In [262…

```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import *
```

In [263…

```
master_card = pd.read_csv(r"D:\PG-DAI\MachineLearning\Assessment\3 Credit Card Fraud Analysis\creditcard.csv")
```

In [264…

```
master_card.head()
```

Out[264…

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.

5 rows × 31 columns

In [265…

```
master_card.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 [266…

```
master_card.tail(10)
```

Out[266...

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	
284797	172782.0	-0.241923	0.712247	0.399806	-0.463406	0.244531	-1.343668	0.929369	-0.206210	0.106234	...	-0.228876	-0.514376	0.279598	0
284798	172782.0	0.219529	0.881246	-0.635891	0.960928	-0.152971	-1.014307	0.427126	0.121340	-0.285670	...	0.099936	0.337120	0.251791	0
284799	172783.0	-1.775135	-0.004235	1.189786	0.331096	1.196063	5.519980	-1.518185	2.080825	1.159498	...	0.103302	0.654850	-0.348929	0
284800	172784.0	2.039560	-0.175233	-1.196825	0.234580	-0.008713	-0.726571	0.017050	-0.118228	0.435402	...	-0.268048	-0.717211	0.297930	-0
284801	172785.0	0.120316	0.931005	-0.546012	-0.745097	1.130314	-0.235973	0.812722	0.115093	-0.204064	...	-0.314205	-0.808520	0.050343	0
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864	1.014480	-0
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.012463	-1
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.037501	0
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.163298	0
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.376777	0

10 rows × 31 columns

In [267...

```python
del master_card['Time']
```

In [268...

```python
master_card.head(25)
```

Out[268...

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V21	V22	V23	V2
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	...	-0.018307	0.277838	-0.110474	0.0669
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	...	-0.225775	-0.638672	0.101288	-0.3398
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	...	0.247998	0.771679	0.909412	-0.6892
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	...	-0.108300	0.005274	-0.190321	-1.1755
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	...	-0.009431	0.798278	-0.137458	0.1412
5	-0.425966	0.960523	1.141109	-0.168252	0.420987	-0.029728	0.476201	0.260314	-0.568671	-0.371407	...	-0.208254	-0.559825	-0.026398	-0.3714

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V21	V22	V23	V2
6	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.005159	0.081213	0.464960	-0.099254	...	-0.167716	-0.270710	-0.154104	-0.7800
7	-0.644269	1.417964	1.074380	-0.492199	0.948934	0.428118	1.120631	-3.807864	0.615375	1.249376	...	1.943465	-1.015455	0.057504	-0.6497
8	-0.894286	0.286157	-0.113192	-0.271526	2.669599	3.721818	0.370145	0.851084	-0.392048	-0.410430	...	-0.073425	-0.268092	-0.204233	1.0115
9	-0.338262	1.119593	1.044367	-0.222187	0.499361	-0.246761	0.651583	0.069539	-0.736727	-0.366846	...	-0.246914	-0.633753	-0.120794	-0.3850
10	1.449044	-1.176339	0.913860	-1.375667	-1.971383	-0.629152	-1.423236	0.048456	-1.720408	1.626659	...	-0.009302	0.313894	0.027740	0.5005
11	0.384978	0.616109	-0.874300	-0.094019	2.924584	3.317027	0.470455	0.538247	-0.558895	0.309755	...	0.049924	0.238422	0.009130	0.9967
12	1.249999	-1.221637	0.383930	-1.234899	-1.485419	-0.753230	-0.689405	-0.227487	-2.094011	1.323729	...	-0.231809	-0.483285	0.084668	0.3928
13	1.069374	0.287722	0.828613	2.712520	-0.178398	0.337544	-0.096717	0.115982	-0.221083	0.460230	...	-0.036876	0.074412	-0.071407	0.1047
14	-2.791855	-0.327771	1.641750	1.767473	-0.136588	0.807596	-0.422911	-1.907107	0.755713	1.151087	...	1.151663	0.222182	1.020586	0.0283
15	-0.752417	0.345485	2.057323	-1.468643	-1.158394	-0.077850	-0.608581	0.003603	-0.436167	0.747731	...	0.499625	1.353650	-0.256573	-0.0650
16	1.103215	-0.040296	1.267332	1.289091	-0.735997	0.288069	-0.586057	0.189380	0.782333	-0.267975	...	-0.024612	0.196002	0.013802	0.1037
17	-0.436905	0.918966	0.924591	-0.727219	0.915679	-0.127867	0.707642	0.087962	-0.665271	-0.737980	...	-0.194796	-0.672638	-0.156858	-0.8883
18	-5.401258	-5.450148	1.186305	1.736239	3.049106	-1.763406	-1.559738	0.160842	1.233090	0.345173	...	-0.503600	0.984460	2.458589	0.0421
19	1.492936	-1.029346	0.454795	-1.438026	-1.555434	-0.720961	-1.080664	-0.053127	-1.978682	1.638076	...	-0.177650	-0.175074	0.040002	0.2958
20	0.694885	-1.361819	1.029221	0.834159	-1.191209	1.309109	-0.878586	0.445290	-0.446196	0.568521	...	-0.295583	-0.571955	-0.050881	-0.3042
21	0.962496	0.328461	-0.171479	2.109204	1.129566	1.696038	0.107712	0.521502	-1.191311	0.724396	...	0.143997	0.402492	-0.048508	-1.3718
22	1.166616	0.502120	-0.067300	2.261569	0.428804	0.089474	0.241147	0.138082	-0.989162	0.922175	...	0.018702	-0.061972	-0.103855	-0.3704
23	0.247491	0.277666	1.185471	-0.092603	-1.314394	-0.150116	-0.946365	-1.617935	1.544071	-0.829881	...	1.650180	0.200454	-0.185353	0.4230
24	-1.946525	-0.044901	-0.405570	-1.013057	2.941968	2.955053	-0.063063	0.855546	0.049967	0.573743	...	-0.579526	-0.799229	0.870300	0.9834

25 rows × 30 columns

In [269…

```
master_card['Class'].value_counts()
```

Out[269…

```
0     284315
```

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

In [270…

```
master_card[master_card['Class']==1]
```

Out[270…

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V21	V22	V23	
541	-2.312227	1.951992	-1.609851	3.997906	-0.522188	-1.426545	-2.537387	1.391657	-2.770089	-2.772272	...	0.517232	-0.035049	-0.465211	0.3
623	-3.043541	-3.157307	1.088463	2.288644	1.359805	-1.064823	0.325574	-0.067794	-0.270953	-0.838587	...	0.661696	0.435477	1.375966	-0.2
4920	-2.303350	1.759247	-0.359745	2.330243	-0.821628	-0.075788	0.562320	-0.399147	-0.238253	-1.525412	...	-0.294166	-0.932391	0.172726	-0.0
6108	-4.397974	1.358367	-2.592844	2.679787	-1.128131	-1.706536	-3.496197	-0.248778	-0.247768	-4.801637	...	0.573574	0.176968	-0.436207	-0.0
6329	1.234235	3.019740	-4.304597	4.732795	3.624201	-1.357746	1.713445	-0.496358	-1.282858	-2.447469	...	-0.379068	-0.704181	-0.656805	-1.6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
279863	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	-0.882850	0.697211	-2.064945	-5.587794	...	0.778584	-0.319189	0.639419	-0.2
280143	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.326536	-1.413170	0.248525	-1.127396	-3.232153	...	0.370612	0.028234	-0.145640	-0.0
280149	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.003346	-2.234739	1.210158	-0.652250	-3.463891	...	0.751826	0.834108	0.190944	0.0
281144	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	-2.208002	1.058733	-1.632333	-5.245984	...	0.583276	-0.269209	-0.456108	-0.1
281674	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	0.223050	-0.068384	0.577829	-0.888722	...	-0.164350	-0.295135	-0.072173	-0.4

492 rows × 30 columns

In [271…

```
master_card.mean()
```

Out[271…

```
V1        3.918649e-15
V2        5.682686e-16
V3       -8.761736e-15
V4        2.811118e-15
V5       -1.552103e-15
V6        2.040130e-15
V7       -1.698953e-15
V8       -1.893285e-16
V9       -3.147640e-15
```

```
V10       1.772925e-15
V11       9.289524e-16
V12      -1.803266e-15
V13       1.674888e-15
V14       1.475621e-15
V15       3.501098e-15
V16       1.392460e-15
V17      -7.466538e-16
V18       4.258754e-16
V19       9.019919e-16
V20       5.126845e-16
V21       1.473120e-16
V22       8.042109e-16
V23       5.282512e-16
V24       4.456271e-15
V25       1.426896e-15
V26       1.701640e-15
V27      -3.662252e-16
V28      -1.217809e-16
Amount    8.834962e+01
Class     1.727486e-03
dtype: float64
```

In [272…

```
master_card.corr()
```

Out[272…

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V1	1.000000e+00	4.135835e-16	-1.227819e-15	-9.215150e-16	1.812612e-17	-6.506567e-16	-1.005191e-15	-2.433822e-16	-1.513678e-16	7.388135e-17
V2	4.135835e-16	1.000000e+00	3.243764e-16	-1.121065e-15	5.157519e-16	2.787346e-16	2.055934e-16	-5.377041e-17	1.978488e-17	-3.991394e-16
V3	-1.227819e-15	3.243764e-16	1.000000e+00	4.711293e-16	-6.539009e-17	1.627627e-15	4.895305e-16	-1.268779e-15	5.568367e-16	1.156587e-15
V4	-9.215150e-16	-1.121065e-15	4.711293e-16	1.000000e+00	-1.719944e-15	-7.491959e-16	-4.104503e-16	5.697192e-16	6.923247e-16	2.232685e-16
V5	1.812612e-17	5.157519e-16	-6.539009e-17	-1.719944e-15	1.000000e+00	2.408382e-16	2.715541e-16	7.437229e-16	7.391702e-16	-5.202306e-16
V6	-6.506567e-16	2.787346e-16	1.627627e-15	-7.491959e-16	2.408382e-16	1.000000e+00	1.191668e-16	-1.104219e-16	4.131207e-16	5.932243e-17

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V7	-1.005191e-15	2.055934e-16	4.895305e-16	-4.104503e-16	2.715541e-16	1.191668e-16	1.000000e+00	3.344412e-16	1.122501e-15	-7.492834e-17
V8	-2.433822e-16	-5.377041e-17	-1.268779e-15	5.697192e-16	7.437229e-16	-1.104219e-16	3.344412e-16	1.000000e+00	4.356078e-16	-2.801370e-16
V9	-1.513678e-16	1.978488e-17	5.568367e-16	6.923247e-16	7.391702e-16	4.131207e-16	1.122501e-15	4.356078e-16	1.000000e+00	-4.642274e-16
V10	7.388135e-17	-3.991394e-16	1.156587e-15	2.232685e-16	-5.202306e-16	5.932243e-17	-7.492834e-17	-2.801370e-16	-4.642274e-16	1.000000e+00
V11	2.125498e-16	1.975426e-16	1.576830e-15	3.459380e-16	7.203963e-16	1.980503e-15	1.425248e-16	2.487043e-16	1.354680e-16	-4.622103e-16
V12	2.053457e-16	-9.568710e-17	6.310231e-16	-5.625518e-16	7.412552e-16	2.375468e-16	-3.536655e-18	1.839891e-16	-1.079314e-15	1.771869e-15
V13	-2.425603e-17	6.295388e-16	2.807652e-16	1.303306e-16	5.886991e-16	-1.211182e-16	1.266462e-17	-2.921856e-16	2.251072e-15	-5.418460e-16
V14	-5.020280e-16	-1.730566e-16	4.739859e-16	2.282280e-16	6.565143e-16	2.621312e-16	2.607772e-16	-8.599156e-16	3.784757e-15	2.635936e-16
V15	3.547782e-16	-4.995814e-17	9.068793e-16	1.377649e-16	-8.720275e-16	-1.531188e-15	-1.690540e-16	4.127777e-16	-1.051167e-15	5.786332e-16
V16	7.212815e-17	1.177316e-17	8.299445e-16	-9.614528e-16	2.246261e-15	2.623672e-18	5.869302e-17	-5.254741e-16	-1.214086e-15	3.545450e-16
V17	-3.879840e-16	-2.685296e-16	7.614712e-16	-2.699612e-16	1.281914e-16	2.015618e-16	2.177192e-16	-2.269549e-16	1.113695e-15	1.542955e-15
V18	3.230206e-17	3.284605e-16	1.509897e-16	-5.103644e-16	5.308590e-16	1.223814e-16	7.604126e-17	-3.667974e-16	4.993240e-16	3.902423e-16
V19	1.502024e-16	-7.118719e-18	3.463522e-16	-3.980557e-16	-1.450421e-16	-1.865597e-16	-1.881008e-16	-3.875186e-16	-1.376135e-16	3.437633e-17
V20	4.654551e-16	2.506675e-16	-9.316409e-16	-1.857247e-16	-3.554057e-16	-1.858755e-16	9.379684e-16	2.033737e-16	-2.343720e-16	-1.331556e-15
V21	-2.457409e-16	-8.480447e-17	5.706192e-17	-1.949553e-16	-3.920976e-16	5.833316e-17	-2.027779e-16	3.892798e-16	1.936953e-16	1.177547e-15

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V22	-4.290944e-16	1.526333e-16	-1.133902e-15	-6.276051e-17	1.253751e-16	-4.705235e-19	-8.898922e-16	2.026927e-16	-7.071869e-16	-6.418202e-16
V23	6.168652e-16	1.634231e-16	-4.983035e-16	9.164206e-17	-8.428683e-18	1.046712e-16	-4.387401e-16	6.377260e-17	-5.214137e-16	3.214491e-16
V24	-4.425156e-17	1.247925e-17	2.686834e-19	1.584638e-16	-1.149255e-15	-1.071589e-15	7.434913e-18	-1.047097e-16	-1.430343e-16	-1.355885e-16
V25	-9.605737e-16	-4.478846e-16	-1.104734e-15	6.070716e-16	4.808532e-16	4.562861e-16	-3.094082e-16	-4.653279e-16	6.757763e-16	-2.846052e-16
V26	-1.581290e-17	2.057310e-16	-1.238062e-16	-4.247268e-16	4.319541e-16	-1.357067e-16	-9.657637e-16	-1.727276e-16	-7.888853e-16	-3.028119e-16
V27	1.198124e-16	-4.966953e-16	1.045747e-15	3.977061e-17	6.590482e-16	-4.452461e-16	-1.782106e-15	1.299943e-16	-6.709655e-17	-2.197977e-16
V28	2.083082e-15	-5.093836e-16	9.775546e-16	-2.761403e-18	-5.613951e-18	2.594754e-16	-2.776530e-16	-6.200930e-16	1.110541e-15	4.864782e-17
Amount	-2.277087e-01	-5.314089e-01	-2.108805e-01	9.873167e-02	-3.863563e-01	2.159812e-01	3.973113e-01	-1.030791e-01	-4.424560e-02	-1.015021e-01
Class	-1.013473e-01	9.128865e-02	-1.929608e-01	1.334475e-01	-9.497430e-02	-4.364316e-02	-1.872566e-01	1.987512e-02	-9.773269e-02	-2.168829e-01

30 rows × 30 columns

In [273…

```
type(master_card)
```

Out[273…

```
pandas.core.frame.DataFrame
```

In [274…

```
data = master_card
```

In [275…

```
from sklearn.preprocessing import StandardScaler

data['normAmount'] = StandardScaler().fit_transform(data['Amount'].values.reshape(-1, 1))
```

```
data = data.drop(['Amount'],axis=1)
data.head()
```

Out[275…

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V21	V22	V23	V24
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	...	-0.018307	0.277838	-0.110474	0.066928
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	...	-0.225775	-0.638672	0.101288	-0.339846
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	...	0.247998	0.771679	0.909412	-0.689281
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	...	-0.108300	0.005274	-0.190321	-1.175575
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	...	-0.009431	0.798278	-0.137458	0.141267

5 rows × 30 columns

In [276…

```
X = data.iloc[:, data.columns != 'Class']
y = data.iloc[:, data.columns == 'Class']
```

In [277…

```
y.count()
```

Out[277…

```
Class    284807
dtype: int64
```

In [278…

```
# Number of data points in the minority class
number_records_fraud = len(data[data.Class == 1])
fraud_indices = np.array(data[data.Class == 1].index)

# Picking the indices of the normal classes
normal_indices = data[data.Class == 0].index

# Out of the indices we picked, randomly select "x" number (number_records_fraud)
random_normal_indices = np.random.choice(normal_indices, number_records_fraud, replace = False)
random_normal_indices = np.array(random_normal_indices)

# Appending the 2 indices
under_sample_indices = np.concatenate([fraud_indices,random_normal_indices])
```

```
# Under sample dataset
under_sample_data = data.iloc[under_sample_indices,:]

X_undersample = under_sample_data.iloc[:, under_sample_data.columns != 'Class']
y_undersample = under_sample_data.iloc[:, under_sample_data.columns == 'Class']

# Showing ratio
print("Percentage of normal transactions: ", len(under_sample_data[under_sample_data.Class == 0])/len(under_sample_data))
print("Percentage of fraud transactions: ", len(under_sample_data[under_sample_data.Class == 1])/len(under_sample_data))
print("Total number of transactions in resampled data: ", len(under_sample_data))
```

```
Percentage of normal transactions:  0.5
Percentage of fraud transactions:  0.5
Total number of transactions in resampled data:  984
```

In [279…

```
from sklearn.model_selection import train_test_split

# Whole dataset
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.3, random_state = 0)

print("Number transactions original train dataset: ", len(X_train))
print("Number transactions original test dataset: ", len(X_test))
print("Total number of transactions: ", len(X_train)+len(X_test))

# Undersampled dataset
X_train_undersample, X_test_undersample, y_train_undersample, y_test_undersample = train_test_split(X_undersample
                                                                                                    ,y_undersample
                                                                                                    ,test_size = 0.3
                                                                                                    ,random_state = 0)
print("")
print("Number transactions train dataset: ", len(X_train_undersample))
print("Number transactions test dataset: ", len(X_test_undersample))
print("Total number of transactions: ", len(X_train_undersample)+len(X_test_undersample))
```

```
Number transactions original train dataset:  199364
Number transactions original test dataset:  85443
Total number of transactions:  284807

Number transactions train dataset:  688
Number transactions test dataset:  296
Total number of transactions:  984
```

In [280…

```
X_train, X_test, y_train, y_test = X_train_undersample, X_test_undersample, y_train_undersample, y_test_undersample
```

In [281…

```
# Initialze the estimators
clf1 = RandomForestClassifier(random_state=42)
clf2 = SVC(probability=True, random_state=42)
clf3 = LogisticRegression(random_state=42)
clf4 = DecisionTreeClassifier(random_state=42)
clf5 = KNeighborsClassifier()
clf6 = GaussianNB()
clf7 = GradientBoostingClassifier(random_state=42)
```

In [282…

```
# Initiaze the hyperparameters for each dictionary
param1 = {}
param1['classifier__n_estimators'] = [10, 50, 100, 250]
param1['classifier__max_depth'] = [5, 10, 20]
param1['classifier__class_weight'] = [None, {0:1,1:5}, {0:1,1:10}, {0:1,1:25}]
param1['classifier'] = [clf1]

param2 = {}
param2['classifier__C'] = [10**-2, 10**-1, 10**0, 10**1, 10**2]
param2['classifier__class_weight'] = [None, {0:1,1:5}, {0:1,1:10}, {0:1,1:25}]
param2['classifier'] = [clf2]

param3 = {}
param3['classifier__C'] = [10**-2, 10**-1, 10**0, 10**1, 10**2]
param3['classifier__penalty'] = ['l1', 'l2']
param3['classifier__class_weight'] = [None, {0:1,1:5}, {0:1,1:10}, {0:1,1:25}]
param3['classifier'] = [clf3]

param4 = {}
param4['classifier__max_depth'] = [5,10,25,None]
param4['classifier__min_samples_split'] = [2,5,10]
param4['classifier__class_weight'] = [None, {0:1,1:5}, {0:1,1:10}, {0:1,1:25}]
param4['classifier'] = [clf4]

param5 = {}
param5['classifier__n_neighbors'] = [2,5,10,25,50]
param5['classifier'] = [clf5]
```

```
param6 = {}
# param6['var_smoothing'] = np.logspace(0,-9, num=100)
param6['classifier'] = [clf6]

param7 = {}
param7['classifier__n_estimators'] = [10, 50, 100, 250]
param7['classifier__max_depth'] = [5, 10, 20]
param7['classifier'] = [clf7]
```

In [283…

```
np.logspace(0,-9, num=100)
```

Out[283…

```
array([1.00000000e+00, 8.11130831e-01, 6.57933225e-01, 5.33669923e-01,
       4.32876128e-01, 3.51119173e-01, 2.84803587e-01, 2.31012970e-01,
       1.87381742e-01, 1.51991108e-01, 1.23284674e-01, 1.00000000e-01,
       8.11130831e-02, 6.57933225e-02, 5.33669923e-02, 4.32876128e-02,
       3.51119173e-02, 2.84803587e-02, 2.31012970e-02, 1.87381742e-02,
       1.51991108e-02, 1.23284674e-02, 1.00000000e-02, 8.11130831e-03,
       6.57933225e-03, 5.33669923e-03, 4.32876128e-03, 3.51119173e-03,
       2.84803587e-03, 2.31012970e-03, 1.87381742e-03, 1.51991108e-03,
       1.23284674e-03, 1.00000000e-03, 8.11130831e-04, 6.57933225e-04,
       5.33669923e-04, 4.32876128e-04, 3.51119173e-04, 2.84803587e-04,
       2.31012970e-04, 1.87381742e-04, 1.51991108e-04, 1.23284674e-04,
       1.00000000e-04, 8.11130831e-05, 6.57933225e-05, 5.33669923e-05,
       4.32876128e-05, 3.51119173e-05, 2.84803587e-05, 2.31012970e-05,
       1.87381742e-05, 1.51991108e-05, 1.23284674e-05, 1.00000000e-05,
       8.11130831e-06, 6.57933225e-06, 5.33669923e-06, 4.32876128e-06,
       3.51119173e-06, 2.84803587e-06, 2.31012970e-06, 1.87381742e-06,
       1.51991108e-06, 1.23284674e-06, 1.00000000e-06, 8.11130831e-07,
       6.57933225e-07, 5.33669923e-07, 4.32876128e-07, 3.51119173e-07,
       2.84803587e-07, 2.31012970e-07, 1.87381742e-07, 1.51991108e-07,
       1.23284674e-07, 1.00000000e-07, 8.11130831e-08, 6.57933225e-08,
       5.33669923e-08, 4.32876128e-08, 3.51119173e-08, 2.84803587e-08,
       2.31012970e-08, 1.87381742e-08, 1.51991108e-08, 1.23284674e-08,
       1.00000000e-08, 8.11130831e-09, 6.57933225e-09, 5.33669923e-09,
       4.32876128e-09, 3.51119173e-09, 2.84803587e-09, 2.31012970e-09,
       1.87381742e-09, 1.51991108e-09, 1.23284674e-09, 1.00000000e-09])
```

In [284…

```
c=0
for i in(clf1,clf2,clf3,clf4,clf5,clf6,clf7):

    pipeline = Pipeline([('classifier', i)])
    params = [param1, param2, param3, param4, param5, param6, param7]
```

```
    gs = GridSearchCV(pipeline, params[c], cv=3, n_jobs=-1, scoring='roc_auc').fit(X_train, y_train)
#     print(gs.best_params_)
    print(i)
#     print(pipeline, params[c])
    c=c+1
    # ROC-AUC score for the best model
    # Test data performance
    print("Test Precision:",precision_score(gs.predict(X_test), y_test))
    print("Test Recall:",recall_score(gs.predict(X_test), y_test))
    print("Test ROC AUC Score:",roc_auc_score(gs.predict(X_test), y_test))

#     print(gs.estimator)
    print("\n")
```

```
RandomForestClassifier(class_weight={0: 1, 1: 10}, max_depth=10,
                       n_estimators=250, random_state=42)
Test Precision: 0.9047619047619048
Test Recall: 0.9925373134328358
Test ROC AUC Score: 0.953058780173208


SVC(C=10, probability=True, random_state=42)
Test Precision: 0.9183673469387755
Test Recall: 1.0
Test ROC AUC Score: 0.9627329192546583


LogisticRegression(C=0.01, random_state=42)
Test Precision: 0.8707482993197279
Test Recall: 0.9922480620155039
Test ROC AUC Score: 0.9392378034628418


DecisionTreeClassifier(max_depth=5, min_samples_split=10, random_state=42)
Test Precision: 0.9047619047619048
Test Recall: 0.9432624113475178
Test ROC AUC Score: 0.9264699153511782


KNeighborsClassifier(n_neighbors=50)
Test Precision: 0.8435374149659864
Test Recall: 1.0
```

```
Test ROC AUC Score: 0.9331395348837209


GaussianNB()
Test Precision: 0.8571428571428571
Test Recall: 0.9692307692307692
Test ROC AUC Score: 0.9213623725671919


GradientBoostingClassifier(max_depth=5, random_state=42)
Test Precision: 0.9047619047619048
Test Recall: 0.9779411764705882
Test ROC AUC Score: 0.9452205882352941
```

In [285…

```
gs.best_estimator_
```

Out[285…

```
Pipeline(steps=[('classifier',
                 GradientBoostingClassifier(max_depth=5, random_state=42))])
```

In [286…

```
gs.best_score_
```

Out[286…

```
0.9828003847046728
```

In [287…

```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import GridSearchCV

param_grid_nb = {
    'var_smoothing': np.logspace(0,-9, num=100)
}

nbModel_grid = GridSearchCV(estimator=GaussianNB(), param_grid=param_grid_nb, verbose=1, cv=10, n_jobs=-1)
nbModel_grid.fit(X_train, y_train)
print(nbModel_grid.best_estimator_)
```

```
Fitting 10 folds for each of 100 candidates, totalling 1000 fits
GaussianNB(var_smoothing=6.579332246575683e-05)
```

In [288…

```
y_pred = nbModel_grid.predict(X_test)
```

```
print(y_pred)
```

```
[0 0 1 0 0 1 1 0 0 0 0 0 1 1 0 1 0 0 1 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 1 1 1
 0 1 1 1 0 0 1 0 0 1 1 1 0 0 0 0 0 1 1 0 0 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0
 0 1 0 1 1 0 1 0 1 1 0 0 1 0 1 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 1 0 0 1 0 1 0
 1 1 1 0 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0
 0 1 1 0 0 0 0 1 0 0 0 0 0 1 1 0 1 1 1 1 0 0 1 0 0 1 0 1 0 0 0 1 0 1 0 1 0
 1 0 1 0 0 0 0 1 0 1 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 1 0 1
 1 0 0 1 1 1 1 0 0 1 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 0 1 0 1
 1 1 1 1 1 0 0 0 0 0 1 0 1 0 1 0 1 1 0 0 1 0 0 0 1 1 1 1 0 1 1 0 1 0 0 0 0]
```

In [289…

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_pred), ": is the confusion matrix")
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred), ": is the accuracy score")
from sklearn.metrics import precision_score
print(precision_score(y_test, y_pred), ": is the precision score")
from sklearn.metrics import recall_score
print(recall_score(y_test, y_pred), ": is the recall score")
from sklearn.metrics import f1_score
print(f1_score(y_test, y_pred), ": is the f1 score")
```

```
[[145   4]
 [ 21 126]] : is the confusion matrix
0.9155405405405406 : is the accuracy score
0.9692307692307692 : is the precision score
0.8571428571428571 : is the recall score
0.9097472924187725 : is the f1 score
```