DS203 ASSIGNMENT
E4

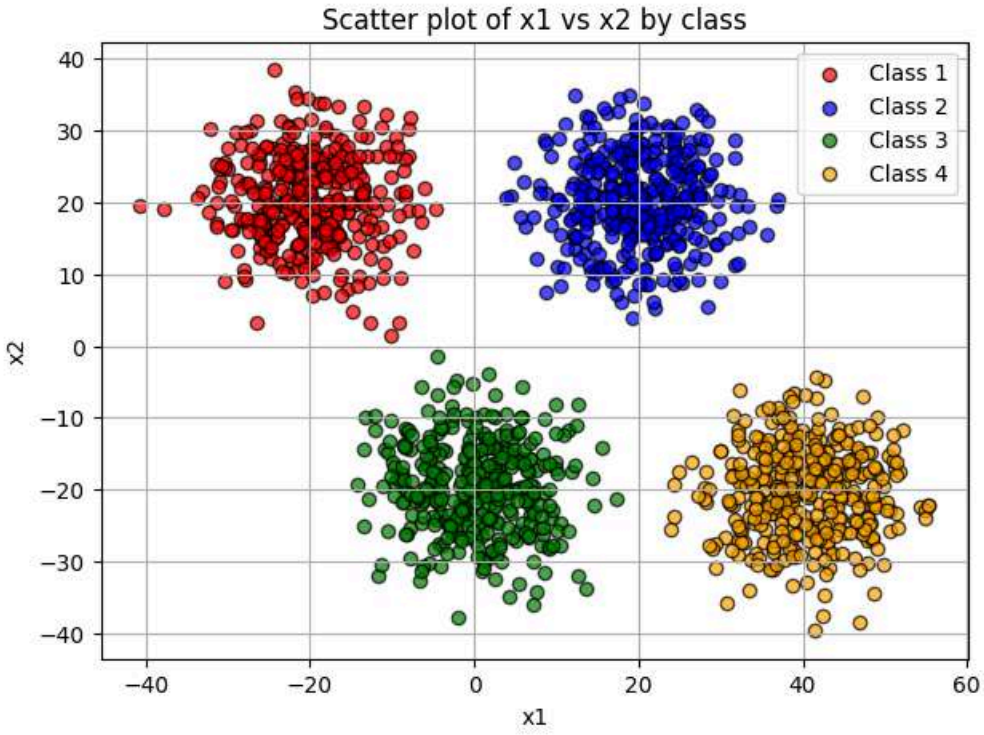
SOURAV SREENATH
24B1841
BTECH-ENGINEERING PHYSICS
INDIAN INSTITUTE OF TECHNOLOGY
BOMBAY

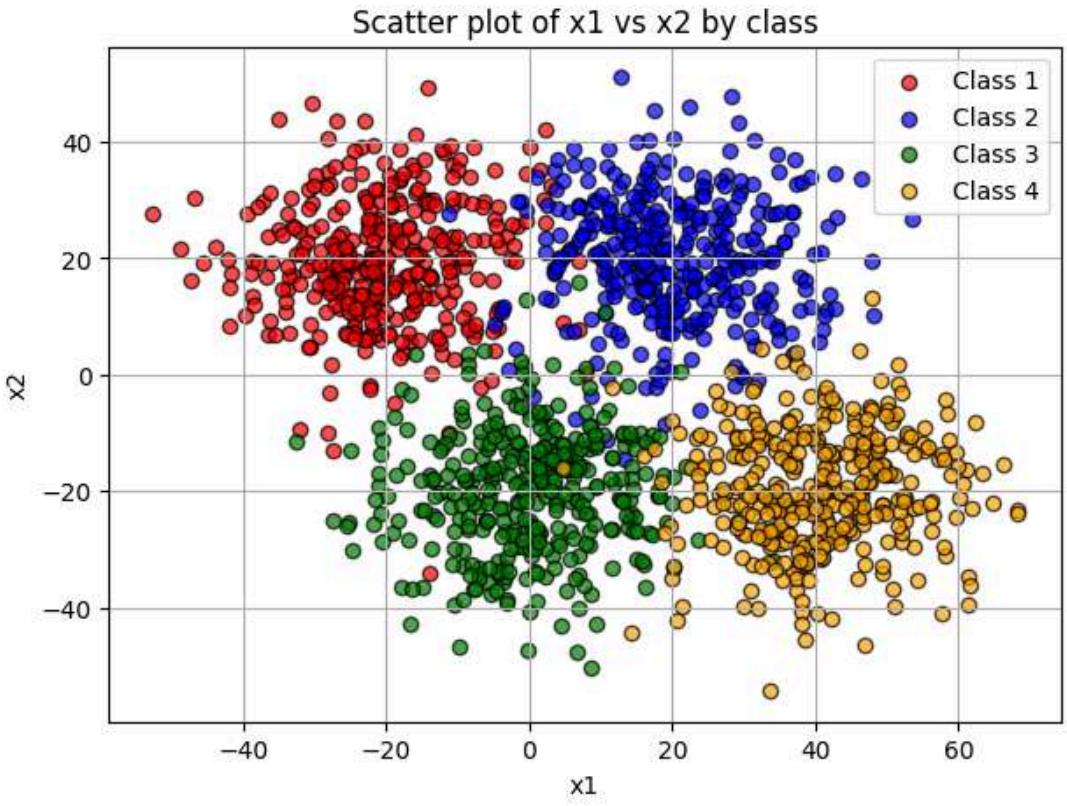
1. **Part 1:**

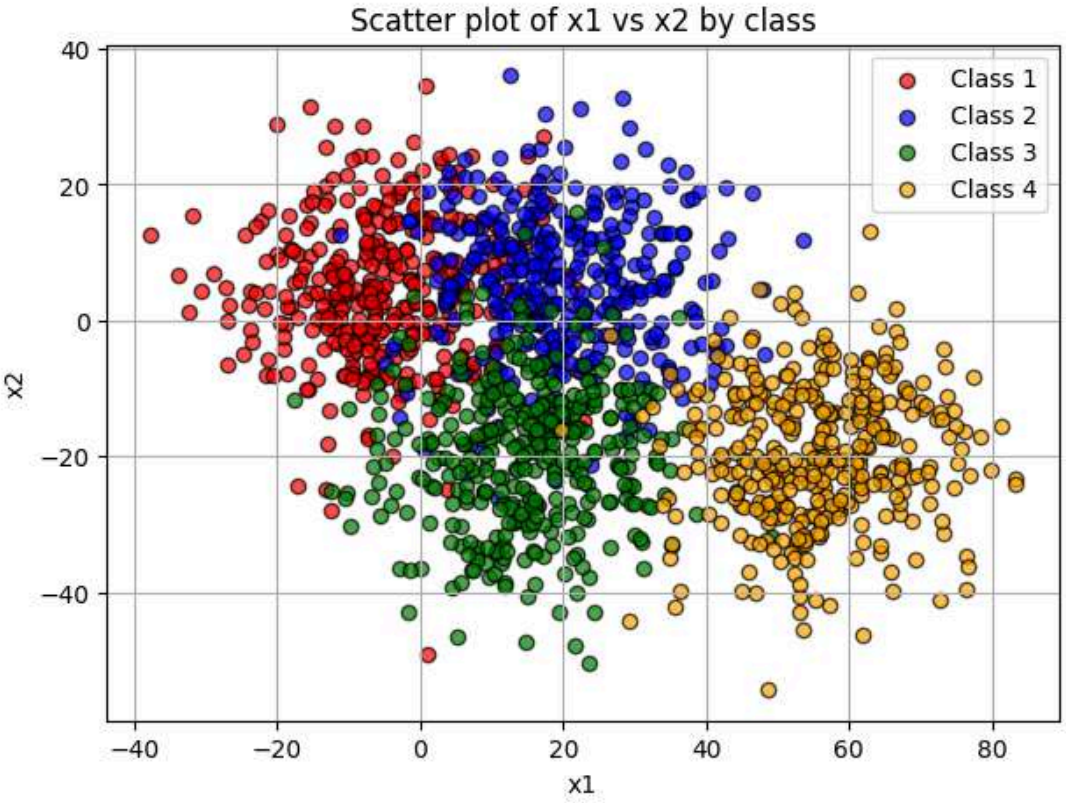
QUESTION 1:

I divided the given 3 datasets into train and test using the sklearn library. I used an 80/20 ratio to choose my train and test datasets. I set some random_states of my choice simply just to introduce some reproducibility into my assignment. An important thing to mention here is, to make the data in the train and test quite similar so that we have a better idea on the weights and all, I have added the stratify argument too, so that the test and train are internally in the classes too, split at around the 80/20 mark.

QUESTION 2:




Scatter plot of x1 vs x2 by class
Class 1
Class 2
Class 3
Class 4
x2
x1
40
20
0
−20
−40
−40
−20
0
20
40
60


Scatter plot of x1 vs x2 by class
Class 1
Class 2
Class 3
Class 4
x2
x1
40
20
0
−20
−40
−40
−20
0
20
40
60
80

There are 3 plots above. The first is the v0 dataset, second v1 and third v2. I'll be following the same usage throughout the assignment analysis.

In the above pictures, its quite clear that the logistic regression will work excellently on v0 compared to v1 and v2 and it will work the worst on v2. Now, why is an easy answer. The v0 set has a clear separation between the points in each class, hence its easy for the model to decide where the line should be. In some sense there can be many lines, which even though not the most optimal, still give a clean separation. But, the same is not true for v1 and v2. In v1, there is some mixing, but it has only started and its not that deep inn too. Only some of the boundary points in each class are getting mixed up with the other classes in some way. This is still fine but logistic regression will perform worser on this dataset than the first because it will have some misclassification, but still not too many. Now comes the worst of them all, v2. The points seem to be very mixed up and its quite difficult for us to even see where a clear boundary lies. This makes it a very difficult challenge for the logistic regression algorithm to choose a good boundary condition without a lot of misclassifications.

Hence, proper working of Logistic Regression:
V0 > V1 > V2

QUESTION 3:

The question only asked us to prepare the models here, but not to analyse them, hence further analysis has been done in the further questions. The code if reviewing is needed on what the models and their parameters are, they are as follows:

```
algorithms = {
    'Logistic Regression': LogisticRegression(max_iter=5000),
    'Logistic Regression + Poly(2)': Pipeline([('poly',
PolynomialFeatures(degree=2)),('logreg',
LogisticRegression(max_iter=10000))]),
    'SVC (linear)': SVC(kernel='linear', probability=True),
    'SVC (rbf)': SVC(kernel='rbf', probability=True),
    'NeuralNet (5)': MLPClassifier(hidden_layer_sizes=(5,),
max_iter=5000),
    'NeuralNet (5,5)': MLPClassifier(hidden_layer_sizes=(5,5),
max_iter=5000),
```

```
    'NeuralNet (5,5,5)': MLPClassifier(hidden_layer_sizes=(5,5,5),
max_iter=5000),
    'NeuralNet (10)': MLPClassifier(hidden_layer_sizes=(10,),
max_iter=5000),
}

for depth in range(2, 6):
    for leaf in range(1, 6):
        name = f'RandomForest d{depth}_l{leaf}'
        algorithms[name] = RandomForestClassifier(max_depth=depth,
min_samples_leaf=leaf)
```

The probability=True has been added to avoid some issues that came when i was making the ROC curves. That and an extra snippet has been taken later to fix issues with SVC not having a built in continuous probability function.

The question asked a small analysis on what linear and rbf meant. Linear, as is obvious from the name, if put forward as the argument, creates all boundaries in between classes as straight lines. Even if there is some complex pattern that the points follow which is followed properly could avoid misclassification, still, linear only chooses the straight line method, which could in theory miss out on some patterns. RBF if toggled, then it uses an exponential function rather than a straight line to map out boundaries. The exponential is not that simple, it has some distance parameters between points, etc which makes the boundary pick up non-linear shapes and patterns in the data. So, for simple usage where the data points are separate, linear would be computationally much more efficient, whereas, in data sets with a lot of mixing of points, RBF can really bring about the pattern of the boundaries.

IMPORTANT: I didnt use the Standard Scaler anywhere because the value of the features was pretty close together and their order of magnitudes were approximately equal.

QUESTION 4:

I have a provision in my code to check the confusion matrices if needed. I have commented out that in the submission as it was not asked, but more clarity could be made from there if needed.

Below are the screen shots of the .csv files i got.

(**NOTE**: I have only used one model for all of the datasets. Hence, I am overwriting them when i go to each new dataset, hence run the code only in the same order as i submitted, else, the models and data asked could mismatch)

	Accuracy	Precision_avg	Precision_1	Precision_2	Precision_3	Precision_4	Recall_avg	Recall_1	Recall_2	Recall_3	Recall_4	F1_avg	F1_1	F1_2	F1_3	F1_4	AUC_avg	AUC_1	AUC_2	AUC_3	AUC_4
Logistic Regression	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Logistic Regression + Poly(2)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
SVC (linear)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
SVC (rbf)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
NeuralNet (5)	0.996528	0.996557629	0.9965398	1	1	0.9896907	0.996528	1	1	0.986111	1	0.996523	0.998267	1	0.993007	0.994819	0.999979	1	1	0.99994	0.999976
NeuralNet (5,5)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
NeuralNet (5,5,5)	0.997396	0.99742268	1	1	1	0.9896907	0.997396	1	1	0.989583	1	0.997396	1	1	0.994764	0.994819	0.999992	1	1	0.999984	0.999984
NeuralNet (10)	0.994792	0.994838896	1	1	0.9964789	0.9828767	0.994792	1	1	0.982639	0.996528	0.994791	1	1	0.98951	0.989655	0.999907	1	1	0.999763	0.999863
RandomForest d2_l1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d2_l2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d2_l3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d2_l4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d2_l5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d3_l1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d3_l2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d3_l3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d3_l4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d3_l5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d4_l1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d4_l2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d4_l3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d4_l4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d4_l5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d5_l1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d5_l2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d5_l3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d5_l4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d5_l5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	Accuracy	Precision_avg	Precision_1	Precision_2	Precision_3	Precision_4	Recall_avg	Recall_1	Recall_2	Recall_3	Recall_4	F1_avg	F1_1	F1_2	F1_3	F1_4	AUC_avg	AUC_1	AUC_2	AUC_3	AUC_4
Logistic Regression	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Logistic Regression + Poly(2)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
SVC (linear)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
SVC (rbf)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
NeuralNet (5)	0.996528	0.996575342	1	1	1	0.98630137	0.9965278	1	1	0.986111	1	0.996528	1	1	0.993007	0.993103	1	1	1	1	1
NeuralNet (5,5)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
NeuralNet (5,5,5)	0.996528	0.996575342	1	1	1	0.98630137	0.9965278	1	1	0.986111	1	0.996528	1	1	0.993007	0.993103	1	1	1	1	1
NeuralNet (10)	0.996528	0.996575342	1	1	1	0.98630137	0.9965278	1	1	0.986111	1	0.996528	1	1	0.993007	0.993103	1	1	1	1	1
RandomForest d2_l1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d2_l2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d2_l3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d2_l4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d2_l5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d3_l1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d3_l2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d3_l3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d3_l4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d3_l5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d4_l1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d4_l2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d4_l3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d4_l4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d4_l5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d5_l1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d5_l2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d5_l3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d5_l4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RandomForest d5_l5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The above are the statistics for the Train(1st) and Test(2nd) data of the dataset v0.

	Accuracy	Precision_avg	Precision_1	Precision_2	Precision_3	Precision_4	Recall_avg	Recall_1	Recall_2	Recall_3	Recall_4	F1_avg	F1_1	F1_2	F1_3	F1_4	AUC_avg	AUC_1	AUC_2	AUC_3	AUC_4
Logistic Regression	0.954861	0.954987611	0.95833333	0.95155709	0.9347079	0.9753521	0.954861	0.958333	0.954861	0.944444	0.961806	0.954905	0.958333	0.953206	0.939551	0.968531	0.996281	0.996467	0.996733	0.994016	0.997906
Logistic Regression + Poly(2)	0.957465	0.957780585	0.96503497	0.96153846	0.92567568	0.9788732	0.957465	0.958333	0.954861	0.951389	0.965278	0.957561	0.961672	0.958188	0.938356	0.972028	0.996671	0.997561	0.996721	0.994253	0.998151
SVC (linear)	0.956597	0.956708072	0.96167247	0.95470383	0.93493151	0.9755245	0.956597	0.958333	0.951389	0.947917	0.96875	0.956637	0.96	0.953043	0.941379	0.972125	0.995926	0.995133	0.996713	0.993916	0.997942
SVC (rbf)	0.955729	0.956096783	0.96126761	0.95155709	0.92929293	0.9822695	0.955729	0.947917	0.954861	0.958333	0.961806	0.955818	0.954545	0.953206	0.94359	0.97193	0.996389	0.997476	0.996568	0.994052	0.99746
NeuralNet (5)	0.954861	0.955357273	0.96842105	0.96071429	0.92026578	0.972028	0.954861	0.958333	0.934028	0.961806	0.965278	0.954938	0.963351	0.947183	0.940577	0.968641	0.995079	0.996005	0.993972	0.99324	0.997098
NeuralNet (5,5)	0.953125	0.95386482	0.96842105	0.97090909	0.91419142	0.9619377	0.953125	0.958333	0.927083	0.961806	0.965278	0.95321	0.963351	0.94849	0.937394	0.963605	0.994565	0.994836	0.993269	0.992509	0.997645
NeuralNet (5,5,5)	0.958333	0.958422106	0.95547945	0.96797153	0.94137931	0.9688581	0.958333	0.96875	0.944444	0.947917	0.972222	0.958327	0.962069	0.956063	0.944637	0.970537	0.996445	0.997267	0.996881	0.993297	0.998336
NeuralNet (10)	0.949653	0.949759812	0.96491228	0.95422535	0.93127148	0.9486301	0.949653	0.954861	0.940972	0.940972	0.961806	0.94967	0.95986	0.947552	0.936097	0.955172	0.996303	0.997247	0.996013	0.994366	0.997585
RandomForest d2_I1	0.958333	0.958423229	0.96491228	0.96478873	0.93835616	0.9656357	0.958333	0.954861	0.951389	0.951389	0.975694	0.958342	0.95986	0.958042	0.944828	0.970639	0.994521	0.995549	0.993431	0.991155	0.99795
RandomForest d2_I2	0.958333	0.958565035	0.96819788	0.96466431	0.93243243	0.9689655	0.958333	0.951389	0.947917	0.958333	0.975694	0.958365	0.95972	0.956217	0.945205	0.972318	0.993646	0.993482	0.991	0.992328	0.997774
RandomForest d2_I3	0.957465	0.957507173	0.95833333	0.96453901	0.94482759	0.9623288	0.957465	0.958333	0.944444	0.951389	0.975694	0.957445	0.958333	0.954386	0.948097	0.968966	0.994546	0.995929	0.99352	0.990867	0.997866
RandomForest d2_I4	0.959201	0.959417872	0.96491228	0.97153025	0.93559322	0.9656357	0.959201	0.954861	0.947917	0.958333	0.975694	0.959226	0.95986	0.959578	0.946827	0.970639	0.993996	0.994952	0.991725	0.991741	0.997565
RandomForest d2_I5	0.960069	0.960165338	0.96503497	0.96478873	0.94197952	0.9688581	0.960069	0.958333	0.951389	0.958333	0.972222	0.960084	0.961672	0.958042	0.950086	0.970537	0.994773	0.996755	0.992724	0.991753	0.997858
RandomForest d3_I1	0.961806	0.962031047	0.96819788	0.96842105	0.93581081	0.9756944	0.961806	0.951389	0.958333	0.961806	0.975694	0.961849	0.95972	0.963351	0.94863	0.975694	0.996541	0.997297	0.996029	0.994858	0.997979
RandomForest d3_I2	0.960069	0.960295902	0.96830986	0.96153846	0.93581081	0.9755245	0.960069	0.954861	0.954861	0.961806	0.96875	0.960121	0.961538	0.958188	0.94863	0.972125	0.996718	0.997715	0.996942	0.993761	0.998453
RandomForest d3_I3	0.961806	0.961987819	0.96503497	0.96491228	0.93898305	0.979021	0.961806	0.958333	0.954861	0.961806	0.972222	0.96185	0.961672	0.95986	0.950257	0.97561	0.996378	0.9971	0.995714	0.994508	0.99819
RandomForest d3_I4	0.960938	0.961208955	0.96491228	0.96503497	0.93243243	0.9824561	0.960938	0.954861	0.958333	0.958333	0.972222	0.961013	0.95986	0.961672	0.945205	0.977312	0.996338	0.997108	0.995696	0.99399	0.998557
RandomForest d3_I5	0.960069	0.960363905	0.96830986	0.96153846	0.93265993	0.9789474	0.960069	0.954861	0.954861	0.961806	0.96875	0.960139	0.961538	0.958188	0.947009	0.973822	0.996608	0.997633	0.995772	0.994474	0.998553
RandomForest d4_I1	0.963542	0.963570254	0.96842105	0.96180556	0.94827586	0.9757785	0.963542	0.958333	0.961806	0.954861	0.979167	0.963546	0.963351	0.961806	0.951557	0.97747	0.997901	0.998497	0.998019	0.99616	0.998929
RandomForest d4_I2	0.965278	0.965423604	0.96842105	0.97192982	0.94217687	0.9791667	0.965278	0.958333	0.961806	0.961806	0.979167	0.965312	0.963351	0.966841	0.95189	0.979167	0.998009	0.998686	0.997928	0.996823	0.998597
RandomForest d4_I3	0.963542	0.963593394	0.96842105	0.96515679	0.94501718	0.9757785	0.963542	0.958333	0.961806	0.954861	0.979167	0.963553	0.963351	0.963478	0.949914	0.97747	0.997542	0.998457	0.996753	0.995901	0.999056
RandomForest d4_I4	0.96441	0.964622078	0.96515679	0.97183099	0.93898305	0.9825175	0.96441	0.961806	0.958333	0.961806	0.975694	0.964466	0.963478	0.965035	0.950257	0.979094	0.997879	0.998634	0.997677	0.996435	0.99877
RandomForest d4_I5	0.963542	0.96375612	0.96830986	0.97183099	0.93918919	0.9756944	0.963542	0.954861	0.958333	0.965278	0.975694	0.963581	0.961538	0.965035	0.952055	0.975694	0.997703	0.998525	0.997826	0.995859	0.998603
RandomForest d5_I1	0.967014	0.967089061	0.97202797	0.96853147	0.94863014	0.9791667	0.967014	0.965278	0.961806	0.961806	0.979167	0.967034	0.968641	0.965157	0.955172	0.979167	0.999041	0.999333	0.998963	0.998312	0.999554
RandomForest d5_I2	0.967882	0.96791383	0.9754386	0.96180556	0.95517241	0.9792388	0.967882	0.965278	0.961806	0.961806	0.982639	0.967888	0.970332	0.961806	0.958478	0.980936	0.998821	0.999196	0.998658	0.998051	0.999381
RandomForest d5_I3	0.965278	0.965380987	0.97192982	0.96503497	0.94539249	0.9791667	0.965278	0.961806	0.958333	0.961806	0.979167	0.965302	0.966841	0.961672	0.953528	0.979167	0.998598	0.999019	0.9982	0.997814	0.999361
RandomForest d5_I4	0.963542	0.963701375	0.97183099	0.95847751	0.94197952	0.9825175	0.963542	0.958333	0.961806	0.958333	0.975694	0.963588	0.965035	0.960139	0.950086	0.979094	0.99837	0.998843	0.998184	0.997378	0.999076
RandomForest d5_I5	0.96441	0.964583227	0.96503497	0.97183099	0.94237288	0.9790941	0.96441	0.958333	0.958333	0.965278	0.975694	0.964447	0.961672	0.965035	0.953688	0.977391	0.998346	0.998814	0.998155	0.997291	0.999124

	Accuracy	Precision_avg	Precision_1	Precision_2	Precision_3	Precision_4	Recall_avg	Recall_1	Recall_2	Recall_3	Recall_4	F1_avg	F1_1	F1_2	F1_3	F1_4	AUC_avg	AUC_1	AUC_2	AUC_3	AUC_4
Logistic Regression	0.934028	0.935112256	0.9571429	0.88157895	0.95652174	0.9452055	0.934028	0.930556	0.930556	0.916667	0.958333	0.93424	0.943662	0.905405	0.93617	0.951724	0.99439	0.997685	0.992863	0.991127	0.995885
Logistic Regression + Poly(2)	0.9375	0.938201758	0.9577465	0.89333333	0.95652174	0.9452055	0.9375	0.944444	0.930556	0.916667	0.958333	0.937627	0.951049	0.911565	0.93617	0.951724	0.994325	0.997042	0.993506	0.990419	0.996335
SVC (linear)	0.930556	0.931759147	0.9571429	0.88157895	0.95588235	0.9324324	0.930556	0.930556	0.930556	0.902778	0.958333	0.930711	0.943662	0.905405	0.928571	0.945205	0.994068	0.997557	0.992798	0.989969	0.995949
SVC (rbf)	0.9375	0.938884957	0.9710145	0.88311688	0.94366197	0.9577465	0.9375	0.930556	0.944444	0.930556	0.944444	0.937805	0.950355	0.912752	0.937063	0.951049	0.992509	0.996335	0.991577	0.987204	0.99492
NeuralNet (5)	0.9375	0.938235969	0.9571429	0.89333333	0.95652174	0.9459459	0.9375	0.930556	0.930556	0.916667	0.972222	0.937575	0.943662	0.911565	0.93617	0.958904	0.991721	0.994856	0.990162	0.98669	0.995177
NeuralNet (5,5)	0.940972	0.941472462	0.9710145	0.90410959	0.93243243	0.9583333	0.940972	0.930556	0.916667	0.958333	0.958333	0.94106	0.950355	0.910345	0.945205	0.958333	0.992734	0.994406	0.990934	0.989262	0.996335
NeuralNet (5,5,5)	0.930556	0.931540827	0.9565217	0.88157895	0.94285714	0.9452055	0.930556	0.916667	0.930556	0.916667	0.958333	0.930719	0.93617	0.905405	0.929577	0.951724	0.992654	0.996399	0.99177	0.986561	0.995885
NeuralNet (10)	0.927083	0.92799031	0.9571429	0.88157895	0.92957746	0.943662	0.927083	0.930556	0.930556	0.916667	0.930556	0.927302	0.943662	0.905405	0.923077	0.937063	0.99267	0.996914	0.991962	0.987847	0.993956
RandomForest d2_I1	0.916667	0.917483118	0.9552239	0.90540541	0.90140845	0.9078947	0.916667	0.888889	0.930556	0.888889	0.958333	0.916552	0.920863	0.917808	0.895105	0.932432	0.989776	0.996721	0.986497	0.981771	0.994117
RandomForest d2_I2	0.923611	0.924414078	0.9552239	0.89333333	0.91666667	0.9324324	0.923611	0.888889	0.930556	0.916667	0.958333	0.923575	0.920863	0.911565	0.916667	0.945205	0.988482	0.995628	0.978234	0.984761	0.995306
RandomForest d2_I3	0.923611	0.924339528	0.9565217	0.90540541	0.92753623	0.9078947	0.923611	0.916667	0.930556	0.888889	0.958333	0.923553	0.93617	0.917808	0.907801	0.932432	0.990419	0.996206	0.987172	0.982639	0.99566
RandomForest d2_I4	0.923611	0.924195179	0.9558824	0.90540541	0.91549296	0.92	0.923611	0.902778	0.930556	0.902778	0.958333	0.923562	0.928571	0.917808	0.909091	0.938776	0.988619	0.996624	0.983218	0.982092	0.992541
RandomForest d2_I5	0.930556	0.93098426	0.9565217	0.90540541	0.92957746	0.9324324	0.930556	0.916667	0.930556	0.916667	0.958333	0.930565	0.93617	0.917808	0.923077	0.945205	0.990966	0.997042	0.982446	0.989069	0.995306
RandomForest d3_I1	0.923611	0.924414078	0.9552239	0.89333333	0.91666667	0.9324324	0.923611	0.888889	0.930556	0.916667	0.958333	0.923575	0.920863	0.911565	0.916667	0.945205	0.989439	0.997042	0.980067	0.985082	0.995563
RandomForest d3_I2	0.920139	0.921105202	0.9545455	0.89333333	0.90410959	0.9324324	0.920139	0.875	0.930556	0.916667	0.958333	0.92004	0.913043	0.911565	0.910345	0.945205	0.992123	0.996142	0.987783	0.988426	0.996142
RandomForest d3_I3	0.920139	0.921044862	0.9558824	0.88157895	0.91428571	0.9324324	0.920139	0.902778	0.930556	0.888889	0.958333	0.920148	0.928571	0.905405	0.901408	0.945205	0.989069	0.997235	0.978909	0.984568	0.995563
RandomForest d3_I4	0.923611	0.924292827	0.9552239	0.90540541	0.90410959	0.9324324	0.923611	0.888889	0.930556	0.916667	0.958333	0.923555	0.920863	0.917808	0.910345	0.945205	0.989149	0.996914	0.978909	0.985468	0.995306
RandomForest d3_I5	0.923611	0.924414078	0.9552239	0.89333333	0.91666667	0.9324324	0.923611	0.888889	0.930556	0.916667	0.958333	0.923575	0.920863	0.911565	0.916667	0.945205	0.988924	0.996399	0.97981	0.984504	0.994985
RandomForest d4_I1	0.916667	0.918017974	0.9552239	0.87012987	0.91428571	0.9324324	0.916667	0.888889	0.930556	0.888889	0.958333	0.916702	0.920863	0.899329	0.901408	0.945205	0.992413	0.996721	0.990162	0.987269	0.995499
RandomForest d4_I2	0.920139	0.921105202	0.9545455	0.89333333	0.90410959	0.9324324	0.920139	0.875	0.930556	0.916667	0.958333	0.92004	0.913043	0.911565	0.910345	0.945205	0.991593	0.996463	0.988233	0.986175	0.995499
RandomForest d4_I3	0.920139	0.921105202	0.9545455	0.89333333	0.90410959	0.9324324	0.920139	0.875	0.930556	0.916667	0.958333	0.92004	0.913043	0.911565	0.910345	0.945205	0.988008	0.995949	0.97698	0.983861	0.995242
RandomForest d4_I4	0.934028	0.934459459	0.9571429	0.90540541	0.94285714	0.9324324	0.934028	0.930556	0.930556	0.916667	0.958333	0.934063	0.943662	0.917808	0.929577	0.945205	0.991753	0.995242	0.989841	0.98669	0.995242
RandomForest d4_I5	0.920139	0.921105202	0.9545455	0.89333333	0.90410959	0.9324324	0.920139	0.875	0.930556	0.916667	0.958333	0.92004	0.913043	0.911565	0.910345	0.945205	0.992284	0.996013	0.990226	0.98759	0.995306
RandomForest d5_I1	0.923611	0.924414078	0.9552239	0.89333333	0.91666667	0.9324324	0.923611	0.888889	0.930556	0.916667	0.958333	0.923575	0.920863	0.911565	0.916667	0.945205	0.989358	0.995628	0.983989	0.984761	0.993056
RandomForest d5_I2	0.927083	0.927806396	0.9558824	0.89333333	0.92957746	0.9324324	0.927083	0.902778	0.930556	0.916667	0.958333	0.927105	0.928571	0.911565	0.923077	0.945205	0.99177	0.995435	0.988426	0.987912	0.995306
RandomForest d5_I3	0.923611	0.924285269	0.9558824	0.89333333	0.91549296	0.9324324	0.923611	0.902778	0.930556	0.902778	0.958333	0.923608	0.928571	0.911565	0.909091	0.945205	0.988297	0.994342	0.982446	0.982317	0.994084
RandomForest d5_I4	0.920139	0.921105202	0.9545455	0.89333333	0.90410959	0.9324324	0.920139	0.875	0.930556	0.916667	0.958333	0.92004	0.913043	0.911565	0.910345	0.945205	0.990435	0.99582	0.984568	0.986883	0.99447
RandomForest d5_I5	0.923611	0.924414078	0.9552239	0.89333333	0.91666667	0.9324324	0.923611	0.888889	0.930556	0.916667	0.958333	0.923575	0.920863	0.911565	0.916667	0.945205	0.991094	0.99537	0.989326	0.985532	0.994149

The above are the statistics for the Train(1st) and Test(2nd) data of the dataset v1.

	Accuracy	Precision_avg	Precision_1	Precision_2	Precision_3	Precision_4	Recall_avg	Recall_1	Recall_2	Recall_3	Recall_4	F1_avg	F1_1	F1_2	F1_3	F1_4	AUC_avg	AUC_1	AUC_2	AUC_3	AUC_4
Logistic Regression	0.860243	0.859772311	0.8501742	0.79928315	0.8344595	0.95517241	0.860243	0.847222	0.774306	0.857639	0.961806	0.859915	0.848696	0.786596	0.84589	0.958478	0.968768	0.971905	0.942962	0.962891	0.997315
Logistic Regression + Poly(2)	0.861111	0.860980187	0.8491228	0.80656934	0.8229508	0.96527778	0.861111	0.840278	0.767361	0.871528	0.965278	0.860744	0.844677	0.786477	0.846543	0.965278	0.969447	0.972817	0.944344	0.963208	0.99742
SVC (linear)	0.861979	0.861744225	0.8521127	0.80071174	0.8322148	0.96193772	0.861979	0.840278	0.78125	0.861111	0.965278	0.861759	0.846154	0.790861	0.846416	0.963605	0.968235	0.970904	0.942122	0.962649	0.997263
SVC (rbf)	0.861979	0.862204427	0.858156	0.81021898	0.8187702	0.96167247	0.861979	0.840278	0.770833	0.878472	0.958333	0.861682	0.849123	0.790036	0.847571	0.96	0.969156	0.973227	0.943263	0.963321	0.996813
NeuralNet (5)	0.861111	0.862299751	0.8576512	0.81454545	0.8089172	0.96808511	0.861111	0.836806	0.777778	0.881944	0.947917	0.861146	0.8471	0.795737	0.843854	0.957895	0.968787	0.971865	0.944818	0.961163	0.997303
NeuralNet (5,5)	0.853299	0.854345292	0.8726592	0.78671329	0.8064516	0.95155709	0.853299	0.809028	0.78125	0.868056	0.954861	0.853235	0.83964	0.783972	0.83612	0.953206	0.969739	0.97184	0.945831	0.963956	0.997328
NeuralNet (5,5,5)	0.858507	0.858304957	0.8430034	0.7919708	0.8333333	0.96491228	0.858507	0.857639	0.753472	0.868056	0.954861	0.858175	0.850258	0.772242	0.85034	0.95986	0.968201	0.972777	0.944344	0.959133	0.996552
NeuralNet (10)	0.861111	0.861411156	0.8571429	0.8057554	0.8175896	0.96515679	0.861111	0.833333	0.777778	0.871528	0.961806	0.860941	0.84507	0.791519	0.843697	0.963478	0.969491	0.972262	0.944991	0.963236	0.997476
RandomForest d2_l1	0.861111	0.862958784	0.8721805	0.77591973	0.8283828	0.97535211	0.861111	0.805556	0.805556	0.871528	0.961806	0.861486	0.837545	0.79046	0.849408	0.968531	0.957138	0.95716	0.922422	0.954369	0.994601
RandomForest d2_l2	0.859375	0.862191706	0.8901961	0.76143791	0.8283828	0.96875	0.859375	0.788194	0.809028	0.871528	0.96875	0.859691	0.836096	0.784512	0.849408	0.96875	0.955325	0.954984	0.918415	0.953987	0.993916
RandomForest d2_l3	0.861979	0.865158909	0.8837209	0.76699029	0.8311258	0.97879859	0.861979	0.791667	0.822917	0.871528	0.961806	0.862552	0.835165	0.79397	0.850847	0.970228	0.955166	0.957909	0.923328	0.944977	0.99445
RandomForest d2_l4	0.860243	0.86371814	0.8937008	0.75641026	0.8361204	0.96864111	0.860243	0.788194	0.819444	0.868056	0.965278	0.860763	0.837638	0.786667	0.851789	0.966957	0.960049	0.962983	0.923798	0.958633	0.994784
RandomForest d2_l5	0.859375	0.863926496	0.902834	0.75078864	0.8333333	0.96875	0.859375	0.774306	0.826389	0.868056	0.96875	0.859878	0.833645	0.786777	0.85034	0.96875	0.961403	0.966708	0.925094	0.958755	0.995053
RandomForest d3_l1	0.862847	0.86577728	0.8821293	0.76948052	0.8327759	0.9787234	0.862847	0.805556	0.822917	0.864583	0.958333	0.863552	0.842105	0.795302	0.848382	0.968421	0.964366	0.969184	0.930256	0.961504	0.996518
RandomForest d3_l2	0.863715	0.867875817	0.8972332	0.75873016	0.8366667	0.97887324	0.863715	0.788194	0.829861	0.871528	0.965278	0.864415	0.839187	0.792703	0.853741	0.972028	0.96529	0.970753	0.930011	0.964629	0.995766
RandomForest d3_l3	0.862847	0.868245811	0.9068826	0.75389408	0.8333333	0.97887324	0.862847	0.777778	0.840278	0.868056	0.965278	0.863624	0.837383	0.794745	0.85034	0.972028	0.965651	0.970695	0.930933	0.964458	0.996516
RandomForest d3_l4	0.864583	0.868047488	0.8945313	0.76038339	0.8417508	0.97552448	0.864583	0.795139	0.826389	0.868056	0.96875	0.865188	0.841912	0.792013	0.854701	0.972125	0.96529	0.969725	0.931643	0.963799	0.995993
RandomForest d3_l5	0.862847	0.86697405	0.8937008	0.75873016	0.8366667	0.97879859	0.862847	0.788194	0.829861	0.871528	0.961806	0.863578	0.837638	0.792703	0.853741	0.970228	0.965337	0.970788	0.931906	0.962507	0.996146
RandomForest d4_l1	0.87066	0.873942528	0.8931298	0.76996805	0.8503401	0.98233216	0.87066	0.8125	0.836806	0.868056	0.965278	0.871436	0.850909	0.801997	0.859107	0.97373	0.973916	0.976462	0.950203	0.970765	0.998234
RandomForest d4_l2	0.871528	0.87385458	0.8876404	0.78032787	0.8451178	0.98233216	0.871528	0.822917	0.826389	0.871528	0.965278	0.872151	0.854054	0.802698	0.85812	0.97373	0.972332	0.975592	0.946474	0.969755	0.997506
RandomForest d4_l3	0.868056	0.871767898	0.8949416	0.76751592	0.8422819	0.98233216	0.868056	0.798611	0.836806	0.871528	0.965278	0.868772	0.844037	0.800664	0.856655	0.97373	0.972704	0.974125	0.947507	0.971782	0.997404
RandomForest d4_l4	0.869792	0.871776244	0.8782288	0.77960526	0.8469388	0.98233216	0.869792	0.826389	0.822917	0.864583	0.965278	0.870399	0.851521	0.800676	0.85567	0.97373	0.972762	0.975731	0.947744	0.969829	0.997745
RandomForest d4_l5	0.869792	0.87289639	0.8957529	0.76996805	0.8503401	0.97552448	0.869792	0.805556	0.836806	0.868056	0.96875	0.870373	0.848263	0.801997	0.859107	0.972125	0.971258	0.974286	0.942911	0.970126	0.997707
RandomForest d5_l1	0.87934	0.881865126	0.8951311	0.78387097	0.862543	0.98591549	0.87934	0.829861	0.84375	0.871528	0.972222	0.880001	0.861261	0.812709	0.867012	0.979021	0.978672	0.979999	0.957835	0.977963	0.998893
RandomForest d5_l2	0.875868	0.878554496	0.8913858	0.77884615	0.8615917	0.98239437	0.875868	0.826389	0.84375	0.864583	0.96875	0.876567	0.857658	0.81	0.863085	0.975524	0.979075	0.980358	0.960128	0.977224	0.998591
RandomForest d5_l3	0.875	0.87711841	0.8827839	0.78431373	0.8591065	0.9822695	0.875	0.836806	0.833333	0.868056	0.961806	0.875687	0.85918	0.808081	0.863558	0.97193	0.978519	0.980053	0.959975	0.975835	0.998212
RandomForest d5_l4	0.873264	0.875238924	0.8847584	0.78501629	0.8522337	0.97894737	0.873264	0.826389	0.836806	0.861111	0.96875	0.873783	0.854578	0.810084	0.856649	0.973822	0.977956	0.978958	0.959105	0.976034	0.997725
RandomForest d5_l5	0.873264	0.875436095	0.8847584	0.78431373	0.8503401	0.98233216	0.873264	0.826389	0.833333	0.868056	0.965278	0.873874	0.854578	0.808081	0.859107	0.97373	0.977558	0.979133	0.957767	0.975055	0.998278

	Accuracy	Precision_avg	Precision_1	Precision_2	Precision_3	Precision_4	Recall_avg	Recall_1	Recall_2	Recall_3	Recall_4	F1_avg	F1_1	F1_2	F1_3	F1_4	AUC_avg	AUC_1	AUC_2	AUC_3	AUC_4
Logistic Regression	0.878472	0.877209128	0.8767123	0.8382353	0.8472222	0.9466667	0.8784722	0.888889	0.791667	0.847222	0.986111	0.877563	0.882759	0.814286	0.847222	0.965986	0.977623	0.978395	0.968493	0.965599	0.998007
Logistic Regression + Poly(2)	0.881944	0.882177326	0.875	0.8636364	0.8311688	0.9589041	0.8819444	0.875	0.791667	0.888889	0.972222	0.881416	0.875	0.826087	0.85906	0.965517	0.976498	0.975694	0.968686	0.963477	0.998135
SVC (linear)	0.878472	0.877304257	0.875	0.8382353	0.8493151	0.9466667	0.8784722	0.875	0.791667	0.861111	0.986111	0.877611	0.875	0.814286	0.855172	0.965986	0.977334	0.978202	0.968428	0.96457	0.998135
SVC (rbf)	0.881944	0.882854633	0.884058	0.8656716	0.8227848	0.9589041	0.8819444	0.847222	0.805556	0.902778	0.972222	0.881556	0.865248	0.834532	0.860927	0.965517	0.976161	0.977302	0.9692	0.961291	0.996849
NeuralNet (5)	0.881944	0.882125176	0.8857143	0.8656716	0.8311688	0.9459459	0.8819444	0.861111	0.805556	0.888889	0.972222	0.881434	0.873239	0.834532	0.85906	0.958904	0.976225	0.977173	0.967078	0.962706	0.997942
NeuralNet (5,5)	0.888889	0.888742428	0.8985507	0.8676471	0.8421053	0.9466667	0.8888889	0.861111	0.819444	0.888889	0.986111	0.888285	0.879433	0.842857	0.864865	0.965986	0.977157	0.975502	0.966885	0.9683	0.997942
NeuralNet (5,5,5)	0.868056	0.867125122	0.8533333	0.8382353	0.8309859	0.9459459	0.8680556	0.888889	0.791667	0.819444	0.972222	0.867278	0.870748	0.814286	0.825175	0.958904	0.975453	0.976466	0.965985	0.96142	0.997942
NeuralNet (10)	0.878472	0.878878423	0.8857143	0.8656716	0.8181818	0.9459459	0.8784722	0.861111	0.805556	0.875	0.972222	0.878078	0.873239	0.834532	0.845638	0.958904	0.977527	0.978459	0.967914	0.965985	0.997749
RandomForest d2_l1	0.854167	0.854996535	0.8769231	0.7837838	0.8133333	0.9459459	0.8541667	0.791667	0.805556	0.847222	0.972222	0.853868	0.832117	0.794521	0.829932	0.958904	0.95925	0.954507	0.939043	0.951067	0.99238
RandomForest d2_l2	0.857639	0.860089243	0.9047619	0.7763158	0.8133333	0.9459459	0.8576389	0.791667	0.819444	0.847222	0.972222	0.857644	0.844444	0.797297	0.829932	0.958904	0.957722	0.952418	0.939397	0.947209	0.991866
RandomForest d2_l3	0.861111	0.863084032	0.890625	0.7894737	0.8133333	0.9589041	0.8611111	0.791667	0.833333	0.847222	0.972222	0.861124	0.838235	0.810811	0.829932	0.965517	0.956806	0.953961	0.940233	0.941583	0.991448
RandomForest d2_l4	0.864583	0.867531493	0.9206349	0.7792208	0.8243243	0.9459459	0.8645833	0.805556	0.833333	0.847222	0.972222	0.864787	0.859259	0.805369	0.835616	0.958904	0.961934	0.958365	0.940811	0.957015	0.991544
RandomForest d2_l5	0.864583	0.867531493	0.9206349	0.7792208	0.8243243	0.9459459	0.8645833	0.805556	0.833333	0.847222	0.972222	0.864787	0.859259	0.805369	0.835616	0.958904	0.963381	0.963059	0.942773	0.954861	0.992831
RandomForest d3_l1	0.868056	0.86954691	0.9076923	0.8026316	0.8219178	0.9459459	0.8680556	0.819444	0.847222	0.833333	0.972222	0.868032	0.861314	0.824324	0.827586	0.958904	0.965535	0.967528	0.941197	0.962834	0.99058
RandomForest d3_l2	0.868056	0.871478659	0.9206349	0.7820513	0.8243243	0.9589041	0.8680556	0.805556	0.847222	0.847222	0.972222	0.868432	0.859259	0.813333	0.835616	0.965517	0.968718	0.972447	0.944219	0.963799	0.994406
RandomForest d3_l3	0.868056	0.871478659	0.9206349	0.7820513	0.8243243	0.9589041	0.8680556	0.805556	0.847222	0.847222	0.972222	0.868432	0.859259	0.813333	0.835616	0.965517	0.966805	0.970711	0.942451	0.962609	0.991448
RandomForest d3_l4	0.871528	0.874611707	0.921875	0.7820513	0.8356164	0.9589041	0.8715278	0.819444	0.847222	0.847222	0.972222	0.871969	0.867647	0.813333	0.841379	0.965517	0.967994	0.972287	0.94512	0.961645	0.992927
RandomForest d3_l5	0.868056	0.870421557	0.90625	0.7922078	0.8243243	0.9589041	0.8680556	0.805556	0.847222	0.847222	0.972222	0.868217	0.852941	0.818792	0.835616	0.965517	0.970293	0.974441	0.949267	0.963252	0.994213
RandomForest d4_l1	0.871528	0.873605162	0.9076923	0.7922078	0.8356164	0.9589041	0.8715278	0.819444	0.847222	0.847222	0.972222	0.871751	0.861314	0.818792	0.841379	0.965517	0.973106	0.975984	0.957883	0.963863	0.994695
RandomForest d4_l2	0.868056	0.869291957	0.8939394	0.8	0.8243243	0.9589041	0.8680556	0.819444	0.833333	0.847222	0.972222	0.868133	0.855072	0.816327	0.835616	0.965517	0.974232	0.97608	0.960069	0.965953	0.994824
RandomForest d4_l3	0.871528	0.87338808	0.9076923	0.8026316	0.8243243	0.9589041	0.8715278	0.819444	0.847222	0.847222	0.972222	0.871693	0.861314	0.824324	0.835616	0.965517	0.971435	0.974376	0.95515	0.961259	0.994952
RandomForest d4_l4	0.875	0.875796289	0.8970588	0.8	0.8472222	0.9589041	0.875	0.847222	0.833333	0.847222	0.972222	0.875124	0.871429	0.816327	0.847222	0.965517	0.972479	0.976273	0.957658	0.960712	0.995274
RandomForest d4_l5	0.871528	0.872750442	0.9090909	0.8026316	0.8333333	0.9459459	0.8715278	0.833333	0.847222	0.833333	0.972222	0.871532	0.869565	0.824324	0.833333	0.958904	0.971724	0.975148	0.956244	0.960102	0.995403
RandomForest d5_l1	0.875	0.876560759	0.9090909	0.8026316	0.8356164	0.9589041	0.875	0.833333	0.847222	0.847222	0.972222	0.875197	0.869565	0.824324	0.841379	0.965517	0.973476	0.976755	0.961966	0.961516	0.993666
RandomForest d5_l2	0.871528	0.872750442	0.9090909	0.8026316	0.8333333	0.9459459	0.8715278	0.833333	0.847222	0.833333	0.972222	0.871532	0.869565	0.824324	0.833333	0.958904	0.972881	0.976595	0.957819	0.962384	0.994727
RandomForest d5_l3	0.871528	0.872018798	0.8970588	0.8	0.8450704	0.9459459	0.8715278	0.847222	0.833333	0.833333	0.972222	0.871455	0.871429	0.816327	0.839161	0.958904	0.973396	0.976659	0.961677	0.959684	0.995563
RandomForest d5_l4	0.871528	0.872750442	0.9090909	0.8026316	0.8333333	0.9459459	0.8715278	0.833333	0.847222	0.833333	0.972222	0.871532	0.869565	0.824324	0.833333	0.958904	0.976386	0.97727	0.966274	0.965503	0.996496
RandomForest d5_l5	0.864583	0.864732379	0.8823529	0.7972973	0.8333333	0.9459459	0.8645833	0.833333	0.819444	0.833333	0.972222	0.8644	0.857143	0.808219	0.833333	0.958904	0.972865	0.975662	0.960423	0.960198	0.995177

The above are the statistics for the Train(1st) and Test(2nd) data of the dataset v2.

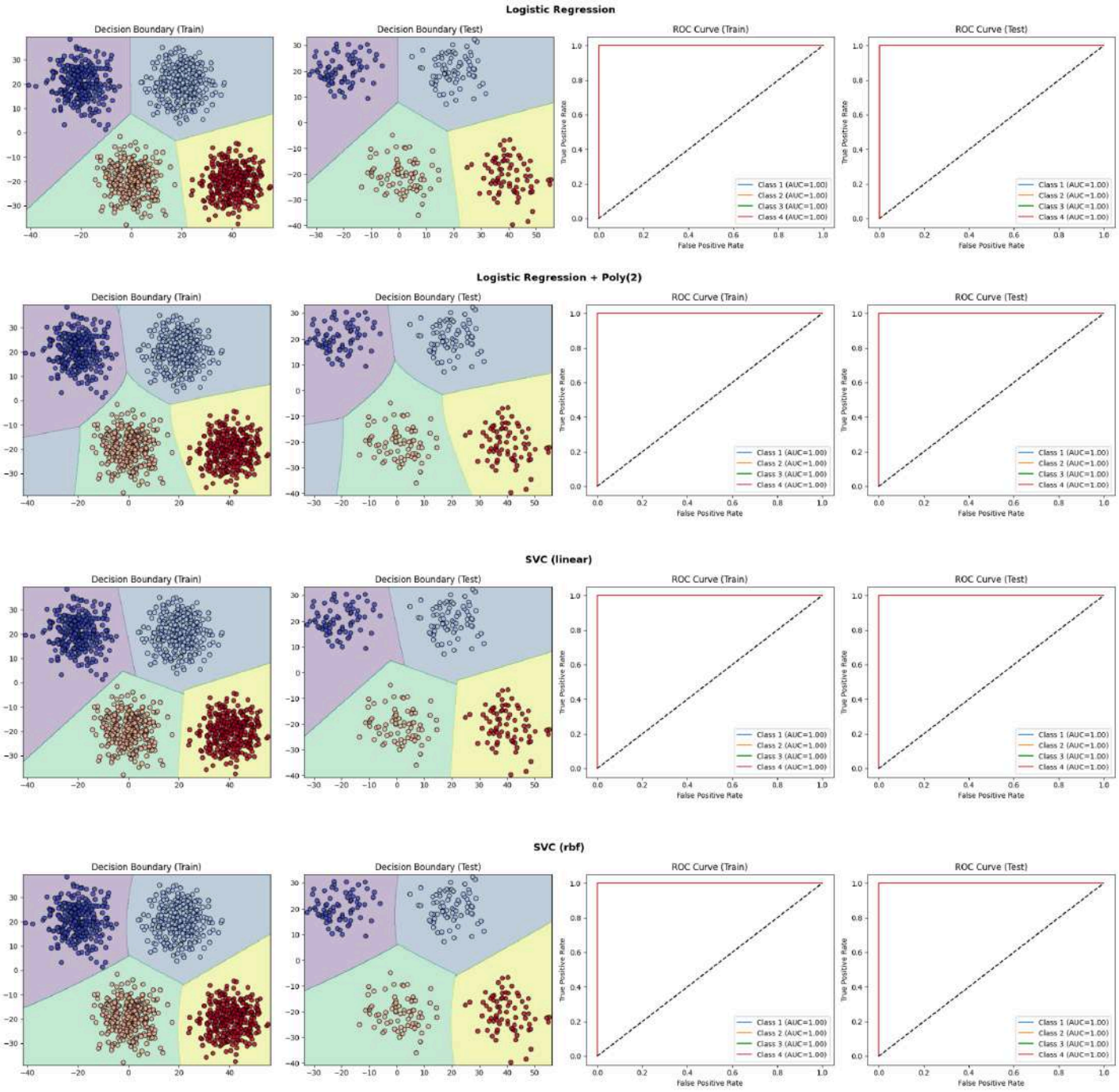
I am not explaining or analysing the above metrics now as a later question asks for the same, that is question number 6.

These are the data entries that were collected and they are supplied on moodle too.

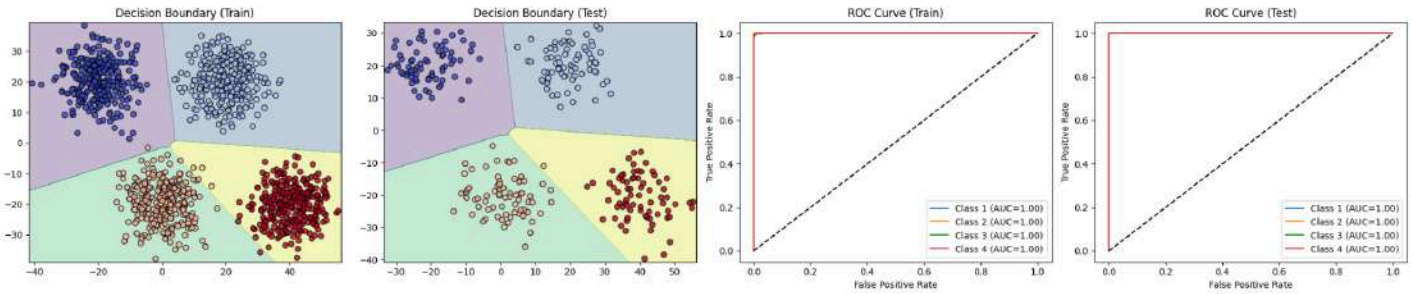
QUESTION 5:

Now, below are the plots of the Decision Boundary plots and also the ROC curves for the train and test data of the datasets.

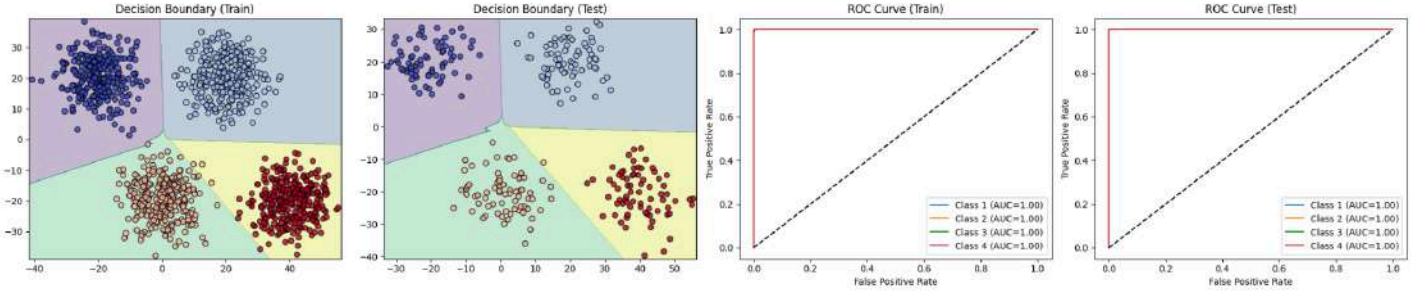
For the Dataset v0,



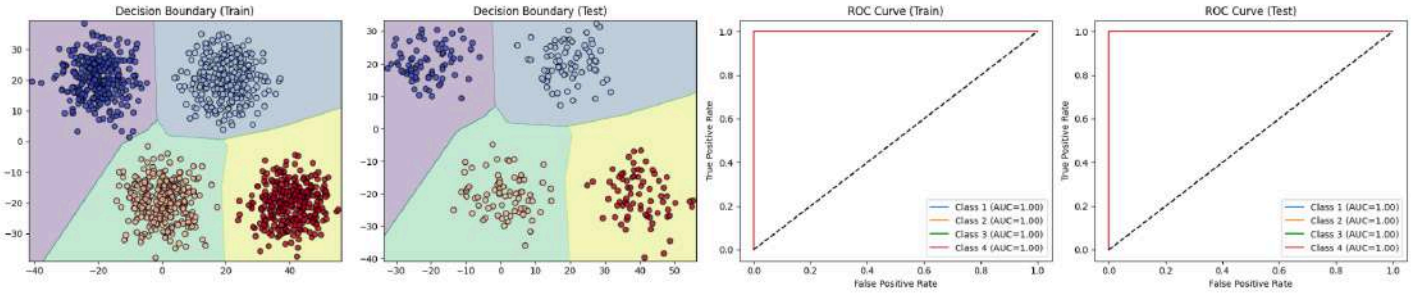
NeuralNet (5)



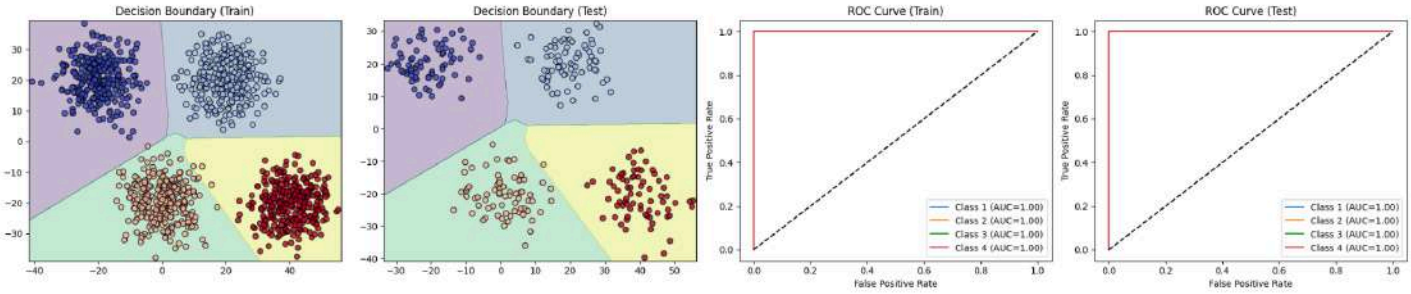
NeuralNet (5,5)



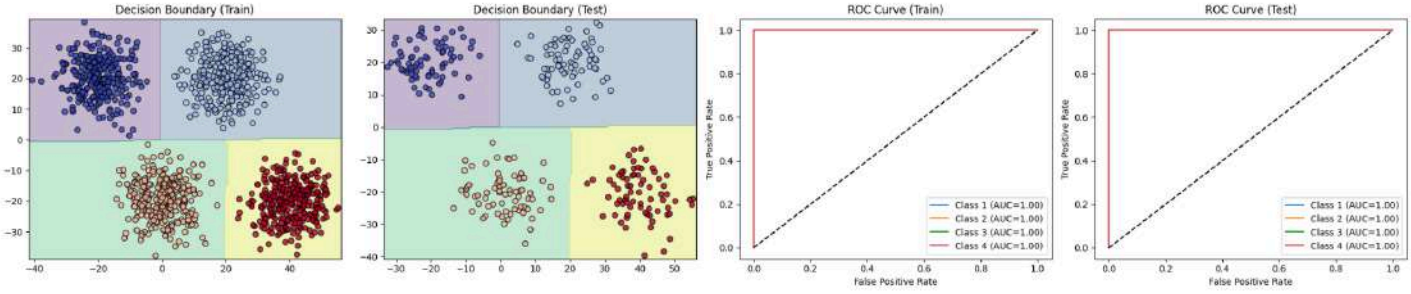
NeuralNet (5,5,5)



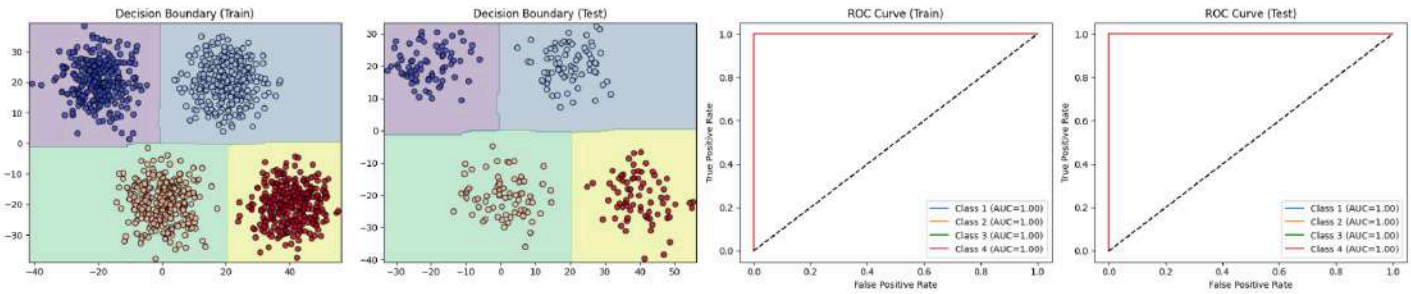
NeuralNet (10)



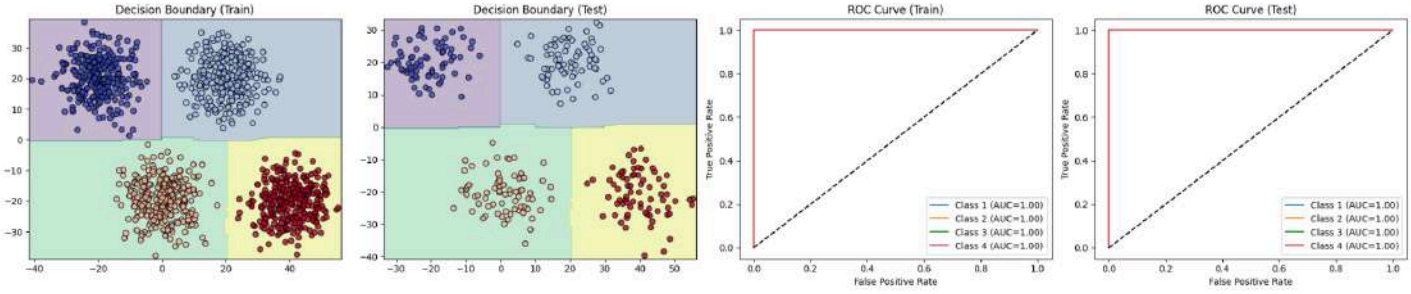
RandomForest d2_l1



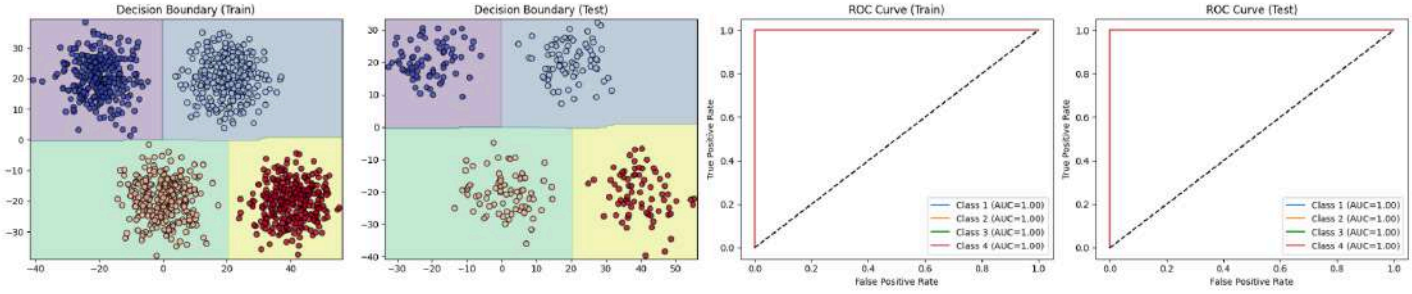
RandomForest d2_l2



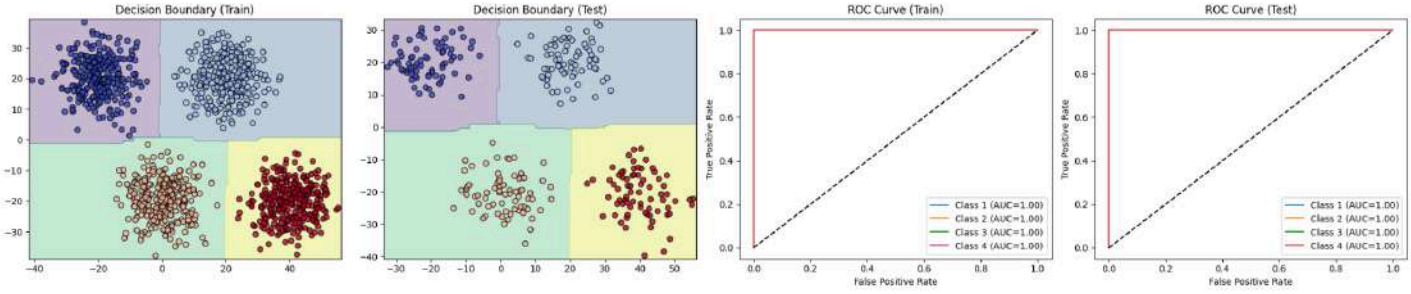
RandomForest d2_l3



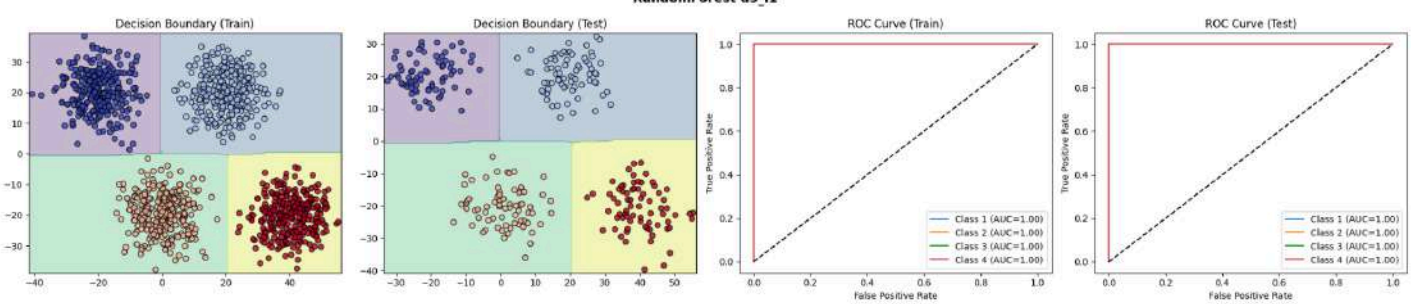
RandomForest d2_l4



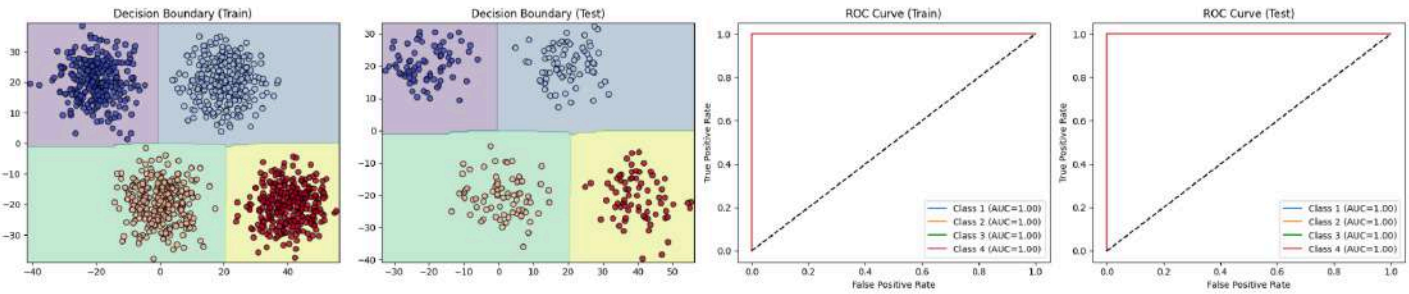
RandomForest d2_l5



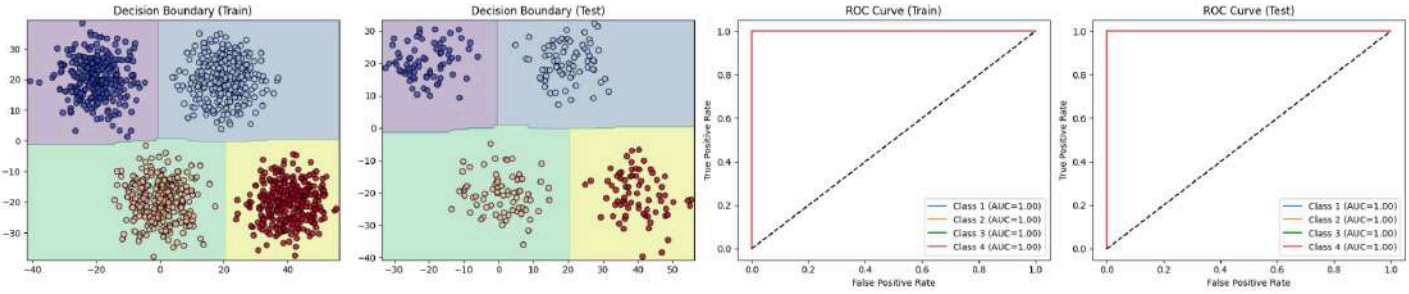
RandomForest d3_l1



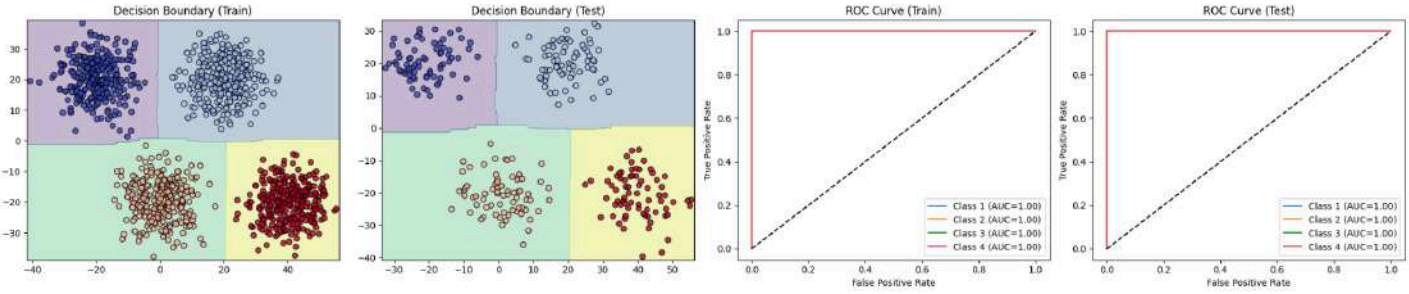
RandomForest d3_l2



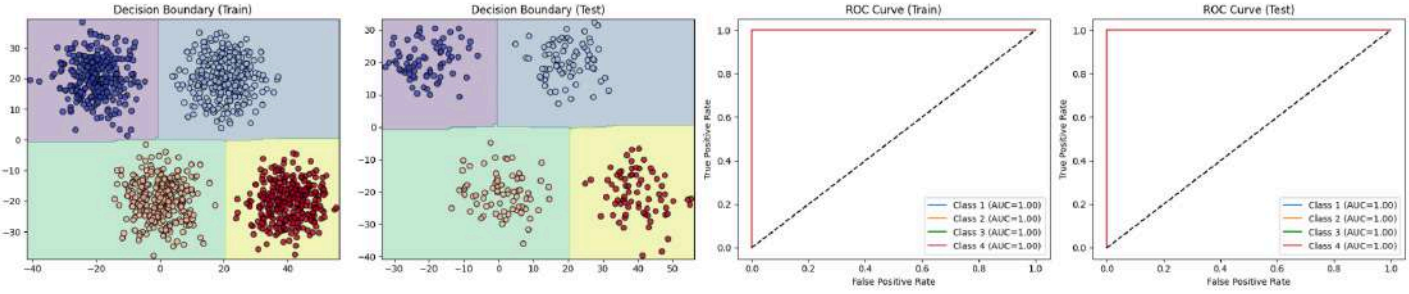
RandomForest d3_l3



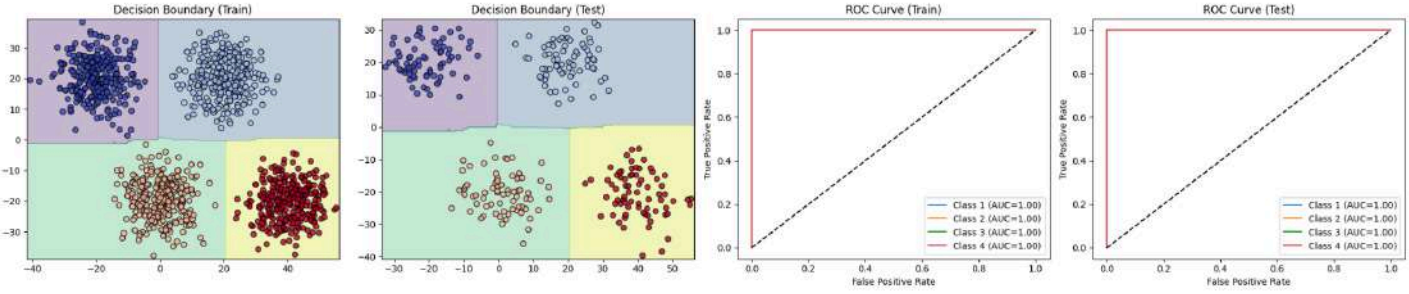
RandomForest d3_l4



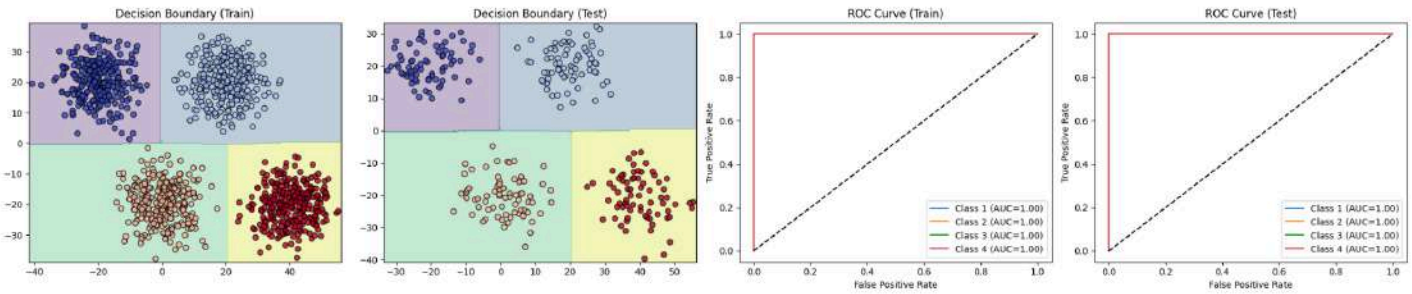
RandomForest d3_l5



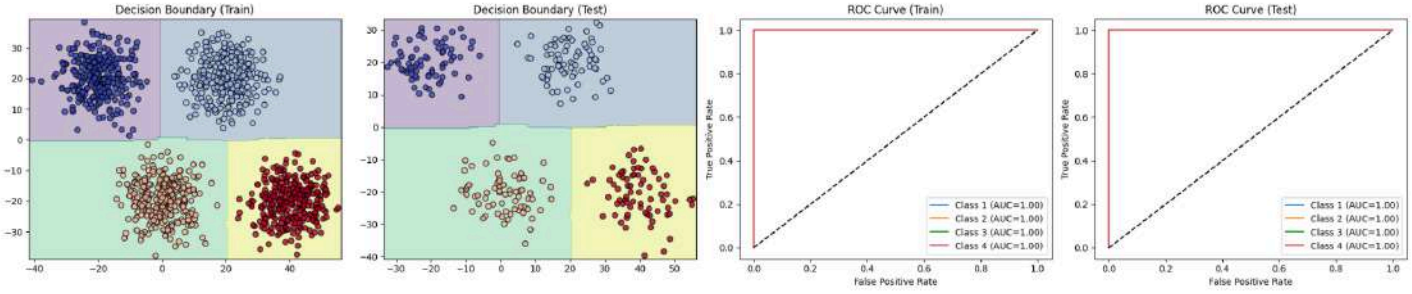
RandomForest d4_l1



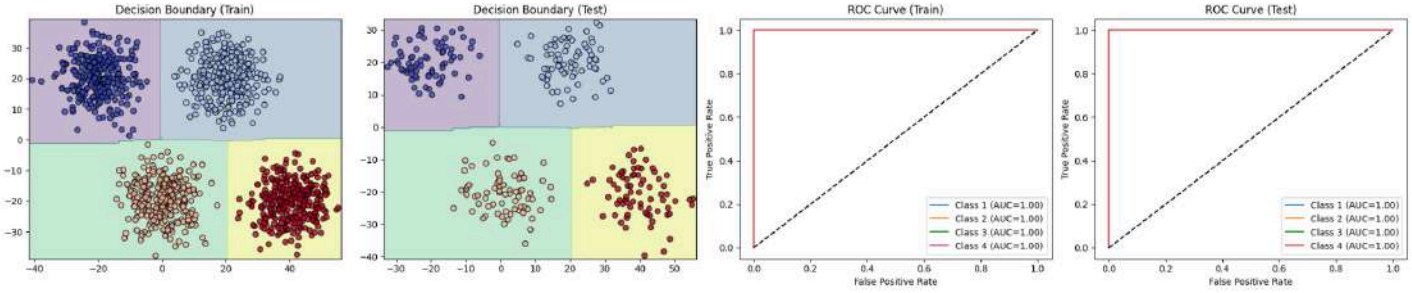
RandomForest d4_l2



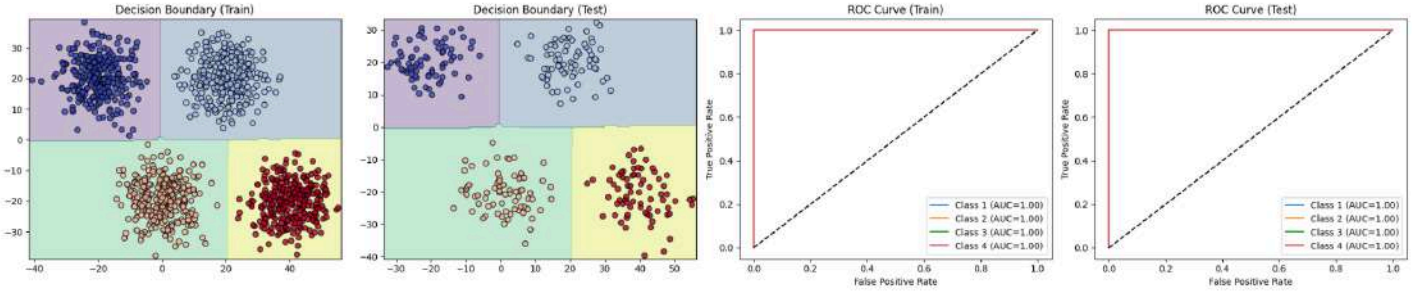
RandomForest d4_l3



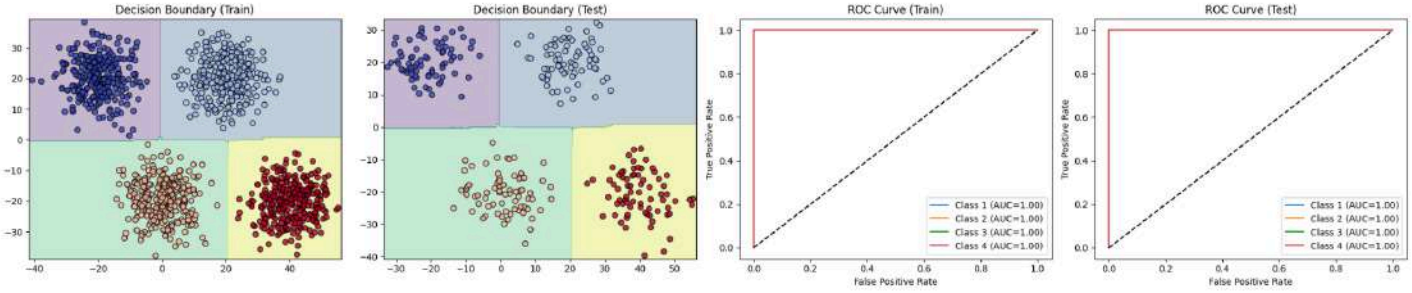
RandomForest d4_l4



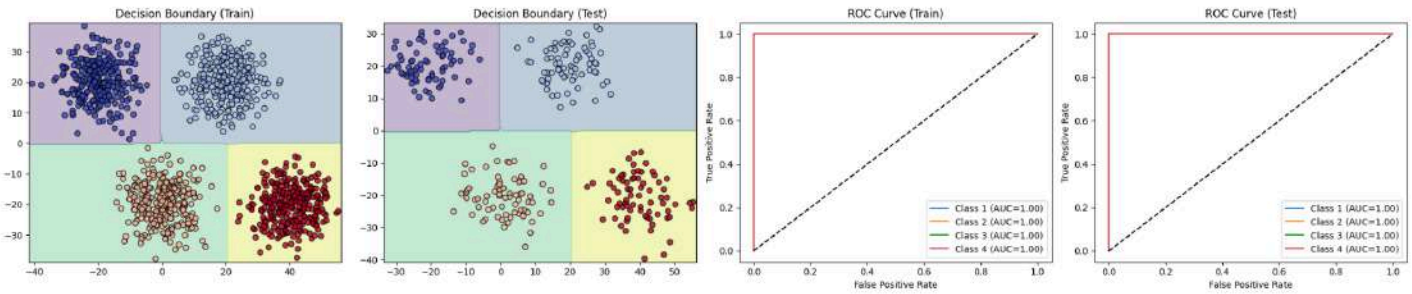
RandomForest d4_l5



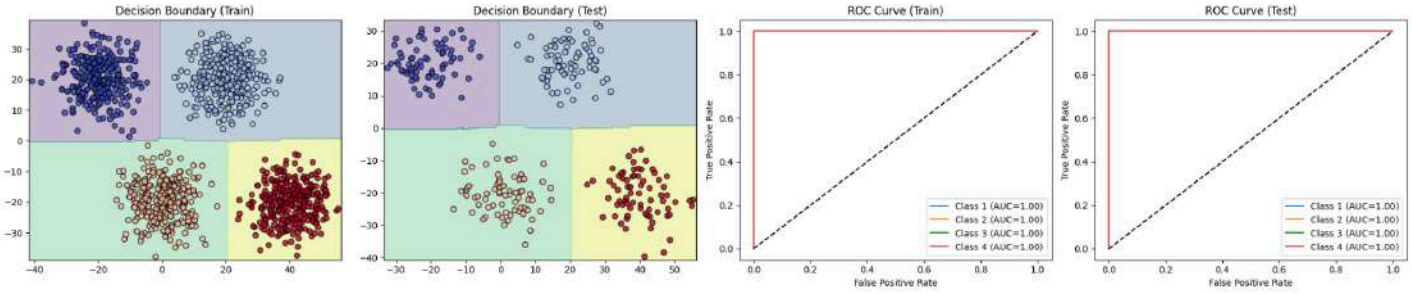
RandomForest d5_l1



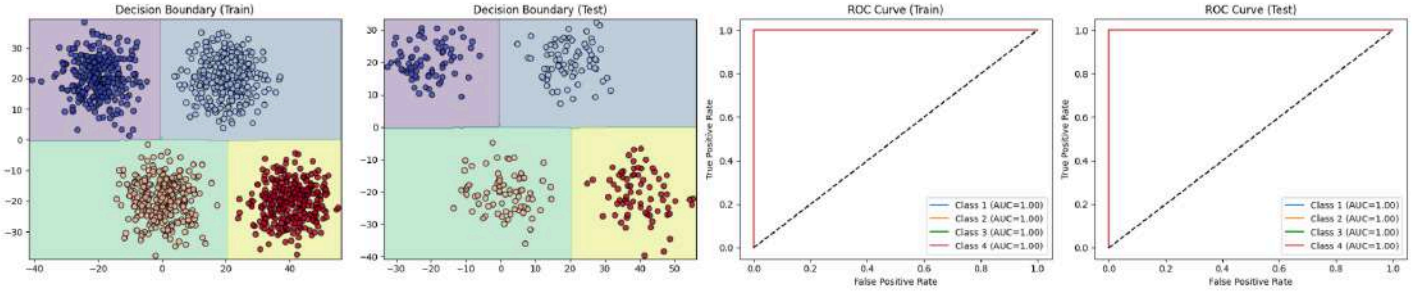
RandomForest d5_l2



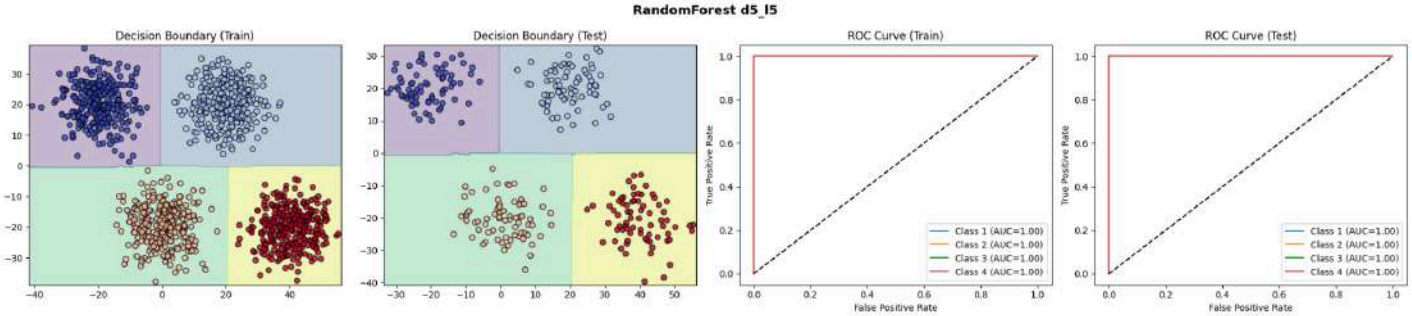
RandomForest d5_l3



RandomForest d5_l4

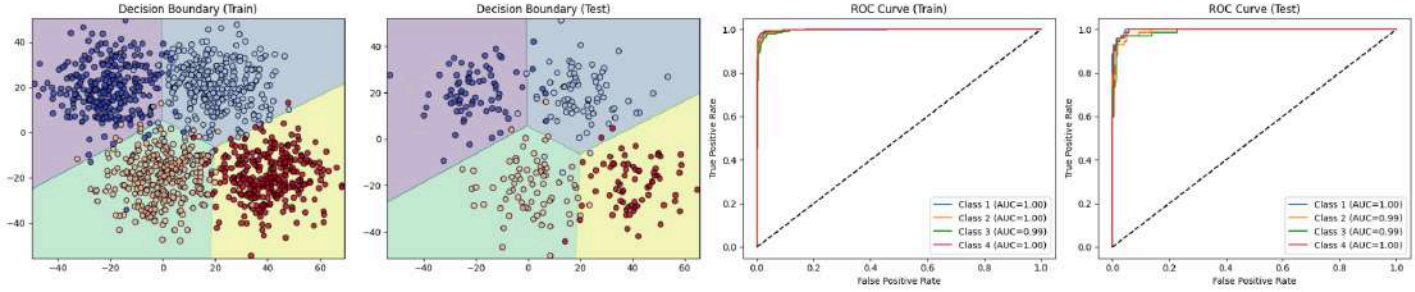


RandomForest d5_l5

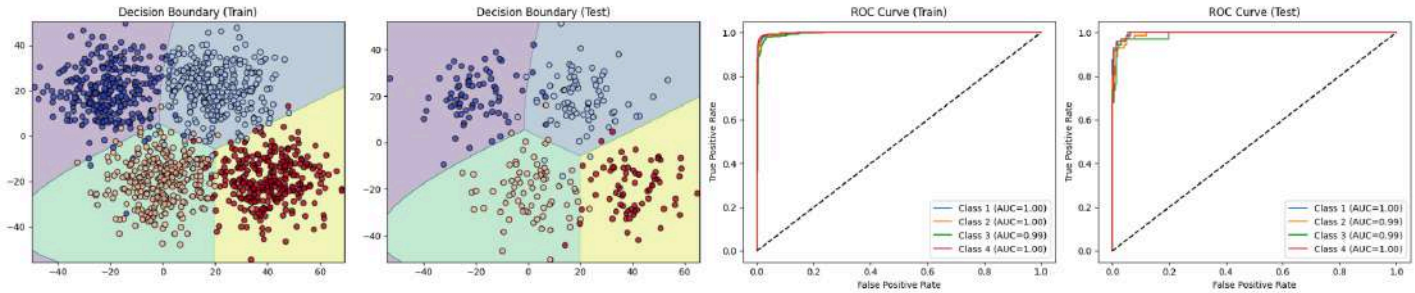


Now, for the Dataset V1,

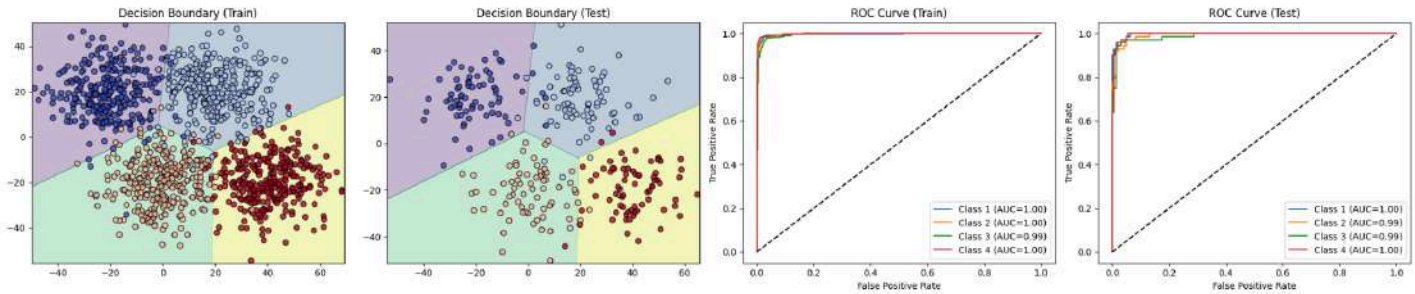
Logistic Regression



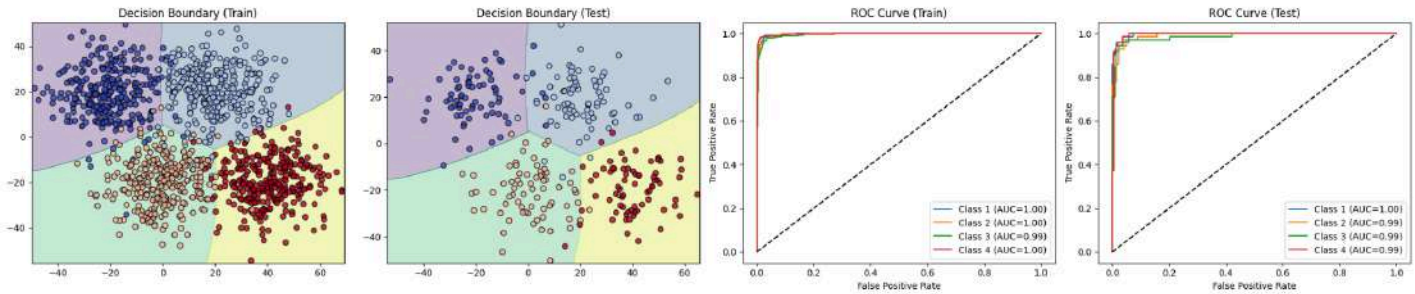
Logistic Regression + Poly(2)



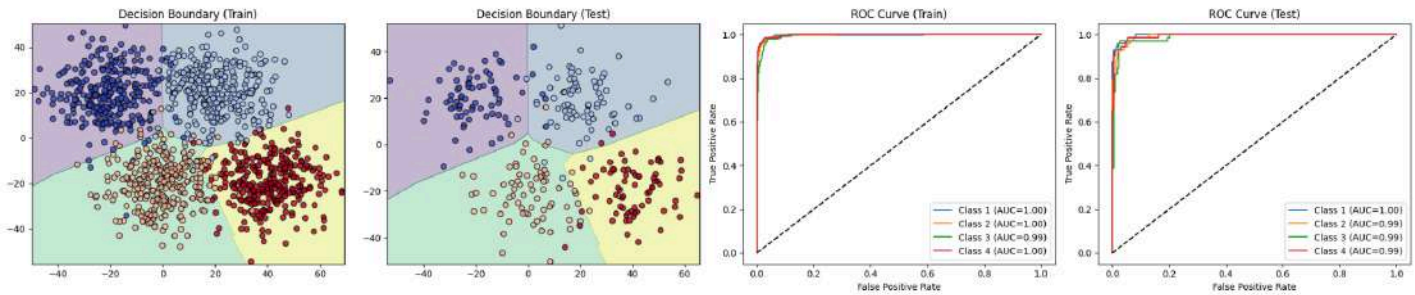
SVC (linear)



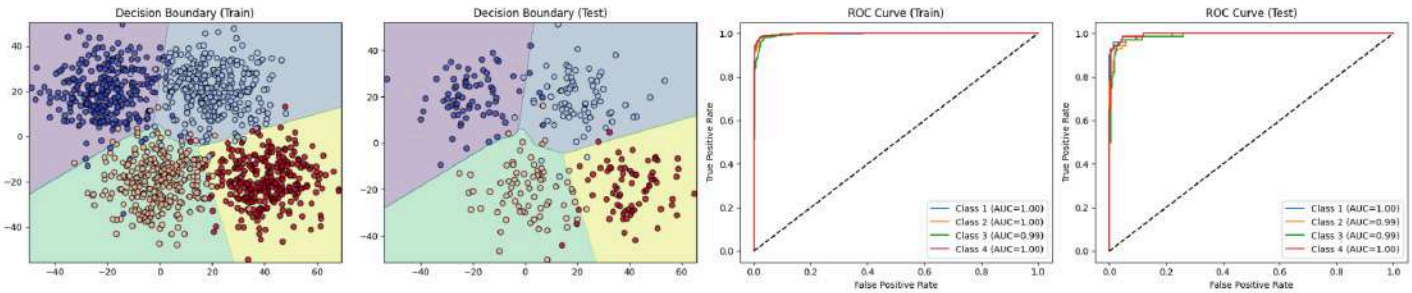
SVC (rbf)



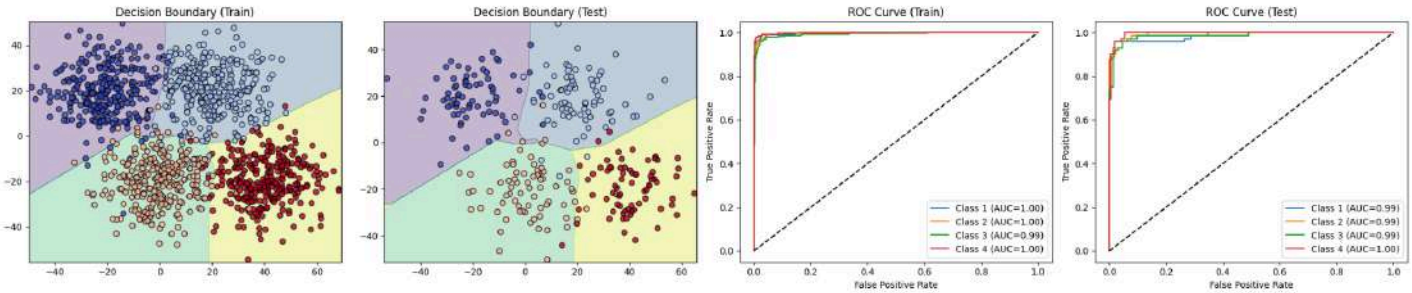
NeuralNet (5)



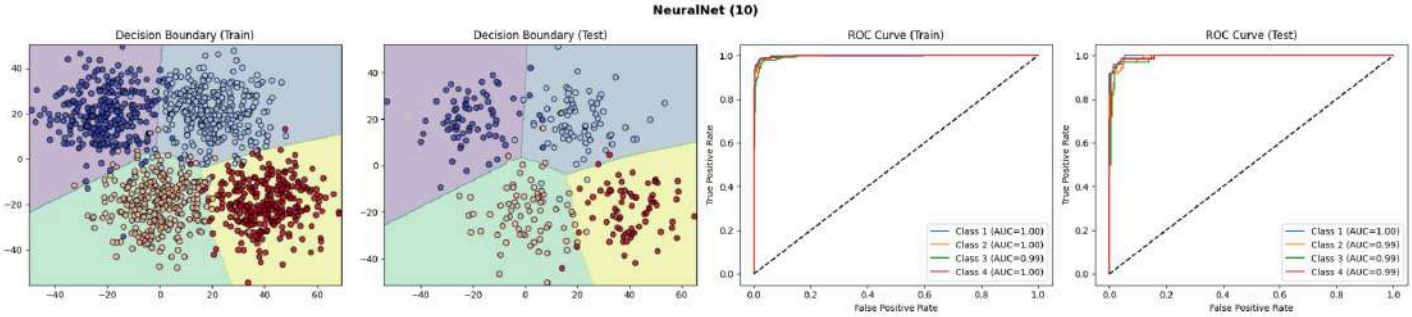
NeuralNet (5,5)



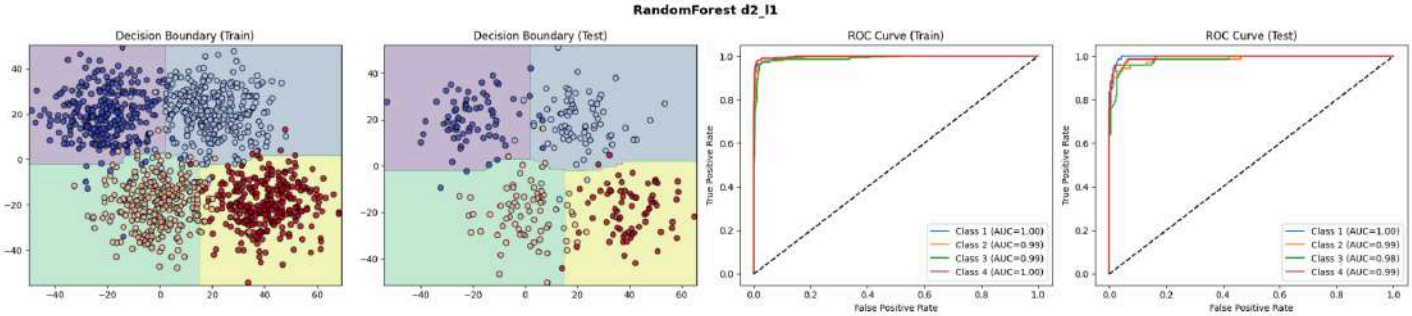
NeuralNet (5,5,5)



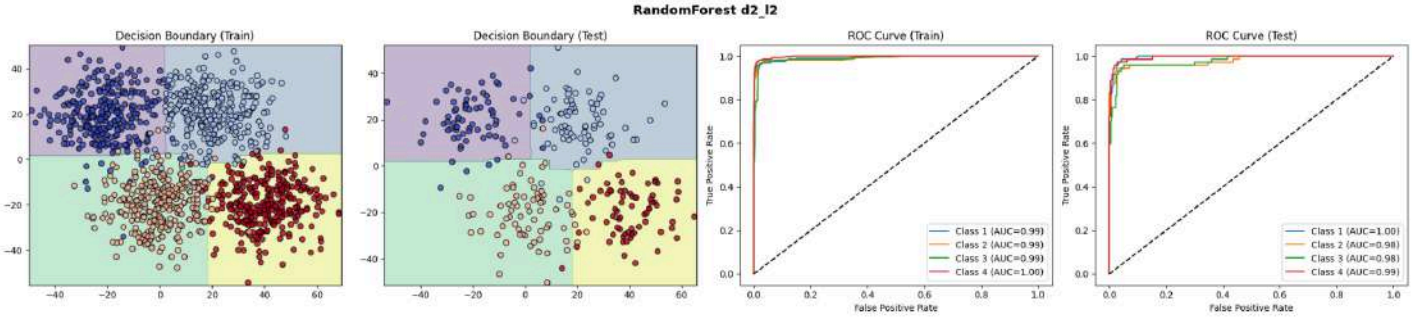
NeuralNet (10)



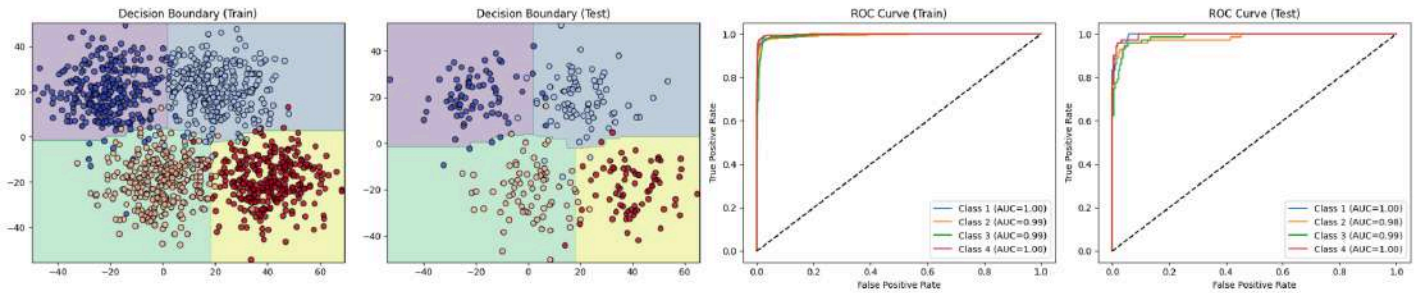
RandomForest d2_l1



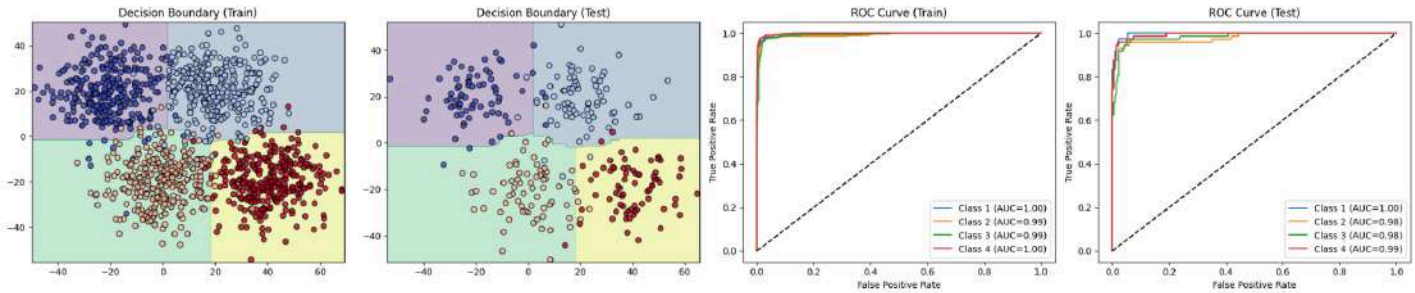
RandomForest d2_l2



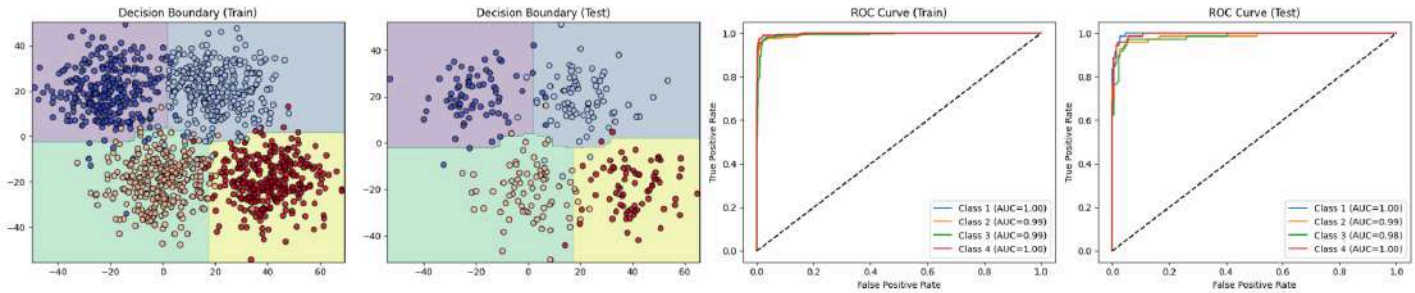
RandomForest d2_l3



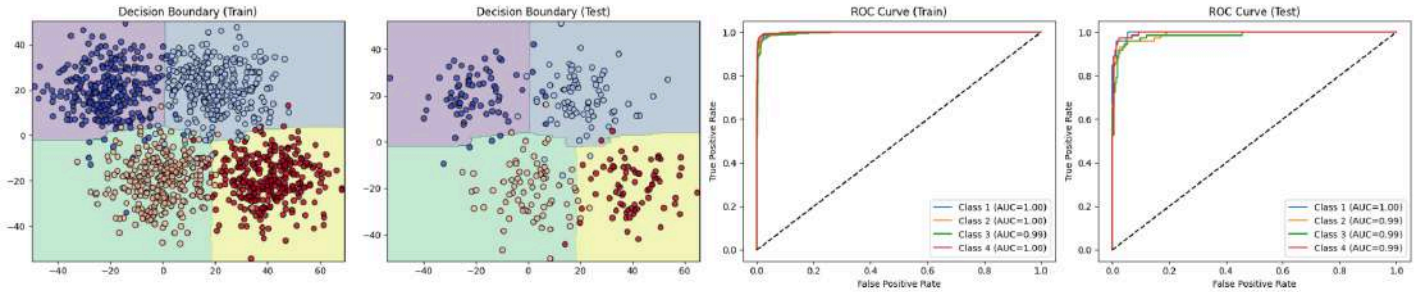
RandomForest d2_l4



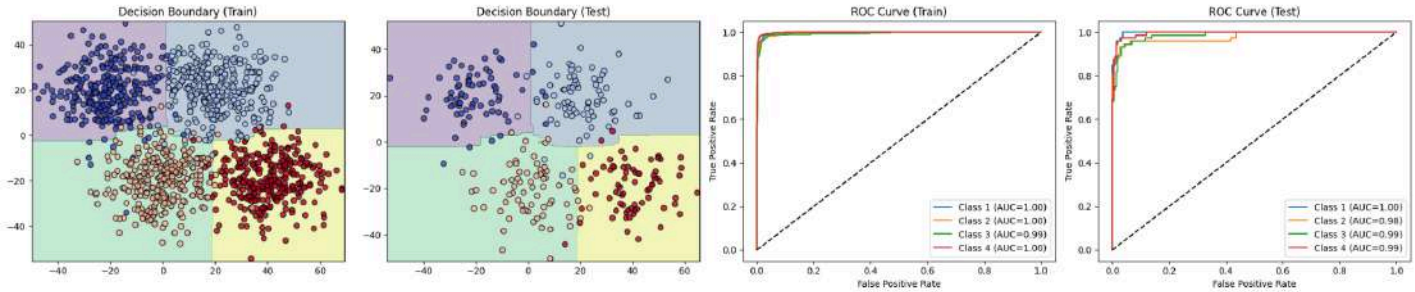
RandomForest d2_l5



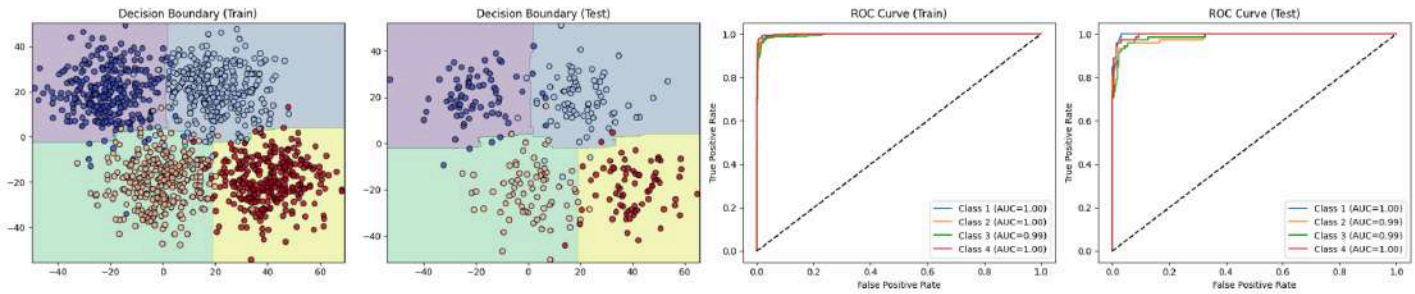
RandomForest d3_l1



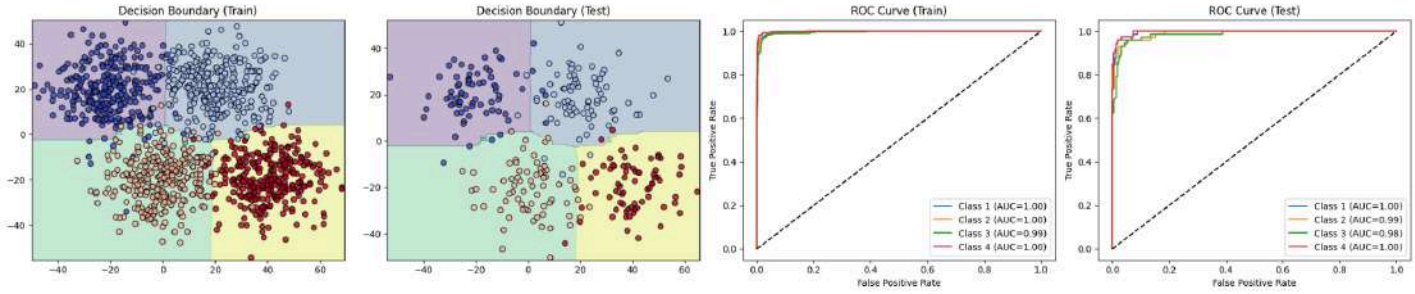
RandomForest d3_l2



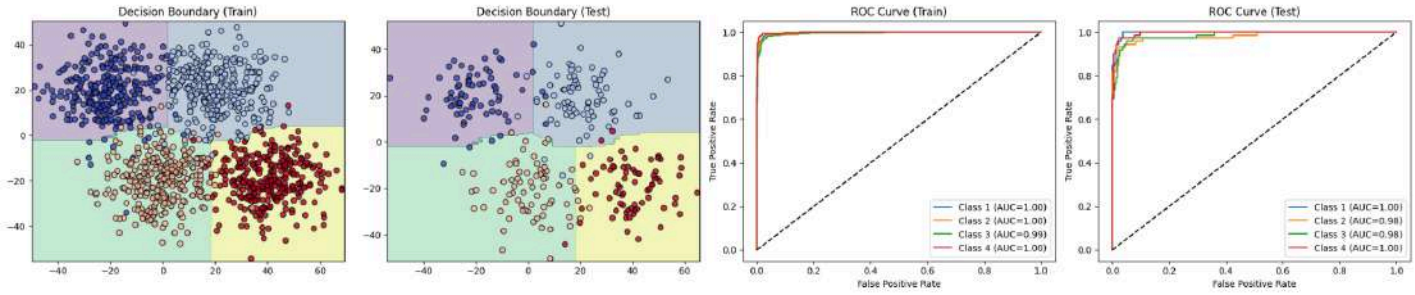
RandomForest d3_l3



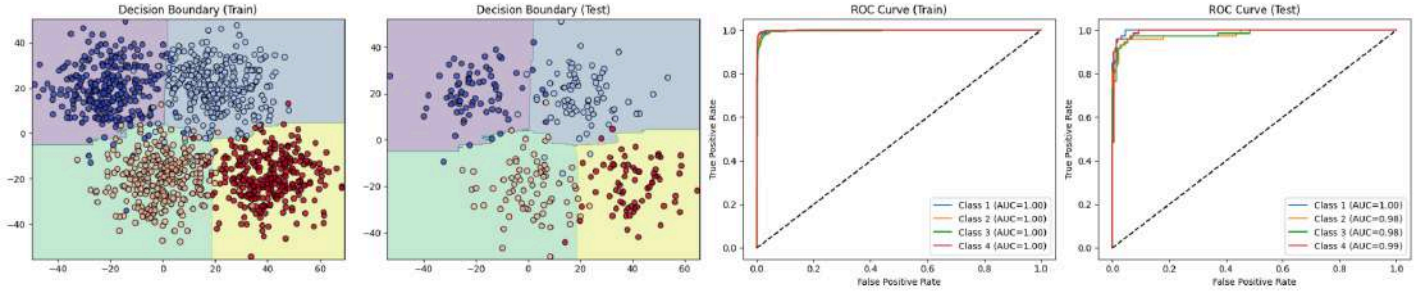
RandomForest d3_l4



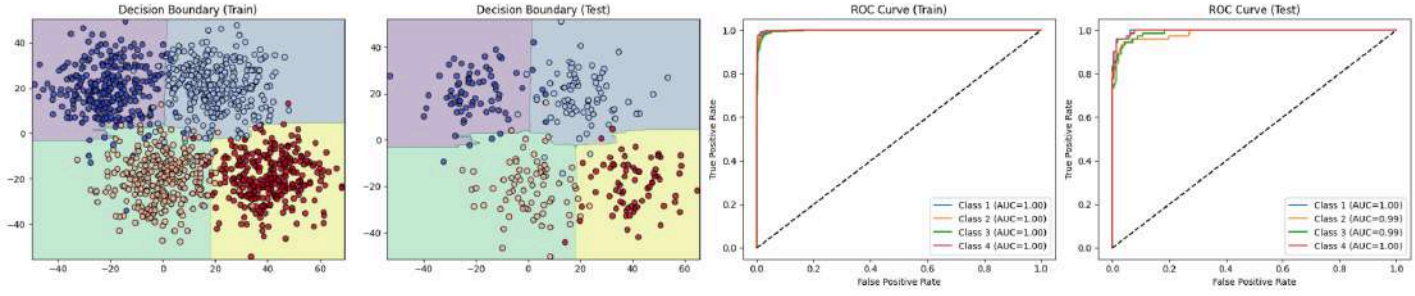
RandomForest d3_l5



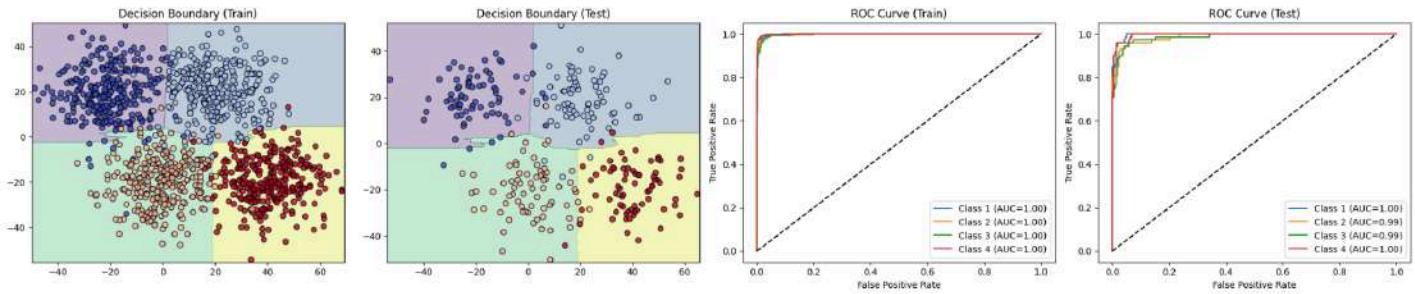
RandomForest d4_l1



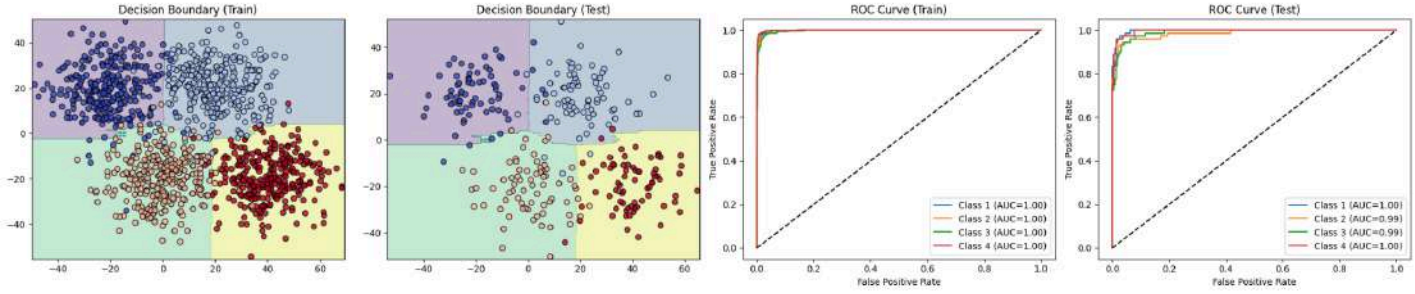
RandomForest d4_l2



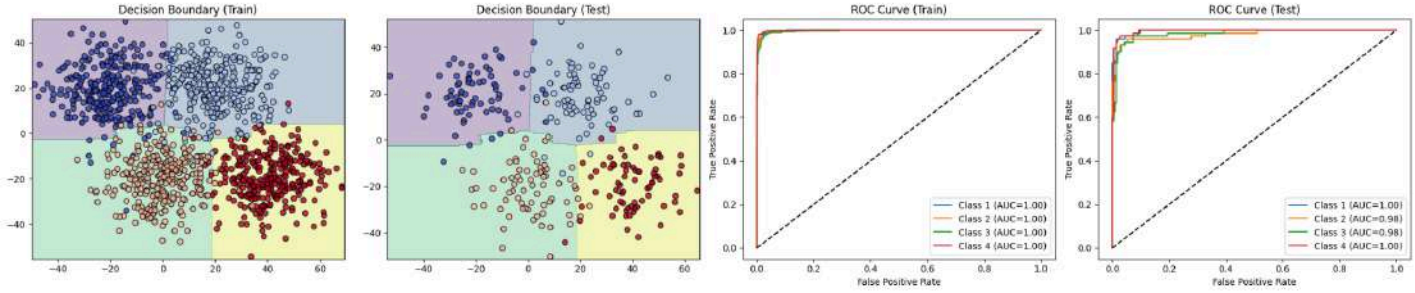
RandomForest d4_l3



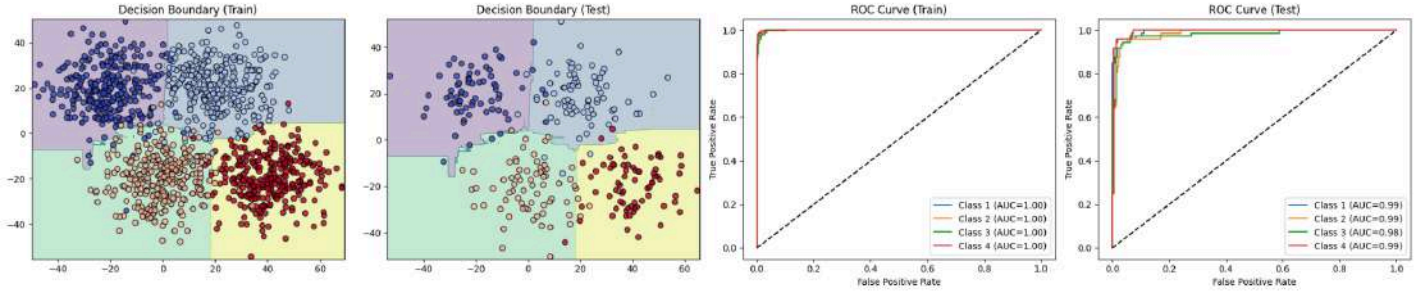
RandomForest d4_l4



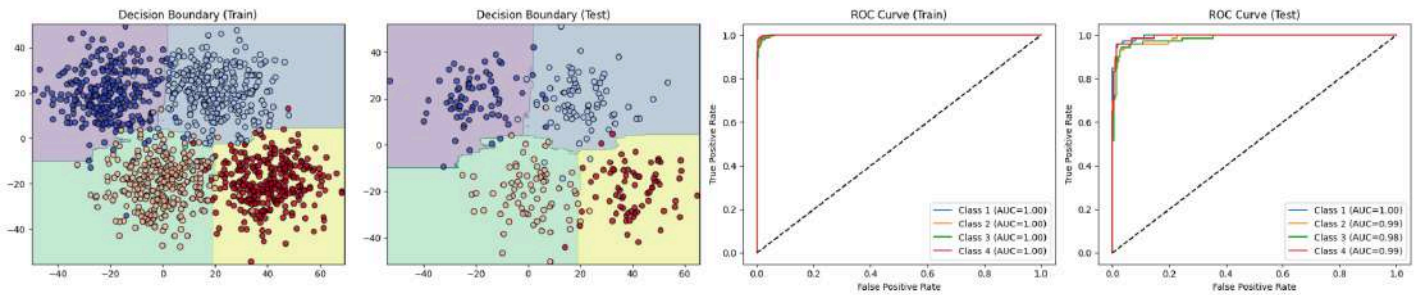
RandomForest d5_l5



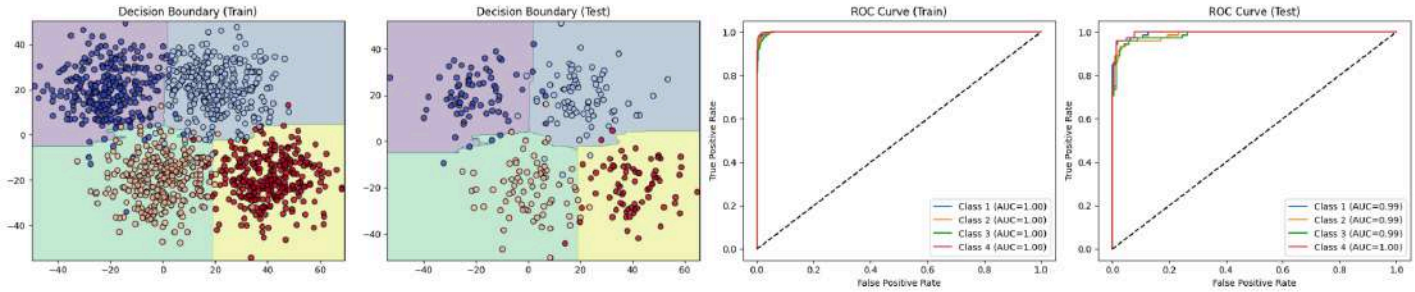
RandomForest d5_l1



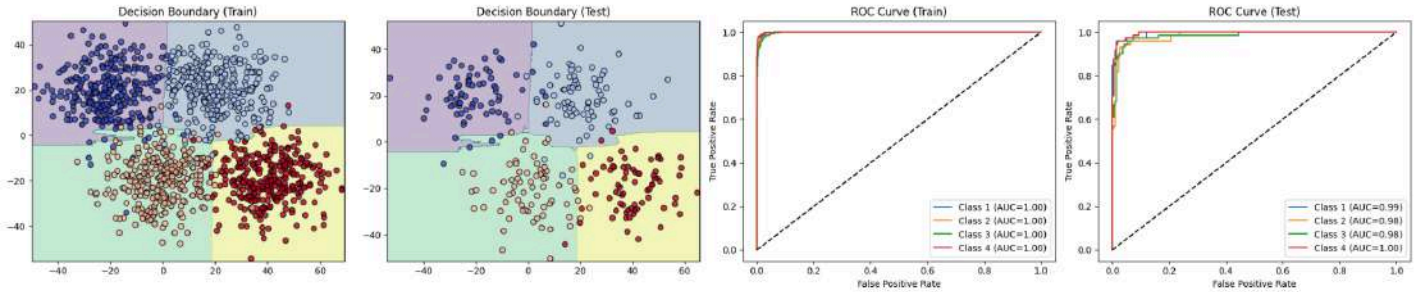
RandomForest d5_l2



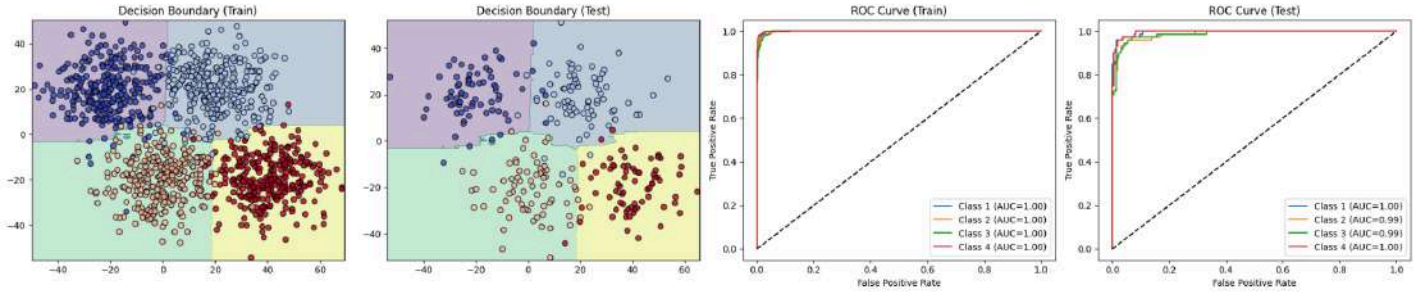
RandomForest d5_l3



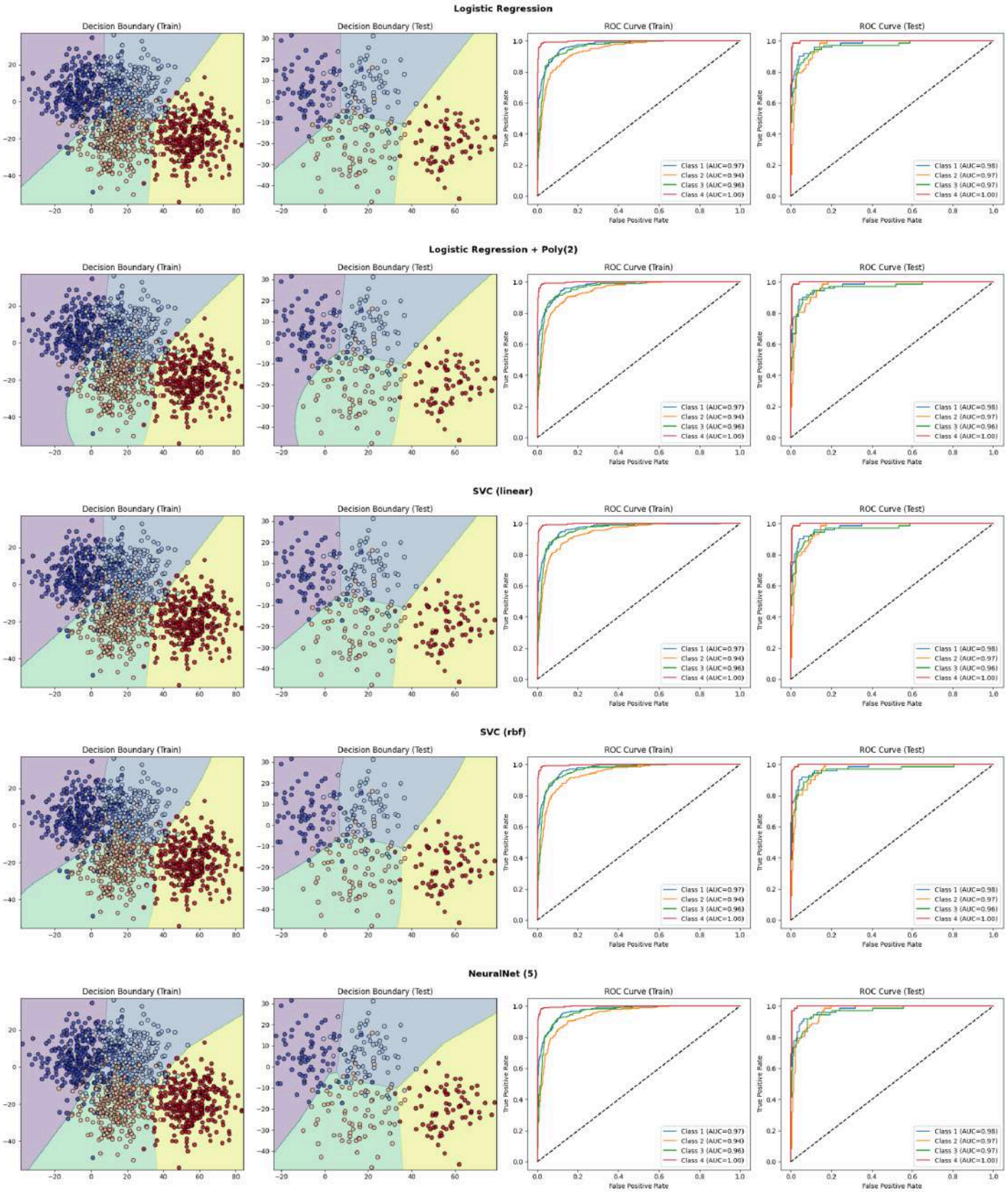
RandomForest d5_l4



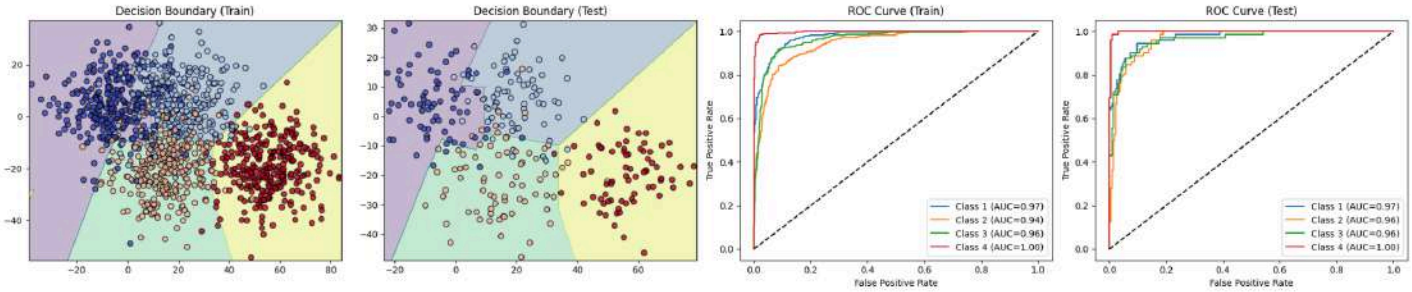
RandomForest d5_l5



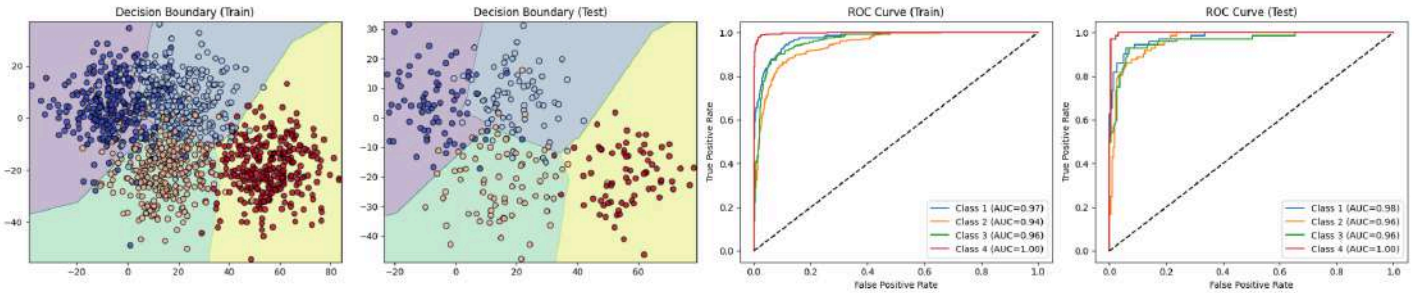
Now, for Dataset v2,



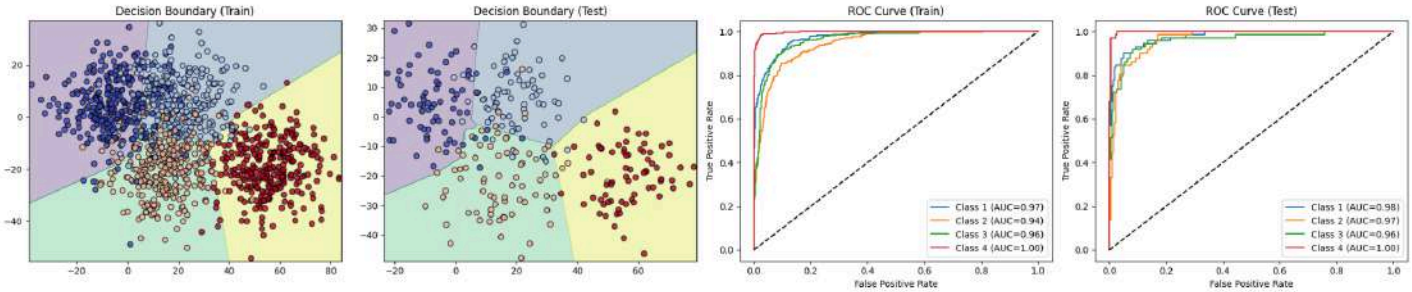
NeuralNet (5,5)



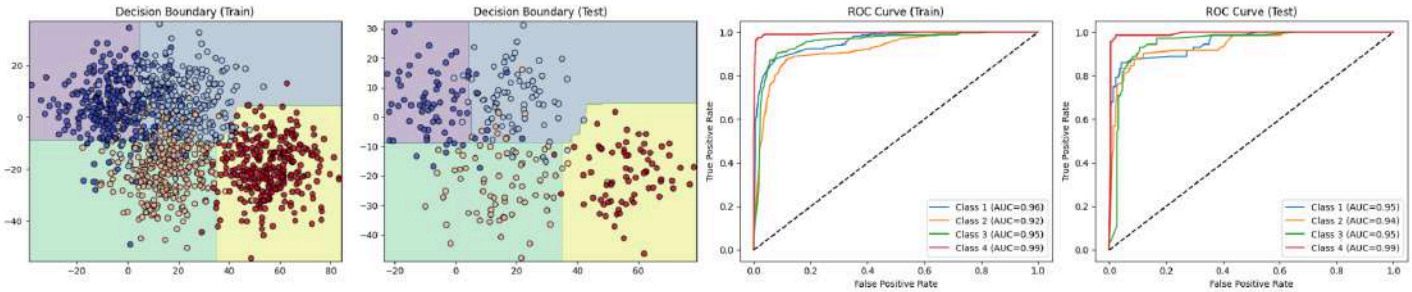
NeuralNet (5,5,5)



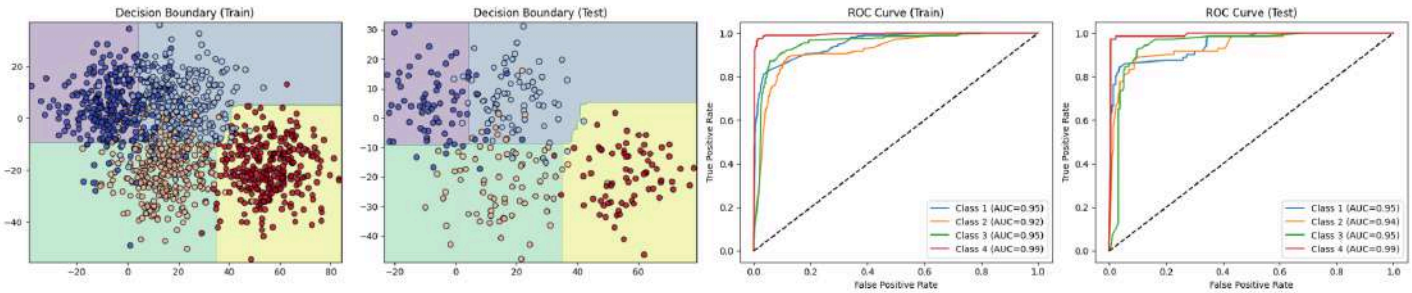
NeuralNet (10)



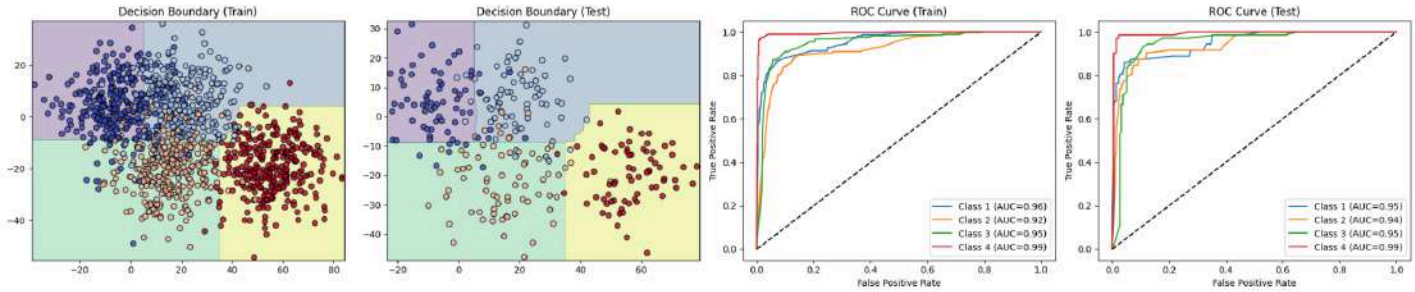
RandomForest d2_l1



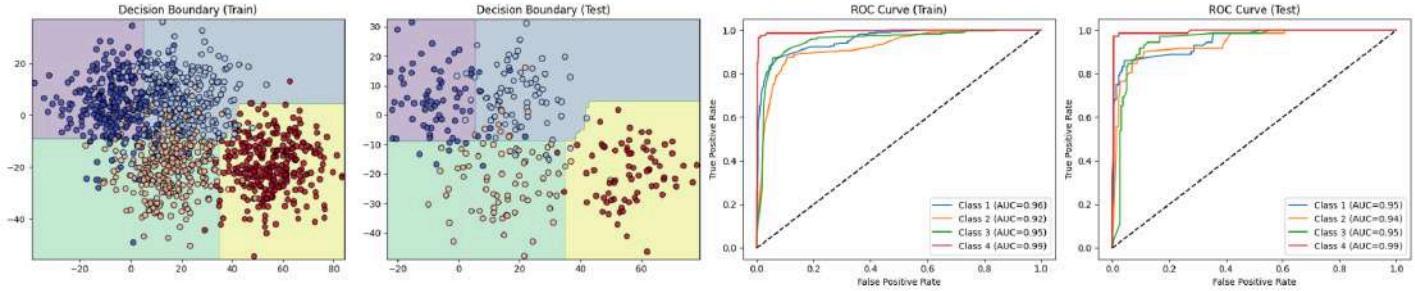
RandomForest d2_l2



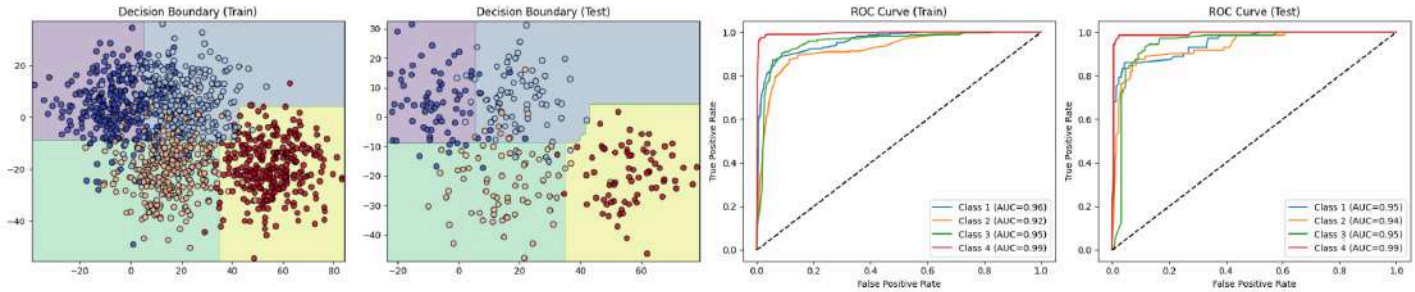
RandomForest d2_l3



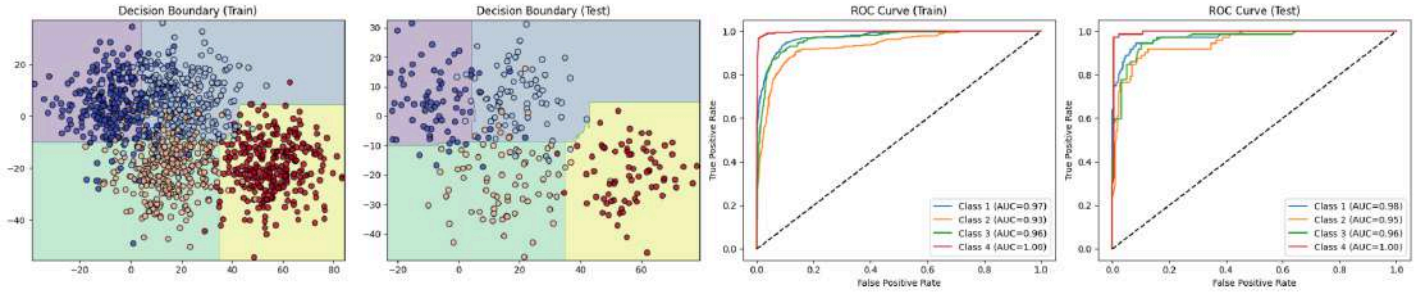
RandomForest d2_l4



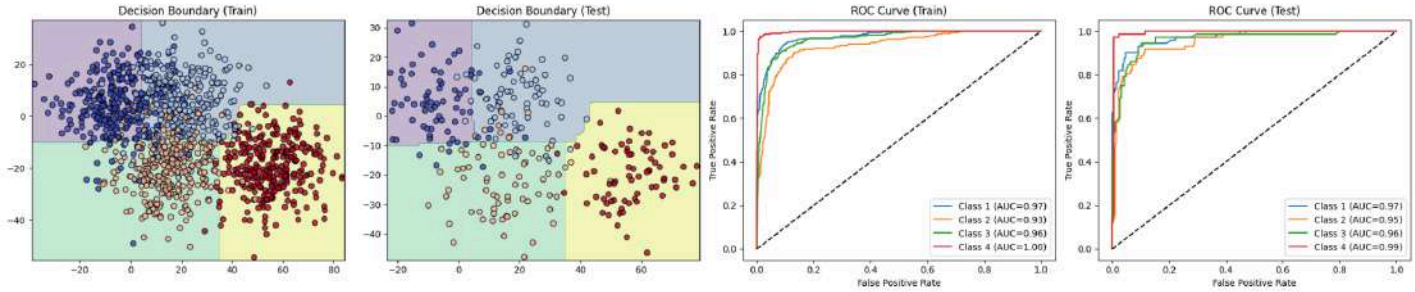
RandomForest d2_l5



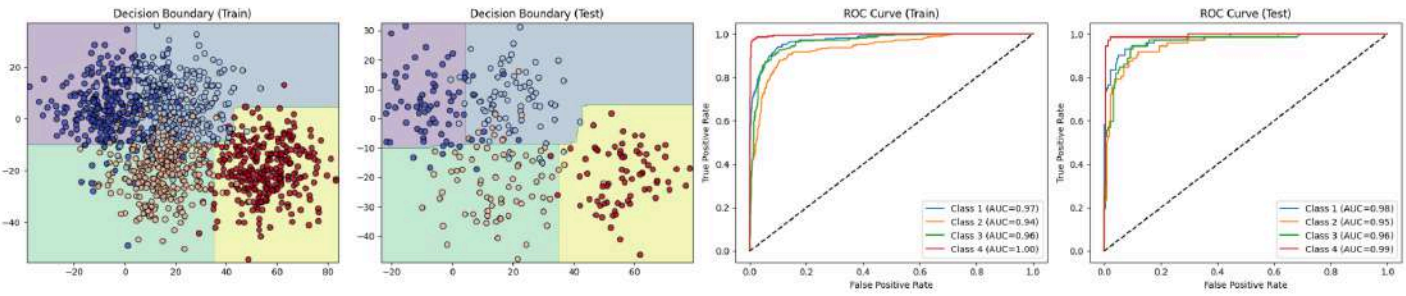
RandomForest d3_l1



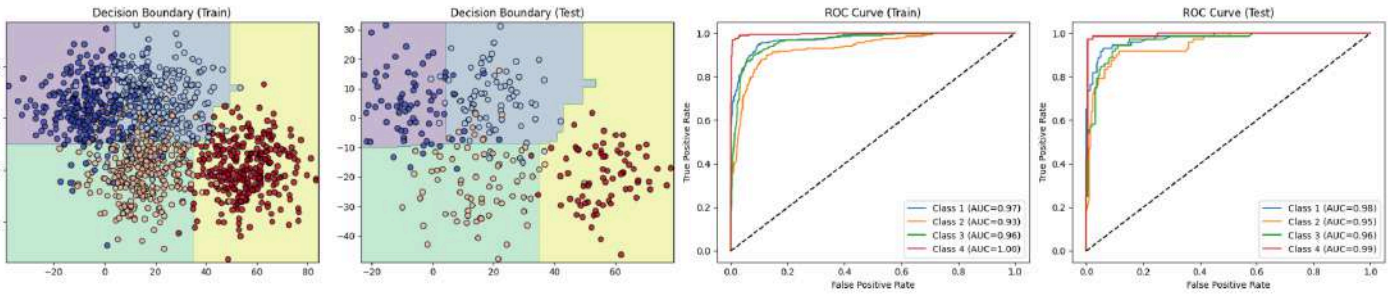
RandomForest d3_l2



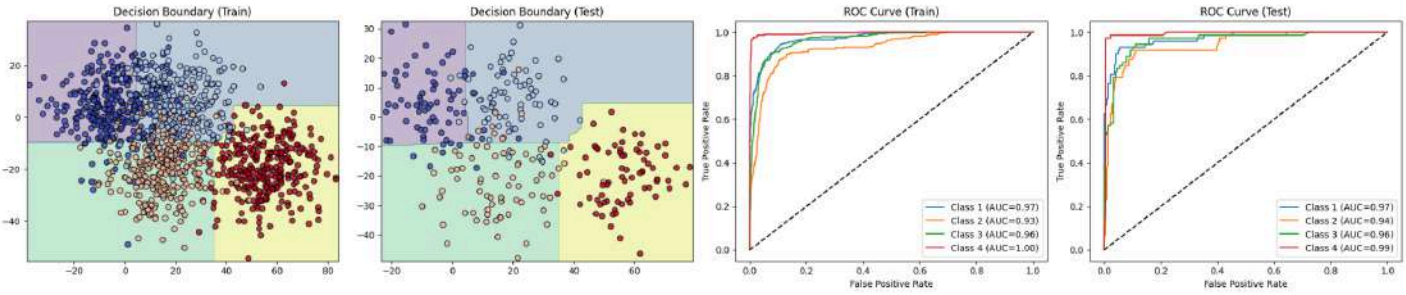
RandomForest d3_l3



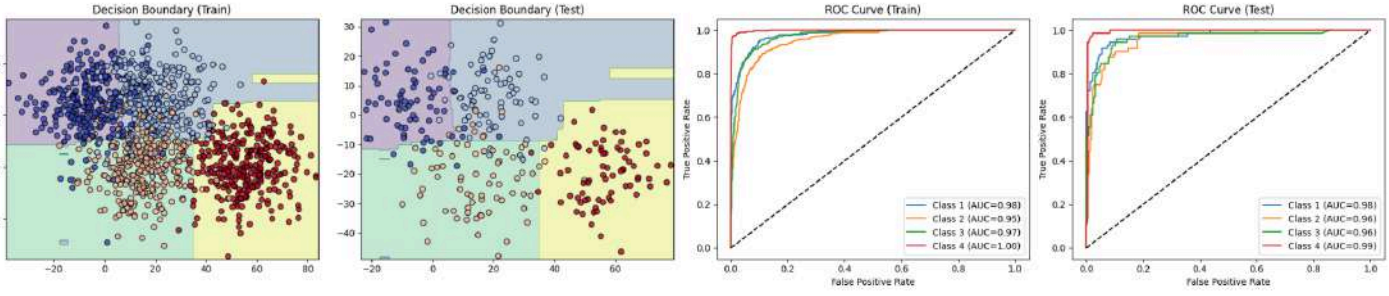
RandomForest d3_l4



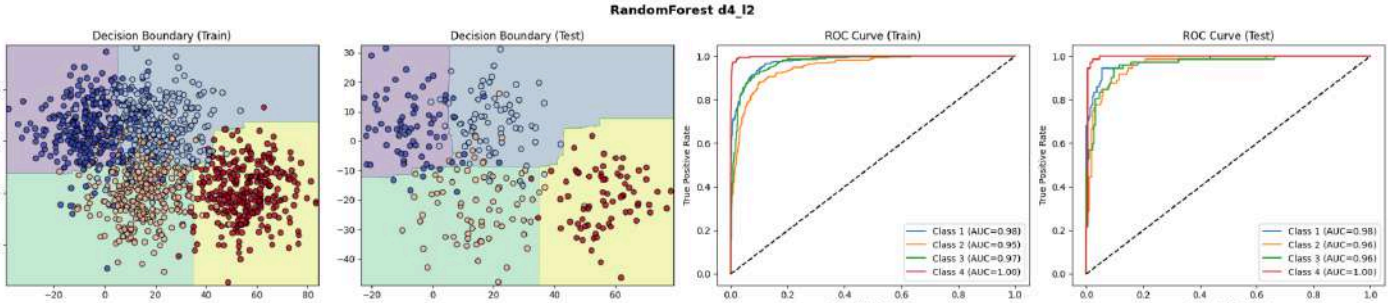
RandomForest d3_l5



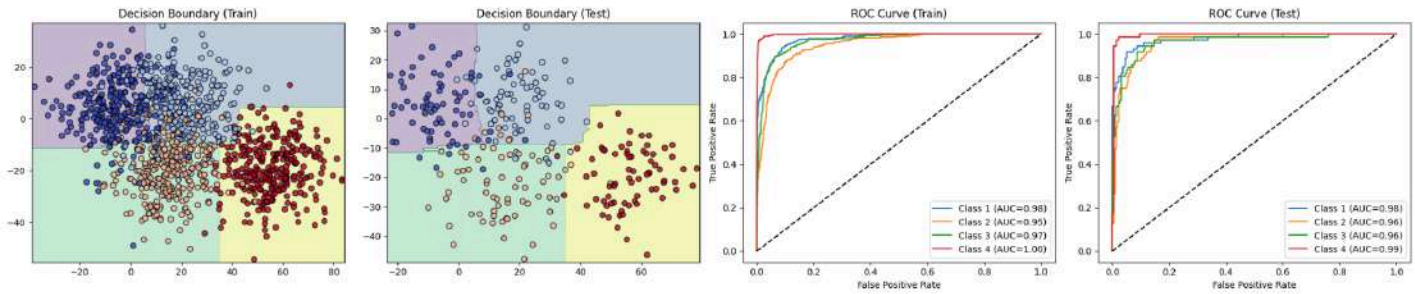
RandomForest d4_l1



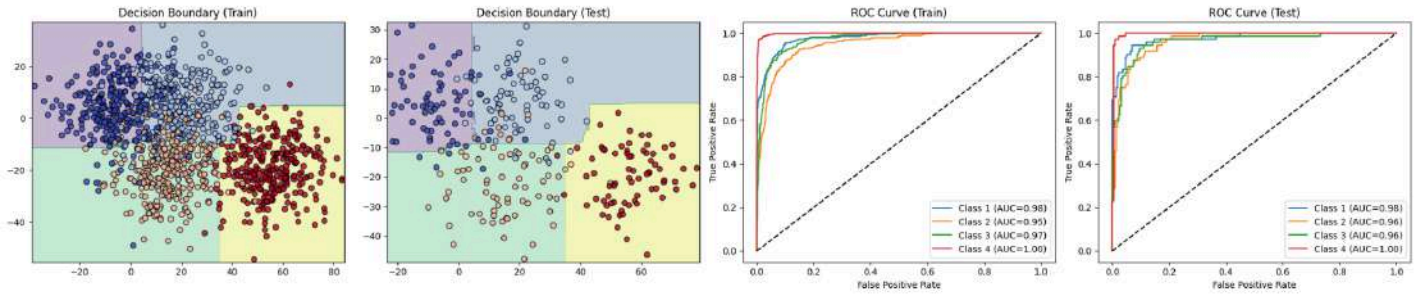
RandomForest d4_l2



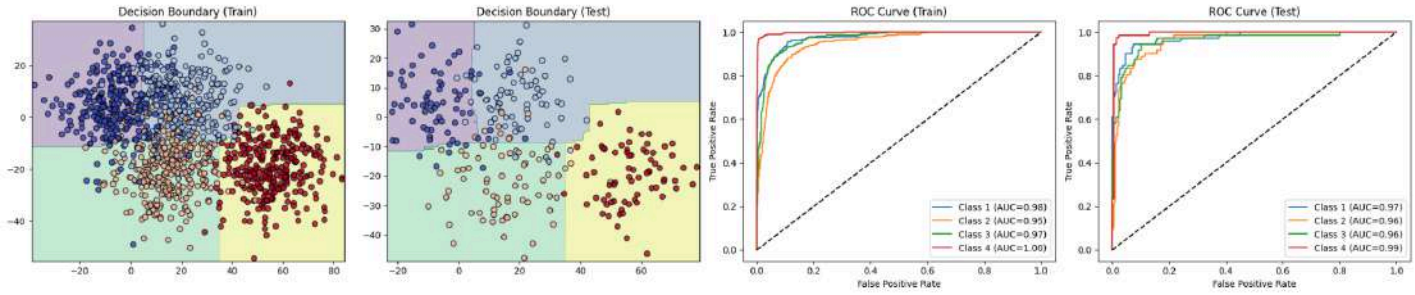
RandomForest d4_l3



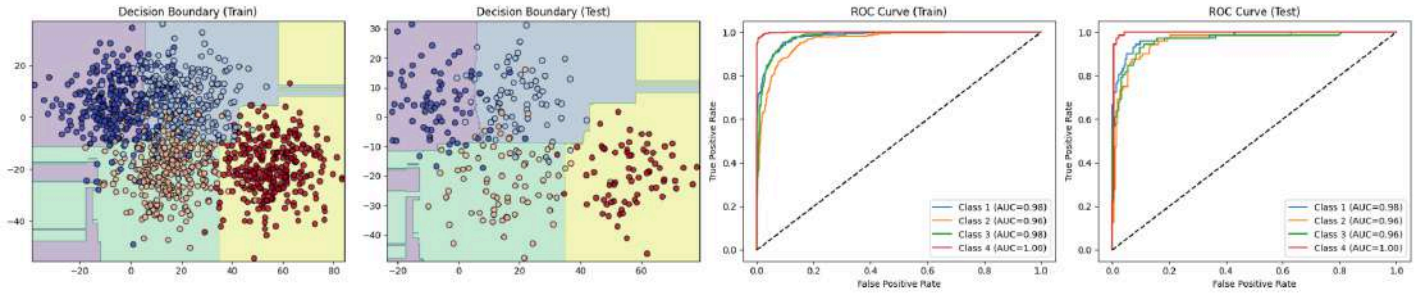
RandomForest d4_l4



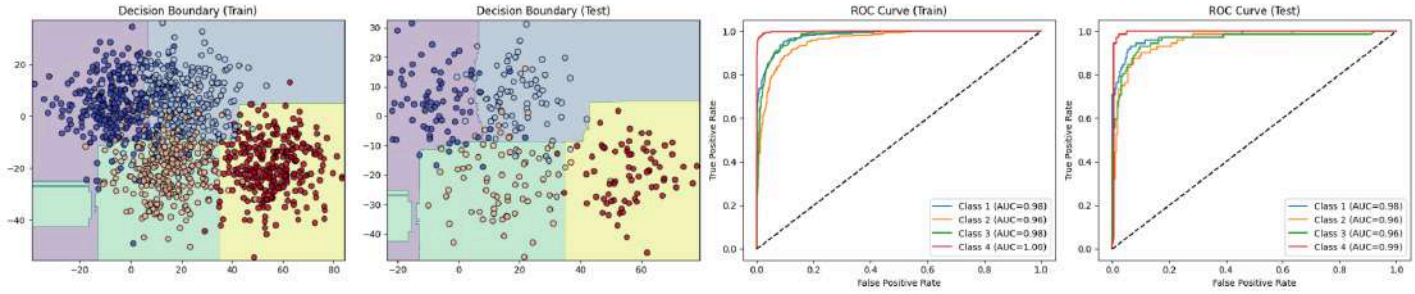
RandomForest d4_l5

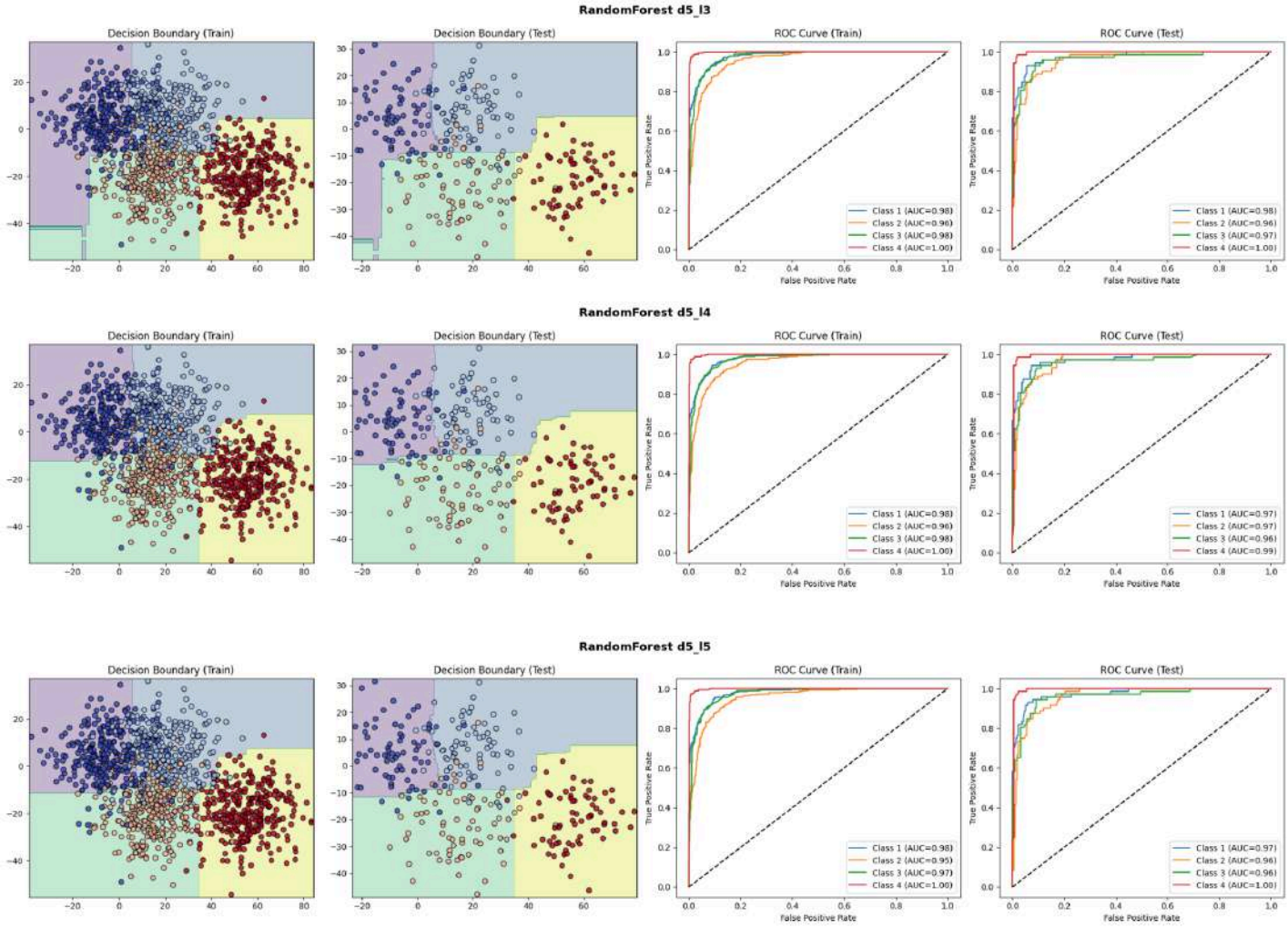


RandomForest d5_l1



RandomForest d5_l2





Similar to the previous question, the explanation to this question was not asked in this question, rather the next question, so i'll provide a detailed analysis in the next question.

QUESTION 6:

In this question, I will be answering the question of how good each and every model looks like for the dataset and which ones perform better than the others.

I'll answer this question combined, looking at the plots and the metrics saved in the csv file too.

Starting with the Logistic Regression model, it works with clear linear boundaries and as the data in the dataset 1(v0) is separated out, its quite easy to find them and differentiate them. Hence, we see perfect 1s across the board for the Logistic Reg model for v0, which is expected. For v1, Logistic Reg starts to fall back. It gets values of precision, recall, F1 which are lower than the average seen in all the models. This means the model wasnt performing that great. Its quite obvious from the plots too. The linear boundaries werent able to get a hold of the non-linear pattern in the dots placements. Hence the linear boundary actually failed to take a note of the exact pattern. A similar case was with the v2 dataset. Here, in theory the model should have fallen back a lot more, but as the data was so mixed up, even the best models were not able to make a clean separation, hence the model was still at an average or just below average performance, which clear from the acc=0.860, prec=0.859, etc, which are lower than the others but not too different, due to the large mixed form of the data.

I talked about Log Reg earlier, but not the Log Regr(with poly = 2). This was because there is not much of a difference between the two. I kept the poly at 2 to add more features but not to add too many as to bring in an overfit model. The Log Reg (poly) model has similar stats as the Log Reg model but its only slightly better at most things, hence it also has a similar story.

The SVC models are some of the most consistent models which always place at an above average range without much variation in their behaviour. This is mainly the case with the rbf model. The linear model performs with some similarity to the Log Reg model but in a slightly better manner. This is clear from the accuracy and precision data. The rbf model performs almost on par with its linear counterpart, but as its boundary is able to bend in ways the linear SVC cannot, it captures a greater pattern, hence its slightly better than the linear model. The SVC model is perfect in the v0 case where the classes are clearly separated whereas in the case of v1 and v2, we see drops. The v1 model performs good without any signs of overfit. The train and test stats didnt show much of a difference. There was a 0.1-0.2 difference but i believe it just to be due to the point selection. The v2 model shows a very sharp decrease in the model

effectiveness. Its still better than many others, but as the decrease is in all models, I believe it the mixing of the datapoints thats causing this issue. The plots support this theory, especially the ROC plots. The SVC isnt an outlier in any case.

The Neural Networks are the worst performing models in the case of v0. There the data was already very predictable as the classes were separate by a huge margin. But the Neural Network failed to reach the separation line. This could be an issue with the number of iterations, which maybe if increased would allow the Neural Networks to also converge. But 5000 already seemed like a lot and i didnt want to risk any overfitting. Hence i left it as it is. Contrary to what was seen in the case of v0, the Neural Network becomes one of the strongest models in the case of v1 and v2. Their accuracy values are an example to this, along with many other statistics. They show little overfitting which can be counted as negligible as in most cases, its only a max of 0.1-0.2 and the plots dont have much of an issue, even though some slight issues do pop up. One major observation here, inside the Neural Networks is that the lower ones always work better. In the sense the 5 and 5,5 models work the best. The higher models of the Neural network show some bit of overfitting and also too high complexity. It seems as though the simplex models look at the problem in a simper way yielding better results than the higher models which apply too much complexity following the patterns and in some way actually worsening the model. It seems like some overfitting is occurring for the high complexity models.

The Random Forest had the best conclusions of all the models. The Random Forest Classifier exactly showed how a lot of complexity can lead to an overfit model or at least slightly overfit model. The v0 dataset didnt reveal much as all the models of the RF got perfect scores. The plots were also clean. The model in the v1 case showed surprising results. As v1 had some separation between the data and as it was not completely mixed, this revealed a lot about how the RF Classifier performed. The higher complexity models, ie: the deep rooted trees showed a big difference in the train and test statistics, which was close to 0.4, the highest of all the models in this assignment. This seriously meant that the RF Classifier worked on the train data much better than the test data. Normally in models, 0.4 difference is not that high to say overfitting, but here as we are looking a set of very basic models, compared with each other, the onset of overfitting is enough to state an overfit model. The RF model in the v2 case behave in a similar way, but the models are much closer together in values as the data is quite mixed up and the entire spectrum of models we have are struggling here. But from the v1 dataset its clear choosing a tree with minimal depth and

items per leaf gives the best trees with as low overfitting as possible. The higher complexity models overfit. This is very evident, i would say one of the major finding we can actually easily see from the plots too. The overfitting is quite visible in the plots of the RF Classifier. The ROC plots for all the v2 models look bad as the data is quite mixed hence, i havent talked a lot about ROC.

2. Part 2:

QUESTION 7:

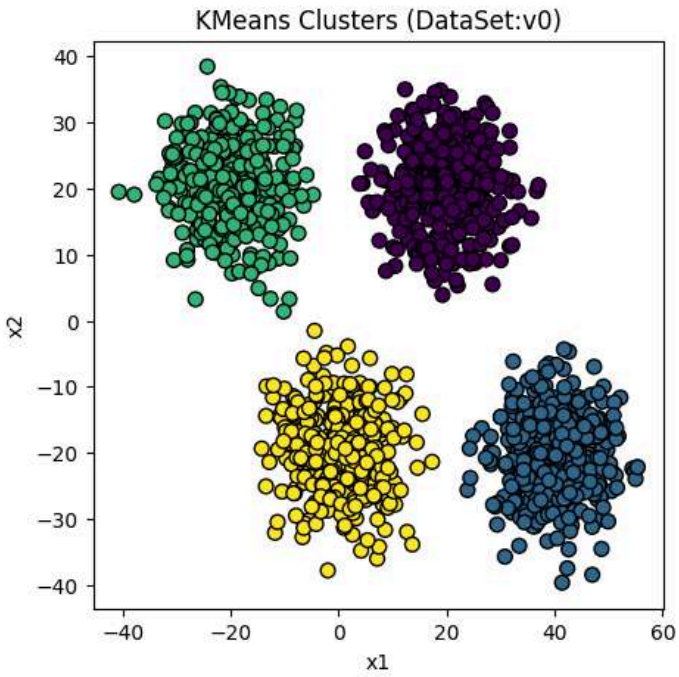
For this question, my idea is to create two clustering algorithms to work on the 3 datasets and play with the number of clusters. I know the ball park range for the cluster numbers, but still, i'll play with them a bit and find out what happens in different cases.

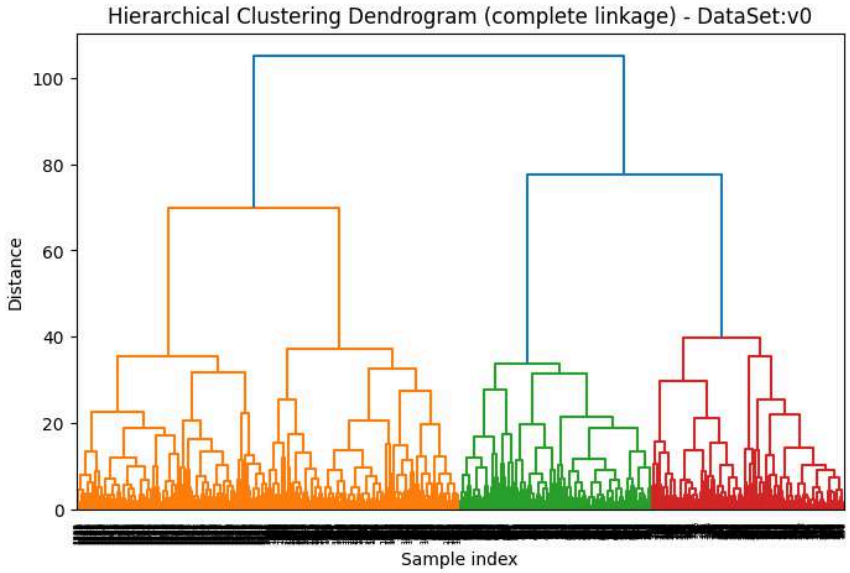
For the v0 case, its pretty obvious that as they are well separated clusters, the number of clusters will be 4. So for v0, i'll put in the cluster numbers as 3, 4 and 5 and see how the values change.

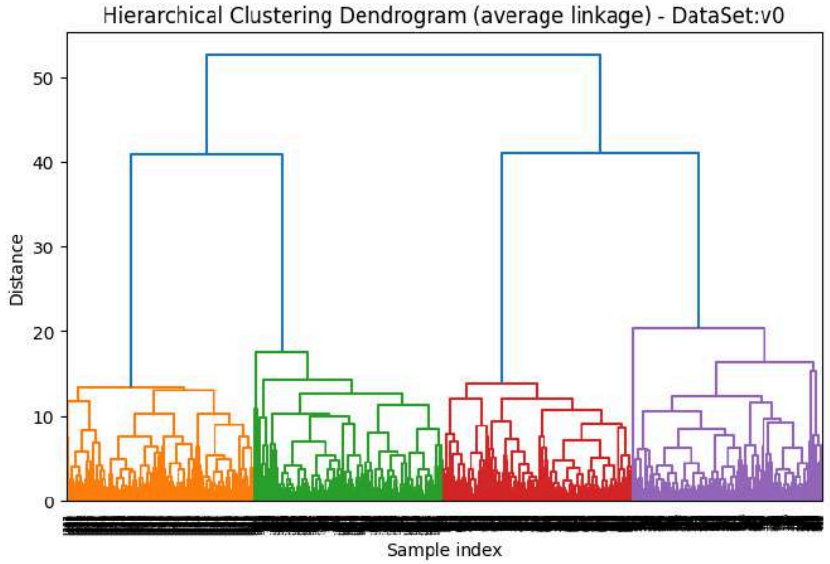
This might or might not be the case for v1 and v2 though.(cluster num = 4)(because the data is mixed and no clear separation is there)

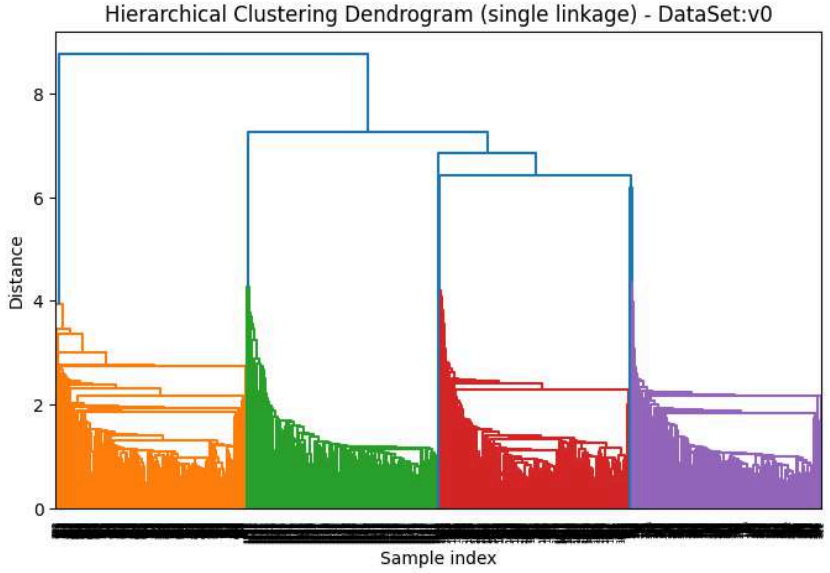
For v0 and number of clusters = 4,

	Algorithm	Silhouette	Calinski-Harabasz	Davies-Bouldin
0	KMeans	0.711366	5692.814940	0.391861
1	Hierarchical-complete	0.711366	5692.814940	0.391861
2	Hierarchical-average	0.711366	5692.814940	0.391861
3	Hierarchical-single	0.366891	962.070432	0.725926



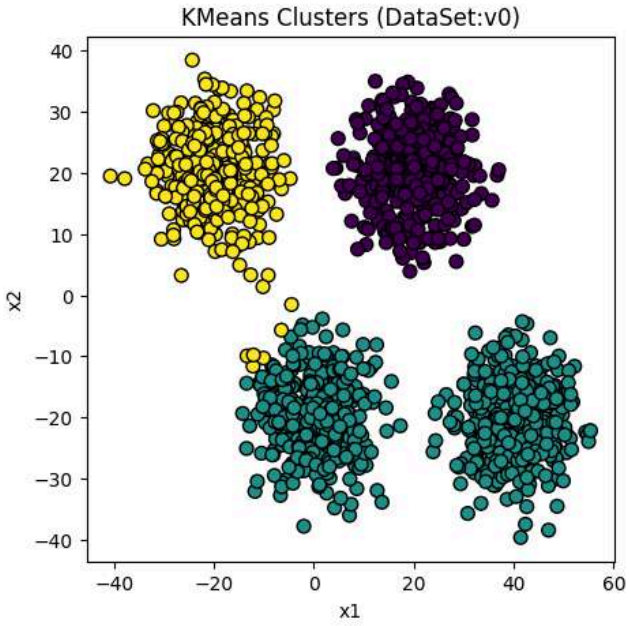

Hierarchical Clustering Dendrogram (complete linkage) - DataSet:v0
Distance
Sample index
0
20
40
60
80
100

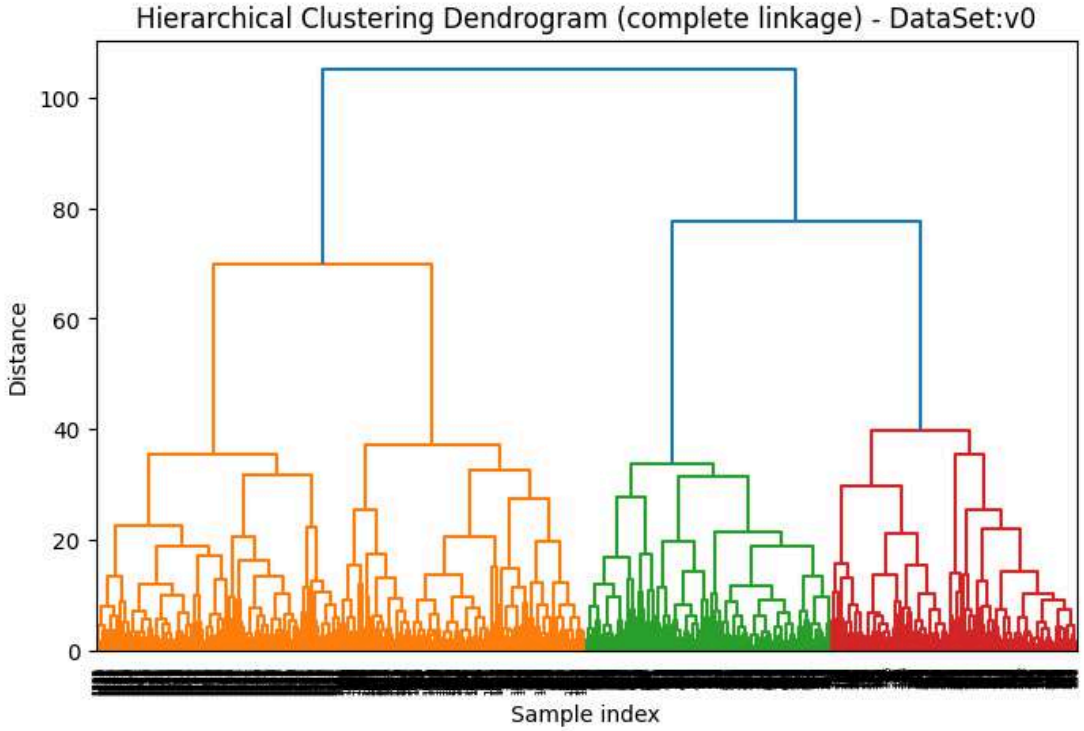

Hierarchical Clustering Dendrogram (average linkage) - DataSet:v0
Distance
Sample index
0
10
20
30
40
50

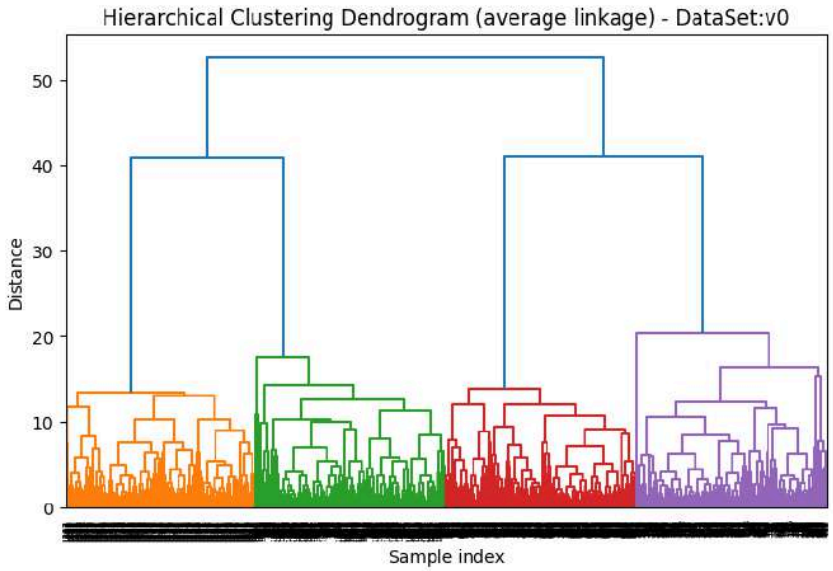

Hierarchical Clustering Dendrogram (single linkage) - DataSet:v0
Distance
Sample index
0
2
4
6
8

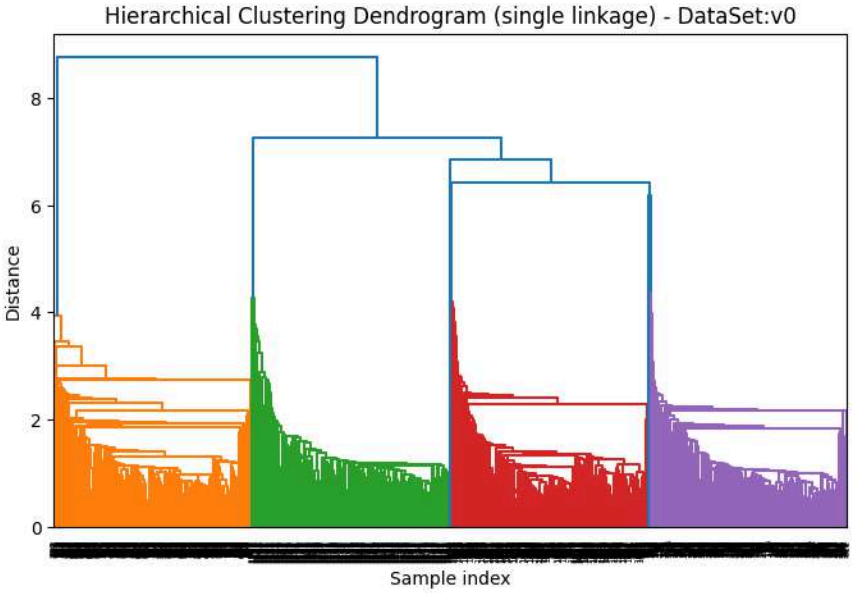
For v0 and number of clusters = 3,

	Algorithm	Silhouette	Calinski-Harabasz	Davies-Bouldin
0	KMeans	0.563013	1854.420424	0.643954
1	Hierarchical-complete	0.564299	1848.888727	0.644547
2	Hierarchical-average	0.564299	1848.888727	0.644547
3	Hierarchical-single	0.513225	1437.133101	0.770115



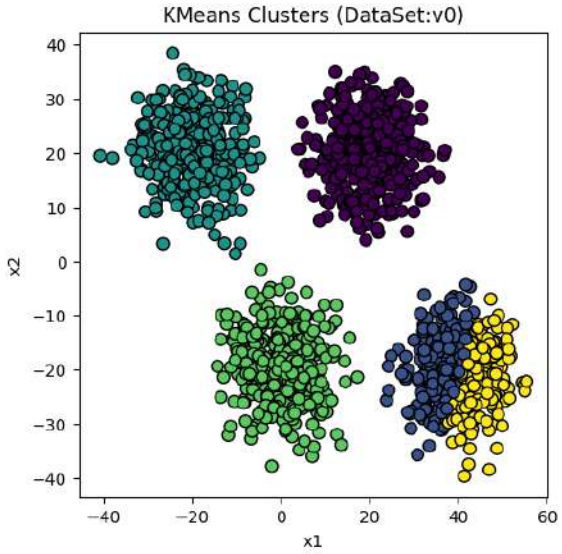


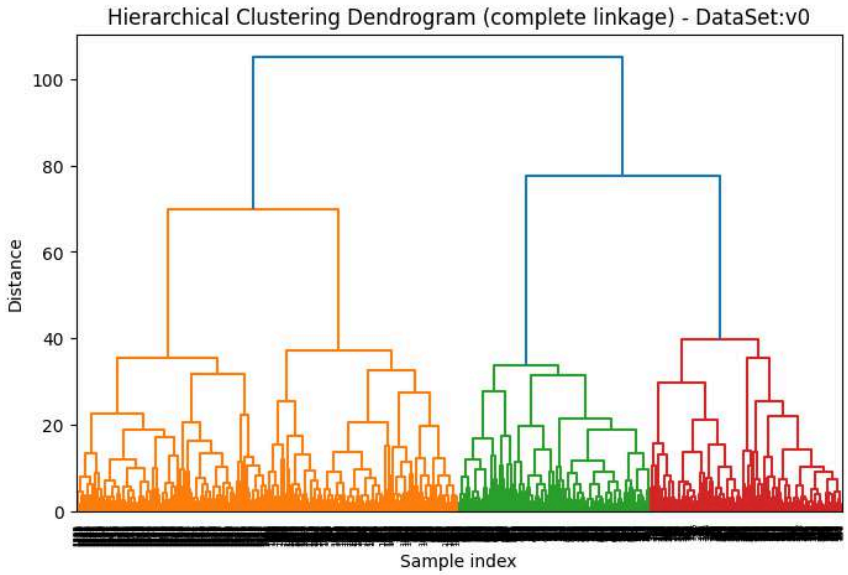


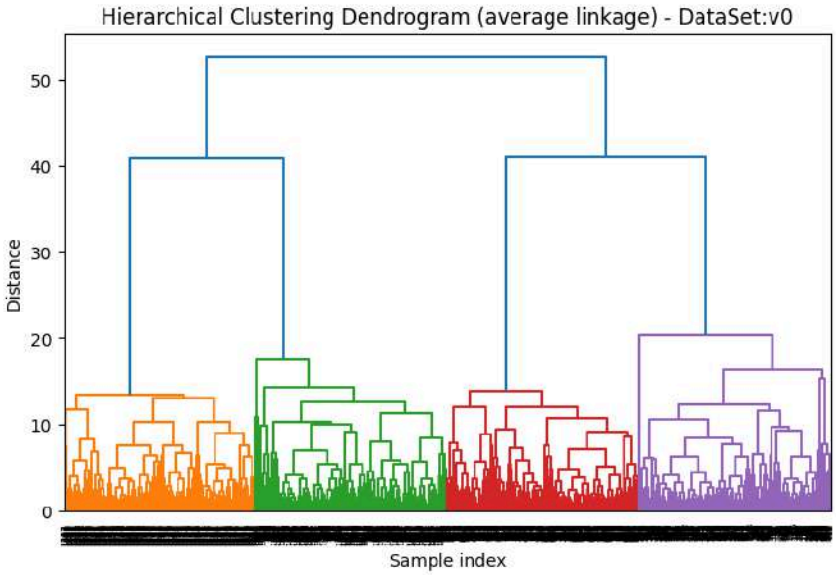


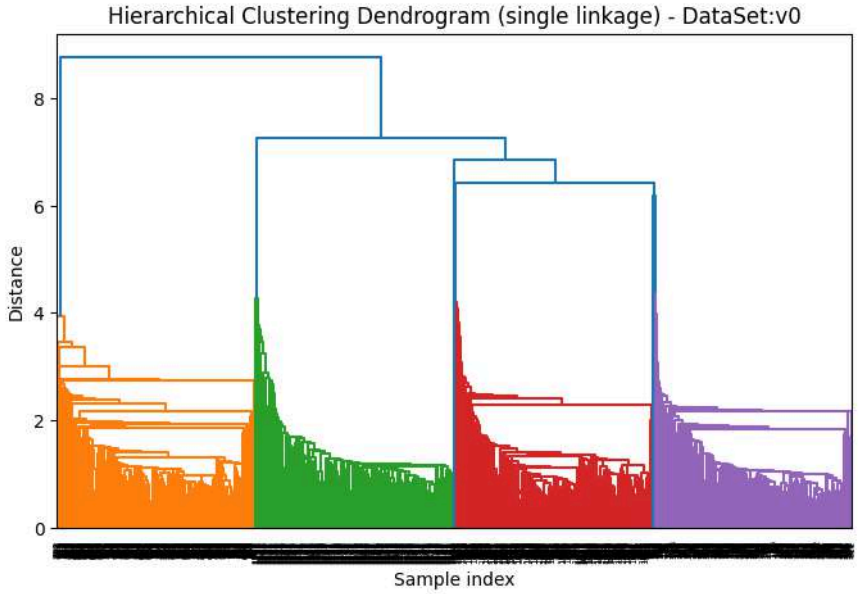
For v0 and number of clusters = 5,

	Algorithm	Silhouette	Calinski-Harabasz	Davies-Bouldin
0	KMeans	0.602719	4694.462434	0.739883
1	Hierarchical-complete	0.587296	4583.692266	0.823568
2	Hierarchical-average	0.637992	4298.082701	0.423538
3	Hierarchical-single	0.630779	4280.902943	0.403086


KMeans Clusters (DataSet:v0)
x1
x2


Hierarchical Clustering Dendrogram (complete linkage) - DataSet:v0
Distance
Sample index


Hierarchical Clustering Dendrogram (average linkage) - DataSet:v0
Distance
Sample index

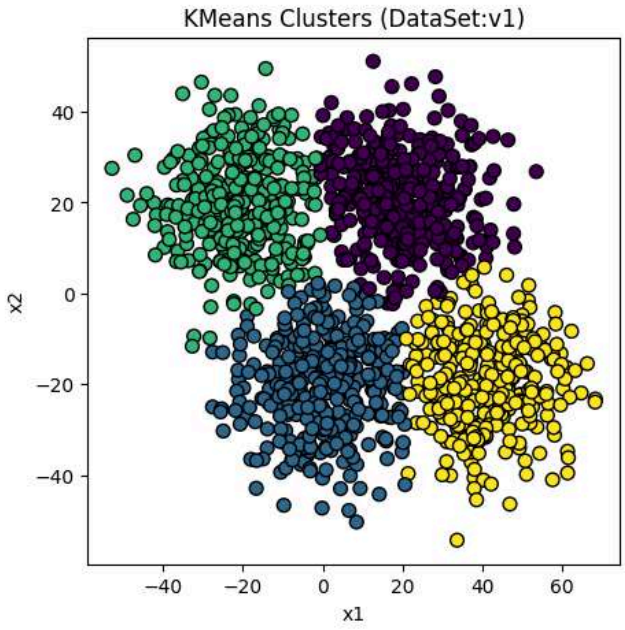


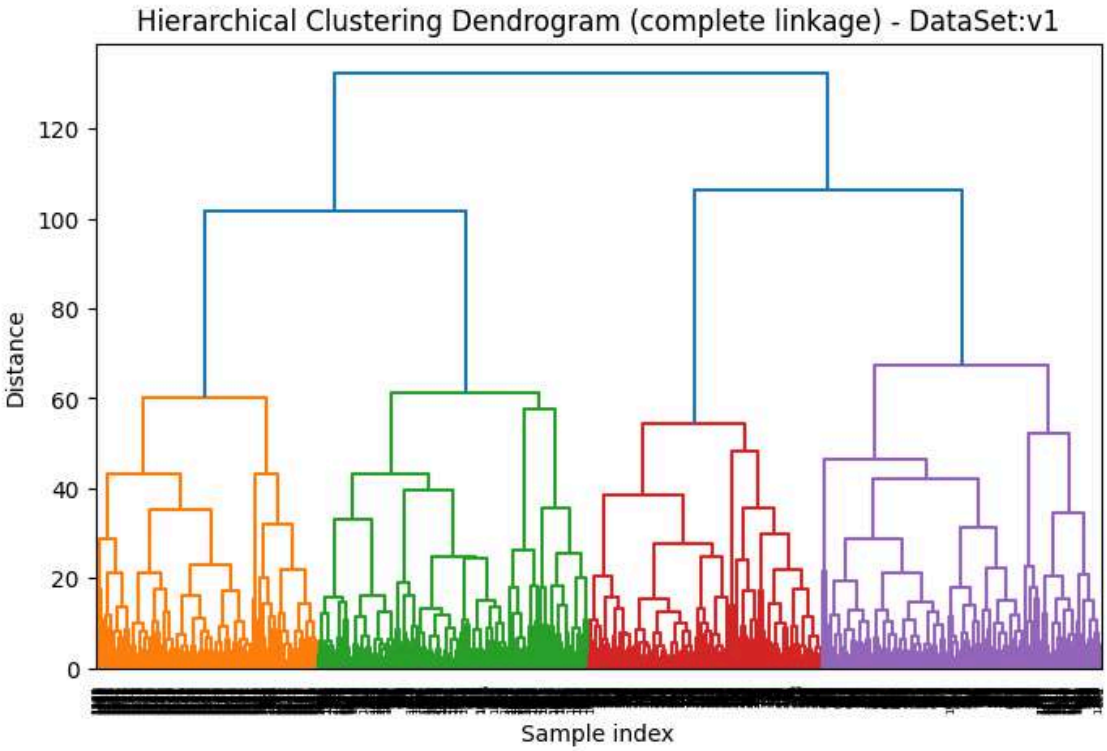
Looking at the above 3 cases, we see a very interesting result. The Kmeans clustering technique fixes the number of clusters we want and give a result appropriately, hence we see 1 less or 1 more cluster being in a place it shouldnt be. To make things more numerical and less visual, focus on the DB score. It should be as low as possible. We see that the cluster number 4 for the KMeans has the least DB Score. So logically, and visually it does make sense that the computer believes the optimal number of clusters to be 4. This is because there are 4 neatly separated clusters. Applying the same logic to Hierarchical clustering, we see that the values bounce around a lot but still on average the clusters = 4 has the least DB score. Another major thing is that, we see that visually, whatever the number of clusters we start at, the dendrogram has 4 separate colours(leaving the complete case, its causing issue for the clusters=4 case too). As colours are allotted on the basis of distance between the clusters, the computer itself is trying to tell us that, there are only 4 clusters that should be there or there are only 4 ones which stay separated from each other in a good way. This is quite an interesting and important result.

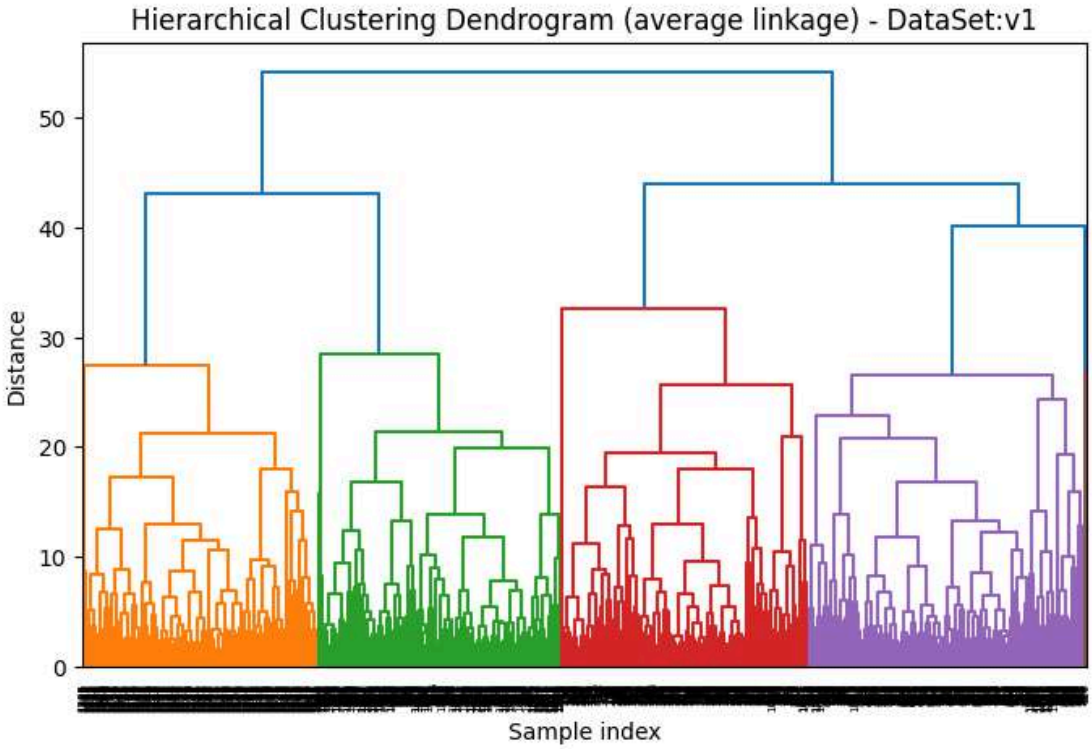
This means the algorithm started at some starting points but still shows us the results in the way, the best optimized solution or number of clusters lies.

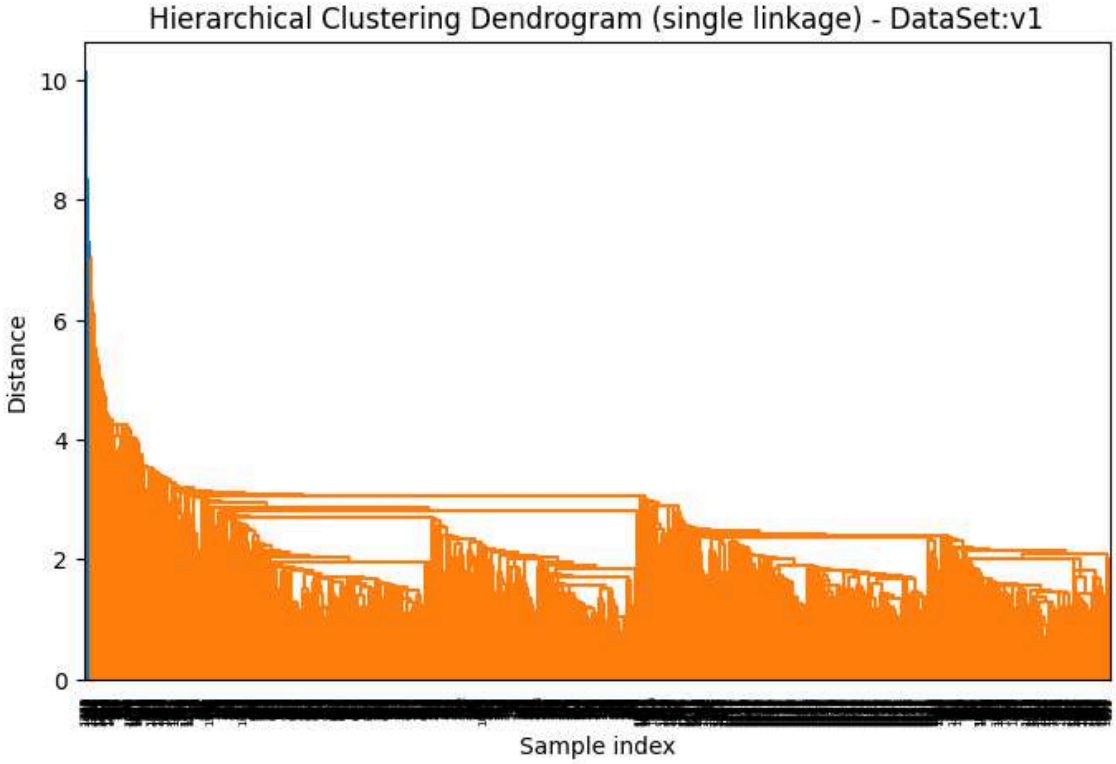
For v1 and number of clusters = 4,

	Algorithm	Silhouette	Calinski-Harabasz	Davies-Bouldin
0	KMeans	0.518400	2145.300461	0.631055
1	Hierarchical-complete	0.466145	1738.922832	0.678357
2	Hierarchical-average	0.506444	2046.344126	0.639017
3	Hierarchical-single	-0.126007	2.706717	0.583387



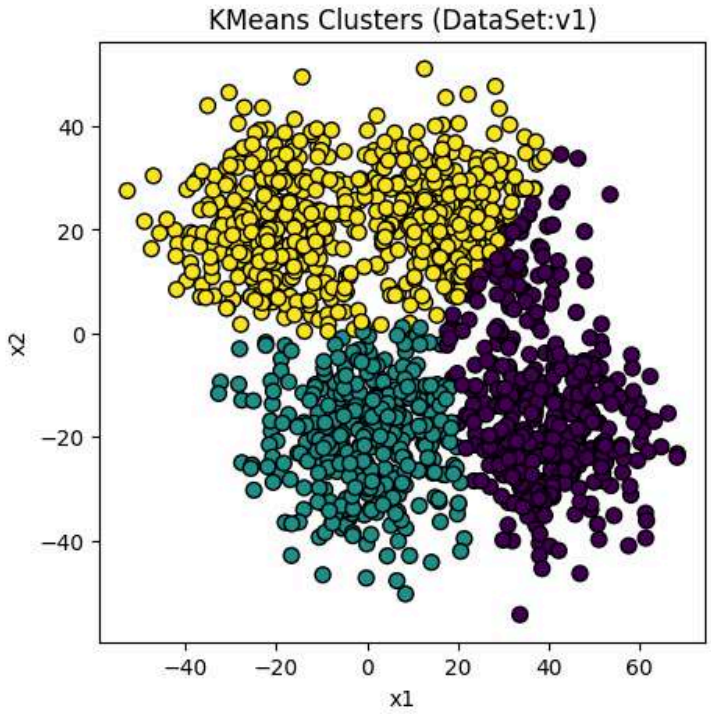


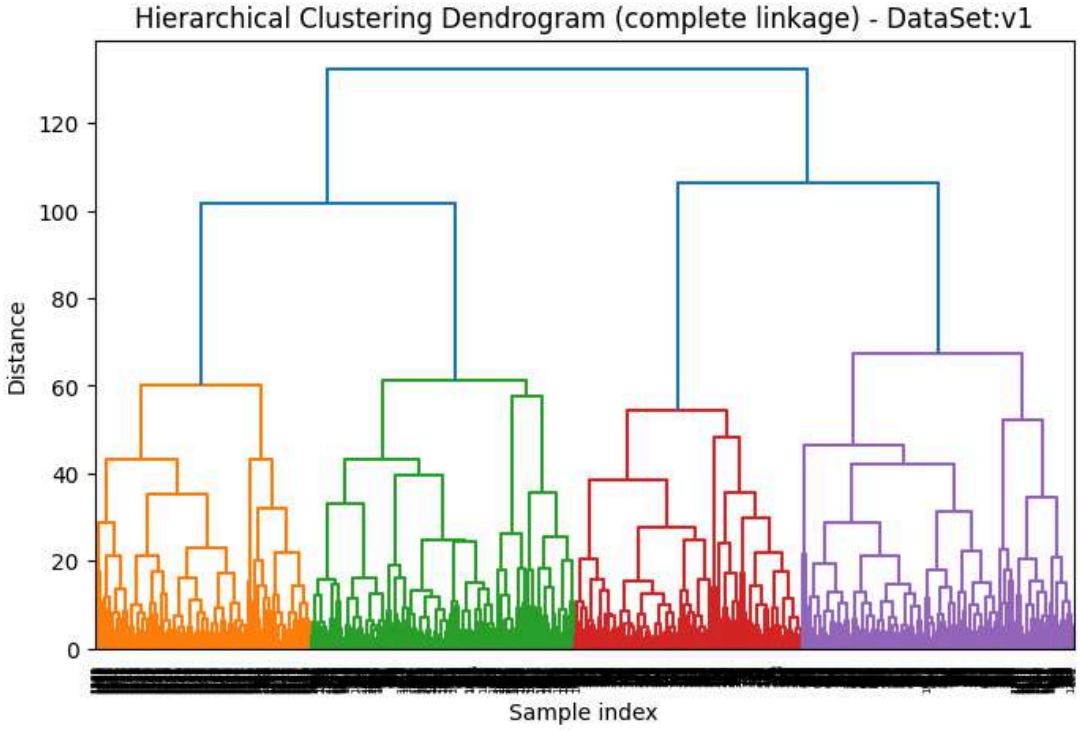

Hierarchical Clustering Dendrogram (average linkage) - DataSet:v1
Distance
0
10
20
30
40
50
Sample index

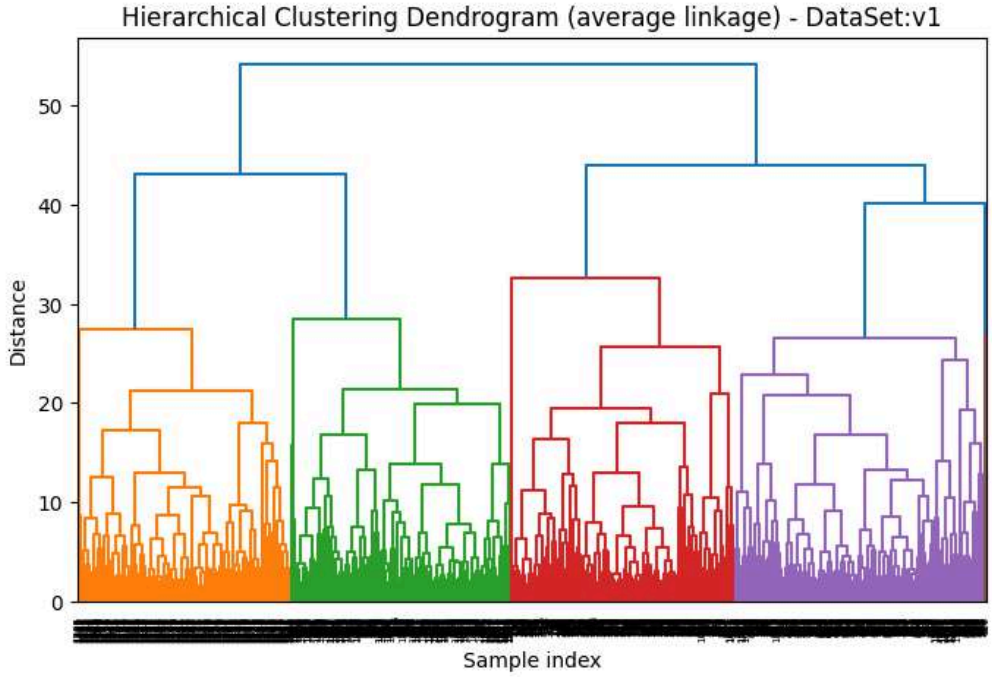

Hierarchical Clustering Dendrogram (single linkage) - DataSet:v1
Distance
0
2
4
6
8
10
Sample index

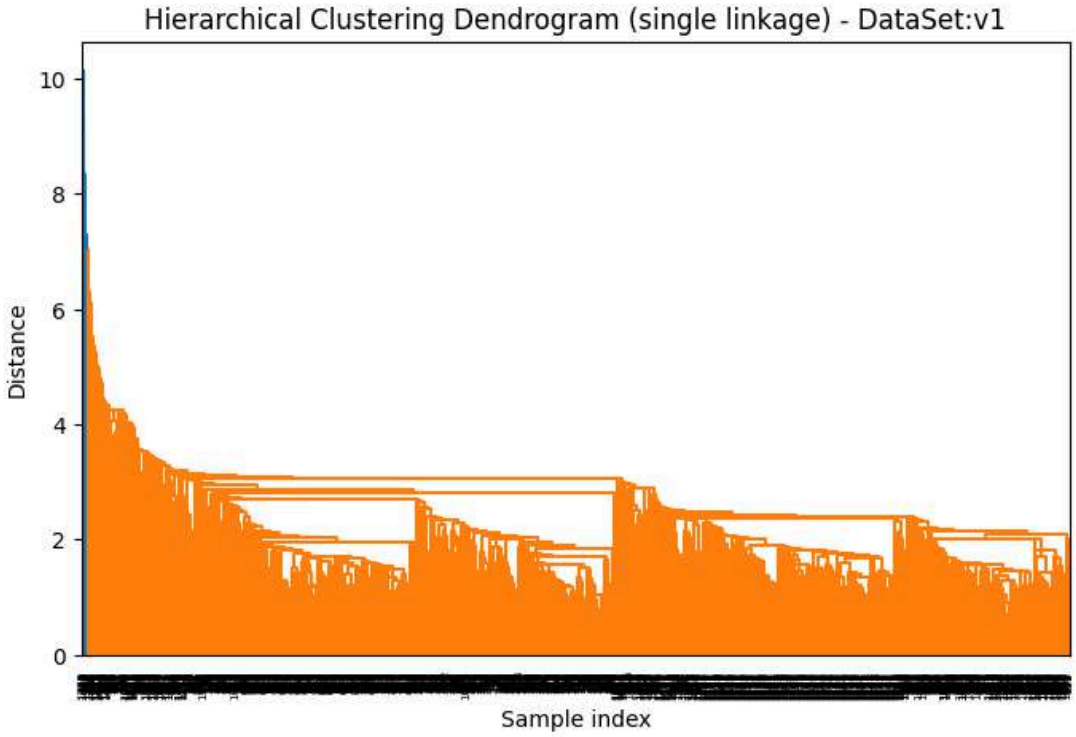
For v1 and number of clusters = 3,

	Algorithm	Silhouette	Calinski-Harabasz	Davies-Bouldin
0	KMeans	0.427810	1343.816278	0.804923
1	Hierarchical-complete	0.389278	1103.057057	0.883336
2	Hierarchical-average	0.422336	1252.799794	0.826160
3	Hierarchical-single	0.059076	2.714654	0.587573



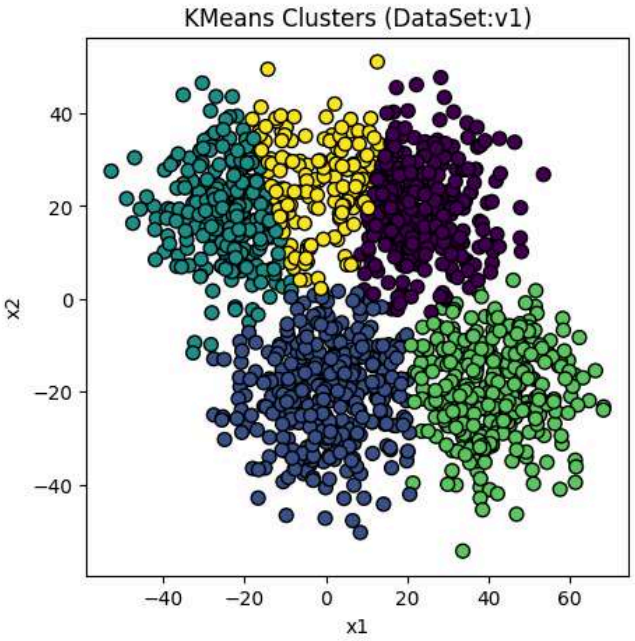


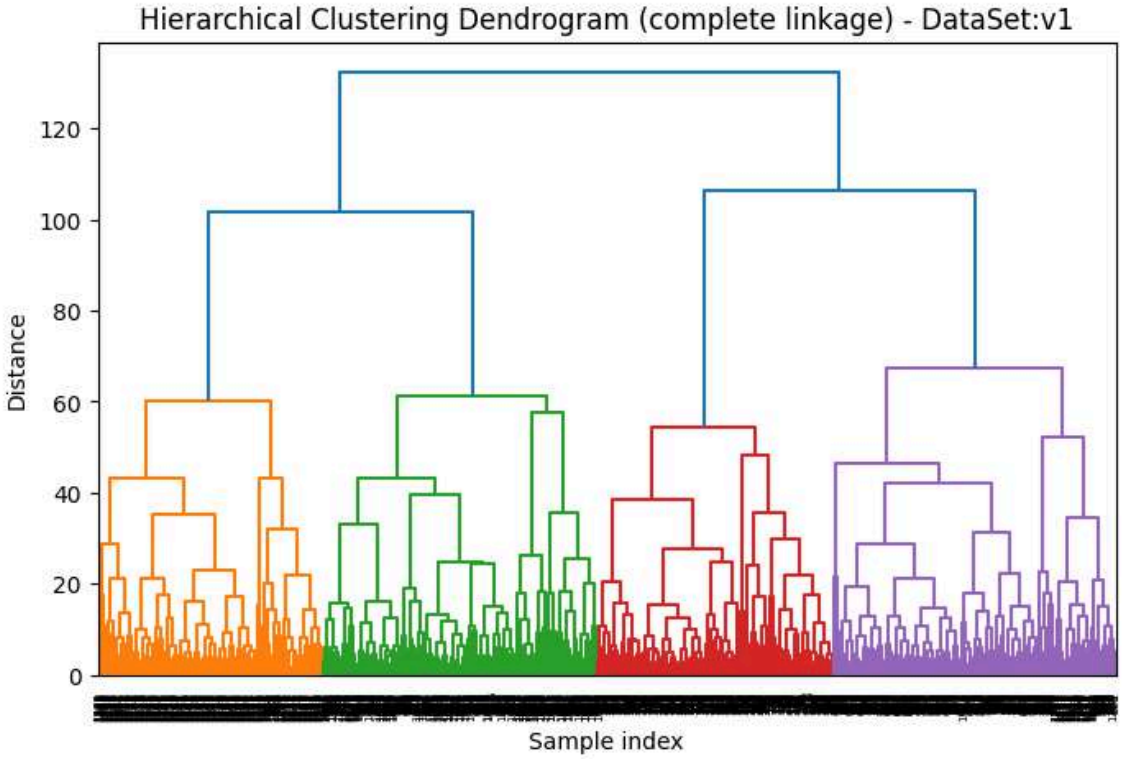

Hierarchical Clustering Dendrogram (average linkage) - DataSet:v1
Distance
0
10
20
30
40
50
Sample index

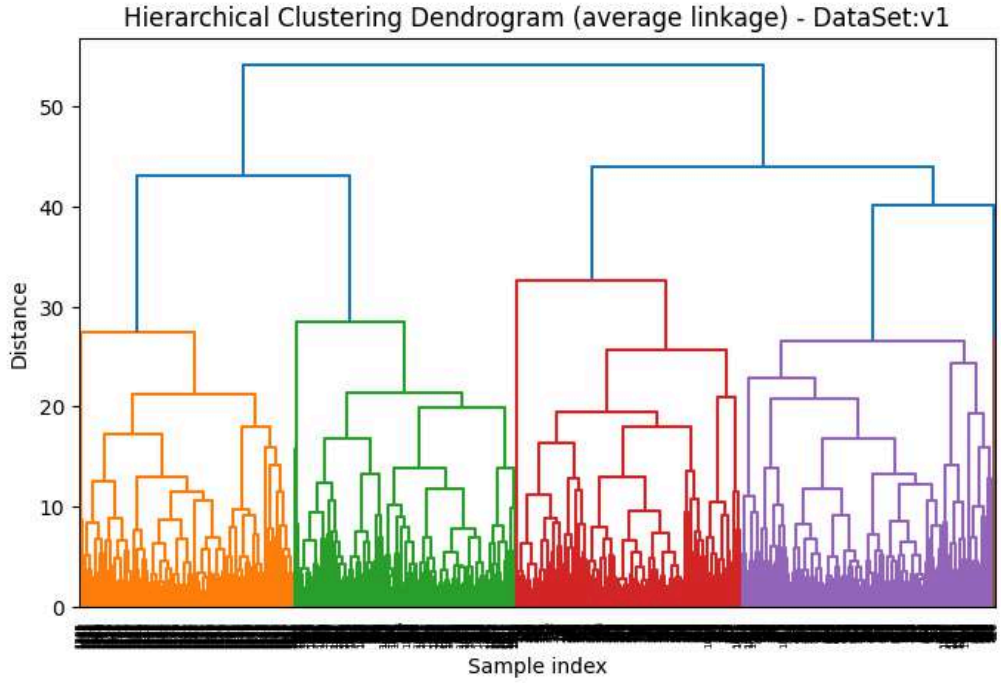

Hierarchical Clustering Dendrogram (single linkage) - DataSet:v1
Distance
0
2
4
6
8
10
Sample index

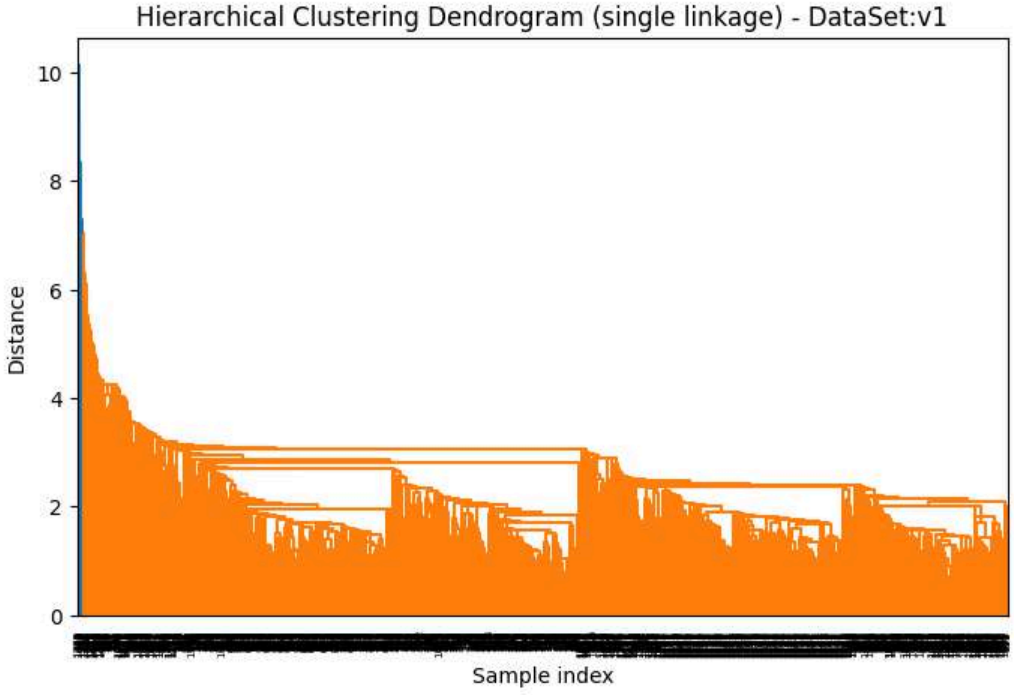
For v1 and number of clusters = 5,

	Algorithm	Silhouette	Calinski-Harabasz	Davies-Bouldin
0	KMeans	0.440427	1827.915354	0.815061
1	Hierarchical-complete	0.417635	1581.070230	0.831683
2	Hierarchical-average	0.468396	1550.189806	0.684737
3	Hierarchical-single	-0.219598	2.603053	0.591239





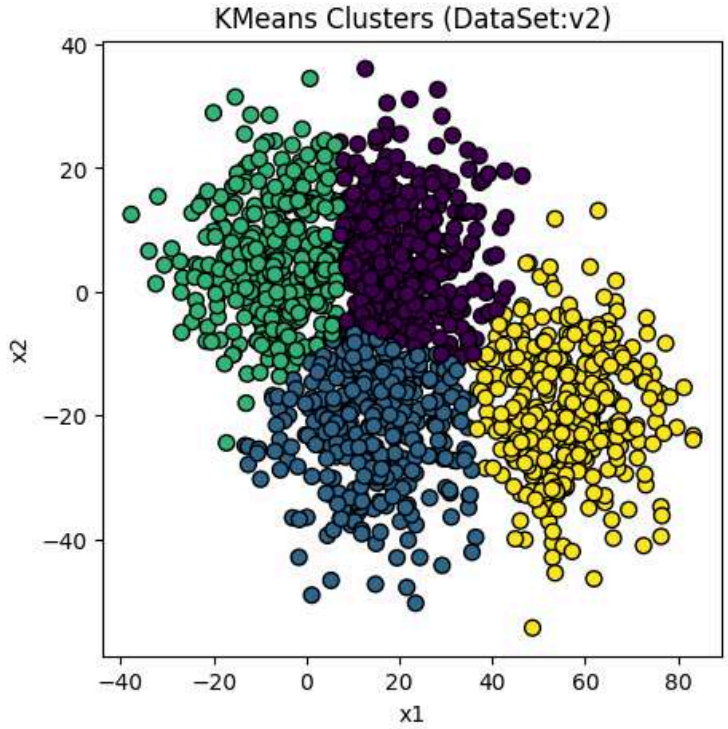

Hierarchical Clustering Dendrogram (average linkage) - DataSet:v1
Distance
0
10
20
30
40
50
Sample index

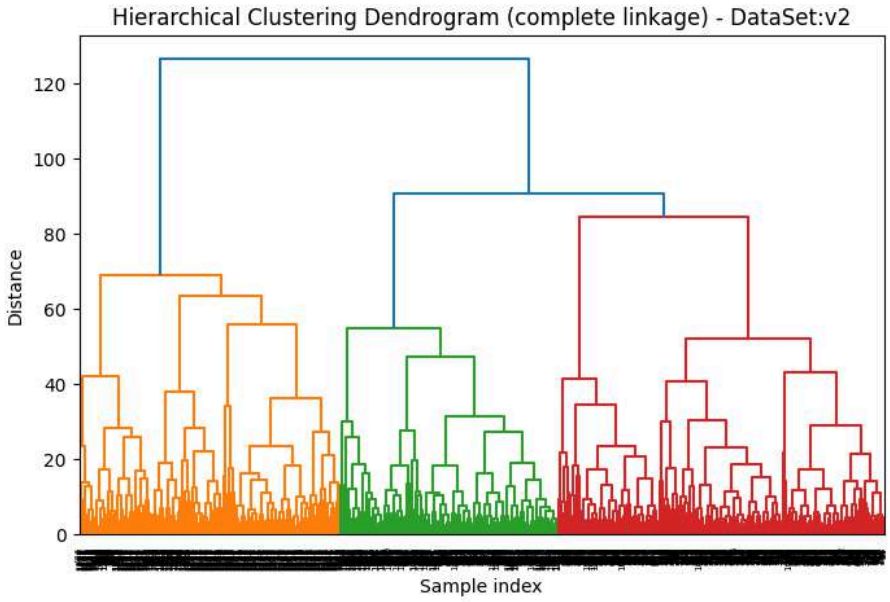

Hierarchical Clustering Dendrogram (single linkage) - DataSet:v1
Distance
0
2
4
6
8
10
Sample index

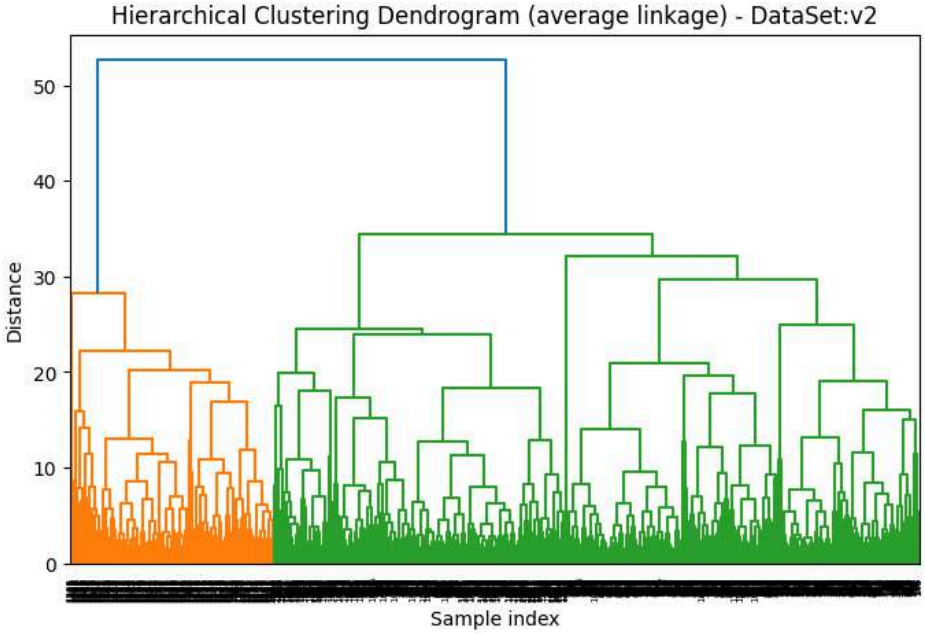
We get some similar results as the first dataset v0, where whatever the initial number of clusters, we end up with a majority number of clusters as 4. Intrinsically, this doesnt mean that there are 4 clusters. There may be lesser than or more than the specified amount, but the distance measure keeps a tab on the different clusters which have a good amount of distance between them. Clusters which are too close to each other, dont satisfy the criteria, thus the dendrogram's colour represents the number of "clusters"(not actual number of clusters we provide) which has an appropriate distance between them such that they actually have some independence of their own. The CH score and DB score also agree with this as they are low and high respectively in the case of clusters not equal to 4 case, which is unfavorable.

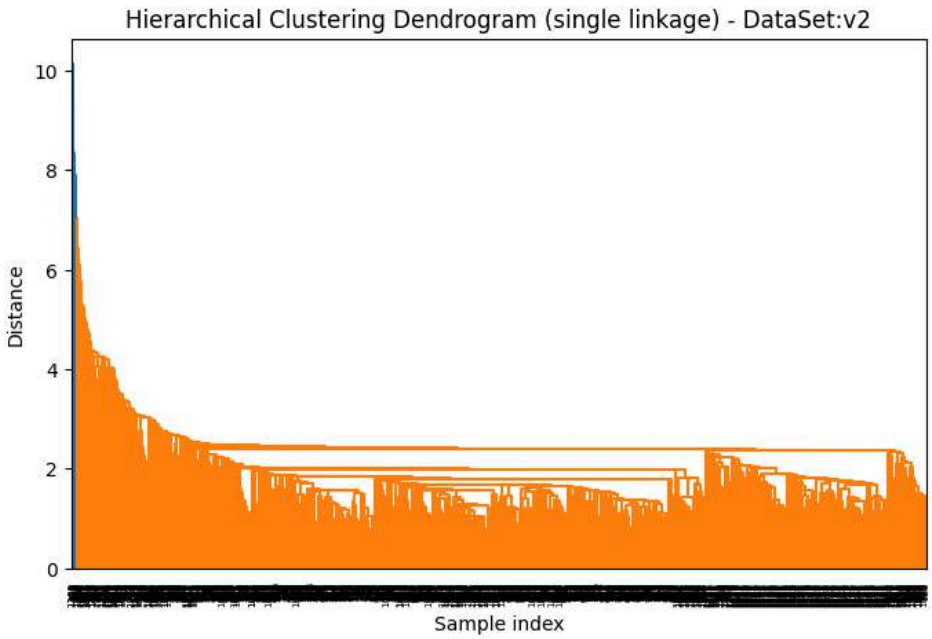
For v2 and number of clusters = 4,

	Algorithm	Silhouette	Calinski-Harabasz	Davies-Bouldin
0	KMeans	0.434750	1801.982370	0.777800
1	Hierarchical-complete	0.368436	1416.760025	0.806122
2	Hierarchical-average	0.329516	1019.274460	0.920648
3	Hierarchical-single	-0.049018	2.838465	0.546112



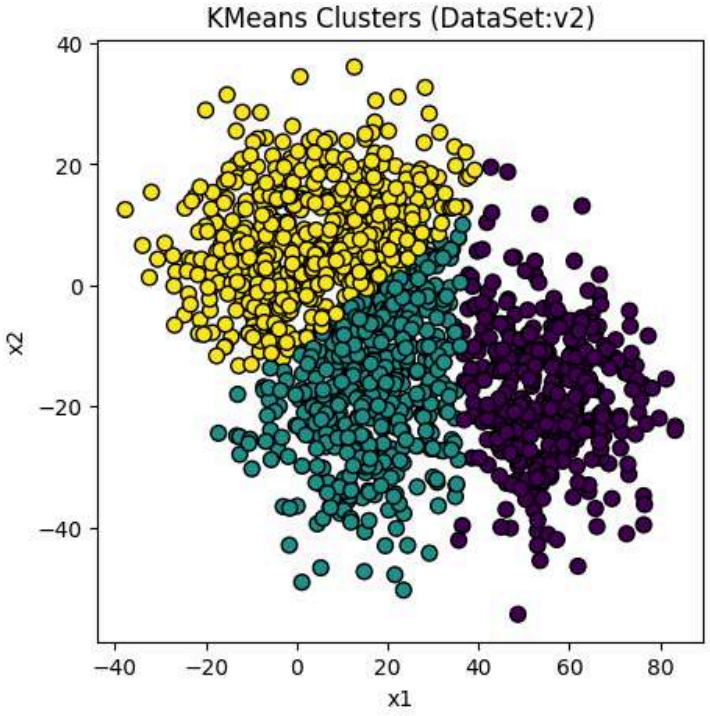

Hierarchical Clustering Dendrogram (complete linkage) - DataSet:v2
Distance
Sample index
0
20
40
60
80
100
120

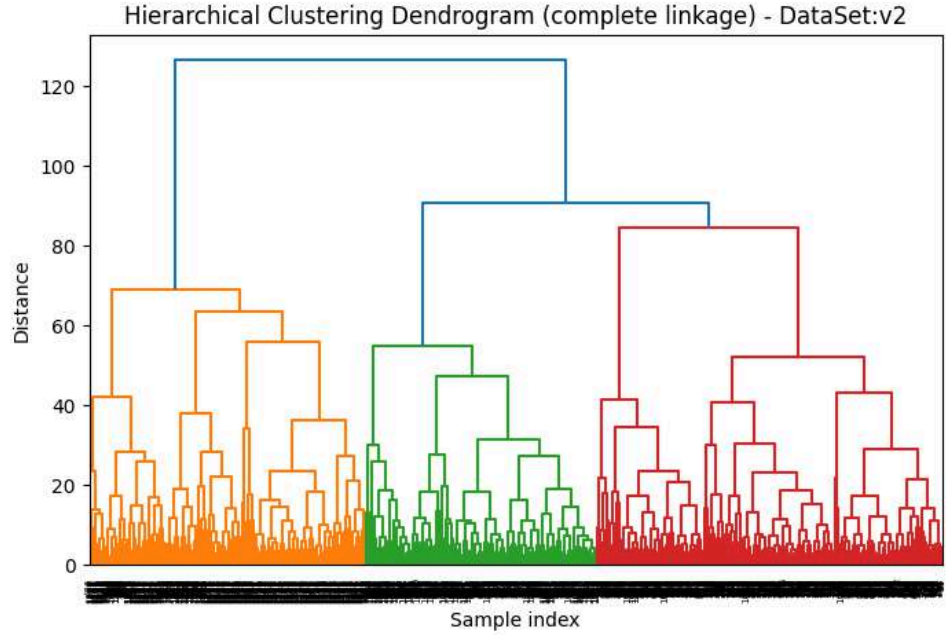

Hierarchical Clustering Dendrogram (average linkage) - DataSet:v2
Distance
Sample index
0
10
20
30
40
50

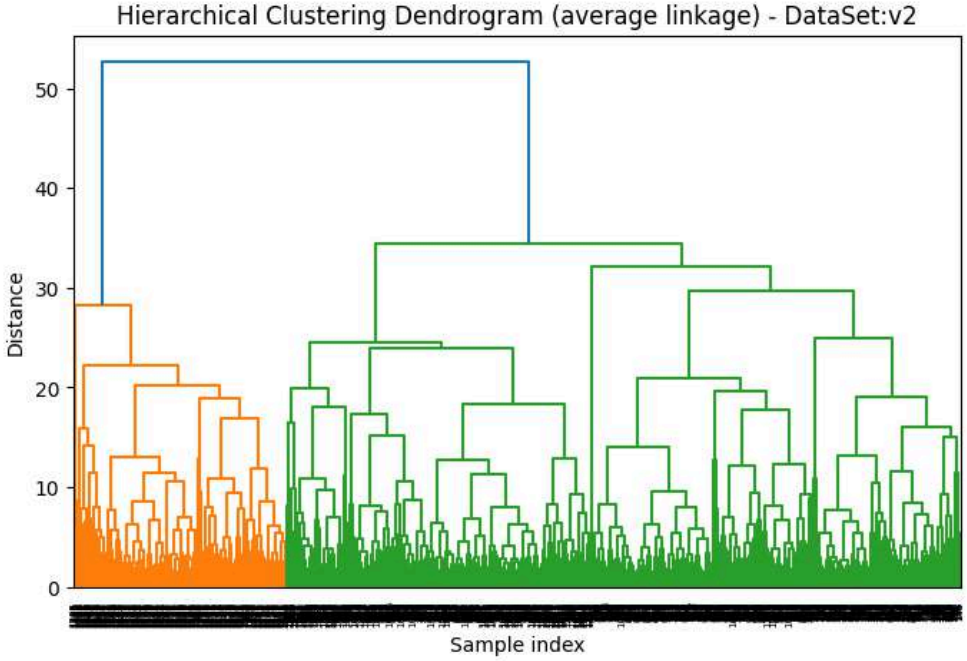


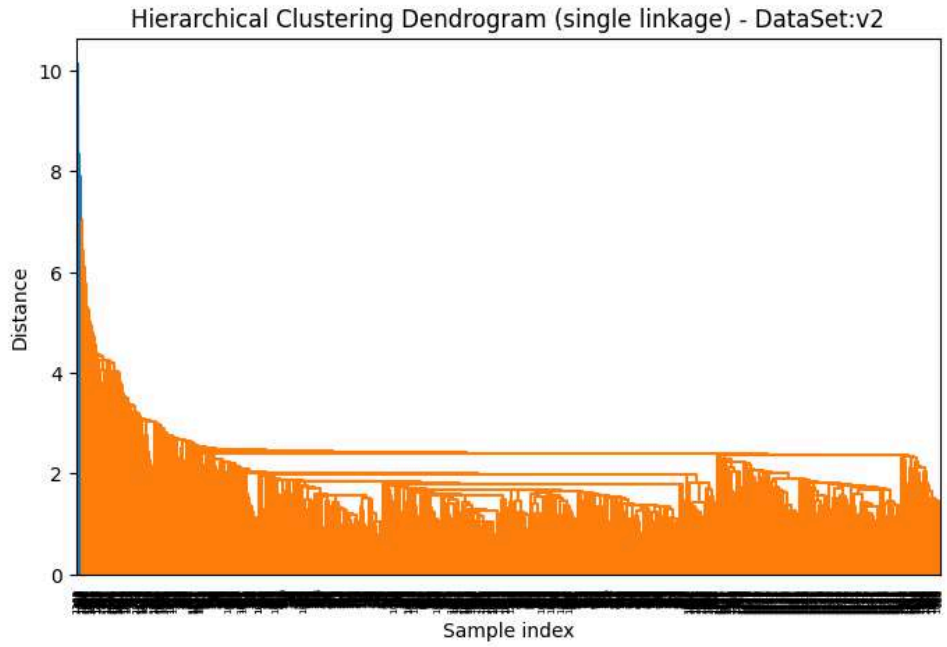
For v2 and number of clusters =3,

	Algorithm	Silhouette	Calinski-Harabasz	Davies-Bouldin
0	KMeans	0.402918	1632.883804	0.932963
1	Hierarchical-complete	0.362101	1424.202453	0.976999
2	Hierarchical-average	0.391668	1521.621548	0.951439
3	Hierarchical-single	0.166565	2.982523	0.540160



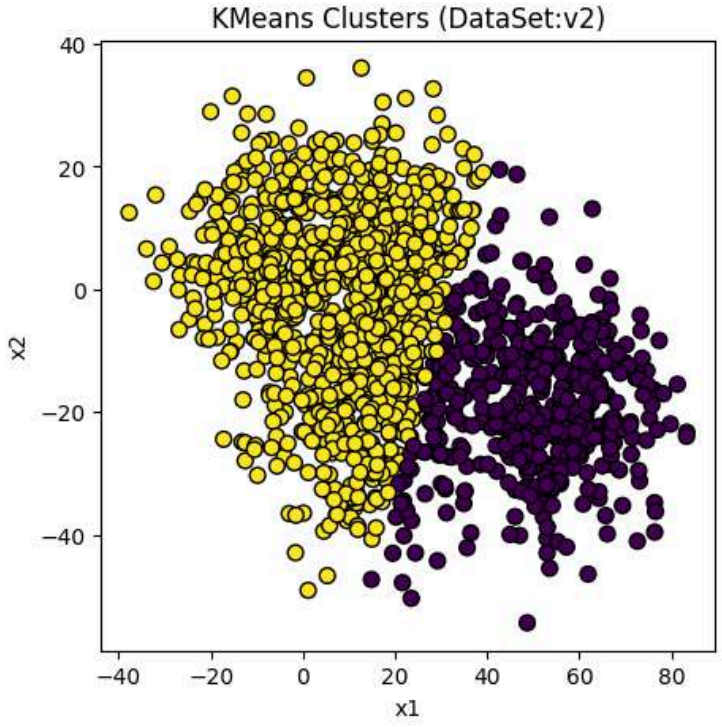

Hierarchical Clustering Dendrogram (complete linkage) - DataSet:v2
Distance
Sample index
0
20
40
60
80
100
120

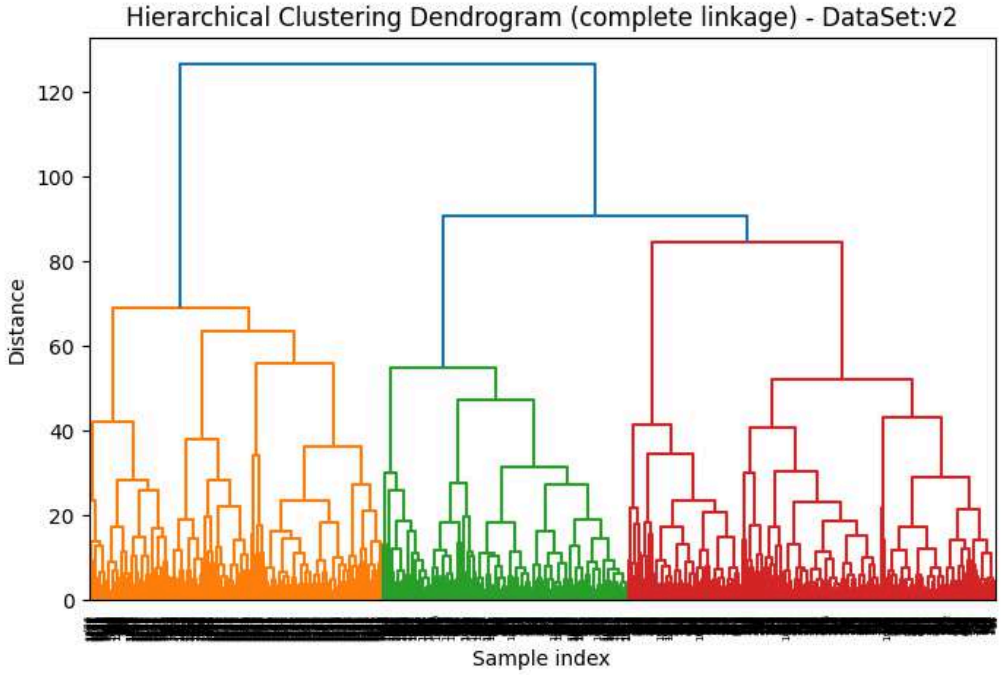

Hierarchical Clustering Dendrogram (average linkage) - DataSet:v2
Distance
Sample index
0
10
20
30
40
50

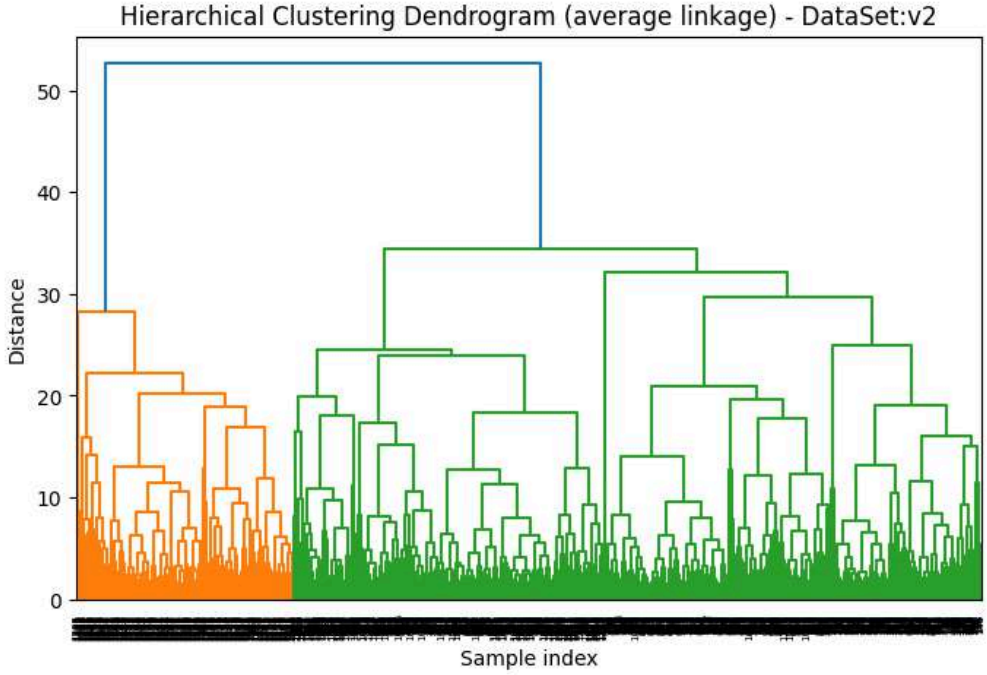


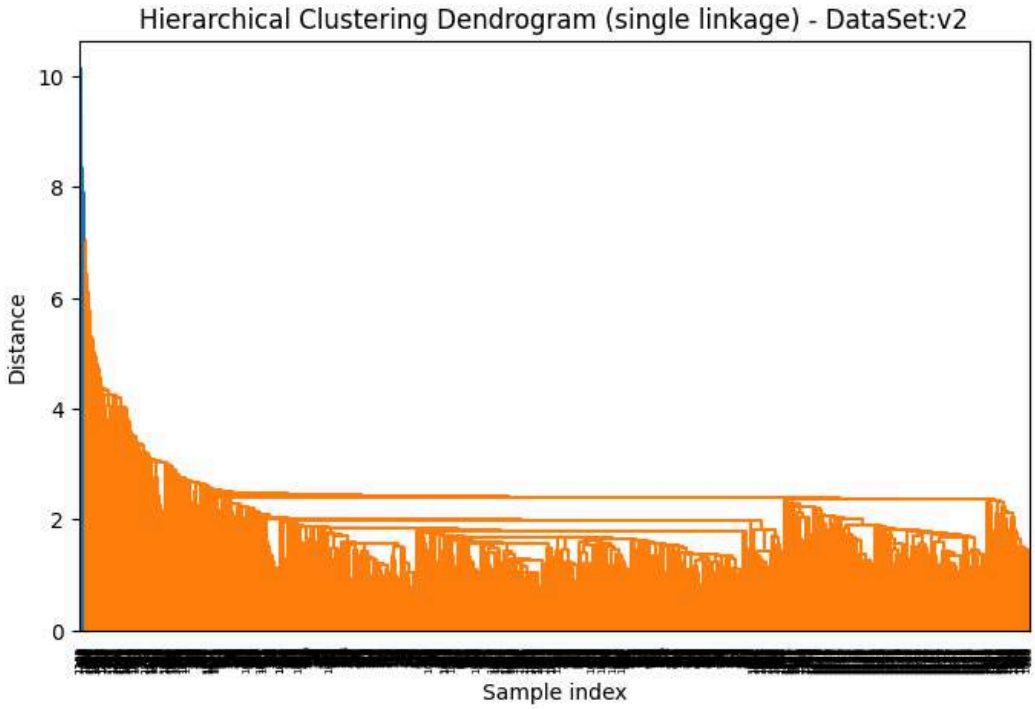
For v2 and number of clusters =2,

	Algorithm	Silhouette	Calinski-Harabasz	Davies-Bouldin
0	KMeans	0.475564	1699.934559	0.757559
1	Hierarchical-complete	0.463307	1636.325732	0.769176
2	Hierarchical-average	0.479751	1503.765003	0.671597
3	Hierarchical-single	0.307103	3.449789	0.484718




Hierarchical Clustering Dendrogram (complete linkage) - DataSet:v2
Distance
120
100
80
60
40
20
0
Sample index


Hierarchical Clustering Dendrogram (average linkage) - DataSet:v2
Distance
50
40
30
20
10
0
Sample index



In this, it is obvious that i have left my pattern of taking 3, 4 and 5. The reason i did this is because from the first cluster, for complete, single and average linkage, i didnt get any number of colour of magnitude 5. I got one of 2. As 5 didnt appear, I thought that, if the optimal number of cluster was not going towards 5, i'll have to take an extra value to the left of 4, so i took 2. Here the answer is not that clear. The answer here depends on the linkage method chosen. The linkage method chosen decided what number of clusters one has to take for the best split. Example, from all the above plots we get that for complete linkage we get that the optimal number of clusters for best split is 3 and for average is 2. Here, single isnt that great of a choice because in single we check the distance between the two closest points of two clusters and as all the clusters are so close to each other due to the mixing of data, we get a single cluster as the distance parameter is always very small compared to the threshold needed for different colours.

For v0 and v1 also, different linkages would of had different number of clusters like this, but as there was a majority, i selected only the majority number of clusters.

QUESTION 8:

In this exercise, I have learnt how to understand how to visualize given data and make some prediction of it beforehand.

I understood how to understand to implement Logistic Reg, Log Reg with polynomial features, Neural Networks, RF Classifiers and also SVC. I understood in depth how they change when the given data has classes which are close together, far apart, etc.

I also understood how to predict how a plot will look by simply looking at its metrics like AUC, Precision, F1, etc. They are the key to how the data is arranged and how the classifier makes clusters from them.

I understood how to find out whether any model is overfitting or not by looking at the plot and also by comparing its test and train metrics.

Mainly in this exercise I learnt a lot of clustering. How to implement it was easy. But the major thing i learnt was using the metrics associated with it to understand how the algorithm works.

In clustering i understood how Kmeans works in the case of optimum number of clusters and also non-optimum conditions and using hierarchical clustering to understand the best optimum.

Most of my learning are in the explanations themselves which I understood by analysis of the data.