In [167]:
```
import pandas as pd
df=pd.read_csv('creditcard.csv')
df.head()
```

Out[167]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0

5 rows × 31 columns

In [168]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set_style("whitegrid")
```

In [169]:
```
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 [170]:
```
df.describe()
```

Out[170]:

	Time	V1	V2	V3	V4	V5
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05
mean	94813.859575	1.165980e-15	3.416908e-16	-1.373150e-15	2.086869e-15	9.604066e-16
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02

	Time	V1	V2	V3	V4	V5
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01

8 rows × 31 columns

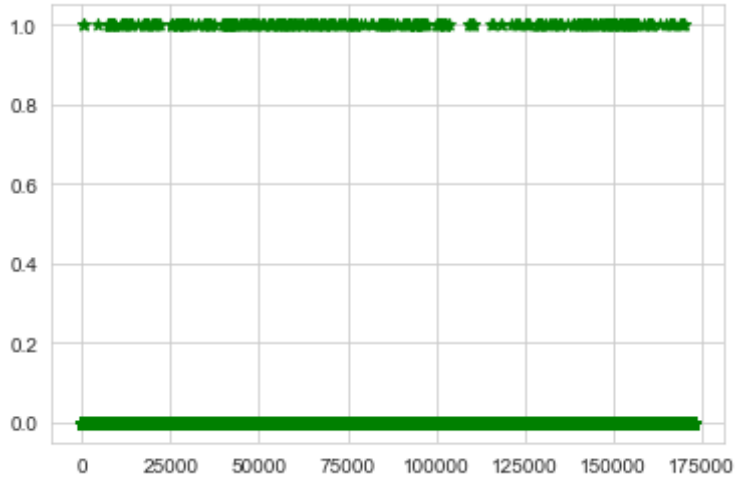
In [142]:
```
df.isnull().sum().sum()
```

Out[142]:
```
0
```

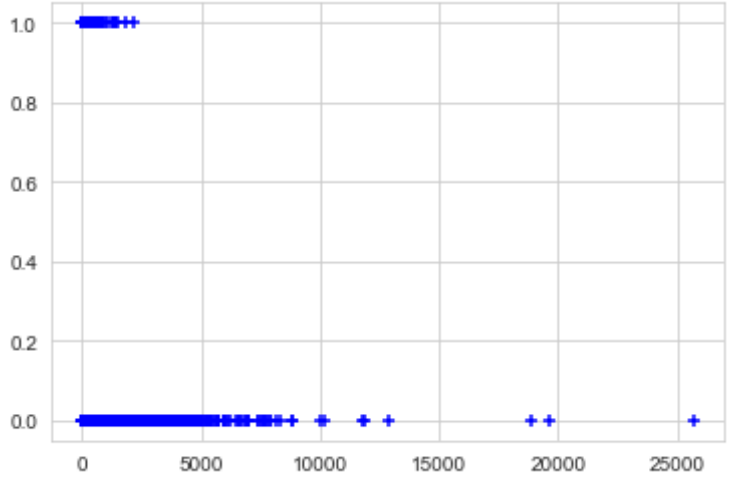
In [143]:
```
df.Class.value_counts()
```

Out[143]:
```
0    284315
1       492
Name: Class, dtype: int64
```

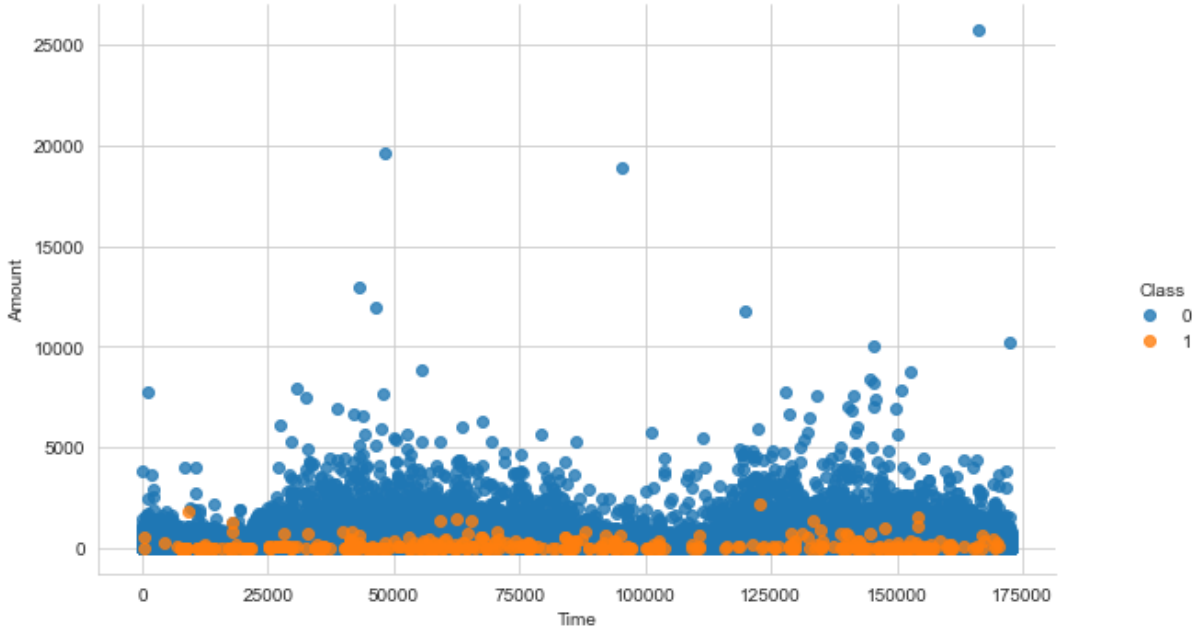
In [144]:
```
y=df.Class
plt.scatter(df.Time,y,c='g',marker='*')

plt.show()
```



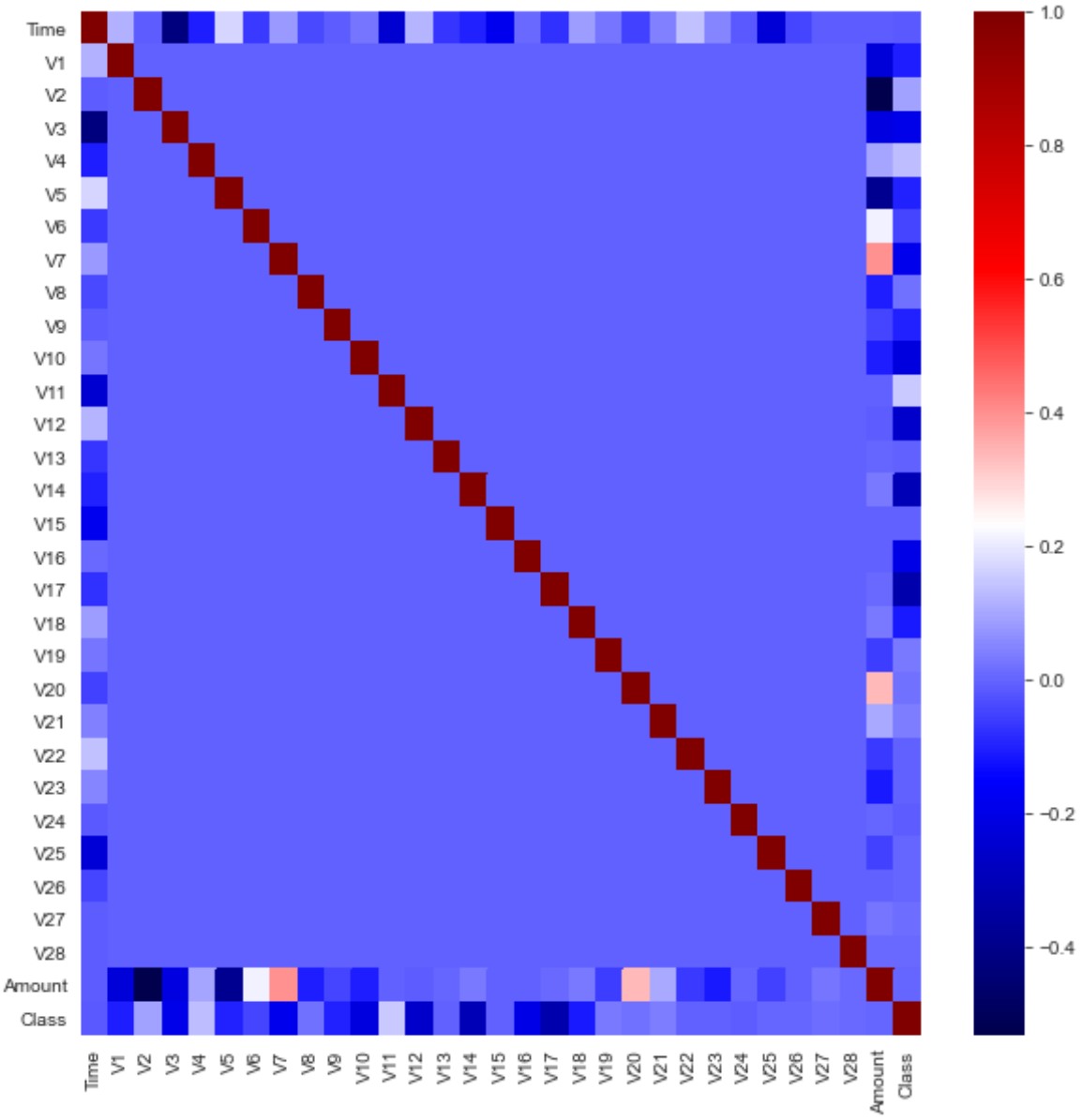
In [145]:
```
plt.scatter(df.Amount,y,c='b',marker='+')
plt.show()
```



In [146]:
```
sns.lmplot('Time', 'Amount', df, hue='Class', fit_reg=False)
fig = plt.gcf()
fig.set_size_inches(10, 5)
plt.show()
```



In [147]:
```
plt.figure(figsize=(10,10))
sns.heatmap(data=df.corr(),cmap="seismic")
plt.show();
```



In [148]:
```
y
```

Out[148]:
```
0         0
```

```
1         0
2         0
3         0
4         0
         ..
284802    0
284803    0
284804    0
284805    0
284806    0
Name: Class, Length: 284807, dtype: int64
```

In [149]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
scalar=StandardScaler()
```

In [150]:
```
inputs=df.drop(['Class'],axis='columns')
inputs
```

Out[150]:

	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
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0
...	...	...	...	...	...	...	...	...	...
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0

284807 rows × 30 columns

In [151]:
```
output=df.Class
output
```

Out[151]:
```
0         0
1         0
2         0
3         0
4         0
         ..
284802    0
284803    0
284804    0
284805    0
284806    0
Name: Class, Length: 284807, dtype: int64
```

In [152]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, precision
_score, recall_score, f1_score
def print_score(label, prediction, train=True):
    if train:
        print("train result :------------------------------\n")
        print(f"accuracy score : {accuracy_score(label,prediction)*100:
.2f}%")
        print(f"\tprecision score: {precision_score(label,prediction)*1
00:.2f}%")
        print(f"\t\trecall score: {recall_score(label,prediction)*100:.
2f}%")
        print(f"\t\t\tf1 score: {f1_score(label,prediction)*100:.2f}%")
        print(f"confusion matrix: \n {confusion_matrix(y_train,predicti
on)}\n")
    elif train==False:
        print("test result :------------------------------\n")
        print(f"accuracy score : {accuracy_score(label,prediction)*100:
.2f}%")
        print(f"\tprecision score: {precision_score(label,prediction)*1
00:.2f}%")
```

```
        print(f"\t\trecall score: {recall_score(label,prediction)*100:.
2f}%")
        print(f"\t\t\tf1 score: {f1_score(label,prediction)*100:.2f}%")
        print(f"confusion matrix: \n {confusion_matrix(label,prediction
)}\n")
```

In [153]:
```
x_train_v,x_test,y_train_v,y_test=train_test_split(inputs,output,test_s
ize=0.3,random_state=42)
x_train,x_validate,y_train,y_validate=train_test_split(x_train_v,y_trai
n_v,test_size=0.3,random_state=42)
x_train=scalar.fit_transform(x_train)
x_test=scalar.transform(x_test)
x_validate=scalar.transform(x_validate)
```

In [154]:
```
print(f"x_train: {x_train.shape}, y_train : {y_train.shape}\n")
print(f"x_validate :{x_validate.shape},y_validate : {y_validate.shape}
\n")
print(f" x_test: {x_test.shape}, y_test: {y_test.shape}\n")
```

```
x_train: (139554, 30), y_train : (139554,)

x_validate :(59810, 30),y_validate : (59810,)

 x_test: (85443, 30), y_test: (85443,)
```

In [155]:
```
from sklearn.svm import SVC
model=SVC()
model.fit(x_train,y_train)
```

Out[155]:
```
SVC()
```

In [156]:
```
model.fit(x_validate,y_validate)
```

Out[156]:
```
SVC()
```

In [157]:
```
model.score(x_test,y_test)
```

Out[157]:
```
0.9990402958697612
```

In [158]:
```
y_train_pred=model.predict(x_train)
y_test_pred=model.predict(x_test)
y_valid_pred=model.predict(x_validate)
print_score(y_train,y_train_pred.round(),train=True)
print_score(y_test,y_test_pred.round(),train=False)
```

```
train result :------------------------------

accuracy score : 99.90%
        precision score: 95.20%
                recall score: 46.12%
                        f1 score: 62.14%
confusion matrix:
 [[139290      6]
 [   139    119]]

test result :------------------------------

accuracy score : 99.90%
        precision score: 88.57%
                recall score: 45.59%
                        f1 score: 60.19%
confusion matrix:
 [[85299     8]
 [   74    62]]
```

In [210]:
```
from sklearn.utils import resample
minority_class=df[df.Class==1]
majority_class=df[df.Class==0]
majority_down=resample(majority_class,replace=False, n_samples=10*(mino
rity_class.shape[0]),random_state=42)
df_1=pd.concat([majority_down,minority_class])
df_1.Class.value_counts()
```

Out[210]:
```
0    4920
```

```
1     492
Name: Class, dtype: int64
```

In [211]:
```
df_1.shape
```

Out[211]:
```
(5412, 31)
```

In [212]:
```
x1=df_1.drop(['Class'],axis='columns')
y1=df_1.Class
x1_train,x1_test,y1_train,y1_test=train_test_split(x1,y1,test_size=0.3,
random_state=42)
x1_train=scalar.fit_transform(x1_train)
x1_test=scalar.transform(x1_test)
```

In [213]:
```
print(f"x1_train: {x1_train.shape}, y1_train : {y1_train.shape}\n")
print(f" x1_test: {x1_test.shape}, y1_test: {y1_test.shape}\n")
```

```
x1_train: (3788, 30), y1_train : (3788,)

 x1_test: (1624, 30), y1_test: (1624,)
```

In [214]:
```
#supoort vector machine
model2=SVC()
model2.fit(x1_train,y1_train)
```

Out[214]:
```
SVC()
```

In [215]:
```
model2.score(x1_test,y1_test)
```

Out[215]:
```
0.9852216748768473
```

In [216]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, precision
_score, recall_score, f1_score
def print_score1(label, prediction, train=True):
    if train:
        print("train result :------------------------------\n")
```

```
        print(f"accuracy score : {accuracy_score(label,prediction)*100:
.2f}%")
        print(f"\tprecision score: {precision_score(label,prediction)*1
00:.2f}%")
        print(f"\t\trecall score: {recall_score(label,prediction)*100:.
2f}%")
        print(f"\t\t\tf1 score: {f1_score(label,prediction)*100:.2f}%")
        print(f"confusion matrix: \n {confusion_matrix(y1_train,predict
ion)}\n")
    elif train==False:
        print("test result :-------------------------------\n")
        print(f"accuracy score : {accuracy_score(label,prediction)*100:
.2f}%")
        print(f"\tprecision score: {precision_score(label,prediction)*1
00:.2f}%")
        print(f"\t\trecall score: {recall_score(label,prediction)*100:.
2f}%")
        print(f"\t\t\tf1 score: {f1_score(label,prediction)*100:.2f}%")
        print(f"confusion matrix: \n {confusion_matrix(label,prediction
)}\n")
```

In [217]:
```
y1_train_pred=model2.predict(x1_train)
y1_test_pred=model2.predict(x1_test)
print_score1(y1_train,y1_train_pred.round(),train=True)
print_score1(y1_test,y1_test_pred.round(),train=False)
```

```
train result :-------------------------------

accuracy score : 98.68%
        precision score: 98.68%
                recall score: 86.67%
                        f1 score: 92.28%
confusion matrix:
 [[3439    4]
 [  46  299]]

test result :-------------------------------
```

```
accuracy score : 98.52%
        precision score: 97.67%
                recall score: 85.71%
                        f1 score: 91.30%
confusion matrix:
 [[1474    3]
 [  21  126]]
```

In [218]:
```
#logistic regression
from sklearn.linear_model import LogisticRegression
```

In [219]:
```
m1=LogisticRegression()
m1.fit(x1_train,y1_train)
```

Out[219]:
```
LogisticRegression()
```

In [220]:
```
m1.score(x1_test,y1_test)
```

Out[220]:
```
0.9839901477832512
```

In [221]:
```
y2_train_pred=m1.predict(x1_train)
y2_test_pred=m1.predict(x1_test)
print_score1(y1_train,y2_train_pred.round(),train=True)
print_score1(y1_test,y2_test_pred.round(),train=False)
```

```
train result :------------------------------

accuracy score : 98.47%
        precision score: 97.36%
                recall score: 85.51%
                        f1 score: 91.05%
confusion matrix:
 [[3435    8]
 [  50  295]]

test result :------------------------------
```

```
accuracy score : 98.40%
        precision score: 95.49%
                recall score: 86.39%
                        f1 score: 90.71%
confusion matrix:
 [[1471    6]
 [  20  127]]
```

In [222]:
```
#decision tree
from sklearn import tree
```

In [223]:
```
m2=tree.DecisionTreeClassifier()
```

In [224]:
```
m2.fit(x1_train,y1_train)
```

Out[224]:
```
DecisionTreeClassifier()
```

In [225]:
```
m2.score(x1_test,y1_test)
```

Out[225]:
```
0.9698275862068966
```

In [226]:
```
y3_train_pred=m2.predict(x1_train)
y3_test_pred=m2.predict(x1_test)
print_score1(y1_train,y3_train_pred.round(),train=True)
print_score1(y1_test,y3_test_pred.round(),train=False)
```

```
train result :------------------------------

accuracy score : 100.00%
        precision score: 100.00%
                recall score: 100.00%
                        f1 score: 100.00%
confusion matrix:
 [[3443    0]
 [   0  345]]

test result :------------------------------
```

```
accuracy score : 96.98%
        precision score: 80.25%
                recall score: 88.44%
                        f1 score: 84.14%
confusion matrix:
 [[1445   32]
 [  17  130]]
```

In [227]:
```
#random forest
from sklearn.ensemble import RandomForestClassifier
```

In [228]:
```
m3=RandomForestClassifier()
```

In [229]:
```
m3.fit(x1_train,y1_train)
```

Out[229]:
```
RandomForestClassifier()
```

In [230]:
```
m3.score(x1_test,y1_test)
```

Out[230]:
```
0.9858374384236454
```

In [231]:
```
y4_train_pred=m3.predict(x1_train)
y4_test_pred=m3.predict(x1_test)
print_score1(y1_train,y4_train_pred.round(),train=True)
print_score1(y1_test,y4_test_pred.round(),train=False)
```

```
train result :------------------------------

accuracy score : 100.00%
        precision score: 100.00%
                recall score: 100.00%
                        f1 score: 100.00%
confusion matrix:
 [[3443    0]
 [   0  345]]
```

```
test result :------------------------------

accuracy score : 98.58%
        precision score: 97.69%
                recall score: 86.39%
                        f1 score: 91.70%
confusion matrix:
 [[1474    3]
 [  20  127]]
```

In [232]:
```
#k nearest neighbours
from sklearn.neighbors import KNeighborsClassifier
```

In [233]:
```
m4=KNeighborsClassifier()
```

In [234]:
```
m4.fit(x1_train,y1_train)
```

Out[234]:
```
KNeighborsClassifier()
```

In [235]:
```
m4.score(x1_test,y1_test)
```

Out[235]:
```
0.9833743842364532
```

In [236]:
```
y5_train_pred=m4.predict(x1_train)
y5_test_pred=m4.predict(x1_test)
print_score1(y1_train,y5_train_pred.round(),train=True)
print_score1(y1_test,y5_test_pred.round(),train=False)
```

```
train result :------------------------------

accuracy score : 98.52%
        precision score: 98.65%
                recall score: 84.93%
                        f1 score: 91.28%
confusion matrix:
 [[3439    4]
 [  52  293]]
```

```
test result :------------------------------

accuracy score : 98.34%
        precision score: 95.45%
                recall score: 85.71%
                        f1 score: 90.32%
confusion matrix:
 [[1471    6]
 [  21  126]]
```

In [237]:
```
#naive bayes
from sklearn.naive_bayes import GaussianNB
```

In [238]:
```
m5=GaussianNB()
```

In [239]:
```
m5.fit(x1_train,y1_train)
```

Out[239]:
```
GaussianNB()
```

In [240]:
```
m5.score(x1_test,y1_test)
```

Out[240]:
```
0.9624384236453202
```

In [241]:
```
y6_train_pred=m5.predict(x1_train)
y6_test_pred=m5.predict(x1_test)
print_score1(y1_train,y6_train_pred.round(),train=True)
print_score1(y1_test,y6_test_pred.round(),train=False)
```

```
train result :------------------------------

accuracy score : 95.56%
        precision score: 72.52%
                recall score: 82.61%
                        f1 score: 77.24%
confusion matrix:
 [[3335  108]
```

```
 [  60  285]]

test result :--------------------------------

accuracy score : 96.24%
        precision score: 75.90%
                recall score: 85.71%
                        f1 score: 80.51%
confusion matrix:
 [[1437   40]
 [  21  126]]
```

In [ ]: