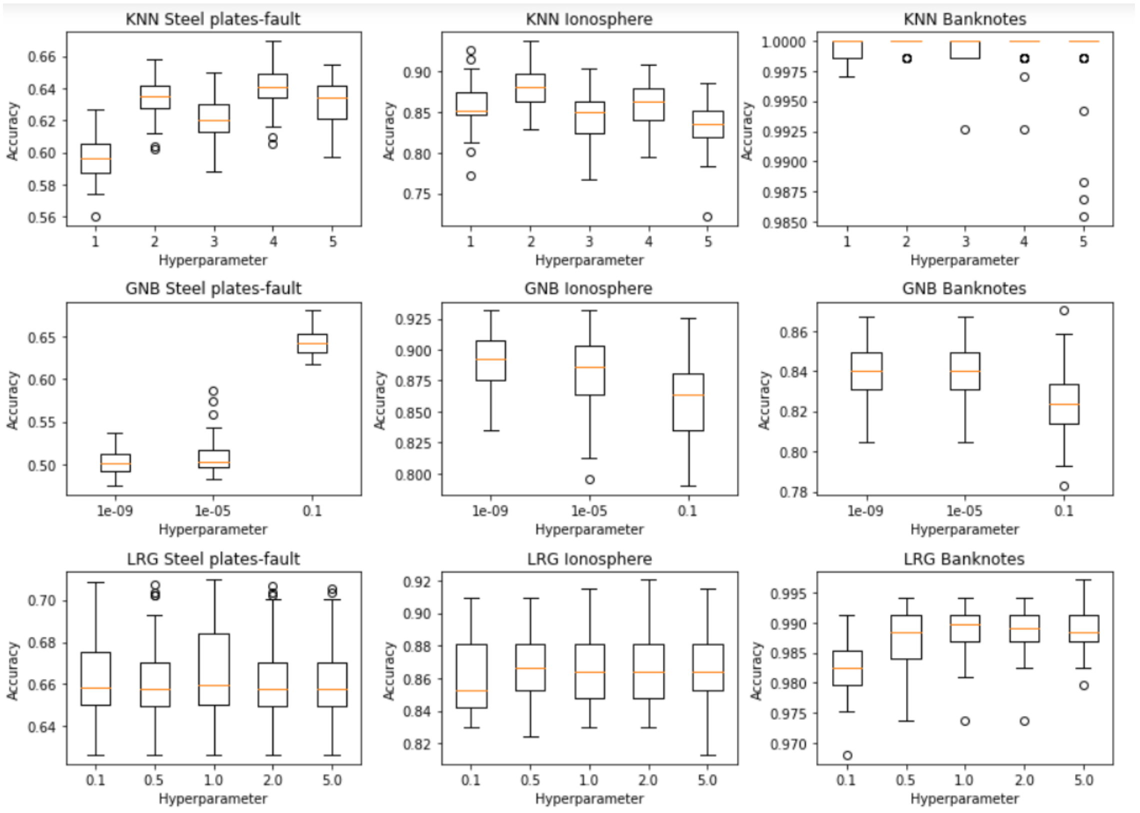
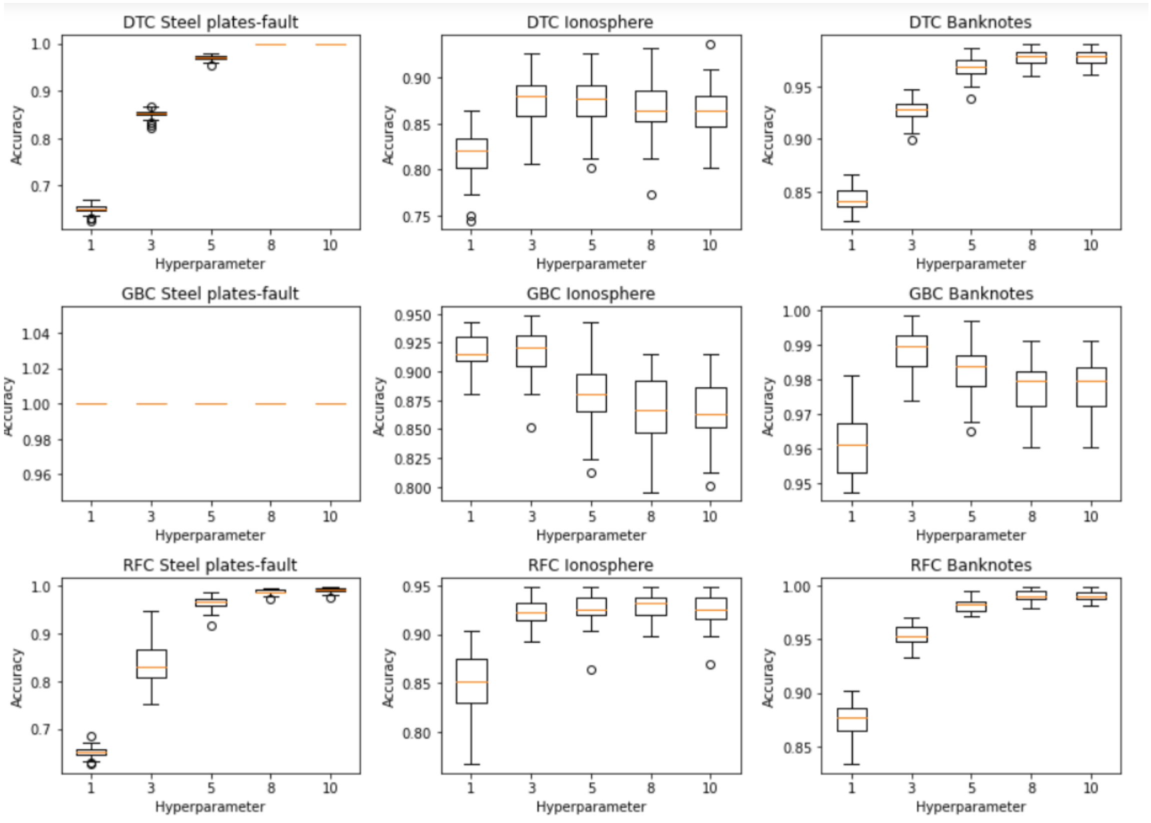

KNN Steel plates-fault
KNN Ionosphere
KNN Banknotes
GNB Steel plates-fault
GNB Ionosphere
GNB Banknotes
LRG Steel plates-fault
LRG Ionosphere
LRG Banknotes
Accuracy
Hyperparameter
1e-09
1e-05
0.1


DTC Steel plates-fault
DTC Ionosphere
DTC Banknotes
GBC Steel plates-fault
GBC Ionosphere
GBC Banknotes
RFC Steel plates-fault
RFC Ionosphere
RFC Banknotes
Accuracy
Hyperparameter

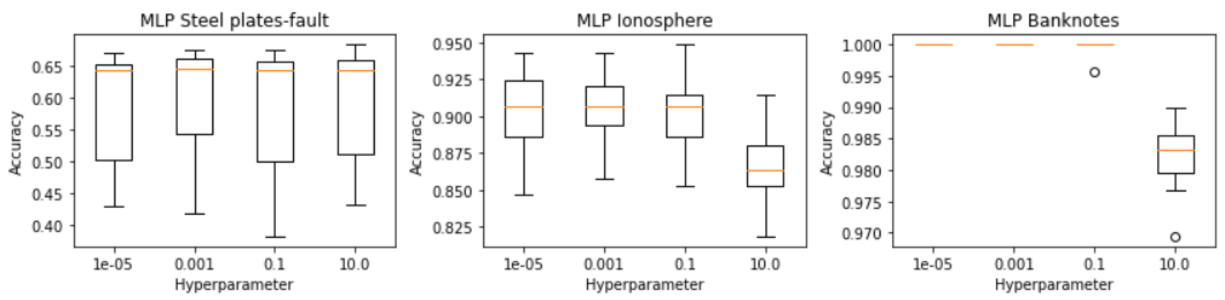


Table 1 Lowests Test Error (1 - Highest Mean Accuracy)

```
+------+---------------------+-------------+-------------+
|      |   Steel Plates Fault |  Ionosphere |   Banknotes |
+======+=====================+=============+=============+
| KNN  |           0.360124  |   0.118864  | 0.000174927 |
+------+---------------------+-------------+-------------+
| GNB  |           0.355819  |   0.111023  | 0.160029    |
+------+---------------------+-------------+-------------+
| LOGR |           0.335221  |   0.133295  | 0.0109038   |
+------+---------------------+-------------+-------------+
| DTC  |           0         |   0.125227  | 0.0219242   |
+------+---------------------+-------------+-------------+
| GBC  |           0         |   0.0818182 | 0.0109621   |
+------+---------------------+-------------+-------------+
| RFC  |           0.00951596 |  0.0725    | 0.0100875   |
+------+---------------------+-------------+-------------+
| MLPC |           0.394995  |   0.0954545 | 0           |
+------+---------------------+-------------+-------------+
```

Table 2 Best Hyperparameter (Parameter associated with Lowest TE in table 1)

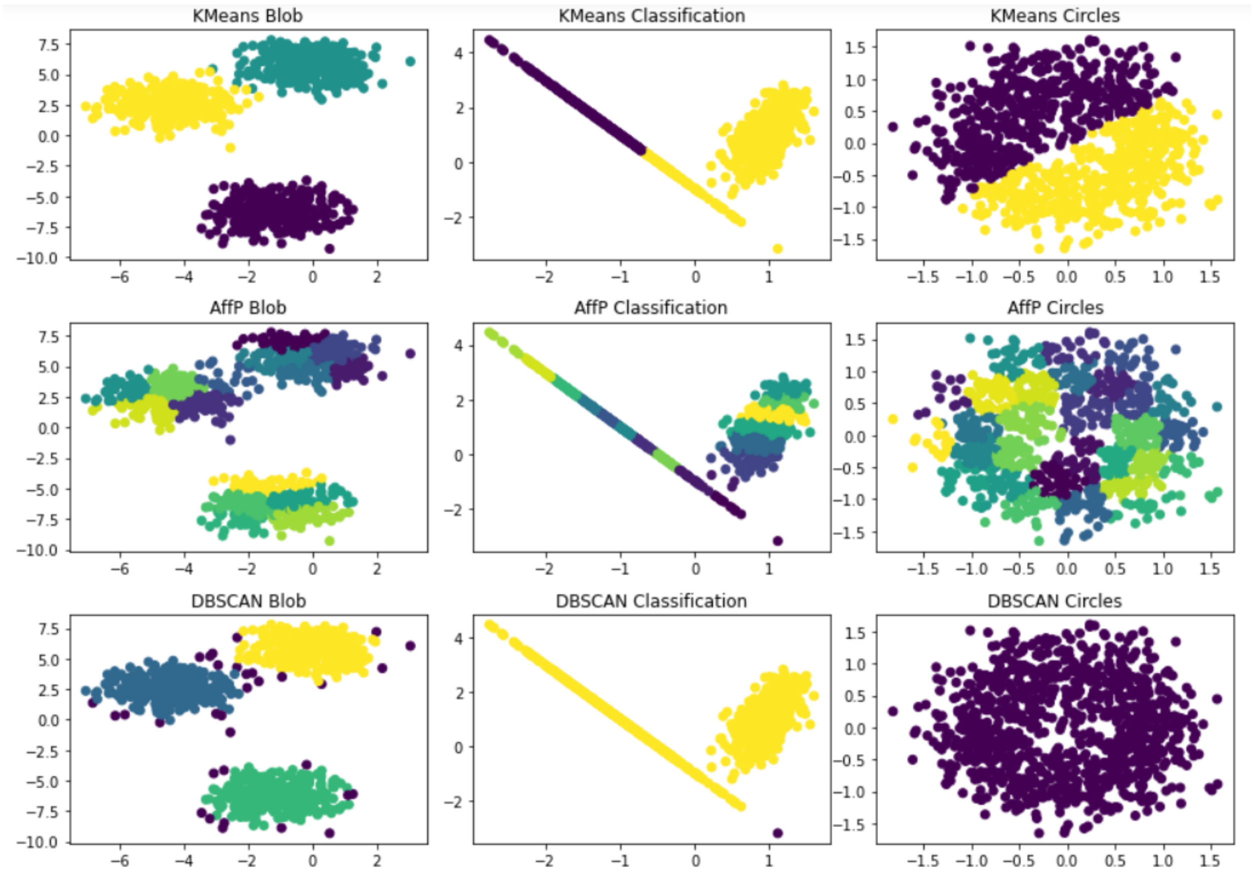
```
+------+--------------------+-------------+-------------+
|      |   Steel Plate Fault |  Ionosphere |   Banknotes |
+======+====================+=============+=============+
| KNN  |                  4 |           2 |           2 |
+------+--------------------+-------------+-------------+
| GNB  |                0.1 |       1e-09 |       1e-09 |
+------+--------------------+-------------+-------------+
| LOGR |                  1 |           2 |           1 |
+------+--------------------+-------------+-------------+
| DTC  |                  8 |           5 |          10 |
+------+--------------------+-------------+-------------+
| GBC  |                  1 |           3 |           3 |
+------+--------------------+-------------+-------------+
| RFC  |                 10 |           8 |           8 |
+------+--------------------+-------------+-------------+
| MLPC |              0.001 |       0.001 |       1e-05 |
+------+--------------------+-------------+-------------+
```

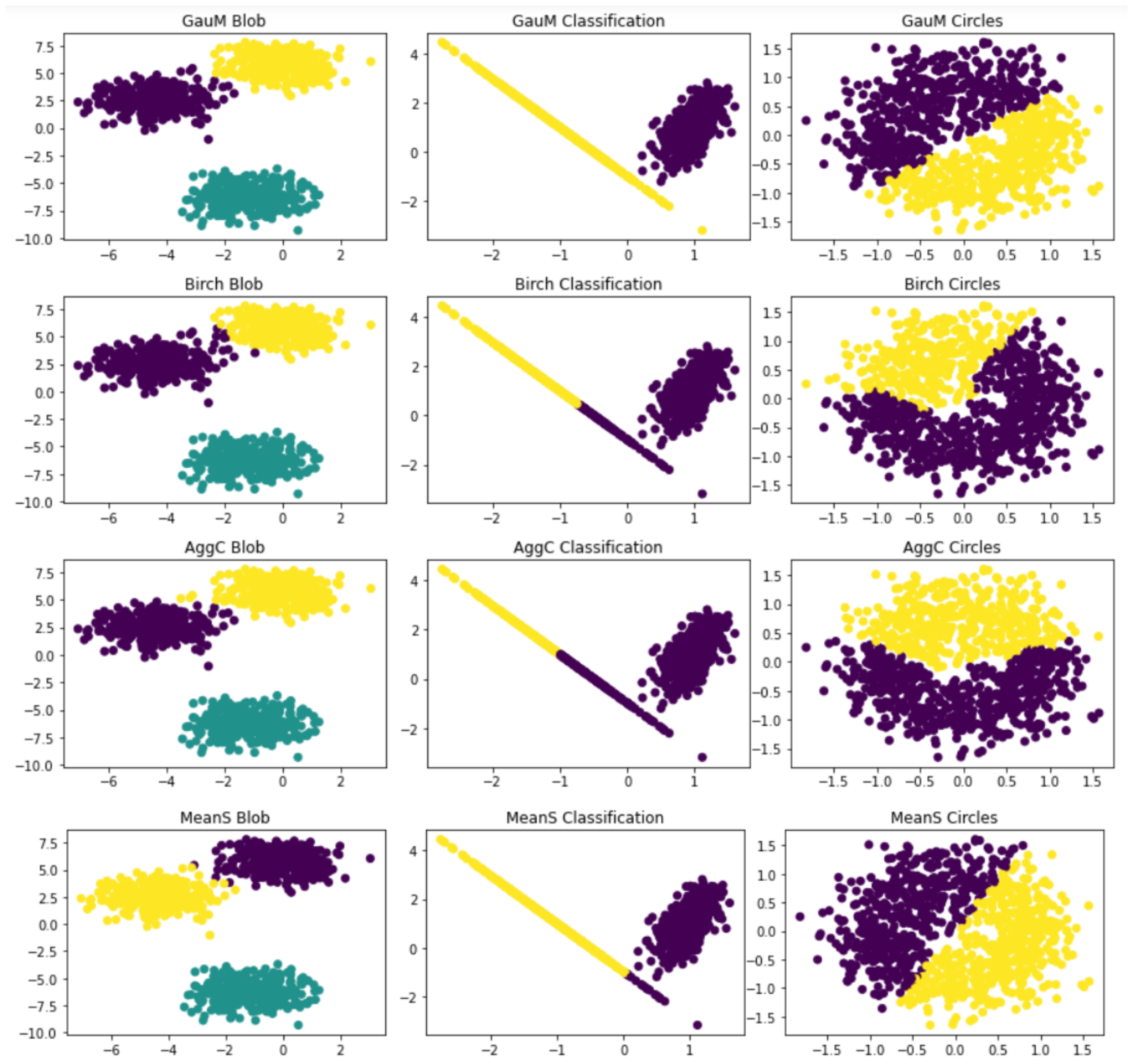
For K Nearest Neighbor the best hyperparameters in order of the datasets is 4,2,2 based on the highest mean accuracy (lowest test error) which is an accurate representation. This makes sense with a model like KNN where the hyperparameters are sensitive and so setting k too low can mean its overfitting and setting k too high could mean its underfitting hence why these parameters around the middle makes sense. One key feature of the results for KNN is that this model fits well for the Banknotes dataset which has less features/x values with only 4 compared to 34 for Ionosphere.
For Gaussian Naive Bayes the best hyperparameters in order of the datasets are 0.1,1e-09 and 1e-09. The hyperparameter is highly sensitive to changing the variance smoothing as with low count the model can overfit so the variance smoothing acts as a lower limit for variance. The

model seems to act similarly for both the ionosphere and banknotes datasets where they share the same best hyperparameter and also produce lower test error than SPF with under 20%.
For Logistic Regression the best hyperparameters in order of the datasets are 1,2 and 1. These values are on the higher range for its complexity control which means this model performs better to a weaker regularization as the lower the C the higher the regularization. This model according to the graphs shows that it isn't super sensitive to the complexity control hyperparameters as the accuracy recorded is consistent between parameters.
For Decision Tree Classifiers the best hyperparameters in order of the datasets are 8,5 and 10. This model is sensitive to the hyperparameter as it determines its accuracy. This is because this model's complexity control is max depth of the tree so the higher the max depth most of the time would mean a more accurate classification. This is why for both SPF and Banknotes the higher max depth means more accurate it is and they are both able to achieve 100% accuracy/0% test error at times.
For Gradient Boosting Classifiers the best hyperparameters in order of the datasets are 1 (but all 5 share 100% accuracy),3 and 3. This works similarly to the Decision Tree Classifier however it creates an ensemble of various estimators. It creates many weaker models and then combines them into a stronger one which is more accurate. This means that the lower max-depth will result in higher accuracies as shown with lower best hyperparameters than Decision Tree Classifier and the lower max depth allows the models to be weaker to then be combined. Once again the model is sensitive to the hyperparameters as shown by the decrease in accuracy as the max depth grows.
For Random Forest Classifiers the best hyperparameters in order of the datasets are 10,8 and 8. This works similarly to Gradient Boosting Classifiers where it is an ensemble and works by creating multiple trees and combining them to make a more accurate result. This produces the same result as Decision Tree Classifiers where the higher max depth results in higher accuracies but this model also causes the results to be even more accurate. Once again the model is sensitive to the hyperparameters as shown by the increase in accuracy as the max depth grows hence the higher best hyperparameters.
For MLP Classifiers the best hyperparameters in order of the datasets are 0.001,0.001 and 1e-05. This model works very similarly to Logistic Regression where the complexity control acts as a smoothing. It works by applying a penalty to encourage weight values to stay small and these results are in the lower range for alpha meaning it is more accurate with lower penalty. This model according to the graphs shows that it isn't super sensitive to the complexity control hyperparameters as the accuracy recorded is consistent between parameters.

Overall the Decision Tree models show the best performance in terms of accuracy with the ensembles being Gradient Boosting Classifier and Random Forest Classifier being more accurate. These models are costly in terms of time and resources but do provide the best performance as they can go into more depth creating a better classifier and also appear to be very consistent as they have a small variance.
One thing to note is that for the models Logistic Regression and MLPClassifier a message appeared stating that the maximum iterations had been reached for the default values of 100

and 200 iterations respectively. This meant that these models had an error converging and these models were not fully optimized.





For the KMeans algorithm it appeared to work well on all 3 datasets by clearly defining the clusters. This method worked particularly well on the blobs as this algorithm works by picking a specified number of centroids (3 for blobs, 2 for classification and circles) and clustering the points closest to these centroids then updating the centroids this allowed for the centroids to be centered in the middle of each of the blobs and allow them to cluster appropriately. The other 2 models showed to have a split down the middle/diagonally and assigned them by the sides of this middle line, this happened because the centroids where centered on each side of the center and so it matched the points that way.
For Mean Shift, the algorithm works very similarly to KMeans as it acts on a centroid hence why the results are very similar however the circles finds a different split along a different line and so does classification. The difference is this algorithm uses a sliding window which moves to find the dense areas in the data points (by moving with the changing mean) and then picks that

mean as the centroid to cluster. This explains why for classification the split is more sided to the right as the data is more dense there with the right blob of points and so the rightmost centroid is more centered to the right than with KMeans. The split for circles would most likely be caused due to the data being denser to the centroids from this algorithm than KMeans.
For Affinity Propagation, I had a convergence warning error appear when setting default values and the graphs were not able to cluster so instead I set a damping to make them converge. The values I set were 0.6 for blobs and 0.85 for classification to allow it to converge and limit iterations from being maxed out. The results from this algorithm did not compose well on the data as it provided multiple classifiers for the data where it seemed more clear for blobs and classifiers at least that there should only be 3/2 clusters/classifiers. This algorithm works by identifying exemplars among other data points and forming clusters based on these exemplars. It does this by sending and receiving information between points so the method is costly and then they converge to form the points. This is why there are lots of clusters because there are no limits set to the algorithm so it picks appropriately the amount of clusters based on its exemplars.
For DBSCAN, this algorithm is also density based like Affinity Propagation however the result differed significantly. This is because this algorithm separates the high density clusters from the low density clusters which are basically outliers so the method works on 2 parameters being eps and minPts to define what are outliers and what are part of the high density clusters. The algorithm worked as expected on blobs where it found the 3 clear clusters and then the points which were spaced out from these clusters were identified as outliers and grouped together. For classification the algorithm clustered the data as one by merging them as densely packed despite clear separation in the data. The algorithm also identified one outlier on the right due to a large distance separation. For the circles the algorithm failed to identify any distinct clusters and instead grouped the data as one cluster which makes sense for the algorithm since the data is one large densely packed area and there are no low density areas.
For Gaussian Mixture Model, this algorithm works by using means and covariances to make a generative model. This model is a soft version of KMeans so it doesn't act from a centroid but instead based on the probabilities of being in each group hence why the grouping seems more accurate to what a human would classify the data as. The classification data shows the clustering separated by the blob and the line and the blobs data is assigned appropriately and the circle is done by a split almost identical to KMeans.
For Agglomerative Clustering, this algorithm is hierarchical based (nested) where it builds the tree from the bottom up by keeping on merging 2 clusters with linkage criteria until all clusters have been merged into a single nested cluster. The results show the 3 blobs being identified appropriately and the vertical split being apparent for classification as with the centroid algorithms and the circles having a split where one cluster is smaller than the other.
For BIRCH it is very similar to Agglomerative Clustering where it is hierarchical however this algorithm takes a top down approach to build a summary of the data. This is further shown with the similar clustering results for the data as Agglomerative. The algorithm identifies the 3 blobs successfully and takes the split down the middle for the classification data. For the circles data a smaller section was left for the second cluster than for Agglomerative.