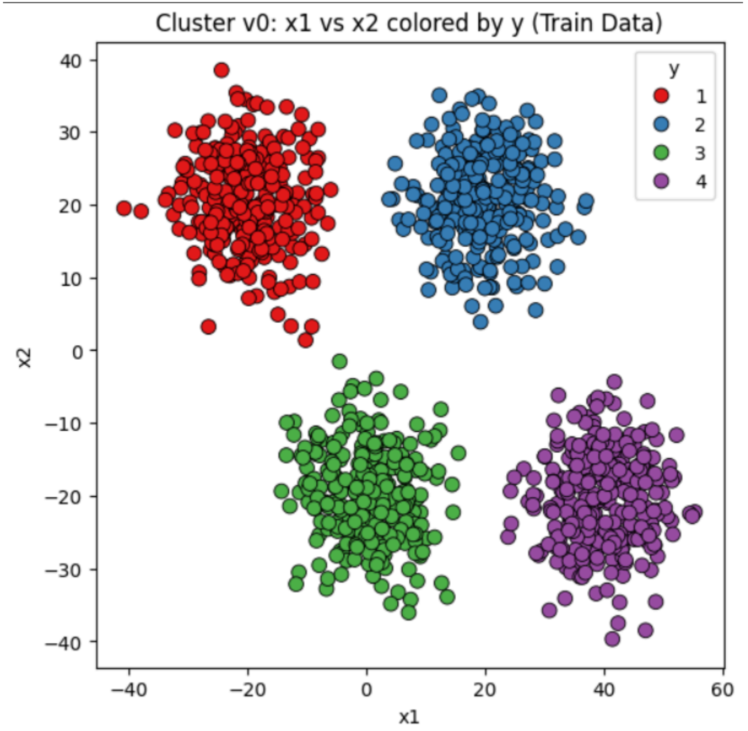
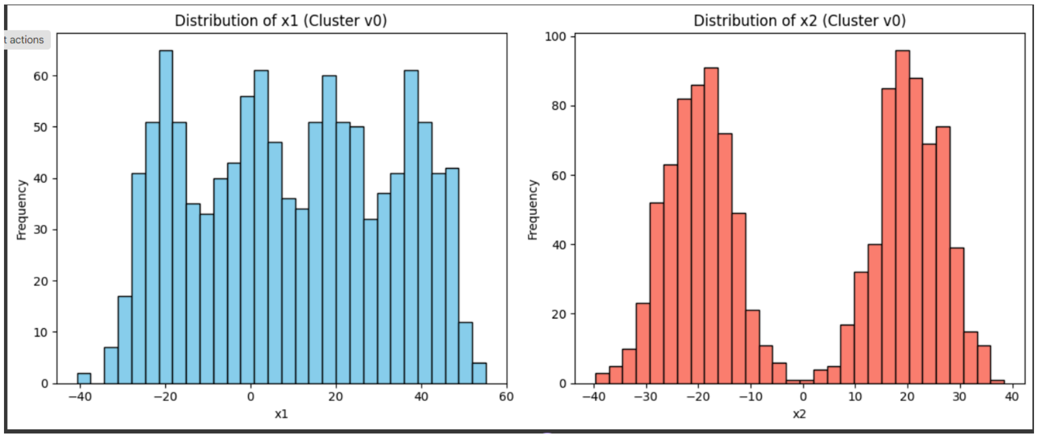
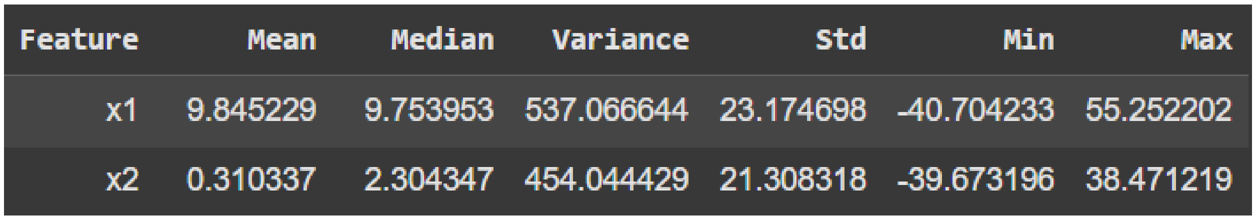
E4 DS 203 - 24B2224



In the x1 vs x2 plot of cluster v0, it is very evident that there are 4 separate classes (y = 1,2,3,4). Therefore multinomial logistic regression (softmax) can be used as a classifier.The classes look roughly linearly separable in feature space. Also the points seem to be concentrated in a region for a particular class except for some outliers in each one of them.

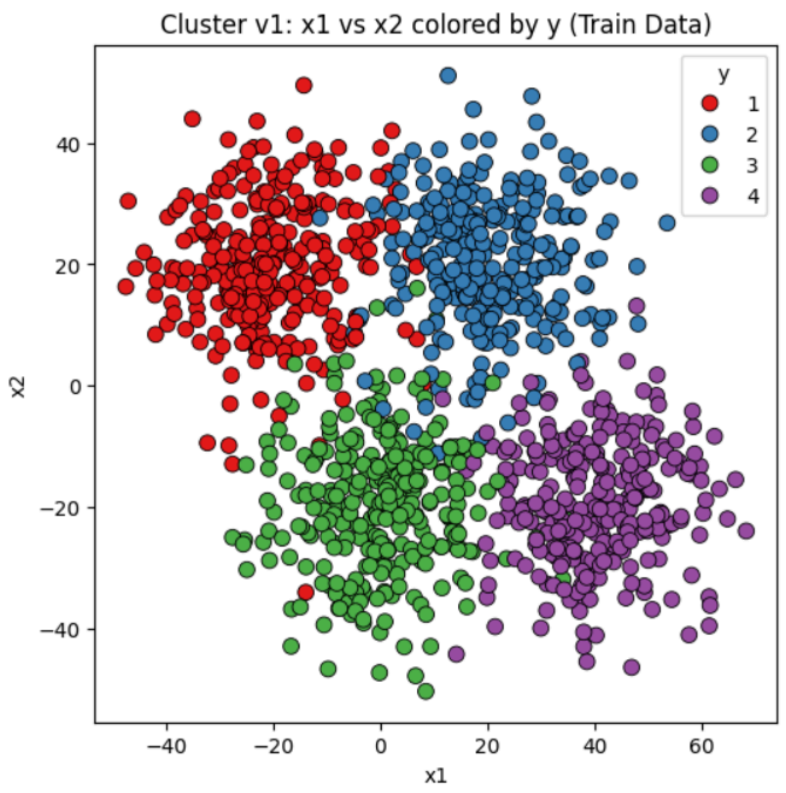




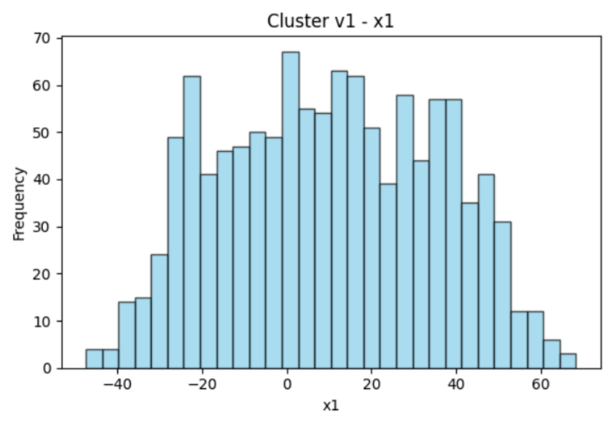
Feature	Mean	Median	Variance	Std	Min	Max
x1	9.845229	9.753953	537.066644	23.174698	-40.704233	55.252202
x2	0.310337	2.304347	454.044429	21.308318	-39.673196	38.471219

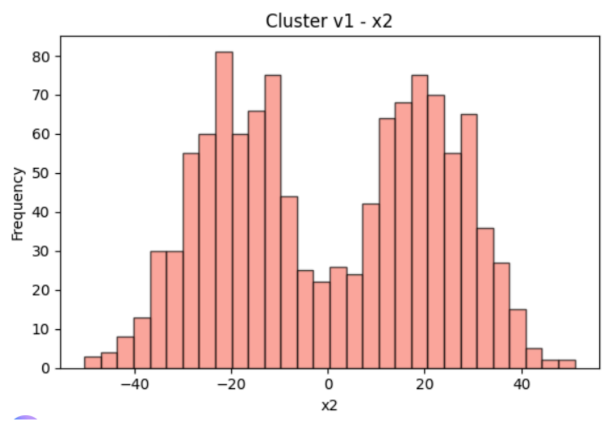
x1: The histogram shows 4 peaks for x1, values are almost distributed between -30 and 50, with highest frequency near -20..

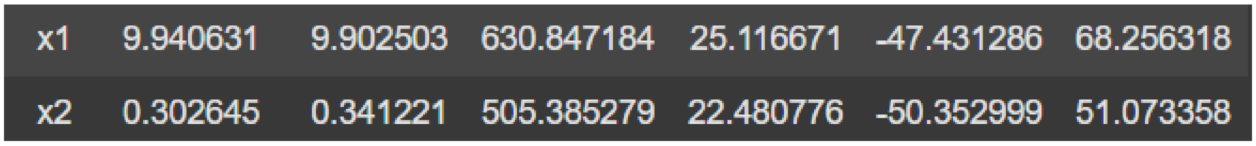
x2: The histogram shows 2 peaks for x2, values are roughly distributed between two regions: between -30 and -10 and between 10 and 30, with the highest frequency near 20..



In the x1 vs x2 plot of cluster v1, although the points of the same class seem to be concentrated in a region, there are much more outliers of each class overlapping with the points of other classes as compared to the plot of cluster v0. However there is still some kind of a border or distinction between any two classes, unlike in the case of cluster v2.



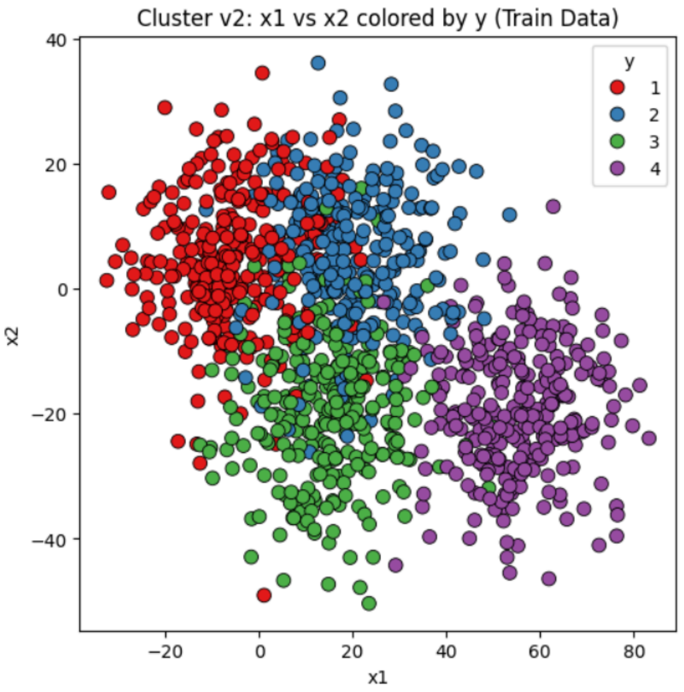




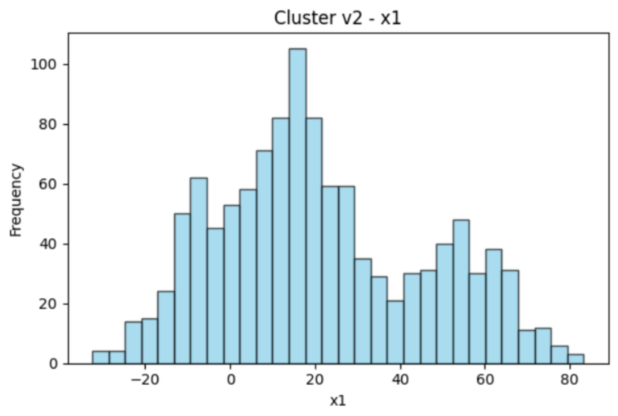
x1	9.940631	9.902503	630.847184	25.116671	-47.431286	68.256318
x2	0.302645	0.341221	505.385279	22.480776	-50.352999	51.073358

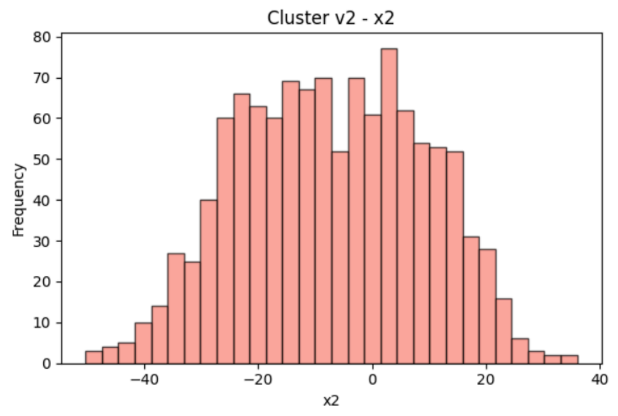
x1: The values of x1 are highly concentrated between -20 and 40 roughly, with the highest frequency near 0..

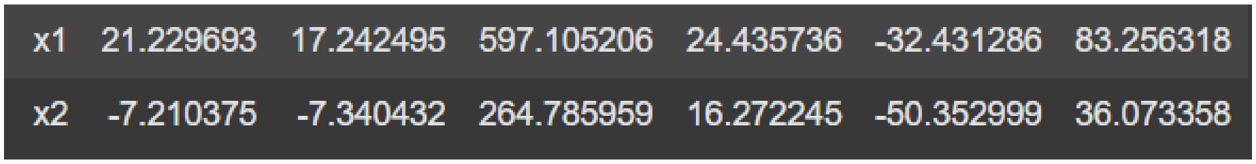
x2: The values of x2 are highly concentrated between -40 and -10 and between 10 and 30, with the highest frequency near 20.



In this plot, the points of the same class are more overlapping as compared to the two other plots. Also there are many more outliers. For example, there are red coloured points near the region of green coloured points and green coloured points in the region of purple coloured points. There are no visible distinctions between any two classes as such.







x1	21.229693	17.242495	597.105206	24.435736	-32.431286	83.256318
x2	-7.210375	-7.340432	264.785959	16.272245	-50.352999	36.073358

x1: The distribution is less concentrated around the mean as compared to x2, with most of the values lying in the range of 0-20, with the highest frequency near 20.

x2 :The distribution is highly concentrated around the mean, with most of the values lying between roughly -30 and 15, with the highest frequency near 0.

There are no missing rows or columns or cells in any of the three datasets and hence no need for any imputation. Also in all of the three datasets, y takes discrete values. So classification is possible. Out of all the three datasets, logistic regression model is most suitable for cluster v0 and least suitable for cluster v2.

3.c. Linear Kernel : A linear kernel is simply the **dot product** of two feature vectors:
It assumes that the data is **linearly separable**, i.e., a straight line (or hyperplane in higher dimensions) can separate the classes.

When to use:

Data is already linearly separable.
Large feature space; you don't need extra complexity.

Pros :

- Simple and fast
- Less prone to overfitting

Cons:

- Cannot capture non linear relationships

RBF kernels

- Also called Gaussian Kernel. Measures similarity based on distance:
- $$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right)$$

  It maps data into a **higher-dimensional space**, allowing a linear separator in that space to correspond to a **nonlinear boundary** in the original space.

When to use:

Data is **not linearly separable**.
You need more flexibility in the decision boundary.

Pros:

Can handle complex, nonlinear relationships.

Cons:

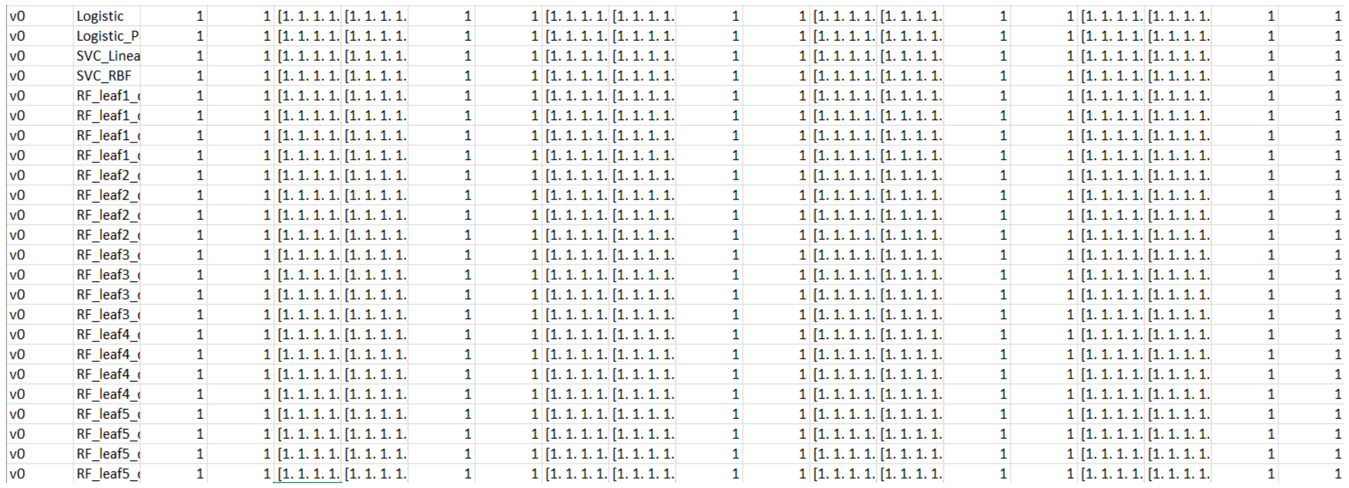
- Slower, especially with large datasets.
- Risk of overfitting if γ is too high.

6. All the plots and ROC curves of the models have been generated in the code uploaded.
General inferences from the metrics:

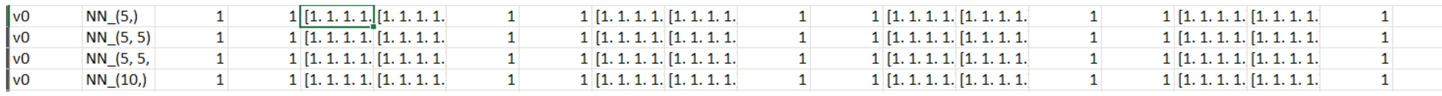
- As the depth of the random forest and the number of parameters in the neural network increases, the overfitting increases. So NN(5,5,5) will be more overfitting than NN(10) which is more overfitting than NN(5).
- In random forest, for the same number of leaves, as the depth increases, the value of the metrics increases for the train data but decreases for the test data which implies overfitting.

In all of the tables below, the columns are in the following order : accuracy_train, accuracy_test, precision_train, precision_test, precision_train_avg, precision_test_avg, recall_train, recall_test, recall_train_avg, recall_test_avg, f1_train, f1_test, f1_train_avg, f1_test_avg, AUC_train, AUC_test, AUC_train_avg, AUC_test_avg.

V0



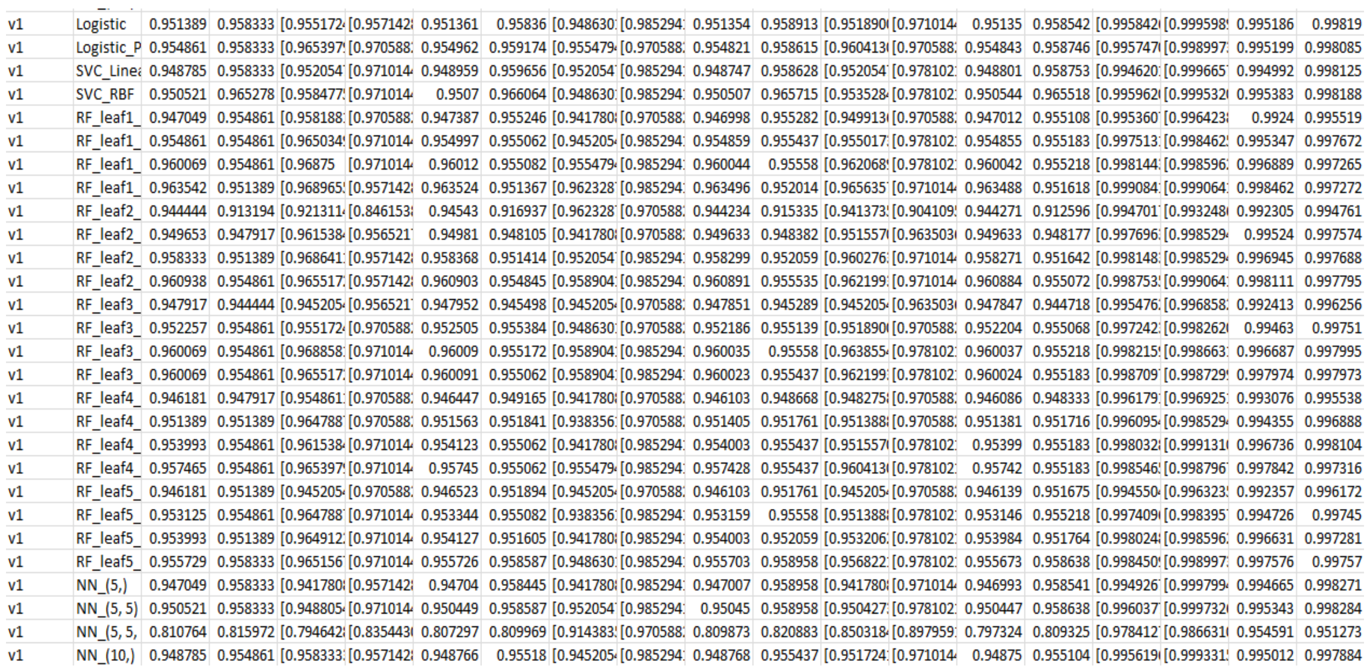
v0	Logistic	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	Logistic_P	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	SVC_Linea	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	SVC_RBF	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf1_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf1_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf1_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf1_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf2_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf2_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf2_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf2_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf3_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf3_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf3_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf3_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf4_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf4_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf4_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf4_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf5_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf5_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf5_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1
v0	RF_leaf5_(	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1



v0	NN_(5,)	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1
v0	NN_(5, 5)	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1
v0	NN_(5, 5,	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1
v0	NN_(10,)	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1	1	[1. 1. 1. 1.	[1. 1. 1. 1.	1

All the models fit perfectly for the dataset v0 because all the metrics of all the models, whether for train set or test set are equal to 1. It is also evident from the initial scatter plot of x1 and x2, all the models could create perfect classification boundaries for cluster v0 dataset.
The ROC curve is also a perfectly horizontal line in the case of v0 dataset.

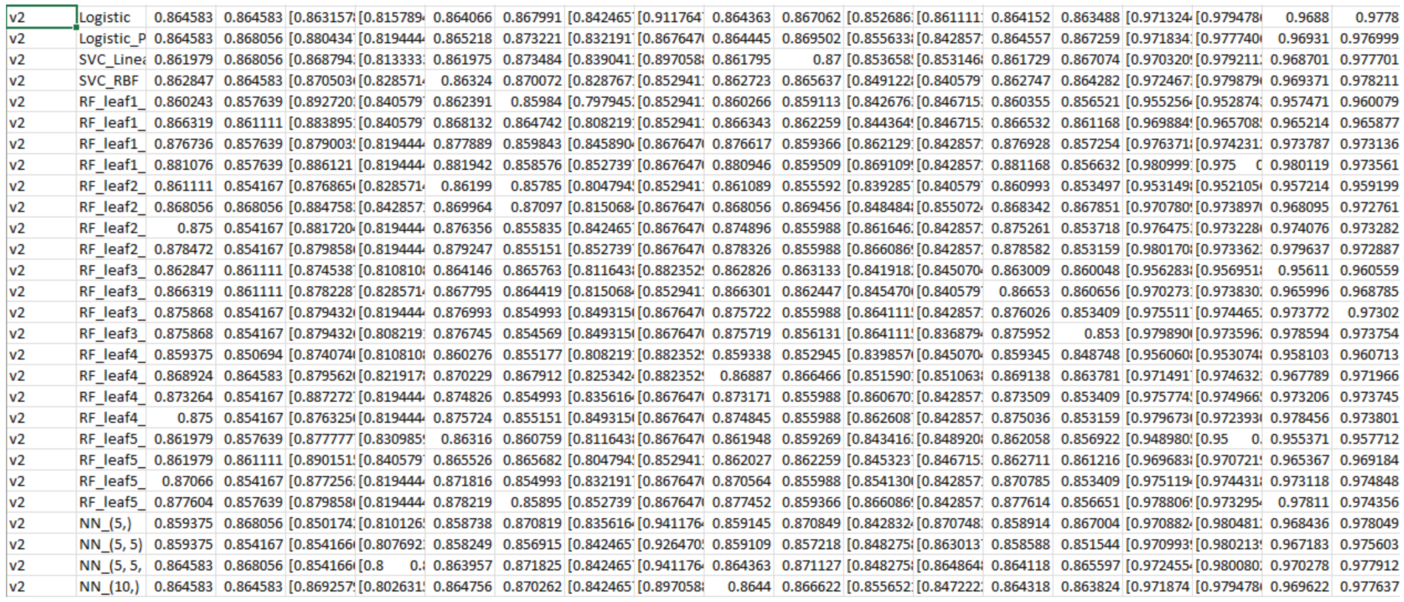
V1



v1	Logistic	0.951389	0.958333	[0.955172	[0.957142	0.951361	0.95836	[0.948630	[0.985294	0.951354	0.958913	[0.951890	[0.971014	0.95135	0.958542	[0.995842	[0.999598	0.995186	0.99819
v1	Logistic_P	0.954861	0.958333	[0.965397	[0.970588	0.954962	0.959174	[0.955479	[0.970588	0.954821	0.958615	[0.960413	[0.970588	0.954843	0.958746	[0.995747	[0.998997	0.995199	0.998085
v1	SVC_Linea	0.948785	0.958333	[0.952054	[0.971014	0.948959	0.959656	[0.952054	[0.985294	0.948747	0.958628	[0.952054	[0.978102	0.948801	0.958753	[0.994620	[0.999665	0.994992	0.998125
v1	SVC_RBF	0.950521	0.965278	[0.958477	[0.971014	0.9507	0.966064	[0.948630	[0.985294	0.950507	0.965715	[0.953528	[0.978102	0.950544	0.965518	[0.995962	[0.999532	0.995383	0.998188
v1	RF_leaf1_	0.947049	0.954861	[0.958188	[0.970588	0.947387	0.955246	[0.941780	[0.970588	0.946998	0.955282	[0.949913	[0.970588	0.947012	0.955108	[0.995360	[0.996423	0.9924	0.995519
v1	RF_leaf1_	0.954861	0.954861	[0.965034	[0.971014	0.954997	0.955062	[0.945205	[0.985294	0.954859	0.955437	[0.955017	[0.978102	0.954855	0.955183	[0.997513	[0.998462	0.995347	0.997672
v1	RF_leaf1_	0.960069	0.954861	[0.96875	[0.971014	0.96012	0.955082	[0.955479	[0.985294	0.960044	0.95558	[0.962068	[0.978102	0.960042	0.955218	[0.998144	[0.998596	0.996889	0.997265
v1	RF_leaf1_	0.963542	0.951389	[0.968965	[0.957142	0.963524	0.951367	[0.962328	[0.985294	0.963496	0.952014	[0.965635	[0.971014	0.963488	0.951618	[0.999084	[0.999064	0.998462	0.997272
v1	RF_leaf2_	0.944444	0.913194	[0.921311	[0.846153	0.94543	0.916937	[0.962328	[0.970588	0.944234	0.915335	[0.941373	[0.904109	0.944271	0.912596	[0.994701	[0.993248	0.992305	0.994761
v1	RF_leaf2_	0.949653	0.947917	[0.961538	[0.956521	0.94981	0.948105	[0.941780	[0.970588	0.949633	0.948382	[0.951557	[0.963503	0.949633	0.948177	[0.997696	[0.998529	0.99524	0.997574
v1	RF_leaf2_	0.958333	0.951389	[0.968641	[0.957142	0.958368	0.951414	[0.952054	[0.985294	0.958299	0.952059	[0.960276	[0.971014	0.958271	0.951642	[0.998148	[0.998529	0.996945	0.997688
v1	RF_leaf2_	0.960938	0.954861	[0.965517	[0.957142	0.960903	0.954845	[0.958904	[0.985294	0.960891	0.955535	[0.962199	[0.971014	0.960884	0.955072	[0.998753	[0.999064	0.998111	0.997795
v1	RF_leaf3_	0.947917	0.944444	[0.945205	[0.956521	0.947952	0.945498	[0.945205	[0.970588	0.947851	0.945289	[0.945205	[0.963503	0.947847	0.944718	[0.995476	[0.996858	0.992413	0.996256
v1	RF_leaf3_	0.952257	0.954861	[0.955172	[0.970588	0.952505	0.955384	[0.948630	[0.970588	0.952186	0.955139	[0.951890	[0.970588	0.952204	0.955068	[0.997242	[0.998262	0.99463	0.99751
v1	RF_leaf3_	0.960069	0.954861	[0.968858	[0.971014	0.96009	0.955172	[0.958904	[0.985294	0.960035	0.95558	[0.963855	[0.978102	0.960037	0.955218	[0.998215	[0.998663	0.996687	0.997995
v1	RF_leaf3_	0.960069	0.954861	[0.965517	[0.971014	0.960091	0.955062	[0.958904	[0.985294	0.960023	0.955437	[0.962199	[0.978102	0.960024	0.955183	[0.998709	[0.998729	0.997974	0.997973
v1	RF_leaf4_	0.946181	0.947917	[0.954861	[0.970588	0.946447	0.949165	[0.941780	[0.970588	0.946103	0.948668	[0.948275	[0.970588	0.946086	0.948333	[0.996179	[0.996925	0.993076	0.995538
v1	RF_leaf4_	0.951389	0.951389	[0.964788	[0.970588	0.951563	0.951841	[0.938356	[0.970588	0.951405	0.951761	[0.951388	[0.970588	0.951381	0.951716	[0.996095	[0.998529	0.994355	0.996888
v1	RF_leaf4_	0.953993	0.954861	[0.961538	[0.971014	0.954123	0.955062	[0.941780	[0.985294	0.954003	0.955437	[0.951557	[0.978102	0.95399	0.955183	[0.998032	[0.999131	0.996736	0.998104
v1	RF_leaf4_	0.957465	0.954861	[0.965397	[0.971014	0.95745	0.955062	[0.955479	[0.985294	0.957428	0.955437	[0.960413	[0.978102	0.95742	0.955183	[0.998546	[0.998796	0.997842	0.997316
v1	RF_leaf5_	0.946181	0.951389	[0.945205	[0.970588	0.946523	0.951894	[0.945205	[0.970588	0.946103	0.951761	[0.945205	[0.970588	0.946139	0.951675	[0.994550	[0.996323	0.992357	0.996172
v1	RF_leaf5_	0.953125	0.954861	[0.964788	[0.971014	0.953344	0.955082	[0.938356	[0.985294	0.953159	0.95558	[0.951388	[0.978102	0.953146	0.955218	[0.997409	[0.998395	0.994726	0.99745
v1	RF_leaf5_	0.953993	0.951389	[0.964912	[0.971014	0.954127	0.951605	[0.941780	[0.985294	0.954003	0.952059	[0.953206	[0.978102	0.953984	0.951764	[0.998024	[0.998596	0.996631	0.997281
v1	RF_leaf5_	0.955729	0.958333	[0.965156	[0.971014	0.955726	0.958587	[0.948630	[0.985294	0.955703	0.958958	[0.956822	[0.978102	0.955673	0.958638	[0.998450	[0.998997	0.997576	0.99757
v1	NN_(5,)	0.947049	0.958333	[0.941780	[0.957142	0.94704	0.958445	[0.941780	[0.985294	0.947007	0.958958	[0.941780	[0.971014	0.946993	0.958541	[0.994926	[0.999799	0.994665	0.998271
v1	NN_(5, 5)	0.950521	0.958333	[0.948805	[0.971014	0.950449	0.958587	[0.952054	[0.985294	0.95045	0.958958	[0.950427	[0.978102	0.950447	0.958638	[0.996037	[0.999732	0.995343	0.998284
v1	NN_(5, 5,	0.810764	0.815972	[0.794642	[0.835443	0.807297	0.809969	[0.914383	[0.970588	0.809873	0.820883	[0.850318	[0.897959	0.797324	0.809325	[0.978412	[0.986631	0.954591	0.951273
v1	NN_(10,)	0.948785	0.954861	[0.958333	[0.957142	0.948766	0.95518	[0.945205	[0.985294	0.948768	0.955437	[0.951724	[0.971014	0.94875	0.955104	[0.995619	[0.999331	0.995012	0.997884

- All the metrics are close to 1 for all the models, out of which AUC_test_avg is highest for Support Vector Classifier implemented using rbf kernel. It also performs better than SVC using linear kernel on both test and train data. This indicates the increase in the non-linearity of the classification boundary.
- Assuming neither false positives nor false negatives are costly, F1 score can be used to evaluate all the models. The neural network with 2 layers of 5 neurons each performs best on the test set while NN(10) performs better on the train set.
- In terms of F1 score, logistic regression with polynomial features performs better on the train data as well as test data than normal logistic regression, which also implies the latter is underfitting.
- Random forest with depth 1 and 5 leaves performs best on the train data in terms of F1 score and SVC with rbf kernel performs best on the test data.

V2



v2	Logistic	0.864583	0.864583	[0.863157	[0.815789	0.864066	0.867991	[0.842465	[0.911764	0.864363	0.867062	[0.852686	[0.861111	0.864152	0.863488	[0.971324	[0.979478	0.9688	0.9778
v2	Logistic_P	0.864583	0.868056	[0.880434	[0.819444	0.865218	0.873221	[0.832191	[0.867647	0.864445	0.869502	[0.855633	[0.842857	0.864557	0.867259	[0.971834	[0.977740	0.96931	0.976999
v2	SVC_Linea	0.861979	0.868056	[0.868794	[0.813333	0.861975	0.873484	[0.839041	[0.897058	0.861795	0.87	[0.853658	[0.853146	0.861729	0.867074	[0.970320	[0.979211	0.968701	0.977701
v2	SVC_RBF	0.862847	0.864583	[0.870503	[0.828571	0.86324	0.870072	[0.828767	[0.852941	0.862723	0.865637	[0.849122	[0.840579	0.862747	0.864282	[0.972467	[0.979879	0.969371	0.978211
v2	RF_leaf1_	0.860243	0.857639	[0.892720	[0.840579	0.862391	0.85984	[0.797945	[0.852941	0.860266	0.859113	[0.842676	[0.846715	0.860355	0.856521	[0.955256	[0.952874	0.957471	0.960079
v2	RF_leaf1_	0.866319	0.861111	[0.883895	[0.840579	0.868132	0.864742	[0.808219	[0.852941	0.866343	0.862259	[0.844364	[0.846715	0.866532	0.861168	[0.969884	[0.965708	0.965214	0.965877
v2	RF_leaf1_	0.876736	0.857639	[0.879003	[0.819444	0.877889	0.859843	[0.845890	[0.867647	0.876617	0.859366	[0.862129	[0.842857	0.876928	0.857254	[0.976371	[0.974231	0.973787	0.973136
v2	RF_leaf1_	0.881076	0.857639	[0.886121	[0.819444	0.881942	0.858576	[0.852739	[0.867647	0.880946	0.859509	[0.869109	[0.842857	0.881168	0.856632	[0.980999	[0.975 0	0.980119	0.973561
v2	RF_leaf2_	0.861111	0.854167	[0.876865	[0.828571	0.86199	0.85785	[0.804794	[0.852941	0.861089	0.855592	[0.839285	[0.840579	0.860993	0.853497	[0.953149	[0.952105	0.957214	0.959199
v2	RF_leaf2_	0.868056	0.868056	[0.884758	[0.842857	0.869964	0.87097	[0.815068	[0.867647	0.868056	0.869456	[0.848484	[0.855072	0.868342	0.867851	[0.970780	[0.973897	0.968095	0.972761
v2	RF_leaf2_	0.875	0.854167	[0.881720	[0.819444	0.876356	0.855835	[0.842465	[0.867647	0.874896	0.855988	[0.861646	[0.842857	0.875261	0.853718	[0.976475	[0.973228	0.974076	0.973282
v2	RF_leaf2_	0.878472	0.854167	[0.879858	[0.819444	0.879247	0.855151	[0.852739	[0.867647	0.878326	0.855988	[0.866086	[0.842857	0.878582	0.853159	[0.980170	[0.973362	0.979637	0.972887
v2	RF_leaf3_	0.862847	0.861111	[0.874538	[0.810810	0.864146	0.865763	[0.811643	[0.882352	0.862826	0.863133	[0.841918	[0.845070	0.863009	0.860048	[0.956283	[0.956951	0.95611	0.960559
v2	RF_leaf3_	0.866319	0.861111	[0.878228	[0.828571	0.867795	0.864419	[0.815068	[0.852941	0.866301	0.862447	[0.845470	[0.840579	0.86653	0.860656	[0.970273	[0.973830	0.965996	0.968785
v2	RF_leaf3_	0.875868	0.854167	[0.879432	[0.819444	0.876993	0.854993	[0.849315	[0.867647	0.875722	0.855988	[0.864111	[0.842857	0.876026	0.853409	[0.975511	[0.974465	0.973772	0.97302
v2	RF_leaf3_	0.875868	0.854167	[0.879432	[0.808219	0.876745	0.854569	[0.849315	[0.867647	0.875719	0.856131	[0.864111	[0.836879	0.875952	0.853	[0.979890	[0.973596	0.978594	0.973754
v2	RF_leaf4_	0.859375	0.850694	[0.874074	[0.810810	0.860276	0.855177	[0.808219	[0.882352	0.859338	0.852945	[0.839857	[0.845070	0.859345	0.848748	[0.956060	[0.953074	0.958103	0.960713
v2	RF_leaf4_	0.868924	0.864583	[0.879562	[0.821917	0.870229	0.867912	[0.825342	[0.882352	0.86887	0.866466	[0.851590	[0.851063	0.869138	0.863781	[0.971491	[0.974632	0.967789	0.971966
v2	RF_leaf4_	0.873264	0.854167	[0.887272	[0.819444	0.874826	0.854993	[0.835616	[0.867647	0.873171	0.855988	[0.860670	[0.842857	0.873509	0.853409	[0.975774	[0.974966	0.973206	0.973745
v2	RF_leaf4_	0.875	0.854167	[0.876325	[0.819444	0.875724	0.855151	[0.849315	[0.867647	0.874845	0.855988	[0.862608	[0.842857	0.875036	0.853159	[0.979673	[0.972393	0.978456	0.973801
v2	RF_leaf5_	0.861979	0.857639	[0.877777	[0.830985	0.86316	0.860759	[0.811643	[0.867647	0.861948	0.859269	[0.843416	[0.848920	0.862058	0.856922	[0.948980	[0.95 0.	0.955371	0.957712
v2	RF_leaf5_	0.861979	0.861111	[0.890151	[0.840579	0.865526	0.865682	[0.804794	[0.852941	0.862027	0.862259	[0.845323	[0.846715	0.862711	0.861216	[0.969683	[0.970721	0.965367	0.969184
v2	RF_leaf5_	0.87066	0.854167	[0.877256	[0.819444	0.871816	0.854993	[0.832191	[0.867647	0.870564	0.855988	[0.854130	[0.842857	0.870785	0.853409	[0.975119	[0.974431	0.973118	0.974848
v2	RF_leaf5_	0.877604	0.857639	[0.879858	[0.819444	0.878219	0.85895	[0.852739	[0.867647	0.877452	0.859366	[0.866086	[0.842857	0.877614	0.856651	[0.978806	[0.973295	0.97811	0.974356
v2	NN_(5,)	0.859375	0.868056	[0.850174	[0.810126	0.858738	0.870819	[0.835616	[0.941176	0.859145	0.870849	[0.842832	[0.870748	0.858914	0.867004	[0.970882	[0.980481	0.968436	0.978049
v2	NN_(5, 5)	0.859375	0.854167	[0.854166	[0.807692	0.858249	0.856915	[0.842465	[0.926470	0.859109	0.857218	[0.848275	[0.863013	0.858588	0.851544	[0.970993	[0.980213	0.967183	0.975603
v2	NN_(5, 5,	0.864583	0.868056	[0.854166	[0.8 0.	0.863957	0.871825	[0.842465	[0.941176	0.864363	0.871127	[0.848275	[0.864864	0.864118	0.865597	[0.972455	[0.980080	0.970278	0.977912
v2	NN_(10,)	0.864583	0.864583	[0.869257	[0.802631	0.864756	0.870262	[0.842465	[0.897058	0.8644	0.866622	[0.855652	[0.847222	0.864318	0.863824	[0.971874	[0.979478	0.969622	0.977637

- The values of almost all the metrics have become less than 0.9, which means that the models are not fitting the data points properly. There is difference in the metrics for the train and test data which indicates overfitting in some of the cases. AUC is still greater than 0.9 for all the models and greatest for NN(5) on test data and RF(leaf1, depth5) on train data.
- It can also be seen that SVC using rbf kernel is better compared to that using linear kernel which also indicates the increasing non linearity in the boundary
- In v2, the neural network with 3 layers of 5 neurons each performs best on the test data . Both 2-layer and 3-layer NN perform similarly on the train data.
- In v2 as well, logistic regression with polynomial features fits both train and test data better than normal logistic regression, again underfitting.
- In terms of F1 score, RF(leaf1, depth2) performed best on the train set and RF(leaf2, depth3) performed best on the test set.

Conclusion: Different datasets require different models, and the choice of models depends on various factors such as the size of the dataset, the distribution of classes, and the model's performance on key evaluation metrics including accuracy, precision, recall, F1-score, and AUC.

ROC curve analysis of different models on v1 and v2 datasets:

It is evident from the ROC curves that all the models perform better on v1 dataset than on v2 dataset, as the curves are closer to the y-axis when the models are applied on the former dataset.

7. The metrics used to evaluate k means clustering are :

Silhouette Score

Measures how similar each point is to its own cluster compared to other clusters.

Its value ranges from -1 to 1.
+1: Perfectly clustered, far from other clusters
0: On the border between two clusters
-1: Likely assigned to the wrong cluster
High silhouette → k-means clusters are tight and well separated.

Silhouette for a single point:
Cohesion : Average distance from i to all other points in its own cluster.

$$a_i = \frac{1}{|C_i| - 1} \sum_{j \in C_i, j \neq i} d(i, j)$$

Separation: lowest average distance from iii to all points in any other cluster:

$$b_i = \min_{k \neq C_i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j)$$

Silhouette for point i is given by the formula:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

Silhouette score is the average of the silhouettes of all the points.

Davies-Bouldin Index

- Measures average similarity between clusters.
- Ranges from 0 to ∞ (lower is better)
- Lower DBI → clusters are compact and well separated
- Higher DBI → clusters may overlap or be elongated
- Since k-means minimizes within-cluster variance, a good k-means solution should naturally have a low DBI.

Calculation:
1) First the cluster scatter is computed:

$$S_i = \frac{1}{|C_i|} \sum_{x \in C_i} \|x - \mu_i\|$$

2) Compute pairwise cluster separation (M_i□**)** – distance between cluster centroids i and j:

$$M_{ij} = \|\mu_i - \mu_j\|$$

3) Compute the cluster similarity R_i□:

$$R_{ij} = \frac{S_i + S_j}{M_{ij}}$$

   Combines within-cluster scatter and distance between clusters

4) Find the worst-case similarity for each cluster i:

$$R_i = \max_{j \neq i} R_{ij}$$

5) Average over all clusters → Davies-Bouldin Index:

Calinski-Harabasz Index

- Ratio of between-cluster variance to within-cluster variance.
- Range: 0 to ∞ (higher is better)
- High CH → clusters are dense and well separated
- Low CH → clusters are loose or overlapping

Calculation

1) Compute within-cluster dispersion (Wk)

$$W_k = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

   - Measures **how far points are from their cluster centroid**
   - Lower Wk → points are tightly packed

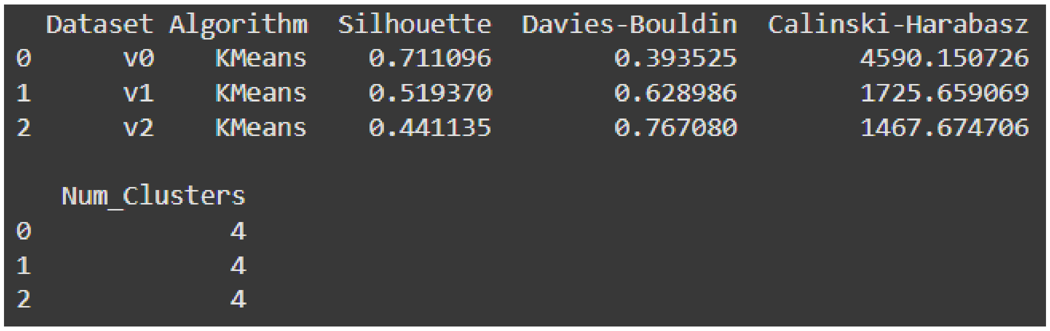
2) Compute between-cluster dispersion (Bk)

$$B_k = \sum_{i=1}^{k} n_i \|\mu_i - \mu\|^2$$

   - Measures how far each cluster centroid is from the overall mean
   - Higher Bk → clusters are well separated

$$CH = \frac{\mathrm{trace}(B_k)/(k-1)}{\mathrm{trace}(W_k)/(n-k)}$$

trace(Bk) and trace(Wk) are sums of squared distances

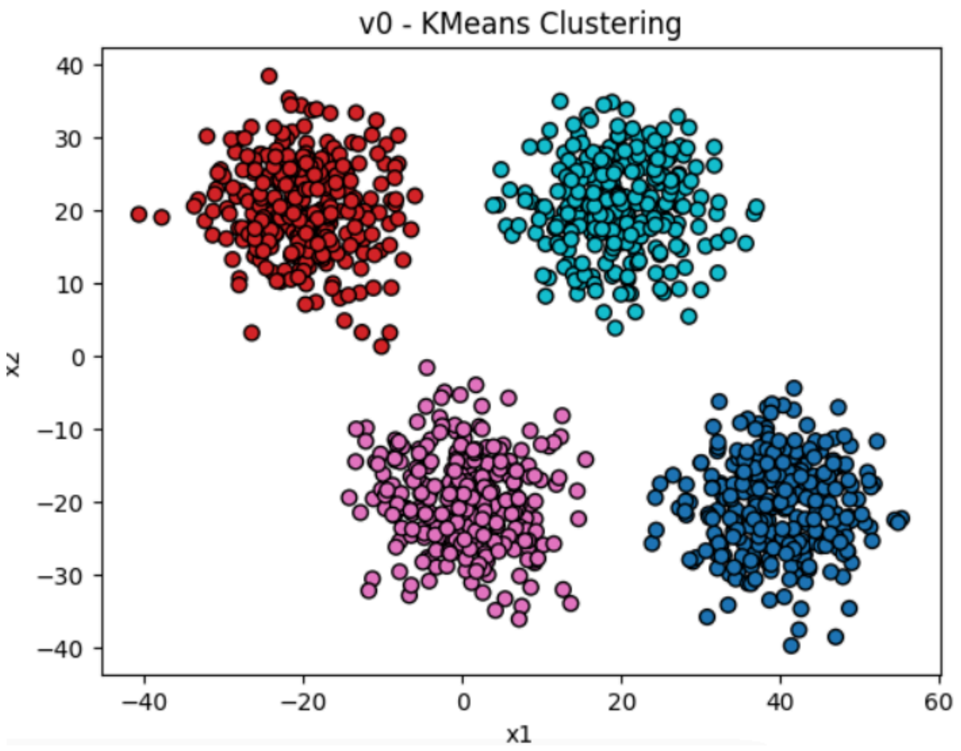


```
  Dataset Algorithm  Silhouette  Davies-Bouldin  Calinski-Harabasz
0      v0    KMeans    0.711096        0.393525        4590.150726
1      v1    KMeans    0.519370        0.628986        1725.659069
2      v2    KMeans    0.441135        0.767080        1467.674706

   Num_Clusters
0             4
1             4
2             4
```

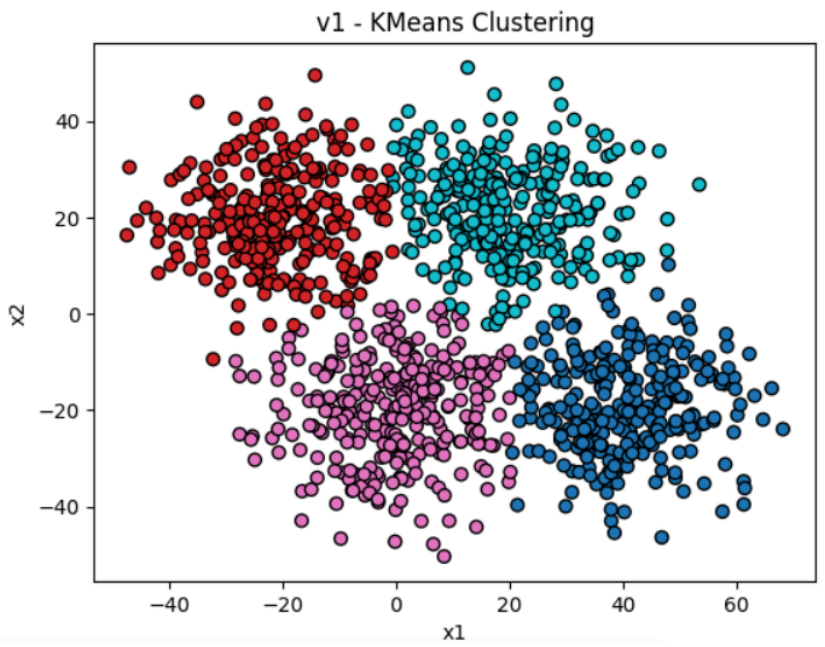
After trying out different values of k for each of the datasets, it was found that the optimum choice of k for all the datasets was 4. In each of the datasets, four clusters were formed using k-means clustering. It is obvious from the metrics that the clustering has been best implemented for v0 dataset, since it has the highest Silhouette score and Calinski-Harabasz index as well as the lowest Davies-Bouldin index. The second best clustering has been implemented in the v1 dataset. All these metrics match with our initial inferences on the datasets.

K-means clustering plots:
V0 dataset:

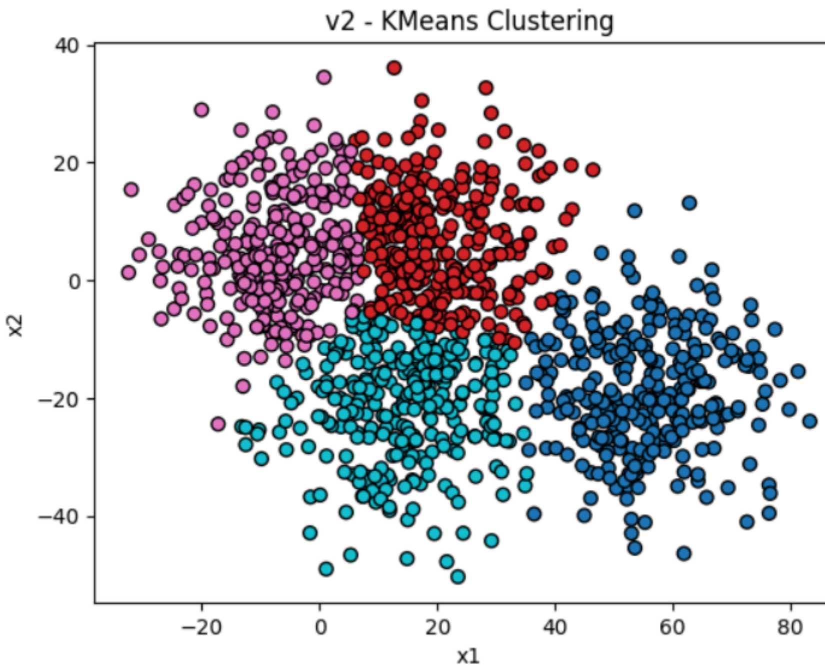


V1-dataset

E4 DS 203 - 24B2224

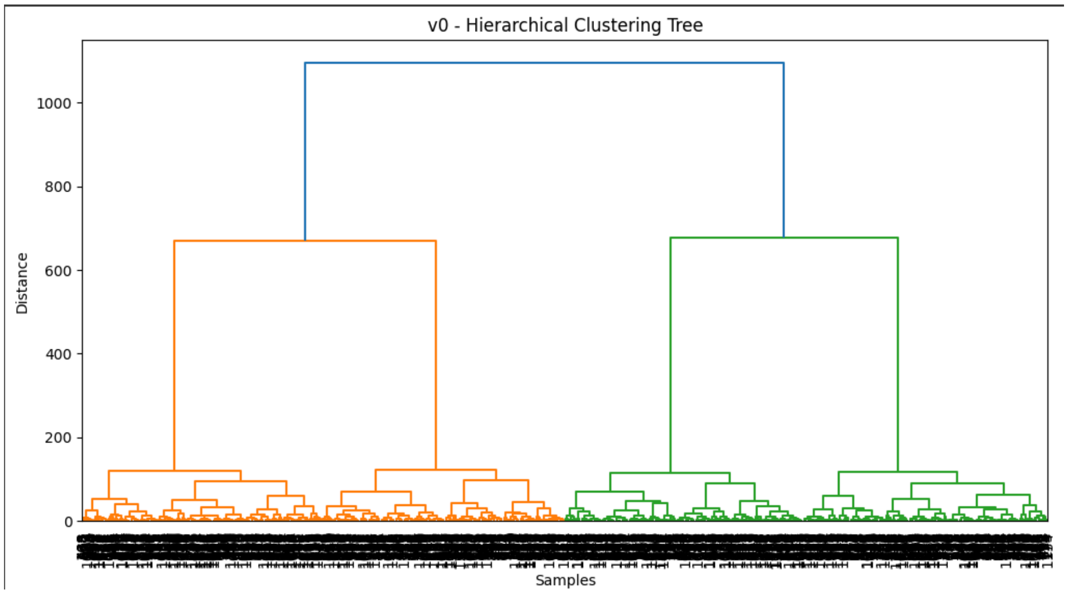


V2-dataset

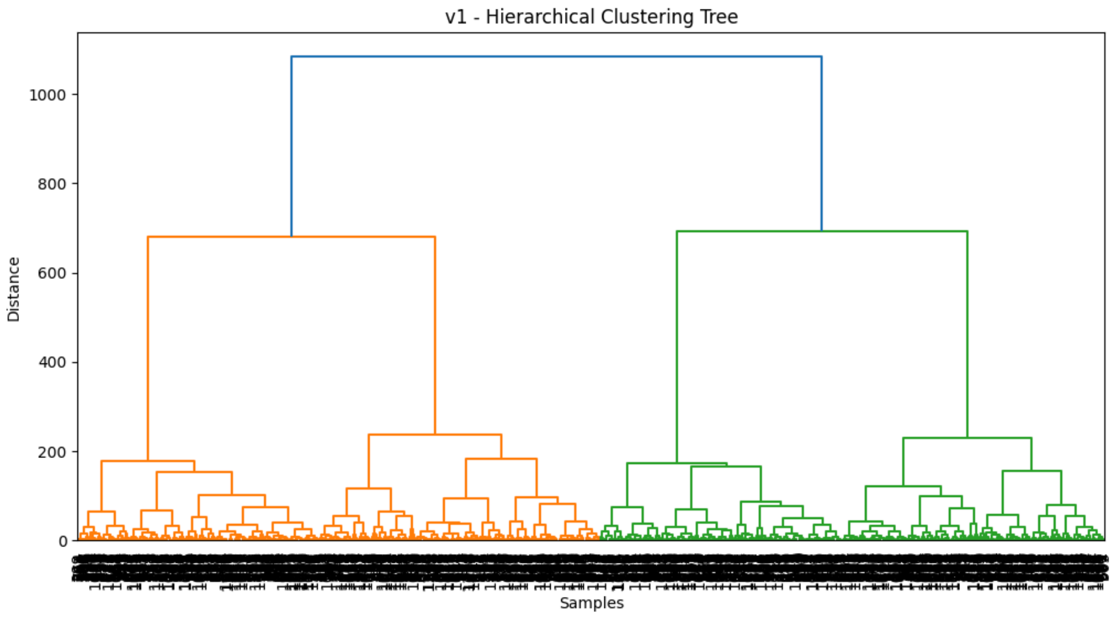


Hierarchical clustering
v0-dataset

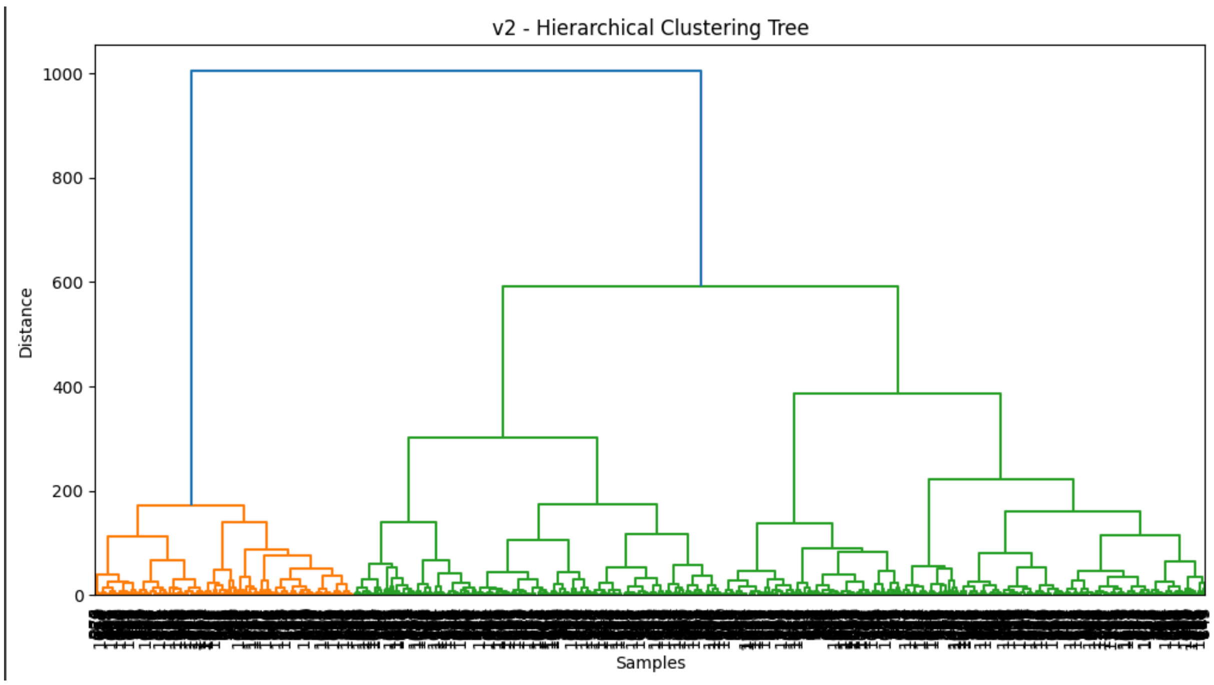
E4 DS 203 - 24B2224

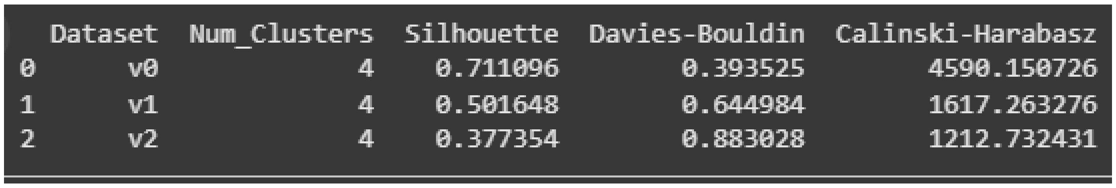


V1-dataset



V2-dataset





```
  Dataset  Num_Clusters  Silhouette  Davies-Bouldin  Calinski-Harabasz
0      v0             4    0.711096        0.393525        4590.150726
1      v1             4    0.501648        0.644984        1617.263276
2      v2             4    0.377354        0.883028        1212.732431
```

The values of the metrics for hierarchical clustering are similar to those for k means clustering. Here also clustering has been best implemented for v0 dataset, followed by v1 and then v2.

Main learnings:
Through this exercise, I learnt:

- how to perform exploratory data analysis
- how model performance varies across datasets
- that the same model cannot be used for all the datasets.
- how to choose a model for a dataset given the values of the evaluation metrics.
- To implement different machine learning models in jupyter notebook