Abdullah Bilal _ ML BSAI_5A

22108164

In [84]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```python
df = pd.read_csv("creditcard.csv")
```

In [3]:

```python
df.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 [4]:

```
df.describe().T
```

Out[4]:

	count	mean	std	min	25%	50%
Time	284807.0	9.481386e+04	47488.145955	0.000000	54201.500000	84692.000000
V1	284807.0	3.918649e-15	1.958696	-56.407510	-0.920373	0.018109
V2	284807.0	5.682686e-16	1.651309	-72.715728	-0.598550	0.065486
V3	284807.0	-8.761736e-15	1.516255	-48.325589	-0.890365	0.179846
V4	284807.0	2.811118e-15	1.415869	-5.683171	-0.848640	-0.019847
V5	284807.0	-1.552103e-15	1.380247	-113.743307	-0.691597	-0.054336
V6	284807.0	2.040130e-15	1.332271	-26.160506	-0.768296	-0.274187
V7	284807.0	-1.698953e-15	1.237094	-43.557242	-0.554076	0.040103
V8	284807.0	-1.893285e-16	1.194353	-73.216718	-0.208630	0.022358
V9	284807.0	-3.147640e-15	1.098632	-13.434066	-0.643098	-0.051429
V10	284807.0	1.772925e-15	1.088850	-24.588262	-0.535426	-0.092917
V11	284807.0	9.289524e-16	1.020713	-4.797473	-0.762494	-0.032757
V12	284807.0	-1.803266e-15	0.999201	-18.683715	-0.405571	0.140033
V13	284807.0	1.674888e-15	0.995274	-5.791881	-0.648539	-0.013568
V14	284807.0	1.475621e-15	0.958596	-19.214325	-0.425574	0.050601
V15	284807.0	3.501098e-15	0.915316	-4.498945	-0.582884	0.048072
V16	284807.0	1.392460e-15	0.876253	-14.129855	-0.468037	0.066413
V17	284807.0	-7.466538e-16	0.849337	-25.162799	-0.483748	-0.065676
V18	284807.0	4.258754e-16	0.838176	-9.498746	-0.498850	-0.003636
V19	284807.0	9.019919e-16	0.814041	-7.213527	-0.456299	0.003735
V20	284807.0	5.126845e-16	0.770925	-54.497720	-0.211721	-0.062481
V21	284807.0	1.473120e-16	0.734524	-34.830382	-0.228395	-0.029450
V22	284807.0	8.042109e-16	0.725702	-10.933144	-0.542350	0.006782
V23	284807.0	5.282512e-16	0.624460	-44.807735	-0.161846	-0.011193
V24	284807.0	4.456271e-15	0.605647	-2.836627	-0.354586	0.040976
V25	284807.0	1.426896e-15	0.521278	-10.295397	-0.317145	0.016594
V26	284807.0	1.701640e-15	0.482227	-2.604551	-0.326984	-0.052139
V27	284807.0	-3.662252e-16	0.403632	-22.565679	-0.070840	0.001342
V28	284807.0	-1.217809e-16	0.330083	-15.430084	-0.052960	0.011244
Amount	284807.0	8.834962e+01	250.120109	0.000000	5.600000	22.000000
Class	284807.0	1.727486e-03	0.041527	0.000000	0.000000	0.000000

```
In [6]: df.shape
```

```
Out[6]: (284807, 31)
```

```
In [7]: df.isnull().sum()
```

```
Out[7]: 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
```

```
In [8]: df.duplicated().sum()
```

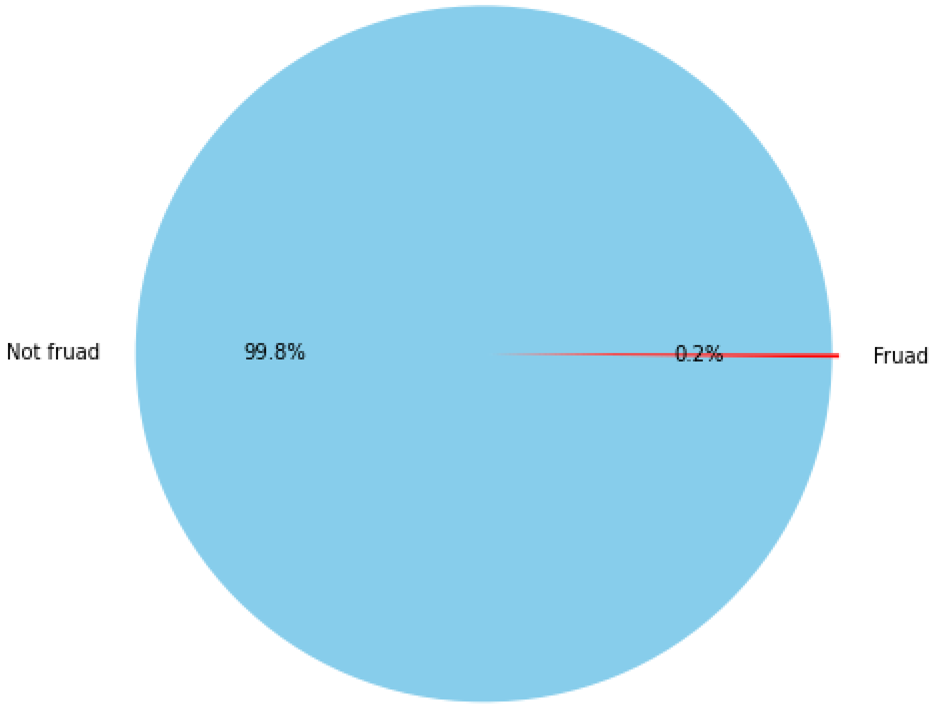
```
Out[8]: 1081
```

```
In [9]: df.drop_duplicates(inplace = True)
```

```
In [10]: df['Class'].value_counts()
```

```
Out[10]: 0    283253
         1       473
         Name: Class, dtype: int64
```

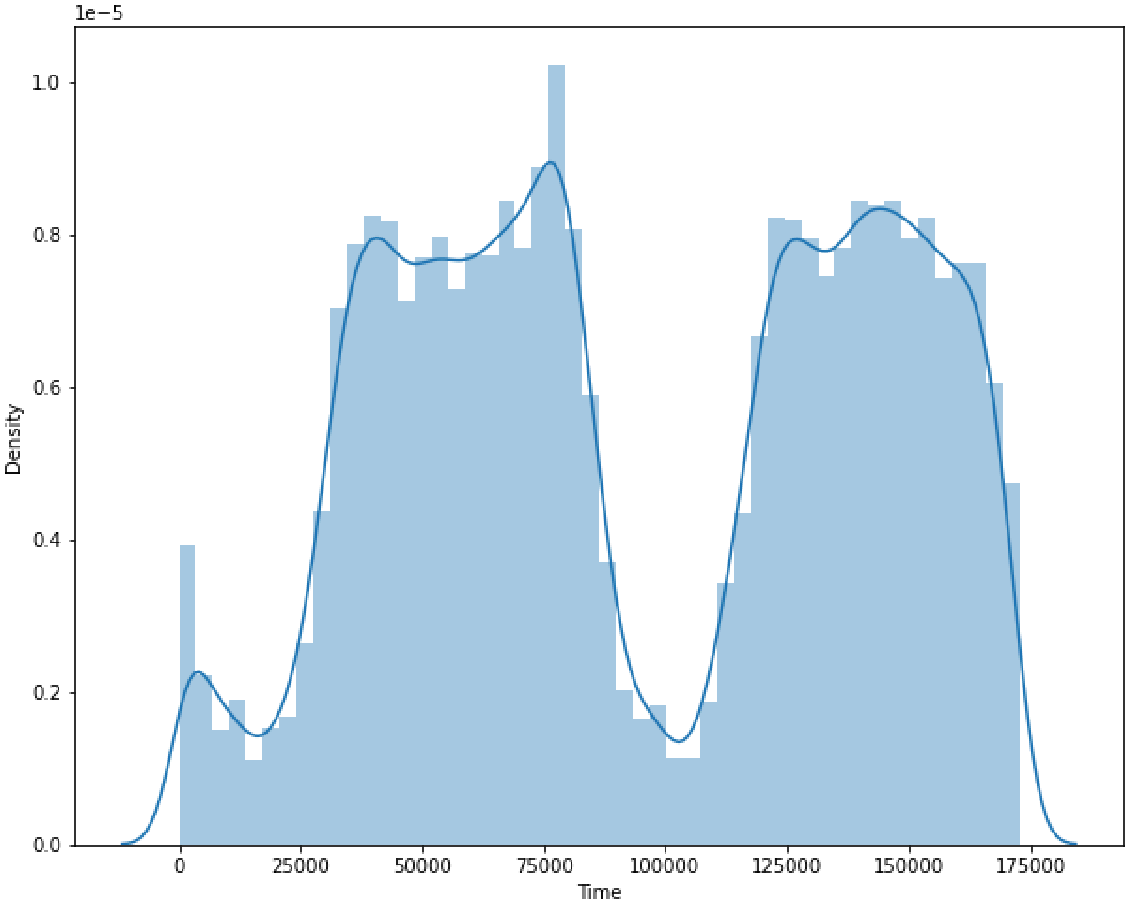
In [11]:
```
plt.figure(figsize=(10,8))
labels = ['Not fruad','Fruad']
color = ['skyblue','red']
ex = [.01,.01]
sizes = df.Class.value_counts().values
plt.pie(sizes,ex,labels,autopct='%1.1f%%',colors=color)
plt.show()
```



In [12]:
```
plt.figure(figsize=(10,8))
sns.distplot(df['Time'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:261
9: FutureWarning: `distplot` is a deprecated function and will be remov
ed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an ax
es-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[12]:
```
<AxesSubplot:xlabel='Time', ylabel='Density'>
```

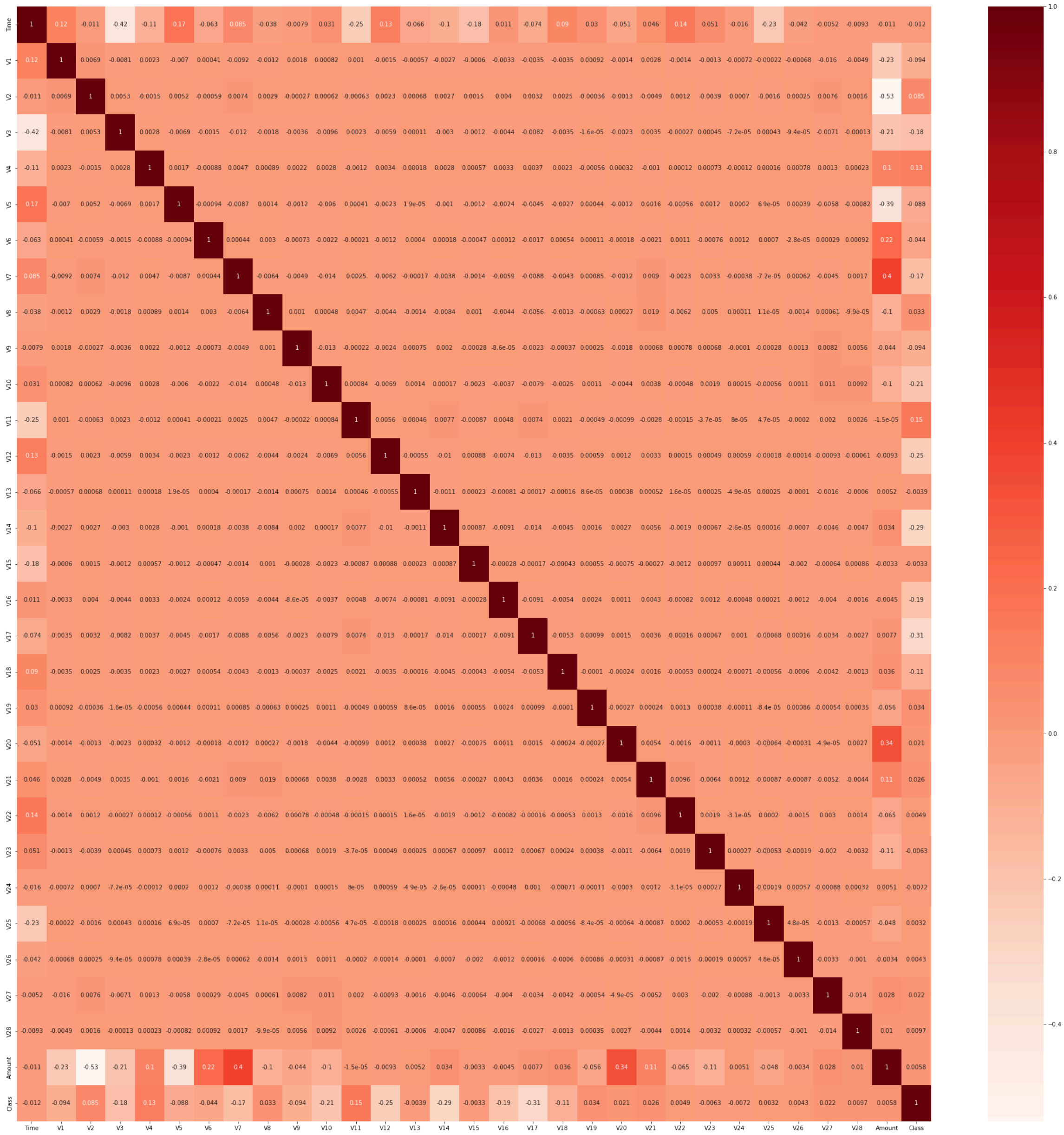


In [13]:

```
plt.figure(figsize=(35,35))
sns.heatmap(df.corr(),annot=True, cmap='Reds')
```

Out[13]:

```
<AxesSubplot:>
```



In [14]:

```
x= df.iloc[:,0:-1]
y = df.iloc[:,-1]
```

In [15]:

```
x.shape
```

Out[15]:

```
(283726, 30)
```

In [16]:

```
y.shape
```

Out[16]:

```
(283726,)
```

In [17]:

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

In [18]:

```
x_train, x_test, y_train,y_test =  train_test_split(x,y, train_size=0.75
```

In [19]:
```
x_train.shape
```
Out[19]:
```
(212794, 30)
```

In [20]:
```
y_train.shape
```
Out[20]:
```
(212794,)
```

In [21]:
```
x_test.shape
```
Out[21]:
```
(70932, 30)
```

In [22]:
```
y_test.shape
```
Out[22]:
```
(70932,)
```

In [23]:
```
from sklearn.linear_model import LogisticRegression
```

In [24]:
```
logit = LogisticRegression()
```

In [25]:
```
logit.fit(x_train,y_train)
```
```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logist
ic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html (http
s://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression (https://scikit-learn.org/stable/modules/linear_model.html#l
ogistic-regression)
  n_iter_i = _check_optimize_result(
```
Out[25]:
```
LogisticRegression()
```

In [44]:
```
y_predict_train= logit.predict(x_train)
y_predict_test= logit.predict(x_test)
```

In [45]:
```
from sklearn.metrics import confusion_matrix, classification_report, acc
```

In [49]:
```
cm_tr = confusion_matrix(y_train, y_predict_train)
cm_tr
```
Out[49]:
```
array([[212348,     89],
       [   137,    220]], dtype=int64)
```

In [50]:
```
cm_tst = confusion_matrix(y_test, y_predict_test)
cm_tst
```
Out[50]:
```
array([[70784,    32],
       [   37,    79]], dtype=int64)
```

In [55]:
```
cl_rep = classification_report(y_train, y_predict_train)
cl_rep
```

Out[55]:
```
'              precision    recall  f1-score   support\n\n           0
1.00      1.00      1.00    212437\n           1       0.71      0.62
0.66       357\n\n    accuracy                           1.00    212794
\n   macro avg       0.86      0.81      0.83    212794\nweighted avg
1.00      1.00      1.00    212794\n'
```

In [62]:
```
print(f"Classification Report\n\n{cl_rep}")
```

```
Classification Report

              precision    recall  f1-score   support

           0       1.00      1.00      1.00    212437
           1       0.71      0.62      0.66       357

    accuracy                           1.00    212794
   macro avg       0.86      0.81      0.83    212794
weighted avg       1.00      1.00      1.00    212794
```

In [57]:
```
cl_rep_tst = classification_report(y_test,y_predict_test)
```

In [61]:
```
print(f"Classification Report\n\n{cl_rep_tst}")
```

```
Classification Report

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     70816
           1       0.71      0.68      0.70       116

    accuracy                           1.00     70932
   macro avg       0.86      0.84      0.85     70932
weighted avg       1.00      1.00      1.00     70932
```

In [73]:
```
ac_train  = accuracy_score(y_train,y_predict_train)
ac_train
```

Out[73]:
```
0.9989379399795107
```

In [74]:
```
ac_train_prct = ac_train *100
ac_train_prct
```

Out[74]:
```
99.89379399795108
```

In [80]:
```
print(f"Accuracy Score : {ac_train_prct:.2f}%")
```

```
Accuracy Score : 99.89%
```

In [81]:
```
ac_test  = accuracy_score(y_test,y_predict_test)
ac_test
```

Out[81]:
```
0.9990272373540856
```

In [82]:

```
ac_tst_prct = ac_test *100
ac_tst_prct
```

Out[82]:

```
99.90272373540856
```

In [83]:

```
print(f"Accuracy Score : {ac_tst_prct:.2f}%")
```

```
Accuracy Score : 99.90%
```

In [86]:

```
import joblib
joblib.dump(logit, "model_filename.pkl")
# load the model using loaded_model = joblib.load("model_filename.pkl")
```

Out[86]:

```
['model_filename.pkl']
```

In [ ]: