In [6]:
```
import numpy as np
import pandas as pd
import sklearn
import scipy
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report,accuracy_score
from sklearn.ensemble import IsolationForest  # IFA algorith (IsolationForest)
from sklearn.neighbors import LocalOutlierFactor #LOF algorith (LocalOutlierFactor
from sklearn.svm import OneClassSVM #  support-vector machines (SVM)
# SVM:-are supervised learning models with associated learning algorithms that ana
# analysis.
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.svm import SVR
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
# from sklearn.model_selection import train_test_split (this line has to import fo
import math
from pylab import rcParams
rcParams['figure.figsize'] = 14, 8
RANDOM_SEED = 42
LABELS = ["Normal", "Fraud"]
from sklearn.preprocessing import StandardScaler
```

In [7]:
```
dataset = pd.read_csv(r"C:\Users\Deepak kumar sharma\OneDrive\Desktop\Jupyter Note
# above line dataset = pd.read_csv(r"path of the .csv file")  then it will run prop
```

In [8]:
```
dataset.head() # 1st 5 row of dataset
```

Out[8]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0..
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0..
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0..

5 rows × 31 columns

In [9]:
```
dataset.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  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 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(30), int64(1)
memory usage: 67.4 MB
```

In [10]:

```
x = dataset.iloc[: , 1:30].values
y = dataset.iloc[:, 30].values
```

In [11]:

```
print("Input Range : ", x.shape)
print("Output Range : ", y.shape)
```

```
Input Range :  (284807, 29)
Output Range :  (284807,)
```

In [12]:

```
print ("Class Labels : \n", y)
```

```
Class Labels : 
 [0 0 0 ... 0 0 0]
```

In [13]:

```
dataset.isnull().values.any() # Good No Null Values!
```

Out[13]:

```
False
```

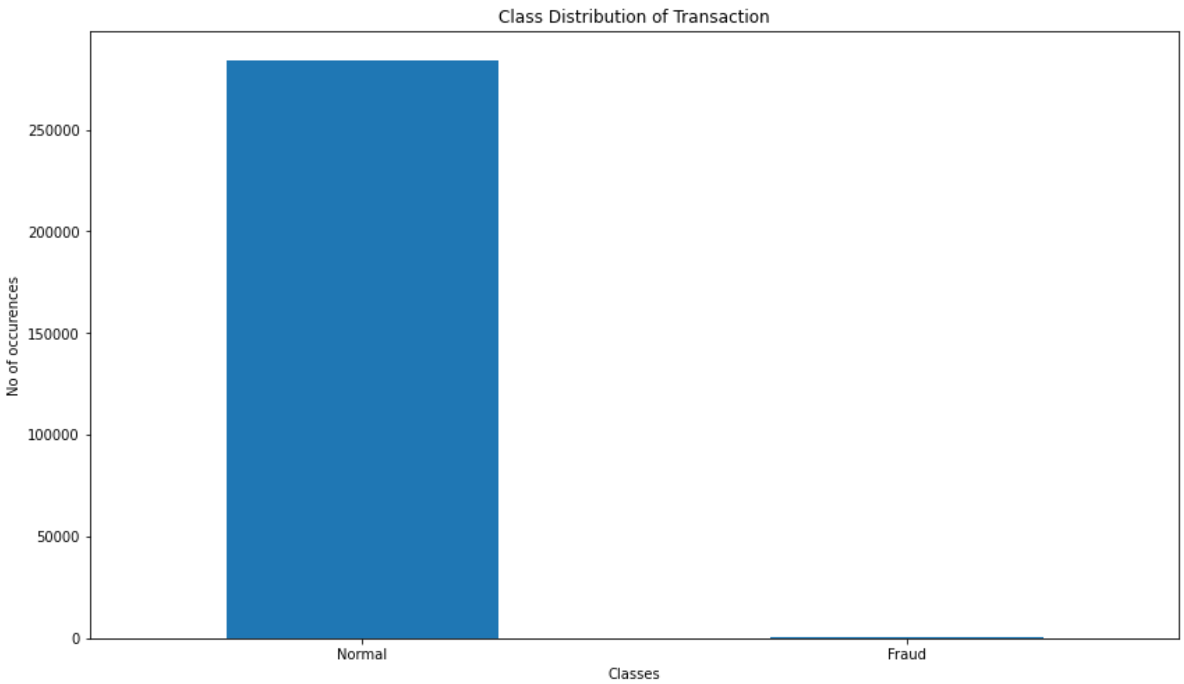
In [14]:

```
set_class = pd.value_counts(dataset['Class'], sort = True) #  pd.value_counts() fu
# coloumn which is used //here outputs how many fraud and non fraud occur , sort()

set_class.plot(kind = 'bar', rot=0) #1. Simple Bar Plot function
 # link:-https://dataindependent.com/pandas/pandas-bar-plot-dataframe-plot-bar/

plt.title("Class Distribution of Transaction")
```

```
plt.xticks(range(2), LABELS) # plt.xticks(ticks=none ,labels=none, **kways)///plt.x

plt.xlabel("Classes")

plt.ylabel("No of occurences")

#1. Simple Bar Plot
#In order to create a bar plot, you need to pass a X and Y values. X will be your c
#bars. Y will be the value of your bars, or how high they are.
#Note: Rot = Rotation. When I specify rot=0, I'm telling pandas not to rotate my x
```

Out[14]:

```
Text(0, 0.5, 'No of occurences')
```



In [15]:

```
fraud_data = dataset[dataset['Class']==1]

normal_data = dataset[dataset['Class']==0]
```

In [ ]:

```
# print(fraud_data.shape,normal_data.shape)
```

In [16]:

```
fraud_data.Amount.describe()
```

Out[16]:

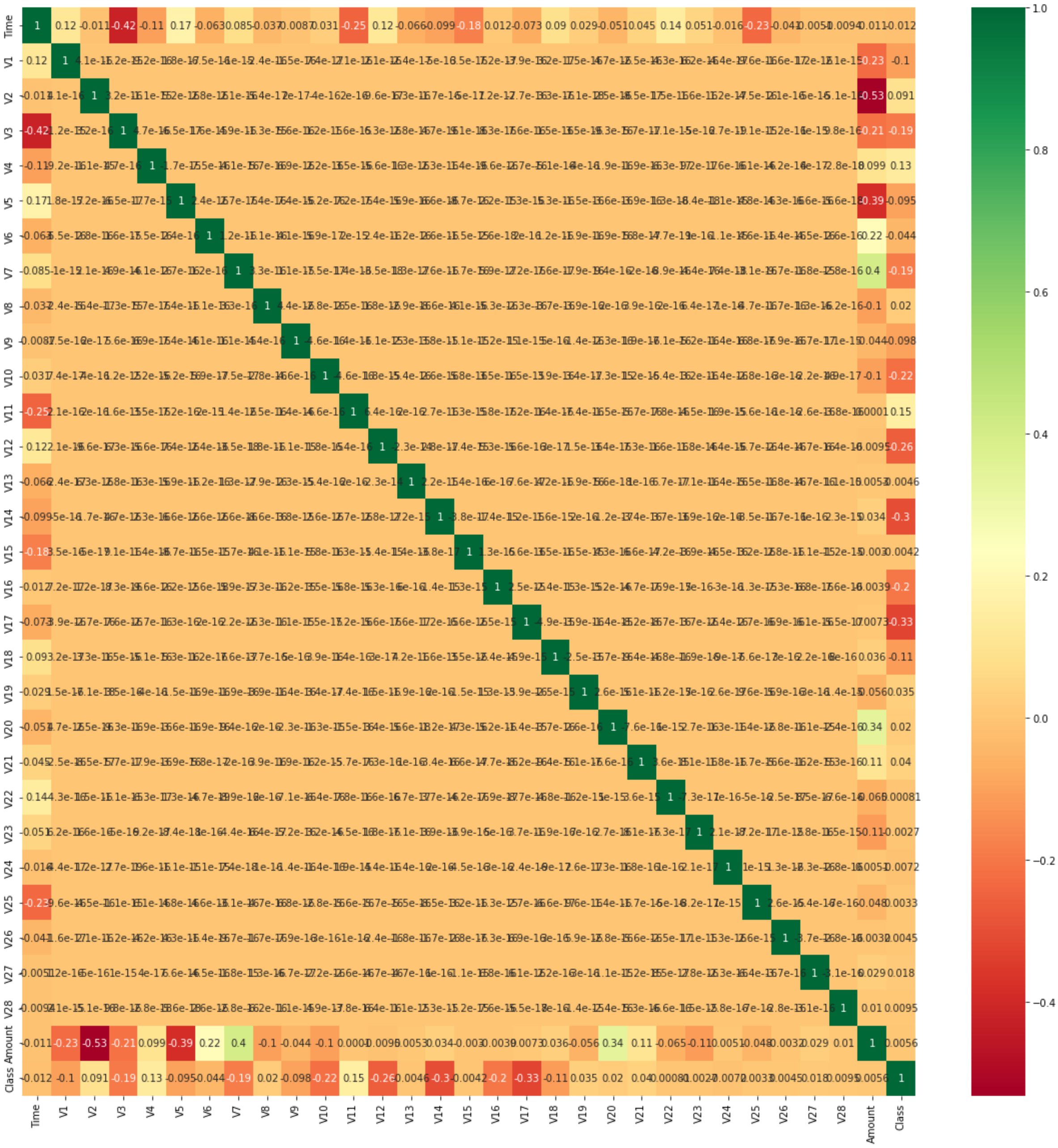
```
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

In [17]:

```
normal_data.Amount.describe()
```

Out[17]:
```
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
```

In [18]:
```
## Correlation
import seaborn as sns
#get correlations of each features in dataset
corrmat = dataset.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(dataset[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



In [27]:
```
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.25, random_state
# this line give error because we have to 1st import  (from sklearn.model_selectio
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [27], in <cell line: 1>()
----> 1 xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.25, rand
om_state = 0)

NameError: name 'train_test_split' is not defined
```

In [28]:
```
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.25, random_state
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [28], in <cell line: 1>()
----> 1 xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.25, rand
om_state = 0)

NameError: name 'train_test_split' is not defined
```

In [19]:
```
from sklearn.model_selection import train_test_split
```

In [20]:
```
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.25, random_state
```

In [21]:
```
print("xtrain.shape : ", xtrain.shape)
print("xtest.shape  : ", xtest.shape)
print("ytrain.shape : ", ytrain.shape)
print("ytest.shape  : ", ytest.shape)
```

```
xtrain.shape :  (213605, 29)
xtest.shape  :  (71202, 29)
ytrain.shape :  (213605,)
ytest.shape  :  (71202,)
```

In [22]:
```
#StandardScaler follows Standard Normal Distribution (SND). Therefore, it makes mea
stdsc = StandardScaler()
xtrain = stdsc.fit_transform(xtrain)
xtest = stdsc.transform(xtest)
#The fit(data) method is used to compute the mean and std dev for a given feature s
#The transform(data) method is used to perform scaling using mean and std dev calcu
#The fit_transform() method does both fit and transform.
```

In [23]:
```
print("Training Set after Standardised : \n", xtrain[0])
```

```
Training Set after Standardised :
 [ 1.04272047  0.06657394 -1.19051456  0.05060912  0.18235446 -1.31399333
  0.58133086 -0.40257892 -0.09319222  0.16481198  1.60036637  1.18028602
 -0.24273404  1.08764203 -0.35935009 -0.76863613 -0.28881862 -0.39536117
  0.13774039 -0.34055771  0.32484688  1.13026957  0.03716189  0.90724443
  0.61754959  0.39904973 -0.21031503 -0.2607924  -0.35356699]
```

In [35]:
```
dt_classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
dt_classifier.fit(xtrain, ytrain)
```

Out[35]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In [36]:
```
y_pred_decision_tree = dt_classifier.predict(xtest)
```

In [37]:
```
print("y_pred_decision_tree : \n", y_pred_decision_tree)
```

```
y_pred_decision_tree :
 [0 0 0 ... 0 0 0]
```

In [38]:
```
com_decision = confusion_matrix(ytest, y_pred_decision_tree)
print("confusion Matrix : \n", com_decision)
```

```
confusion Matrix :
 [[71052    30]
 [   25    95]]
```

In [39]:
```
Accuracy_Model = ((com_decision[0][0] + com_decision[1][1]) / com_decision.sum()) 
print("Accuracy_Decison     : ", Accuracy_Model)

Error_rate_Model= ((com_decision[0][1] + com_decision[1][0]) / com_decision.sum())
print("Error_rate_Decison   : ", Error_rate_Model)

# True Fake Rate
Specificity_Model= (com_decision[1][1] / (com_decision[1][1] + com_decision[0][1]))
print("Specificity_Decison : ", Specificity_Model)

# True Genuine Rate
Sensitivity_Model = (com_decision[0][0] / (com_decision[0][0] + com_decision[1][0]
print("Sensitivity_Decison : ", Sensitivity_Model)
```

```
Accuracy_Decison    :  99.92275497879272
Error_rate_Decison  :  0.07724502120726946
Specificity_Decison :  76.0
Sensitivity_Decison :  99.96482687789299
```

In [24]:
```
svc_classifier = SVC(kernel = 'rbf', random_state =0)
svc_classifier.fit(xtrain, ytrain)
```

Out[24]:
```
SVC
SVC(random_state=0)
```

In [25]:
```
y_pred2 = svc_classifier.predict(xtest)
```

In [26]:
```
print("y_pred_randomforest : \n", y_pred2)
```

```
y_pred_randomforest :
 [0 0 0 ... 0 0 0]
```

In [27]:
```
cm2 = confusion_matrix(ytest, y_pred2)
print("Confusion Matrix : \n\n", cm2)
```

```
Confusion Matrix :

 [[71077     5]
 [   44    76]]
```

In [28]:
```
# Validating the Prediction
Accuracy_Model = ((cm2[0][0] + cm2[1][1]) / cm2.sum()) *100
print("Accuracy_svc     : ", Accuracy_Model)

Error_rate_Model = ((cm2[0][1] + cm2[1][0]) / cm2.sum()) *100
print("Error_rate_svc   : ", Error_rate_Model)

# True Fake Rate
Specificity_Model= (cm2[1][1] / (cm2[1][1] + cm2[0][1])) *100
print("Specificity_svc : ", Specificity_Model)

# True Genuine Rate
Sensitivity_Model= (cm2[0][0] / (cm2[0][0] + cm2[1][0])) *100
print("Sensitivity_svc : ", Sensitivity_Model)
```

```
Accuracy_svc     :  99.93118170837899
Error_rate_svc   :  0.06881829162102188
Specificity_svc :  93.82716049382715
Sensitivity_svc :  99.93813360329578
```

In [32]:

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [32], in <cell line: 2>()
      1 # training the logistic model on training data data
----> 2 model=LogisticRegression()

NameError: name 'LogisticRegression' is not defined
```

In [34]:

```python
from sklearn.linear_model import LogisticRegression
```

In [35]:

```python
# training the logistic model on training data data
model=LogisticRegression()
```

In [36]:

```python
model.fit(xtrain, ytrain)
```

Out[36]:

```
▾ LogisticRegression
LogisticRegression()
```

In [37]:

```python
# accuracy score  on training data
x_train_prediction=model.predict(xtrain)
training_data_accuracy=accuracy_score(x_train_prediction,ytrain)
```

In [38]:

```python
print("accurcy on training data: ",training_data_accuracy)
```

```
accurcy on training data:  0.99917136771143
```

In [40]:

```python
# accuracy score  on test data
x_test_prediction=model.predict(xtest)
test_data_accuracy=accuracy_score(x_test_prediction,ytest)
```

In [41]:

```python
print("accurcy on test data: ",test_data_accuracy)
```

```
accurcy on test data:  0.9992977725344794
```

In [ ]: