

Dataset Analysis

Data

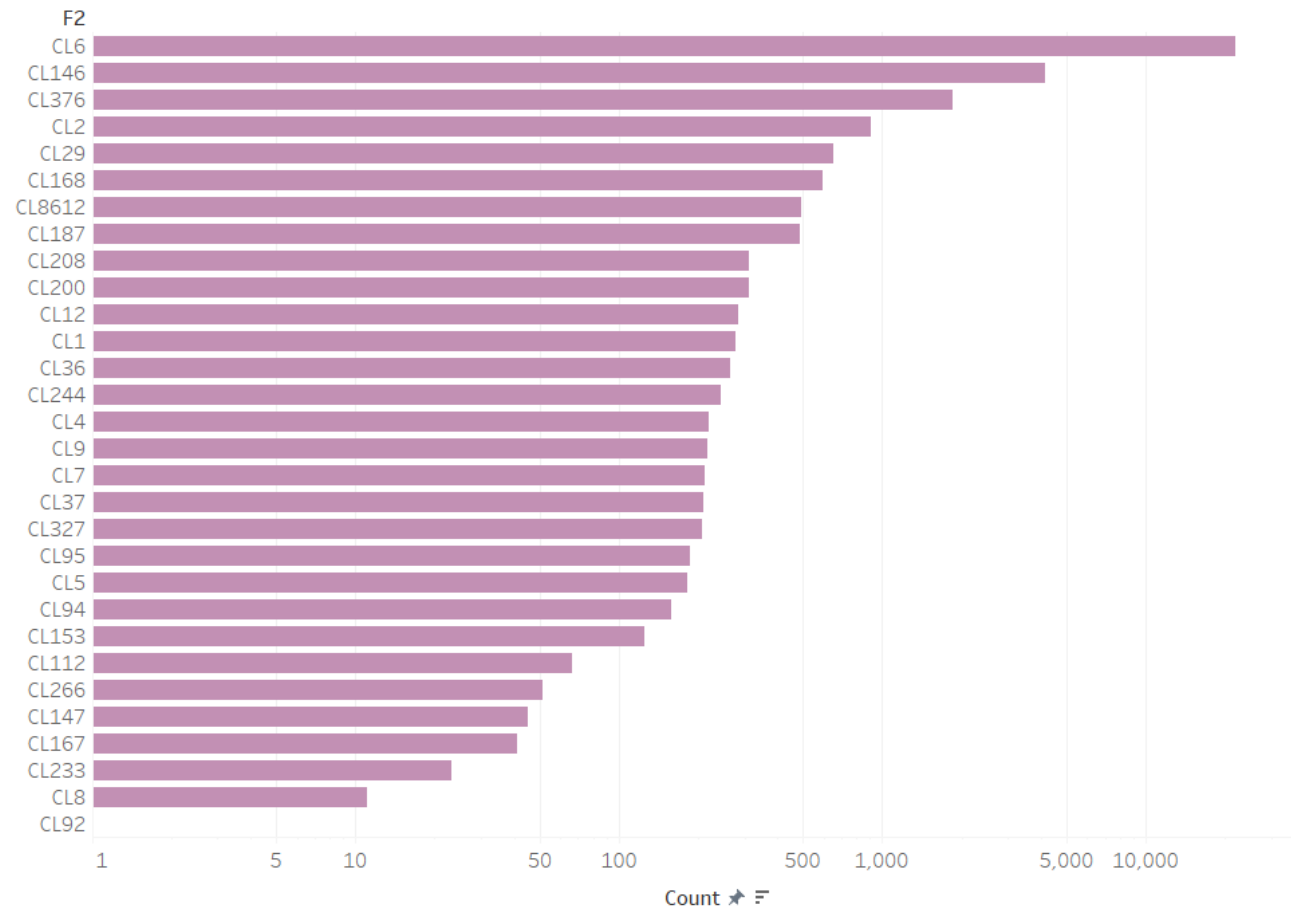
x

y

Data

y

Distribution of classes in dataset

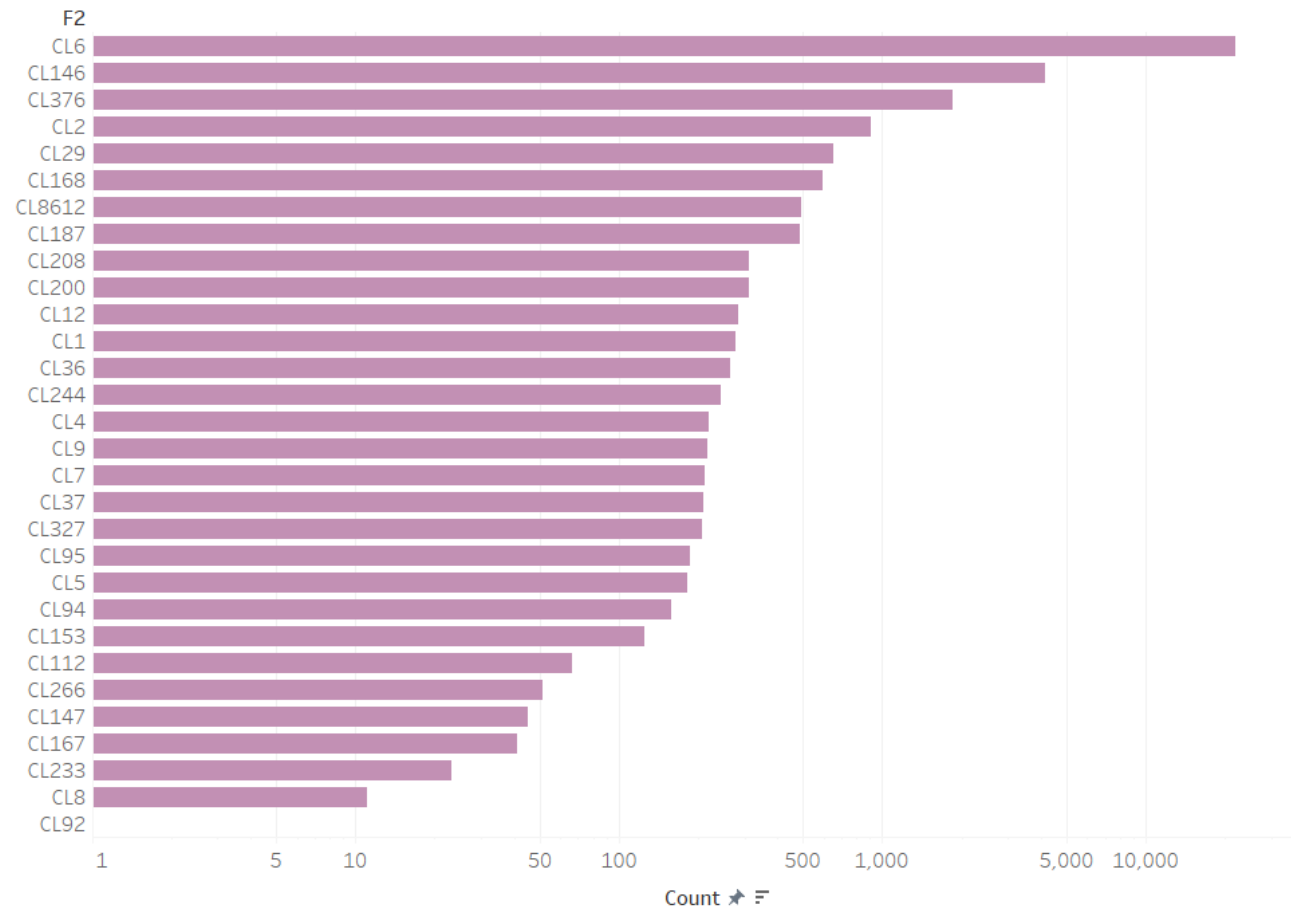


Multi Class Problem!

Data

y

Distribution of classes in dataset



Multi Class Problem!

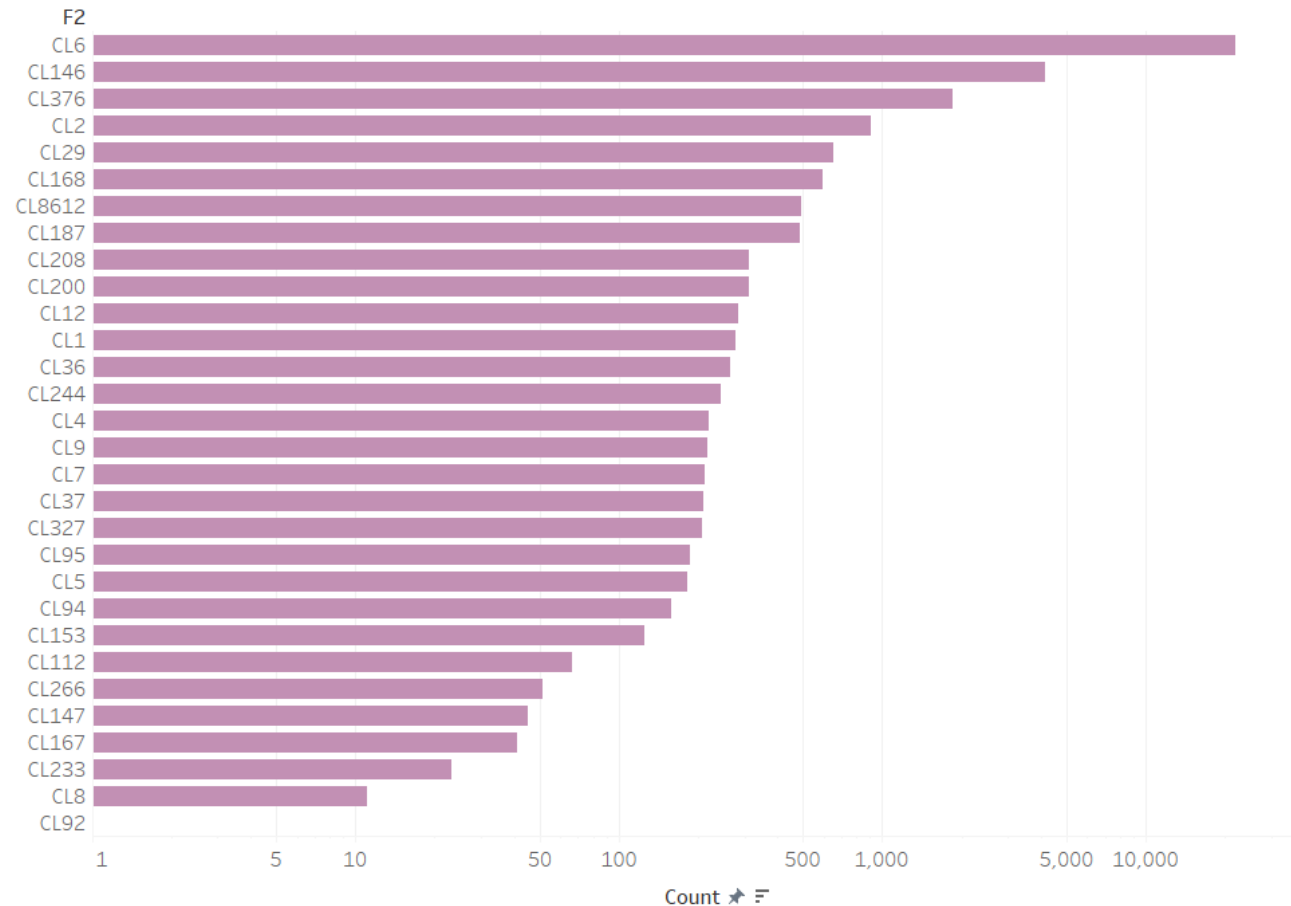
Highly Imbalance !!

- Stratify while splitting
- Synthetically model the minority the classes

Data

y

Distribution of classes in dataset

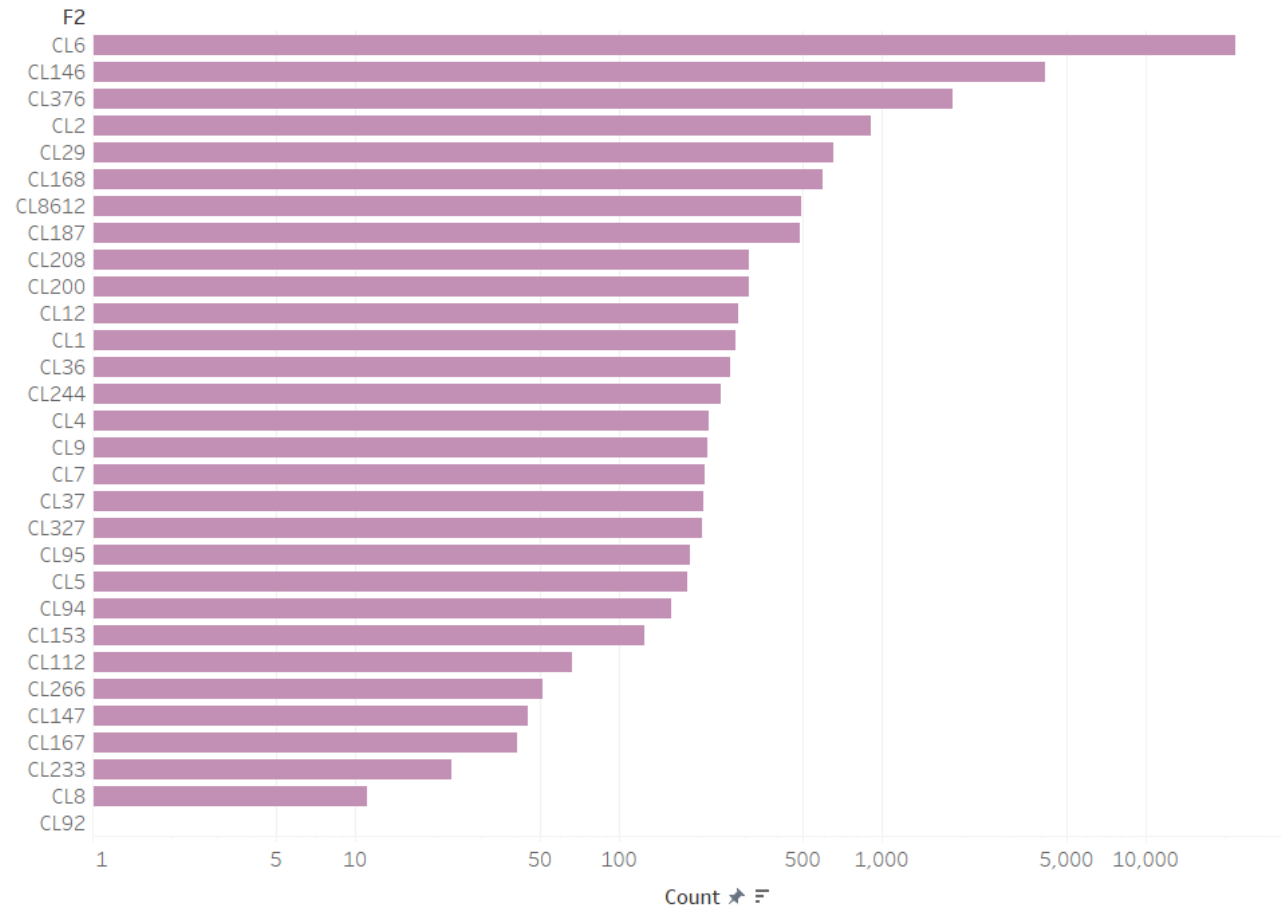


Multi Class Problem!
Highly Imbalance !!
Identify the metric – 'F1 – score' : average

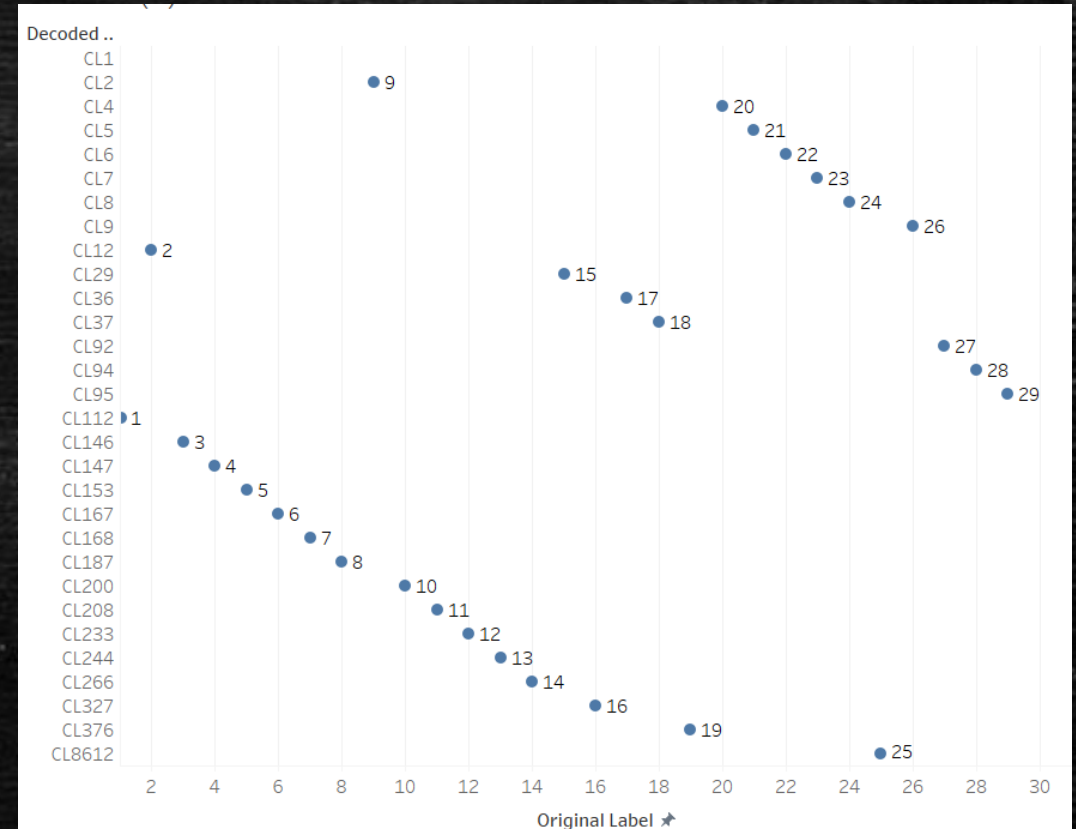
Data

y

Distribution of classes in dataset



Multi Class Problem!
Highly Imbalance !!
Identify the metric – 'F1 – score' : average
Encode the labels



Data

X

34553 samples, 7672 features

Data

X

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 34553 entries, 0 to 34552  
Columns: 7673 entries, GR1 to Class  
dtypes: category(1), float64(7672)  
memory usage: 2.0 GB
```

34553 samples, 7672 features

- Features – numeric
- No duplicated value

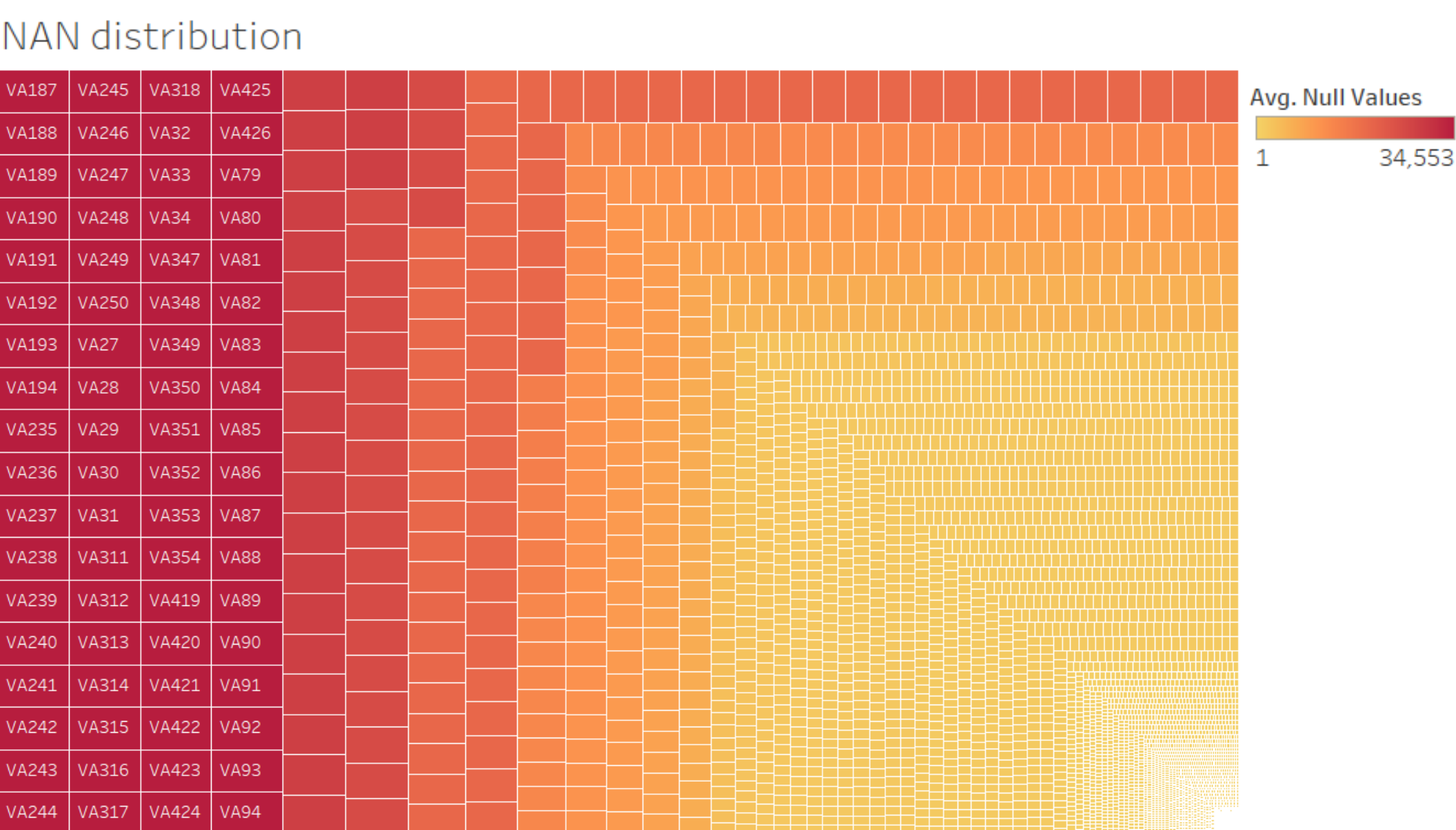
X – a lot of features, **NANS**

- NAN - Values

34553 samples, 7672 features

- Features – numeric

- No duplicated value

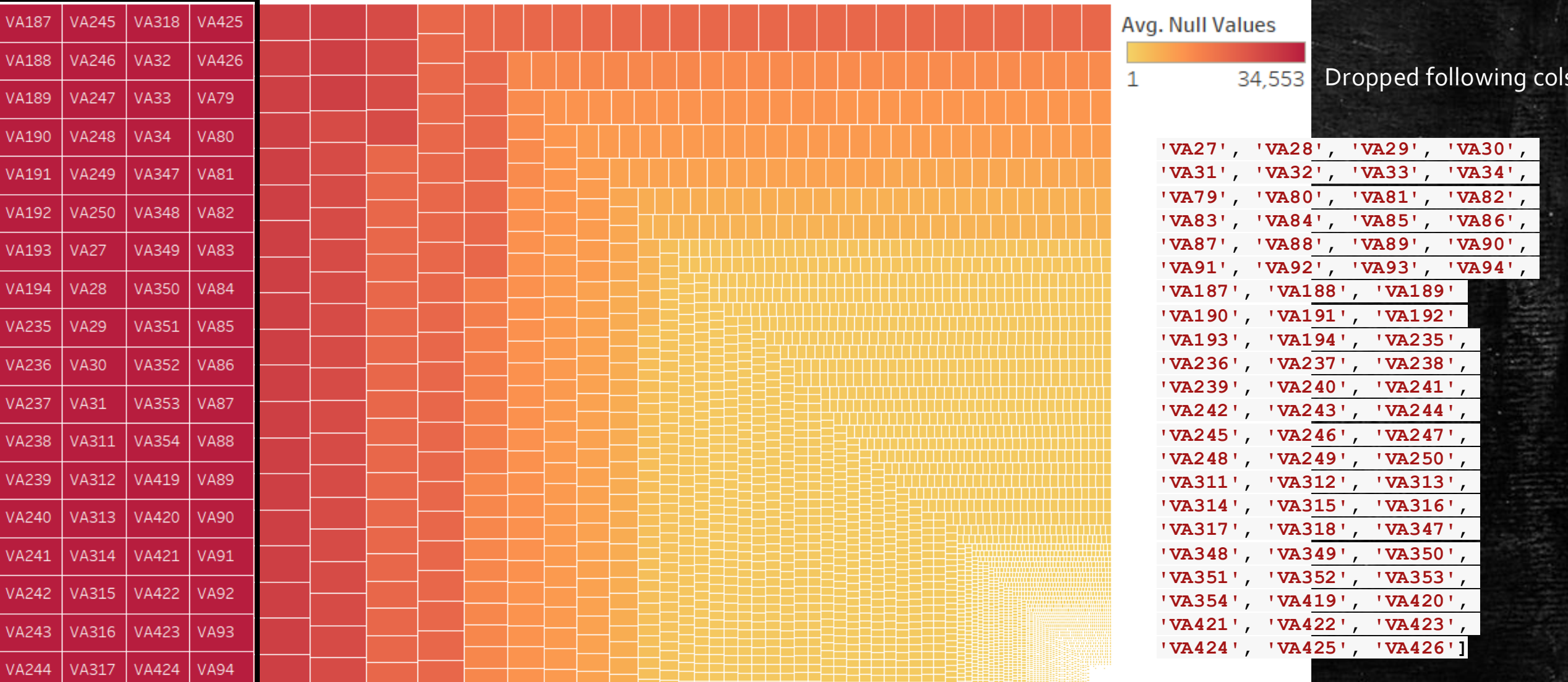


X – a lot of features, **NANS**

- **NAN – Values** – need to handle NANs properly

34553 samples, 7672 features
- Features – numeric
- No duplicated value

NAN distribution



X – a lot of features, NANS, cols with unique value count 1 or 2

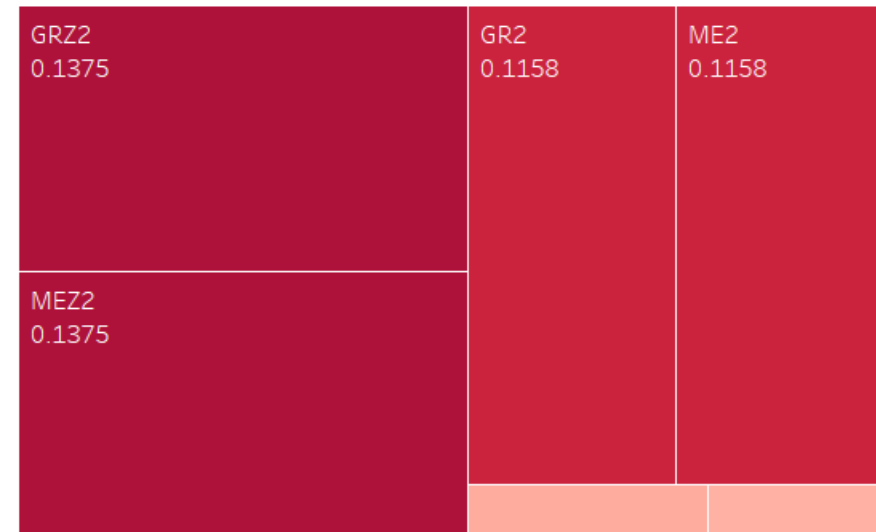
Dropped following cols. They had only one unique value.

Features : 7672 -> 7539

ME2	0.1158	PL2	VA68	VA16
GR2	0.1158	PL27	VA69	VA17
ME314	0.0000	PL28	VA70	VA18
ME313	0.0000	PL29	VA71	VA59
ME312	0.0000	PL30	VA72	VA60
ME311	0.0000	PL31	VA73	VA61
GR352	0.0000	PL32	VA74	VA62
GR351	0.0000	PL51	VA131	VA63
GR348	0.0000	PL52	VA132	VA64
GR347	0.0000	PL53	VA133	VA160
GR34	0.0000	PL54	VA134	VA161
GR33	0.0000	PL369	VA147	VA162
GR32	0.0000	PL370	VA148	VAZ15
GR318	0.0000	PL371	VA149	VAZ131
GR317	0.0000	PL372	VA150	
GR316	0.0000	PL373	VA151	
GR315	0.0000	PL374	VA152	
GR31	0.0000	PL437	VA153	
GR146	0.0000	PL438	VA154	
GR145	0.0000	PL439	VA155	
GR142	0.0000	PL440	VA156	
GR141	0.0000	VA2	VA157	
VAZ18		VA15	VA158	
VAZ16		VAZ132	VA159	
		VAZ13	VA66	
		VA65	VA67	

They had only two unique value. Dropped 82 columns

The features with 2 unique values and their average. - 82 null values (either filled with 0 or 0 and NaN)



X – a lot of features, NANS, cols with unique value count 1 or 2

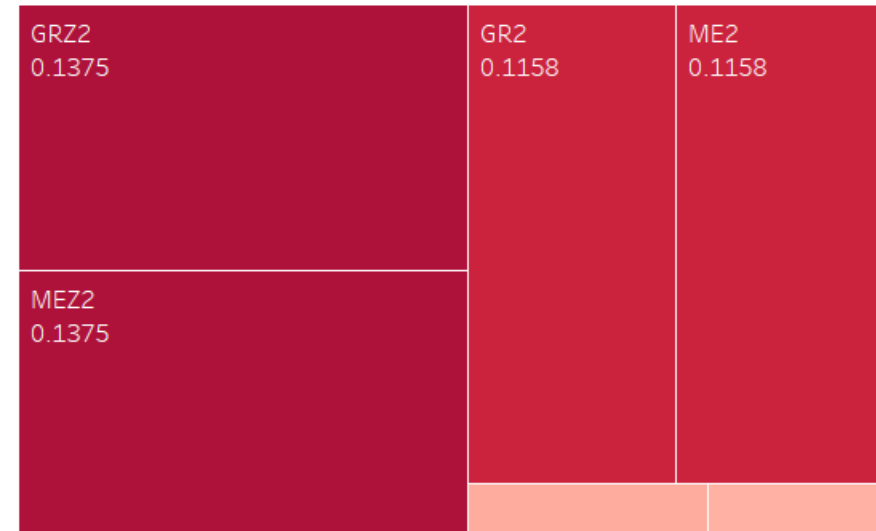
Dropped following cols. They had only one unique value.

Features : 7672 -> 7539

ME2	0.1158	PL2	VA68	VA16
GR2	0.1158	PL27	VA69	VA17
ME314	0.0000	PL28	VA70	VA18
ME313	0.0000	PL29	VA71	VA59
ME312	0.0000	PL30	VA72	VA60
ME311	0.0000	PL31	VA73	VA61
GR352	0.0000	PL32	VA74	VA62
GR351	0.0000	PL51	VA131	VA63
GR348	0.0000	PL52	VA132	VA64
GR347	0.0000	PL53	VA133	VA160
GR34	0.0000	PL54	VA134	VA161
GR33	0.0000	PL369	VA147	VA162
GR32	0.0000	PL370	VA148	VAZ15
GR318	0.0000	PL371	VA149	VAZ131
GR317	0.0000	PL372	VA150	
GR316	0.0000	PL373	VA151	
GR315	0.0000	PL374	VA152	
GR31	0.0000	PL437	VA153	
GR146	0.0000	PL438	VA154	
GR145	0.0000	PL439	VA155	
GR142	0.0000	PL440	VA156	
GR141	0.0000	VA2	VA157	
VAZ18		VA15	VA158	
VAZ16		VAZ132	VA159	
		VAZ13	VA66	
		VA65	VA67	

They had only two unique value. Dropped 82 columns

The features with 2 unique values and their average. - 82 null values (either filled with 0 or 0 and NaN)

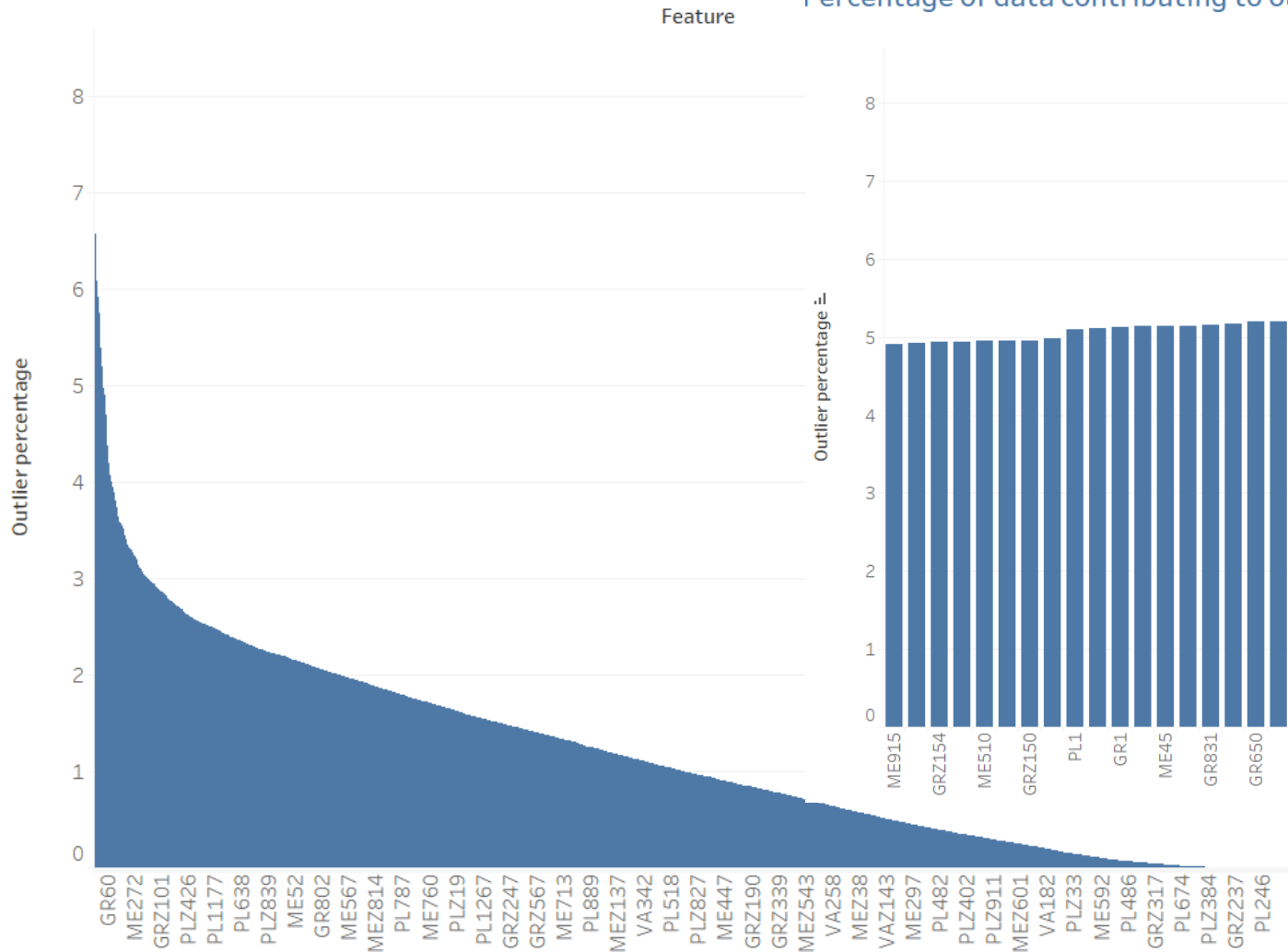


X – a lot of features, NANS, cols with unique value , **Outliers**

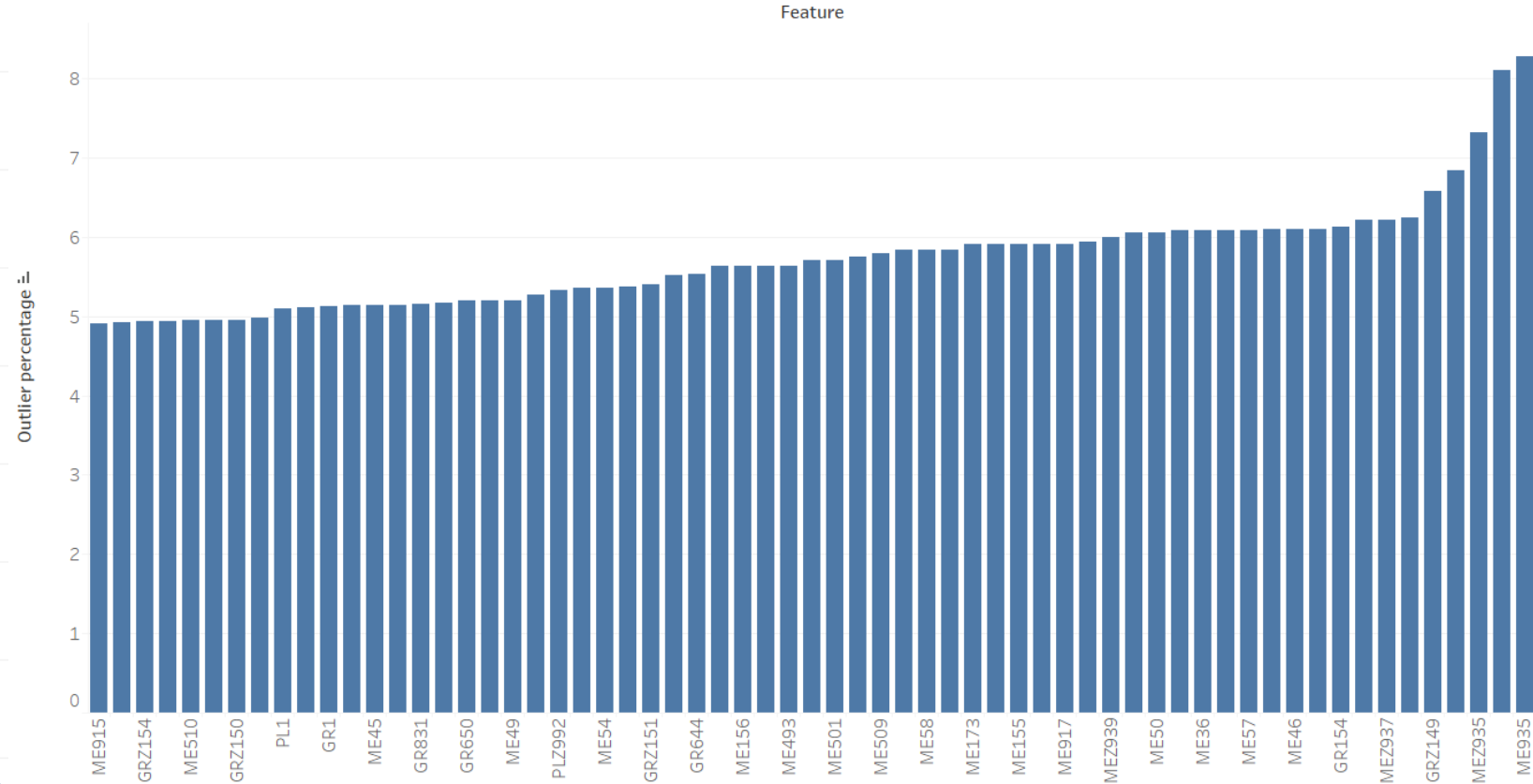
Features : 7672 -> 7539

- Outlier – Values (3 sigma away)

Percentage of data contributing to outliers (3 sigma away)



Percentage of data contributing to outliers (3 sigma away)

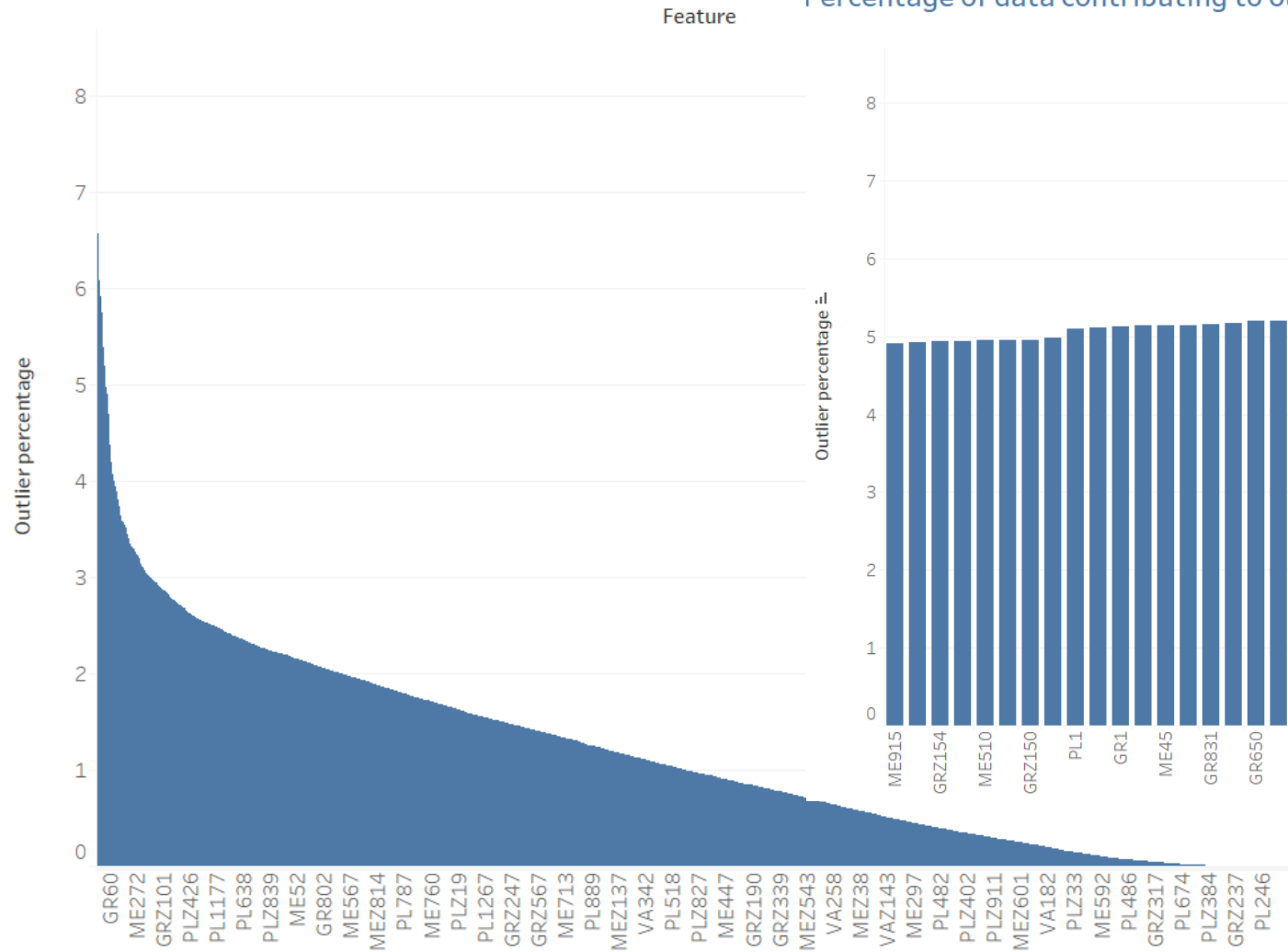


X – a lot of features, NANS, cols with unique value , **Outliers**

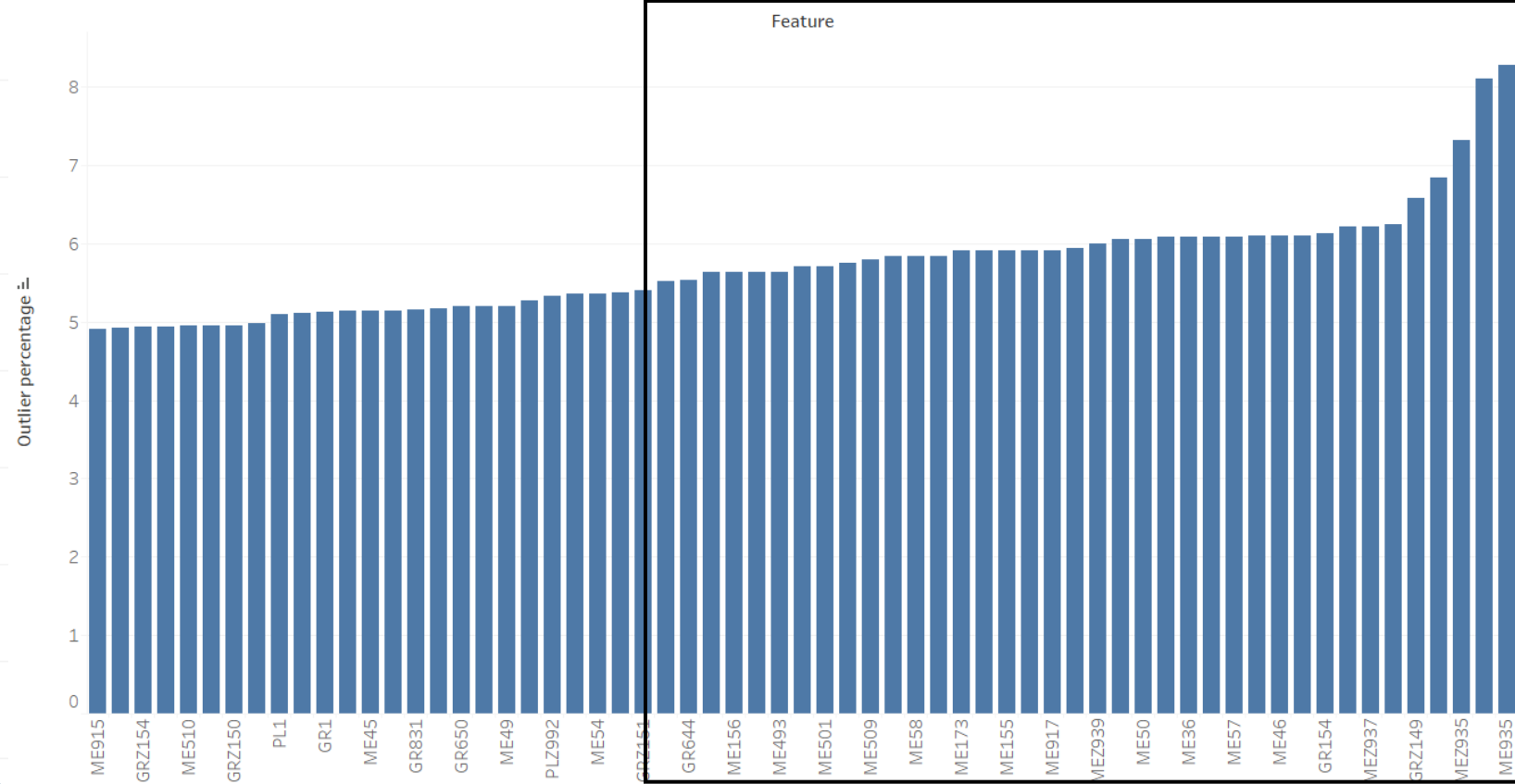
Features : 7672 -> 7539

- Outlier – Values (3 sigma away)

Percentage of data contributing to outliers (3 sigma away)



Percentage of data contributing to outliers (3 sigma away)



X, y- split using stratification

X_train, y_train

X_test, y_test

80% : 20% stratified split

P12
P122
P125
P129
P130
P131
P132
P133
P135
P136
P137
P138
P139
P140
P141
P142
P143
P144
P145
P146
P147
P148
P149
P150
P151
P152
P153
P154
P155
P156
P157
P158
P159
P160
P161
P162
P163
P164
P165
P166
P167
P168
P169
P170
P171
P172
P173
P174
P175
P176
P177
P178
P179
P180
P181
P182
P183
P184
P185
P186
P187
P188
P189
P190
P191
P192
P193
P194
P195
P196
P197
P198
P199
P200
P201
P202
P203
P204
P205
P206
P207
P208
P209
P210
P211
P212
P213
P214
P215
P216
P217
P218
P219
P220
P221
P222
P223
P224
P225
P226
P227
P228
P229
P230
P231
P232
P233
P234
P235
P236
P237
P238
P239
P240
P241
P242
P243
P244
P245
P246
P247
P248
P249
P250
P251
P252
P253
P254
P255
P256
P257
P258
P259
P260
P261
P262
P263
P264
P265
P266
P267
P268
P269
P270
P271
P272
P273
P274
P275
P276
P277
P278
P279
P280
P281
P282
P283
P284
P285
P286
P287
P288
P289
P290
P291
P292
P293
P294
P295
P296
P297
P298
P299
P300
P301
P302
P303
P304
P305
P306
P307
P308
P309
P310
P311
P312
P313
P314
P315
P316
P317
P318
P319
P320
P321
P322
P323
P324
P325
P326
P327
P328
P329
P330
P331
P332
P333
P334
P335
P336
P337
P338
P339
P340
P341
P342
P343
P344
P345
P346
P347
P348
P349
P350
P351
P352
P353
P354
P355
P356
P357
P358
P359
P360
P361
P362
P363
P364
P365
P366
P367
P368
P369
P370
P371
P372
P373
P374
P375
P376
P377
P378
P379
P380
P381
P382
P383
P384
P385
P386
P387
P388
P389
P390
P391
P392
P393
P394
P395
P396
P397
P398
P399
P400
P401
P402
P403
P404
P405
P406
P407
P408
P409
P410
P411
P412
P413
P414
P415
P416
P417
P418
P419
P420
P421
P422
P423
P424
P425
P426
P427
P428
P429
P430
P431
P432
P433
P434
P435
P436
P437
P438
P439
P440
P441
P442
P443
P444
P445
P446
P447
P448
P449
P450
P451
P452
P453
P454
P455
P456
P457
P458
P459
P460
P461
P462
P463
P464
P465
P466
P467
P468
P469
P470
P471
P472
P473
P474
P475
P476
P477
P478
P479
P480
P481
P482
P483
P484
P485
P486
P487
P488
P489
P490
P491
P492
P493
P494
P495
P496
P497
P498
P499
P500
P501
P502
P503
P504
P505
P506
P507
P508
P509
P510
P511
P512
P513
P514
P515
P516
P517
P518
P519
P520
P521
P522
P523
P524
P525
P526
P527
P528
P529
P530
P531
P532
P533
P534
P535
P536
P537
P538
P539
P540
P541
P542
P543
P544
P545
P546
P547
P548
P549
P550
P551
P552
P553
P554
P555
P556
P557
P558
P559
P560
P561
P562
P563
P564
P565
P566
P567
P568
P569
P570
P571
P572
P573
P574
P575
P576
P577
P578
P579
P580
P581
P582
P583
P584
P585
P586
P587
P588
P589
P590
P591
P592
P593
P594
P595
P596
P597
P598
P599
P600
P601
P602
P603
P604
P605
P606
P607
P608
P609
P610
P611
P612
P613
P614
P615
P616
P617
P618
P619
P620
P621
P622
P623
P624
P625
P626
P627
P628
P629
P630
P631
P632
P633
P634
P635
P636
P637
P638
P639
P640
P641
P642
P643
P644
P645
P646
P647
P648
P649
P650
P651
P652
P653
P654
P655
P656
P657
P658
P659
P660
P661
P662
P663
P664
P665
P666
P667
P668
P669
P670
P671
P672
P673
P674
P675
P676
P677
P678
P679
P680
P681
P682
P683
P684
P685
P686
P687
P688
P689
P690
P691
P692
P693
P694
P695
P696
P697
P698
P699
P700
P701
P702
P703
P704
P705
P706
P707
P708
P709
P710
P711
P712
P713
P714
P715
P716
P717
P718
P719
P720
P721
P722
P723
P724
P725
P726
P727
P728
P729
P730
P731
P732
P733
P734
P735
P736
P737
P738
P739
P740
P741
P742
P743
P744
P745
P746
P747
P748
P749
P750
P751
P752
P753
P754
P755
P756
P757
P758
P759
P760
P761
P762
P763
P764
P765
P766
P767
P768
P769
P770
P771
P772
P773
P774
P775
P776
P777
P778
P779
P780
P781
P782
P783
P784
P785
P786
P787
P788
P789
P790
P791
P792
P793
P794
P795
P796
P797
P798
P799
P800
P801
P802
P803
P804
P805
P806
P807
P808
P809
P810
P811
P812
P813
P814
P815
P816
P817
P818
P819
P820
P821
P822
P823
P824
P825
P826
P827
P828
P829
P830
P831
P832
P833
P834
P835
P836
P837
P838
P839
P840
P841
P842
P843
P844
P845
P846
P847
P848
P849
P850
P851
P852
P853
P854
P855
P856
P857
P858
P859
P860
P861
P862
P863
P864
P865
P866
P867
P868
P869
P870
P871
P872
P873
P874
P875
P876
P877
P878
P879
P880
P881
P882
P883
P884
P885
P886
P887
P888
P889
P890
P891
P892
P893
P894
P895
P896
P897
P898
P899
P900
P901
P902
P903
P904
P905
P906
P907
P908
P909
P910
P911
P912
P913
P914
P915
P916
P917
P918
P919
P920
P921
P922
P923
P924
P925
P926
P927
P928
P929
P930
P931
P932
P933
P934
P935
P936
P937
P938
P939
P940
P941
P942
P943
P944
P945
P946
P947
P948
P949
P950
P951
P952
P953
P954
P955
P956
P957
P958
P959
P960
P961
P962
P963
P964
P965
P966
P967
P968
P969
P970
P971
P972
P973
P974
P975
P976
P977
P978
P979
P980
P981
P982
P983
P984
P985
P986
P987
P988
P989
P990
P991
P992
P993
P994
P995
P996
P997
P998
P999
P1000

X – a lot of features, **NANS**

X : Imputaton

Median

Knn (nearest neighbor)

Filled with zero

P12
P122
P122
P120
P120
P131
P132
P131
P131
P133
P134
P1372
P1373
P1374
P1437
P1438
P1439
V135
V137
V137
V138
V139
V140
V141
V142
V143
V144
V145
V146
V147
V148
V149
V150
V151
V152
V153
V154
V155
V156
V157
V158
V159
V160
V161
V162
V163
V164
V165
V166
V167
V168
V169
V170
V171
V172
V173
V174
V175
V176
V177
V178
V179
V180
V181
V182
V183
V184
V185
V186
V187
V188
V189
V190
V191
V192
V193
V194
V195
V196
V197
V198
V199
V200
V201
V202
V203
V204
V205
V206
V207
V208
V209
V210
V211
V212
V213
V214
V215
V216
V217
V218
V219
V220
V221
V222
V223
V224
V225
V226
V227
V228
V229
V230
V231
V232
V233
V234
V235
V236
V237
V238
V239
V240
V241
V242
V243
V244
V245
V246
V247
V248
V249
V250
V251
V252
V253
V254
V255
V256
V257
V258
V259
V260
V261
V262
V263
V264
V265
V266
V267
V268
V269
V270
V271
V272
V273
V274
V275
V276
V277
V278
V279
V280
V281
V282
V283
V284
V285
V286
V287
V288
V289
V290
V291
V292
V293
V294
V295
V296
V297
V298
V299
V300
V301
V302
V303
V304
V305
V306
V307
V308
V309
V310
V311
V312
V313
V314
V315
V316
V317
V318
V319
V320
V321
V322
V323
V324
V325
V326
V327
V328
V329
V330
V331
V332
V333
V334
V335
V336
V337
V338
V339
V340
V341
V342
V343
V344
V345
V346
V347
V348
V349
V350
V351
V352
V353
V354
V355
V356
V357
V358
V359
V360
V361
V362
V363
V364
V365
V366
V367
V368
V369
V370
V371
V372
V373
V374
V375
V376
V377
V378
V379
V380
V381
V382
V383
V384
V385
V386
V387
V388
V389
V390
V391
V392
V393
V394
V395
V396
V397
V398
V399
V400
V401
V402
V403
V404
V405
V406
V407
V408
V409
V410
V411
V412
V413
V414
V415
V416
V417
V418
V419
V420
V421
V422
V423
V424
V425
V426
V427
V428
V429
V430
V431
V432
V433
V434
V435
V436
V437
V438
V439
V440
V441
V442
V443
V444
V445
V446
V447
V448
V449
V450
V451
V452
V453
V454
V455
V456
V457
V458
V459
V460
V461
V462
V463
V464
V465
V466
V467
V468
V469
V470
V471
V472
V473
V474
V475
V476
V477
V478
V479
V480
V481
V482
V483
V484
V485
V486
V487
V488
V489
V490
V491
V492
V493
V494
V495
V496
V497
V498
V499
V500
V501
V502
V503
V504
V505
V506
V507
V508
V509
V510
V511
V512
V513
V514
V515
V516
V517
V518
V519
V520
V521
V522
V523
V524
V525
V526
V527
V528
V529
V530
V531
V532
V533
V534
V535
V536
V537
V538
V539
V540
V541
V542
V543
V544
V545
V546
V547
V548
V549
V550
V551
V552
V553
V554
V555
V556
V557
V558
V559
V560
V561
V562
V563
V564
V565
V566
V567
V568
V569
V570
V571
V572
V573
V574
V575
V576
V577
V578
V579
V580
V581
V582
V583
V584
V585
V586
V587
V588
V589
V590
V591
V592
V593
V594
V595
V596
V597
V598
V599
V600
V601
V602
V603
V604
V605
V606
V607
V608
V609
V610
V611
V612
V613
V614
V615
V616
V617
V618
V619
V620
V621
V622
V623
V624
V625
V626
V627
V628
V629
V630
V631
V632
V633
V634
V635
V636
V637
V638
V639
V640
V641
V642
V643
V644
V645
V646
V647
V648
V649
V650
V651
V652
V653
V654
V655
V656
V657
V658
V659
V660
V661
V662
V663
V664
V665
V666
V667
V668
V669
V670
V671
V672
V673
V674
V675
V676
V677
V678
V679
V680
V681
V682
V683
V684
V685
V686
V687
V688
V689
V690
V691
V692
V693
V694
V695
V696
V697
V698
V699
V700
V701
V702
V703
V704
V705
V706
V707
V708
V709
V710
V711
V712
V713
V714
V715
V716
V717
V718
V719
V720
V721
V722
V723
V724
V725
V726
V727
V728
V729
V730
V731
V732
V733
V734
V735
V736
V737
V738
V739
V740
V741
V742
V743
V744
V745
V746
V747
V748
V749
V750
V751
V752
V753
V754
V755
V756
V757
V758
V759
V760
V761
V762
V763
V764
V765
V766
V767
V768
V769
V770
V771
V772
V773
V774
V775
V776
V777
V778
V779
V780
V781
V782
V783
V784
V785
V786
V787
V788
V789
V790
V791
V792
V793
V794
V795
V796
V797
V798
V799
V800
V801
V802
V803
V804
V805
V806
V807
V808
V809
V810
V811
V812
V813
V814
V815
V816
V817
V818
V819
V820
V821
V822
V823
V824
V825
V826
V827
V828
V829
V830
V831
V832
V833
V834
V835
V836
V837
V838
V839
V840
V841
V842
V843
V844
V845
V846
V847
V848
V849
V850
V851
V852
V853
V854
V855
V856
V857
V858
V859
V860
V861
V862
V863
V864
V865
V866
V867
V868
V869
V870
V871
V872
V873
V874
V875
V876
V877
V878
V879
V880
V881
V882
V883
V884
V885
V886
V887
V888
V889
V890
V891
V892
V893
V894
V895
V896
V897
V898
V899
V900
V901
V902
V903
V904
V905
V906
V907
V908
V909
V910
V911
V912
V913
V914
V915
V916
V917
V918
V919
V920
V921
V922
V923
V924
V925
V926
V927
V928
V929
V930
V931
V932
V933
V934
V935
V936
V937
V938
V939
V940
V941
V942
V943
V944
V945
V946
V947
V948
V949
V950
V951
V952
V953
V954
V955
V956
V957
V958
V959
V960
V961
V962
V963
V964
V965
V966
V967
V968
V969
V970
V971
V972
V973
V974
V975
V976
V977
V978
V979
V980
V981
V982
V983
V984
V985
V986
V987
V988
V989
V990
V991
V992
V993
V994
V995
V996
V997
V998
V999
V1000

X – a lot of features, NANS, **Scaling**

X -> Imputed -> Scaled

Robust Scaling

MinMax Scaler

Standard Scaler

P10
P122
P125
P126
P130
P131
P132
P133
P134
P135
P136
P137
P138
P139
P140
P141
P142
P143
P144
P145
P146
P147
P148
P149
P150
P151
P152
P153
P154
P155
P156
P157
P158
P159
P160
P161
P162
P163
P164
P165
P166
P167
P168
P169
P170
P171
P172
P173
P174
P175
P176
P177
P178
P179
P180
P181
P182
P183
P184
P185
P186
P187
P188
P189
P190
P191
P192
P193
P194
P195
P196
P197
P198
P199
P200
P201
P202
P203
P204
P205
P206
P207
P208
P209
P210
P211
P212
P213
P214
P215
P216
P217
P218
P219
P220
P221
P222
P223
P224
P225
P226
P227
P228
P229
P230
P231
P232
P233
P234
P235
P236
P237
P238
P239
P240
P241
P242
P243
P244
P245
P246
P247
P248
P249
P250
P251
P252
P253
P254
P255
P256
P257
P258
P259
P260
P261
P262
P263
P264
P265
P266
P267
P268
P269
P270
P271
P272
P273
P274
P275
P276
P277
P278
P279
P280
P281
P282
P283
P284
P285
P286
P287
P288
P289
P290
P291
P292
P293
P294
P295
P296
P297
P298
P299
P300
P301
P302
P303
P304
P305
P306
P307
P308
P309
P310
P311
P312
P313
P314
P315
P316
P317
P318
P319
P320
P321
P322
P323
P324
P325
P326
P327
P328
P329
P330
P331
P332
P333
P334
P335
P336
P337
P338
P339
P340
P341
P342
P343
P344
P345
P346
P347
P348
P349
P350
P351
P352
P353
P354
P355
P356
P357
P358
P359
P360
P361
P362
P363
P364
P365
P366
P367
P368
P369
P370
P371
P372
P373
P374
P375
P376
P377
P378
P379
P380
P381
P382
P383
P384
P385
P386
P387
P388
P389
P390
P391
P392
P393
P394
P395
P396
P397
P398
P399
P400
P401
P402
P403
P404
P405
P406
P407
P408
P409
P410
P411
P412
P413
P414
P415
P416
P417
P418
P419
P420
P421
P422
P423
P424
P425
P426
P427
P428
P429
P430
P431
P432
P433
P434
P435
P436
P437
P438
P439
P440
P441
P442
P443
P444
P445
P446
P447
P448
P449
P450
P451
P452
P453
P454
P455
P456
P457
P458
P459
P460
P461
P462
P463
P464
P465
P466
P467
P468
P469
P470
P471
P472
P473
P474
P475
P476
P477
P478
P479
P480
P481
P482
P483
P484
P485
P486
P487
P488
P489
P490
P491
P492
P493
P494
P495
P496
P497
P498
P499
P500
P501
P502
P503
P504
P505
P506
P507
P508
P509
P510
P511
P512
P513
P514
P515
P516
P517
P518
P519
P520
P521
P522
P523
P524
P525
P526
P527
P528
P529
P530
P531
P532
P533
P534
P535
P536
P537
P538
P539
P540
P541
P542
P543
P544
P545
P546
P547
P548
P549
P550
P551
P552
P553
P554
P555
P556
P557
P558
P559
P560
P561
P562
P563
P564
P565
P566
P567
P568
P569
P570
P571
P572
P573
P574
P575
P576
P577
P578
P579
P580
P581
P582
P583
P584
P585
P586
P587
P588
P589
P590
P591
P592
P593
P594
P595
P596
P597
P598
P599
P600
P601
P602
P603
P604
P605
P606
P607
P608
P609
P610
P611
P612
P613
P614
P615
P616
P617
P618
P619
P620
P621
P622
P623
P624
P625
P626
P627
P628
P629
P630
P631
P632
P633
P634
P635
P636
P637
P638
P639
P640
P641
P642
P643
P644
P645
P646
P647
P648
P649
P650
P651
P652
P653
P654
P655
P656
P657
P658
P659
P660
P661
P662
P663
P664
P665
P666
P667
P668
P669
P670
P671
P672
P673
P674
P675
P676
P677
P678
P679
P680
P681
P682
P683
P684
P685
P686
P687
P688
P689
P690
P691
P692
P693
P694
P695
P696
P697
P698
P699
P700
P701
P702
P703
P704
P705
P706
P707
P708
P709
P710
P711
P712
P713
P714
P715
P716
P717
P718
P719
P720
P721
P722
P723
P724
P725
P726
P727
P728
P729
P730
P731
P732
P733
P734
P735
P736
P737
P738
P739
P740
P741
P742
P743
P744
P745
P746
P747
P748
P749
P750
P751
P752
P753
P754
P755
P756
P757
P758
P759
P760
P761
P762
P763
P764
P765
P766
P767
P768
P769
P770
P771
P772
P773
P774
P775
P776
P777
P778
P779
P780
P781
P782
P783
P784
P785
P786
P787
P788
P789
P790
P791
P792
P793
P794
P795
P796
P797
P798
P799
P800
P801
P802
P803
P804
P805
P806
P807
P808
P809
P810
P811
P812
P813
P814
P815
P816
P817
P818
P819
P820
P821
P822
P823
P824
P825
P826
P827
P828
P829
P830
P831
P832
P833
P834
P835
P836
P837
P838
P839
P840
P841
P842
P843
P844
P845
P846
P847
P848
P849
P850
P851
P852
P853
P854
P855
P856
P857
P858
P859
P860
P861
P862
P863
P864
P865
P866
P867
P868
P869
P870
P871
P872
P873
P874
P875
P876
P877
P878
P879
P880
P881
P882
P883
P884
P885
P886
P887
P888
P889
P890
P891
P892
P893
P894
P895
P896
P897
P898
P899
P900
P901
P902
P903
P904
P905
P906
P907
P908
P909
P910
P911
P912
P913
P914
P915
P916
P917
P918
P919
P920
P921
P922
P923
P924
P925
P926
P927
P928
P929
P930
P931
P932
P933
P934
P935
P936
P937
P938
P939
P940
P941
P942
P943
P944
P945
P946
P947
P948
P949
P950
P951
P952
P953
P954
P955
P956
P957
P958
P959
P960
P961
P962
P963
P964
P965
P966
P967
P968
P969
P970
P971
P972
P973
P974
P975
P976
P977
P978
P979
P980
P981
P982
P983
P984
P985
P986
P987
P988
P989
P990
P991
P992
P993
P994
P995
P996
P997
P998
P999
P1000

X – a lot of features, NANS, Scaling

```
50 def pre_process(df_train, imputation, scaling, df_test):
51     # Impute missing values
52     if imputation == 'median':
53         imputer = SimpleImputer(strategy=imputation)
54         df_train_imputed = pd.DataFrame(imputer.fit_transform(df_train), columns=df_train.columns)
55         df_test_imputed = pd.DataFrame(imputer.transform(df_test), columns=df_train.columns)
56         print(imputation)
57     elif imputation == 'zero':
58         print(imputation)
59         df_train_imputed = df_train.fillna(0)
60         df_test_imputed = df_test.fillna(0)
61
62     elif imputation == 'knn':
63         print(imputation)
64         imputer = KNNImputer(n_neighbor = 5)
65         df_train_imputed = pd.DataFrame(imputer.fit_transform(df_train), columns=df_train.columns)
66         df_test_imputed = pd.DataFrame(imputer.transform(df_test), columns=df_train.columns)
67
68     # df_train_imputed = df_train
69     if scaling == 'minmax':
70         scaler = MinMaxScaler()
71         X_t = scaler.fit_transform(df_train_imputed)
72         # df_train_imputed_scaled = pd.DataFrame(scaler.fit_transform(df_train), columns=df_train.
73         # columns)
74         df_test_imputed_scaled = pd.DataFrame(scaler.transform(df_test_imputed), columns=df_train.
75         columns)
76     elif scaling == 'robust':
77         robust = RobustScaler()
78         X_t = robust.fit_transform(df_train_imputed)
79         # df_train_imputed_scaled = pd.DataFrame(robust.fit_transform(df_train), columns=df_train.
80         # columns)
81         df_test_imputed_scaled = pd.DataFrame(robust.transform(df_test_imputed), columns=df_test.
82         columns)
83     elif scaling == 'std':
84         std = StandardScaler()
85         df_train_imputed_scaled = pd.DataFrame(std.fit_transform(df_train_imputed), columns=df_train.
86         columns)
87         df_test_imputed_scaled = pd.DataFrame(std.transform(df_test_imputed), columns=df_test.columns)
88
89         #del df_train_imputed
90     else:
91         # raise ValueError("Invalid scaling method. Choose 'minmax'.")
92     return df_test_imputed_scaled, df_test_imputed
93
94 strategies = ['zero', 'median', 'knn']#''
95 scalings = ['robust', 'minmax', 'std']#, 'robust', 'standard']
96 |
```

```
94 for strat in strategies:
95     for scaling in scalings:
96         # Pre-process the data
97         print(scaling)
98         print(strat)
99         df_test_imputed_scaled, df_test_imputed = pre_process(df_train = X_train, df_test = X_test,
100         imputation=strat, scaling=scaling)
101     # print(f'shape : {df_train_imputed.shape}')
102     # df_train_imputed_scaled = pre_process(X_train, scaling=scaling)
103     print(f'shape : {df_test_imputed_scaled.shape}')
104
105     # Save the pre-processed data
106     #df_train_imputed.to_csv(f'/mnt/gpfs3_amd/scratch/rgu245/intel/processed_files/X_train_imputed_
107     {imputation}_{scaling}.csv', index=False)
108     #print('file 1 saved')
109     df_test_imputed_scaled.to_csv(f'/mnt/gpfs3_amd/scratch/rgu245/intel/processed_files/
110     X_test_imputed_{strat}_zero_{scaling}.csv', index=False)
111     df_test_imputed.to_csv(f'/mnt/gpfs3_amd/scratch/rgu245/intel/processed_files/X_test_imputed_
112     {strat}.csv', index=False)
113
114     print('file2 saved')
115 exit()
```


X – a lot of features, NANS, Scaling

```
50 def pre_process(df_train, imputation, scaling, df_test):
51     # Impute missing values
52     if imputation == 'median':
53         imputer = SimpleImputer(strategy=imputation)
54         df_train_imputed = pd.DataFrame(imputer.fit_transform(df_train), columns=df_train.columns)
55         df_test_imputed = pd.DataFrame(imputer.transform(df_test), columns=df_train.columns)
56         print(imputation)
57     elif imputation == 'zero':
58         print(imputation)
59         df_train_imputed = df_train.fillna(0)
60         df_test_imputed = df_test.fillna(0)
61
62     elif imputation == 'knn':
63         print(imputation)
64         imputer = KNNImputer(n_neighbor = 5)
65         df_train_imputed = pd.DataFrame(imputer.fit_transform(df_train), columns=df_train.columns)
66         df_test_imputed = pd.DataFrame(imputer.transform(df_test), columns=df_train.columns)
67
68     # df_train_imputed = df_train
69     if scaling == 'minmax':
70         scaler = MinMaxScaler()
71         X_t = scaler.fit_transform(df_train_imputed)
72         # df_train_imputed_scaled = pd.DataFrame(scaler.fit_transform(df_train), columns=df_train.
73         # columns)
74         df_test_imputed_scaled = pd.DataFrame(scaler.transform(df_test_imputed), columns=df_train.
75         columns)
76     elif scaling == 'robust':
77         robust = RobustScaler()
78         X_t = robust.fit_transform(df_train_imputed)
79         # df_train_imputed_scaled = pd.DataFrame(robust.fit_transform(df_train), columns=df_train.
80         # columns)
81         df_test_imputed_scaled = pd.DataFrame(robust.transform(df_test_imputed), columns=df_test.
82         columns)
83     elif scaling == 'std':
84         std = StandardScaler()
85         df_train_imputed_scaled = pd.DataFrame(std.fit_transform(df_train_imputed), columns=df_train.
86         columns)
87         df_test_imputed_scaled = pd.DataFrame(std.transform(df_test_imputed), columns=df_test.columns)
88         #del df_train_imputed
89     else:
90         # raise ValueError("Invalid scaling method. Choose 'minmax'.")
91         return df_test_imputed_scaled, df_test_imputed
92
93 strategies = ['zero', 'median', 'knn']#''
94 scalings = ['robust', 'minmax', 'std']#, 'robust', 'standard'
```

```
94 for strat in strategies:
95     for scaling in scalings:
96         # Pre-process the data
97         print(scaling)
98         print(strat)
99         df_test_imputed_scaled, df_test_imputed = pre_process(df_train = X_train, df_test = X_test,
100         imputation=strat, scaling=scaling)
101     # print(f'shape : {df_train_imputed.shape}')
102     # df_train_imputed_scaled = pre_process(X_train, scaling=scaling)
103     print(f'shape : {df_test_imputed_scaled.shape}')
104
105     # Save the pre-processed data
106     #df_train_imputed.to_csv(f'/mnt/gpfs3_amd/scratch/rgu245/intel/processed_files/X_train_imputed_
107     {imputation}_{scaling}.csv', index=False)
108     #print('file 1 saved')
109     df_test_imputed_scaled.to_csv(f'/mnt/gpfs3_amd/scratch/rgu245/intel/processed_files/
110     X_test_imputed_{strat}_{scaling}.csv', index=False)
111     df_test_imputed.to_csv(f'/mnt/gpfs3_amd/scratch/rgu245/intel/processed_files/X_test_imputed_
112     {strat}.csv', index=False)
113
114     print('file2 saved')
115
116 exit()
```

X – Imputed, Scaled, CURSE OF DIMENSIONALITY

X -> Imputed -> Scaled

Feature Extraction

Feature selection

P12
P122
P125
P126
P130
P131
P132
P133
P134
P135
P136
P137
P138
P139
P140
P141
P142
P143
P144
P145
P146
P147
P148
P149
P150
P151
P152
P153
P154
P155
P156
P157
P158
P159
P160
P161
P162
P163
P164
P165
P166
P167
P168
P169
P170
P171
P172
P173
P174
P175
P176
P177
P178
P179
P180
P181
P182
P183
P184
P185
P186
P187
P188
P189
P190
P191
P192
P193
P194
P195
P196
P197
P198
P199
P200
P201
P202
P203
P204
P205
P206
P207
P208
P209
P210
P211
P212
P213
P214
P215
P216
P217
P218
P219
P220
P221
P222
P223
P224
P225
P226
P227
P228
P229
P230
P231
P232
P233
P234
P235
P236
P237
P238
P239
P240
P241
P242
P243
P244
P245
P246
P247
P248
P249
P250
P251
P252
P253
P254
P255
P256
P257
P258
P259
P260
P261
P262
P263
P264
P265
P266
P267
P268
P269
P270
P271
P272
P273
P274
P275
P276
P277
P278
P279
P280
P281
P282
P283
P284
P285
P286
P287
P288
P289
P290
P291
P292
P293
P294
P295
P296
P297
P298
P299
P300
P301
P302
P303
P304
P305
P306
P307
P308
P309
P310
P311
P312
P313
P314
P315
P316
P317
P318
P319
P320
P321
P322
P323
P324
P325
P326
P327
P328
P329
P330
P331
P332
P333
P334
P335
P336
P337
P338
P339
P340
P341
P342
P343
P344
P345
P346
P347
P348
P349
P350
P351
P352
P353
P354
P355
P356
P357
P358
P359
P360
P361
P362
P363
P364
P365
P366
P367
P368
P369
P370
P371
P372
P373
P374
P375
P376
P377
P378
P379
P380
P381
P382
P383
P384
P385
P386
P387
P388
P389
P390
P391
P392
P393
P394
P395
P396
P397
P398
P399
P400
P401
P402
P403
P404
P405
P406
P407
P408
P409
P410
P411
P412
P413
P414
P415
P416
P417
P418
P419
P420
P421
P422
P423
P424
P425
P426
P427
P428
P429
P430
P431
P432
P433
P434
P435
P436
P437
P438
P439
P440
P441
P442
P443
P444
P445
P446
P447
P448
P449
P450
P451
P452
P453
P454
P455
P456
P457
P458
P459
P460
P461
P462
P463
P464
P465
P466
P467
P468
P469
P470
P471
P472
P473
P474
P475
P476
P477
P478
P479
P480
P481
P482
P483
P484
P485
P486
P487
P488
P489
P490
P491
P492
P493
P494
P495
P496
P497
P498
P499
P500
P501
P502
P503
P504
P505
P506
P507
P508
P509
P510
P511
P512
P513
P514
P515
P516
P517
P518
P519
P520
P521
P522
P523
P524
P525
P526
P527
P528
P529
P530
P531
P532
P533
P534
P535
P536
P537
P538
P539
P540
P541
P542
P543
P544
P545
P546
P547
P548
P549
P550
P551
P552
P553
P554
P555
P556
P557
P558
P559
P560
P561
P562
P563
P564
P565
P566
P567
P568
P569
P570
P571
P572
P573
P574
P575
P576
P577
P578
P579
P580
P581
P582
P583
P584
P585
P586
P587
P588
P589
P590
P591
P592
P593
P594
P595
P596
P597
P598
P599
P600
P601
P602
P603
P604
P605
P606
P607
P608
P609
P610
P611
P612
P613
P614
P615
P616
P617
P618
P619
P620
P621
P622
P623
P624
P625
P626
P627
P628
P629
P630
P631
P632
P633
P634
P635
P636
P637
P638
P639
P640
P641
P642
P643
P644
P645
P646
P647
P648
P649
P650
P651
P652
P653
P654
P655
P656
P657
P658
P659
P660
P661
P662
P663
P664
P665
P666
P667
P668
P669
P670
P671
P672
P673
P674
P675
P676
P677
P678
P679
P680
P681
P682
P683
P684
P685
P686
P687
P688
P689
P690
P691
P692
P693
P694
P695
P696
P697
P698
P699
P700
P701
P702
P703
P704
P705
P706
P707
P708
P709
P710
P711
P712
P713
P714
P715
P716
P717
P718
P719
P720
P721
P722
P723
P724
P725
P726
P727
P728
P729
P730
P731
P732
P733
P734
P735
P736
P737
P738
P739
P740
P741
P742
P743
P744
P745
P746
P747
P748
P749
P750
P751
P752
P753
P754
P755
P756
P757
P758
P759
P760
P761
P762
P763
P764
P765
P766
P767
P768
P769
P770
P771
P772
P773
P774
P775
P776
P777
P778
P779
P780
P781
P782
P783
P784
P785
P786
P787
P788
P789
P790
P791
P792
P793
P794
P795
P796
P797
P798
P799
P800
P801
P802
P803
P804
P805
P806
P807
P808
P809
P810
P811
P812
P813
P814
P815
P816
P817
P818
P819
P820
P821
P822
P823
P824
P825
P826
P827
P828
P829
P830
P831
P832
P833
P834
P835
P836
P837
P838
P839
P840
P841
P842
P843
P844
P845
P846
P847
P848
P849
P850
P851
P852
P853
P854
P855
P856
P857
P858
P859
P860
P861
P862
P863
P864
P865
P866
P867
P868
P869
P870
P871
P872
P873
P874
P875
P876
P877
P878
P879
P880
P881
P882
P883
P884
P885
P886
P887
P888
P889
P890
P891
P892
P893
P894
P895
P896
P897
P898
P899
P900
P901
P902
P903
P904
P905
P906
P907
P908
P909
P910
P911
P912
P913
P914
P915
P916
P917
P918
P919
P920
P921
P922
P923
P924
P925
P926
P927
P928
P929
P930
P931
P932
P933
P934
P935
P936
P937
P938
P939
P940
P941
P942
P943
P944
P945
P946
P947
P948
P949
P950
P951
P952
P953
P954
P955
P956
P957
P958
P959
P960
P961
P962
P963
P964
P965
P966
P967
P968
P969
P970
P971
P972
P973
P974
P975
P976
P977
P978
P979
P980
P981
P982
P983
P984
P985
P986
P987
P988
P989
P990
P991
P992
P993
P994
P995
P996
P997
P998
P999
P1000

X – Imputed, Scaled, CURSE OF DIMENSIONALITY

X -> Imputed -> Scaled

Feature Extraction

Variance Threshold

Principal Component Analysis
(using different nComponents)

Kernel PCA

PCA : N Components = 876, 1500, 3000
kPCA : 875, 500

P10
P122
P125
P126
P130
P131
P132
P133
P134
P135
P136
P137
P138
P139
P140
P141
P142
P143
P144
P145
P146
P147
P148
P149
P150
P151
P152
P153
P154
P155
P156
P157
P158
P159
P160
P161
P162
P163
P164
P165
P166
P167
P168
P169
P170
P171
P172
P173
P174
P175
P176
P177
P178
P179
P180
P181
P182
P183
P184
P185
P186
P187
P188
P189
P190
P191
P192
P193
P194
P195
P196
P197
P198
P199
P200
P201
P202
P203
P204
P205
P206
P207
P208
P209
P210
P211
P212
P213
P214
P215
P216
P217
P218
P219
P220
P221
P222
P223
P224
P225
P226
P227
P228
P229
P230
P231
P232
P233
P234
P235
P236
P237
P238
P239
P240
P241
P242
P243
P244
P245
P246
P247
P248
P249
P250
P251
P252
P253
P254
P255
P256
P257
P258
P259
P260
P261
P262
P263
P264
P265
P266
P267
P268
P269
P270
P271
P272
P273
P274
P275
P276
P277
P278
P279
P280
P281
P282
P283
P284
P285
P286
P287
P288
P289
P290
P291
P292
P293
P294
P295
P296
P297
P298
P299
P300
P301
P302
P303
P304
P305
P306
P307
P308
P309
P310
P311
P312
P313
P314
P315
P316
P317
P318
P319
P320
P321
P322
P323
P324
P325
P326
P327
P328
P329
P330
P331
P332
P333
P334
P335
P336
P337
P338
P339
P340
P341
P342
P343
P344
P345
P346
P347
P348
P349
P350
P351
P352
P353
P354
P355
P356
P357
P358
P359
P360
P361
P362
P363
P364
P365
P366
P367
P368
P369
P370
P371
P372
P373
P374
P375
P376
P377
P378
P379
P380
P381
P382
P383
P384
P385
P386
P387
P388
P389
P390
P391
P392
P393
P394
P395
P396
P397
P398
P399
P400
P401
P402
P403
P404
P405
P406
P407
P408
P409
P410
P411
P412
P413
P414
P415
P416
P417
P418
P419
P420
P421
P422
P423
P424
P425
P426
P427
P428
P429
P430
P431
P432
P433
P434
P435
P436
P437
P438
P439
P440
P441
P442
P443
P444
P445
P446
P447
P448
P449
P450
P451
P452
P453
P454
P455
P456
P457
P458
P459
P460
P461
P462
P463
P464
P465
P466
P467
P468
P469
P470
P471
P472
P473
P474
P475
P476
P477
P478
P479
P480
P481
P482
P483
P484
P485
P486
P487
P488
P489
P490
P491
P492
P493
P494
P495
P496
P497
P498
P499
P500
P501
P502
P503
P504
P505
P506
P507
P508
P509
P510
P511
P512
P513
P514
P515
P516
P517
P518
P519
P520
P521
P522
P523
P524
P525
P526
P527
P528
P529
P530
P531
P532
P533
P534
P535
P536
P537
P538
P539
P540
P541
P542
P543
P544
P545
P546
P547
P548
P549
P550
P551
P552
P553
P554
P555
P556
P557
P558
P559
P560
P561
P562
P563
P564
P565
P566
P567
P568
P569
P570
P571
P572
P573
P574
P575
P576
P577
P578
P579
P580
P581
P582
P583
P584
P585
P586
P587
P588
P589
P590
P591
P592
P593
P594
P595
P596
P597
P598
P599
P600
P601
P602
P603
P604
P605
P606
P607
P608
P609
P610
P611
P612
P613
P614
P615
P616
P617
P618
P619
P620
P621
P622
P623
P624
P625
P626
P627
P628
P629
P630
P631
P632
P633
P634
P635
P636
P637
P638
P639
P640
P641
P642
P643
P644
P645
P646
P647
P648
P649
P650
P651
P652
P653
P654
P655
P656
P657
P658
P659
P660
P661
P662
P663
P664
P665
P666
P667
P668
P669
P670
P671
P672
P673
P674
P675
P676
P677
P678
P679
P680
P681
P682
P683
P684
P685
P686
P687
P688
P689
P690
P691
P692
P693
P694
P695
P696
P697
P698
P699
P700
P701
P702
P703
P704
P705
P706
P707
P708
P709
P710
P711
P712
P713
P714
P715
P716
P717
P718
P719
P720
P721
P722
P723
P724
P725
P726
P727
P728
P729
P730
P731
P732
P733
P734
P735
P736
P737
P738
P739
P740
P741
P742
P743
P744
P745
P746
P747
P748
P749
P750
P751
P752
P753
P754
P755
P756
P757
P758
P759
P760
P761
P762
P763
P764
P765
P766
P767
P768
P769
P770
P771
P772
P773
P774
P775
P776
P777
P778
P779
P780
P781
P782
P783
P784
P785
P786
P787
P788
P789
P790
P791
P792
P793
P794
P795
P796
P797
P798
P799
P800
P801
P802
P803
P804
P805
P806
P807
P808
P809
P810
P811
P812
P813
P814
P815
P816
P817
P818
P819
P820
P821
P822
P823
P824
P825
P826
P827
P828
P829
P830
P831
P832
P833
P834
P835
P836
P837
P838
P839
P840
P841
P842
P843
P844
P845
P846
P847
P848
P849
P850
P851
P852
P853
P854
P855
P856
P857
P858
P859
P860
P861
P862
P863
P864
P865
P866
P867
P868
P869
P870
P871
P872
P873
P874
P875
P876
P877
P878
P879
P880
P881
P882
P883
P884
P885
P886
P887
P888
P889
P890
P891
P892
P893
P894
P895
P896
P897
P898
P899
P900
P901
P902
P903
P904
P905
P906
P907
P908
P909
P910
P911
P912
P913
P914
P915
P916
P917
P918
P919
P920
P921
P922
P923
P924
P925
P926
P927
P928
P929
P930
P931
P932
P933
P934
P935
P936
P937
P938
P939
P940
P941
P942
P943
P944
P945
P946
P947
P948
P949
P950
P951
P952
P953
P954
P955
P956
P957
P958
P959
P960
P961
P962
P963
P964
P965
P966
P967
P968
P969
P970
P971
P972
P973
P974
P975
P976
P977
P978
P979
P980
P981
P982
P983
P984
P985
P986
P987
P988
P989
P990
P991
P992
P993
P994
P995
P996
P997
P998
P999
P1000

X – Imputed, Scaled, CURSE OF DIMENSIONALITY

```

59 def dim_reduction(df_train, df_test, dimred_proc, nComp ):
60     if dimred_proc == 'pca':
61         pca_init = PCA(n_components = nComp)
62         x_pca = pca_init.fit_transform(df_train)
63         x_test = pca_init.transform(df_test)
64         # pk.dump(x_pca, open(f"/mnt/gpfs3_amd/scratch/rgu245/intel/processed_files/"))
65         x_pca_df = pd.DataFrame(x_test, columns=[f'PC{i+1}' for i in range(nComp)])
66
67         return x_pca_df
68 nComponents = [800, 1500, 3000]
69 for comp in nComponents:
70     print(comp)
71     x_pca = dim_reduction(X_train_imputed_scaled, X_test, 'pca', comp)
72     x_pca.to_csv(f'/mnt/gpfs3_amd/scratch/rgu245/intel/processed_files/X_test_in
73     print('file 1 saved')
74     del x_pca
75 print('done')
76 #import pickle
77 #print('loading.pkl')
78 ## Load the pickled object from the .pkl file
79 #with open('/mnt/gpfs3_amd/scratch/rgu245/intel/processed_files/pca_800.pkl', 'rb')
80 #    pca = pickle.load(f)
81 #
82 #print('loaded achar')# Run the loaded object
83 #result = pca.transform(X_test)
84 #print('result ')
85 #X_test_pca800 = pd.DataFrame(result, columns=[f'PC{i+1}' for i in range(800)])

```

```

7
8 def dim_reduction(df_train, dimred_proc, nComp ):
9     if dimred_proc == 'kpca':
10         kpca = KernelPCA(n_components=nComp, kernel='rbf', gamma = 0.00001)
11
12     # Fit and transform the data
13     X_kpca = kpca.fit_transform(df_train)
14
15     pk.dump(X_kpca, open(f"/mnt/gpfs3_amd/scratch/rgu245/intel/processed_files/kpca.pkl", 'wb'))
16     x_pca_df = pd.DataFrame(X_kpca, columns=[f'PC{i+1}' for i in range(nComp)])
17
18     return x_pca_df
19
20 nComponents = [500,875]
21 for comp in nComponents:
22     print(comp)
23     x_pca = dim_reduction(X_train_imputed_scaled, 'kpca', comp)
24     x_pca.to_csv(f"/mnt/gpfs3_amd/scratch/rgu245/intel/processed_files/X_train_{comp}.csv")
25     print('file 1 saved')
26     del x_pca
27 print('done')
28 exit(0)

```


X – Imputed, Scaled, **Train Model**

X -> Imputed -> Scaled

Feature Extraction

Train different models!

P12
P122
P125
P126
P130
P131
P132
P133
P134
P135
P136
P137
P138
P139
P140
P141
P142
P143
P144
P145
P146
P147
P148
P149
P150
P151
P152
P153
P154
P155
P156
P157
P158
P159
P160
P161
P162
P163
P164
P165
P166
P167
P168
P169
P170
P171
P172
P173
P174
P175
P176
P177
P178
P179
P180
P181
P182
P183
P184
P185
P186
P187
P188
P189
P190
P191
P192
P193
P194
P195
P196
P197
P198
P199
P200
P201
P202
P203
P204
P205
P206
P207
P208
P209
P210
P211
P212
P213
P214
P215
P216
P217
P218
P219
P220
P221
P222
P223
P224
P225
P226
P227
P228
P229
P230
P231
P232
P233
P234
P235
P236
P237
P238
P239
P240
P241
P242
P243
P244
P245
P246
P247
P248
P249
P250
P251
P252
P253
P254
P255
P256
P257
P258
P259
P260
P261
P262
P263
P264
P265
P266
P267
P268
P269
P270
P271
P272
P273
P274
P275
P276
P277
P278
P279
P280
P281
P282
P283
P284
P285
P286
P287
P288
P289
P290
P291
P292
P293
P294
P295
P296
P297
P298
P299
P300
P301
P302
P303
P304
P305
P306
P307
P308
P309
P310
P311
P312
P313
P314
P315
P316
P317
P318
P319
P320
P321
P322
P323
P324
P325
P326
P327
P328
P329
P330
P331
P332
P333
P334
P335
P336
P337
P338
P339
P340
P341
P342
P343
P344
P345
P346
P347
P348
P349
P350
P351
P352
P353
P354
P355
P356
P357
P358
P359
P360
P361
P362
P363
P364
P365
P366
P367
P368
P369
P370
P371
P372
P373
P374
P375
P376
P377
P378
P379
P380
P381
P382
P383
P384
P385
P386
P387
P388
P389
P390
P391
P392
P393
P394
P395
P396
P397
P398
P399
P400
P401
P402
P403
P404
P405
P406
P407
P408
P409
P410
P411
P412
P413
P414
P415
P416
P417
P418
P419
P420
P421
P422
P423
P424
P425
P426
P427
P428
P429
P430
P431
P432
P433
P434
P435
P436
P437
P438
P439
P440
P441
P442
P443
P444
P445
P446
P447
P448
P449
P450
P451
P452
P453
P454
P455
P456
P457
P458
P459
P460
P461
P462
P463
P464
P465
P466
P467
P468
P469
P470
P471
P472
P473
P474
P475
P476
P477
P478
P479
P480
P481
P482
P483
P484
P485
P486
P487
P488
P489
P490
P491
P492
P493
P494
P495
P496
P497
P498
P499
P500
P501
P502
P503
P504
P505
P506
P507
P508
P509
P510
P511
P512
P513
P514
P515
P516
P517
P518
P519
P520
P521
P522
P523
P524
P525
P526
P527
P528
P529
P530
P531
P532
P533
P534
P535
P536
P537
P538
P539
P540
P541
P542
P543
P544
P545
P546
P547
P548
P549
P550
P551
P552
P553
P554
P555
P556
P557
P558
P559
P560
P561
P562
P563
P564
P565
P566
P567
P568
P569
P570
P571
P572
P573
P574
P575
P576
P577
P578
P579
P580
P581
P582
P583
P584
P585
P586
P587
P588
P589
P590
P591
P592
P593
P594
P595
P596
P597
P598
P599
P600
P601
P602
P603
P604
P605
P606
P607
P608
P609
P610
P611
P612
P613
P614
P615
P616
P617
P618
P619
P620
P621
P622
P623
P624
P625
P626
P627
P628
P629
P630
P631
P632
P633
P634
P635
P636
P637
P638
P639
P640
P641
P642
P643
P644
P645
P646
P647
P648
P649
P650
P651
P652
P653
P654
P655
P656
P657
P658
P659
P660
P661
P662
P663
P664
P665
P666
P667
P668
P669
P670
P671
P672
P673
P674
P675
P676
P677
P678
P679
P680
P681
P682
P683
P684
P685
P686
P687
P688
P689
P690
P691
P692
P693
P694
P695
P696
P697
P698
P699
P700
P701
P702
P703
P704
P705
P706
P707
P708
P709
P710
P711
P712
P713
P714
P715
P716
P717
P718
P719
P720
P721
P722
P723
P724
P725
P726
P727
P728
P729
P730
P731
P732
P733
P734
P735
P736
P737
P738
P739
P740
P741
P742
P743
P744
P745
P746
P747
P748
P749
P750
P751
P752
P753
P754
P755
P756
P757
P758
P759
P760
P761
P762
P763
P764
P765
P766
P767
P768
P769
P770
P771
P772
P773
P774
P775
P776
P777
P778
P779
P780
P781
P782
P783
P784
P785
P786
P787
P788
P789
P790
P791
P792
P793
P794
P795
P796
P797
P798
P799
P800
P801
P802
P803
P804
P805
P806
P807
P808
P809
P810
P811
P812
P813
P814
P815
P816
P817
P818
P819
P820
P821
P822
P823
P824
P825
P826
P827
P828
P829
P830
P831
P832
P833
P834
P835
P836
P837
P838
P839
P840
P841
P842
P843
P844
P845
P846
P847
P848
P849
P850
P851
P852
P853
P854
P855
P856
P857
P858
P859
P860
P861
P862
P863
P864
P865
P866
P867
P868
P869
P870
P871
P872
P873
P874
P875
P876
P877
P878
P879
P880
P881
P882
P883
P884
P885
P886
P887
P888
P889
P890
P891
P892
P893
P894
P895
P896
P897
P898
P899
P900
P901
P902
P903
P904
P905
P906
P907
P908
P909
P910
P911
P912
P913
P914
P915
P916
P917
P918
P919
P920
P921
P922
P923
P924
P925
P926
P927
P928
P929
P930
P931
P932
P933
P934
P935
P936
P937
P938
P939
P940
P941
P942
P943
P944
P945
P946
P947
P948
P949
P950
P951
P952
P953
P954
P955
P956
P957
P958
P959
P960
P961
P962
P963
P964
P965
P966
P967
P968
P969
P970
P971
P972
P973
P974
P975
P976
P977
P978
P979
P980
P981
P982
P983
P984
P985
P986
P987
P988
P989
P990
P991
P992
P993
P994
P995
P996
P997
P998
P999
P1000

PCA : N Components = 875, 1500, 3000
kPCA : 875, 500

X – Imputed, Scaled, CURSE OF DIMENSIONALITY

X -> Imputed -> Scaled

Feature selection

Model selection using

Random Forest

XGBoost

SHAP values

P12
P122
P125
P126
P130
P131
P132
P133
P134
P135
P136
P137
P138
P139
P140
P141
P142
P143
P144
P145
P146
P147
P148
P149
P150
P151
P152
P153
P154
P155
P156
P157
P158
P159
P160
P161
P162
P163
P164
P165
P166
P167
P168
P169
P170
P171
P172
P173
P174
P175
P176
P177
P178
P179
P180
P181
P182
P183
P184
P185
P186
P187
P188
P189
P190
P191
P192
P193
P194
P195
P196
P197
P198
P199
P200
P201
P202
P203
P204
P205
P206
P207
P208
P209
P210
P211
P212
P213
P214
P215
P216
P217
P218
P219
P220
P221
P222
P223
P224
P225
P226
P227
P228
P229
P230
P231
P232
P233
P234
P235
P236
P237
P238
P239
P240
P241
P242
P243
P244
P245
P246
P247
P248
P249
P250
P251
P252
P253
P254
P255
P256
P257
P258
P259
P260
P261
P262
P263
P264
P265
P266
P267
P268
P269
P270
P271
P272
P273
P274
P275
P276
P277
P278
P279
P280
P281
P282
P283
P284
P285
P286
P287
P288
P289
P290
P291
P292
P293
P294
P295
P296
P297
P298
P299
P300
P301
P302
P303
P304
P305
P306
P307
P308
P309
P310
P311
P312
P313
P314
P315
P316
P317
P318
P319
P320
P321
P322
P323
P324
P325
P326
P327
P328
P329
P330
P331
P332
P333
P334
P335
P336
P337
P338
P339
P340
P341
P342
P343
P344
P345
P346
P347
P348
P349
P350
P351
P352
P353
P354
P355
P356
P357
P358
P359
P360
P361
P362
P363
P364
P365
P366
P367
P368
P369
P370
P371
P372
P373
P374
P375
P376
P377
P378
P379
P380
P381
P382
P383
P384
P385
P386
P387
P388
P389
P390
P391
P392
P393
P394
P395
P396
P397
P398
P399
P400
P401
P402
P403
P404
P405
P406
P407
P408
P409
P410
P411
P412
P413
P414
P415
P416
P417
P418
P419
P420
P421
P422
P423
P424
P425
P426
P427
P428
P429
P430
P431
P432
P433
P434
P435
P436
P437
P438
P439
P440
P441
P442
P443
P444
P445
P446
P447
P448
P449
P450
P451
P452
P453
P454
P455
P456
P457
P458
P459
P460
P461
P462
P463
P464
P465
P466
P467
P468
P469
P470
P471
P472
P473
P474
P475
P476
P477
P478
P479
P480
P481
P482
P483
P484
P485
P486
P487
P488
P489
P490
P491
P492
P493
P494
P495
P496
P497
P498
P499
P500
P501
P502
P503
P504
P505
P506
P507
P508
P509
P510
P511
P512
P513
P514
P515
P516
P517
P518
P519
P520
P521
P522
P523
P524
P525
P526
P527
P528
P529
P530
P531
P532
P533
P534
P535
P536
P537
P538
P539
P540
P541
P542
P543
P544
P545
P546
P547
P548
P549
P550
P551
P552
P553
P554
P555
P556
P557
P558
P559
P560
P561
P562
P563
P564
P565
P566
P567
P568
P569
P570
P571
P572
P573
P574
P575
P576
P577
P578
P579
P580
P581
P582
P583
P584
P585
P586
P587
P588
P589
P590
P591
P592
P593
P594
P595
P596
P597
P598
P599
P600
P601
P602
P603
P604
P605
P606
P607
P608
P609
P610
P611
P612
P613
P614
P615
P616
P617
P618
P619
P620
P621
P622
P623
P624
P625
P626
P627
P628
P629
P630
P631
P632
P633
P634
P635
P636
P637
P638
P639
P640
P641
P642
P643
P644
P645
P646
P647
P648
P649
P650
P651
P652
P653
P654
P655
P656
P657
P658
P659
P660
P661
P662
P663
P664
P665
P666
P667
P668
P669
P670
P671
P672
P673
P674
P675
P676
P677
P678
P679
P680
P681
P682
P683
P684
P685
P686
P687
P688
P689
P690
P691
P692
P693
P694
P695
P696
P697
P698
P699
P700
P701
P702
P703
P704
P705
P706
P707
P708
P709
P710
P711
P712
P713
P714
P715
P716
P717
P718
P719
P720
P721
P722
P723
P724
P725
P726
P727
P728
P729
P730
P731
P732
P733
P734
P735
P736
P737
P738
P739
P740
P741
P742
P743
P744
P745
P746
P747
P748
P749
P750
P751
P752
P753
P754
P755
P756
P757
P758
P759
P760
P761
P762
P763
P764
P765
P766
P767
P768
P769
P770
P771
P772
P773
P774
P775
P776
P777
P778
P779
P780
P781
P782
P783
P784
P785
P786
P787
P788
P789
P790
P791
P792
P793
P794
P795
P796
P797
P798
P799
P800
P801
P802
P803
P804
P805
P806
P807
P808
P809
P810
P811
P812
P813
P814
P815
P816
P817
P818
P819
P820
P821
P822
P823
P824
P825
P826
P827
P828
P829
P830
P831
P832
P833
P834
P835
P836
P837
P838
P839
P840
P841
P842
P843
P844
P845
P846
P847
P848
P849
P850
P851
P852
P853
P854
P855
P856
P857
P858
P859
P860
P861
P862
P863
P864
P865
P866
P867
P868
P869
P870
P871
P872
P873
P874
P875
P876
P877
P878
P879
P880
P881
P882
P883
P884
P885
P886
P887
P888
P889
P890
P891
P892
P893
P894
P895
P896
P897
P898
P899
P900
P901
P902
P903
P904
P905
P906
P907
P908
P909
P910
P911
P912
P913
P914
P915
P916
P917
P918
P919
P920
P921
P922
P923
P924
P925
P926
P927
P928
P929
P930
P931
P932
P933
P934
P935
P936
P937
P938
P939
P940
P941
P942
P943
P944
P945
P946
P947
P948
P949
P950
P951
P952
P953
P954
P955
P956
P957
P958
P959
P960
P961
P962
P963
P964
P965
P966
P967
P968
P969
P970
P971
P972
P973
P974
P975
P976
P977
P978
P979
P980
P981
P982
P983
P984
P985
P986
P987
P988
P989
P990
P991
P992
P993
P994
P995
P996
P997
P998
P999
P1000

X – Imputed, Scaled, CURSE OF DIMENSIONALITY

X -> Imputed -> Scaled

Feature selection

Model selection using

Random Forest

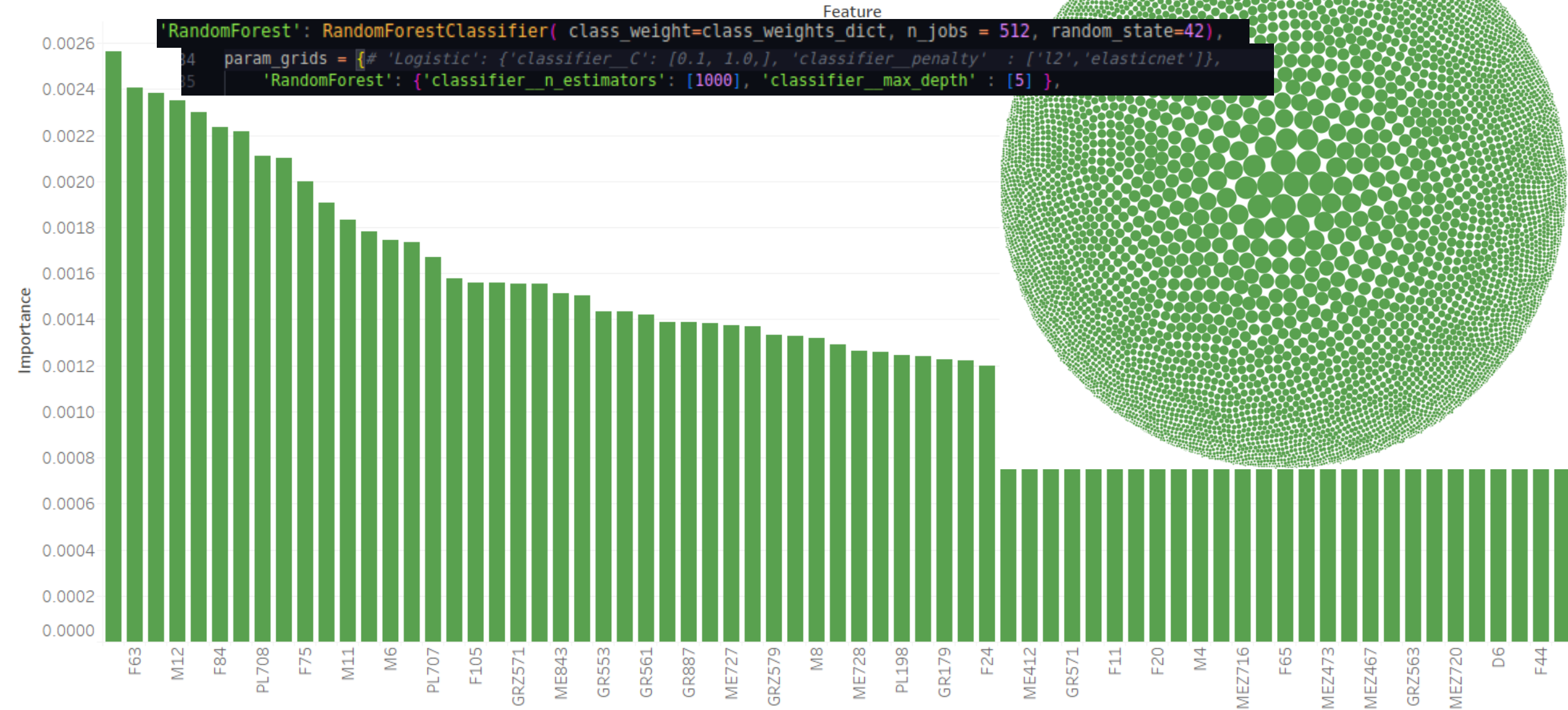
XGBoost

SHAP values

P12
P122
P125
P126
P130
P131
P132
P133
P134
P135
P136
P137
P138
P139
P140
P141
P142
P143
P144
P145
P146
P147
P148
P149
P150
P151
P152
P153
P154
P155
P156
P157
P158
P159
P160
P161
P162
P163
P164
P165
P166
P167
P168
P169
P170
P171
P172
P173
P174
P175
P176
P177
P178
P179
P180
P181
P182
P183
P184
P185
P186
P187
P188
P189
P190
P191
P192
P193
P194
P195
P196
P197
P198
P199
P200
P201
P202
P203
P204
P205
P206
P207
P208
P209
P210
P211
P212
P213
P214
P215
P216
P217
P218
P219
P220
P221
P222
P223
P224
P225
P226
P227
P228
P229
P230
P231
P232
P233
P234
P235
P236
P237
P238
P239
P240
P241
P242
P243
P244
P245
P246
P247
P248
P249
P250
P251
P252
P253
P254
P255
P256
P257
P258
P259
P260
P261
P262
P263
P264
P265
P266
P267
P268
P269
P270
P271
P272
P273
P274
P275
P276
P277
P278
P279
P280
P281
P282
P283
P284
P285
P286
P287
P288
P289
P290
P291
P292
P293
P294
P295
P296
P297
P298
P299
P300
P301
P302
P303
P304
P305
P306
P307
P308
P309
P310
P311
P312
P313
P314
P315
P316
P317
P318
P319
P320
P321
P322
P323
P324
P325
P326
P327
P328
P329
P330
P331
P332
P333
P334
P335
P336
P337
P338
P339
P340
P341
P342
P343
P344
P345
P346
P347
P348
P349
P350
P351
P352
P353
P354
P355
P356
P357
P358
P359
P360
P361
P362
P363
P364
P365
P366
P367
P368
P369
P370
P371
P372
P373
P374
P375
P376
P377
P378
P379
P380
P381
P382
P383
P384
P385
P386
P387
P388
P389
P390
P391
P392
P393
P394
P395
P396
P397
P398
P399
P400
P401
P402
P403
P404
P405
P406
P407
P408
P409
P410
P411
P412
P413
P414
P415
P416
P417
P418
P419
P420
P421
P422
P423
P424
P425
P426
P427
P428
P429
P430
P431
P432
P433
P434
P435
P436
P437
P438
P439
P440
P441
P442
P443
P444
P445
P446
P447
P448
P449
P450
P451
P452
P453
P454
P455
P456
P457
P458
P459
P460
P461
P462
P463
P464
P465
P466
P467
P468
P469
P470
P471
P472
P473
P474
P475
P476
P477
P478
P479
P480
P481
P482
P483
P484
P485
P486
P487
P488
P489
P490
P491
P492
P493
P494
P495
P496
P497
P498
P499
P500
P501
P502
P503
P504
P505
P506
P507
P508
P509
P510
P511
P512
P513
P514
P515
P516
P517
P518
P519
P520
P521
P522
P523
P524
P525
P526
P527
P528
P529
P530
P531
P532
P533
P534
P535
P536
P537
P538
P539
P540
P541
P542
P543
P544
P545
P546
P547
P548
P549
P550
P551
P552
P553
P554
P555
P556
P557
P558
P559
P560
P561
P562
P563
P564
P565
P566
P567
P568
P569
P570
P571
P572
P573
P574
P575
P576
P577
P578
P579
P580
P581
P582
P583
P584
P585
P586
P587
P588
P589
P590
P591
P592
P593
P594
P595
P596
P597
P598
P599
P600
P601
P602
P603
P604
P605
P606
P607
P608
P609
P610
P611
P612
P613
P614
P615
P616
P617
P618
P619
P620
P621
P622
P623
P624
P625
P626
P627
P628
P629
P630
P631
P632
P633
P634
P635
P636
P637
P638
P639
P640
P641
P642
P643
P644
P645
P646
P647
P648
P649
P650
P651
P652
P653
P654
P655
P656
P657
P658
P659
P660
P661
P662
P663
P664
P665
P666
P667
P668
P669
P670
P671
P672
P673
P674
P675
P676
P677
P678
P679
P680
P681
P682
P683
P684
P685
P686
P687
P688
P689
P690
P691
P692
P693
P694
P695
P696
P697
P698
P699
P700
P701
P702
P703
P704
P705
P706
P707
P708
P709
P710
P711
P712
P713
P714
P715
P716
P717
P718
P719
P720
P721
P722
P723
P724
P725
P726
P727
P728
P729
P730
P731
P732
P733
P734
P735
P736
P737
P738
P739
P740
P741
P742
P743
P744
P745
P746
P747
P748
P749
P750
P751
P752
P753
P754
P755
P756
P757
P758
P759
P760
P761
P762
P763
P764
P765
P766
P767
P768
P769
P770
P771
P772
P773
P774
P775
P776
P777
P778
P779
P780
P781
P782
P783
P784
P785
P786
P787
P788
P789
P790
P791
P792
P793
P794
P795
P796
P797
P798
P799
P800
P801
P802
P803
P804
P805
P806
P807
P808
P809
P810
P811
P812
P813
P814
P815
P816
P817
P818
P819
P820
P821
P822
P823
P824
P825
P826
P827
P828
P829
P830
P831
P832
P833
P834
P835
P836
P837
P838
P839
P840
P841
P842
P843
P844
P845
P846
P847
P848
P849
P850
P851
P852
P853
P854
P855
P856
P857
P858
P859
P860
P861
P862
P863
P864
P865
P866
P867
P868
P869
P870
P871
P872
P873
P874
P875
P876
P877
P878
P879
P880
P881
P882
P883
P884
P885
P886
P887
P888
P889
P890
P891
P892
P893
P894
P895
P896
P897
P898
P899
P900
P901
P902
P903
P904
P905
P906
P907
P908
P909
P910
P911
P912
P913
P914
P915
P916
P917
P918
P919
P920
P921
P922
P923
P924
P925
P926
P927
P928
P929
P930
P931
P932
P933
P934
P935
P936
P937
P938
P939
P940
P941
P942
P943
P944
P945
P946
P947
P948
P949
P950
P951
P952
P953
P954
P955
P956
P957
P958
P959
P960
P961
P962
P963
P964
P965
P966
P967
P968
P969
P970
P971
P972
P973
P974
P975
P976
P977
P978
P979
P980
P981
P982
P983
P984
P985
P986
P987
P988
P989
P990
P991
P992
P993
P994
P995
P996
P997
P998
P999
P1000

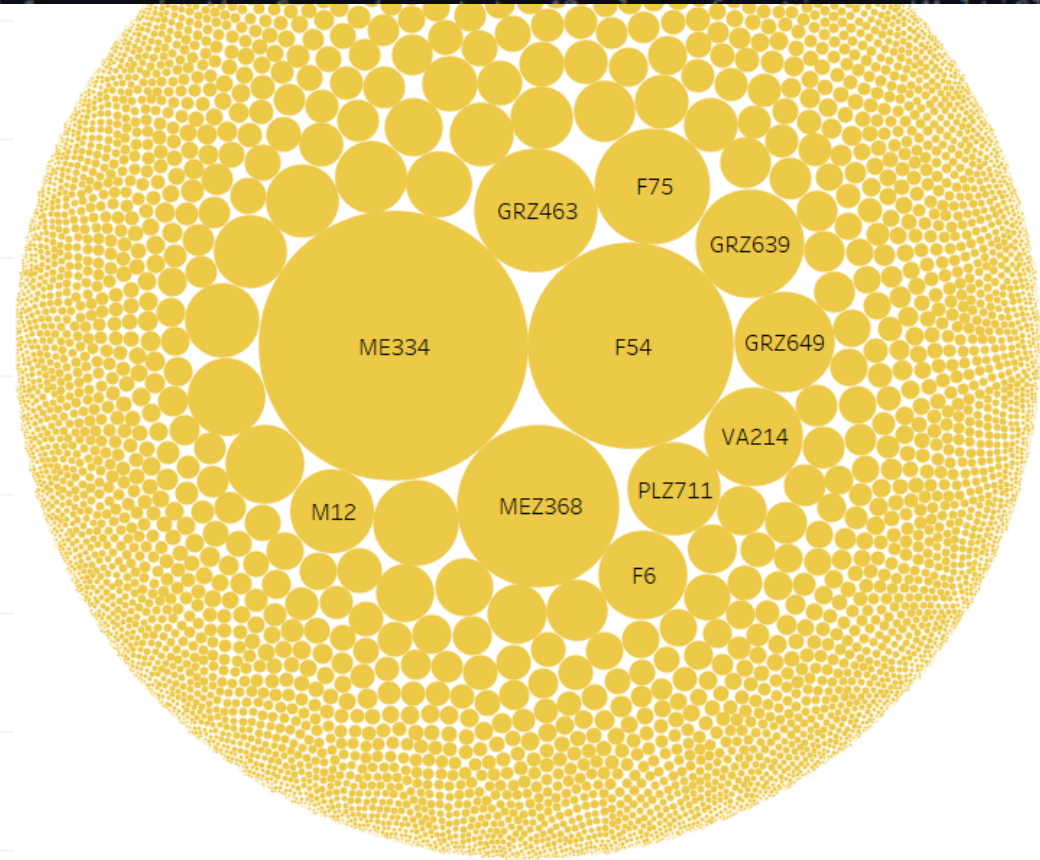
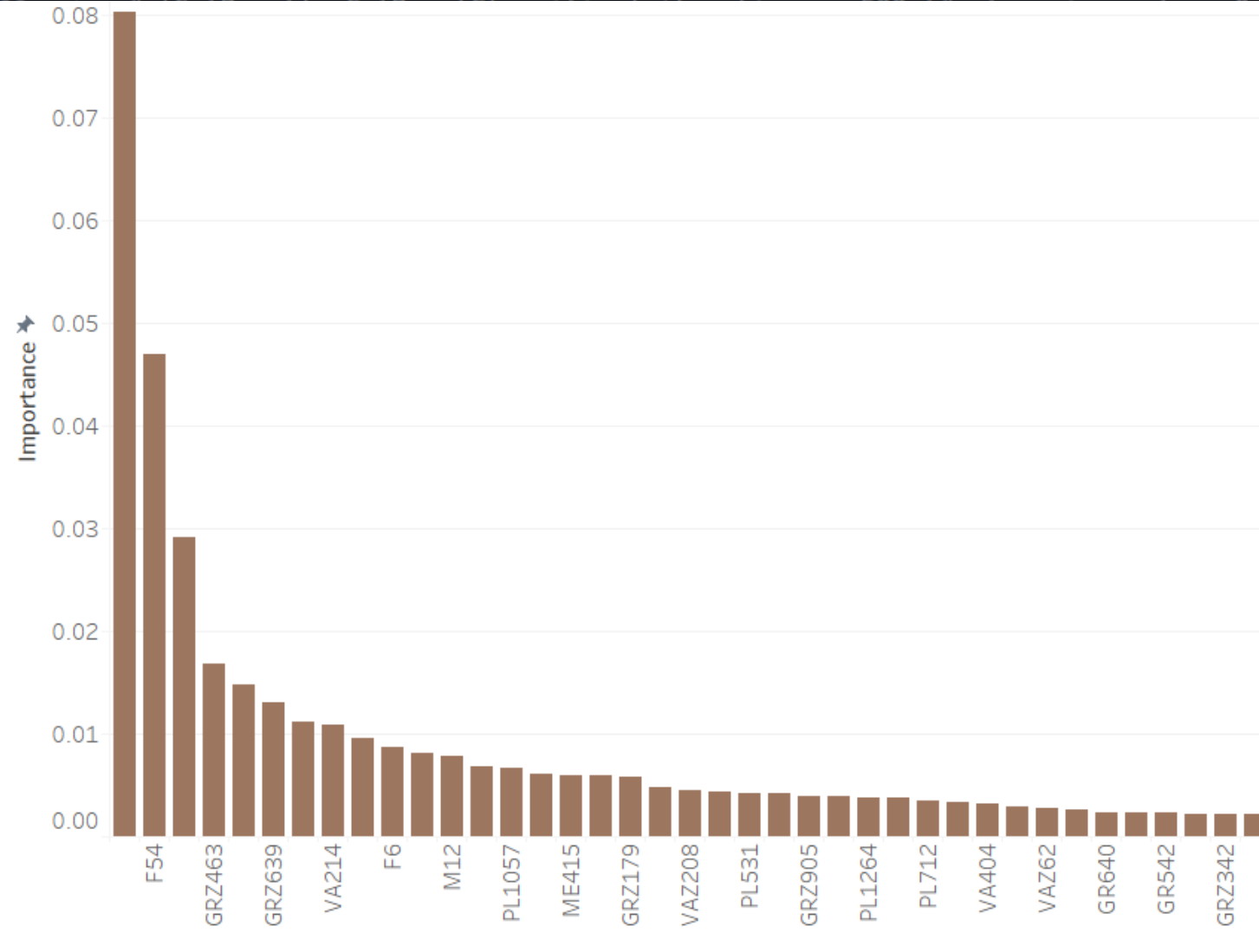
X – Imputed, Scaled, Model Selection

Important Features from Random Forest



Feature Importance from XGB-boost

```
58  
59 'XGBoost': XGBClassifier(learning_rate =0.1, max_depth =6, min_child_weight =3, n_estimators= 500,objective = 'multiclass',num_cla
```



X – Imputed, Scaled, Features Reduced , **TRAIN**, stratified y

```
# Compute class weights based on inverse class frequency
class_labels = np.unique(y_train_noC) # Assuming y_train contains the target labels
class_weights = compute_class_weight(class_weight="balanced", classes=np.unique(y_train_noC), y=y_train_noC)
class_weights_dict = dict(zip(class_labels, class_weights))

print('dataset loaded')
```

Class weight distribution
was estimated

```
68 #-----CLASSIFIERS-----#
69 classifiers = {
70     'Logistic': LogisticRegression(class_weight=class_weights_dict,multi_class='ovr',random_state=42),
71     'RandomForest': RandomForestClassifier( class_weight=class_weights_dict, n_jobs = 512, random_state=42),
72     'XGBoost': XGBClassifier(objective = 'multiclass',num_class = 29, random_state=42),
73     'CatBoost': CatBoostClassifier( random_state=42, loss_function = 'MultiClass'),
74     'NaiveBayes': GaussianNB(),
75     'SVM': SVC(random_state=42),
76     #'LightGBM': LGBMClassifier(objective = 'multiclass', class_weight = class_weights_dict,n_jobs = 512,
77     random_state=42),
78 }
79
80
81
82
83
84 param_grids = { 'Logistic': {'classifier__C': [0.1, 1.0,2.0], 'classifier__penalty' : ['l2','elasticnet']},
85     'RandomForest': {'classifier__n_estimators': [100,500,1000], 'classifier__max_depth' : [10,5,3,15] },
86
87     'XGBoost': {'classifier__n_estimators': [100, 500, 1000], 'classifier__learning_rate': [0.1, 1.0, 10],
88     'classifier__max_depth' : [3,6, 9], 'classifier__min_child_weight' : [1,3]},# 'classifier__eval_metric' =
89     ['merror', 'mlogloss']},
90
91     'CatBoost': {'classifier__iterations': [100, 500, 1000], 'classifier__learning_rate': [0.1, 1.0, 0.5],
92     'classifier__max_depth' : [3,6, 9] },
93     'NaiveBayes': {}, # Naive Bayes does not have hyperparameters to tune
94     'SVM': {'classifier__C': [1.0, 10.0], 'classifier__decision_function_shape' : ['ovo', 'ovr']},
95     'LightGBM': {'classifier__n_estimators': [100, 500], 'classifier__learning_rate': [ 0.1, 1.0],
96     'classifier__max_depth' : [3,5]},
97 }
```


X – Imputed, Scaled, Features Reduced , **TRAIN**, stratified y

```
# Compute class weights based on inverse class frequency
class_labels = np.unique(y_train_noC) # Assuming y_train contains the target labels
class_weights = compute_class_weight(class_weight="balanced", classes=np.unique(y_train_noC), y=y_train_noC)
class_weights_dict = dict(zip(class_labels, class_weights))

print('dataset loaded')
```

```
#-----CLASSIFIERS-----#
69 classifiers = {
70     'Logistic': LogisticRegression(class_weight=class_weights_dict, multi_class='ovr', random_state=42),
71     'RandomForest': RandomForestClassifier(class_weight=class_weights_dict, n_jobs = 512, random_state=42),
72     'XGBoost': XGBClassifier(objective = 'multiclass', num_class = 29, random_state=42),
73     'CatBoost': CatBoostClassifier(random_state=42, loss_function = 'MultiClass'),
74     'NaiveBayes': GaussianNB(),
75     'SVM': SVC(random_state=42),
76     #'LightGBM': LGBMClassifier(objective = 'multiclass', class_weight = class_weights_dict, n_jobs = 512,
77                                random_state=42),
78 }
79
80
81
82
83
84 param_grids = { 'Logistic': {'classifier_C': [0.1, 1.0, 2.0], 'classifier_penalty' : ['l2', 'elasticnet']},
85                 'RandomForest': {'classifier_n_estimators': [100, 500, 1000], 'classifier_max_depth' : [10, 5, 3, 15] },
86
87                 'XGBoost': {'classifier_n_estimators': [100, 500, 1000], 'classifier_learning_rate': [0.1, 1.0, 10],
88                             'classifier_max_depth' : [3, 6, 9], 'classifier_min_child_weight' : [1, 3]}, # 'classifier_eval_metric' =
89                             ['merror', 'mlogloss']},
90
91                 'CatBoost': {'classifier_iterations': [100, 500, 1000], 'classifier_learning_rate': [0.1, 1.0, 0.5],
92                             'classifier_max_depth' : [3, 6, 9] },
93                 'NaiveBayes': {}, # Naive Bayes does not have hyperparameters to tune
94                 'SVM': {'classifier_C': [1.0, 10.0], 'classifier_decision_function_shape' : ['ovo', 'ovr']},
95                 'LightGBM': {'classifier_n_estimators': [100, 500], 'classifier_learning_rate': [0.1, 1.0],
96                             'classifier_max_depth' : [3, 5]},
97 }
```

Different classifiers were defined.

X – Imputed, Scaled, Features Reduced , **TRAIN**, stratified y

```
# Compute class weights based on inverse class frequency
class_labels = np.unique(y_train_noC) # Assuming y_train contains the target labels
class_weights = compute_class_weight(class_weight="balanced", classes=np.unique(y_train_noC), y=y_train_noC)
class_weights_dict = dict(zip(class_labels, class_weights))

print('dataset loaded')
```

```
68 #-----CLASSIFIERS-----#
69 classifiers = {
70     'Logistic': LogisticRegression(class_weight=class_weights_dict,multi_class='ovr',random_state=42),
71     'RandomForest': RandomForestClassifier( class_weight=class_weights_dict, n_jobs = 512, random_state=42),
72     'XGBoost': XGBClassifier(objective = 'multiclass',num_class = 29, random_state=42),
73     'CatBoost': CatBoostClassifier( random_state=42, loss_function = 'MultiClass'),
74     'NaiveBayes': GaussianNB(),
75     'SVM': SVC(random_state=42),
76     #'LightGBM': LGBMClassifier(objective = 'multiclass', class_weight = class_weights_dict,n_jobs = 512,
77     random_state=42),
78 }
79
80
81
82
83
84 param_grids = { 'Logistic': {'classifier__C': [0.1, 1.0,2.0], 'classifier__penalty' : ['l2','elasticnet']},
85     'RandomForest': {'classifier__n_estimators': [100,500,1000], 'classifier__max_depth' : [10,5,3,15] },
86
87     'XGBoost': {'classifier__n_estimators': [100, 500, 1000], 'classifier__learning_rate': [0.1, 1.0, 10],
88     'classifier__max_depth' : [3,6, 9], 'classifier__min_child_weight' : [1,3]},# 'classifier__eval_metric' =
89     ['merror', 'mlogloss']},
90
91     'CatBoost': {'classifier__iterations': [100, 500, 1000], 'classifier__learning_rate': [0.1, 1.0, 0.5],
92     'classifier__max_depth' : [3,6, 9] },
93     'NaiveBayes': {}, # Naive Bayes does not have hyperparameters to tune
94     'SVM': {'classifier__C': [1.0, 10.0], 'classifier__decision_function_shape' : ['ovo', 'ovr']},
95     'LightGBM': {'classifier__n_estimators': [100, 500], 'classifier__learning_rate': [ 0.1, 1.0],
96     'classifier__max_depth' : [3,5]},
97 }
```

Different parameters were tuned

X – Imputed, Scaled, Features Reduced , TRAIN

```
103 #-----GRIDSEARCH-----#
104 nSplit = 5
105 print(f'nSplit : {nSplit}')
106 # Initialize StratifiedKfold cross-validator
107 stratified_kfold = StratifiedKFold(n_splits=nSplit, shuffle=True, random_state=42)
108 results = []
109 # Fit and save the best model for each classifier
110 for metrics_name, metrictype in scorers.items():
111     print(f'----{metrics_name}----')
112     metric_results = []
113     for name, pipeline in pipelines.items():
114         # Define the parameter grid for the current classifier
115         param_grid = param_grids.get(name, {})
116
117         #for clf_name, clf in classifiers.items():
118             print(f'-----{name}----')
119
120             # Get parameter grid for current classifier
121
122             grid_search = GridSearchCV(pipeline, param_grid, cv=stratified_kfold, scoring=metrictype, n_jobs=512,
123                                       verbose=4)
124             grid_search.fit(X_train, y_train)
125             # print(len(y_predicted))
126
127 # Save the best model
128 print('grid-search over')
129 best_model = grid_search.best_estimator_
130 print('best Model')
131 dump(best_model, f'/mnt/gpfs3_amd/scratch/rgu245/intel/final/dim-red/models-best-params/f1/nsplit1{name}')
132 best_model_{metrics_name}_rob_med_pca1500.joblib')
133 # Append results to the list for current metric
134 metric_results.append({'Classifier': name,
135                       'Metric': metrics_name,
136                       'Best Score': grid_search.best_score_,
137                       'Best Parameters': grid_search.best_params_})
138 print(f'metric Result
139 -----
140 -{metric_results}')
```

- Jobs were run in parallel
- Grid search was performed across Stratified kFold to get the best parameters


```
matplotlib successful
libs imported
files loaded
dataset loaded
nSplit : 5
----f1_weighted---
----RandomForest---
Fitting 5 folds for each of 9 candidates, totalling 45 fits
[CV 1/5] END classifier_max_depth=3, classifier_n_estimators=100,, score=0.605 total time= 18.2s
[CV 2/5] END classifier_max_depth=3, classifier_n_estimators=100,, score=0.578 total time= 21.1s
[CV 3/5] END classifier_max_depth=3, classifier_n_estimators=500,, score=0.606 total time= 35.1s
[CV 3/5] END classifier_max_depth=5, classifier_n_estimators=100,, score=0.655 total time= 40.4s
[CV 5/5] END classifier_max_depth=5, classifier_n_estimators=100,, score=0.619 total time= 41.2s
[CV 2/5] END classifier_max_depth=5, classifier_n_estimators=100,, score=0.632 total time= 42.2s
F1 score on validation fold: 0.761370265637153
Fitting 5 folds for each of 9 candidates, totalling 45 fits
F1 score on validation fold: 0.7591858793118769
Fitting 5 folds for each of 9 candidates, totalling 45 fits
F1 score on validation fold: 0.7581896563000856
Fitting 5 folds for each of 9 candidates, totalling 45 fits
[CV 5/5] END classifier_max_depth=5, classifier_n_estimators=100,, score=0.642 total time= 50.7s
[CV 5/5] END classifier_max_depth=3, classifier_n_estimators=100,, score=0.593 total time= 52.3s
[CV 1/5] END classifier_max_depth=5, classifier_n_estimators=500,, score=0.658 total time= 54.7s
[CV 3/5] END classifier_max_depth=5, classifier_n_estimators=500,, score=0.654 total time= 1.0min
[CV 4/5] END classifier_max_depth=10, classifier_n_estimators=100,, score=0.746 total time= 1.0min
[CV 1/5] END classifier_max_depth=3, classifier_n_estimators=100,, score=0.609 total time= 1.0min
[CV 3/5] END classifier_max_depth=3, classifier_n_estimators=1000,, score=0.606 total time= 1.1min
[CV 2/5] END classifier_max_depth=5, classifier_n_estimators=500,, score=0.649 total time= 1.1min
F1 score on validation fold: 0.7618506629340698
Fitting 5 folds for each of 9 candidates, totalling 45 fits
F1 score on validation fold: 0.7624971668562314
Average F1 score for RandomForest: 0.7606187262078834
```

160.7606187262078834

```
217 [CV 1/5] END classifier_max_depth=10, classifier_n_estimators=500,, score=0.760 total time= 1.6min
218 [CV 4/5] END classifier_max_depth=3, classifier_n_estimators=500,, score=0.621 total time= 1.7min
219 [CV 2/5] END classifier_max_depth=3, classifier_n_estimators=1000,, score=0.609 total time= 1.7min
220 [CV 2/5] END classifier_max_depth=3, classifier_n_estimators=1000,, score=0.619 total time= 1.7min
221 [CV 3/5] END classifier_max_depth=3, classifier_n_estimators=100,, score=0.594 total time= 11.5s
222 [CV 3/5] END classifier_max_depth=5, classifier_n_estimators=1000,, score=0.667 total time= 1.7min
223 [CV 5/5] END classifier_max_depth=5, classifier_n_estimators=100,, score=0.604 total time= 1.3min
224 [CV 4/5] END classifier_max_depth=10, classifier_n_estimators=500,, score=0.756 total time= 1.6min
225 [CV 5/5] END classifier_max_depth=10, classifier_n_estimators=500,, score=0.751 total time= 1.8min
226 [CV 3/5] END classifier_max_depth=5, classifier_n_estimators=500,, score=0.673 total time= 1.7min
227 [CV 2/5] END classifier_max_depth=3, classifier_n_estimators=500,, score=0.597 total time= 1.3min
228 [CV 5/5] END classifier_max_depth=10, classifier_n_estimators=500,, score=0.754 total time= 1.7min
229 [CV 4/5] END classifier_max_depth=10, classifier_n_estimators=1000,, score=0.767 total time= 1.8min
230 [CV 2/5] END classifier_max_depth=5, classifier_n_estimators=1000,, score=0.654 total time= 1.7min
231 [CV 4/5] END classifier_max_depth=10, classifier_n_estimators=100,, score=0.734 total time= 1.4min
232 [CV 3/5] END classifier_max_depth=10, classifier_n_estimators=1000,, score=0.774 total time= 1.7min
233 [CV 1/5] END classifier_max_depth=3, classifier_n_estimators=100,, score=0.585 total time= 12.8s
234 [CV 5/5] END classifier_max_depth=5, classifier_n_estimators=1000,, score=0.628 total time= 1.7min
235 [CV 4/5] END classifier_max_depth=3, classifier_n_estimators=1000,, score=0.628 total time= 1.7min
236 [CV 4/5] END classifier_max_depth=3, classifier_n_estimators=1000,, score=0.614 total time= 1.7min
237 [CV 2/5] END classifier_max_depth=10, classifier_n_estimators=1000,, score=0.754 total time= 1.7min
238 [CV 5/5] END classifier_max_depth=3, classifier_n_estimators=1000,, score=0.600 total time= 1.8min
239 [CV 3/5] END classifier_max_depth=5, classifier_n_estimators=100,, score=0.616 total time= 1.0min
240 [CV 3/5] END classifier_max_depth=10, classifier_n_estimators=500,, score=0.773 total time= 1.8min
241 [CV 1/5] END classifier_max_depth=10, classifier_n_estimators=1000,, score=0.759 total time= 1.8min
242 [CV 1/5] END classifier_max_depth=5, classifier_n_estimators=1000,, score=0.670 total time= 1.8min
243 [CV 3/5] END classifier_max_depth=10, classifier_n_estimators=1000,, score=0.755 total time= 1.8min
244 [CV 5/5] END classifier_max_depth=5, classifier_n_estimators=500,, score=0.632 total time= 1.8min
245 [CV 4/5] END classifier_max_depth=5, classifier_n_estimators=500,, score=0.635 total time= 1.4min
246 [CV 2/5] END classifier_max_depth=10, classifier_n_estimators=1000,, score=0.756 total time= 1.8min
247 [CV 1/5] END classifier_max_depth=10, classifier_n_estimators=100,, score=0.738 total time= 1.4min
248 [CV 4/5] END classifier_max_depth=10, classifier_n_estimators=1000,, score=0.763 total time= 1.8min
249 [CV 2/5] END classifier_max_depth=3, classifier_n_estimators=100,, score=0.609 total time= 11.9s
250 [CV 4/5] END classifier_max_depth=5, classifier_n_estimators=1000,, score=0.644 total time= 1.8min
251 [CV 2/5] END classifier_max_depth=10, classifier_n_estimators=100,, score=0.732 total time= 1.3min
252 [CV 1/5] END classifier_max_depth=10, classifier_n_estimators=1000,, score=0.768 total time= 1.8min
253 [CV 5/5] END classifier_max_depth=10, classifier_n_estimators=100,, score=0.717 total time= 1.4min
254 [CV 5/5] END classifier_max_depth=10, classifier_n_estimators=1000,, score=0.756 total time= 1.8min
```

```
44 [CV 1/5] END classifier_max_depth=3, classifier_n_estimators=500,, score=0.587 total time= 1.4min
45 [CV 5/5] END classifier_max_depth=3, classifier_n_estimators=500,, score=0.588 total time= 1.4min
46 [CV 3/5] END classifier_max_depth=5, classifier_n_estimators=500,, score=0.621 total time= 1.5min
47 [CV 5/5] END classifier_max_depth=3, classifier_n_estimators=1000,, score=0.591 total time= 1.5min
48 [CV 3/5] END classifier_max_depth=10, classifier_n_estimators=100,, score=0.727 total time= 1.5min
49 [CV 5/5] END classifier_max_depth=5, classifier_n_estimators=500,, score=0.617 total time= 1.5min
50 [CV 1/5] END classifier_max_depth=5, classifier_n_estimators=500,, score=0.627 total time= 1.5min
51 [CV 4/5] END classifier_max_depth=3, classifier_n_estimators=1000,, score=0.604 total time= 1.5min
52 [CV 2/5] END classifier_max_depth=3, classifier_n_estimators=1000,, score=0.610 total time= 1.5min
53 [CV 1/5] END classifier_max_depth=3, classifier_n_estimators=1000,, score=0.593 total time= 1.5min
54 [CV 2/5] END classifier_max_depth=10, classifier_n_estimators=500,, score=0.752 total time= 1.5min
55 [CV 3/5] END classifier_max_depth=3, classifier_n_estimators=1000,, score=0.592 total time= 1.5min
56 [CV 5/5] END classifier_max_depth=10, classifier_n_estimators=500,, score=0.741 total time= 1.5min
57 [CV 4/5] END classifier_max_depth=10, classifier_n_estimators=500,, score=0.753 total time= 1.5min
58 [CV 2/5] END classifier_max_depth=5, classifier_n_estimators=500,, score=0.653 total time= 1.5min
59 [CV 5/5] END classifier_max_depth=5, classifier_n_estimators=1000,, score=0.619 total time= 1.5min
60 [CV 4/5] END classifier_max_depth=5, classifier_n_estimators=1000,, score=0.635 total time= 1.5min
61 [CV 2/5] END classifier_max_depth=5, classifier_n_estimators=1000,, score=0.659 total time= 1.6min
62 [CV 3/5] END classifier_max_depth=5, classifier_n_estimators=1000,, score=0.621 total time= 1.6min
63 [CV 3/5] END classifier_max_depth=10, classifier_n_estimators=500,, score=0.752 total time= 1.6min
64 [CV 4/5] END classifier_max_depth=10, classifier_n_estimators=500,, score=0.754 total time= 1.6min
65 [CV 1/5] END classifier_max_depth=5, classifier_n_estimators=1000,, score=0.627 total time= 1.8min
66 [CV 4/5] END classifier_max_depth=10, classifier_n_estimators=1000,, score=0.755 total time= 1.8min
67 [CV 3/5] END classifier_max_depth=10, classifier_n_estimators=1000,, score=0.752 total time= 1.8min
68 [CV 2/5] END classifier_max_depth=10, classifier_n_estimators=1000,, score=0.752 total time= 1.8min
69 [CV 1/5] END classifier_max_depth=10, classifier_n_estimators=1000,, score=0.752 total time= 1.8min
70 [CV 5/5] END classifier_max_depth=10, classifier_n_estimators=1000,, score=0.745 total time= 1.8min
71 [CV 5/5] END classifier_max_depth=3, classifier_n_estimators=100,, score=0.593 total time= 1.1min
72 [CV 1/5] END classifier_max_depth=3, classifier_n_estimators=100,, score=0.576 total time= 0.9s
73 [CV 3/5] END classifier_max_depth=3, classifier_n_estimators=100,, score=0.600 total time= 1.2min
74 [CV 4/5] END classifier_max_depth=3, classifier_n_estimators=100,, score=0.581 total time= 1.3min
75 [CV 5/5] END classifier_max_depth=10, classifier_n_estimators=100,, score=0.745 total time= 1.3min
76 [CV 2/5] END classifier_max_depth=10, classifier_n_estimators=100,, score=0.728 total time= 1.3min
77 [CV 4/5] END classifier_max_depth=3, classifier_n_estimators=100,, score=0.604 total time= 18.8s
78 [CV 1/5] END classifier_max_depth=10, classifier_n_estimators=100,, score=0.745 total time= 1.3min
79 [CV 3/5] END classifier_max_depth=3, classifier_n_estimators=100,, score=0.604 total time= 19.7s
80 [CV 4/5] END classifier_max_depth=10, classifier_n_estimators=100,, score=0.745 total time= 1.3min
81 [CV 5/5] END classifier_max_depth=3, classifier_n_estimators=100,, score=0.588 total time= 27.4s
82 [CV 1/5] END classifier_max_depth=5, classifier_n_estimators=100,, score=0.663 total time= 1.3min
83 [CV 2/5] END classifier_max_depth=3, classifier_n_estimators=100,, score=0.615 total time= 24.8s
84 [CV 2/5] END classifier_max_depth=10, classifier_n_estimators=1000,, score=0.743 total time= 1.8min
85 [CV 5/5] END classifier_max_depth=10, classifier_n_estimators=100,, score=0.730 total time= 37.1s
86 [CV 5/5] END classifier_max_depth=5, classifier_n_estimators=500,, score=0.643 total time= 1.4min
87 [CV 5/5] END classifier_max_depth=3, classifier_n_estimators=500,, score=0.596 total time= 45.1s
88 [CV 3/5] END classifier_max_depth=3, classifier_n_estimators=1000,, score=0.603 total time= 1.7min
89 [CV 1/5] END classifier_max_depth=10, classifier_n_estimators=100,, score=0.749 total time= 48.8s
90 [CV 1/5] END classifier_max_depth=5, classifier_n_estimators=1000,, score=0.657 total time= 1.6min
91 [CV 5/5] END classifier_max_depth=5, classifier_n_estimators=500,, score=0.625 total time= 47.7s
92 [CV 4/5] END classifier_max_depth=5, classifier_n_estimators=500,, score=0.622 total time= 1.5min
93 [CV 1/5] END classifier_max_depth=5, classifier_n_estimators=100,, score=0.647 total time= 49.2s
94 [CV 5/5] END classifier_max_depth=10, classifier_n_estimators=1000,, score=0.760 total time= 1.8min
95 [CV 3/5] END classifier_max_depth=10, classifier_n_estimators=100,, score=0.738 total time= 54.0s
96 [CV 1/5] END classifier_max_depth=3, classifier_n_estimators=1000,, score=0.606 total time= 1.3min
97 [CV 1/5] END classifier_max_depth=3, classifier_n_estimators=500,, score=0.597 total time= 56.1s
98 [CV 4/5] END classifier_max_depth=5, classifier_n_estimators=100,, score=0.619 total time= 1.4min
```


X – Imputed, Scaled, Features Reduced , TRAIN, MODEL EVALUATION (Best parameters)

Imputed	Scaling	Dim Reduction	nDimensions	Model	Avg auc-roc score
0	minmax	1500	pca	logistic	0.958007458
0	minmax	1500	pca	RandomForest	0.938632022
0	minmax	1500	pca	XGBoost	0.970529713
0	minmax	1500	pca	CatBoost	0.976035182
0	minmax	1500	pca	Naive Bayes	0.874

Imputed	Scaling	Dim Reduction	nDimensions	Model	Avg auc-roc score
0	minmax	875	kpca	Logistic	0.895272466
0	minmax	875	kpca	RandomForest	0.950761852
0	minmax	875	kpca	XGBoost	0.972538066
0	minmax	875	kpca	CatBoost	0.976813318
0	minmax	875	kpca	Naive Bayes	0.876

X – Imputed, Scaled, Features Reduced , TRAIN, MODEL EVALUATION (Best parameters)

Imputed	Scaling	Dim Reduction	nDimensions	Model	Avg F1 score
median	robust	pca	1500	RandomForest	0.755166382
median	robust	pca	1500	XGBoost	0.871753945
median	robust	pca	1500	CatBoost	0.858095323
median	robust	pca	1500	Naive Bayes	0.783
median	robust	pca	1500	SVM	0.922

Imputed	Scaling	Dim Reduction	nDimensions	Model	Avg F1 score
o	stdscaled	3000	pca	RandomForest	0.831177229
o	stdscaled	3000	pca	XGBoost	0.868948278
o	stdscaled	3000	pca	CatBoost	0.873476535
o	stdscaled	3000	pca	Naive Bayes	0.855489329
o	stdscaled	3000	pca	svm	o

X – Imputed, Scaled, Features Reduced , TRAIN, MODEL EVALUATION (Best parameters)

Imputed	Scaling	Dim Reduction	nDimensions	Model	Avg F1 score
0	minmax	kpca	875	RandomForest	0.828768
0	minmax	kpca	875	XGBoost	0.889695
0	minmax	kpca	875	CatBoost	0.886021
0	minmax	kpca	875	Naive Bayes	0.783
0	minmax	kpca	875	SVM	0.922

Imputed	Scaling	nDimensions	Dim Reduction	Model	Avg F1 score
0	minmax	1500	pca	RandomForest	0.819196863
0	minmax	1500	pca	XGBoost	0.887455895
0	minmax	1500	pca	CatBoost	0.88271576
0	minmax	1500	pca	Naive Bayes	0.755606434
0	minmax	1500	pca	SVM	0.921611118

X – Imputed, Scaled, Features Reduced , TRAIN, MODEL EVALUATION (Best parameters)

Imputed	Scaling	Dim Reduction	nDimensions	Model	Avg F1 score
0	minmax	kpca	875	RandomForest	0.828768
0	minmax	kpca	875	XGBoost	0.889695
0	minmax	kpca	875	CatBoost	0.886021
0	minmax	kpca	875	Naive Bayes	0.783
0	minmax	kpca	875	SVM	0.922

Imputed	Scaling	nDimensions	Dim Reduction	Model	Avg F1 score
0	minmax	1500	pca	RandomForest	0.819196863
0	minmax	1500	pca	XGBoost	0.887455895
0	minmax	1500	pca	CatBoost	0.88271576
0	minmax	1500	pca	Naive Bayes	0.755606434
0	minmax	1500	pca	SVM	0.921611118

X – Imputed, Scaled, Features Reduced , TRAIN, Best MODEL EVALUATION

Imputed	Scaling	Dim Reduction	nDimensions	Model	Avg F1 score
0	minmax	kpca	875	RandomForest	0.828768
0	minmax	kpca	875	XGBoost	0.889695
0	minmax	kpca	875	CatBoost	0.886021
0	minmax	kpca	875	Naive Bayes	0.783
0	minmax	kpca	875	SVM	0.922

Imputed	Scaling	nDimensions	Dim Reduction	Model	Avg F1 score
0	minmax	1500	pca	RandomForest	0.819196863
0	minmax	1500	pca	XGBoost	0.887455895
0	minmax	1500	pca	CatBoost	0.88271576
0	minmax	1500	pca	Naive Bayes	0.755606434
0	minmax	1500	pca	SVM	0.921611118

X – Imputed, Scaled, Features Reduced , Synthetic Minority Oversampling, MODEL Performance

```
25 # Define the SMOTETomek sampler
26 smt_tomek = SMOTETomek(tomek=TomekLinks(sampling_strategy='majority'), n_jobs =
512)
27 |
28 # Fit and resample the dataset
29
22 smote = SMOTE(random_state=42, sampling_strategy = 'not majority', n_jobs = 512)
23 _, y_train_resampled = smote.fit_resample(X_train, y_train)
```

- Under sampling and the over sampling was performed for the class distribution.
- Minority samples were SMOTEd only

Imputed	Scaling	Dim Reduction	nDimensions	Model	Avg F1 score
0	minmax	modelseletcion - rfxgb	1000	Logistic Regression	0.892149496
0	minmax	modelseletcion - rfxgb	1000	RandomForest	0.855393666
0	minmax	modelseletcion - rfxgb	1000	XGBoost	0.978986299
0	minmax	modelseletcion - rfxgb	1000	CatBoost	0.912783372

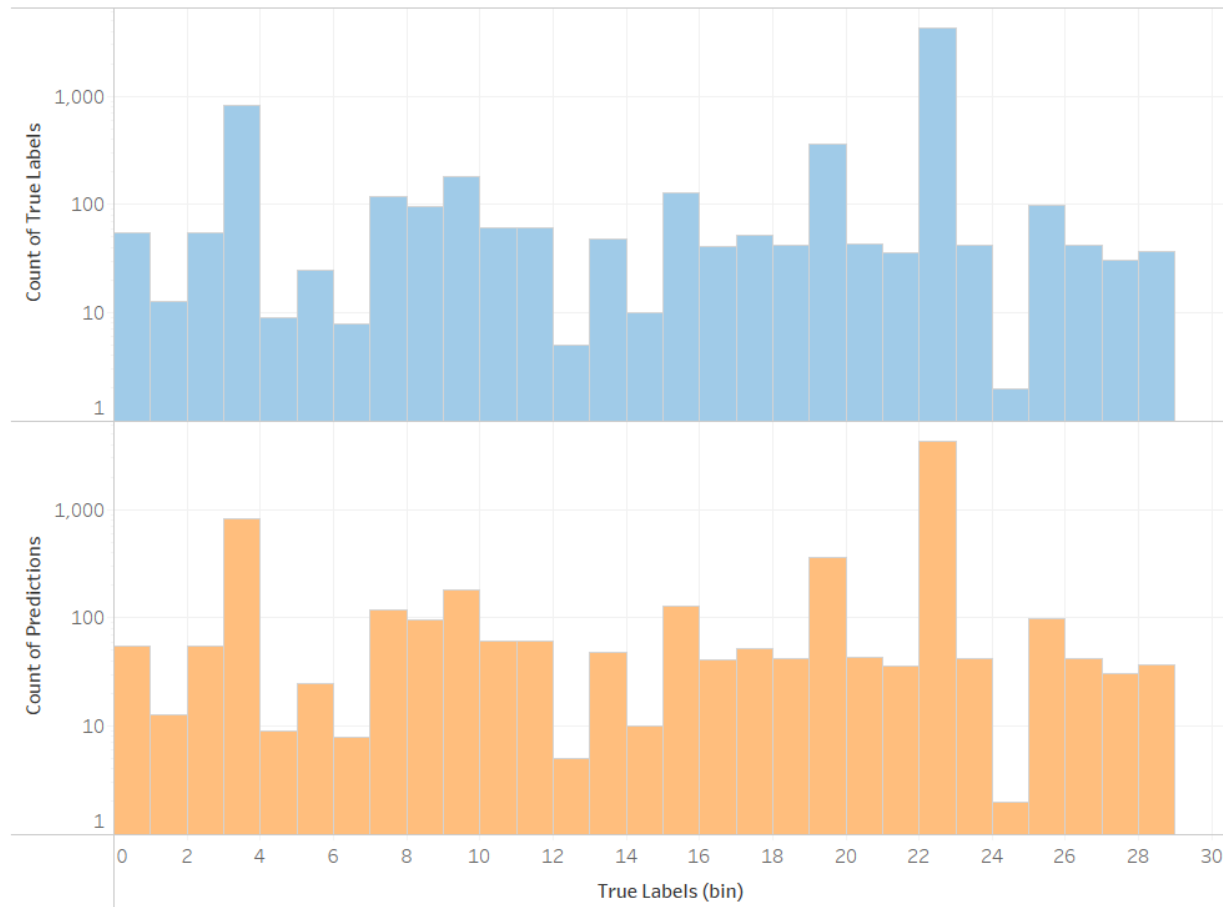
Training results for dataset that involved the synthetic generation of classes. The number of features were reduced to 1000 using Random Forest best features.

X – Imputed, Scaled, Features Reduced , TRAIN, TESTING SET

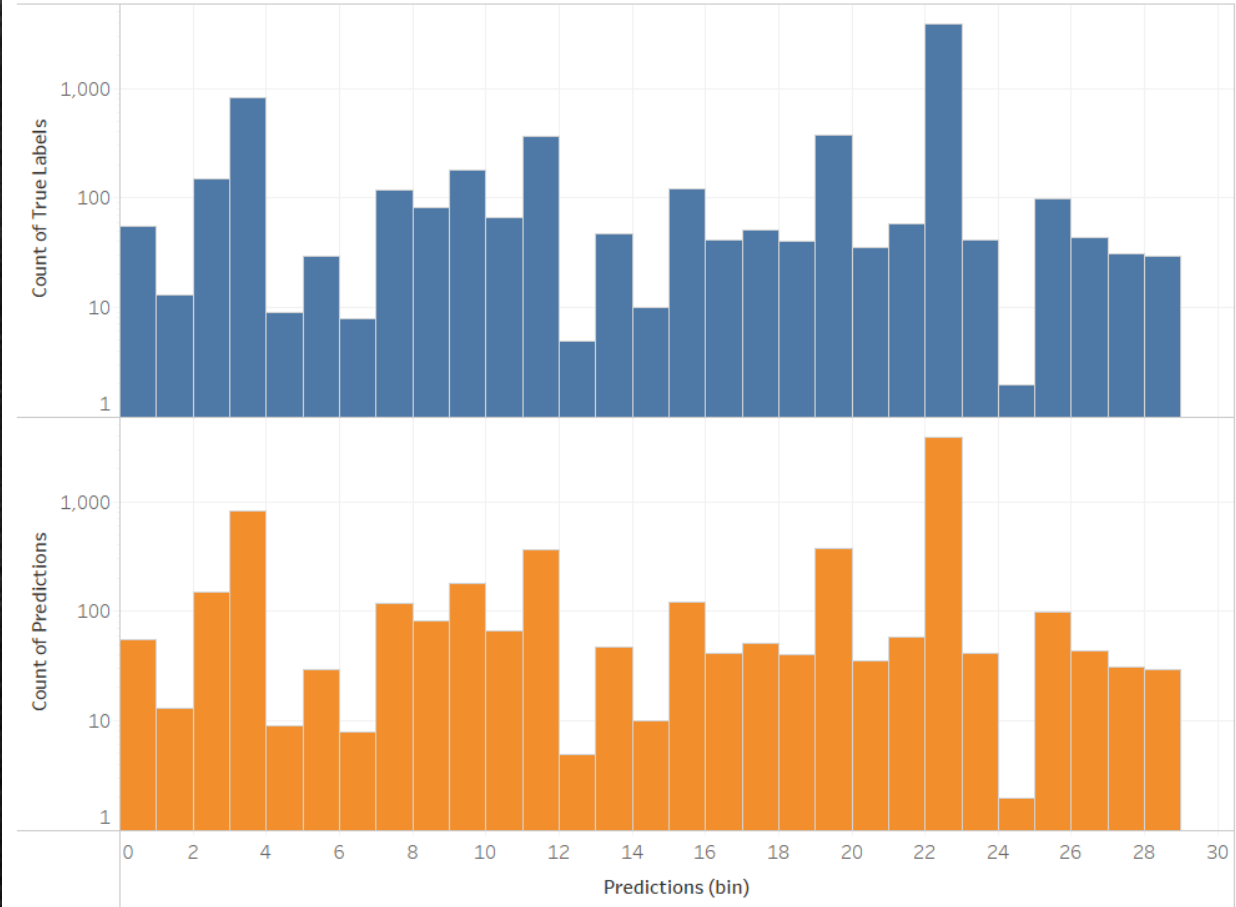
XGBoost : {'classifier__learning_rate': 0.1, 'classifier__max_depth': 6, 'classifier__min_child_weight': 3, 'classifier__n_estimators': 500}
– scaled : MinMax, Nans – o, PCA

Random Forest : 'classifier__max_depth': 10, 'classifier__n_estimators': 1000

XG Boost Prediction comparison with the True

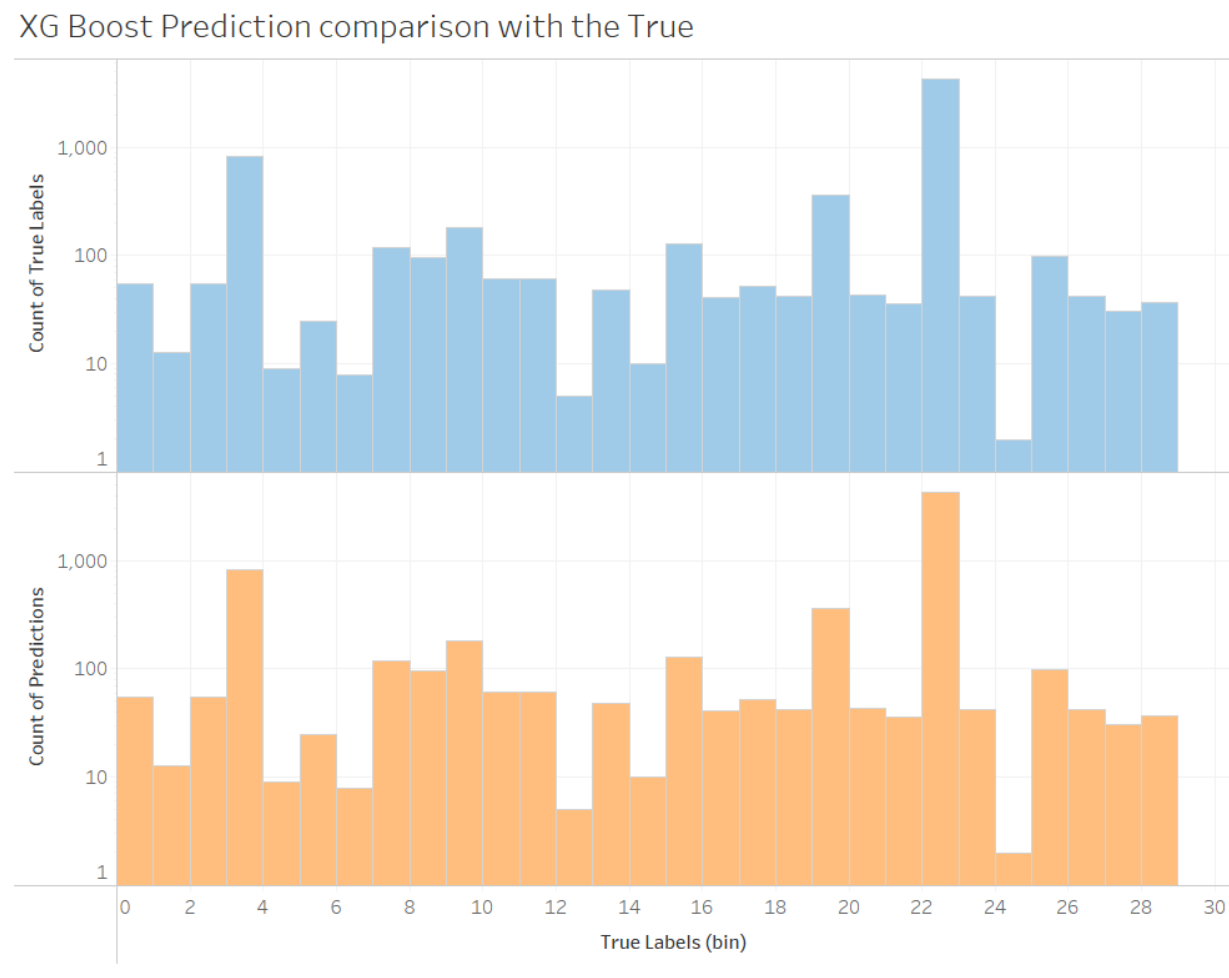


Random Forest - F1 weighted - Imputed Zero, min max scaled



X – Imputed, Scaled, **Features Reduced PCA** , TRAIN, TESTING

XGBoost : {'classifier__learning_rate': 0.1, 'classifier__max_depth': 6, 'classifier__min_child_weight': 3, 'classifier__n_estimators': 500}
– scaled : MinMax, Nans – o
Average F1-score : 0.94



	precision	recall	f1-score	support
0	1.00	1.00	1.00	55
1	1.00	1.00	1.00	13
2	0.35	0.95	0.51	56
3	0.96	0.97	0.97	824
4	1.00	1.00	1.00	9
5	0.83	1.00	0.91	25
6	1.00	1.00	1.00	8
7	0.99	1.00	1.00	119
8	0.98	0.82	0.89	97
9	0.92	0.92	0.92	181
10	0.92	0.98	0.95	62
11	0.16	0.97	0.28	62
12	1.00	1.00	1.00	5
13	1.00	0.98	0.99	49
14	1.00	1.00	1.00	10
15	0.99	0.95	0.97	129
16	0.98	1.00	0.99	41
17	1.00	1.00	1.00	52
18	0.98	0.95	0.96	42
19	0.92	0.94	0.93	369
20	1.00	0.82	0.90	44
21	0.62	1.00	0.77	36
22	1.00	0.90	0.95	4369
23	1.00	1.00	1.00	42
24	1.00	1.00	1.00	2
25	0.99	1.00	0.99	99
26	0.89	0.91	0.90	43
27	1.00	1.00	1.00	31
28	1.00	0.81	0.90	37

accuracy			0.92	6911
macro avg	0.91	0.96	0.92	6911
weighted avg	0.97	0.92	0.94	6911

X – Imputed, Scaled, **Features Reduced model selection** , TRAIN, TESTING

The 1000 top features are extracted from the XGBoost model

	precision	recall	f1-score	support
0	1.00	1.00	1.00	55
1	1.00	1.00	1.00	13
2	0.21	0.91	0.34	56
3	0.96	0.97	0.97	824
4	1.00	1.00	1.00	9
5	0.86	1.00	0.93	25
6	1.00	1.00	1.00	8
7	1.00	1.00	1.00	119
8	0.92	0.88	0.90	97
9	0.91	0.91	0.91	181
10	0.83	0.97	0.90	62
11	0.26	0.97	0.41	62
12	1.00	1.00	1.00	5
13	0.98	0.98	0.98	49
14	1.00	1.00	1.00	10
15	0.98	0.96	0.97	129
16	0.98	1.00	0.99	41
17	1.00	1.00	1.00	52
18	0.97	0.93	0.95	42
19	0.92	0.93	0.92	369
20	1.00	0.80	0.89	44
21	0.60	1.00	0.75	36
22	1.00	0.91	0.95	4369
23	1.00	1.00	1.00	42
24	1.00	1.00	1.00	2
25	0.99	0.99	0.99	99
26	0.83	1.00	0.91	43
27	1.00	1.00	1.00	31
28	1.00	0.78	0.88	37

accuracy			0.93	6911
macro avg	0.90	0.96	0.91	6911
weighted avg	0.97	0.93	0.94	6911

The 1000 top features are extracted from the Random Forest model

	precision	recall	f1-score	support
0	1.00	1.00	1.00	55
1	1.00	1.00	1.00	13
2	0.24	0.95	0.38	56
3	0.96	0.97	0.97	824
4	1.00	1.00	1.00	9
5	0.81	1.00	0.89	25
6	1.00	1.00	1.00	8
7	1.00	1.00	1.00	119
8	0.95	0.89	0.91	97
9	0.90	0.92	0.91	181
10	0.87	0.95	0.91	62
11	0.22	0.97	0.35	62
12	1.00	1.00	1.00	5
13	1.00	0.98	0.99	49
14	0.91	1.00	0.95	10
15	0.98	0.95	0.97	129
16	0.98	1.00	0.99	41
17	1.00	1.00	1.00	52
18	1.00	0.93	0.96	42
19	0.92	0.93	0.92	369
20	1.00	0.75	0.86	44
21	0.60	1.00	0.75	36
22	1.00	0.91	0.95	4369
23	1.00	1.00	1.00	42
24	1.00	1.00	1.00	2
25	0.99	1.00	0.99	99
26	0.91	0.98	0.94	43
27	1.00	1.00	1.00	31
28	1.00	0.81	0.90	37

accuracy			0.93	6911
macro avg	0.90	0.96	0.91	6911
weighted avg	0.97	0.93	0.94	6911

X – Imputed, Scaled, **Features Reduced model selection** , TRAIN, TESTING

The 1000 top features are extracted from the XGBoost model

	precision	recall	f1-score	support
0	1.00	1.00	1.00	55
1	1.00	1.00	1.00	13
2	0.21	0.91	0.34	56
3	0.96	0.97	0.97	824
4	1.00	1.00	1.00	9
5	0.86	1.00	0.93	25
6	1.00	1.00	1.00	8
7	1.00	1.00	1.00	119
8	0.92	0.88	0.90	97
9	0.91	0.91	0.91	181
10	0.83	0.97	0.90	62
11	0.26	0.97	0.41	62
12	1.00	1.00	1.00	5
13	0.98	0.98	0.98	49
14	1.00	1.00	1.00	10
15	0.98	0.96	0.97	129
16	0.98	1.00	0.99	41
17	1.00	1.00	1.00	52
18	0.97	0.93	0.95	42
19	0.92	0.93	0.92	369
20	1.00	0.80	0.89	44
21	0.60	1.00	0.75	36
22	1.00	0.91	0.95	4369
23	1.00	1.00	1.00	42
24	1.00	1.00	1.00	2
25	0.99	0.99	0.99	99
26	0.83	1.00	0.91	43
27	1.00	1.00	1.00	31
28	1.00	0.78	0.88	37
accuracy			0.93	6911
macro avg	0.90	0.96	0.91	6911
weighted avg	0.97	0.93	0.94	6911

The 1000 top features are extracted from the Random Forest model

	precision	recall	f1-score	support
0	1.00	1.00	1.00	55
1	1.00	1.00	1.00	13
2	0.24	0.95	0.38	56
3	0.96	0.97	0.97	824
4	1.00	1.00	1.00	9
5	0.81	1.00	0.89	25
6	1.00	1.00	1.00	8
7	1.00	1.00	1.00	119
8	0.95	0.89	0.91	97
9	0.90	0.92	0.91	181
10	0.87	0.95	0.91	62
11	0.22	0.97	0.35	62
12	1.00	1.00	1.00	5
13	1.00	0.98	0.99	49
14	0.91	1.00	0.95	10
15	0.98	0.95	0.97	129
16	0.98	1.00	0.99	41
17	1.00	1.00	1.00	52
18	1.00	0.93	0.96	42
19	0.92	0.93	0.92	369
20	1.00	0.75	0.86	44
21	0.60	1.00	0.75	36
22	1.00	0.91	0.95	4369
23	1.00	1.00	1.00	42
24	1.00	1.00	1.00	2
25	0.99	0.99	0.99	99
26	0.91	0.98	0.94	43
27	1.00	1.00	1.00	31
28	1.00	0.81	0.90	37
accuracy			0.93	6911
macro avg	0.90	0.96	0.91	6911
weighted avg	0.97	0.93	0.94	6911

CONCLUSION

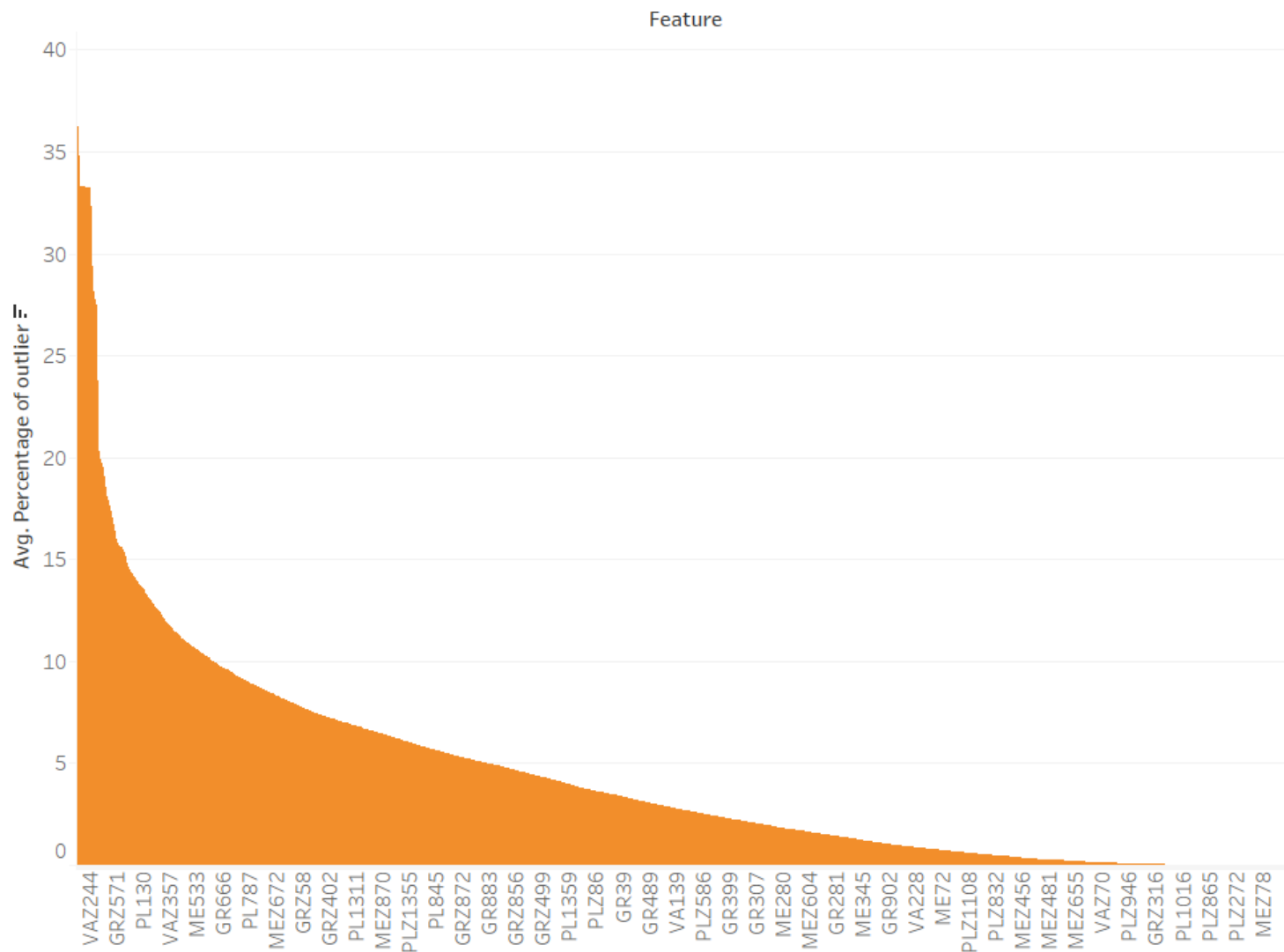
- Necessary EDA was done by handling NaNs, scaling the data
- Dimension reduction was performed to train the dataset.
- Different models were trained on the cluster and XGBoost gave the best results with avg F1 score of 0.94 on test dataset.
- Model Selection was performed using best features from XGBoost and Random Forest. They gave the same average F1 score
- Models were also trained for synthetically generated minority classes

X – a lot of features, NANS, cols with unique value , **Outliers**

Features : 7672 -> 7539

- Outlier – Values IQR

Outliers through IQR



Need to be careful while dealing with dataset

