In [ ]:

```
# Credit/Debit Card Fraud Detection
```

In [3]:

```
import pandas as pd
from pandas.plotting import scatter_matrix
import numpy as np
import matplotlib.pyplot as plt
import os
from imblearn.over_sampling import ADASYN
from collections import Counter
import seaborn as sn
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
%matplotlib inline

flatui = ["#9b59b6", "#3498db", "#95a5a6", "#e74c3c", "#34495e", "#2ecc71"]
sn.set_palette(flatui)
```

```
Using TensorFlow backend.
```

In [4]:

```

import warnings
warnings.filterwarnings('ignore')
```

In [5]:

```

def plot_confusion_matrix(cm, classes, title, cmap):
    "plotting confusion matrix"
    plt.clf()
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    classnames = classes
    plt.title(title)
    plt.ylabel('True')
    plt.xlabel('Predicted')
    tick_marks = np.arange(len(classnames))
    plt.xticks(tick_marks, classnames, rotation=0)
    plt.yticks(tick_marks, classnames)
    s = [['TN','FP'], ['FN', 'TP']]

    for i in range(2):
        for j in range(2):
            plt.text(j,i, str(s[i][j])+" = "+str(cm[i][j]))
    plt.show()
```

In [6]:

```
def plot_roc(arg1, arg2, arg3):
    "a function to plot roc_auc"
    fig, ax = plt.subplots(figsize=(8, 6))
    for i, v in arg1:
        y_score = v.predict_proba(arg2)[:, 1]
        fpr, tpr, _ = metrics.roc_curve(arg3, y_score)
        roc_auc = metrics.auc(fpr, tpr)
        plt.plot(fpr, tpr,lw=2, label= i + ' (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic Curve')
    plt.legend(loc="lower right")
    plt.show()
```

In [7]:

```
df = pd.read_csv(r"C:\Users\HP\Desktop\creditcard.csv")
print('The dataset contains {0} rows and {1} columns.'.format(df.shape[0], df.shape[1]))
print('Normal transactions count: ', df['Class'].value_counts().values[0])
print('Fraudulent transactions count: ', df['Class'].value_counts().values[1])
```

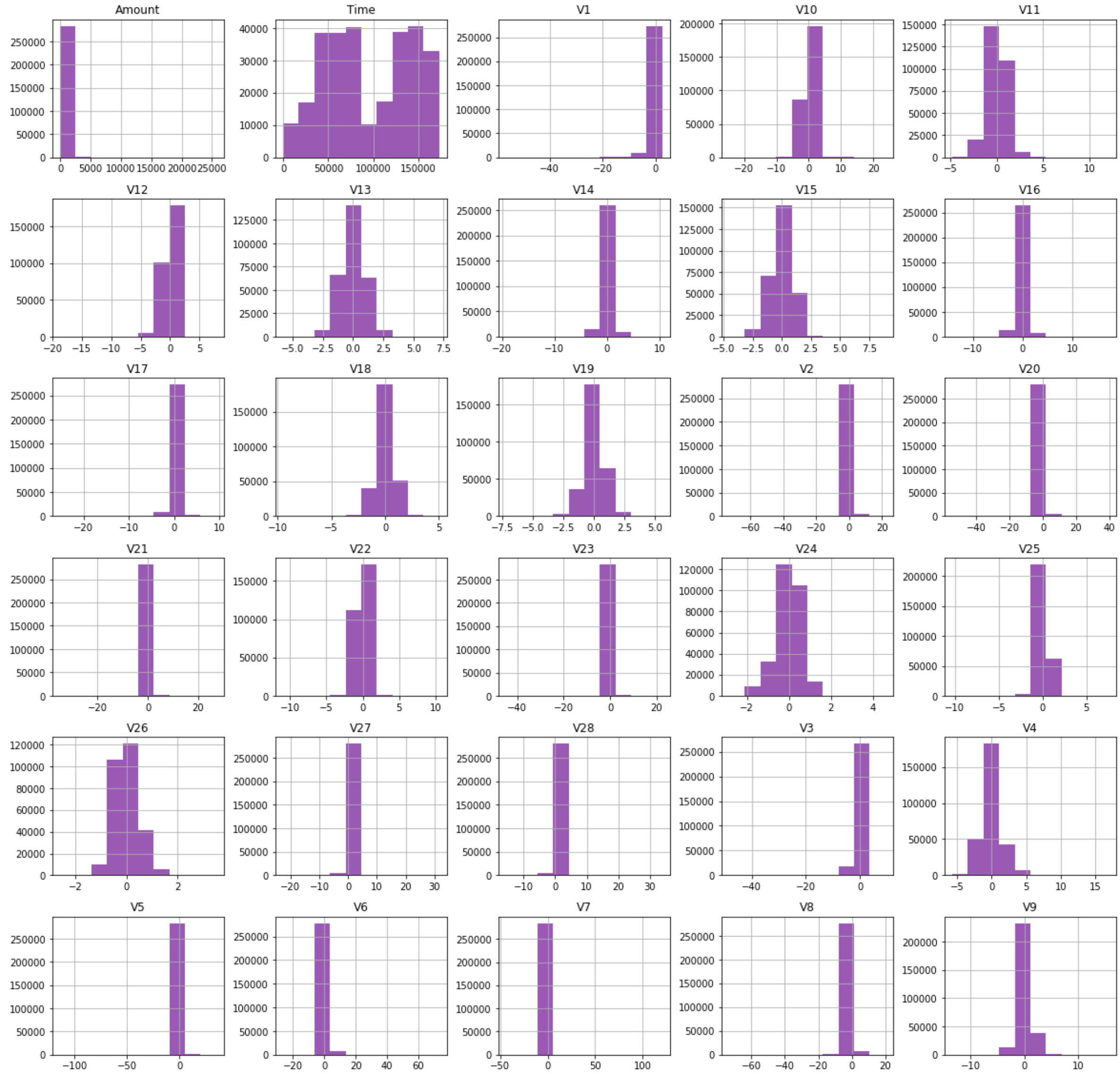
```
The dataset contains 284807 rows and 31 columns.
Normal transactions count:  284315
Fraudulent transactions count:  492
```

In [8]:

```

X = df.iloc[:, :-1]
y = df['Class']
scaler = StandardScaler()
scaled_X = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(scaled_X, y, test_size=0.30, random_sta
```

In [9]:

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



In [10]:

```
LGR = LogisticRegression()
LGR.fit(X_train, y_train);
```

In [11]:

```

RDF = RandomForestClassifier(random_state=0)
RDF.fit(X_train, y_train);
```

In [12]:

```
modl = [('Logistic Regression', LGR), ('Random Forest Method', RDF)]
models = [md for md in modl]
```

In [13]:

```
print()

for a,b in models:
    scores = cross_val_score(b, X_train, y_train, cv=10)
    accuracy = metrics.accuracy_score(y_train, b.predict(X_train))
    confusion_matrix = metrics.confusion_matrix(y_train, b.predict(X_train))
    classification = metrics.classification_report(y_train, b.predict(X_train))
    print('************************* {} *************************'.format(a))
    print()
    print ("Mean Score: ", '{}%'.format(np.round(scores.mean(), 3) * 100))
    print()
    print ("Model Accuracy: ", '{}%'.format(np.round(accuracy, 3) * 100))
    print()
    print("Confusion Matrix:" "\n", confusion_matrix)
    print()
    print("Classification Report:" "\n", classification)
    print()
```

```

************************* Logistic Regression *************************

Mean Score:  99.9%

Model Accuracy:  99.9%

Confusion Matrix:
 [[198981     27]
 [   136    220]]

Classification Report:
               precision    recall  f1-score   support

         '0'       1.00      1.00      1.00    199008
         '1'       0.89      0.62      0.73       356

    accuracy                           1.00    199364
   macro avg       0.95      0.81      0.86    199364
weighted avg       1.00      1.00      1.00    199364


************************* Random Forest Method *************************

Mean Score:  99.9%

Model Accuracy:  100.0%

Confusion Matrix:
 [[199007      1]
 [    14    342]]

Classification Report:
               precision    recall  f1-score   support

         '0'       1.00      1.00      1.00    199008
         '1'       1.00      0.96      0.98       356

    accuracy                           1.00    199364
   macro avg       1.00      0.98      0.99    199364
weighted avg       1.00      1.00      1.00    199364
```

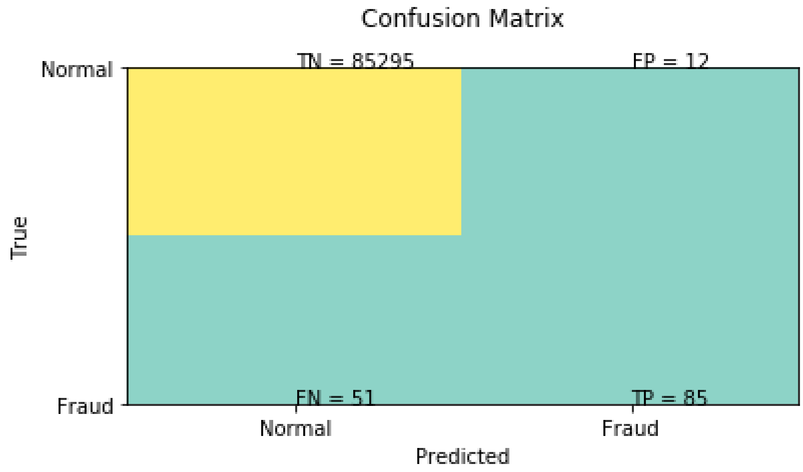
In [14]:

```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
```

In [15]:

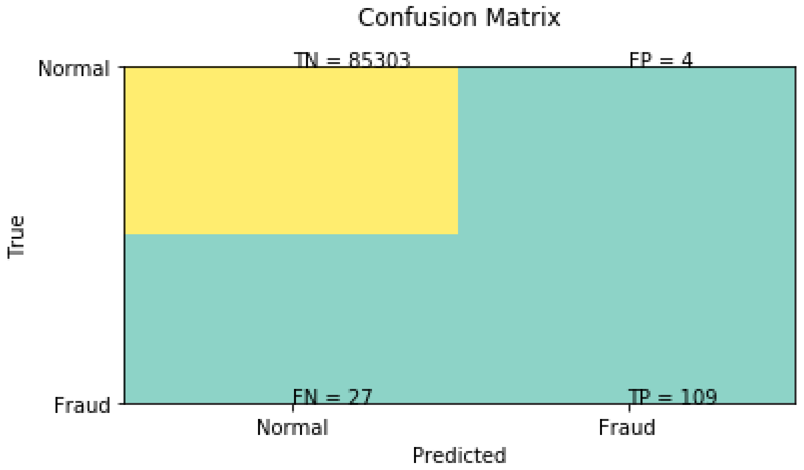
```
classnf = {'Normal':0, 'Fraud':1}
print()
print('************************** Model Test Results **************************' "\n")

for a, b in models:
    accuracy = metrics.accuracy_score(y_test, b.predict(X_test))
    confusion_matrix = metrics.confusion_matrix(y_test, b.predict(X_test))
    classification = metrics.classification_report(y_test, b.predict(X_test))
    print('********** {} **********'.format(a))
    print ("Model Accuracy: ",  '{}%'.format(np.round(accuracy, 3) * 100))
    print()
    print("Confusion Matrix:" "\n", confusion_matrix)
    print()
    print("Matrix Plot : ")
    plot_confusion_matrix(confusion_matrix, classes = list(classnf.keys()), title='Confusio
    print()
    print("Classification Report:" "\n", classification)
    print()
```

```
************************** Model Test Results **************************

********** Logistic Regression **********
Model Accuracy:  99.9%

Confusion Matrix:
 [[85295    12]
 [   51    85]]

Matrix Plot : 
```



```
Classification Report:
               precision    recall  f1-score   support

         '0'       1.00      1.00      1.00     85307
         '1'       0.88      0.62      0.73       136

    accuracy                           1.00     85443
   macro avg       0.94      0.81      0.86     85443
weighted avg       1.00      1.00      1.00     85443
```

```
********** Random Forest Method **********
Model Accuracy:  100.0%

Confusion Matrix:
 [[85303     4]
 [   27   109]]

Matrix Plot :
```



```
Classification Report:
               precision    recall  f1-score   support

         '0'       1.00      1.00      1.00     85307
         '1'       0.96      0.80      0.88       136

    accuracy                           1.00     85443
   macro avg       0.98      0.90      0.94     85443
weighted avg       1.00      1.00      1.00     85443
```

```
In [ ]:
```