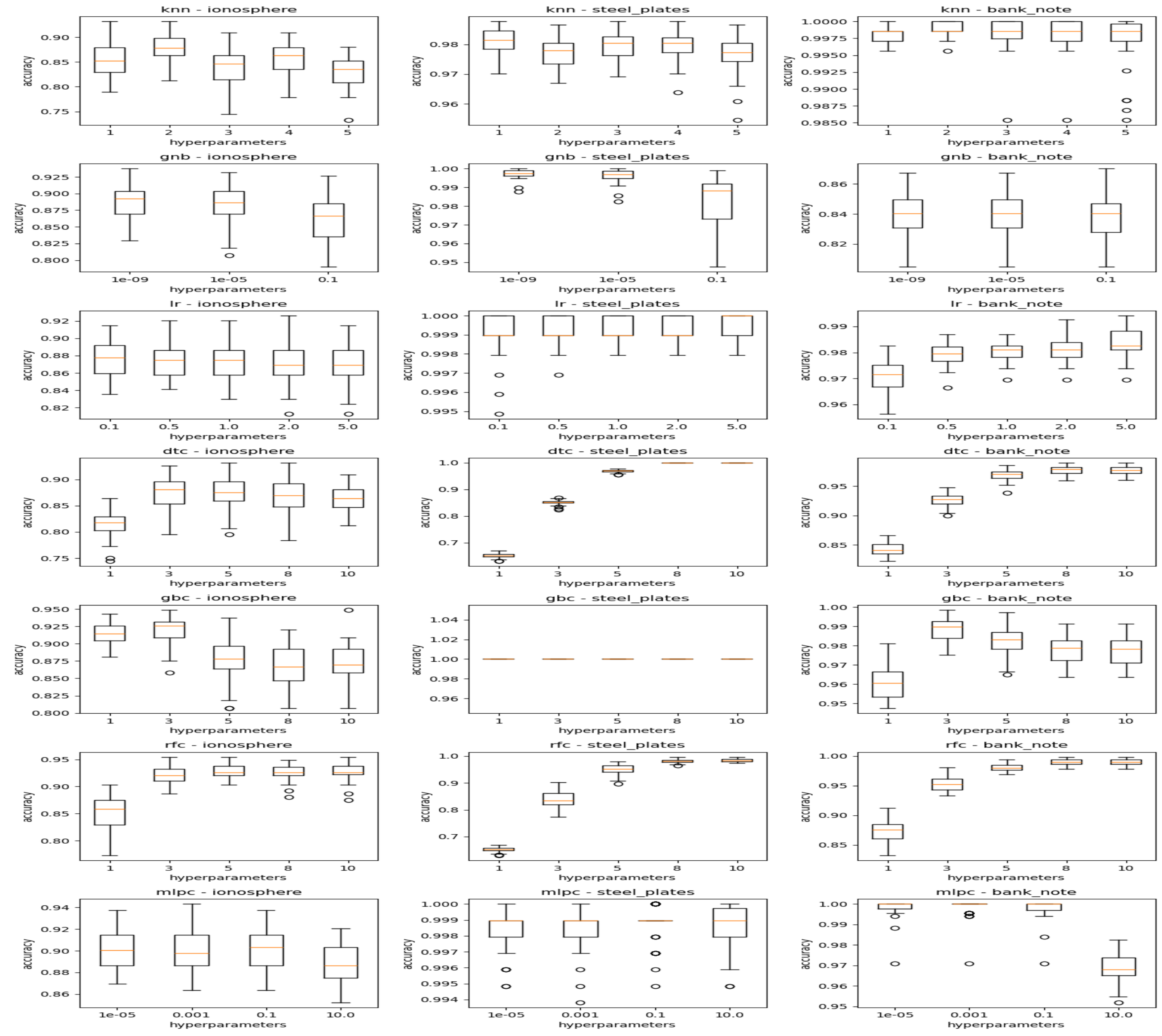
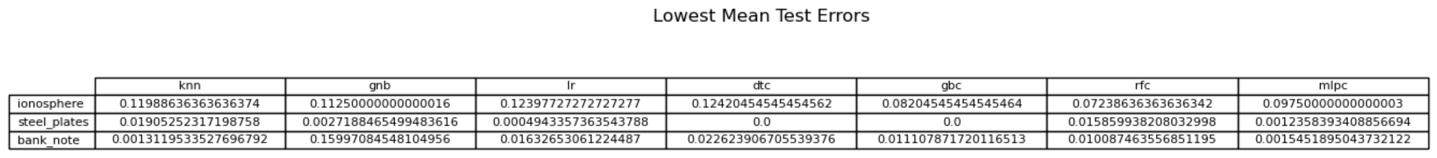
COMP309 Assignment One - Report
greenthom - 300536064

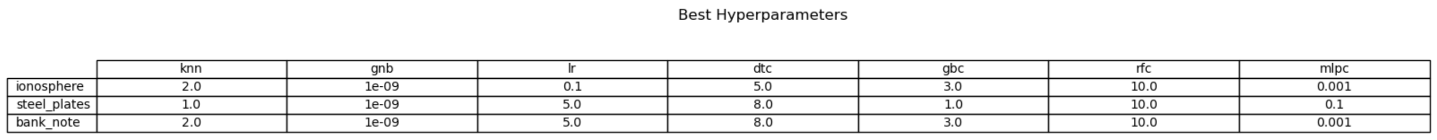
Part One: Classification
Task Two:
Build a 7-by-3 (7 classifiers and 3 datasets) table to present the boxplots on the classifier accuracy versus parameter values. It probably won't fit into one summary figure. You should structure it as separate main plots, one per classifier, each consisting of several subplots for different datasets. You need to consider the parameter and the corresponding values for each classifier as shown in Table i. Other hyperparameters should be left as default.



Task Three:
Present two summary tables, with rows being classifiers, and columns being datasets. Table (1) is to contain the lowest mean value of the test errors obtained from each classifier with various hyperparameter settings. Table (2) is to contain the corresponding hyperparameter values for obtaining the best test errors.


Lowest Mean Test Errors

	knn	gnb	lr	dtc	gbc	rfc	mlpc
ionosphere	0.11988636363636374	0.11250000000000016	0.12397727272727277	0.12420454545454562	0.08204545454545464	0.07238636363636342	0.09750000000000003
steel_plates	0.01905252317198758	0.0027188465499483616	0.0004943357363543788	0.0	0.0	0.015859938208032998	0.0012358393408856694
bank_note	0.0013119533527696792	0.15997084548104956	0.01632653061224487	0.022623906705539376	0.011107871720116513	0.010087463556851195	0.0015451895043732122


Best Hyperparameters

	knn	gnb	lr	dtc	gbc	rfc	mlpc
ionosphere	2.0	1e-09	0.1	5.0	3.0	10.0	0.001
steel_plates	1.0	1e-09	5.0	8.0	1.0	10.0	0.1
bank_note	2.0	1e-09	5.0	8.0	3.0	10.0	0.001

Task Four:
Write a paragraph to compare and analyze the overall results as captured in these two tables including which model has the best performance and why, and how sensitive these models are to the complexity control hyperparameter.

- The model with the best performance is Gradient Boosting Classifier with all datasets performing a very low mean test error. Ionosphere's best hyperparameter for gbc was 3, steel_plates best hyperparameter for gbc was 1 and bank_notes best hyperparameter for gbc was 3. It has perfect accuracy also on the lowest mean test error on the steel_plates dataset indicated by 0.0. Based on a low best hyperparameter this means that the model is not overfitting by restricting the complexity of the individual trees. This also means that the model is performing well on unseen data at these low hyperparameters showing that the model is very effective against unseen data.
- Most of the models are relatively insensitive to the hyperparameter apart from random forest and decision tree which work very similar, and this can see by the changes made at lower hyperparameter values. They both become less sensitive to the hyperparameter changes when at higher hyperparameter values as indicated in the table above. Once these two models reach a certain hyperparameter this makes them unable to make a 'better' or more decision nodes which makes them unable to further improve.
- For bank-note based on the Lowest Mean test errors table we can see it achieved a lowest mean error of 0.0013119533527696792 for algorithm KNN. We can then determine bank_notes best value of KNN is 2. This indicates that KNN with a low hyperparameter of 2 showing that there is minimal chance of overfitting, and the model fitted the training data closely which potentially led to high variance. Due to the hyperparameter being even also this might've led to ambiguity in class voting where if k is an even number there can be a tie in the nearest neighbors belonging to each class. This means that the accuracy being reflected might not be entirely accurate for the dataset.
- For Steel-plates based on the Lowest Mean Test Errors table we can see it achieved a lowest mean error of 0.0 twice for algorithms decision tree classifier and gradient boosting classifier. This indicates possible overfitting/underfitting of the data showing that the dataset is sensitive. We can use this information to identify that a hyperparameter of 8 for decision tree classifier and 1 for gradient bosting classifier gets the best value for steel_plates. I would argue that a hyperparameter of 1 for gradient boosting classifier is not very accurate of the entire dataset as a lower hyperparameter indicates only fitting a single decision trees to analyze the data. The hyperparameter of 8 for decision tree classifier shows that the tree was 'deep' indicating a good analysis of the patterns in the data even if it is prone of overfitting.
- For ionosphere based on the lowest mean test errors table we can see that it achieved a lowest mean error of 0.07238636363636342 for algorithm random forest classifier. We can then determine ionosphere best value of random forest classifier is 10. This indicates that random

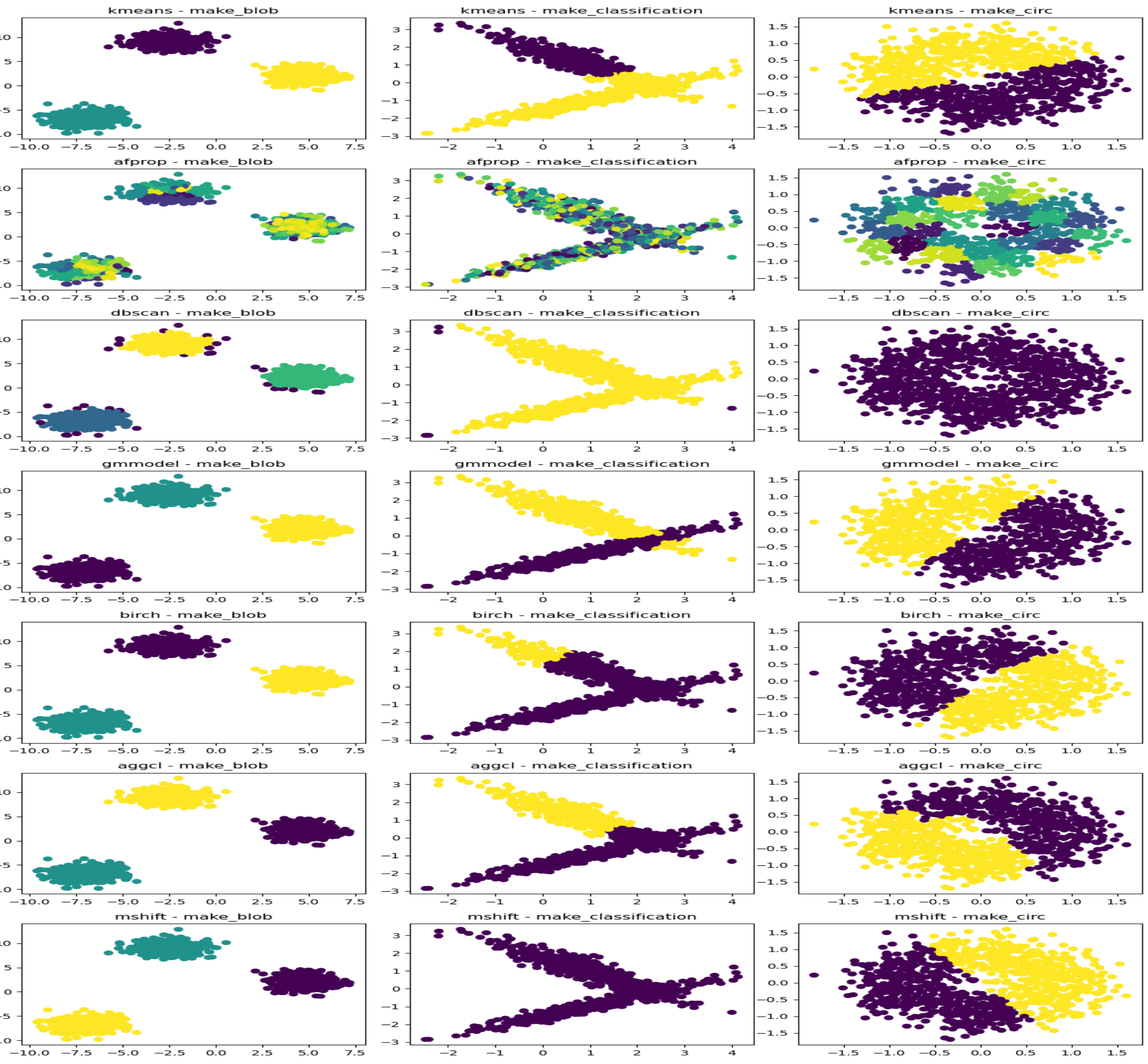
forest classifier with a higher hyperparameter of 10 found patterns in the ionosphere dataset the best, however there is a chance of overfitting.

Part Two: Clustering

Task Two:

Fit the clustering models on the datasets and predict a cluster for each example in the datasets. For clustering models that require specifying the number of clusters, use the insights obtained from the three data-generating functions. Build a 7-by-3 (7 clustering algorithms and 3 datasets) table to present the scatter plots showing the clusters generated by each algorithm.

- Page below.



Task Three (Observations):
Write a paragraph to compare and analyze the results of the clustering algorithms on the three datasets, highlight the characteristics of these algorithms.

- K-Means: is a clustering algorithm that partitions instances into K clusters, where each cluster is defined by a centroid, which is the average position of all its members. In my implementation K-Means is applied to each of the three datasets with a specified number of clusters (n_clusters = 3,2). The algorithm starts with randomly placed centroids and iterates by assigning each data point to the nearest centroid where it then recalculates the centroids on the assigned points until they converge. After training each data point it is assigned to the cluster with the nearest centroid, and the algorithm then looks for the minimized sum of squared distances between points and their centroids, optimizing the clustering outcome. We can see that it identified the 3 clusters (n_clusters=3) effectively for blob dataset and managed to identify the 2 main clusters effectively for make_class. This dataset however does not work well for circles unless specified the n_clusters to 1. This would have to be explicitly stated as the default is 8. K-Means is efficient for large datasets with clear separated clusters however it struggles in overlapping clusters as seen in make_circles.
- Affinity Propagation: clustering algorithm that evaluates the attractiveness of different data points to form clusters. For each of the three datasets I use the algorithm to evaluate the similarity between points and exchanges messages to determine which points are attractive as potential cluster centers (exemplars). Points send messages to others indicating their attractiveness, and each target point responds to inform the senders of its availability to form a cluster. This iterative process continues until a set of cluster centers is identified. After fitting the model to each dataset, it then predicts the cluster assignment for each data point based on these cluster centers. As we can see the algorithm has identified multiple cluster centers within the three main clusters for make_blob. However, for make_class dataset, the algorithm struggles to find/identify a larger cluster center in the dataset. For make_circles it identifies multiple cluster centers within the main cluster similar to make_blobs. Can assume that this algorithm works well for more closer data that isn't spread across a range of values and that it performs better if there are multiple points in one area. While affinity propagation determines the number of clusters, this is more or less determined to how well it interprets a dataset and does not do well with new unseen data.
- DBSCAN: clustering algorithm that identifies clusters based on dense regions of points. I left the eps (defines the circle around each point) and minPts(specifies the minimum number of points that must be within a circle for it to be a cluster) as default. For each dataset, the algorithm groups points into clusters based on their density, where points in a dense region are grouped together, and isolated points or those in less dense regions are classified as noise. After fitting the model to each dataset, I then predict (fit_predict) to assign a cluster label to each datapoint to mark some points as outliers/noise if they do not meet the density/outside of the default eps. Here we can see for make_blobs and make_circles that the algorithm successfully identified the main clusters in the dataset and explicitly identified the outliers to each of the main clusters (due to being outside the default eps). For make_circles it did not identify any outliers or noise however as it interpreted all the data as one main cluster, so it was all determined as part of the eps. This shows that DBSCAN works well with circular clusters, and if the clusters are all centralized in one area. The algorithm is good with finding clusters however its performance does depend on its parameters (e.g. eps) to be set to conform to the dataset it is analyzing.
- Gaussian Mixture Model: is a principled statistical method for clustering. Used it to identify clusters within the three datasets by modelling the data as mixtures of multiple Gaussian distributions. Used n_components (3,2) for the datasets, assuming that the data in each case

is generated from either three or two Gaussian distributions. After fitting the model, it learns the parameters (e.g. means, weights) of these distributions, where the model then assigns each data point to one of the identified Gaussian clusters to capture the structure of the data as a combination of Gaussian clusters. Here we can see for all three datasets, make_blob(3), make_class(2), make_circ(2), it identified all main clusters successfully as a combination of Gaussian Clusters. The algorithm is great with datasets with gaussian clusters however it can struggle with non-gaussian clusters and requires it to be specified with the number of components before-hand as the default number of components (clusters) is 1.

- BIRCH: is a clustering algorithm that doesn't scale well to large datasets and operates as a hierarchical clustering method. Used BIRCH to cluster the three datasets by building a data structure, CF tree. This tree summarizes dense regions of clusters by creating clustering features that represent these regions, adding them to the tree to ensure it remains balanced and efficient. For each dataset, I specify n_clusters (3,2) to indicate that the model is to identify this number of clusters. After fitting the model, the algorithm assigns each data point to one of the specified clusters based on the tree structure. This allows for efficient clustering by summarizing dense regions. Based on my output, we can see that make_blobs worked successfully, identifying the main three clusters in that dataset. In make_class, it worked well for identifying the cluster, however there is a spread of one cluster moving to the other dataset (based on comparison to the gmm model), however the clusters do overlap so it may have identified one cluster before the other. Make_circles were successful in identifying the '2' main clusters in the data. We can assume that BIRCH's approach makes it scalable for a large dataset and good at handling noise, however if there are unclear splits in the dataset or the dataset is performing best on a dataset at a low hyperparameter, this can result in not accurately partitioning clusters and if there are no clear boundaries.
- Agglomerative Clustering: is a hierarchical clustering algorithm that forms clusters by recursively merging the most similar items. Used the algorithm to identify clusters in the three datasets, where it starts by treating each data point as its own cluster and then iteratively merges the closest cluster based on their similarity until the specified n_cluster(3,2), where it'll need to find 2-3 clusters in that specific dataset. After fitting the model (fit_predict), the algorithm will then assign each data point to one of the clusters formed based on the hierarchical merging process that continues until the desired number of clusters I specified has been achieved. Based on the output we can see for make_blobs, make_class and make_circ, it identified the correct number of clusters that had been specified. We can assume that agglomerative clustering can handle various cluster shapes due to its ability to create clear boundaries in a dataset where separations are needed. This algorithm does have its drawbacks as these separations may lead to less natural splits in an uneven dataset distribution.
- Mean Shift: is a clustering algorithm that incrementally moves each data point towards the centroid of points in its local neighborhood. The algorithm clusters the data from the three datasets, where it then iterates by shifting each data point towards the mean of the points within its neighborhood, which is defined by the bandwidth parameter (set as default). For each dataset the mean shift algorithm does not require predefining the number of clusters and can find the clusters based on the density of points. After fitting the model, it assigns each data point to a cluster based on the density shifts, grouping the points that converged around the max density. Based on the output, we can see that make_blobs(3) and make_circles(2) worked successfully without specifying the number of clusters and identified the main clusters in each dataset. However, make_class only identified one main cluster, that is probably due to the points being more spread than the other two datasets, as well as the clusters crossing over, which may of lead to the algorithm interpreting the two main clusters as one whole one.

This is okay just based on the cross-over but a further look into whether it can identify the clusters if they do not cross-over but still spread and near each other could be looked into. My assumption is if the data clusters were separate, it would identify it two clusters correctly therefore it is identifying one mean cluster is correct. This however shows that mean shift is accurate at identifying dense regions and interprets split clusters based on density changes. It however is inefficient at determining a boundary or an accurate split and closely clustered data to identify a classification in the pattern.

REFERENCES

- ChatGPT (For debugging on graph outputs and table outputs for part 1 and 2. Also help debugging on classification algorithms)
- https://scikit-learn.org/stable/ (for understanding and help coding all classification and clustering methods)
- https://wiki.python.org/moin/ForLoop (for understanding python for loops and help debugging and creating code for outputs)
- https://pandas.pydata.org/docs/ (for understanding pandas python documentation for coding methods)