DRONACHARYA
College of Engineering

PROJECT-2 REPORT

ON

Credit card Fraud detection

Prepared By:

- Sushant Sharma (22161)
- Lakhan Singh (22080)

Submitted To:

Name: Prof. Naveen Rathi

Dept.: CSE

Department of Computer Science and Engineering

DRONACHARYA COLLEGE OF ENGINEERING

FARUKHNAGAR, GURUGRAM, HARYANA

Problem Statement:

The Credit Card Fraud Detection Problem includes modeling past credit card transactions with the knowledge of the ones that turned out to be a fraud. This model is then used to identify whether a new transaction is fraudulent or not. Our aim here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classifications.

What is Credit Card Fraud?

Credit card fraud is when someone uses another person's credit card or account information to make unauthorized purchases or access funds through cash advances. Credit card fraud doesn't just happen online; it happens in brick-and-mortar stores, too. As a business owner, you can avoid serious headaches – and unwanted publicity – by recognizing potentially fraudulent use of credit cards in your payment environment.

Observations

- Very few transactions are actually fraudulent (less than 1%). The data set is highly skewed, consisting of 492 frauds in a total of 284,807 observations. This resulted in only 0.172% fraud cases. This skewed set is justified by the low number of fraudulent transactions.
- The dataset consists of numerical values from the 28 'Principal Component Analysis (PCA)' transformed features, namely V1 to V28. Furthermore, there is no metadata about the original features provided, so pre-analysis or feature study could not be done.
- The 'Time' and 'Amount' features are not transformed data.
- There is no missing value in the dataset.

IMPORTING LIBRARIES:

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pylab import rcParams
import warnings
warnings.filterwarnings('ignore')
```

READING DATASET :

In [2]:

```
data=pd.read_csv('/kaggle/input/creditcardfraud/creditcard.csv')
```

In [3]:

```
data.head()
```

Out[3]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-	...	0.247998	0.771679	0.909412
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458

5 rows × 31 columns

NULL VALUES:

In [4]:

```
data.isnull().sum()
```

Out[4]:

```
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
```

Thus there are no null values in the dataset.

INFORMATION

In [5]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
Time      284807 non-null float64
V1        284807 non-null float64
V2        284807 non-null float64
V3        284807 non-null float64
V4        284807 non-null float64
V5        284807 non-null float64
V6        284807 non-null float64
V7        284807 non-null float64
V8        284807 non-null float64
V9        284807 non-null float64
V10       284807 non-null float64
V11       284807 non-null float64
V12       284807 non-null float64
V13       284807 non-null float64
V14       284807 non-null float64
V15       284807 non-null float64
V16       284807 non-null float64
V17       284807 non-null float64
V18       284807 non-null float64
V19       284807 non-null float64
V20       284807 non-null float64
V21       284807 non-null float64
V22       284807 non-null float64
V23       284807 non-null float64
V24       284807 non-null float64
V25       284807 non-null float64
V26       284807 non-null float64
V27       284807 non-null float64
V28       284807 non-null float64
Amount    284807 non-null float64
Class     284807 non-null int64
```

```
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

DRONACHARYA
College of Engineering

DESCRIPTIVE STATISTICS

In [6]:

```
data.describe().T.head()
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
Time	284807.0	9.481386e+04	47488.145955	0.000000	54201.500000	84692.000000	139320.500000	172792.000000
V1	284807.0	3.919560e-15	1.958696	-56.407510	-0.920373	0.018109	1.315642	2.454930
V2	284807.0	5.688174e-16	1.651309	-72.715728	-0.598550	0.065486	0.803724	22.057729
V3	284807.0	-8.769071e-15	1.516255	-48.325589	-0.890365	0.179846	1.027196	9.382558
V4	284807.0	2.782312e-15	1.415869	-5.683171	-0.848640	-0.019847	0.743341	16.875344

In [7]:

```
data.shape
```

Out[7]:

```
(284807, 31)
```

Thus there are 284807 rows and 31 columns.

In [8]:

```
data.columns
```

Out[8]:

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

FRAUD CASES AND GENUINE CASES

In [9]:

```
fraud_cases=len(data[data['Class']==1])
```

In [10]:

```
print(' Number of Fraud Cases:',fraud_cases)
```

```
 Number of Fraud Cases: 492
```

In [11]:

```
non_fraud_cases=len(data[data['Class']==0])
```

In [12]:

```
print('Number of Non Fraud Cases:',non_fraud_cases)
```

```
Number of Non Fraud Cases: 284315
```

In [13]:

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

In [14]:

```
genuine=data[data['Class']==0]
```

In [15]:

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

Out[15]:

```
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 [16]:

```
genuine.Amount.describe()
```

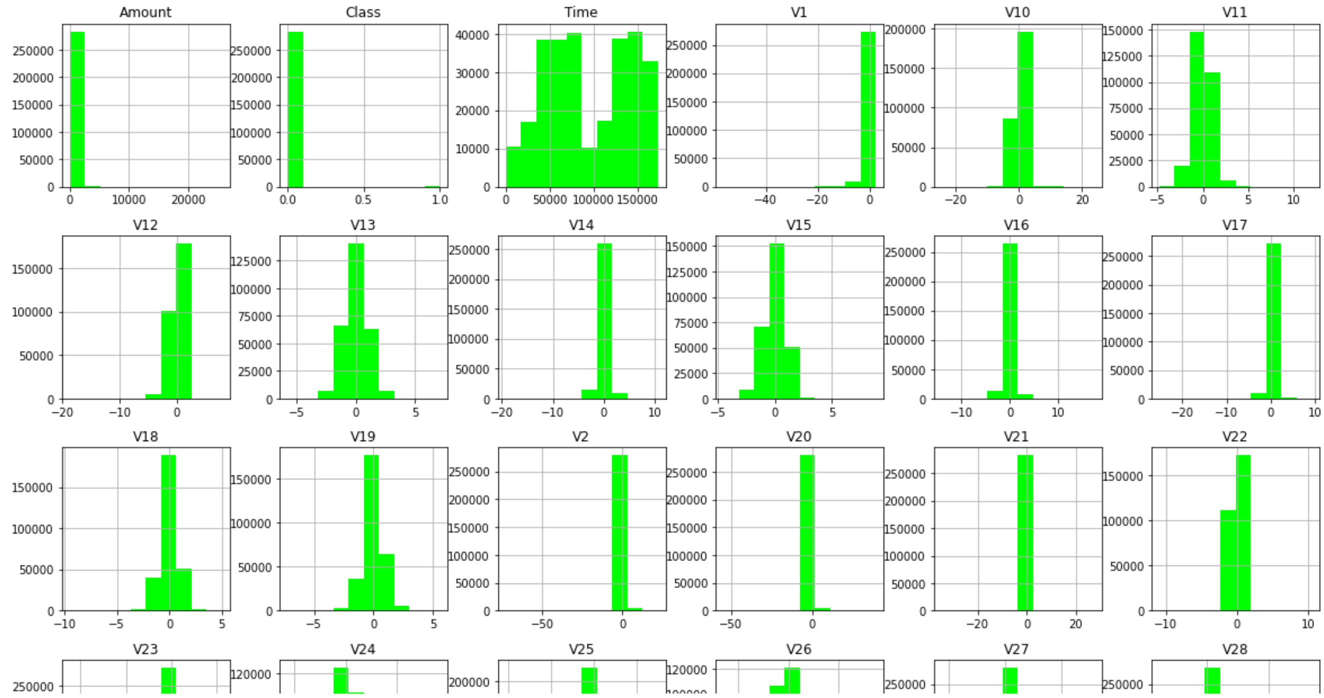
Out[16]:

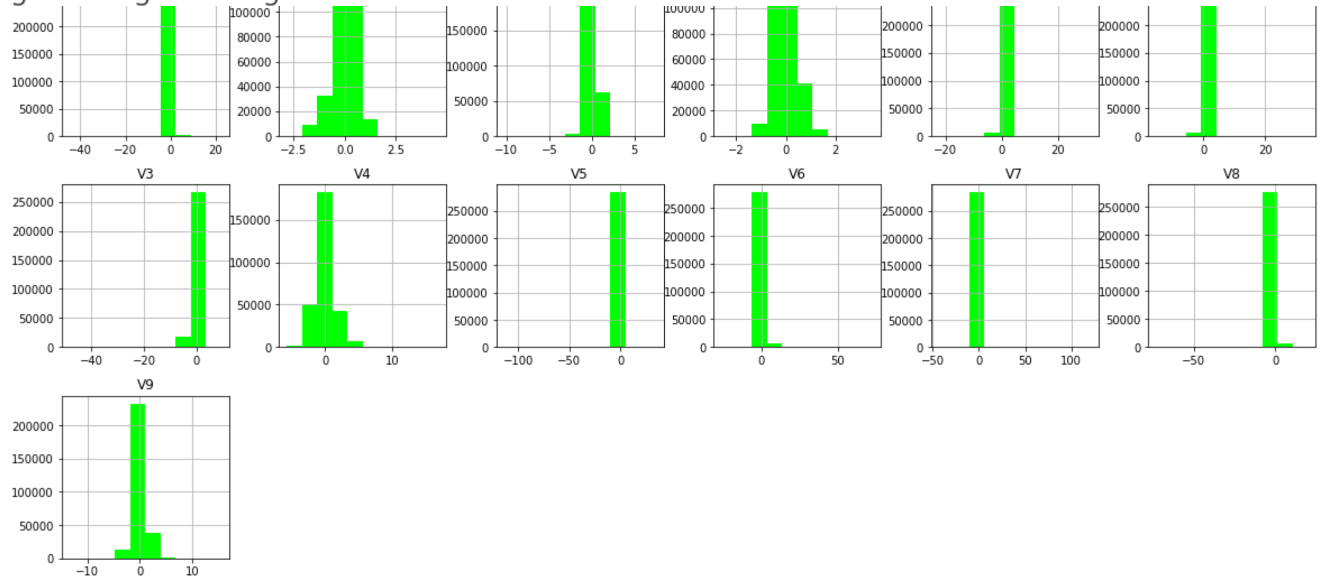
```
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

EDA

In [17]:

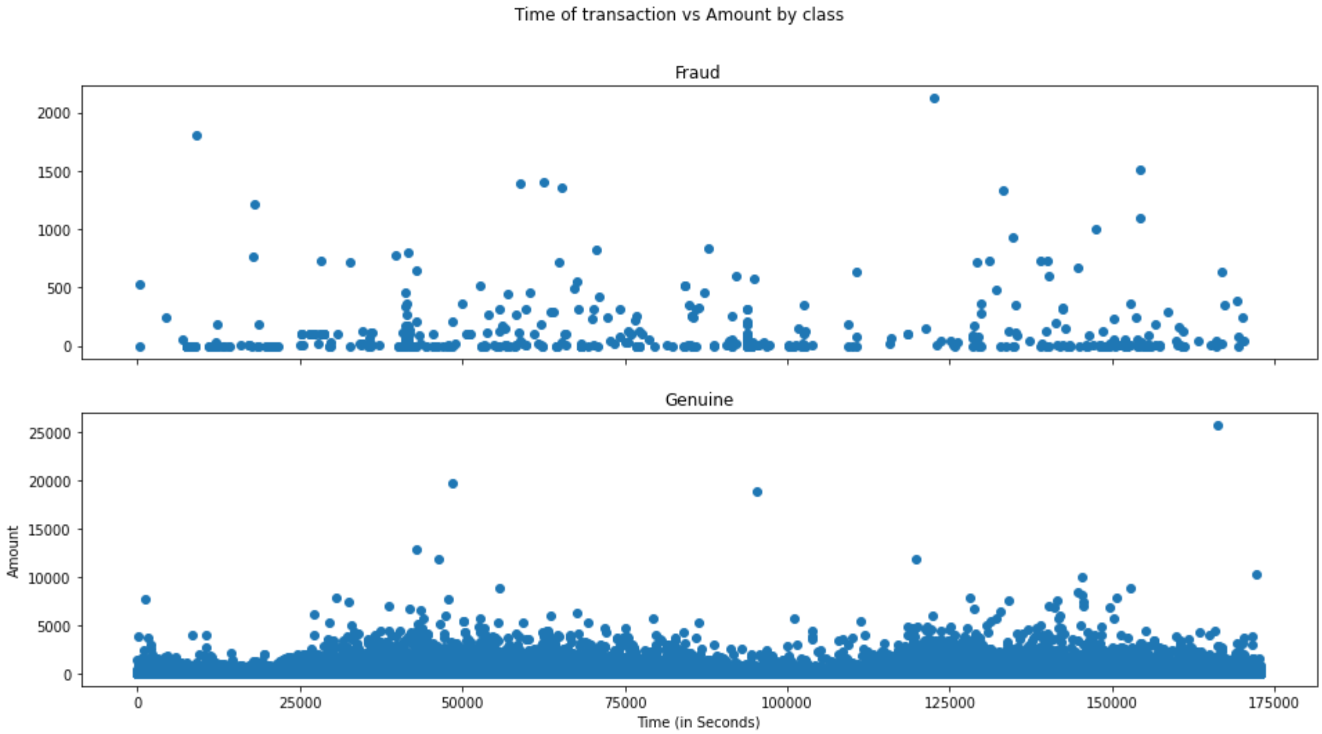
```
data.hist(figsize=(20,20),color='lime')
plt.show()
```





In [18]:

```
rcParams['figure.figsize'] = 16, 8
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(genuine.Time, genuine.Amount)
ax2.set_title('Genuine')
plt.xlabel('Time (in Seconds)')
plt.ylabel('Amount')
plt.show()
```



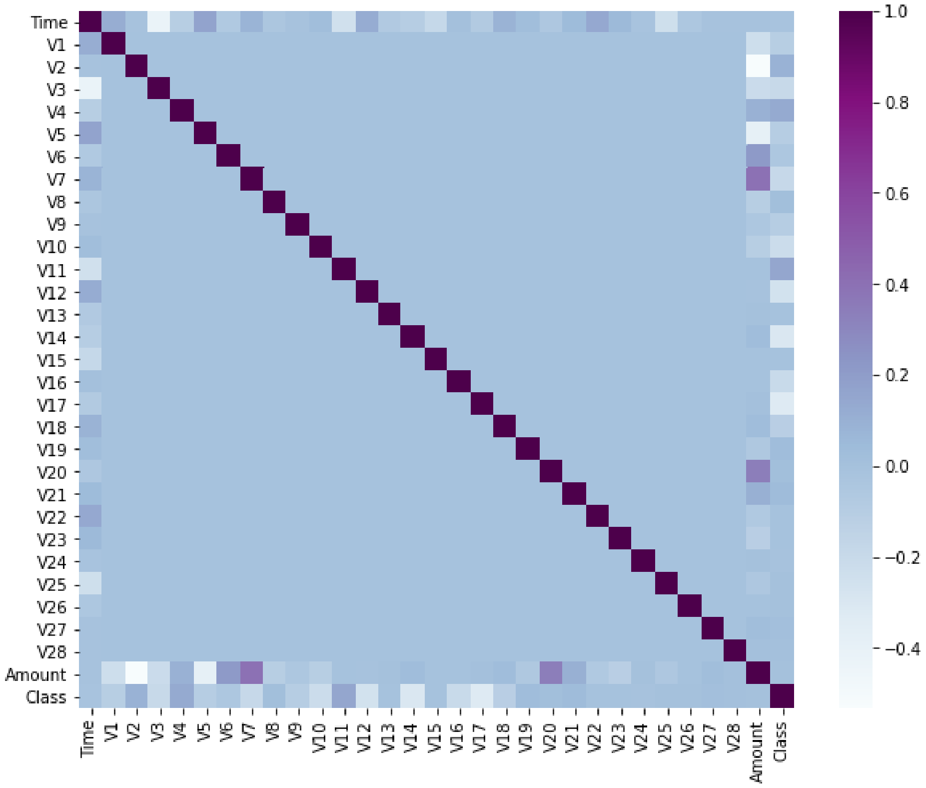
CORRELATION

In [19]:

```
plt.figure(figsize=(10,8))
corr=data.corr()
sns.heatmap(corr,cmap='BuPu')
```

Out[19]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f4c890f89b0>



Let us build our models:

In [20]:

```
from sklearn.model_selection import train_test_split
```

Model 1:

In [21]:

```
X=data.drop(['Class'],axis=1)
```

In [22]:

```
y=data['Class']
```

In [23]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.30,random_state=123)
```

In [24]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [25]:

```
rfc=RandomForestClassifier()
```

In [26]:

```
model=rfc.fit(X_train,y_train)
```

In [27]:

```
prediction=model.predict(X_test)
```

In [28]:

```
from sklearn.metrics import accuracy_score
```

In [29]:

```
accuracy_score(y_test,prediction)
```

Out[29]:

```
0.9995786664794073
```

Model 2:

In [30]:

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

In [31]:

```
X1=data.drop(['Class'],axis=1)
```

In [32]:

```
y1=data['Class']
```

In [33]:

```
X1_train,X1_test,y1_train,y1_test=train_test_split(X1,y1,test_size=0.3,random_state=123)
```

In [34]:

```
lr=LogisticRegression()
```

In [35]:

```
model2=lr.fit(X1_train,y1_train)
```

In [36]:

```
prediction2=model2.predict(X1_test)
```

In [37]:

```
accuracy_score(y1_test,prediction2)
```

Out[37]:

```
0.9988764439450862
```

Model 3:

In [38]:

```
from sklearn.tree import DecisionTreeRegressor
```

In [39]:

```
X2=data.drop(['Class'],axis=1)
```

In [40]:

```
y2=data['Class']
```

In [41]:

```
dt=DecisionTreeRegressor()
```

In [42]:

```
X2_train,X2_test,y2_train,y2_test=train_test_split(X2,y2,test_size=0.3,random_state=123)
```

In [43]:

```
model3=dt.fit(X2_train,y2_train)
```

In [44]:

```
prediction3=model3.predict(X2_test)
```

In [45]:

```
accuracy_score(y2_test,prediction3)
```

Out[45]:

```
0.999133925541004
```

Overall models performed with a very high accuracy.

In [ ]:

```

```