In [431]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.datasets import load_wine
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import make_classification
from sklearn.metrics import accuracy_score
import plotly.express as px
```

In [432]:
```
dataset = pd.read_csv('modified_dataset.csv')
```

In [433]:
```
dataset.head()
```

Out[433]:

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	street	...	long	city_pop	
0	0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	F	561 Perry Cove	...	-81.1781	3495	Psy cc
1	1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	F	43039 Riley Greens Suite 393	...	-118.2105	149	ec need
2	2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanchez	M	594 White Dale Suite 530	...	-112.2620	4154	cor
3	3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy	White	M	9443 Cynthia Court Apt. 038	...	-112.1138	1939	Paten
4	4	2019-01-01 00:03:06	375534208663984	fraud_Keeling-Crist	misc_pos	41.96	Tyler	Garcia	M	408 Bradley Rest	...	-79.4629	99	n psych

5 rows × 24 columns

In [434]:
```
dataset.isnull().sum()
```

Out[434]:
```
Unnamed: 0                   0
trans_date_trans_time        0
cc_num                       0
merchant                     0
category                     0
amt                       1318
first                        0
last                         0
gender                       0
street                       0
city                         0
state                        0
zip                          0
lat                          0
long                         0
city_pop                     0
job                          0
dob                          0
trans_num                    0
unix_time                    0
merch_lat                    0
merch_long                   0
is_fraud                     0
dummy                  1295348
dtype: int64
```

In [435]:
```
dataset = dataset.drop(['dummy'],axis=1)
```

In [436]:
```
dataset.isnull().sum()
```

Out[436]:
```
Unnamed: 0                 0
trans_date_trans_time      0
cc_num                     0
merchant                   0
category                   0
amt                     1318
first                      0
last                       0
gender                     0
street                     0
city                       0
state                      0
zip                        0
lat                        0
long                       0
city_pop                   0
job                        0
dob                        0
trans_num                  0
unix_time                  0
merch_lat                  0
merch_long                 0
is_fraud                   0
dtype: int64
```

In [437]:
```
dataset = dataset[dataset['amt'].notnull()]
```

In [438]:
```
dataset.isnull().sum()
```

Out[438]:
```
Unnamed: 0              0
trans_date_trans_time   0
cc_num                  0
merchant                0
category                0
amt                     0
first                   0
last                    0
gender                  0
street                  0
city                    0
state                   0
zip                     0
lat                     0
long                    0
city_pop                0
job                     0
dob                     0
trans_num               0
unix_time               0
merch_lat               0
merch_long              0
is_fraud                0
dtype: int64
```

In [439]:
```
copy_dataset = dataset
```

In [440]:
```
dataset.is_fraud.sum()
```

Out[440]:
```
7499
```

In [441]:
```
dataset.keys() # merchant, category, first, last, gender, street, city, state, job,
```

Out[441]:
```
Index(['Unnamed: 0', 'trans_date_trans_time', 'cc_num', 'merchant', 'category',
       'amt', 'first', 'last', 'gender', 'street', 'city', 'state', 'zip',
       'lat', 'long', 'city_pop', 'job', 'dob', 'trans_num', 'unix_time',
       'merch_lat', 'merch_long', 'is_fraud'],
      dtype='object')
```

In [442]:
```
# merchant encoding
enc = LabelEncoder()
# Apply the encoding to the "Accessible" column
dataset['merchant_enc'] = enc.fit_transform(dataset['merchant'])
# Compare the two columns
dataset[['merchant','merchant_enc']]
```

Out[442]:

	merchant	merchant_enc
0	fraud_Rippin, Kub and Mann	514
1	fraud_Heller, Gutmann and Zieme	241
2	fraud_Lind-Buckridge	390
3	fraud_Kutch, Hermiston and Farrell	360
4	fraud_Keeling-Crist	297
...	...	...
1296670	fraud_Reichel Inc	499
1296671	fraud_Abernathy and Sons	2
1296672	fraud_Stiedemann Ltd	599
1296673	fraud_Reinger, Weissnat and Strosin	509
1296674	fraud_Langosh, Wintheiser and Hyatt	370

1295357 rows × 2 columns

In [443]:
```
# category encoding
enc = LabelEncoder()
# Apply the encoding to the "Accessible" column
dataset['category_enc'] = enc.fit_transform(dataset['category'])
# Compare the two columns
dataset[['category','category_enc']]
```

Out[443]:

	category	category_enc
0	misc_net	8
1	grocery_pos	4
2	entertainment	0
3	gas_transport	2
4	misc_pos	9
...	...	...
1296670	entertainment	0
1296671	food_dining	1
1296672	food_dining	1
1296673	food_dining	1
1296674	food_dining	1

1295357 rows × 2 columns

In [444]:
```
# first encoding
enc = LabelEncoder()
# Apply the encoding to the "Accessible" column
dataset['first_enc'] = enc.fit_transform(dataset['first'])
# Compare the two columns
dataset[['first','first_enc']]
```

Out[444]:

	first	first_enc
0	Jennifer	162
1	Stephanie	309
2	Edward	115
3	Jeremy	163
4	Tyler	336
...	...	...
1296670	Erik	121
1296671	Jeffrey	160
1296672	Christopher	74
1296673	Joseph	179
1296674	Jeffrey	160

1295357 rows × 2 columns

In [445]:
```
# last encoding
enc = LabelEncoder()
# Apply the encoding to the "Accessible" column
dataset['last_enc'] = enc.fit_transform(dataset['last'])
# Compare the two columns
dataset[['last','last_enc']]
```

Out[445]:

	last	last_enc
0	Banks	18
1	Gill	157
2	Sanchez	381
3	White	463
4	Garcia	149
...	...	...
1296670	Patterson	332
1296671	White	463
1296672	Castaneda	67
1296673	Murray	304
1296674	Smith	404

1295357 rows × 2 columns

In [446]:
```
# gender encoding
enc = LabelEncoder()
# Apply the encoding to the "Accessible" column
dataset['gender_enc'] = enc.fit_transform(dataset['gender'])
# Compare the two columns
dataset[['gender','gender_enc']]
```

Out[446]:

	gender	gender_enc
0	F	0
1	F	0
2	M	1
3	M	1
4	M	1
...	...	...
1296670	M	1
1296671	M	1
1296672	M	1
1296673	M	1
1296674	M	1

1295357 rows × 2 columns

In [447]:
```
# street encoding
enc = LabelEncoder()
# Apply the encoding to the "Accessible" column
dataset['street_enc'] = enc.fit_transform(dataset['street'])
# Compare the two columns
dataset[['street','street_enc']]
```

Out[447]:

	street	street_enc
0	561 Perry Cove	568
1	43039 Riley Greens Suite 393	435
2	594 White Dale Suite 530	602
3	9443 Cynthia Court Apt. 038	930
4	408 Bradley Rest	418
...	...	...
1296670	162 Jessica Row Apt. 072	154
1296671	8617 Holmes Terrace Suite 651	856
1296672	1632 Cohen Drive Suite 639	158
1296673	42933 Ryan Underpass	433
1296674	135 Joseph Mountains	127

1295357 rows × 2 columns

In [448]:
```
# city encoding
enc = LabelEncoder()
# Apply the encoding to the "Accessible" column
dataset['city_enc'] = enc.fit_transform(dataset['city'])
# Compare the two columns
dataset[['city','city_enc']]
```

Out[448]:

	city	city_enc
0	Moravian Falls	526
1	Orient	612
2	Malad City	468
3	Boulder	84
4	Doe Hill	216
...	...	...
1296670	Hatch	330
1296671	Tuscarora	813
1296672	High Rolls Mountain Park	346
1296673	Manderson	471
1296674	Sula	782

1295357 rows × 2 columns

In [449]:
```
# state encoding
enc = LabelEncoder()
# Apply the encoding to the "Accessible" column
dataset['state_enc'] = enc.fit_transform(dataset['state'])
# Compare the two columns
dataset[['state','state_enc']]
```

Out[449]:

	state	state_enc
0	NC	27
1	WA	47
2	ID	13
3	MT	26
4	VA	45
...	...	...
1296670	UT	44
1296671	MD	20
1296672	NM	32
1296673	SD	41
1296674	MT	26

1295357 rows × 2 columns

In [450]:
```
# job encoding
enc = LabelEncoder()
# Apply the encoding to the "Accessible" column
dataset['job_enc'] = enc.fit_transform(dataset['job'])
# Compare the two columns
dataset[['job','job_enc']]
```

Out[450]:

	job	job_enc
0	Psychologist, counselling	370
1	Special educational needs teacher	428
2	Nature conservation officer	307
3	Patent attorney	328
4	Dance movement psychotherapist	116
...	...	...
1296670	Geoscientist	215
1296671	Production assistant, television	360
1296672	Naval architect	308
1296673	Volunteer coordinator	485
1296674	Therapist, horticultural	467

1295357 rows × 2 columns

In [451]:
```
dataset['trans_date_trans_time'] = pd.to_datetime(dataset['trans_date_trans_time'])

# Extract the month using the .dt accessor
dataset['Month'] = dataset['trans_date_trans_time'].dt.month
```

In [452]:
```
dataset.keys()
```

Out[452]:
```
Index(['Unnamed: 0', 'trans_date_trans_time', 'cc_num', 'merchant', 'category',
       'amt', 'first', 'last', 'gender', 'street', 'city', 'state', 'zip',
       'lat', 'long', 'city_pop', 'job', 'dob', 'trans_num', 'unix_time',
       'merch_lat', 'merch_long', 'is_fraud', 'merchant_enc', 'category_enc',
       'first_enc', 'last_enc', 'gender_enc', 'street_enc', 'city_enc',
       'state_enc', 'job_enc', 'Month'],
      dtype='object')
```

In [453]:
```
encoded_dataset = dataset[['cc_num', 'amt', 'zip', 'lat', 'long', 'city_pop', 'dob', 'trans_num', 'unix_time',
       'merch_lat', 'merch_long', 'is_fraud', 'merchant_enc', 'category_enc',
       'first_enc', 'last_enc', 'gender_enc', 'street_enc', 'city_enc', 'state_enc', 'job_enc', 'Month']]
```

In [454]:
```
dcorr = encoded_dataset.corr()
dcorr
```

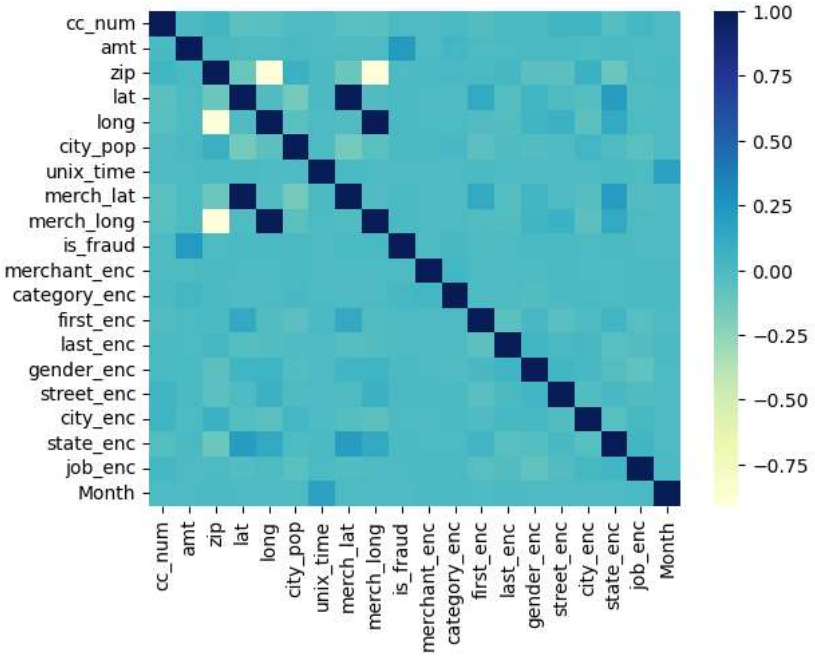
```
C:\Users\AYON ROY\AppData\Local\Temp\ipykernel_20236\176387756.py:1: FutureWarning: The default value of numeric_only in DataFr
ame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeri
c_only to silence this warning.
  dcorr = encoded_dataset.corr()
```

Out[454]:

	cc_num	amt	zip	lat	long	city_pop	unix_time	merch_lat	merch_long	is_fraud	merchant_enc	category_enc	first_
cc_num	1.000000	0.001760	0.041460	-0.059325	-0.048295	-0.008953	0.000346	-0.058997	-0.048269	-0.000960	0.000108	0.001223	-0.02
amt	0.001760	1.000000	0.001889	-0.001959	-0.000224	0.005832	-0.000273	-0.001907	-0.000186	0.219254	-0.002632	0.030867	-0.00
zip	0.041460	0.001889	1.000000	-0.114307	-0.909738	0.078513	0.000651	-0.113583	-0.908930	-0.002103	0.001087	0.002374	0.00
lat	-0.059325	-0.001959	-0.114307	1.000000	-0.015482	-0.155769	0.000676	0.993592	-0.015458	0.001878	-0.002252	-0.008659	0.13
long	-0.048295	-0.000224	-0.909738	-0.015482	1.000000	-0.052752	-0.000622	-0.015398	0.999119	0.001668	-0.000674	-0.000768	-0.01
city_pop	-0.008953	0.005832	0.078513	-0.155769	-0.052752	1.000000	-0.001750	-0.154823	-0.052724	0.002161	0.001877	0.009398	-0.06
unix_time	0.000346	-0.000273	0.000651	0.000676	-0.000622	-0.001750	1.000000	0.000602	-0.000616	-0.005047	-0.001021	0.000132	0.00
merch_lat	-0.058997	-0.001907	-0.113583	0.993592	-0.015398	-0.154823	0.000602	1.000000	-0.015377	0.001720	-0.002251	-0.008518	0.13
merch_long	-0.048269	-0.000186	-0.908930	-0.015458	0.999119	-0.052724	-0.000616	-0.015377	1.000000	0.001668	-0.000650	-0.000739	-0.01
is_fraud	-0.000960	0.219254	-0.002103	0.001878	0.001668	0.002161	-0.005047	0.001720	0.001668	1.000000	-0.000547	0.020186	-0.00
merchant_enc	0.000108	-0.002632	0.001087	-0.002252	-0.000674	0.001877	-0.001021	-0.002251	-0.000650	-0.000547	1.000000	0.032273	-0.00
category_enc	0.001223	0.030867	0.002374	-0.008659	-0.000768	0.009398	0.000132	-0.008518	-0.000739	0.020186	0.032273	1.000000	-0.00
first_enc	-0.027143	-0.003574	0.004773	0.131868	-0.016483	-0.066601	0.000459	0.130961	-0.016497	-0.003235	-0.001565	-0.001938	1.00
last_enc	0.006888	-0.004699	0.028266	-0.034894	-0.025539	-0.012011	0.000109	-0.034529	-0.025537	-0.000109	-0.000465	-0.004291	-0.05
gender_enc	0.001068	0.001038	-0.065941	0.042916	0.050410	-0.028633	-0.000942	0.042629	0.050357	0.007594	-0.000893	-0.028266	0.01
street_enc	0.046512	0.001365	-0.053842	-0.012706	0.071317	-0.012491	-0.001108	-0.012573	0.071275	0.001459	0.000686	0.002212	-0.04
city_enc	0.049229	-0.000545	0.074672	-0.032059	-0.066631	0.034903	-0.000497	-0.031844	-0.066646	-0.002091	-0.000305	0.000799	-0.00
state_enc	-0.036396	0.001689	-0.116260	0.213309	0.132455	-0.012173	0.001078	0.211882	0.132335	0.001690	-0.000422	0.000180	0.04
job_enc	0.029196	-0.000610	-0.005664	-0.027592	-0.002756	-0.054990	0.000046	-0.027408	-0.002754	-0.000079	0.001358	0.003537	-0.03
Month	-0.000300	-0.001730	0.000854	-0.001112	-0.001207	0.000145	0.184862	-0.001162	-0.001192	-0.012403	0.000461	0.000756	0.00

In [455]:
```
sns.heatmap(dcorr, cmap = 'YlGnBu')
```

Out[455]: <Axes: >



In [456]:
```
# Removing zip, long, merch_long Colomns
encoded_dataset2 = dataset[['cc_num', 'amt', 'lat', 'city_pop', 'dob', 'trans_num', 'unix_time',
       'merch_lat', 'is_fraud', 'merchant_enc', 'category_enc',
       'first_enc', 'last_enc', 'gender_enc', 'street_enc', 'city_enc', 'state_enc', 'job_enc', 'Month']]
```

In [457]:
```
dcorr2 = encoded_dataset2.corr()
dcorr2
```

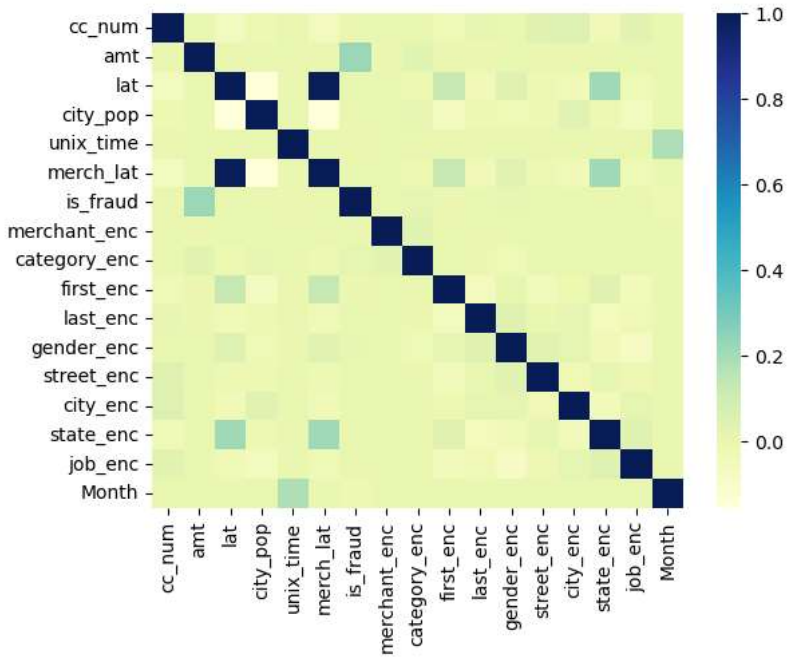
```
C:\Users\AYON ROY\AppData\Local\Temp\ipykernel_20236\1041761430.py:1: FutureWarning: The default value of numeric_only in DataF
rame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numer
ic_only to silence this warning.
  dcorr2 = encoded_dataset2.corr()
```

Out[457]:

	cc_num	amt	lat	city_pop	unix_time	merch_lat	is_fraud	merchant_enc	category_enc	first_enc	last_enc	gender_enc	stree
cc_num	1.000000	0.001760	-0.059325	-0.008953	0.000346	-0.058997	-0.000960	0.000108	0.001223	-0.027143	0.006888	0.001068	0.0
amt	0.001760	1.000000	-0.001959	0.005832	-0.000273	-0.001907	0.219254	-0.002632	0.030867	-0.003574	-0.004699	0.001038	0.0(
lat	-0.059325	-0.001959	1.000000	-0.155769	0.000676	0.993592	0.001878	-0.002252	-0.008659	0.131868	-0.034894	0.042916	-0.0
city_pop	-0.008953	0.005832	-0.155769	1.000000	-0.001750	-0.154823	0.002161	0.001877	0.009398	-0.066601	-0.012011	-0.028633	-0.0
unix_time	0.000346	-0.000273	0.000676	-0.001750	1.000000	0.000602	-0.005047	-0.001021	0.000132	0.000459	0.000109	-0.000942	-0.0(
merch_lat	-0.058997	-0.001907	0.993592	-0.154823	0.000602	1.000000	0.001720	-0.002251	-0.008518	0.130961	-0.034529	0.042629	-0.0
is_fraud	-0.000960	0.219254	0.001878	0.002161	-0.005047	0.001720	1.000000	-0.000547	0.020186	-0.003235	-0.000109	0.007594	0.0(
merchant_enc	0.000108	-0.002632	-0.002252	0.001877	-0.001021	-0.002251	-0.000547	1.000000	0.032273	-0.001565	-0.000465	-0.000893	0.0(
category_enc	0.001223	0.030867	-0.008659	0.009398	0.000132	-0.008518	0.020186	0.032273	1.000000	-0.001938	-0.004291	-0.028266	0.0(
first_enc	-0.027143	-0.003574	0.131868	-0.066601	0.000459	0.130961	-0.003235	-0.001565	-0.001938	1.000000	-0.056969	0.015140	-0.0
last_enc	0.006888	-0.004699	-0.034894	-0.012011	0.000109	-0.034529	-0.000109	-0.000465	-0.004291	-0.056969	1.000000	0.043740	0.0(
gender_enc	0.001068	0.001038	0.042916	-0.028633	-0.000942	0.042629	0.007594	-0.000893	-0.028266	0.015140	0.043740	1.000000	0.0
street_enc	0.046512	0.001365	-0.012706	-0.012491	-0.001108	-0.012573	0.001459	0.000686	0.002212	-0.045453	0.005608	0.042715	1.0(
city_enc	0.049229	-0.000545	-0.032059	0.034903	-0.000497	-0.031844	-0.002091	-0.000305	0.000799	-0.009210	0.017538	0.016063	-0.0
state_enc	-0.036396	0.001689	0.213309	-0.012173	0.001078	0.211882	0.001690	-0.000422	0.000180	0.043086	-0.048166	-0.031812	0.0
job_enc	0.029196	-0.000610	-0.027592	-0.054990	0.000046	-0.027408	-0.000079	0.001358	0.003537	-0.039655	-0.024602	-0.079918	-0.0
Month	-0.000300	-0.001730	-0.001112	0.000145	0.184862	-0.001162	-0.012403	0.000461	0.000756	0.000152	0.001089	-0.000253	0.0(

In [458]:
```
sns.heatmap(dcorr2, cmap = 'YlGnBu')
```

Out[458]: <Axes: >



In [459]:
```
# Removing
encoded_dataset3 = dataset[['cc_num', 'amt', 'lat', 'city_pop', 'unix_time', 'is_fraud', 'merchant_enc',
                            'category_enc', 'first_enc', 'last_enc', 'gender_enc', 'street_enc', 'city_enc', 'state_enc',
                            'job_enc', 'Month']]
```

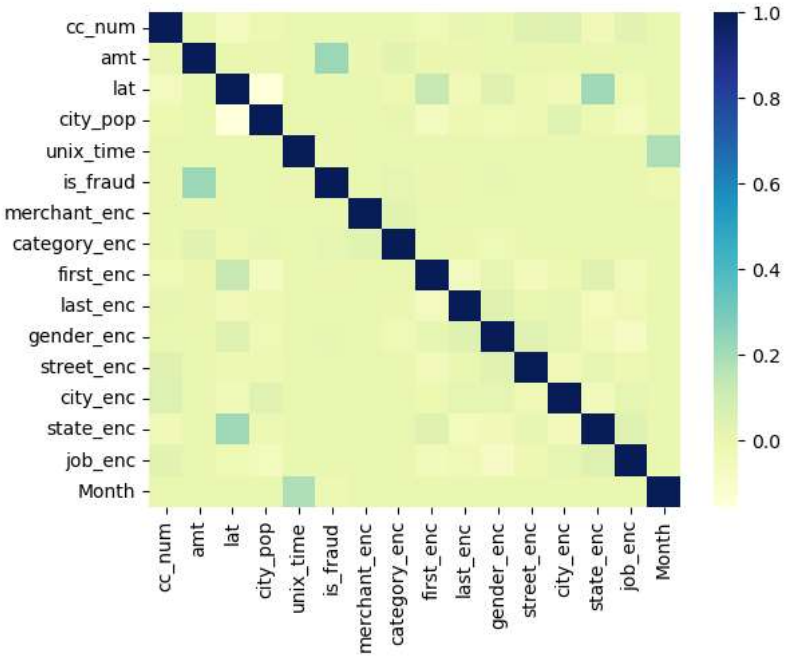
In [460]:
```
dcorr3 = encoded_dataset3.corr()
dcorr3
```

Out[460]:

	cc_num	amt	lat	city_pop	unix_time	is_fraud	merchant_enc	category_enc	first_enc	last_enc	gender_enc	street_enc	city
cc_num	1.000000	0.001760	-0.059325	-0.008953	0.000346	-0.000960	0.000108	0.001223	-0.027143	0.006888	0.001068	0.046512	0.04
amt	0.001760	1.000000	-0.001959	0.005832	-0.000273	0.219254	-0.002632	0.030867	-0.003574	-0.004699	0.001038	0.001365	-0.00
lat	-0.059325	-0.001959	1.000000	-0.155769	0.000676	0.001878	-0.002252	-0.008659	0.131868	-0.034894	0.042916	-0.012706	-0.03
city_pop	-0.008953	0.005832	-0.155769	1.000000	-0.001750	0.002161	0.001877	0.009398	-0.066601	-0.012011	-0.028633	-0.012491	0.03
unix_time	0.000346	-0.000273	0.000676	-0.001750	1.000000	-0.005047	-0.001021	0.000132	0.000459	0.000109	-0.000942	-0.001108	-0.00
is_fraud	-0.000960	0.219254	0.001878	0.002161	-0.005047	1.000000	-0.000547	0.020186	-0.003235	-0.000109	0.007594	0.001459	-0.00
merchant_enc	0.000108	-0.002632	-0.002252	0.001877	-0.001021	-0.000547	1.000000	0.032273	-0.001565	-0.000465	-0.000893	0.000686	-0.00
category_enc	0.001223	0.030867	-0.008659	0.009398	0.000132	0.020186	0.032273	1.000000	-0.001938	-0.004291	-0.028266	0.002212	0.00
first_enc	-0.027143	-0.003574	0.131868	-0.066601	0.000459	-0.003235	-0.001565	-0.001938	1.000000	-0.056969	0.015140	-0.045453	-0.00
last_enc	0.006888	-0.004699	-0.034894	-0.012011	0.000109	-0.000109	-0.000465	-0.004291	-0.056969	1.000000	0.043740	0.005608	0.01
gender_enc	0.001068	0.001038	0.042916	-0.028633	-0.000942	0.007594	-0.000893	-0.028266	0.015140	0.043740	1.000000	0.042715	0.01
street_enc	0.046512	0.001365	-0.012706	-0.012491	-0.001108	0.001459	0.000686	0.002212	-0.045453	0.005608	0.042715	1.000000	-0.02
city_enc	0.049229	-0.000545	-0.032059	0.034903	-0.000497	-0.002091	-0.000305	0.000799	-0.009210	0.017538	0.016063	-0.024799	1.00
state_enc	-0.036396	0.001689	0.213309	-0.012173	0.001078	0.001690	-0.000422	0.000180	0.043086	-0.048166	-0.031812	0.014045	-0.04
job_enc	0.029196	-0.000610	-0.027592	-0.054990	0.000046	-0.000079	0.001358	0.003537	-0.039655	-0.024602	-0.079918	-0.020161	0.02
Month	-0.000300	-0.001730	-0.001112	0.000145	0.184862	-0.012403	0.000461	0.000756	0.000152	0.001089	-0.000253	0.000345	-0.00

In [461]:
```
sns.heatmap(dcorr3, cmap = 'YlGnBu')
```

Out[461]: <Axes: >



In [462]:
```
X_train, X_test, y_train, y_test = train_test_split(dataset[['cc_num', 'amt', 'lat', 'city_pop', 'unix_time', 'merchant_enc'
                            'category_enc', 'first_enc', 'last_enc', 'gender_enc', 'street_enc', 'city_enc', 'state_enc',
                            'job_enc', 'Month']], dataset['is_fraud'], test_size=0.25, random_state=0)
```

In [463]:
```
X_train
```

Out[463]:

	cc_num	amt	lat	city_pop	unix_time	merchant_enc	category_enc	first_enc	last_enc	gender_enc	street_enc	city_enc	state
382036	3589255887819806	58.87	39.6991	3766	1340540881	7	6	93	288	1	624	321	
1216676	371284424780634	74.26	33.5466	254282	1369383580	55	2	163	294	1	578	464	
495435	4225990116481262579	60.16	20.0827	4878	1344058875	442	4	46	400	1	270	355	
21990	377834944388609	4.64	36.4539	6434	1326511498	22	12	243	210	1	540	633	
715132	4607072969078276	158.37	35.8985	1493	1351810337	15	6	44	337	0	29	179	
...	...	...	...	...	...	...	...	...	...	...	...	...	...
360151	5456776410929288	50.46	41.0472	139	1339879741	343	10	322	227	0	657	587	
152466	4561546772499	57.44	34.9889	7268	1332495075	241	4	91	44	1	709	527	
964394	4025612008285111	38.44	41.1762	1274	1359037745	397	10	207	226	0	225	175	
118070	60423098130	34.55	36.9412	83	1331160074	619	0	157	166	1	871	20	
306023	348379466427142	91.02	28.1384	139650	1338204209	472	4	243	376	1	746	497	

971517 rows × 15 columns

In [464]:
```
X_test
```

Out[464]:

	cc_num	amt	lat	city_pop	unix_time	merchant_enc	category_enc	first_enc	last_enc	gender_enc	street_enc	city_enc	state
657194	4760122961662	55.14	41.1116	1202	1349561368	482	7	281	220	1	466	26	
1158031	4715741951931168360	9.87	36.6966	471	1367102697	612	7	281	110	1	469	110	
843069	4306586894261180	55.36	40.1186	4049	1355474616	602	4	276	446	0	861	828	
1010096	3536918668124647	1.82	37.3304	973849	1361327204	22	12	246	222	0	457	727	
932907	347073228412010	76.90	44.6873	110	1357417263	210	0	286	217	1	415	17	
...	...	...	...	...	...	...	...	...	...	...	...	...	...
776376	676281772837	9.41	30.8836	3430	1354137853	253	5	6	31	0	220	873	
547416	213174467670432	62.63	39.4015	663	1345650758	178	0	152	31	1	701	131	
1245809	503848303379	2.83	26.5290	3776	1370198126	545	12	137	444	1	271	720	
69446	4586810168620942	193.56	32.5486	2408	1328914358	618	9	246	168	0	695	237	
169852	676292294938	18.87	41.2336	104	1333143780	188	6	220	129	0	593	751	

323840 rows × 15 columns

In [465]:
```
y_train
```

Out[465]:
```
382036     0
1216676    0
495435     0
21990      0
715132     0
          ..
360151     0
152466     0
964394     0
118070     0
306023     0
Name: is_fraud, Length: 971517, dtype: int64
```

In [466]:
```
y_test.shape
```

Out[466]:
```
(323840,)
```

In [ ]:
```

```

In [467]:
```
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train_MinMaxscaled = scaler.transform(X_train)
X_train_MinMaxscaled
```

Out[467]:
```
array([[7.18939599e-04, 1.99910874e-03, 4.21547073e-01, ...,
        4.00000000e-01, 7.05882353e-01, 4.54545455e-01],
       [7.43586249e-05, 2.53075353e-03, 2.89706468e-01, ...,
        8.60000000e-01, 2.71805274e-01, 3.63636364e-01],
       [8.46493767e-01, 2.04367156e-03, 1.19144049e-03, ...,
        2.20000000e-01, 6.87626775e-01, 6.36363636e-01],
       ...,
       [8.06344616e-04, 1.29335807e-03, 4.53199532e-01, ...,
        7.60000000e-01, 5.65922921e-01, 0.00000000e+00],
       [1.38030187e-12, 1.15897872e-03, 3.62448624e-01, ...,
        7.20000000e-01, 9.12778905e-02, 1.81818182e-01],
       [6.97706102e-05, 3.10972471e-03, 1.73815310e-01, ...,
        1.80000000e-01, 2.41379310e-01, 3.63636364e-01]])
```

In [468]:
```
scaler = MinMaxScaler()
scaler.fit(X_test)
X_test_MinMaxscaled = scaler.transform(X_test)
X_test_MinMaxscaled
```

Out[468]:
```
array([[9.41382355e-07, 2.03969993e-03, 4.51815232e-01, ...,
        7.00000000e-01, 2.31237323e-01, 8.18181818e-01],
       [9.44594299e-01, 3.34173225e-04, 3.57207144e-01, ...,
        7.20000000e-01, 8.84381339e-01, 2.72727273e-01],
       [8.62625745e-04, 2.04798833e-03, 4.30536448e-01, ...,
        3.00000000e-01, 2.63691684e-01, 1.00000000e+00],
       ...,
       [8.88223825e-08, 6.89444195e-05, 1.39327822e-01, ...,
        1.80000000e-01, 8.72210953e-01, 4.54545455e-01],
       [9.18756321e-04, 7.25461061e-03, 2.68320540e-01, ...,
        8.00000000e-01, 8.19472617e-01, 9.09090909e-02],
       [1.23364055e-07, 6.73244140e-04, 4.54429544e-01, ...,
        7.60000000e-01, 3.95537525e-01, 1.81818182e-01]])
```

In [469]:
```
X_test_MinMaxscaled.shape
```

Out[469]:
```
(323840, 15)
```

In [470]:
```
y_train
```

Out[470]:
```
382036     0
1216676    0
495435     0
21990      0
715132     0
          ..
360151     0
152466     0
964394     0
118070     0
306023     0
Name: is_fraud, Length: 971517, dtype: int64
```

In [471]:
```
scaler2 = StandardScaler()
scaler2.fit(X_train)
X_train_Standardscaled = scaler.transform(X_train)
X_train_Standardscaled
```

Out[471]:
```
array([[7.18939599e-04, 2.18022599e-03, 4.21547073e-01, ...,
        4.00000000e-01, 7.05882353e-01, 4.54545455e-01],
       [7.43586249e-05, 2.76003725e-03, 2.89706468e-01, ...,
        8.60000000e-01, 2.71805274e-01, 3.63636364e-01],
       [8.46493767e-01, 2.22882615e-03, 1.19144049e-03, ...,
        2.20000000e-01, 6.87626775e-01, 6.36363636e-01],
       ...,
       [8.06344616e-04, 1.41053501e-03, 4.53199532e-01, ...,
        7.60000000e-01, 5.65922921e-01, 0.00000000e+00],
       [1.38030187e-12, 1.26398102e-03, 3.62448624e-01, ...,
        7.20000000e-01, 9.12778905e-02, 1.81818182e-01],
       [6.97706102e-05, 3.39146265e-03, 1.73815310e-01, ...,
        1.80000000e-01, 2.41379310e-01, 3.63636364e-01]])
```

KNN

In [472]:
```
knn=KNeighborsClassifier()
knn.fit(X_train_MinMaxscaled, y_train)
```

Out[472]:
```
KNeighborsClassifier
KNeighborsClassifier()
```

In [473]:
```
y_pred = knn.predict(X_test_MinMaxscaled)
```

In [474]:
```
count = 0
for i in range( len(y_test) ):
    if y_pred[i] != y_test.iloc[i]:
        count = count + 1
error = count/len(y_pred)
print( "Error = %f " % (error*100) + '%' )
knn_accuracy_withoutUndersample = (1-error)
print( "Accuracy = %f " % (knn_accuracy_withoutUndersample*100) + '%' )
```

```
Error = 0.554904 %
Accuracy = 99.445096 %
```

In [475]:
```
# Visual

# pd.plotting.scatter_matrix(dataset[:7500], c=y_train, figsize=(100, 100),
#                            marker='o', hist_kwds={'bins': 20}, s=20,
#                            alpha=.8)
```

In [476]:
```
knn_cm = confusion_matrix(y_test, y_pred)
print(knn_cm)
```

```
[[321931    107]
 [  1690    112]]
```

In [477]:
```
sns.heatmap(knn_cm, annot=True, cmap='Pastel1_r', xticklabels=['Fraud' ,'Not Fraud'],yticklabels=['Fraud' ,'Not Fraud'])
# Column Actual val, Row Predicted val
```

Out[477]:
```
<Axes: >
```

Logistic Regression

In [478]:
```
logistic_regression = LogisticRegression()
logistic_regression.fit(X_train_MinMaxscaled,y_train)
```

Out[478]:
```
LogisticRegression
LogisticRegression()
```

In [479]:
```
y_pred = logistic_regression.predict(X_test_MinMaxscaled)
```

In [481]:
```
count = 0
for i in range( len(y_test) ):
    if y_pred[i] != y_test.iloc[i]:
        count = count + 1
error = count/len(y_pred)
print( "Error = %f " % (error*100) + '%' )
lr_accuracy_withoutundersample = (1-error)
print( "Accuracy = %f " % (lr_accuracy_withoutundersample*100) + '%' )
```

```
Error = 0.578990 %
Accuracy = 99.421010 %
```

In [482]:
```
logistic_regression_cm = confusion_matrix(y_test, y_pred)
print(logistic_regression_cm)
```

```
[[321965     73]
 [  1802      0]]
```

In [483]:
```
sns.heatmap(logistic_regression_cm, annot=True, cmap='Pastel1_r', xticklabels=['Fraud' ,'Not Fraud'],yticklabels=['Fraud' ,'
```

Out[483]: <Axes: >

Naive Bayes

In [484]:
```
naive_bayes = GaussianNB()
naive_bayes.fit(X_train_MinMaxscaled, y_train)
```

Out[484]:
```
GaussianNB
GaussianNB()
```

In [485]:
```
y_pred = naive_bayes.predict(X_test_MinMaxscaled)
```

In [486]:
```
count = 0
for i in range(len(y_test)):
    if y_pred[i] != y_test.iloc[i]:
        count = count + 1
error = count/len(y_pred)
print( "Error = %f " % (error*100) + '%' )
nb_accuracy_withoutundersample = (1-error)
print( "Accuracy = %f " % (nb_accuracy_withoutundersample*100) + '%' )
```

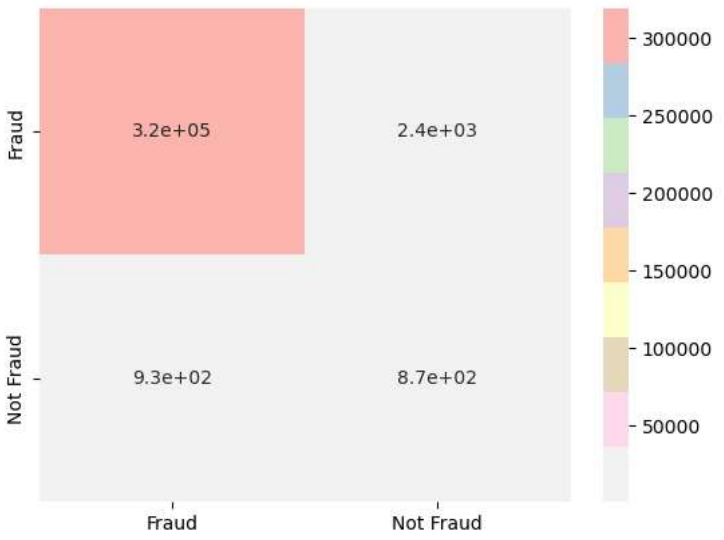
```
Error = 1.042181 %
Accuracy = 98.957819 %
```

In [487]:
```
logistic_regression_cm = confusion_matrix(y_test, y_pred)
print(logistic_regression_cm)
```

```
[[319591   2447]
 [   928    874]]
```

In [488]:
```
sns.heatmap(logistic_regression_cm, annot=True, cmap='Pastel1_r', xticklabels=['Fraud' ,'Not Fraud'],yticklabels=['Fraud' ,'
```

Out[488]: <Axes: >



Under Sampling

In [489]:
```
updated_dataset = dataset[['cc_num', 'amt', 'lat', 'city_pop', 'unix_time', 'merchant_enc','category_enc', 'first_enc', 'las
fraud = updated_dataset[dataset.is_fraud==1]
not_fraud =  updated_dataset[updated_dataset.is_fraud==0]
sampled_not_fraud = not_fraud.sample(n=fraud.shape[0])
updated_dataset = pd.concat([fraud, sampled_not_fraud])
updated_dataset.shape
```

Out[489]: (14998, 16)

In [525]:
```
X_train, X_test, y_train, y_test = train_test_split(updated_dataset[['cc_num', 'amt', 'lat', 'city_pop', 'unix_time', 'merch
                            'category_enc', 'first_enc', 'last_enc', 'gender_enc', 'street_enc', 'city_enc', 'state_enc',
                            'job_enc', 'Month']], updated_dataset['is_fraud'], test_size=0.25, random_state=0)
```

In [526]:
```
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train_MinMaxscaled = scaler.transform(X_train)
X_train_MinMaxscaled
```

Out[526]:
```
array([[8.15338324e-04, 7.55016011e-02, 1.91879776e-01, ...,
        8.60000000e-01, 3.42799189e-01, 1.00000000e+00],
       [7.10587727e-04, 2.77466641e-03, 5.24921678e-01, ...,
        4.40000000e-01, 2.55578093e-01, 3.63636364e-01],
       [6.81296087e-05, 7.70004787e-02, 2.01707017e-01, ...,
        8.60000000e-01, 4.62474645e-01, 0.00000000e+00],
       ...,
       [8.84620840e-08, 5.04505928e-03, 4.66391092e-01, ...,
        1.20000000e-01, 4.46247465e-01, 9.09090909e-02],
       [9.60930359e-01, 2.55622534e-05, 4.04290471e-01, ...,
        3.00000000e-01, 2.29208925e-01, 1.81818182e-01],
       [7.14120290e-04, 2.14102463e-01, 4.38312955e-01, ...,
        4.80000000e-01, 5.17241379e-01, 5.45454545e-01]])
```

In [509]:
```
scaler = MinMaxScaler()
scaler.fit(X_test)
X_test_MinMaxscaled = scaler.transform(X_test)
X_test_MinMaxscaled
```

Out[509]:
```
array([[1.31207631e-03, 3.34252419e-03, 1.48090052e-01, ...,
        1.80000000e-01, 7.72819473e-01, 2.72727273e-01],
       [6.10939053e-06, 2.84829882e-03, 4.38561528e-01, ...,
        4.80000000e-01, 9.61460446e-01, 0.00000000e+00],
       [1.20410887e-03, 3.19425658e-02, 4.30461447e-01, ...,
        7.60000000e-01, 6.04462475e-01, 5.45454545e-01],
       ...,
       [8.33052115e-04, 1.76490480e-02, 4.37226515e-01, ...,
        7.60000000e-01, 8.33671400e-01, 9.09090909e-01],
       [4.26930358e-05, 2.88688309e-02, 3.86528151e-01, ...,
        3.20000000e-01, 3.69168357e-01, 1.81818182e-01],
       [7.01450116e-04, 2.94801096e-03, 2.95468669e-01, ...,
        6.00000000e-02, 2.27180527e-01, 1.81818182e-01]])
```

In [527]:
```
# KNN Model
knn=KNeighborsClassifier()
knn.fit(X_train_MinMaxscaled, y_train)
```

Out[527]:
```
KNeighborsClassifier

KNeighborsClassifier()
```

In [528]:
```
y_pred = knn.predict(X_test_MinMaxscaled)
```

In [529]:
```
count = 0
for i in range( len(y_test) ):
    if y_pred[i] != y_test.iloc[i]:
        count = count + 1
error = count/len(y_pred)
print( "Error = %f " % (error*100) + '%' )
knn_accuracy = (1-error)
print( "Accuracy = %f " % (knn_accuracy*100) + '%' )
```

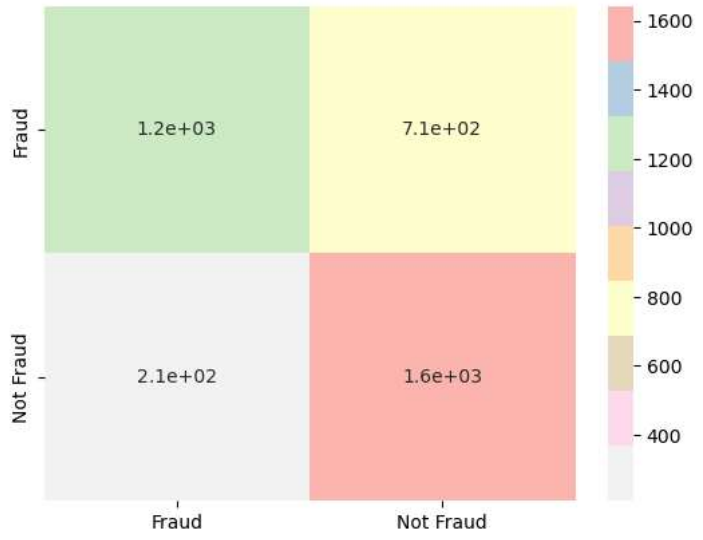
```
Error = 24.426667 %
Accuracy = 75.573333 %
```

In [530]:
```
knn_cm = confusion_matrix(y_test, y_pred)
print(knn_cm)
```

```
[[1192  707]
 [ 209 1642]]
```

In [531]:
```
sns.heatmap(knn_cm, annot=True, cmap='Pastel1_r', xticklabels=['Fraud' ,'Not Fraud'],yticklabels=['Fraud' ,'Not Fraud'])
```

Out[531]:
```
<Axes: >
```



In [532]:
```
#Logistic_regression

logistic_regression = LogisticRegression()
logistic_regression.fit(X_train_MinMaxscaled,y_train)
```

Out[532]:
```
LogisticRegression
LogisticRegression()
```

In [533]:
```
y_pred = logistic_regression.predict(X_test_MinMaxscaled)
```

In [534]:
```
count = 0
for i in range( len(y_test) ):
    if y_pred[i] != y_test.iloc[i]:
        count = count + 1
error = count/len(y_pred)
print( "Error = %f " % (error*100) + '%' )
lr_accuracy = (1-error)
print( "Accuracy = %f " % (lr_accuracy*100) + '%' )
```

```
Error = 19.440000 %
Accuracy = 80.560000 %
```

In [535]:
```
logistic_regression_cm = confusion_matrix(y_test, y_pred)
print(logistic_regression_cm)
```

```
[[1574  325]
 [ 404 1447]]
```

In [536]:
```
atmap(logistic_regression_cm, annot=True, cmap='Pastel1_r', xticklabels=['Fraud' ,'Not Fraud'],yticklabels=['Fraud' ,'Not Fraud'])
```

Out[536]: <Axes: >

In [537]:
```
# Naive Bayes
naive_bayes = GaussianNB()
naive_bayes.fit(X_train_MinMaxscaled, y_train)
```

Out[537]:
```
GaussianNB
GaussianNB()
```

In [538]:
```
y_pred = naive_bayes.predict(X_test_MinMaxscaled)
```

In [539]:
```
count = 0
for i in range(len(y_test)):
    if y_pred[i] != y_test.iloc[i]:
        count = count + 1
error = count/len(y_pred)
print( "Error = %f " % (error*100) + '%' )
nb_accuracy = (1-error)
print( "Accuracy = %f " % (nb_accuracy*100) + '%' )
```

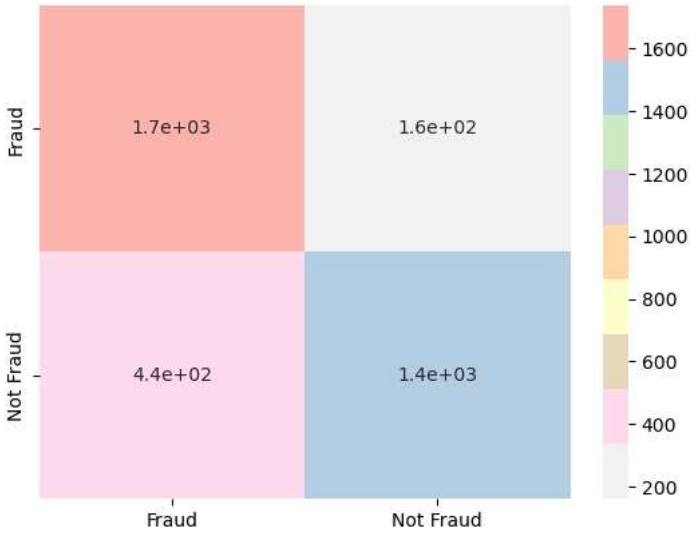
```
Error = 16.000000 %
Accuracy = 84.000000 %
```

In [540]:
```
logistic_regression_cm = confusion_matrix(y_test, y_pred)
print(logistic_regression_cm)
```

```
[[1738  161]
 [ 439 1412]]
```

In [541]:
```
atmap(logistic_regression_cm, annot=True, cmap='Pastel1_r', xticklabels=['Fraud' ,'Not Fraud'],yticklabels=['Fraud' ,'Not Fraud'])
```

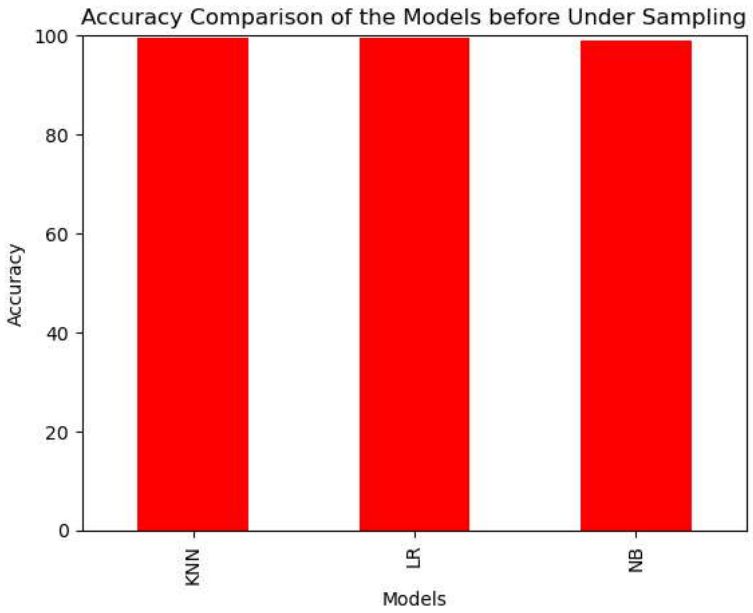
Out[541]: <Axes: >



Comparison

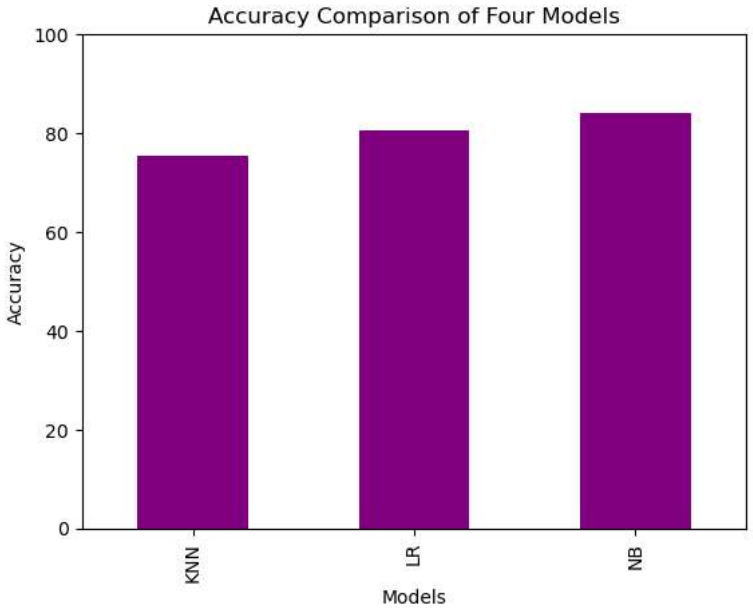
In [553]:
```
# WithOut Undersampling

accuracy = {'Model': ['KNN', 'LR', 'NB'],
    'Accuracy': [knn_accuracy_withoutUndersample*100, lr_accuracy_withoutundersample*100, nb_accuracy_withoutundersample*100]}
df = pd.DataFrame(accuracy)
ax = df.plot(kind='bar', x='Model', y='Accuracy', color='red', legend=False)
ax.set_ylim([0, 100])
ax.set_title('Accuracy Comparison of the Models before Under Sampling')
ax.set_xlabel('Models')
ax.set_ylabel('Accuracy')
plt.show()
```



In [554]:

```
accuracy2 = {'Model': ['KNN', 'LR', 'NB'],
        'Accuracy': [knn_accuracy*100, lr_accuracy*100, nb_accuracy*100]}
df = pd.DataFrame(accuracy2)
ax = df.plot(kind='bar', x='Model', y='Accuracy', color='purple', legend=False)
ax.set_ylim([0, 100])
ax.set_title('Accuracy Comparison of Four Models')
ax.set_xlabel('Models')
ax.set_ylabel('Accuracy')
plt.show()
```



In [555]:

```
dict1 = {'KNN':knn_accuracy_withoutUndersample*100, 'LR':lr_accuracy_withoutundersample*100, 'NB':nb_accuracy_withoutundersa

dict2 ={'KNN':knn_accuracy*100, 'LR':lr_accuracy*100, 'NB':nb_accuracy*100}

keys = list(dict1.keys())
values_dict1 = list(dict1.values())
values_dict2 = list(dict2.values())

plt.bar(keys, values_dict1, label='Dictionary 1', color='red')
plt.bar(keys, values_dict2, label='Dictionary 2', color='purple')

plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Comparison of Three Models (Before & After Under Sampling)')
plt.legend()

plt.show()
```