```python
# Loading Liberaries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Loading the data using pandas
credit_card_data =
pd.read_csv("/kaggle/input/creditcardfraud/creditcard.csv")

# data info
credit_card_data.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
```

```
credit_card_data.describe()
```

```
                Time            V1            V2            V3
V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05
2.848070e+05
mean    94813.859575  1.168375e-15  3.416908e-16 -1.379537e-15
2.074095e-15
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00
1.415869e+00
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -
5.683171e+00
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -
8.486401e-01
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -
1.984653e-02
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00
7.433413e-01
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00
1.687534e+01

                 V5            V6            V7            V8
V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
2.848070e+05
mean   9.604066e-16  1.487313e-15 -5.556467e-16  1.213481e-16 -
2.406331e-15
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00
1.098632e+00
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -
1.343407e+01
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -
6.430976e-01
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -
5.142873e-02
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01
5.971390e-01
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01
1.559499e+01

       ...           V21           V22           V23           V24  \
count  ...  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   ...  1.654067e-16 -3.568593e-16  2.578648e-16  4.473266e-15
std    ...  7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01
min    ... -3.483038e+01 -1.093314e+01 -4.480774e+01 -2.836627e+00
25%    ... -2.283949e-01 -5.423504e-01 -1.618463e-01 -3.545861e-01
50%    ... -2.945017e-02  6.781943e-03 -1.119293e-02  4.097606e-02
75%    ...  1.863772e-01  5.285536e-01  1.476421e-01  4.395266e-01
max    ...  2.720284e+01  1.050309e+01  2.252841e+01  4.584549e+00
```

```
                V25           V26           V27           V28
Amount  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
284807.000000
mean   5.340915e-16  1.683437e-15 -3.660091e-16 -1.227390e-16
88.349619
std    5.212781e-01  4.822270e-01  4.036325e-01  3.300833e-01
250.120109
min   -1.029540e+01 -2.604551e+00 -2.256568e+01 -1.543008e+01
0.000000
25%   -3.171451e-01 -3.269839e-01 -7.083953e-02 -5.295979e-02
5.600000
50%    1.659350e-02 -5.213911e-02  1.342146e-03  1.124383e-02
22.000000
75%    3.507156e-01  2.409522e-01  9.104512e-02  7.827995e-02
77.165000
max    7.519589e+00  3.517346e+00  3.161220e+01  3.384781e+01
25691.160000

               Class
count  284807.000000
mean        0.001727
std         0.041527
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000

[8 rows x 31 columns]
```

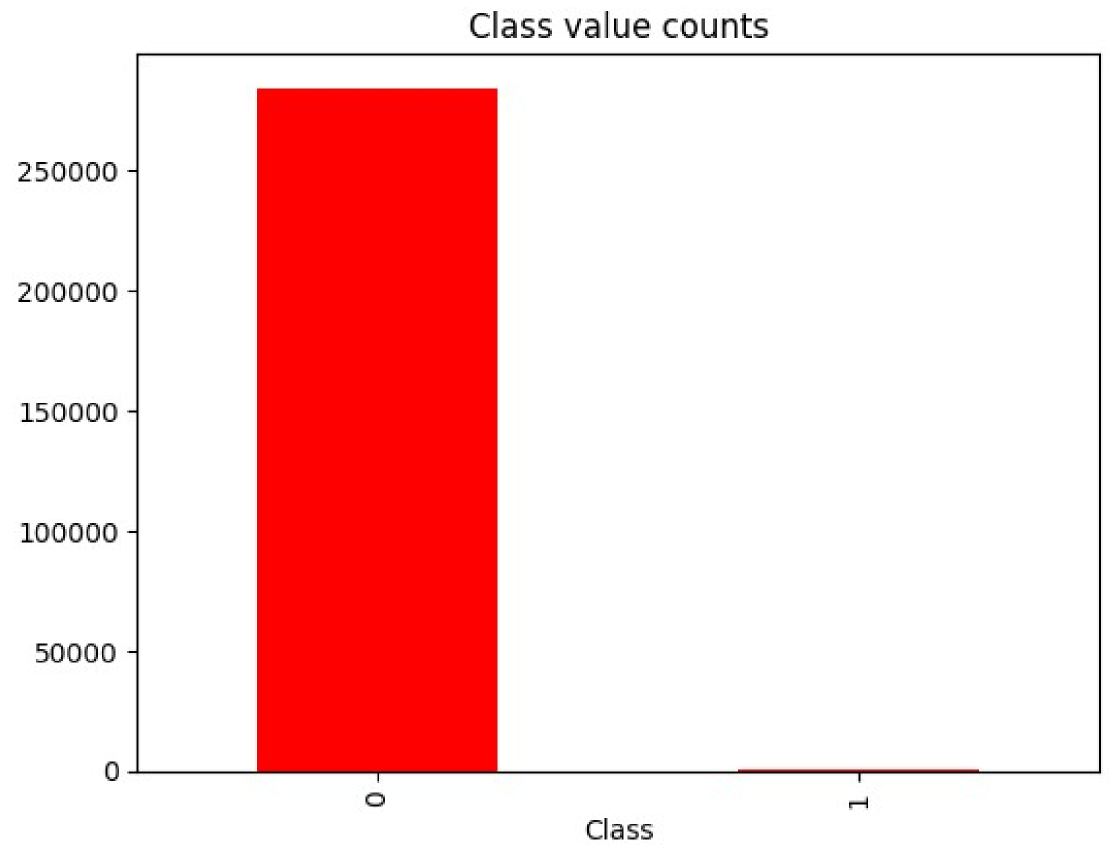
```
credit_card_data.isnull().sum()
```

```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
```

```
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

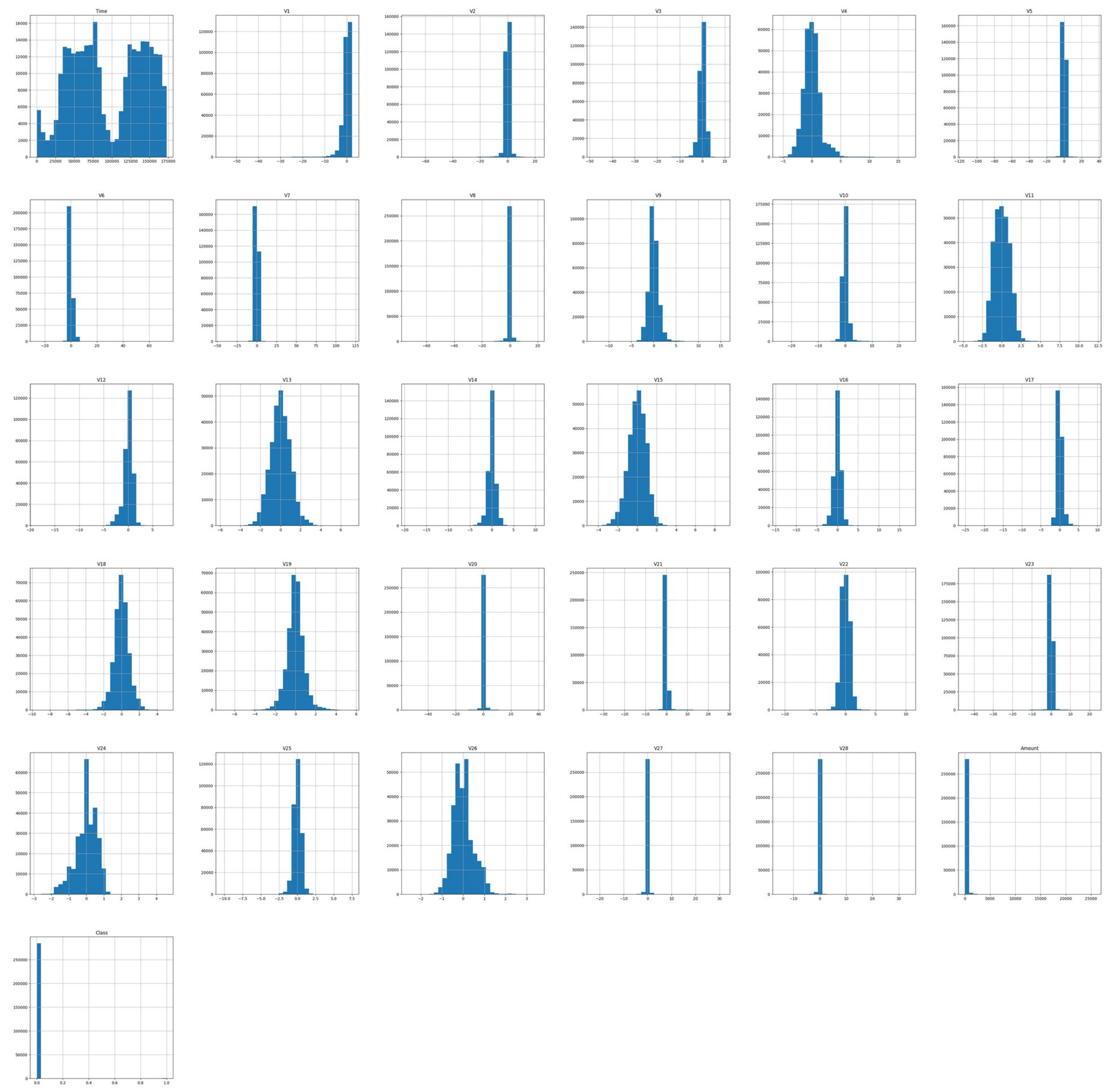
```
value_counts = credit_card_data['Class'].value_counts()

value_counts.plot.bar(title = 'Class value counts',color='r')
```

```
<Axes: title={'center': 'Class value counts'}, xlabel='Class'>
```



```
credit_card_data.hist(bins=30, figsize=(50,50))
```

```
array([[<Axes: title={'center': 'Time'}>, <Axes: title={'center':
'V1'}>,
        <Axes: title={'center': 'V2'}>, <Axes: title={'center':
'V3'}>,
        <Axes: title={'center': 'V4'}>, <Axes: title={'center':
'V5'}>],
       [<Axes: title={'center': 'V6'}>, <Axes: title={'center':
'V7'}>,
        <Axes: title={'center': 'V8'}>, <Axes: title={'center':
'V9'}>,
        <Axes: title={'center': 'V10'}>, <Axes: title={'center':
'V11'}>],
       [<Axes: title={'center': 'V12'}>, <Axes: title={'center':
'V13'}>,
        <Axes: title={'center': 'V14'}>, <Axes: title={'center':
'V15'}>,
        <Axes: title={'center': 'V16'}>, <Axes: title={'center':
'V17'}>],
       [<Axes: title={'center': 'V18'}>, <Axes: title={'center':
'V19'}>,
        <Axes: title={'center': 'V20'}>, <Axes: title={'center':
'V21'}>,
        <Axes: title={'center': 'V22'}>, <Axes: title={'center':
'V23'}>],
       [<Axes: title={'center': 'V24'}>, <Axes: title={'center':
'V25'}>,
        <Axes: title={'center': 'V26'}>, <Axes: title={'center':
'V27'}>,
        <Axes: title={'center': 'V28'}>,
        <Axes: title={'center': 'Amount'}>],
       [<Axes: title={'center': 'Class'}>, <Axes: >, <Axes: >, <Axes:
>,
        <Axes: >, <Axes: >]], dtype=object)
```



```
credit_card_data[["Time","Amount"]].describe()
```

```
                 Time         Amount
count   284807.000000  284807.000000
mean     94813.859575      88.349619
std      47488.145955     250.120109
min          0.000000       0.000000
25%      54201.500000       5.600000
50%      84692.000000      22.000000
75%     139320.500000      77.165000
max     172792.000000   25691.160000
```

```
# scaling Time and Amount
from sklearn.preprocessing import StandardScaler
credit_card_data["Amount"]=
StandardScaler().fit_transform(credit_card_data["Amount"].values.resha
pe(-1, 1))
credit_card_data["Time"]=
StandardScaler().fit_transform(credit_card_data["Time"].values.reshape
(-1, 1))

credit_card_data[["Time","Amount"]].describe()
```

```
               Time        Amount
count  2.848070e+05  2.848070e+05
mean  -3.065637e-16  2.913952e-17
std    1.000002e+00  1.000002e+00
min   -1.996583e+00 -3.532294e-01
25%   -8.552120e-01 -3.308401e-01
50%   -2.131453e-01 -2.652715e-01
75%    9.372174e-01 -4.471707e-02
max    1.642058e+00  1.023622e+02
```

```
X=credit_card_data.drop("Class", axis=1)
y=credit_card_data["Class"].values.reshape(-1,1)



from imblearn.over_sampling import SMOTE
smote = SMOTE(sampling_strategy='auto', random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

from sklearn.model_selection import train_test_split
X_train_resampled,X_test,y_train_resampled,y_test=
train_test_split(X_resampled,y_resampled)

from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(X_train_resampled, y_train_resampled)
y_pred=model.predict(X_test)

from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.92      0.98      0.95     70939
           1       0.97      0.92      0.95     71219

    accuracy                           0.95    142158
   macro avg       0.95      0.95      0.95    142158
weighted avg       0.95      0.95      0.95    142158
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, y_pred)
```

```
0.9481624082732796
```

```
 from sklearn.metrics import precision_score, recall_score
print(precision_score(y_test,y_pred))
print(recall_score(y_test,y_pred))
```

```
0.9749583382930603
0.9200494250129881
```

```
from sklearn.metrics import f1_score
f1_score(y_test,y_pred)
```

```
0.9467083733664675
```

```
from sklearn.linear_model import SGDClassifier
sgd_clf = SGDClassifier(random_state=42)
sgd_clf.fit(X_train_resampled, y_train_resampled)
y_pred2 = sgd_clf.predict(X_test)
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, y_pred2)
```

```
0.9487041679988829
```

```
from sklearn.metrics import precision_score, recall_score
print(precision_score(y_test,y_pred2))
print(recall_score(y_test,y_pred2))
```

```
0.9720269392095469
0.9241073309088866
```

```
from sklearn.metrics import f1_score
f1_score(y_test,y_pred2)
```

```
0.9474616165324236
```