

```
In [1]: 1 import warnings
        2 warnings.filterwarnings('ignore')
```

```
In [2]: 1 import numpy as np # Linear algebra
        2 import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
        3
        4 import matplotlib.pyplot as plt
        5 import seaborn as sb
        6
        7 import xgboost as xgb
        8 from xgboost import XGBClassifier
        9
       10
       11
       12 from sklearn.ensemble import IsolationForest
       13 from sklearn.neighbors import KNeighborsClassifier
       14 from sklearn.model_selection import train_test_split
       15
       16 from sklearn.neighbors import LocalOutlierFactor
       17 from sklearn.svm import OneClassSVM
       18 from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
       19
       20 # Input data files are available in the read-only "../input/" directory
       21 # For example, running this (by clicking run or pressing Shift+Enter) will list
       22
       23 import os
       24 for dirname, _, filenames in os.walk('../'):
       25     for filename in filenames:
       26         print(os.path.join(dirname, filename))
       27
       28 # You can write up to 20GB to the current directory (/kaggle/working/) that gets
       29 # You can also write temporary files to /kaggle/temp/, but they won't be saved
       30
       31
       32
       33
       34
       35
       36
       37
       38
       39
       40
       41
       42
       43
       44
       45
       46
       47
       48
       49
       50
       51
       52
       53
       54
       55
       56
       57
       58
       59
       60
       61
       62
       63
       64
       65
       66
       67
       68
       69
       70
       71
       72
       73
       74
       75
       76
       77
       78
       79
       80
       81
       82
       83
       84
       85
       86
       87
       88
       89
       90
       91
       92
       93
       94
       95
       96
       97
       98
       99
      100
      101
      102
      103
      104
      105
      106
      107
      108
      109
      110
      111
      112
      113
      114
      115
      116
      117
      118
      119
      120
      121
      122
      123
      124
      125
      126
      127
      128
      129
      130
      131
      132
      133
      134
      135
      136
      137
      138
      139
      140
      141
      142
      143
      144
      145
      146
      147
      148
      149
      150
      151
      152
      153
      154
      155
      156
      157
      158
      159
      160
      161
      162
      163
      164
      165
      166
      167
      168
      169
      170
      171
      172
      173
      174
      175
      176
      177
      178
      179
      180
      181
      182
      183
      184
      185
      186
      187
      188
      189
      190
      191
      192
      193
      194
      195
      196
      197
      198
      199
      200
      201
      202
      203
      204
      205
      206
      207
      208
      209
      210
      211
      212
      213
      214
      215
      216
      217
      218
      219
      220
      221
      222
      223
      224
      225
      226
      227
      228
      229
      230
      231
      232
      233
      234
      235
      236
      237
      238
      239
      240
      241
      242
      243
      244
      245
      246
      247
      248
      249
      250
      251
      252
      253
      254
      255
      256
      257
      258
      259
      260
      261
      262
      263
      264
      265
      266
      267
      268
      269
      270
      271
      272
      273
      274
      275
      276
      277
      278
      279
      280
      281
      282
      283
      284
      285
      286
      287
      288
      289
      290
      291
      292
      293
      294
      295
      296
      297
      298
      299
      300
      301
      302
      303
      304
      305
      306
      307
      308
      309
      310
      311
      312
      313
      314
      315
      316
      317
      318
      319
      320
      321
      322
      323
      324
      325
      326
      327
      328
      329
      330
      331
      332
      333
      334
      335
      336
      337
      338
      339
      340
      341
      342
      343
      344
      345
      346
      347
      348
      349
      350
      351
      352
      353
      354
      355
      356
      357
      358
      359
      360
      361
      362
      363
      364
      365
      366
      367
      368
      369
      370
      371
      372
      373
      374
      375
      376
      377
      378
      379
      380
      381
      382
      383
      384
      385
      386
      387
      388
      389
      390
      391
      392
      393
      394
      395
      396
      397
      398
      399
      400
      401
      402
      403
      404
      405
      406
      407
      408
      409
      410
      411
      412
      413
      414
      415
      416
      417
      418
      419
      420
      421
      422
      423
      424
      425
      426
      427
      428
      429
      430
      431
      432
      433
      434
      435
      436
      437
      438
      439
      440
      441
      442
      443
      444
      445
      446
      447
      448
      449
      450
      451
      452
      453
      454
      455
      456
      457
      458
      459
      460
      461
      462
      463
      464
      465
      466
      467
      468
      469
      470
      471
      472
      473
      474
      475
      476
      477
      478
      479
      480
      481
      482
      483
      484
      485
      486
      487
      488
      489
      490
      491
      492
      493
      494
      495
      496
      497
      498
      499
      500
      501
      502
      503
      504
      505
      506
      507
      508
      509
      510
      511
      512
      513
      514
      515
      516
      517
      518
      519
      520
      521
      522
      523
      524
      525
      526
      527
      528
      529
      530
      531
      532
      533
      534
      535
      536
      537
      538
      539
      540
      541
      542
      543
      544
      545
      546
      547
      548
      549
      550
      551
      552
      553
      554
      555
      556
      557
      558
      559
      560
      561
      562
      563
      564
      565
      566
      567
      568
      569
      570
      571
      572
      573
      574
      575
      576
      577
      578
      579
      580
      581
      582
      583
      584
      585
      586
      587
      588
      589
      590
      591
      592
      593
      594
      595
      596
      597
      598
      599
      600
      601
      602
      603
      604
      605
      606
      607
      608
      609
      610
      611
      612
      613
      614
      615
      616
      617
      618
      619
      620
      621
      622
      623
      624
      625
      626
      627
      628
      629
      630
      631
      632
      633
      634
      635
      636
      637
      638
      639
      640
      641
      642
      643
      644
      645
      646
      647
      648
      649
      650
      651
      652
      653
      654
      655
      656
      657
      658
      659
      660
      661
      662
      663
      664
      665
      666
      667
      668
      669
      670
      671
      672
      673
      674
      675
      676
      677
      678
      679
      680
      681
      682
      683
      684
      685
      686
      687
      688
      689
      690
      691
      692
      693
      694
      695
      696
      697
      698
      699
      700
      701
      702
      703
      704
      705
      706
      707
      708
      709
      710
      711
      712
      713
      714
      715
      716
      717
      718
      719
      720
      721
      722
      723
      724
      725
      726
      727
      728
      729
      730
      731
      732
      733
      734
      735
      736
      737
      738
      739
      740
      741
      742
      743
      744
      745
      746
      747
      748
      749
      750
      751
      752
      753
      754
      755
      756
      757
      758
      759
      760
      761
      762
      763
      764
      765
      766
      767
      768
      769
      770
      771
      772
      773
      774
      775
      776
      777
      778
      779
      780
      781
      782
      783
      784
      785
      786
      787
      788
      789
      790
      791
      792
      793
      794
      795
      796
      797
      798
      799
      800
      801
      802
      803
      804
      805
      806
      807
      808
      809
      810
      811
      812
      813
      814
      815
      816
      817
      818
      819
      820
      821
      822
      823
      824
      825
      826
      827
      828
      829
      830
      831
      832
      833
      834
      835
      836
      837
      838
      839
      840
      841
      842
      843
      844
      845
      846
      847
      848
      849
      850
      851
      852
      853
      854
      855
      856
      857
      858
      859
      860
      861
      862
      863
      864
      865
      866
      867
      868
      869
      870
      871
      872
      873
      874
      875
      876
      877
      878
      879
      880
      881
      882
      883
      884
      885
      886
      887
      888
      889
      890
      891
      892
      893
      894
      895
      896
      897
      898
      899
      900
      901
      902
      903
      904
      905
      906
      907
      908
      909
      910
      911
      912
      913
      914
      915
      916
      917
      918
      919
      920
      921
      922
      923
      924
      925
      926
      927
      928
      929
      930
      931
      932
      933
      934
      935
      936
      937
      938
      939
      940
      941
      942
      943
      944
      945
      946
      947
      948
      949
      950
      951
      952
      953
      954
      955
      956
      957
      958
      959
      960
      961
      962
      963
      964
      965
      966
      967
      968
      969
      970
      971
      972
      973
      974
      975
      976
      977
      978
      979
      980
      981
      982
      983
      984
      985
      986
      987
      988
      989
      990
      991
      992
      993
      994
      995
      996
      997
      998
      999
     1000
     1001
     1002
     1003
     1004
     1005
     1006
     1007
     1008
     1009
     1010
     1011
     1012
     1013
     1014
     1015
     1016
     1017
     1018
     1019
     1020
     1021
     1022
     1023
     1024
     1025
     1026
     1027
     1028
     1029
     1030
     1031
     1032
     1033
     1034
     1035
     1036
     1037
     1038
     1039
     1040
     1041
     1042
     1043
     1044
     1045
     1046
     1047
     1048
     1049
     1050
     1051
     1052
     1053
     1054
     1055
     1056
     1057
     1058
     1059
     1060
     1061
     1062
     1063
     1064
     1065
     1066
     1067
     1068
     1069
     1070
     1071
     1072
     1073
     1074
     1075
     1076
     1077
     1078
     1079
     1080
     1081
     1082
     1083
     1084
     1085
     1086
     1087
     1088
     1089
     1090
     1091
     1092
     1093
     1094
     1095
     1096
     1097
     1098
     1099
     1100
     1101
     1102
     1103
     1104
     1105
     1106
     1107
     1108
     1109
     1110
     1111
     1112
     1113
     1114
     1115
     1116
     1117
     1118
     1119
     1120
     1121
     1122
     1123
     1124
     1125
     1126
     1127
     1128
     1129
     1130
     1131
     1132
     1133
     1134
     1135
     1136
     1137
     1138
     1139
     1140
     1141
     1142
     1143
     1144
     1145
     1146
     1147
     1148
     1149
     1150
     1151
     1152
     1153
     1154
     1155
     1156
     1157
     1158
     1159
     1160
     1161
     1162
     1163
     1164
     1165
     1166
     1167
     1168
     1169
     1170
     1171
     1172
     1173
     1174
     1175
     1176
     1177
     1178
     1179
     1180
     1181
     1182
     1183
     1184
     1185
     1186
     1187
     1188
     1189
     1190
     1191
     1192
     1193
     1194
     1195
     1196
     1197
     1198
     1199
     1200
     1201
     1202
     1203
     1204
     1205
     1206
     1207
     1208
     1209
     1210
     1211
     1212
     1213
     1214
     1215
     1216
     1217
     1218
     1219
     1220
     1221
     1222
     1223
     1224
     1225
     1226
     1227
     1228
     1229
     1230
     1231
     1232
     1233
     1234
     1235
     1236
     1237
     1238
     1239
     1240
     1241
     1242
     1243
     1244
     1245
     1246
     1247
     1248
     1249
     1250
     1251
     1252
     1253
     1254
     1255
     1256
     1257
     1258
     1259
     1260
     1261
     1262
     1263
     1264
     1265
     1266
     1267
     1268
     1269
     1270
     1271
     1272
     1273
     1274
     1275
     1276
     1277
     1278
     1279
     1280
     1281
     1282
     1283
     1284
     1285
     1286
     1287
     1288
     1289
     1290
     1291
     1292
     1293
     1294
     1295
     1296
     1297
     1298
     1299
     1300
     1301
     1302
     1303
     1304
     1305
     1306
     1307
     1308
     1309
     1310
     1311
     1312
     1313
     1314
     1315
     1316
     1317
     1318
     1319
     1320
     1321
     1322
     1323
     1324
     1325
     1326
     1327
     1328
     1329
     1330
     1331
     1332
     1333
     1334
     1335
     1336
     1337
     1338
     1339
     1340
     1341
     1342
     1343
     1344
     1345
     1346
     1347
     1348
     1349
     1350
     1351
     1352
     1353
     1354
     1355
     1356
     1357
     1358
     1359
     1360
     1361
     1362
     1363
     1364
     1365
     1366
     1367
     1368
     1369
     1370
     1371
     1372
     1373
     1374
     1375
     1376
     1377
     1378
     1379
     1380
     1381
     1382
     1383
     1384
     1385
     1386
     1387
     1388
     1389
     1390
     1391
     1392
     1393
     1394
     1395
     1396
     1397
     1398
     1399
     1400
     1401
     1402
     1403
     1404
     1405
     1406
     1407
     1408
     1409
     1410
     1411
     1412
     1413
     1414
     1415
     1416
     1417
     1418
     1419
     1420
     1421
     1422
     1423
     1424
     1425
     1426
     1427
     1428
     1429
     1430
     1431
     1432
     1433
     1434
     1435
     1436
     1437
     1438
     1439
     1440
     1441
     1442
     1443
     1444
     1445
     1446
     1447
     1448
     1449
     1450
     1451
     1452
     1453
     1454
     1455
     1456
     1457
     1458
     1459
     1460
     1461
     1462
     1463
     1464
     1465
     1466
     1467
     1468
     1469
     1470
     1471
     1472
     1473
     1474
     1475
     1476
     1477
     1478
     1479
     1480
     1481
     1482
     1483
     1484
     1485
     1486
     1487
     1488
     1489
     1490
     1491
     1492
     1493
     1494
     1495
     1496
     1497
     1498
     1499
     1500
     1501
     1502
     1503
     1504
     1505
     1506
     1507
     1508
     1509
     1510
     1511
     1512
     1513
     1514
     1515
     1516
     1517
     1518
     1519
     1520
     1521
     1522
     1523
     1524
     1525
     1526
     1527
     1528
     1529
     1530
     1531
     1532
     1533
     1534
     1535
     1536
     1537
     1538
     1539
     1540
     1541
     1542
     1543
     1544
     1545
     1546
     1547
     1548
     1549
     1550
     1551
     1552
     1553
     1554
     1555
     1556
     1557
     1558
     1559
     1560
     1561
     1562
     1563
     1564
     1565
     1566
     1567
     1568
     1569
     1570
     1571
     1572
     1573
     1574
     1575
     1576
     1577
     1578
     1579
     1580
     1581
     1582
     1583
     1584
     1585
     1586
     1587
     1588
     1589
     1590
     1591
     1592
     1593
     1594
     1595
     1596
     1597
     1598
     1599
     1600
     1601
     1602
     1603
     1604
     1605
     1606
     1607
     1608
     1609
     1610
     1611
     1612
     1613
     1614
     1615
     1616
     1617
     1618
     1619
     1620
     1621
     1622
     1623
     1624
     1625
     1626
     1627
     1628
     1629
     1630
     1631
     1632
     1633
     1634
     1635
     1636
     1637
     1638
     1639
     1640
     1641
     1642
     1643
     1644
     1645
     1646
     1647
     1648
     1649
     1650
     1651
     1652
     1653
     1654
     1655
     1656
     1657
     1658
     1659
     1660
     1661
     1662
     1663
     1664
     1665
     1666
     1667
     1668
     1669
     1670
     1671
     1672
     1673
     1674
     1675
     1676
     1677
     1678
     1679
     1680
     1681
     1682
     1683
     1684
     1685
     1686
     1687
     1688
     1689
     1690
     1691
     1692
     1693
     1694
     1695
     1696
     1697
     1698
     1699
     1700
     1701
     1702
     1703
     1704
     1705
     1706
     1707
     1708
     1709
     1710
     1711
     1712
     1713
     1714
     1715
     1716
     1717
     1718
     1719
     1720
     1721
     1722
     1723
     1724
     1725
     1726
     1727
     1728
     1729
     1730
     1731
     1732
     1733
     1734
     1735
     1736
     1737
     1738
     1739
     1740
     1741
     1742
     1743
     1744
     1745
     1746
     1747
     1748
     1749
     1750
     1751
     1752
     1753
     1754
     1755
     1756
     1757
     1758
     1759
     1760
     1761
     1762
     1763
     1764
     1765
     1766
     1767
     1768
     1769
     1770
     1771
     1772
     1773
     1774
     1775
     1776
     1777
     1778
     1779
     1780
     1781
     1782
     1783
     1784
     1785
     1786
     1787
     1788
     1789
     1790
     1791
     1792
     1793
     1794
     1795
     1796
     1797
     1798
     1799
     1800
     1801
     1802
     1803
     1804
     1805
     1806
     1807
     1808
     1809
     1810
     1811
     1812
     1813
     1814
     1815
     1816
     1817
     1818
     1819
     1820
     1821
     1822
     1823
     1824
     1825
     1826
     1827
     1828
     1829
     1830
     1831
     1832
     1833
     1834
     1835
     1836
     1837
     1838
     1839
     1840
     1841
     1842
     1843
     1844
     1845
     1846
     1847
     1848
     1849
     1850
     1851
     1852
     1853
     1854
     1855
     1856
     1857
     1858
     1859
     1860
     1861
     1862
     1863
     1864
     1865
     1866
     1867
     1868
     1869
     1870
     1871
     1872
     1873
     1874
     1875
     1876
     1877
     1878
     1879
     1880
     1881
     1882
     1883
     1884
     1885
     1886
     1887

```

```
In [4]: 1 df_train['is_anomaly'] = df_train['is_anomaly'].replace(False,0).replace(True,1)
        2 df_train['is_anomaly'].value_counts()
```

```
Out[4]: is_anomaly
0      15054
1       776
Name: count, dtype: int64
```

```
In [5]: 1 df_train.isnull().sum()
        2
```

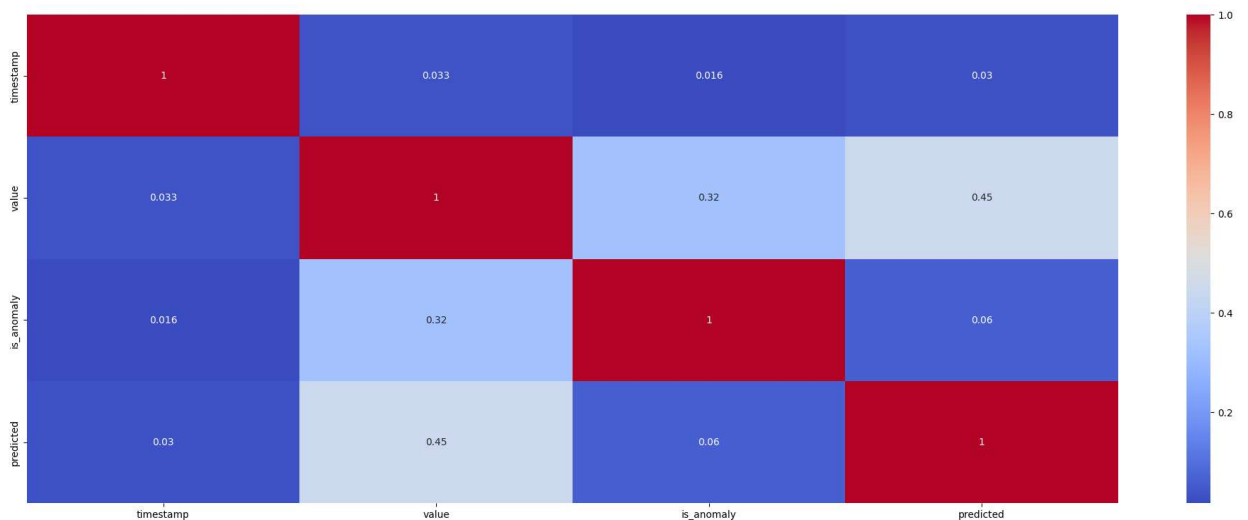
```
Out[5]: timestamp    0
       value        0
       is_anomaly    0
       predicted     0
       dtype: int64
```

```
In [6]: 1 df_train.describe()
```

```
Out[6]:
```

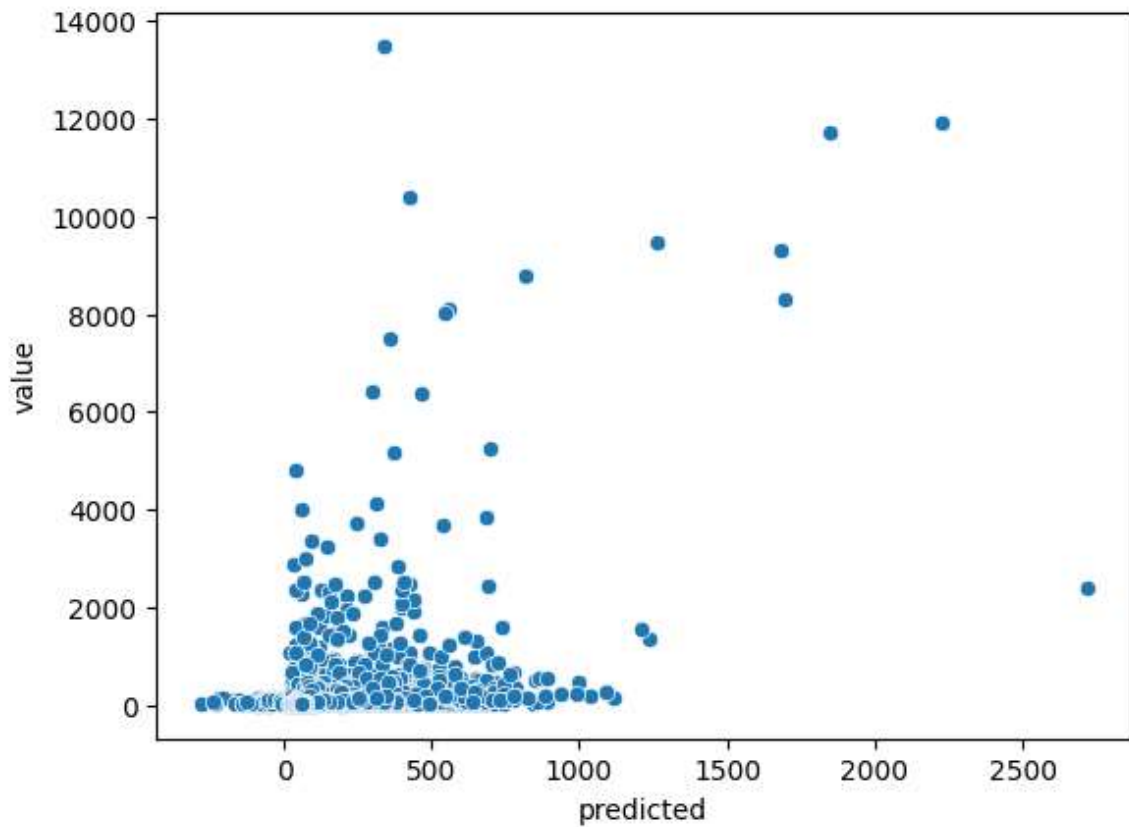
	timestamp	value	is_anomaly	predicted
<b>count</b>	1.583000e+04	15830.000000	15830.000000	15830.000000
<b>mean</b>	1.427383e+09	85.572205	0.049021	71.870715
<b>std</b>	1.370962e+06	321.760918	0.215918	92.450520
<b>min</b>	1.425009e+09	0.000000	0.000000	-281.389070
<b>25%</b>	1.426196e+09	29.000000	0.000000	32.919171
<b>50%</b>	1.427383e+09	47.000000	0.000000	49.771124
<b>75%</b>	1.428570e+09	76.000000	0.000000	75.948052
<b>max</b>	1.429757e+09	13479.000000	1.000000	2716.127200

```
In [7]: 1 plt.figure(figsize=(25, 9))
        2 sb.heatmap(df_train.corr(),annot=True,cmap='coolwarm')
        3 plt.show()
```



```
In [8]: 1 sb.scatterplot(x=df_train['predicted'], y=df_train['value'])
```

```
Out[8]: <Axes: xlabel='predicted', ylabel='value'>
```



```
In [9]: 1 print("Total No of Transactions:",df_train.size)
2
3 Fraud_df = df_train[df_train['is_anomaly']==True]
4 print("No of Anomalous Transactions:",len(Fraud_df))
5
6 Valid_df = df_train[df_train['is_anomaly']==False]
7 print("No of Valid Transactions:",len(Valid_df))
8
9 outlier_fraction = len(Fraud_df)/float(len(df_train))
10 valid_fraction = len(Valid_df)/float(len(df_train))
11 print("Percentage of Anomalous Transactions:",round((outlier_fraction*100),3))
12 print("Percentage of Valid Transactions:",round((valid_fraction*100),3))
```

```
Total No of Transactions: 63320
No of Anomalous Transactions: 776
No of Valid Transactions: 15054
Percentage of Anomalous Transactions: 4.902
Percentage of Valid Transactions: 95.098
```

```
In [10]: 1 X = df_train.drop(columns=['is_anomaly'],inplace=False,axis=1)
          2 X.head(2)
```

```
Out[10]:
```

	timestamp	value	predicted
0	1425008573	42	44.07250
1	1425008873	41	50.70939

```
In [11]: 1 y = df_train['is_anomaly']
          2 y.head(3)
```

```
Out[11]: 0    0
          1    0
          2    0
          Name: is_anomaly, dtype: int64
```

```
In [12]: 1 X.shape
          2 X_train = X.copy(deep=True)
          3 y_train = y.copy(deep=True)
```

```
In [13]: 1 state = np.random.RandomState(42)
          2 X_outliers = state.uniform(low=0, high=1, size=(X_train.shape[0], X_train.shape[1]))
```

```
In [14]: 1 classifiers = {
          2     "Isolation Forest":IsolationForest(n_estimators=100,
          3                                           max_samples=len(X_train),
          4                                           contamination=outlier_fraction,
          5                                           random_state=state,
          6                                           verbose=0),
          7     "Local Outlier Factor":LocalOutlierFactor(n_neighbors=20,
          8                                           algorithm='auto',
          9                                           leaf_size=30,
          10                                          metric='minkowski',
          11                                          novelty=False,
          12                                          p=2, metric_params=None,
          13                                          contamination=outlier_fraction),
          14     "Novelty Local Outlier Factor":LocalOutlierFactor(n_neighbors=20, algorithm='auto',
          15                                          leaf_size=30, metric='minkowski',
          16                                          novelty=True,p=2, metric_params=None,
          17                                          contamination=outlier_fraction),
          18     "Support Vector Machine":OneClassSVM(kernel='rbf', degree=3, gamma=0.1,nu=0.001,
          19                                          max_iter=-1),
          20     "XGBClassifier":XGBClassifier(objective="binary:logistic", random_state=42)
          21 }
```

```

In [15]: 1 f, axes = plt.subplots(1, 5, figsize=(20, 10), sharey='row')
2 for i, (clf_name,clf) in enumerate(classifiers.items()):
3     #Fit the data and tag outliers
4     print("###"*32)
5     if clf_name == "Local Outlier Factor":
6         y_pred = clf.fit_predict(X_train)
7         scores_prediction = clf.negative_outlier_factor_
8     elif clf_name == "Support Vector Machine":
9         clf.fit(X_train)
10        y_pred = clf.predict(X_train)
11    elif clf_name == "Novelty Local Outlier Factor":
12        clf.fit(X_train)
13        y_pred = clf.predict(X_train)
14        scores_prediction = clf.negative_outlier_factor_
15    elif clf_name == "XGBClassifier":
16        clf.fit(X_train,y_train)
17        y_pred = clf.predict(X_train)
18    else:
19        clf.fit(X_train)
20        scores_prediction = clf.decision_function(X_train)
21        y_pred = clf.predict(X_train)
22    # Reshape the prediction values to 0 for Valid transactions , 1 for Fraud
23    y_pred[y_pred == 1] = 0
24    y_pred[y_pred == -1] = 1
25    n_errors = (y_pred != y_train).sum()
26    # Run Classification Metrics
27    print("{}: {}".format(clf_name,n_errors))
28    ac_score = accuracy_score(y_train,y_pred)
29
30    print(f"Accuracy Score :{round(ac_score,2)}")
31    print("Classification Report :")
32    print(classification_report(y_train,y_pred))
33    cf_matrix = confusion_matrix(y_train, y_pred)
34    disp = ConfusionMatrixDisplay(cf_matrix)
35    disp.plot(ax=axes[i], values_format='.0f',cmap = "Blues")
36    axes[i].set_title(clf_name+"f1:"+str(round(ac_score,2)))
37    disp.im_.colorbar.remove()
38    disp.ax_.set_xlabel('')

```

```
#####
#####
Isolation Forest: 1026
Accuracy Score :0.94
Classification Report :
      precision    recall  f1-score   support

     0       0.97       0.97       0.97      15054
     1       0.34       0.34       0.34       776

 accuracy          0.94      15830
 macro avg         0.65       0.65       0.65      15830
weighted avg         0.94       0.94       0.94      15830

#####
#####
Local Outlier Factor: 1244
Accuracy Score :0.92
Classification Report :
      precision    recall  f1-score   support

     0       0.96       0.96       0.96      15054
     1       0.20       0.20       0.20       776

 accuracy          0.92      15830
 macro avg         0.58       0.58       0.58      15830
weighted avg         0.92       0.92       0.92      15830

#####
#####
Novelty Local Outlier Factor: 1139
Accuracy Score :0.93
Classification Report :
      precision    recall  f1-score   support

     0       0.96       0.97       0.96      15054
     1       0.22       0.18       0.20       776

 accuracy          0.93      15830
 macro avg         0.59       0.58       0.58      15830
weighted avg         0.92       0.93       0.92      15830

#####
#####
Support Vector Machine: 12204
Accuracy Score :0.23
Classification Report :
      precision    recall  f1-score   support

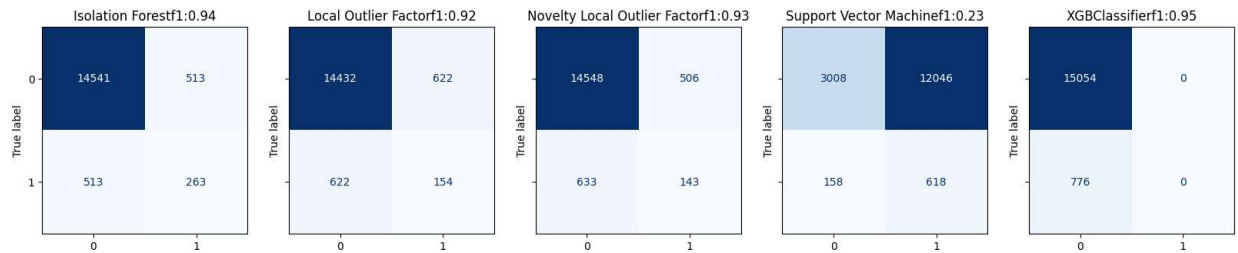
     0       0.95       0.20       0.33      15054
     1       0.05       0.80       0.09       776

 accuracy          0.23      15830
 macro avg         0.50       0.50       0.21      15830
weighted avg         0.91       0.23       0.32      15830

#####
#####
```

XGBClassifier: 776  
 Accuracy Score :0.95  
 Classification Report :

	precision	recall	f1-score	support
0	0.95	1.00	0.97	15054
1	0.00	0.00	0.00	776
accuracy			0.95	15830
macro avg	0.48	0.50	0.49	15830
weighted avg	0.90	0.95	0.93	15830



In [16]: 1 X\_test=pd.read\_csv('./test.csv')

In [17]: 1 clf = XGBClassifier(objective="binary:logistic", random\_state=42)  
 2 clf.fit(X\_train,y\_train)  
 3 y\_test\_pred = clf.predict(X\_test)

In [18]: 1 data={"timestamp":[],"is\_anomaly":[]}  
 2 for id,pred in zip(X["timestamp"].unique(),y\_test\_pred):  
 3 data["timestamp"].append(id)  
 4 data["is\_anomaly"].append(pred)

In [19]: 1 output=pd.DataFrame(data,columns=["timestamp","is\_anomaly"])  
 2 output.head(2)

Out[19]:

	timestamp	is_anomaly
0	1425008573	0
1	1425008873	0

In [20]: 1 output['is\_anomaly'].value\_counts()

Out[20]: is\_anomaly  
 0 3948  
 1 12  
 Name: count, dtype: int64

```
In [21]: 1 output.to_csv('submission.csv', index=False)
         2 print("Your submission was successfully saved!")
         3 output['is_anomaly'].value_counts()
```

Your submission was successfully saved!

```
Out[21]: is_anomaly
         0      3948
         1       12
         Name: count, dtype: int64
```

```
In [23]: 1 {'Isolation Forest': IsolationForest(contamination=0.04902084649399874, max_sampl
         es=15830,
           random_state=RandomState(MT19937) at 0x25B7C42DB40), 'Local Outli
         er Factor': LocalOutlierFactor(contamination=0.04902084649399874), 'Novelty Local
         Outlier Factor': LocalOutlierFactor(contamination=0.04902084649399874, novelty=Tr
         ue), 'Support Vector Machine': OneClassSVM(gamma=0.1, nu=0.05), 'XGBClassifier':
         XGBClassifier(base_score=None, booster=None, callbacks=None,
           colsample_bylevel=None, colsample_bynode=None,
           colsample_bytree=None, device=None, early_stopping_rounds=None,
           enable_categorical=False, eval_metric=None, feature_types=None,
           gamma=None, grow_policy=None, importance_type=None,
           interaction_constraints=None, learning_rate=None, max_bin=None,
           max_cat_threshold=None, max_cat_to_onehot=None,
           max_delta_step=None, max_depth=None, max_leaves=None,
           min_child_weight=None, missing=nan, monotone_constraints=None,
           multi_strategy=None, n_estimators=None, n_jobs=None,
           num_parallel_tree=None, random_state=42, ...)}
```

```
In [ ]: 1
```