```
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
from sklearn.neighbors import LocalOutlierFactor
from sklearn.svm import OneClassSVM
from pylab import rcParams
rcParams['figure.figsize'] = 14, 8
RANDOM_SEED = 42
LABELS = ["Normal", "Fraud"]
```

```
from google.colab import drive
drive.mount ('/content/drive')
```

```
    Mounted at /content/drive
```

```
import pandas as pd
path = "/content/drive/MyDrive/creditcard.csv"
df = pd.read_csv(path)

df.describe
```

```
    <bound method NDFrame.describe of             Time         V1         V2        V3        V4        V5  \
    0            0.0  -1.359807  -0.072781  2.536347  1.378155 -0.338321
    1            0.0   1.191857   0.266151  0.166480  0.448154  0.060018
    2            1.0  -1.358354  -1.340163  1.773209  0.379780 -0.503198
    3            1.0  -0.966272  -0.185226  1.792993 -0.863291 -0.010309
    4            2.0  -1.158233   0.877737  1.548718  0.403034 -0.407193
    ...          ...        ...        ...       ...       ...       ...
    284802  172786.0 -11.881118  10.071785 -9.834783 -2.066656 -5.364473
    284803  172787.0  -0.732789  -0.055080  2.035030 -0.738589  0.868229
    284804  172788.0   1.919565  -0.301254 -3.249640 -0.557828  2.630515
                                 .530483  0.702510  0.689799 -0.377961
                                 .189733  0.703337 -0.506271 -0.012546
```

Saved successfully!

```
                  V6        V7        V8        V9  ...       V21       V22  \
    0       0.462388  0.239599  0.098698  0.363787  ... -0.018307  0.277838
    1      -0.082361 -0.078803  0.085102 -0.255425  ... -0.225775 -0.638672
    2       1.800499  0.791461  0.247676 -1.514654  ...  0.247998  0.771679
    3       1.247203  0.237609  0.377436 -1.387024  ... -0.108300  0.005274
    4       0.095921  0.592941 -0.270533  0.817739  ... -0.009431  0.798278
    ...          ...       ...       ...       ...  ...       ...       ...
    284802 -2.606837 -4.918215  7.305334  1.914428  ...  0.213454  0.111864
    284803  1.058415  0.024330  0.294869  0.584800  ...  0.214205  0.924384
    284804  3.031260 -0.296827  0.708417  0.432454  ...  0.232045  0.578229
    284805  0.623708 -0.686180  0.679145  0.392087  ...  0.265245  0.800049
    284806 -0.649617  1.577006 -0.414650  0.486180  ...  0.261057  0.643078

                 V23       V24       V25       V26       V27       V28  Amount  \
    0      -0.110474  0.066928  0.128539 -0.189115  0.133558 -0.021053  149.62
    1       0.101288 -0.339846  0.167170  0.125895 -0.008983  0.014724    2.69
    2       0.909412 -0.689281 -0.327642 -0.139097 -0.055353 -0.059752  378.66
    3      -0.190321 -1.175575  0.647376 -0.221929  0.062723  0.061458  123.50
    4      -0.137458  0.141267 -0.206010  0.502292  0.219422  0.215153   69.99
    ...          ...       ...       ...       ...       ...       ...     ...
    284802  1.014480 -0.509348  1.436807  0.250034  0.943651  0.823731    0.77
    284803  0.012463 -1.016226 -0.606624 -0.395255  0.068472 -0.053527   24.79
    284804 -0.037501  0.640134  0.265745 -0.087371  0.004455 -0.026561   67.88
    284805 -0.163298  0.123205 -0.569159  0.546668  0.108821  0.104533   10.00
    284806  0.376777  0.008797 -0.473649 -0.818267 -0.002415  0.013649  217.00

            Class
    0           0
    1           0
    2           0
    3           0
    4           0
    ...       ...
    284802      0
    284803      0
    284804      0
    284805      0
    284806      0

    [284807 rows x 31 columns]>
```

```
df.describe()
```

	Time	V1	V2	V3	V4	V5	V6	V7	
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+
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	1.487313e-15	-5.556467e-16	1.213481e-
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+
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+
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-
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-
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-
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+

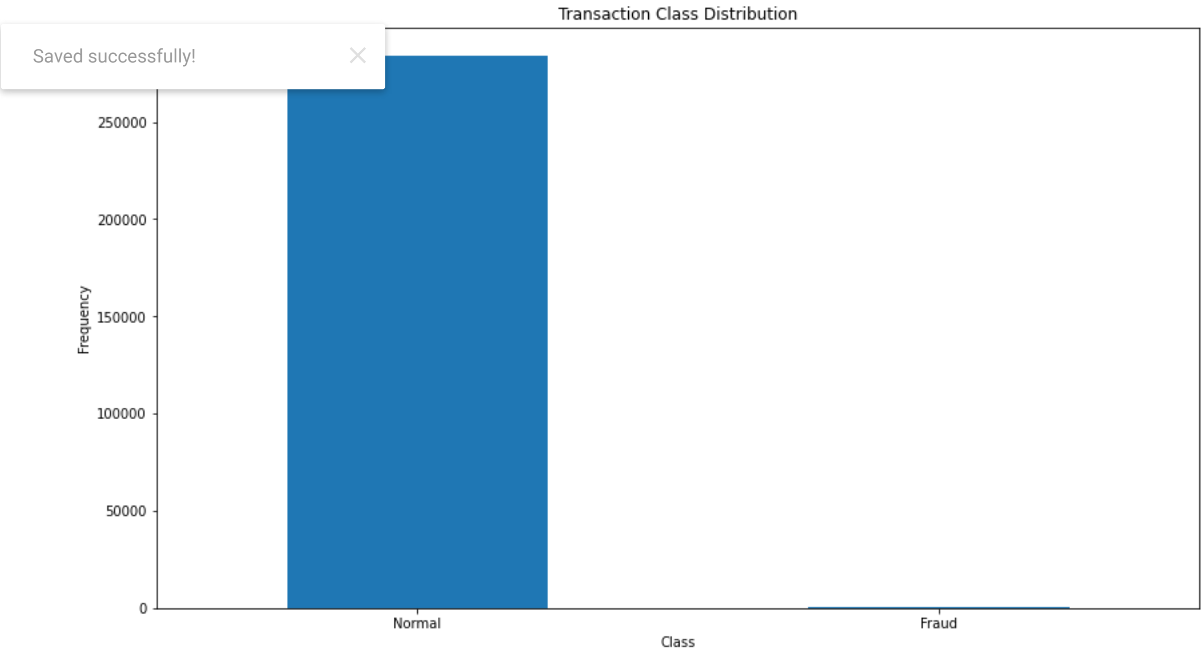
8 rows × 31 columns

```
df.isnull().values.any()
```

```
False
```

```
count_classes = pd.value_counts(df['Class'], sort = True)

count_classes.plot(kind = 'bar', rot=0)

plt.title("Transaction Class Distribution")

plt.xticks(range(2), LABELS)

plt.xlabel("Class")

plt.ylabel("Frequency")
```

```
Text(0, 0.5, 'Frequency')
```




```
fraud = df[df['Class']==1]

normal = df[df['Class']==0]


print(fraud.shape,normal.shape)
```

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

```
fraud.Amount.describe()
```

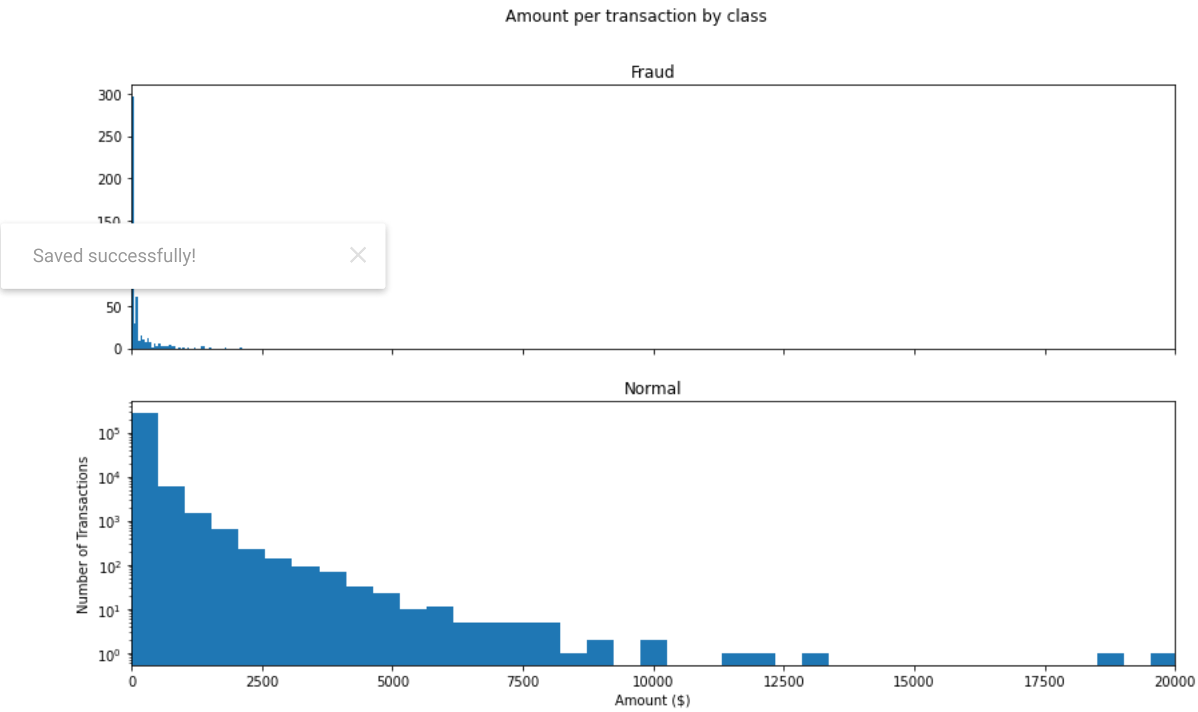
```
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
```

```
75%         105.890000
max        2125.870000
Name: Amount, dtype: float64
```

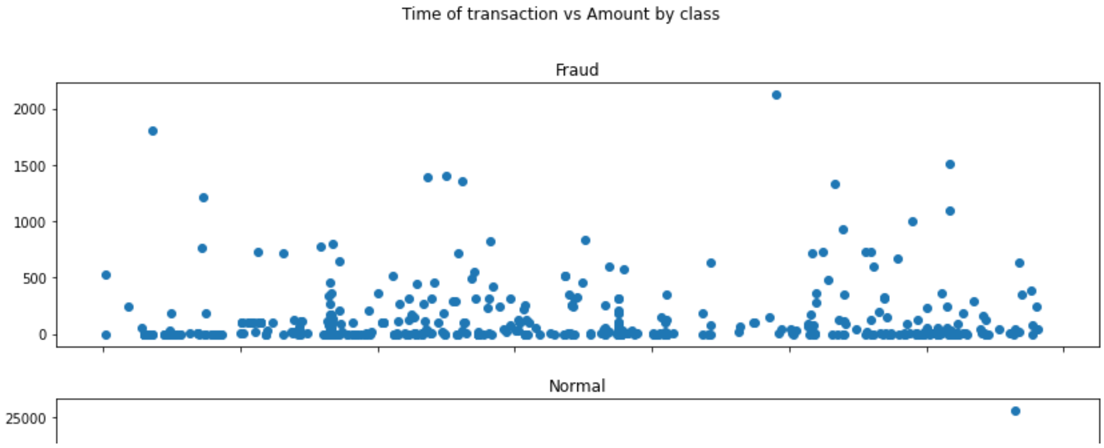
```
normal.Amount.describe()
```

```
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
```

```
f, (ax1, ax2) = plt.subplots(2, 1, sharex=True)
f.suptitle('Amount per transaction by class')
bins = 50
ax1.hist(fraud.Amount, bins = bins)
ax1.set_title('Fraud')
ax2.hist(normal.Amount, bins = bins)
ax2.set_title('Normal')
plt.xlabel('Amount ($)')
plt.ylabel('Number of Transactions')
plt.xlim((0, 20000))
plt.yscale('log')
plt.show();
```




```
# We Will check Do fraudulent transactions occur more often during certain time frame ? Let us find out with a visual representation.

f, (ax1, ax2) = plt.subplots(2, 1, sharex=True)
f.suptitle('Time of transaction vs Amount by class')
ax1.scatter(fraud.Time, fraud.Amount)
ax1.set_title('Fraud')
ax2.scatter(normal.Time, normal.Amount)
ax2.set_title('Normal')
plt.xlabel('Time (in Seconds)')
plt.ylabel('Amount')
plt.show()
```




```
df1= df.sample(frac = 0.1,random_state=1)
```

```
df1.shape
```

```
(28481, 31)
```


```
df.shape
```

```
(284807, 31)
```

```
Fraud = df1[df1['Class']==1]

Valid = df1[df1['Class']==0]

outlier_fraction = len(Fraud)/float(len(Valid))
```

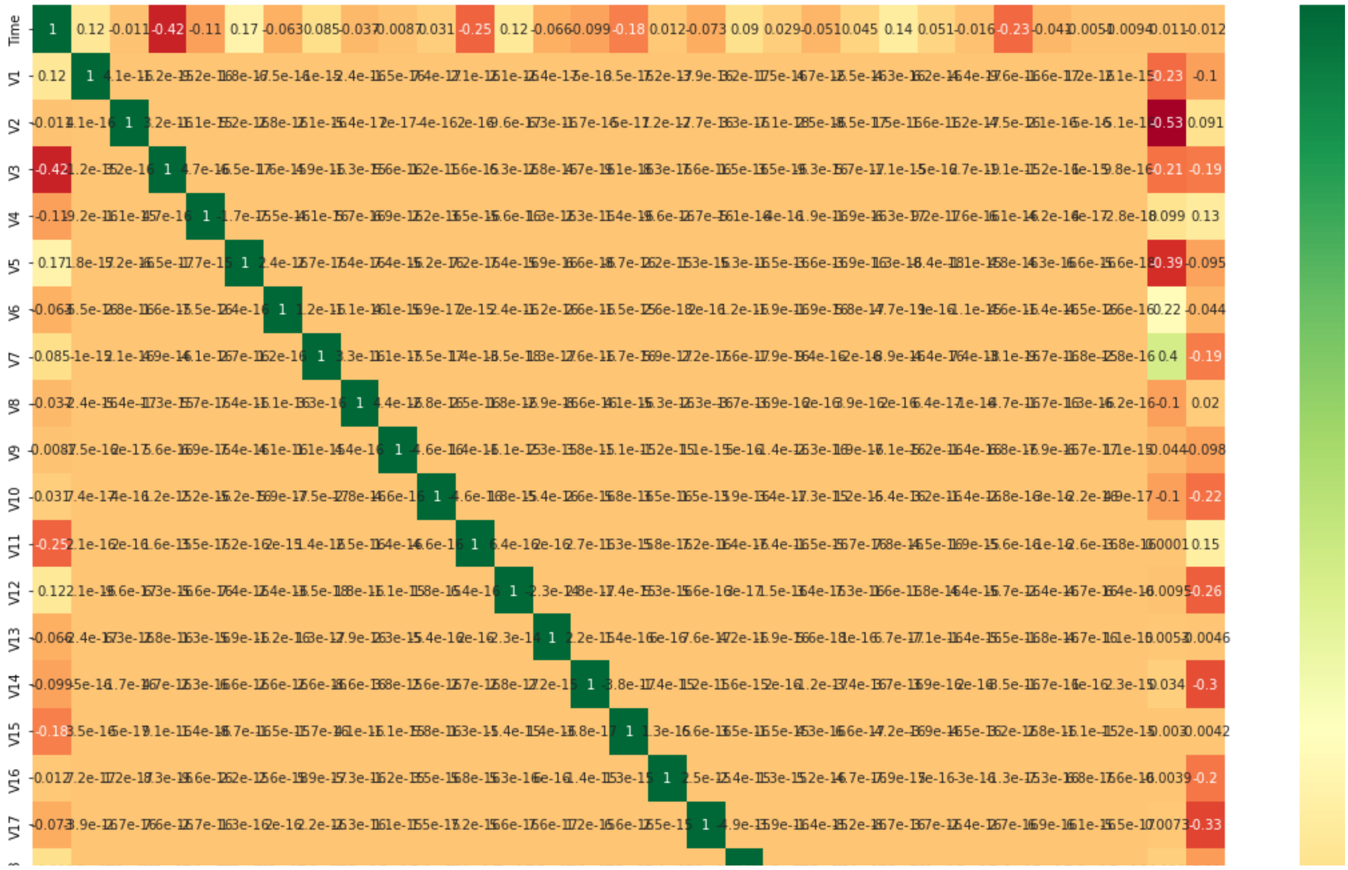
```
print(outlier_fraction)
```

Saved successfully!

```
Fraud)))
print("Valid Cases : {}".format(len(Valid)))
```

```
0.0017234102419808666
Fraud Cases : 49
Valid Cases : 28432
```

```
## Correlation
import seaborn as sns
#get correlations of each features in dataset
corrmat = df1.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```




```
#Create independent and Dependent Features
columns = df.columns.tolist()
# Filter the columns to remove data we do not want
                                       ot in ["Class"]]
                                       ng
target = "Class"
# Define a random state
state = np.random.RandomState(42)
X = df1[columns]
Y = df1[target]
X_outliers = state.uniform(low=0, high=1, size=(X.shape[0], X.shape[1]))
# Print the shapes of X & Y
print(X.shape)
print(Y.shape)
```

Saved successfully!

```
(28481, 30)
(28481,)
```

```
classifiers = {
    "Isolation Forest":IsolationForest(n_estimators=100, max_samples=len(X),
                                       contamination=outlier_fraction,random_state=state, verbose=0),
    "Local Outlier Factor":LocalOutlierFactor(n_neighbors=20, algorithm='auto',
                                              leaf_size=30, metric='minkowski',
                                              p=2, metric_params=None, contamination=outlier_fraction),
    # "Support Vector Machine":OneClassSVM(kernel='rbf', degree=3, gamma=0.1,nu=0.05,
    #                                      max_iter=-1, random_state=state)

}
```

```
# model = Classifiers(C=1.0, cache_size=200, class_weight=None, coef0=0.0, degree=3,
#              gamma=0.0, kernel='linear', max_iter=-1, probability=True,
#              random_state=None, shrinking=True, tol=0.001, verbose=False)
```

```
type(classifiers)
```

```
dict
```

```
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_prediction = clf.negative_outlier_factor_
    elif clf_name == "Support Vector Machine":
        clf.fit(X)
```

```
        clf.fit(X)
        y_pred = clf.predict(X)
    else:
        clf.fit(X)
        scores_prediction = clf.decision_function(X)
        y_pred = clf.predict(X)
    #Reshape the prediction values to 0 for Valid transactions , 1 for Fraud transactions
    y_pred[y_pred == 1] = 0
    y_pred[y_pred == -1] = 1
    n_errors = (y_pred != Y).sum()
    # Run Classification Metrics
    print("{}: {}".format(clf_name,n_errors))
    print("Accuracy Score :")
    print(accuracy_score(Y,y_pred))
    print("Classification Report :")
    print(classification_report(Y,y_pred))
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:420: UserWarning: X does not have valid feature names, but IsolationForest w
  warnings.warn(
Isolation Forest: 73
Accuracy Score :
0.9974368877497279
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28432
           1       0.26      0.27      0.26        49

    accuracy                           1.00     28481
   macro avg       0.63      0.63      0.63     28481
weighted avg       1.00      1.00      1.00     28481

Local Outlier Factor: 97
Accuracy Score :
0.9965942207085425
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28432
           1       0.02      0.02      0.02        49

                                       1.00     28481
   macro avg       0.51      0.51      0.51     28481
weighted avg       1.00      1.00      1.00     28481
```

Saved successfully!

Colab paid products - Cancel contracts here

✓ 15s completed at 12:33 AM