Author : Srinithi Saiprasath

Project : Credit Card Fraud Detection

Email : scsrinithi@gmail.com

IMPORTING LIBRARIES:

In [ ]:
```
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 [ ]:
```
data=pd.read_csv("F:/PROJECTS/Credit_Card_Fraud_Detection/creditcard.csv")
```

In [ ]:
```
data.head()
```

Out[ ]:

	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	0.0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.2
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.3
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.2

5 rows × 31 columns

NULL VALUES:

In [ ]:
```
data.isnull().sum()
```

Out[ ]:
```
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 [ ]:
```
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
```

DESCRIPTIVE STATISTICS

In [ ]:
```
data.describe().T.head()
```

Out[ ]:

	count	mean	std	min	25%	50%	
Time	284807.0	9.481386e+04	47488.145955	0.000000	54201.500000	84692.000000	13
V1	284807.0	1.168375e-15	1.958696	-56.407510	-0.920373	0.018109	
V2	284807.0	3.416908e-16	1.651309	-72.715728	-0.598550	0.065486	
V3	284807.0	-1.379537e-15	1.516255	-48.325589	-0.890365	0.179846	
V4	284807.0	2.074095e-15	1.415869	-5.683171	-0.848640	-0.019847	

In [ ]:
```
data.shape
```

Out[ ]: (284807, 31)

Thus there are 284807 rows and 31 columns.

In [ ]:
```
data.columns
```

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

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

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

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

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

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

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

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

In [ ]:
```
fraud.Amount.describe()
```

Out[ ]:
```
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
```

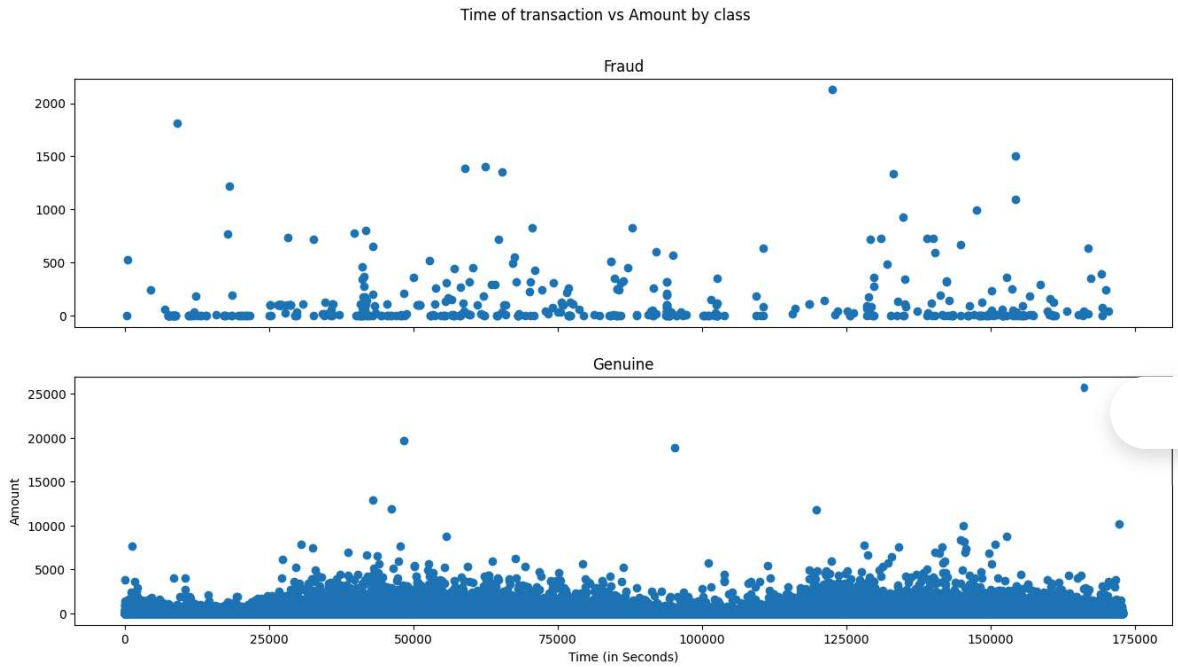
In [ ]:
```
genuine.Amount.describe()
```

Out[ ]:
```
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 [ ]:
```
data.hist(figsize=(20,20),color='lime')
plt.show()
```

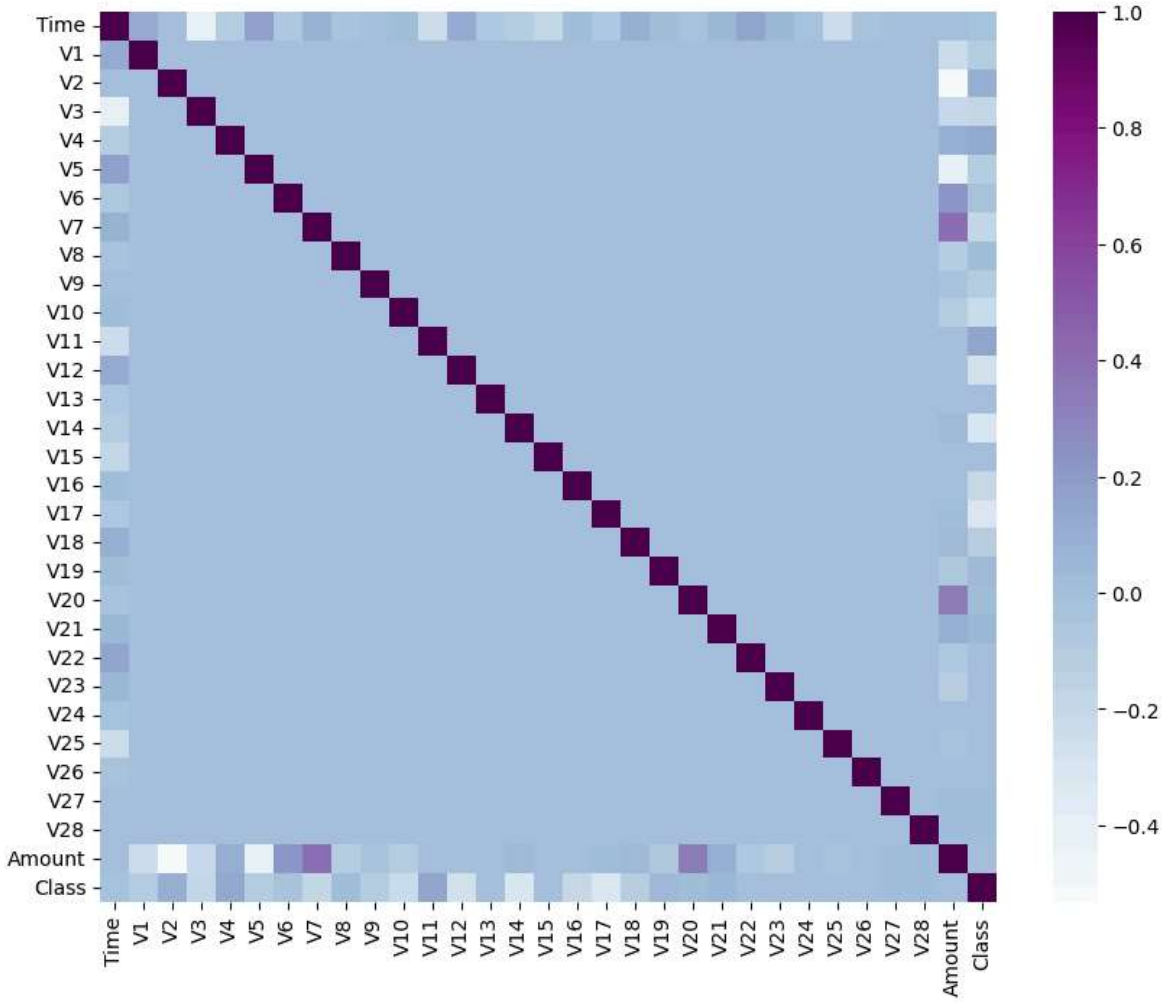
In [ ]:
```
rcParams['figure.figsize'] = 16, 8
f,(ax1, ax2) = plt.subplots(2, 1, sharex=True)
f.suptitle('Time of transaction vs Amount by class')
ax1.scatter(fraud.Time, fraud.Amount)
ax1.set_title('Fraud')
ax2.scatter(genuine.Time, genuine.Amount)
ax2.set_title('Genuine')
plt.xlabel('Time (in Seconds)')
plt.ylabel('Amount')
plt.show()
```



CORRELATION

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

Out[ ]: <Axes: >



Our models

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

Model 1 drafted using Random Forest Algorithm

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

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

In [ ]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.30,random_sta
```

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

In [ ]:
```
rfc=RandomForestClassifier()
```

In [ ]:
```
model=rfc.fit(X_train,y_train)
```

In [ ]:
```
prediction=model.predict(X_test)
```

In [ ]:
```
from sklearn.metrics import accuracy_score
```

In [ ]:
```
accuracy_score(y_test,prediction)
model1_acc = accuracy_score(y_test,prediction)
```

Model 2 Drafted with Regression

In [ ]:
```
from sklearn.linear_model import LogisticRegression
```

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

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

In [ ]:
```
X1_train,X1_test,y1_train,y1_test=train_test_split(X1,y1,test_size=0.3,random_st
```

In [ ]:
```
lr=LogisticRegression()
```

In [ ]:
```
model2=lr.fit(X1_train,y1_train)
```

In [ ]:
```
prediction2=model2.predict(X1_test)
```

In [ ]:
```
accuracy_score(y1_test,prediction2)
model2_acc = accuracy_score(y1_test,prediction2)
```

Model 3:

In [ ]:
```
from sklearn.tree import DecisionTreeRegressor
```

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

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

In [ ]:
```
dt=DecisionTreeRegressor()
```

In [ ]:
```
X2_train,X2_test,y2_train,y2_test=train_test_split(X2,y2,test_size=0.3,random_st
```

In [ ]:
```
model3=dt.fit(X2_train,y2_train)
```

In [ ]:
```
prediction3=model3.predict(X2_test)
```

In [ ]:
```
accuracy_score(y2_test,prediction3)
model3_acc = accuracy_score(y2_test,prediction3)
```

Comparing all 3 Algorithms and plotting thier accuracies

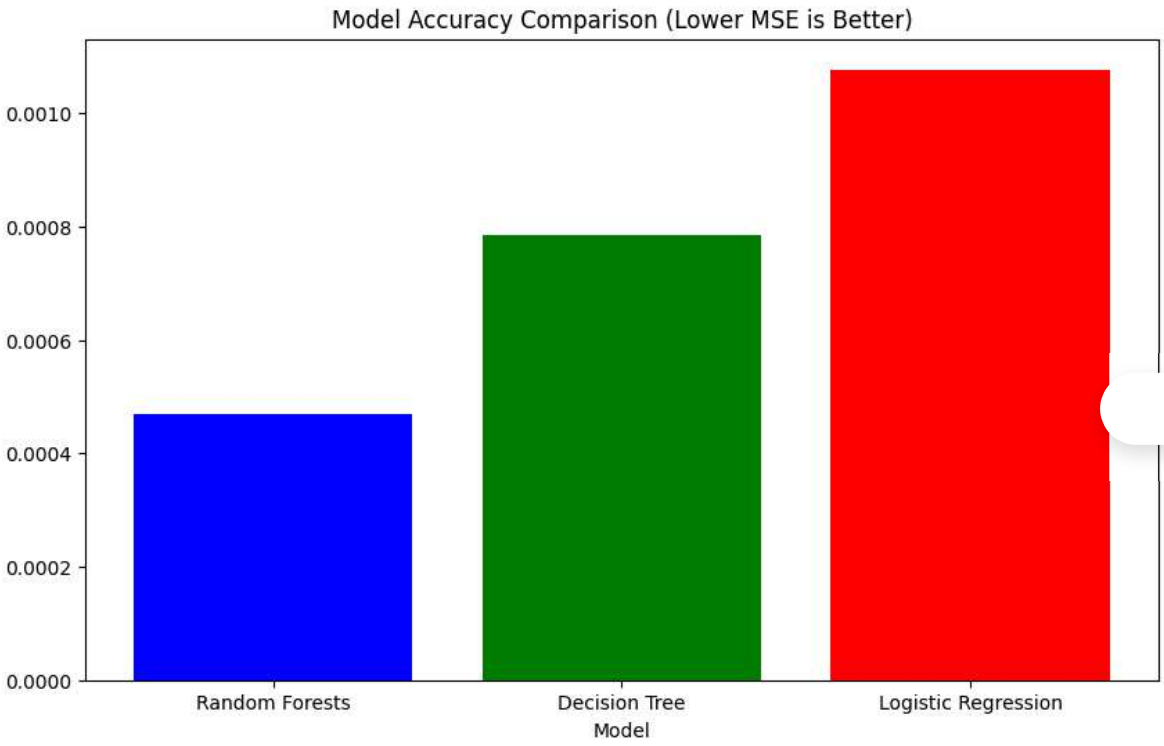
In [ ]:
```
from sklearn.metrics import mean_squared_error
mse_rf = mean_squared_error(y_test,prediction)
mse_lr = mean_squared_error(y1_test,prediction2)
mse_dt = mean_squared_error(y2_test,prediction3)
```

In [ ]:
```
print("Mean squared Error of Random Forest:" , mse_rf)
print("Mean squared Error of Decision Trees:" , mse_dt)
print("Mean squared Error of logistic Regression :" , mse_lr)
```

```
Mean squared Error of Random Forest: 0.00046814835621408426
Mean squared Error of Decision Trees: 0.0007841484966585911
Mean squared Error of logistic Regression : 0.0010767412192923937
```

In [ ]:
```
models = ["Random Forests"  ,"Decision Tree" , "Logistic Regression"]
accuracy_score = [model1_acc ,model2_acc , model3_acc]
mse_val = [ mse_rf ,mse_dt , mse_lr]
```

In [ ]:
```
plt.figure(figsize=(10, 6))
plt.bar(models, mse_val, color=['blue', 'green','red'])
plt.xlabel('Model')
plt.ylabel('')
plt.title('Model Accuracy Comparison (Lower MSE is Better)')
plt.show()
```


Model Accuracy Comparison (Lower MSE is Better)
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Random Forests
Decision Tree
Logistic Regression
Model