In [7]:
```python
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as pt
import seaborn as sns
```

In [2]:
```python
data = pd.read_csv('D:/ML Verzeo/creditcard.csv')
```

In [12]:
```python
data
```

Out[12]:

	Time	V1	V2	V3	V4	V5	V6	V7	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-
...	...	...	...	...	...	...	...	...	
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-

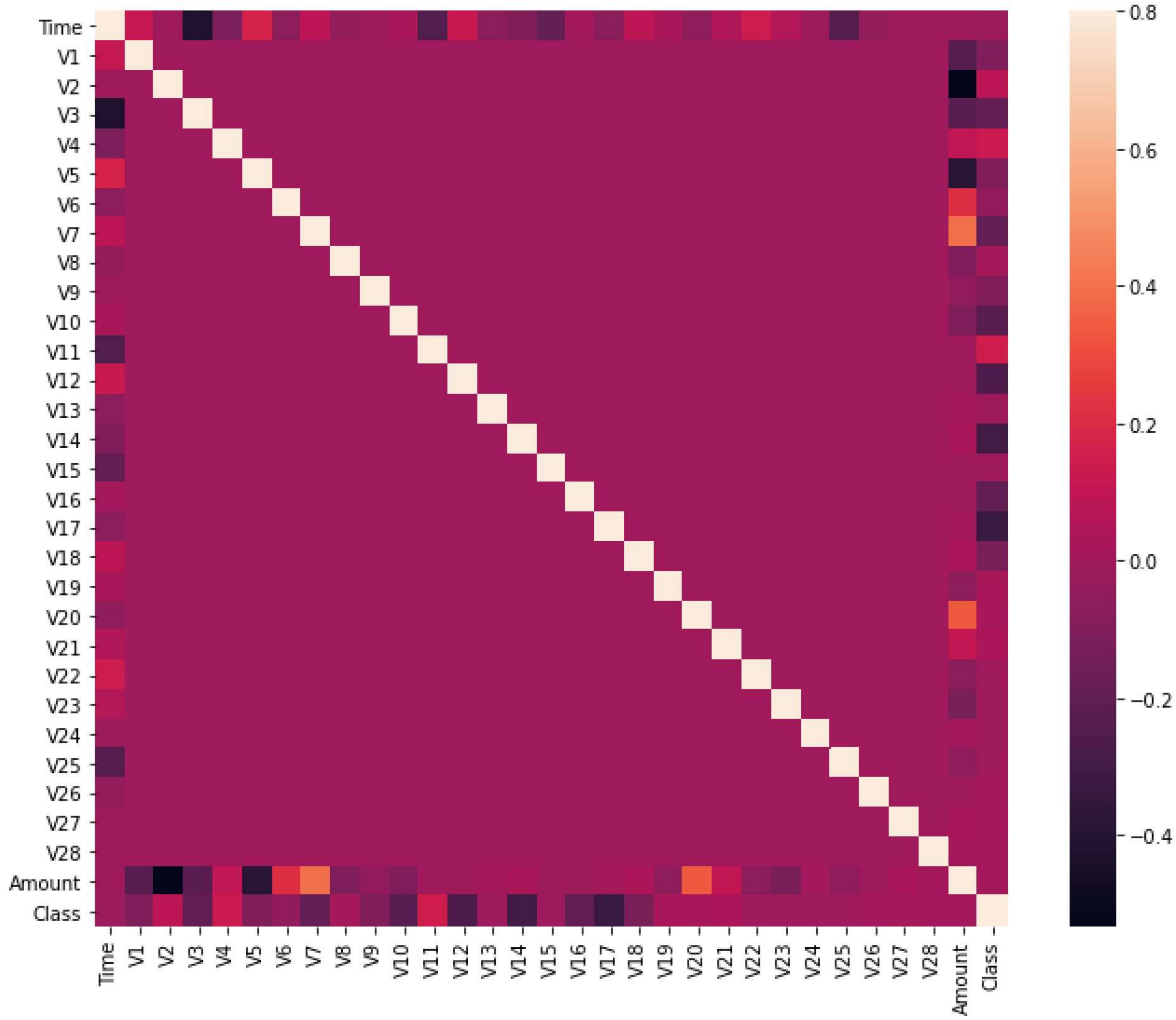
In [6]:
```python
data.corr()
```

Out[6]:

	Time	V1	V2	V3	V4	V5	
Time	1.000000	1.173963e-01	-1.059333e-02	-4.196182e-01	-1.052602e-01	1.730721e-01	-6
V1	0.117396	1.000000e+00	4.697350e-17	-1.424390e-15	1.755316e-17	6.391162e-17	2.3
V2	-0.010593	4.697350e-17	1.000000e+00	2.512175e-16	-1.126388e-16	-2.039868e-16	5.0
V3	-0.419618	-1.424390e-15	2.512175e-16	1.000000e+00	-3.416910e-16	-1.436514e-15	1.4
V4	-0.105260	1.755316e-17	-1.126388e-16	-3.416910e-16	1.000000e+00	-1.940929e-15	-2
V5	0.173072	6.391162e-17	-2.039868e-16	-1.436514e-15	-1.940929e-15	1.000000e+00	7.9
V6	-0.063016	2.398071e-16	5.024680e-16	1.431581e-15	-2.712659e-16	7.926364e-16	1.00

In [10]:
```
corrmat = data.corr()
fig = pt.figure(figsize = (12, 9))

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



Fraud Data Analysis

In [5]:
```
fraud = data[data['Class'] == 1]
valid = data[data['Class'] == 0]
outlier_fraction = len(fraud)/float(len(valid))
print(outlier_fraction)
print(fraud.shape,valid.shape)
```

```
0.0017304750013189597
(492, 31) (284315, 31)
```

In [13]:
```
print("Fraudulent transaction")
fraud.Amount.describe()
```

```
Fraudulent transaction
```

Out[13]:
```
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

Fraudulent Cases are 492

In [15]:
```
print("Valid transaction")
valid.Amount.describe()
```

```
Valid transaction
```

Out[15]:
```
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

Non-Fraudulent Transactions are 2,84,315

In [ ]:

In [ ]:

Data Analysis

In [16]:
```
X = data.iloc[:, :-1]
y = data['Class']
```

In [14]:
```
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
xTrain, xTest, yTrain, yTest = train_test_split(
        X, y, test_size=0.33, random_state=42)
```

1) Random Forest Algorithm

In [15]:
```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()
rfc.fit(xTrain, yTrain)

yPred = rfc.predict(xTest)
from sklearn.metrics import accuracy_score
acc = accuracy_score(yTest, yPred)
print("The accuracy of Random forest is {}".format(acc))
```

```
The accuracy of Random forest is 0.999627608073457
```

Random Forest Algorithm Accuracy = 0.99962

In [ ]:

2) Logistic Regression

In [16]:
```
from sklearn.linear_model.logistic import LogisticRegression
clf=LogisticRegression()
clf.fit(xTrain, yTrain)
yPred=clf.predict(xTest)
acc = accuracy_score(yTest, yPred)
print("The accuracy of Logistic Regression is {}".format(acc))
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/utils/deprecation.py:144: Future
Warning: The sklearn.linear_model.logistic module is  deprecated in version 0.2
2 and will be removed in version 0.24. The corresponding classes / functions sh
ould instead be imported from sklearn.linear_model. Anything that cannot be imp
orted from sklearn.linear_model is now part of the private API.
  warnings.warn(message, FutureWarning)

The accuracy of Logistic Regression is 0.9989041037590305

/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py:940: C
onvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Logistic Regression Accuracy is 0.99890

In [ ]:

3) Decision Tree

In [17]:

```
from sklearn.tree import DecisionTreeClassifier
dtc_clf=DecisionTreeClassifier()
dtc_clf.fit(xTrain, yTrain)
yPred=dtc_clf.predict(xTest)
acc = accuracy_score(yTest, yPred)
print("The accuracy of Decision Tree is {}".format(acc))
```

```
The accuracy of Decision Tree is 0.9991700979922755
```

Decision Tree Accuracy is 0.99917

In [ ]: