Credit Card Fraud

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

In [2]:
```
# Load Data
```

In [3]:
```
data = pd.read_csv('creditcard.csv')
```

In [7]:
```
"Shape of data: ",data.shape
```

Out[7]:
```
('Shape of data: ', (284807, 31))
```

In [8]:
```
data.columns
```

Out[8]:
```
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')
```

In [9]:
```
data.head(5)
```

Out[9]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V
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	0.066
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	-0.339
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	-0.689
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	-1.175
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	0.141

5 rows × 31 columns

In [10]:
```
data.describe()
```

Out[10]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.84
mean	94813.859575	3.919560e-15	5.688174e-16	-8.769071e-15	2.782312e-15	-1.552563e-15	2.010663e-15	-1.694249e-15	-1.927028e-16	-3.1
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.09
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.34
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.4
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.1
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.9
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.55

8 rows × 31 columns

In [11]:
```
data = data.sample(frac = 0.2,random_state = 42)
data.shape
```

Out[11]:
```
(56961, 31)
```

In [17]:
```
fraud = data[data['Class'] == 1]
valid = data[data['Class'] == 0]
```

In [18]:
```
print("Fraud: ",len(fraud))
print("Valid: ",len(valid))
```

```
Fraud:  98
Valid:  56863
```

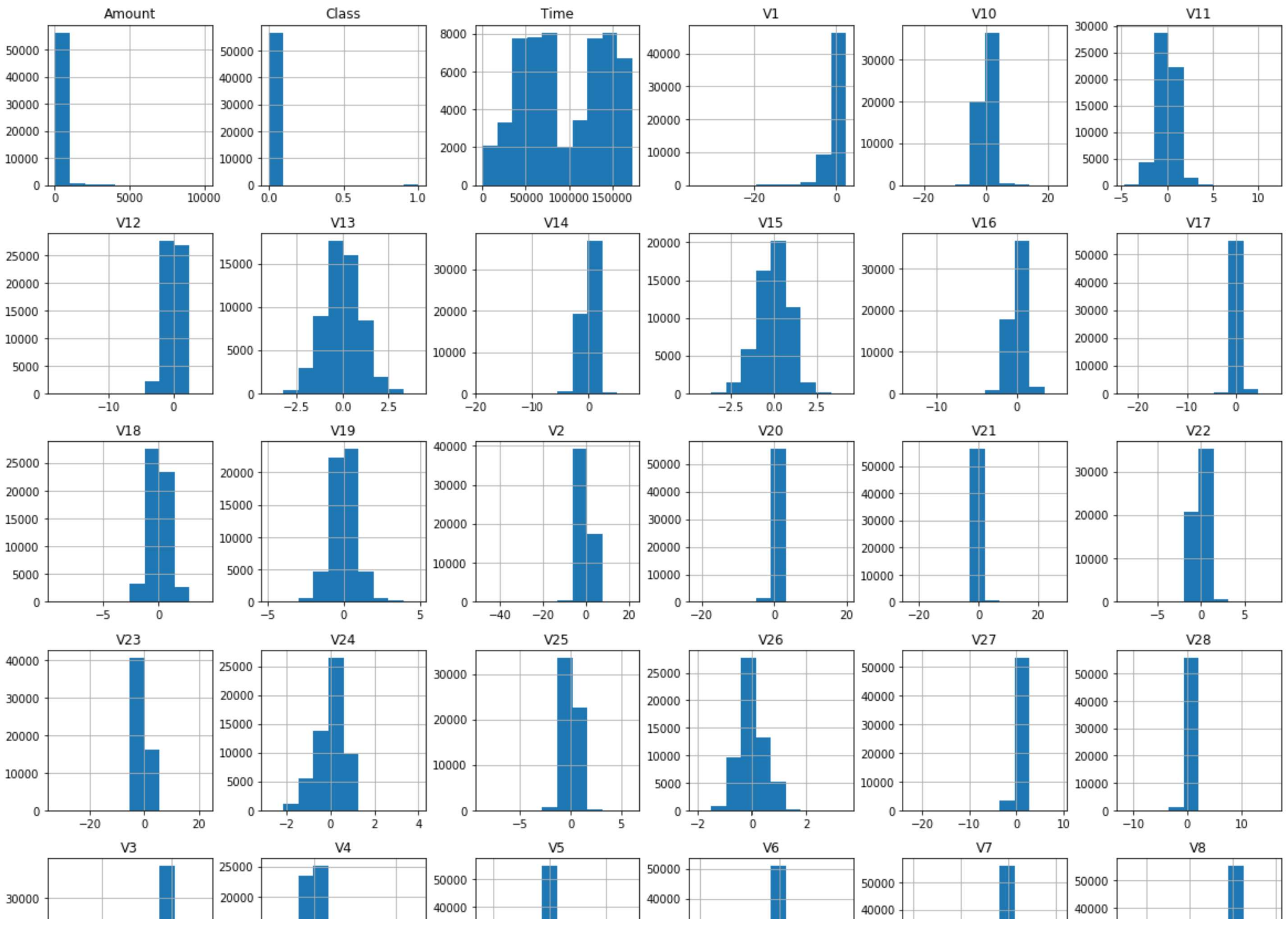
In [20]:
```
# check outliers occurance
```

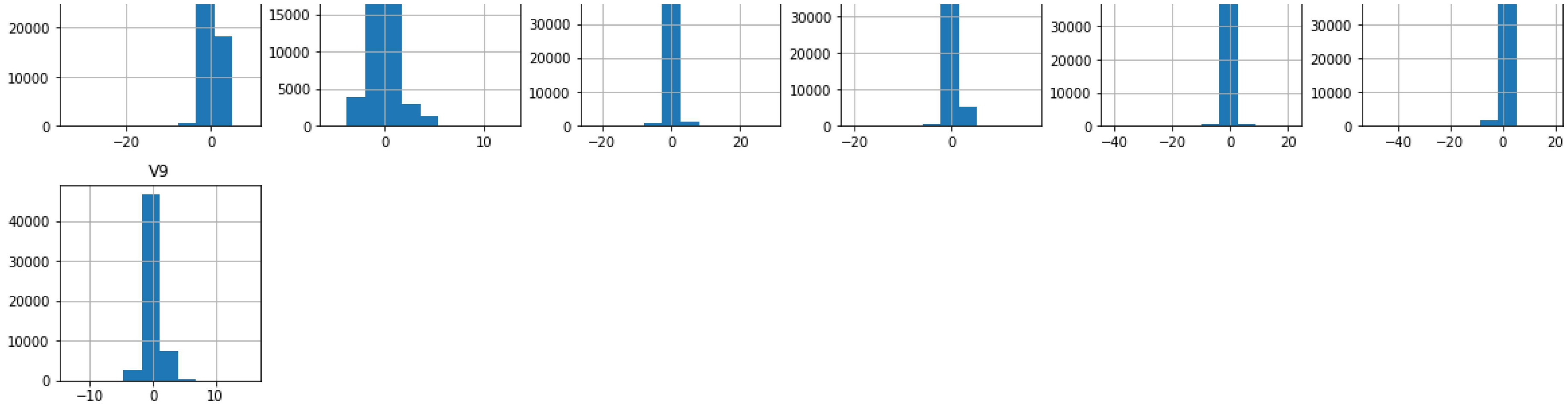
In [22]:
```
outlier = len(fraud)/len(valid)
outlier
```

Out[22]:
```
0.0017234405500940859
```

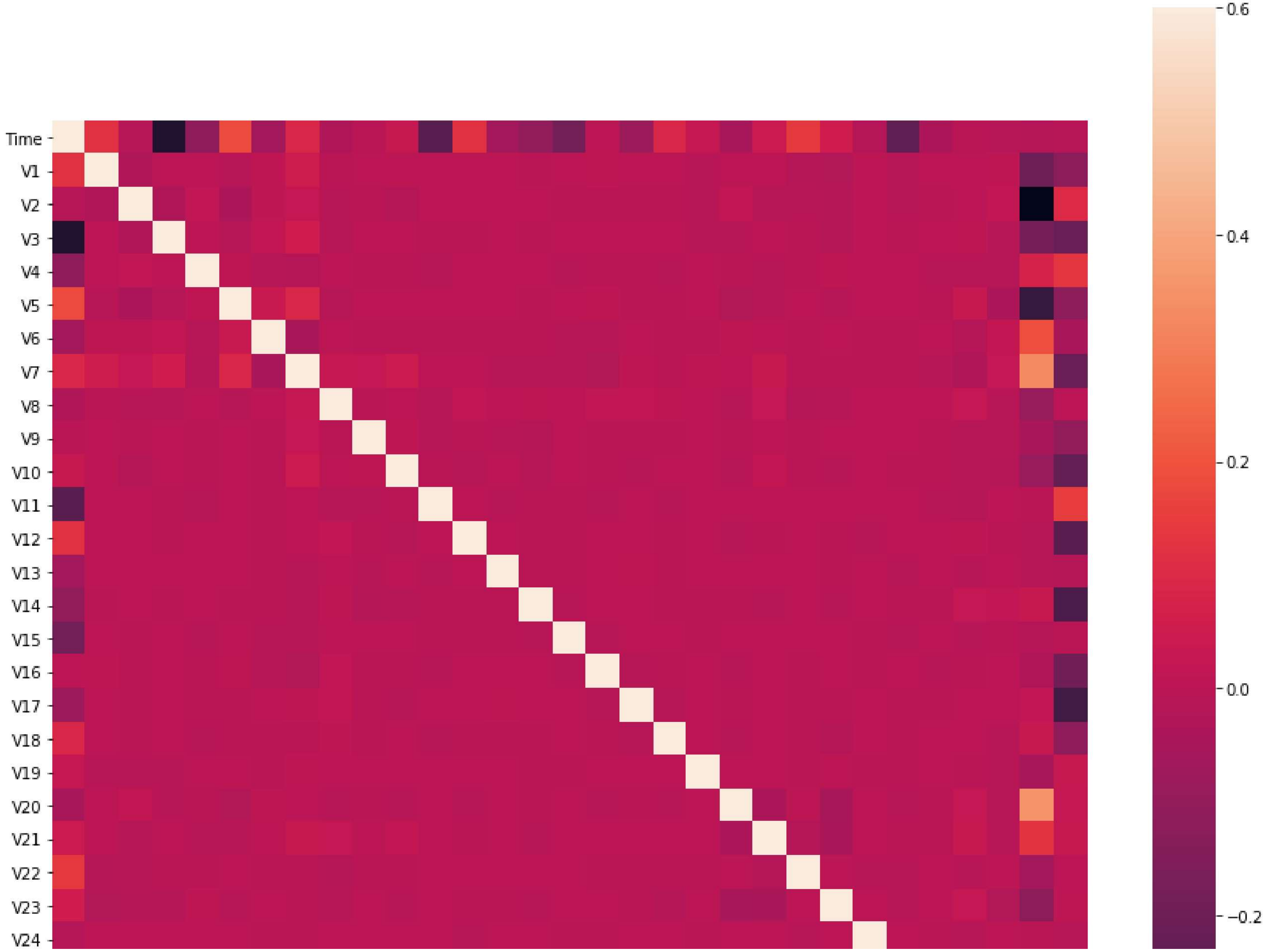
In [23]:
```
# Visualise Data
```

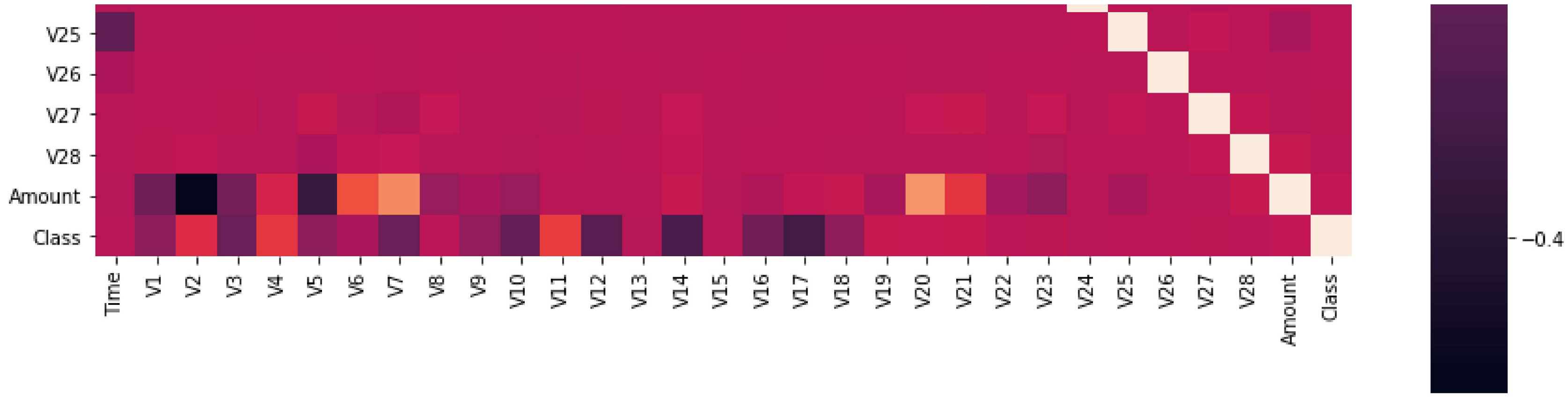
In [27]:
```
data.hist(figsize = (20,20))
plt.show()
```




V9

In [30]:
```
corr = data.corr()
figure = plt.figure(figsize = (15,15))
sns.heatmap(corr, vmax= .6, square = True)
plt.show()
```





In [32]:
```
len(corr['Class'])
```

Out[32]: 31

In [33]:
```
# Select Model
```

In [37]:
```
column = corr.keys()
col_store = []
for i in range(len(corr)):
    if abs(corr['Class'][i] > 0.01):
           col_store.append(column[i])
```

In [38]:
```
print(col_store)
```

```
['V2', 'V4', 'V11', 'V19', 'V20', 'V21', 'V23', 'Amount', 'Class']
```

In [39]:
```
len(col_store)
```

Out[39]: 9

In [41]:
```
feature = data.drop("Class",axis=1) #remove class
target = data["Class"]
```

In [42]:
```
print("Train Set: ", feature.shape)
print("Target Set: ", target.shape)
```

```
Train Set:  (56961, 30)
Target Set:  (56961,)
```

In [43]:
```
# Lets train the model LOF, Isolation Forest
```

In [46]:
```
# define outlier detection tools to be compared
classifiers = {
    "IF": IsolationForest(max_samples = len(feature),
                                        contamination = outlier,
                                        random_state = 1),
    "LOF": LocalOutlierFactor(
        n_neighbors = 20,
        contamination = outlier)}
```

In [51]:
```python
n_of_outliers = len(fraud)

# Fit the model
for i, (clf_name, clf) in enumerate(classifiers.items()):

    # fit the dataframe and tag outliers
    if clf_name == "LOF":

        y_pred = clf.fit_predict(feature)
        scores_pred = clf.negative_outlier_factor_

    else:

        # train/fit classifier on our features
        clf.fit(feature)
        # generate predictions
        scores_pred = clf.decision_function(feature)
        y_pred = clf.predict(feature)

    # 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 != target).sum()

    # Run classification metrics
    print('Accuracy: {0}%\n'.format(accuracy_score(target, y_pred)*100))
```

```
Accuracy: 99.76650690823547%

Accuracy: 99.65766050455575%
```

In [ ]:
```python

```