CREDIT CARD FRAUD DETECTION

importing the libraries and loading the dataset

In [1]:
```
import pandas as pd import
numpy as np
import matplotlib.pyplot as plt import seaborn as
sns
data=pd.read_csv("creditcard.csv") data.columns
```

Out[1]:
```
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')
```

In [2]:
```
data
```

Out[2]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009
...	...	...	...	...	...	...	...	...	...	...	...	
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261

284807 rows × 31 columns

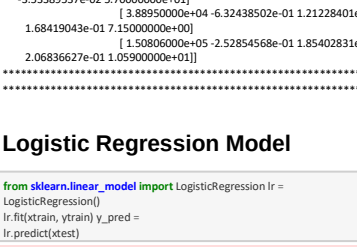
where 1 for fraudulent transactions, 0 otherwise

In [3]:
```
data['Class'].value_counts()
```

Out[3]:
```
0    284315
1       492
Name: Class, dtype: int64
```

drawing countplot for the class column in the dataset

In [4]:
```
plt.close() sns.countplot(x='Class',data=data,palette='hls') plt.show()
```



visualizing the higher dimension data using Pricipal Component Analysis(PCA).

In [5]:
```
from sklearn import decomposition pca =
decomposition.PCA()
```

In [6]:
```
sample_data=data.drop("Class",axis=1) pca.n_components = 2
pca_data = pca.fit_transform(sample_data) pca_data.shape
```

Out[6]: (284807, 2)

In [7]:
```
Class=data['Class']
Class.shape
```

Out[7]: (284807,)

In [8]:
```
pca_data = np.vstack((pca_data.T, Class)).T
```

In [9]:
```
pca_df = pd.DataFrame(data=pca_data, columns=("1st_principal", "2nd_principal", "Class")) pca_df
sns.FacetGrid(pca_df, hue="Class", size=4).map(plt.scatter, '1st_principal', '2nd_principal'
).add_legend()
plt.show()
```

```
C:\Users\TUMMULURI PARVATHI\Anaconda3\lib\site-packages\seaborn\axisgrid.py:230: UserWarning: The `size` paramter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```


defining x and y values

In [10]:
```
x=data.iloc[:,:-1].values
y=data.iloc[:,30].values
print(x.shape) print(y.shape)
```

```
(284807, 30)
(284807,)
```

spliting the dataset into xtrain,xtest,ytrain,ytest

In [11]:
```
from sklearn.model_selection import train_test_split xtrain,xtest,ytrain,ytest=
train_test_split(x,y,test_size=0.3,random_state=0) print(xtrain.shape)#taking 70% of xdata for training the model
print(xtest.shape)#taking the 30% of xdata for testing the model print(ytrain.shape)#taking 70% of ydata for
training the model print(ytest.shape)#taking 70% of ydata for training the model
```

```
(199364, 30)
(85443, 30)
(199364,)
(85443,)
```

printing the xtrain xtest ytrain ytest

In [12]:
```
print(xtrain) print("**********************************************************************************************
**") print(xtest)
print("**********************************************************************************************
**") print(ytrain)
print("**********************************************************************************************
**") print(ytest)
```

```
[[ 1.13921000e+05 -1.32066260e-01  1.07043551e-01 ..... 3.06394657e-01
   7.48169663e-02 4.78900000e+01]
 [ 1.35319000e+05 2.12599441e+00 1.42074055e-02 ... -6.82673794e-02
  -5.76775883e-02 1.98000000e+00]
 [ 1.25498000e+05 -8.66939700e-02  1.66240223e-01 ..... 1.20157065e+00
  -1.13993068e+00 1.70100000e+02]
 ...
 [ 7.66160000e+04 1.51260229e+00 -9.49435315e-01 ... -1.41537086e-03 3.66494418e-03
   3.49000000e+01]
 [ 9.72530000e+04 1.79886333e+00 -1.69979073e+00 ... -3.36155803e-02
  -3.24705125e-02 1.71310000e+02]
 [ 7.48870000e+04 -5.89399721e-01 7.47828393e-01 ... 8.65917860e-02 1.18083774e-01
   1.59900000e+01]]
**********************************************************************************************
[[ 1.25821000e+05 -3.23333572e-01 1.05745525e+00 ...                1.08494430e-01
   1.61139167e-01 4.00000000e+01]
                              [ 1.57235000e+05 -3.49718405e-01 9.32618570e-01 ...  7.68300272e-02
   1.75561960e-01 1.98000000e+00]
                             [ 1.52471000e+05 -1.61471082e+00 -2.40656973e+00 ...  2.86285101e-01
   4.37321868e-01 9.60000000e+01]
 ...
                              [ 5.59110000e+04 -1.21539007e+00 1.37955591e+00 ...  2.34355933e-01
  -3.53389537e-02 5.70000000e+01]
                              [ 3.88950000e+04 -6.32438502e-01 1.21228401e+00 ...  3.45269266e-01
   1.68419043e-01 7.15000000e+00]
                              [ 1.50806000e+05 -2.52854568e-01 1.85402831e+00 ...  3.60196291e-01
   2.06836627e-01 1.05900000e+01]]
********************************************************************************************** [0 0 0 ... 0 0 0]
********************************************************************************************** [0 0 0 ... 0 0 0]
```

Logistic Regression Model

In [13]:
```
from sklearn.linear_model import LogisticRegression lr =
LogisticRegression()
lr.fit(xtrain, ytrain) y_pred =
lr.predict(xtest)
```

```
C:\Users\TUMMULURI PARVATHI\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: Futur eWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

In [14]:
```
from sklearn.metrics import confusion_matrix c1 =
confusion_matrix(ytest, y_pred) print(c1)
```

```
[[85285    11]
 [   70    77]]
```

Accuracy

In [15]:
```
accuracy=(c1[0][0]+c1[1][1])/(c1[0][0]+c1[1][1]+c1[0][1]+c1[1][0])
accuracy*100
```

Out[15]: 99.90519995786666

Random forest Model

In [16]:
```
from sklearn.ensemble import RandomForestClassifier r=
RandomForestClassifier()
r.fit(xtrain,ytrain)
```

```
C:\Users\TUMMULURI PARVATHI\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarni ng: The default value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[16]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=None, verbose=0, warm_start=False)
```

In [17]:
```
ypredr=r.predict(xtest) ypredr
```

Out[17]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

confusion_matrix for the Random forest

In [18]:
```
from sklearn.metrics import confusion_matrix
c2=confusion_matrix(ytest,ypredr)
c2
```

Out[18]:
```
array([[85289,                      7],
                    [ 37, 110]], dtype=int64)
```

Accuracy

In [19]:
```
accuracy1=(c2[0][0]+c2[1][1])/(c2[0][0]+c2[1][1]+c2[0][1]+c2[1][0])
accuracy1*100
```

Out[19]: 99.94850368081644

In [ ]: