.019888  0.006417
V2     -0.001584  0.048796  1.000000  0.027270 -0.022539  0.009666 -0.004411
V3     -0.413547  0.015452  0.027270  1.000000 -0.005423  0.013997 -0.006903
```

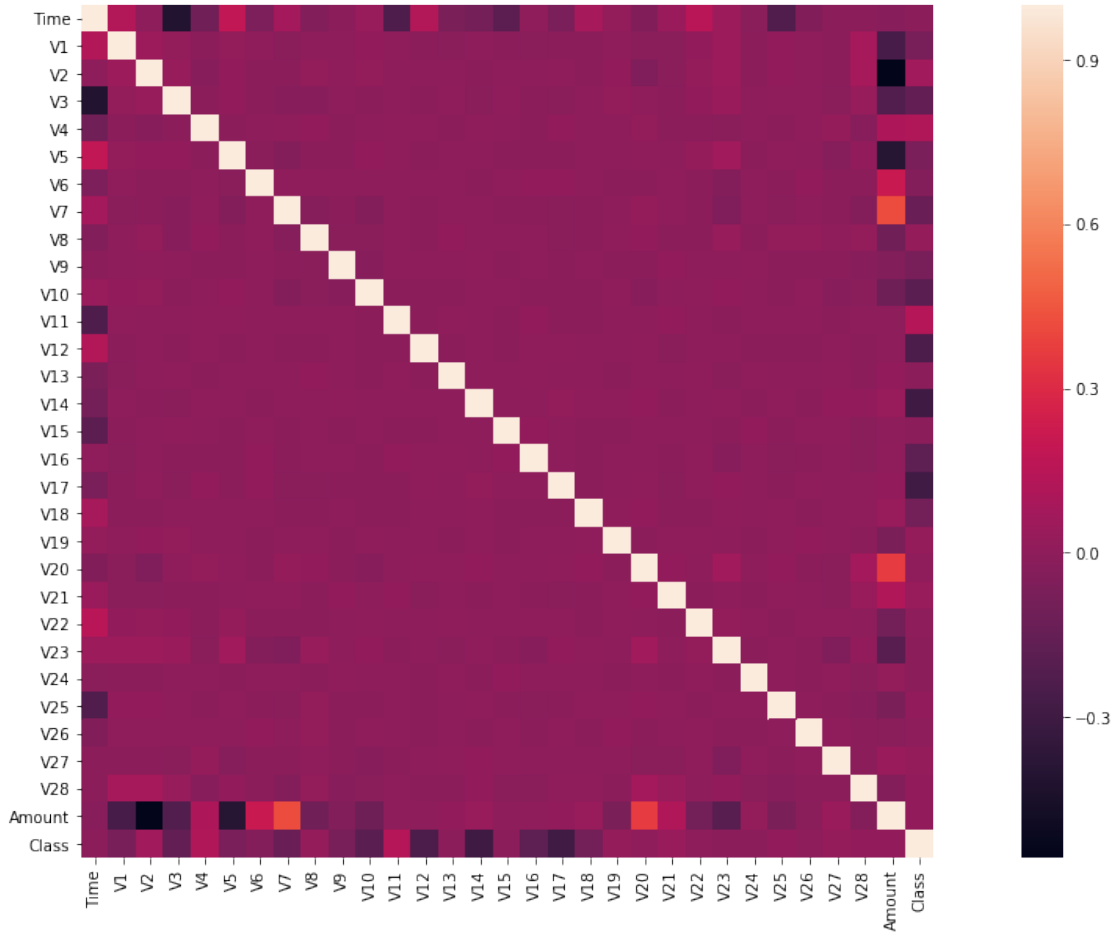
```
V4     -0.104527 -0.010592 -0.022539 -0.005423  1.000000 -0.003708  0.002029
V5      0.182205  0.019888  0.009666  0.013997 -0.003708  1.000000 -0.016656
V6     -0.060483  0.006417 -0.004411 -0.006903  0.002029 -0.016656  1.000000
V7      0.078924 -0.020583 -0.013456 -0.024640  0.004432 -0.037463  0.006923
V8     -0.040474 -0.003013  0.015662 -0.025529  0.011659 -0.013263  0.003695
V9     -0.008428  0.001658  0.003456  0.002525 -0.004395 -0.008506 -0.002762
V10     0.035939  0.010686  0.017218 -0.006955 -0.000669  0.011446 -0.003120
V11    -0.237613  0.007177 -0.002982  0.000836  0.007657  0.006286  0.001582
V12     0.126985 -0.012290  0.000646 -0.008236  0.005942 -0.011393 -0.000219
V13    -0.069353 -0.018849  0.001655  0.004030 -0.011637 -0.002728 -0.001591
V14    -0.093264 -0.001905 -0.006617 -0.016702  0.003485  0.001548 -0.007828
V15    -0.182255 -0.012884  0.002046  0.004421 -0.000575 -0.006731  0.006407
V16     0.007392 -0.015569 -0.003062 -0.012780 -0.004783 -0.014823  0.009237
V17    -0.074555 -0.009644  0.003567 -0.017722  0.012553 -0.013090  0.009201
V18     0.083959 -0.011822 -0.005922  0.001056  0.001852  0.000902 -0.002719
V19     0.019469  0.003860  0.011950  0.020282  0.001759  0.001468 -0.005205
V20    -0.050967 -0.017883 -0.054467  0.003068  0.015348  0.005350 -0.008991
V21     0.041323 -0.016415 -0.020127 -0.006083 -0.004423  0.002288  0.004490
V22     0.150603  0.014896  0.021923  0.014177 -0.011251  0.022065 -0.003705
V23     0.047941  0.049447  0.047591  0.042603 -0.017682  0.064703 -0.036726
V24    -0.020018 -0.003709 -0.011386 -0.001883  0.001829 -0.007184  0.001428
V25    -0.229491  0.014055  0.011838  0.005975 -0.009692  0.006493 -0.015012
V26    -0.048131  0.007203  0.005366  0.006869  0.004087  0.000048  0.009938
V27    -0.005541 -0.011545 -0.009611 -0.017094  0.024489 -0.027934 -0.004811
V28    -0.004339  0.085035  0.084873  0.029973 -0.024554  0.010991 -0.009772
Amount -0.026969 -0.262703 -0.556401 -0.225099  0.111692 -0.397437  0.213007
Class  -0.005087 -0.079820  0.069598 -0.160051  0.122631 -0.073519 -0.035085

              V7        V8        V9    ...         V21       V22       V23  \
Time    0.078924 -0.040474 -0.008428    ...    0.041323  0.150603  0.047941
V1     -0.020583 -0.003013  0.001658    ...   -0.016415  0.014896  0.049447
V2     -0.013456  0.015662  0.003456    ...   -0.020127  0.021923  0.047591
V3     -0.024640 -0.025529  0.002525    ...   -0.006083  0.014177  0.042603
V4      0.004432  0.011659 -0.004395    ...   -0.004423 -0.011251 -0.017682
V5     -0.037463 -0.013263 -0.008506    ...    0.002288  0.022065  0.064703
V6      0.006923  0.003695 -0.002762    ...    0.004490 -0.003705 -0.036726
V7      1.000000 -0.028291 -0.005510    ...    0.007012 -0.013871 -0.055242
V8     -0.028291  1.000000 -0.018645    ...   -0.005651 -0.004195  0.030092
V9     -0.005510 -0.018645  1.000000    ...    0.009462 -0.002297  0.002360
V10    -0.035149 -0.017995 -0.021718    ...    0.001434  0.006397  0.010754
V11     0.000863 -0.002562  0.000587    ...    0.009261 -0.000061 -0.004784
V12    -0.008740 -0.009387  0.002794    ...   -0.016599 -0.002456 -0.000431
V13     0.002445  0.012011  0.001733    ...    0.002637  0.004413 -0.017930
V14    -0.001300 -0.001876  0.004310    ...   -0.007459  0.001944  0.005999
V15    -0.003969 -0.000946 -0.014436    ...    0.000210  0.002279 -0.013669
V16    -0.010928 -0.000048  0.000286    ...   -0.007622  0.004838 -0.026575
V17    -0.017525 -0.020343 -0.007033    ...   -0.015777  0.005118  0.013149
V18    -0.015221 -0.013191 -0.001036    ...   -0.003275 -0.011108  0.008045
```

V19	0.002621	0.003453	-0.003865	...	0.004207	0.003725	0.000711
V20	0.023011	0.013911	-0.012547	...	0.014580	0.000622	0.064035
V21	0.007012	-0.005651	0.009462	...	1.000000	-0.011120	0.001835
V22	-0.013871	-0.004195	-0.002297	...	-0.011120	1.000000	0.015702
V23	-0.055242	0.030092	0.002360	...	0.001835	0.015702	1.000000
V24	0.002899	-0.008821	0.007441	...	-0.005780	0.005390	-0.009760
V25	-0.016941	0.017298	-0.009149	...	0.004783	-0.006371	0.001382
V26	-0.000075	0.015385	-0.003652	...	0.001042	0.008263	-0.005050
V27	-0.012973	0.008495	-0.011701	...	-0.016617	0.004071	-0.052343
V28	-0.037593	0.015525	-0.026290	...	0.035645	0.002156	0.011088
Amount	0.417814	-0.102221	-0.039773	...	0.121948	-0.095698	-0.192711
Class	-0.134247	0.024896	-0.079962	...	0.037570	-0.001683	-0.005856

	V24	V25	V26	V27	V28	Amount	Class
Time	-0.020018	-0.229491	-0.048131	-0.005541	-0.004339	-0.026969	-0.005087
V1	-0.003709	0.014055	0.007203	-0.011545	0.085035	-0.262703	-0.079820
V2	-0.011386	0.011838	0.005366	-0.009611	0.084873	-0.556401	0.069598
V3	-0.001883	0.005975	0.006869	-0.017094	0.029973	-0.225099	-0.160051
V4	0.001829	-0.009692	0.004087	0.024489	-0.024554	0.111692	0.122631
V5	-0.007184	0.006493	0.000048	-0.027934	0.010991	-0.397437	-0.073519
V6	0.001428	-0.015012	0.009938	-0.004811	-0.009772	0.213007	-0.035085
V7	0.002899	-0.016941	-0.000075	-0.012973	-0.037593	0.417814	-0.134247
V8	-0.008821	0.017298	0.015385	0.008495	0.015525	-0.102221	0.024896
V9	0.007441	-0.009149	-0.003652	-0.011701	-0.026290	-0.039773	-0.079962
V10	0.000734	-0.010712	0.006086	-0.025756	-0.017631	-0.122025	-0.191189
V11	-0.002600	-0.001712	-0.002694	-0.004893	-0.000608	-0.000394	0.140513
V12	-0.004571	-0.003269	-0.004400	0.002477	-0.008220	-0.000972	-0.244444
V13	0.006788	-0.002572	-0.000929	0.005606	-0.014200	0.009694	-0.003380
V14	-0.003827	0.006148	-0.004120	0.014247	0.011290	0.035498	-0.296297
V15	0.009640	-0.014448	0.004390	0.005639	-0.020654	0.000165	-0.003760
V16	0.000209	-0.017642	-0.014824	0.002792	-0.018434	0.008798	-0.175216
V17	0.001230	-0.010343	0.005719	0.000143	0.007173	0.012607	-0.293225
V18	-0.001235	0.004092	-0.007275	-0.002274	0.004075	0.038996	-0.098311
V19	-0.004049	0.003764	0.012193	-0.000812	-0.012631	-0.067748	0.025784
V20	-0.001015	0.013898	-0.008713	-0.015707	0.074731	0.367057	0.005640
V21	-0.005780	0.004783	0.001042	-0.016617	0.035645	0.121948	0.037570
V22	0.005390	-0.006371	0.008263	0.004071	0.002156	-0.095698	-0.001683
V23	-0.009760	0.001382	-0.005050	-0.052343	0.011088	-0.192711	-0.005856
V24	1.000000	0.001870	-0.010473	0.008380	-0.015790	0.017335	-0.003727
V25	0.001870	1.000000	0.000248	-0.014799	-0.024547	-0.064454	0.011958
V26	-0.010473	0.000248	1.000000	0.002964	-0.008006	-0.015768	-0.001884
V27	0.008380	-0.014799	0.002964	1.000000	-0.007671	0.039471	0.024421
V28	-0.015790	-0.024547	-0.008006	-0.007671	1.000000	-0.033855	0.014344
Amount	0.017335	-0.064454	-0.015768	0.039471	-0.033855	1.000000	0.012804
Class	-0.003727	0.011958	-0.001884	0.024421	0.014344	0.012804	1.000000

[31 rows x 31 columns]

In [11]:
```
#Visualization of this correlation
fig = plt.figure(figsize=(20,10))
sns.heatmap(corrmat,square='true')
plt.show()
```



In [12]:
```
#Getting Predictor variable and Target variable separate
column = data.columns.tolist()
column = [c for c in column if c!="Class"]
print(column)

target = "Class"
X=data[column]
y=data[target]

print("{}\n{}".format(X.shape,y.shape) )
```

```
['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10', 'V11', 'V12', 'V13', 'V14
(28481, 30)
```

```
(28481,)
```

1.0.1 APPLYING ALGORITHM

In [16]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
from sklearn.linear_model import LogisticRegression



#Define random state
state = 1

#Define the outlier detection methods
classifiers ={"Isolation Forest": IsolationForest(max_samples=len(X),contamination = c
              "Local Outlier Factor":LocalOutlierFactor(n_neighbors=20,contamination=c
```

In [17]:
```
#Fitting the model and evaluation

plt.figure(figsize=(10,8))
#n_outliers=(len(Fraud))


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

    # fit the data and tag outliers
    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)

    # Reshape the prediction values to 0 for valid, 1 for fraud.
    y_pred[y_pred == 1] = 0
    y_pred[y_pred == -1] = 1

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

    # Run classification metrics
    print('{}\n Number oferrors: {}'.format(clf_name, n_errors))
    print("Accuracy: ",accuracy_score(y, y_pred))
    print(classification_report(y, y_pred))
```

```
Isolation Forest
 Number oferrors: 71
```

```
Accuracy:  0.99750711000316
             precision    recall  f1-score   support

          0       1.00      1.00      1.00     28432
          1       0.28      0.29      0.28        49

avg / total       1.00      1.00      1.00     28481

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

          0       1.00      1.00      1.00     28432
          1       0.02      0.02      0.02        49

avg / total       1.00      1.00      1.00     28481



<Figure size 720x576 with 0 Axes>
```