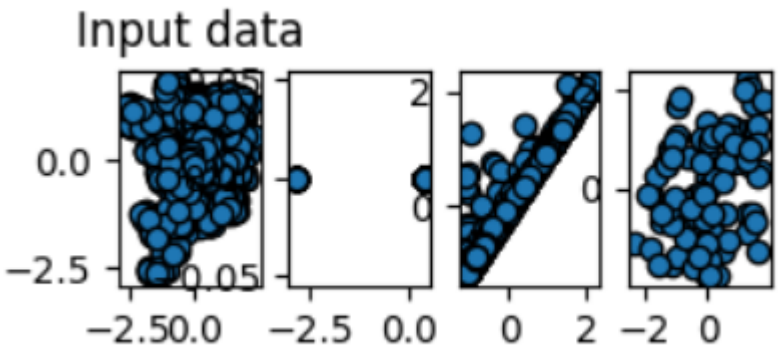
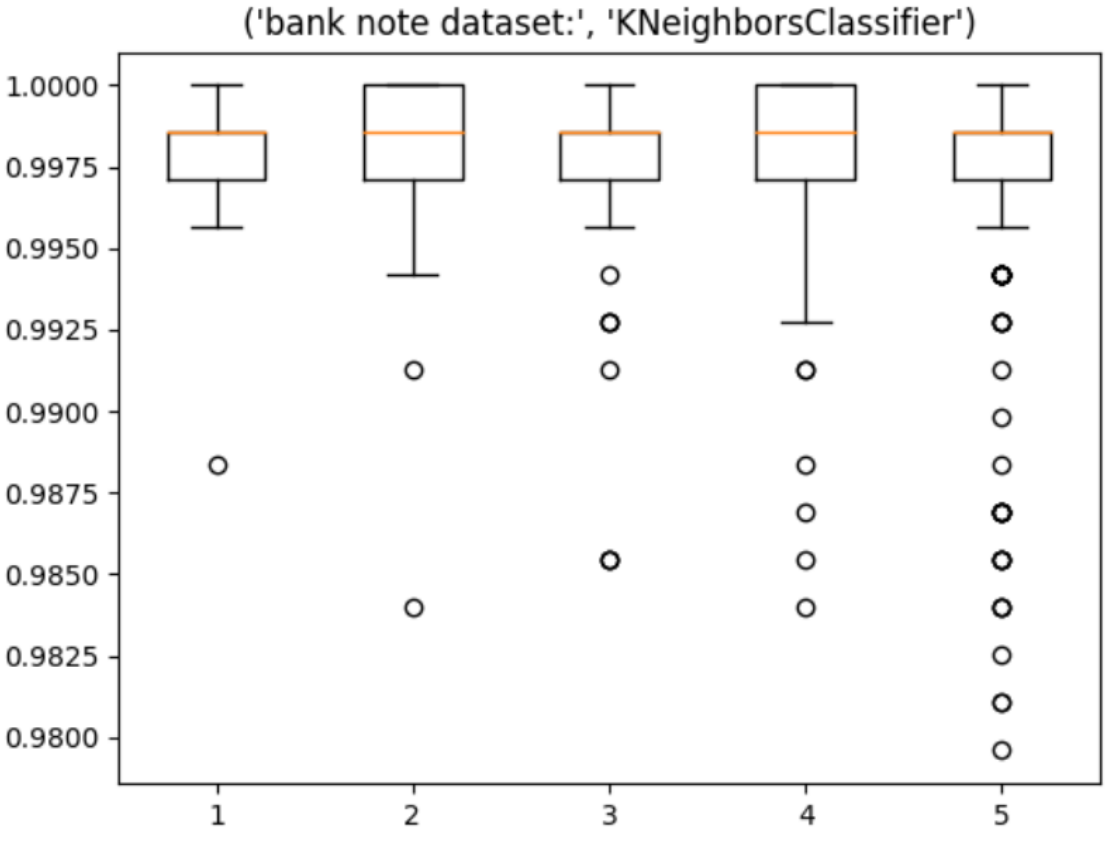
Part one: classifiers

First here is a scatter plot for input data with following order:
[bank note dataset, Ionosphere dataset:,steel_P dataset, myfakedataset]



KNeighborsClassifier

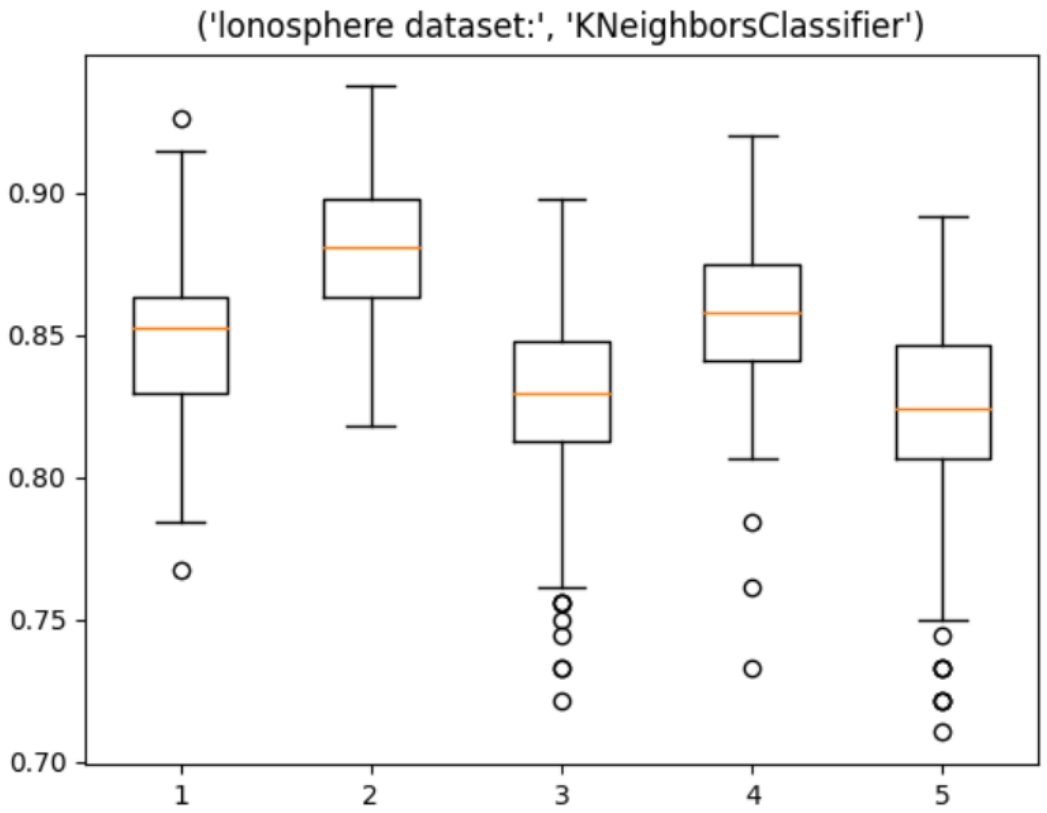
Here are outcome for each datasets(all datasets have been randomly split for 300 times):
Bank Note dataset:



```
KNN of bank note dataset:
The best average value of bank note dataset: is 0.9984110787172011 2
The highest value for the control parameter is 2
```

From the above boxplot for bank note dataset, we can tell the performance for all 5 values are very good, and all accuracies are above 98 percent . When k value is equal to 2, the performance is the best and the screenshot below also justify that. The performance of when value is equal to 4 is also good but compared to 2, it has more outlinears which indicate there are more unusual points when the value is equal to 4. Performances of value =1,3,5 are similar except when value= 3,5 there are more outlinears.

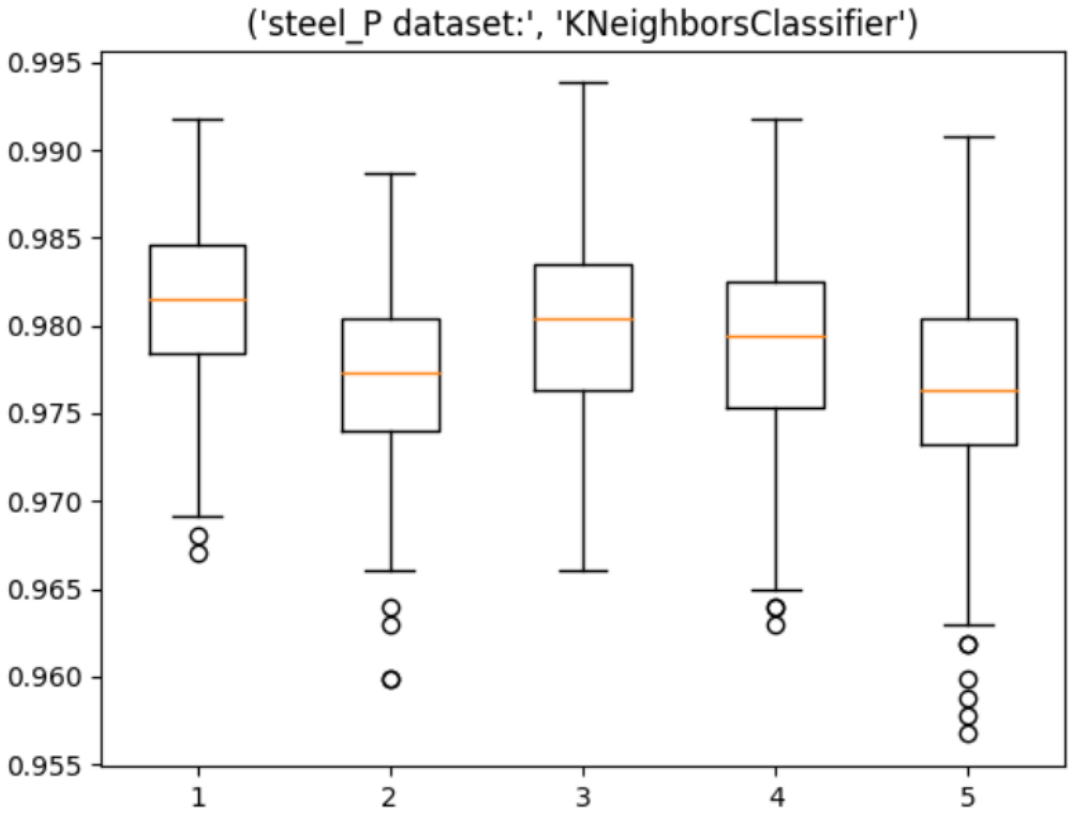
Lonosphere dataset:




```
KNN of lonosphere dataset:
The best average value of lonosphere dataset: is 0.8785795454545454 2
The highest value for the control parameter is 2
```

From the above boxplot, the K nearest classifier is not that good, with average accuracies for 5 values are between 80 to 90 percent. When k value = 2, the performance is the best as it has the best average accuracy and some of the highest outlinear are close to 95 percent accuracy. The value= 4 and 5 situation has the worst performance as they have lowest accuracy and they are unstable as some of the lowest outliers have 65 percent accuracy.

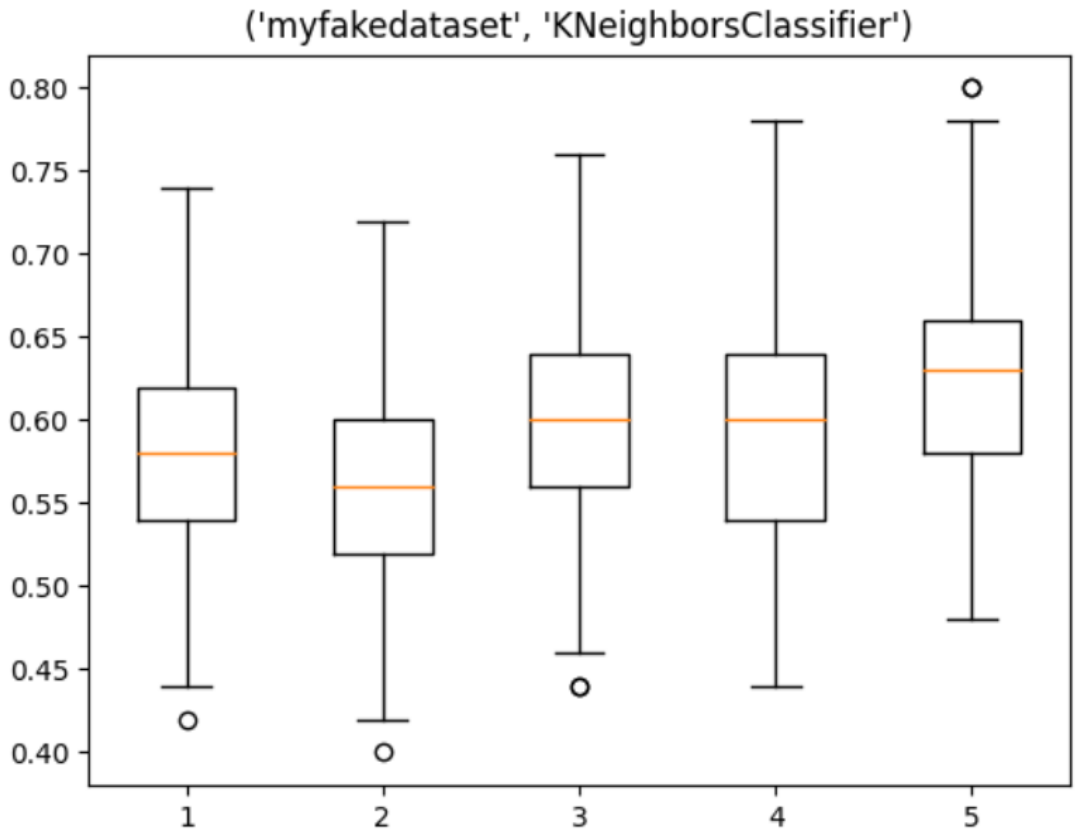
Steel_P dataset:




```
KNN of steel_P dataset:
The best average value of steel_P dataset: is 0.9814349467902506 2
The highest value for the control parameter is 1
```

From the above boxplot, the performance is good, with average accuracies for 5 values are between 95 to 99 percent. When k value = 1, the performance is the best as it has the best average accuracy and the third quartile have over 99 percent accuracy. The value= 5 situation has the worst performance as they have lowest accuracy and one of the lowest outliers has 95 percent accuracy which is very low for this database.

myfakedata:

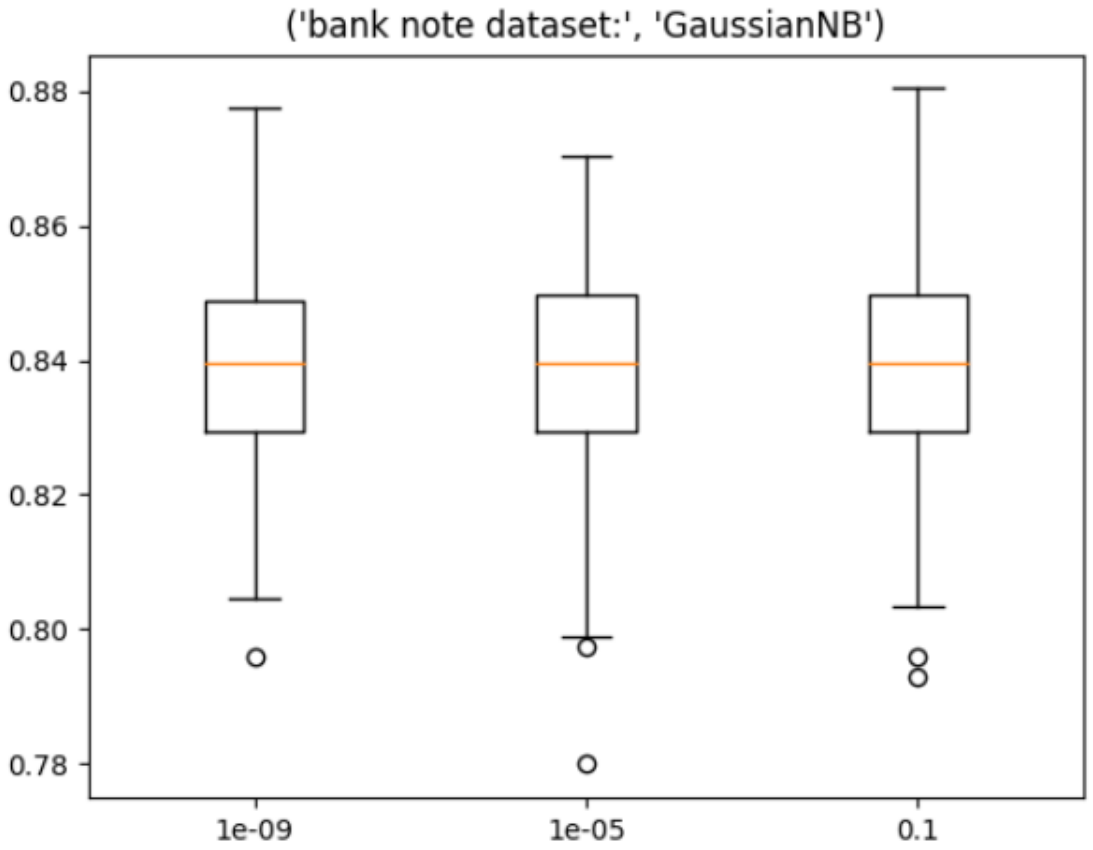



```
KNN of myfakedataset
The best average value of myfakedataset is 0.6302 2
The highest value for the control parameter is 5
```

From the above boxplot, the performance for the fake data is bad with average accuracies for 5 values are between 40 to 80 percent which means the performance is also unstable. When k value = 5, the performance is the best as it has the best average accuracy. The value= 1 situation has the worst performance as they have lowest accuracy.

GaussianNB

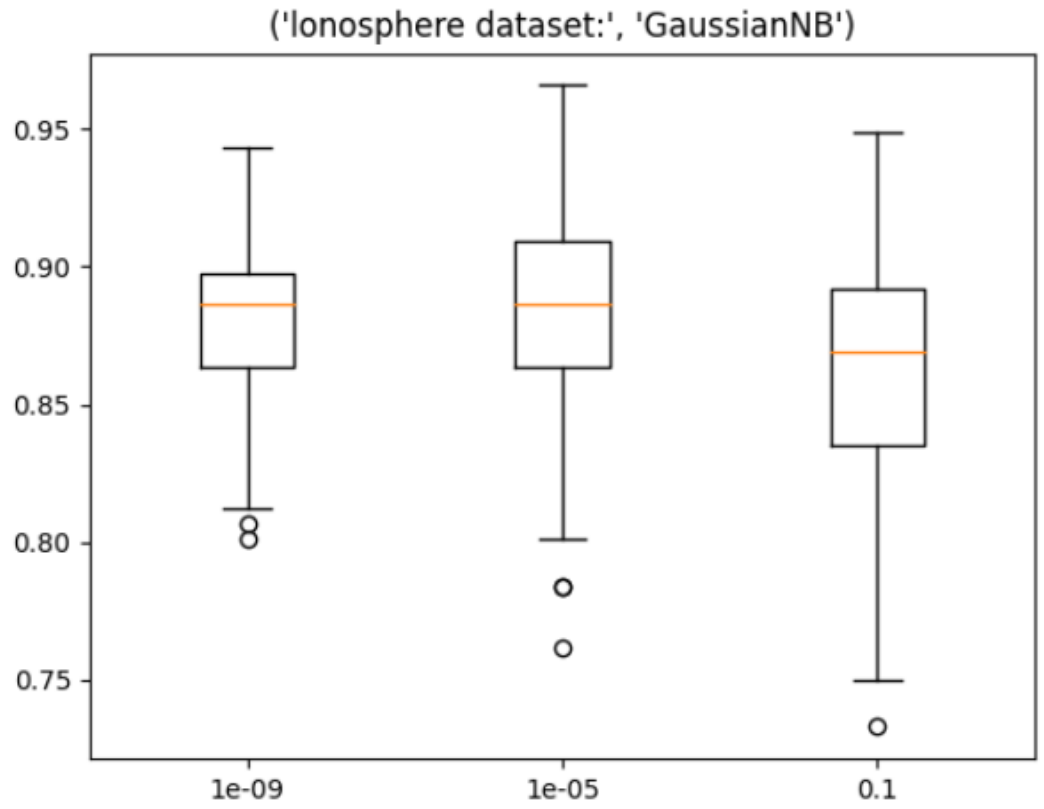
Bank Note dataset:




```
GaussianNB of bank note dataset:
The best average value of bank note dataset: is 0.8397716229348883 4
The highest value for the control parameter is 1e-09
```

From the above boxplot, the performance for this dataset is good. All three controller values have similar performance, the mean,first quartile,third quartile are all similar.The one with the highest average accuracy is 1e-09

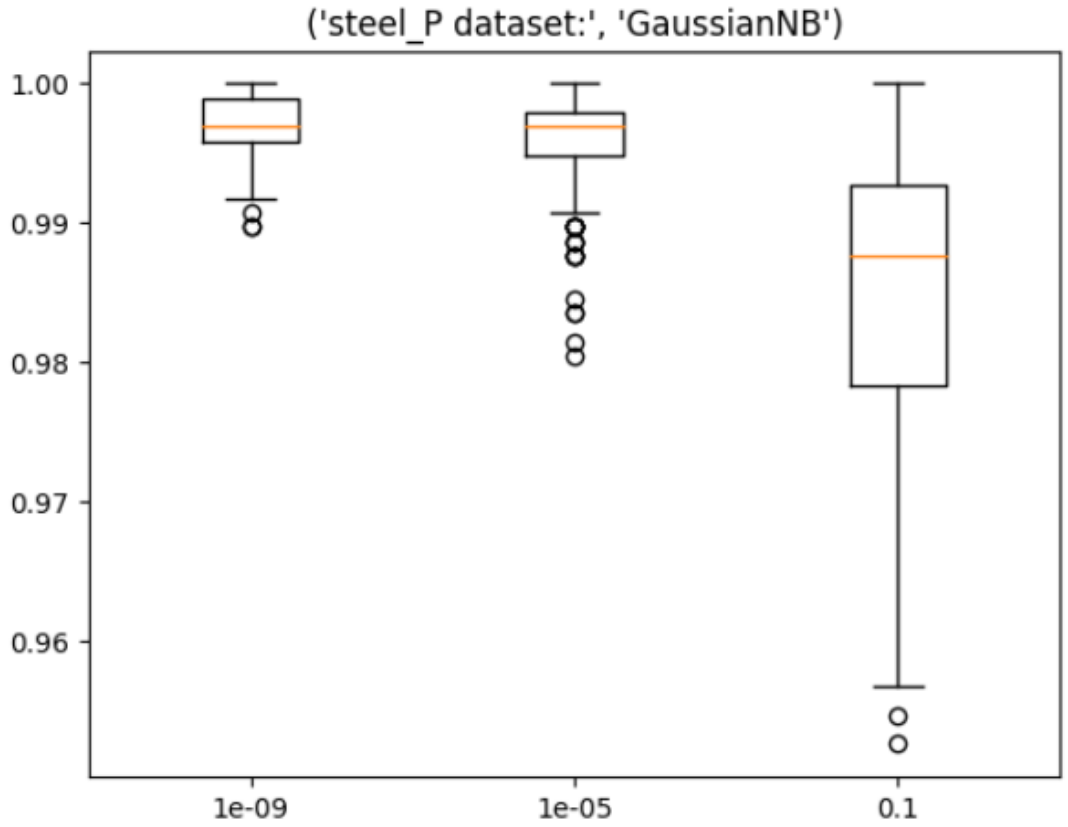
Lonosphere dataset:




```
GaussianNB of lonosphere dataset:
The best average value of lonosphere dataset: is 0.8847537878787879 4
The highest value for the control parameter is 1e-05
```

From the above boxplot, the performance is good but not stable, with average accuracies for 3 values are between 75 to 95 percent. When controller value = 1e-5, the performance is the best as it has the best average accuracy and the third quartile has over 90 percent accuracy.

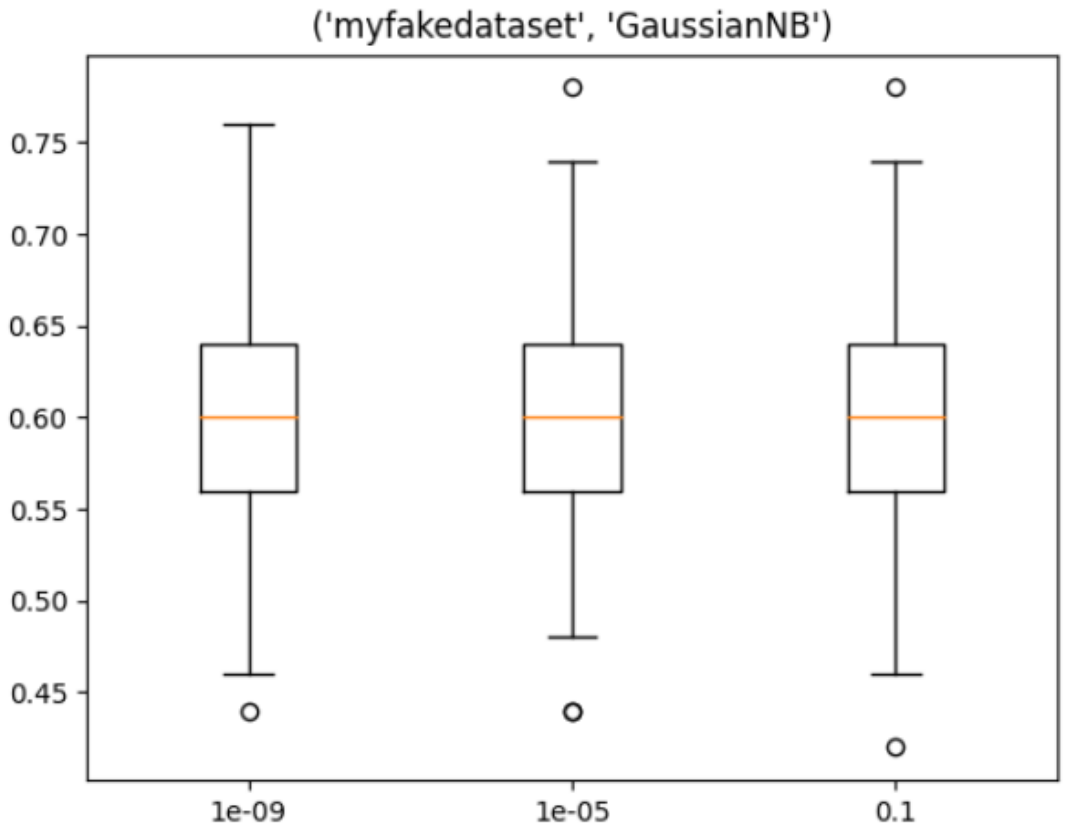
Steel_P dataset:




```
GaussianNB of steel_P dataset:
The best average value of steel_P dataset: is 0.9973120494335737 4
The highest value for the control parameter is 1e-09
```

From the above boxplot, the performance for 1e-9 and 1e-5 is good, with average accuracies for 3 values are between 98 to 100 percent. When controller value = 0.1, the performance is also good but not stable. The highest accuracy and the lowest accuracy's difference is big.

myfakedata:

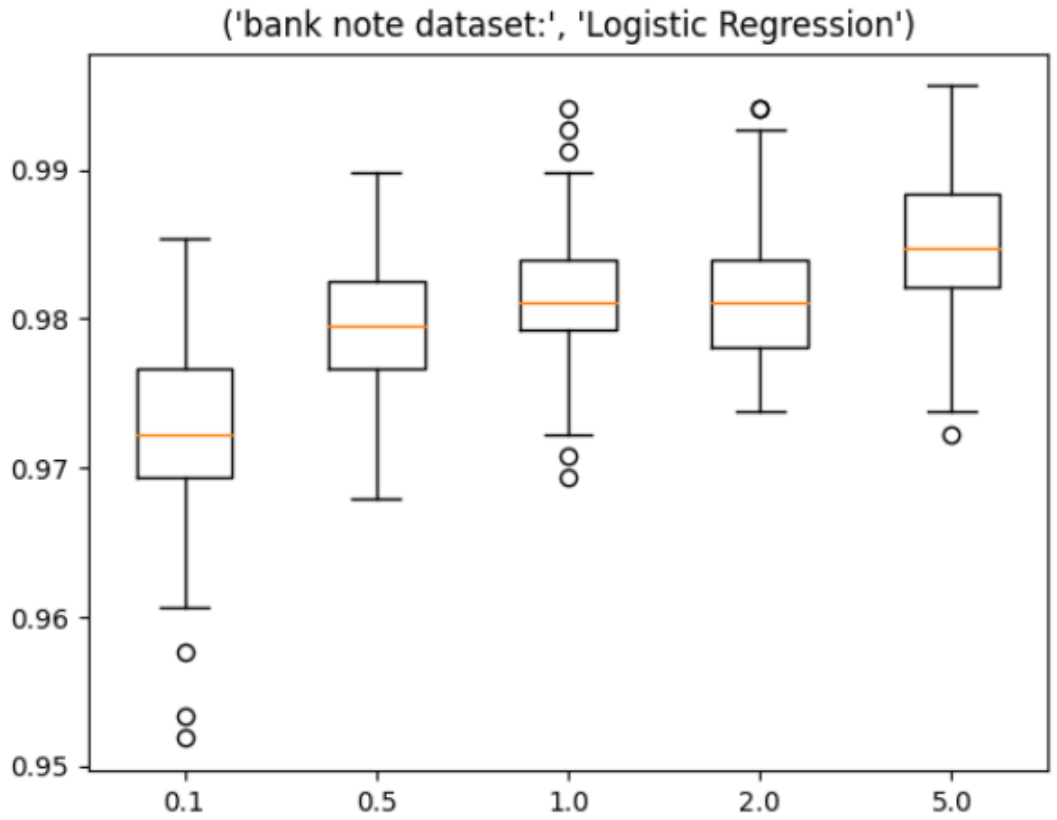



```
GaussianNB of myfakedataset
The best average value of myfakedataset is 0.6024 4
The highest value for the control parameter is 0.1
```

From the above boxplot, the performance for the fake data is bad with average accuracies for 3 values are between 45 to 75 percent which means the performance is also unstable.

LogisticRegression:

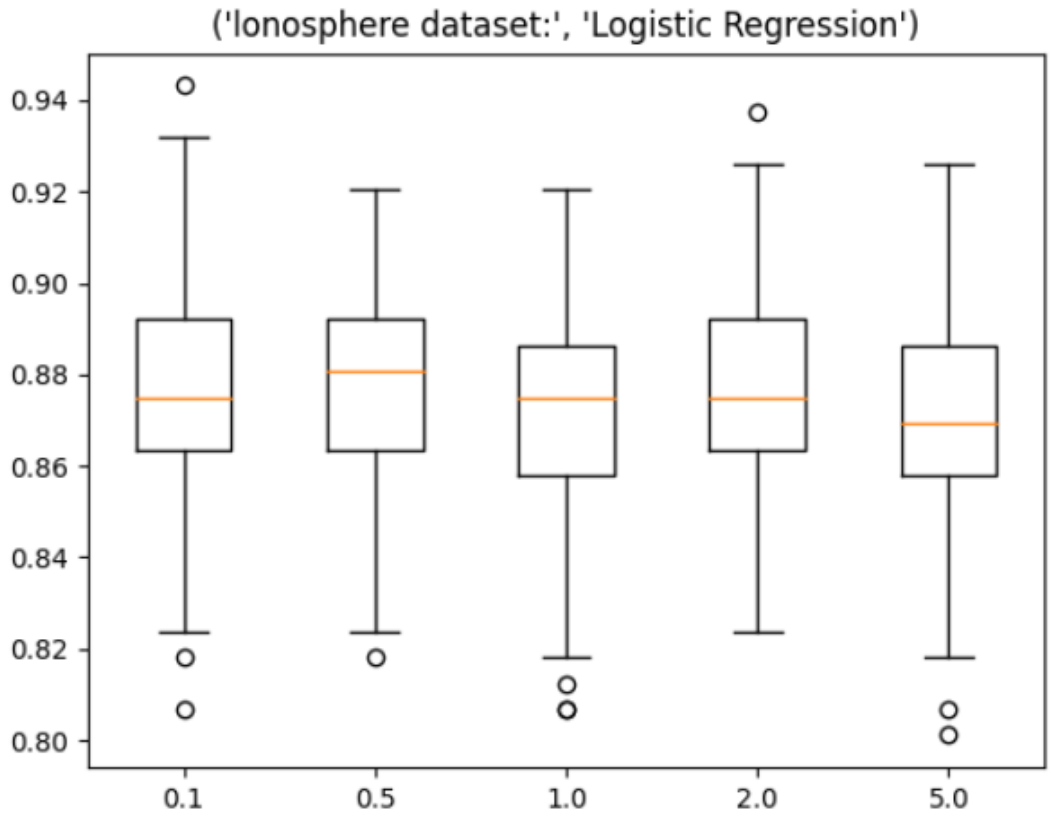
Bank Note dataset:




```
Logistic Regression of bank note dataset:
The best average value of bank note dataset: is 0.9847035957240039 2
The highest value for the control parameter is 5.0
```

From the above boxplot, the performance is good, with average accuracies for 5 values are between 95 to 99 percent. When value = 5, the performance is the best as it has the best average accuracy and the third quartile have over 99 percent accuracy. The value= 0.1 situation has the worst performance as they have lowest accuracy and one of the lowest outliers has 95 percent accuracy which is very low for this database.

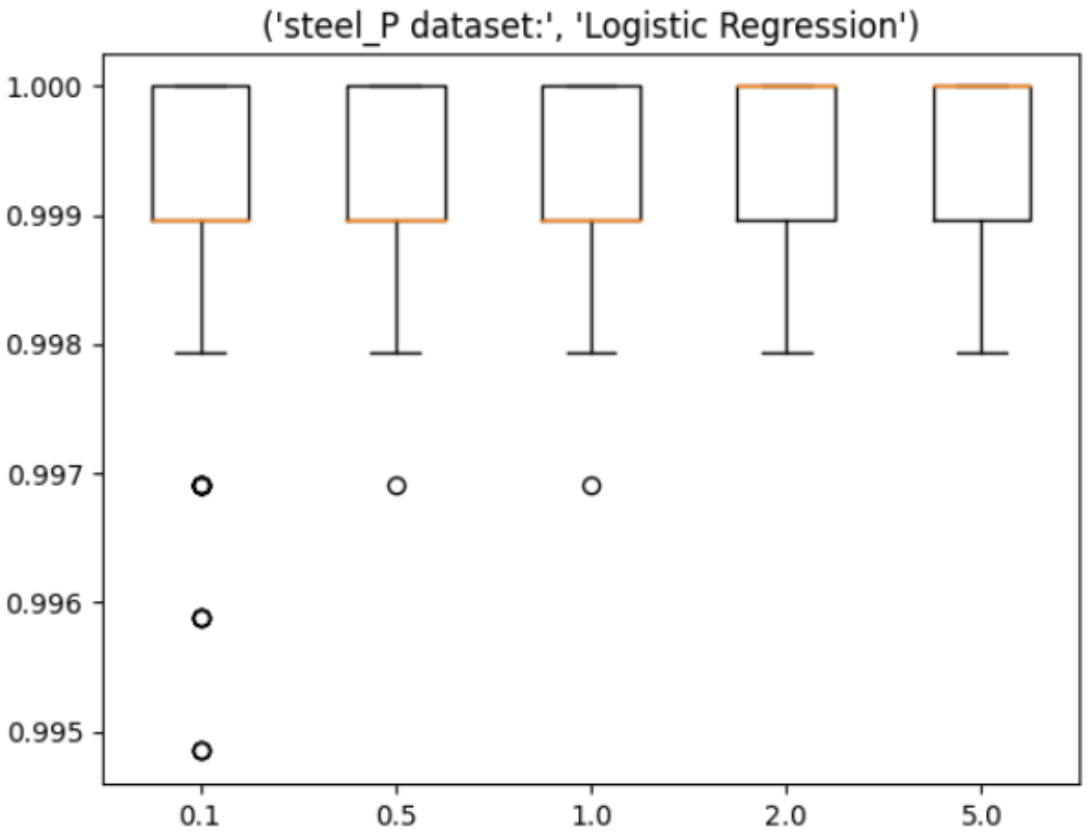
Lonosphere dataset:




```
Logistic Regression of lonosphere dataset:
The best average value of lonosphere dataset: is 0.8760416666666667 2
The highest value for the control parameter is 0.5
```

From the above boxplot, the K nearest classifier is not that good, with average accuracies for 5 values are between 80 to 90 percent. When k value = 0.5, the performance is the best as it has the best average accuracy .The value=5 situation has the worst performance as they have lowest median accuracy .

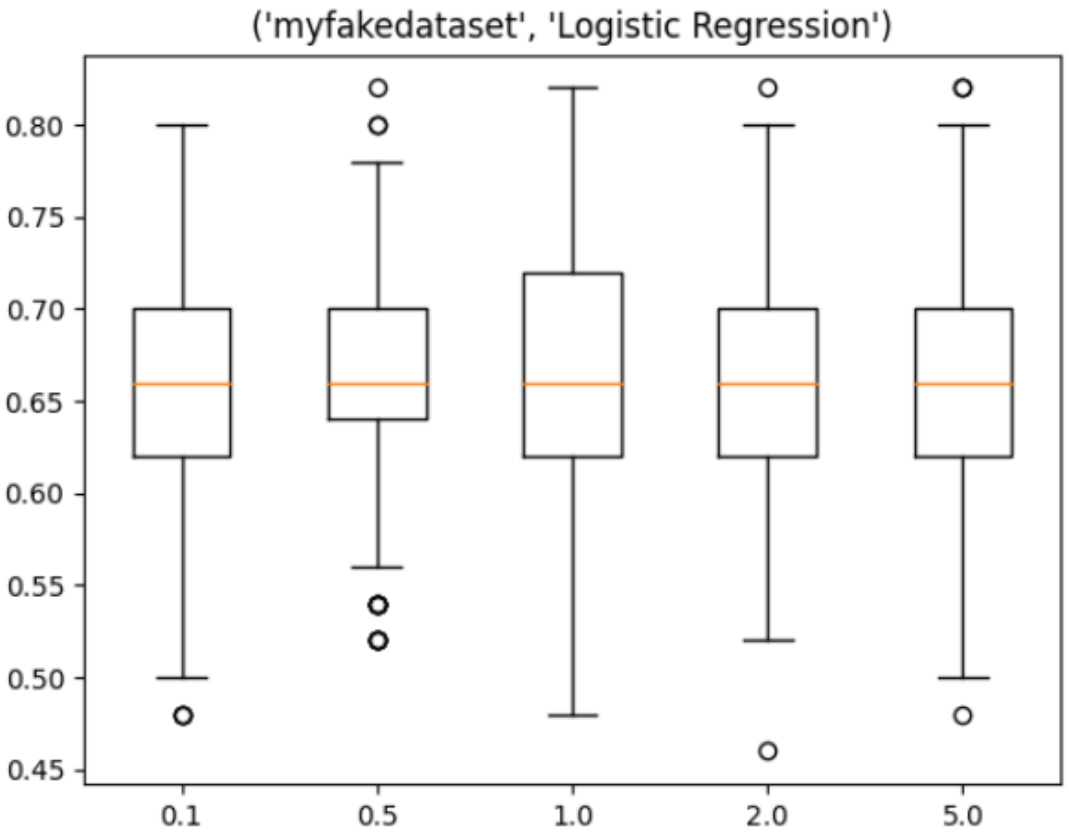
Steel_P dataset:




```
Logistic Regression of steel_P dataset:
The best average value of steel_P dataset: is 0.9997150703741847 2
The highest value for the control parameter is 5.0
```

From the above boxplot, the performance is good, with average accuracies for 5 values are all above 99 percent. The value=4 and value = 5 are similar and have similar median but when value = 5, the performance is the best as it has the best average accuracy and the third quartile have over 99 percent accuracy. The value= 0.1 situation has the worst performance as they have lowest accuracy and one of the lowest outliers has 995 percent accuracy which is very low for this database.

myfakedata:

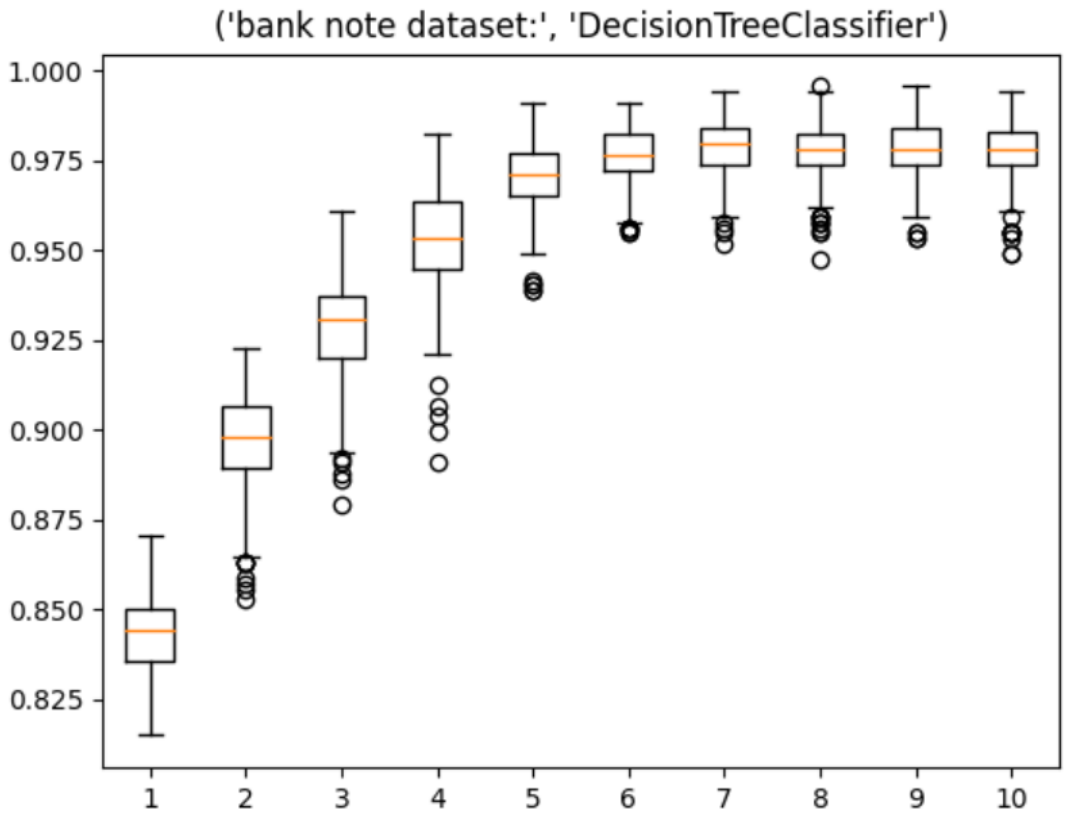



```
Logistic Regression of myfakedataset
The best average value of myfakedataset is 0.6676666666666666 2
The highest value for the control parameter is 0.5
```

From the above boxplot, the performance for the fake data is bad with average accuracies for 3 values are between 45 to 80 percent which means the performance is also unstable.

DecisionTreeClassifier:

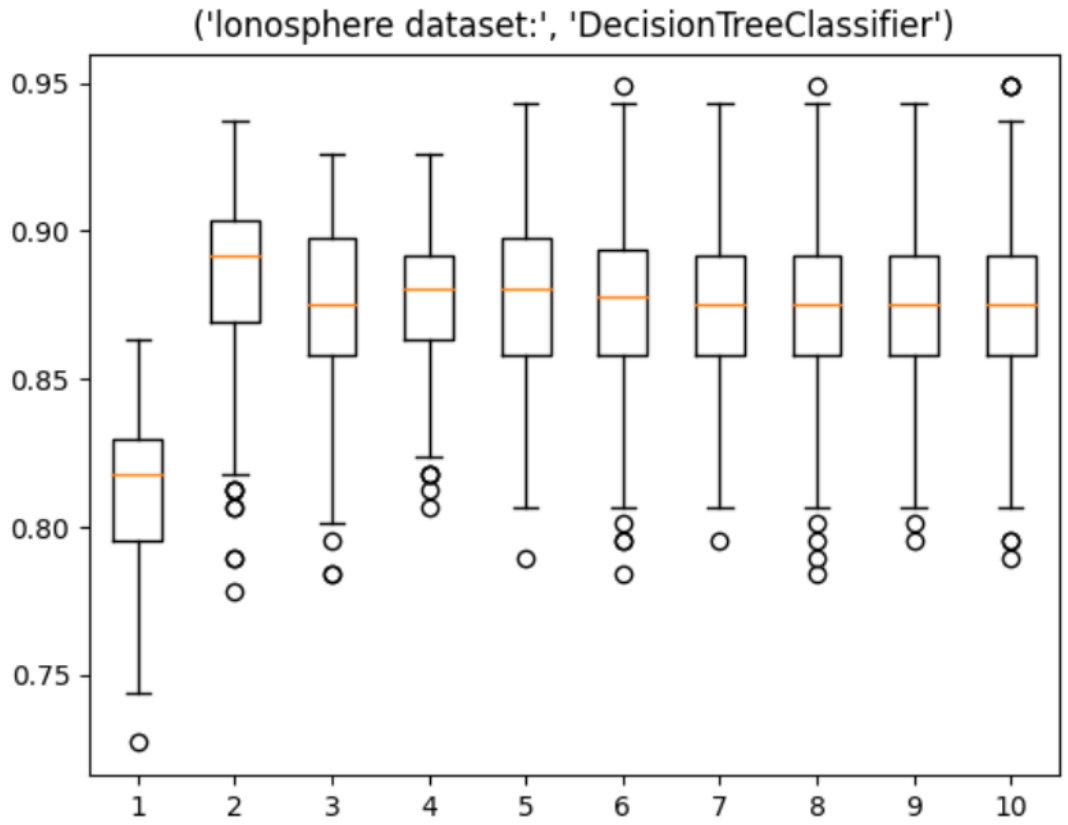
Bank Note dataset:




```
Decision Tree Classifier for bank note dataset:
The best average value of bank note dataset: is 0.978245869776482 2
The highest value for the control parameter is 9
```

From the above boxplot, the performance is good, with average accuracies for the last 6 values are between 90 to 99 percent. When value = 9, the performance is the best as it has the best average accuracy.The value= 1 situation has the worst performance as they have lowest accuracy. For the first 4 values, the performance is not that good but we can see as the max depth increases the performance will increase when max depth values are small.

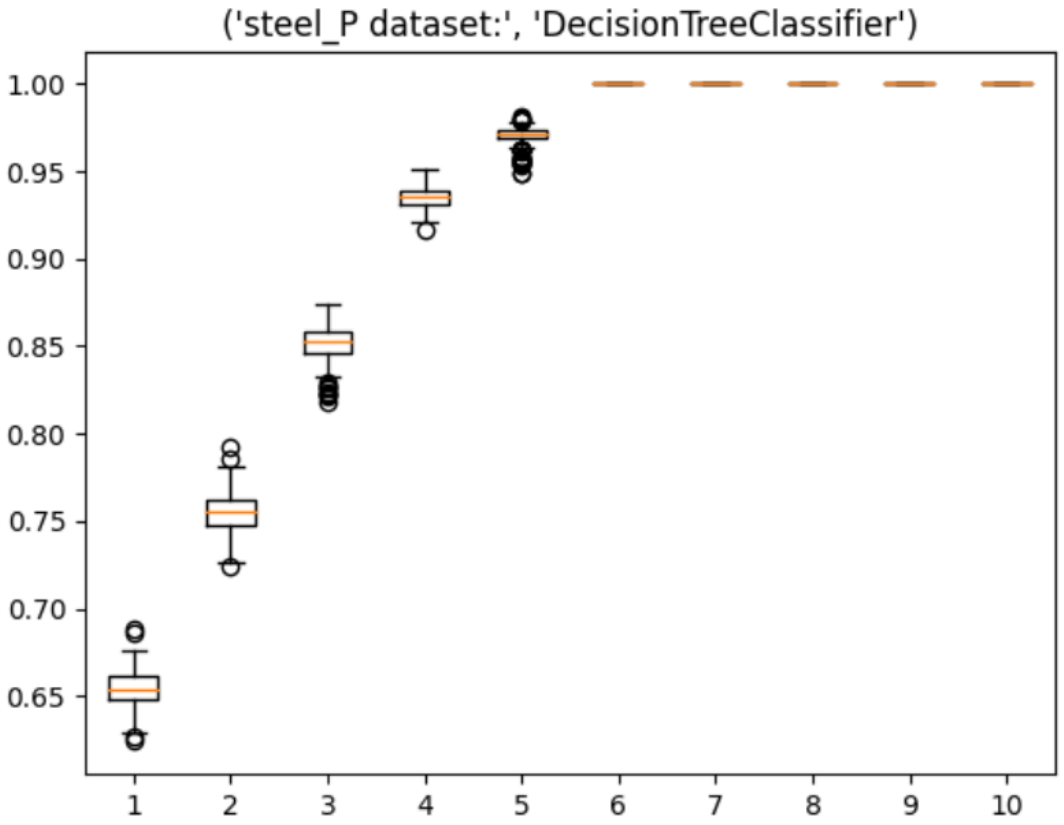
Lonosphere dataset:




```
Decision Tree Classifier for lonosphere dataset:
The best average value of lonosphere dataset: is 0.8833143939393939 2
The highest value for the control parameter is 2
```

From the above boxplot, the performance is good except when max depth equals to 1, with average accuracies for the last 8 values are between 80 to 95 percent. When value = 2, the performance is the best as it has the best average accuracy.The value= 1 situation has the worst performance as they have lowest accuracy.

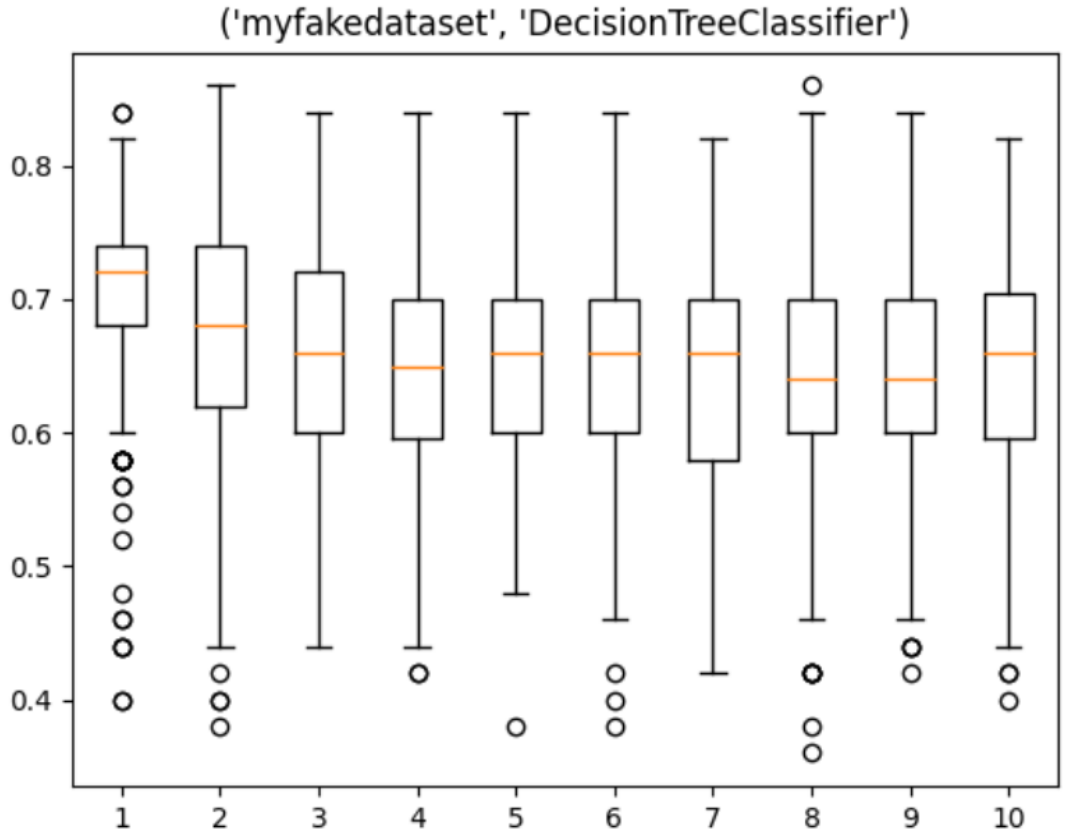
Steel_P dataset:




```
Decision Tree Classifier for steel_P dataset:
The best average value of steel_P dataset: is 1.0 2
The highest value for the control parameter is 6
```

From the above boxplot, the performance is good especially when max depth equals to 6 to 10, they all have 100 percent accuracy for all iterations.The value= 1 situation has the worst performance as they have lowest accuracy. For the first 4 values, the performance is not that good but we can see as the max depth increases the performance will increase when max depth values are small.

myfakedata:

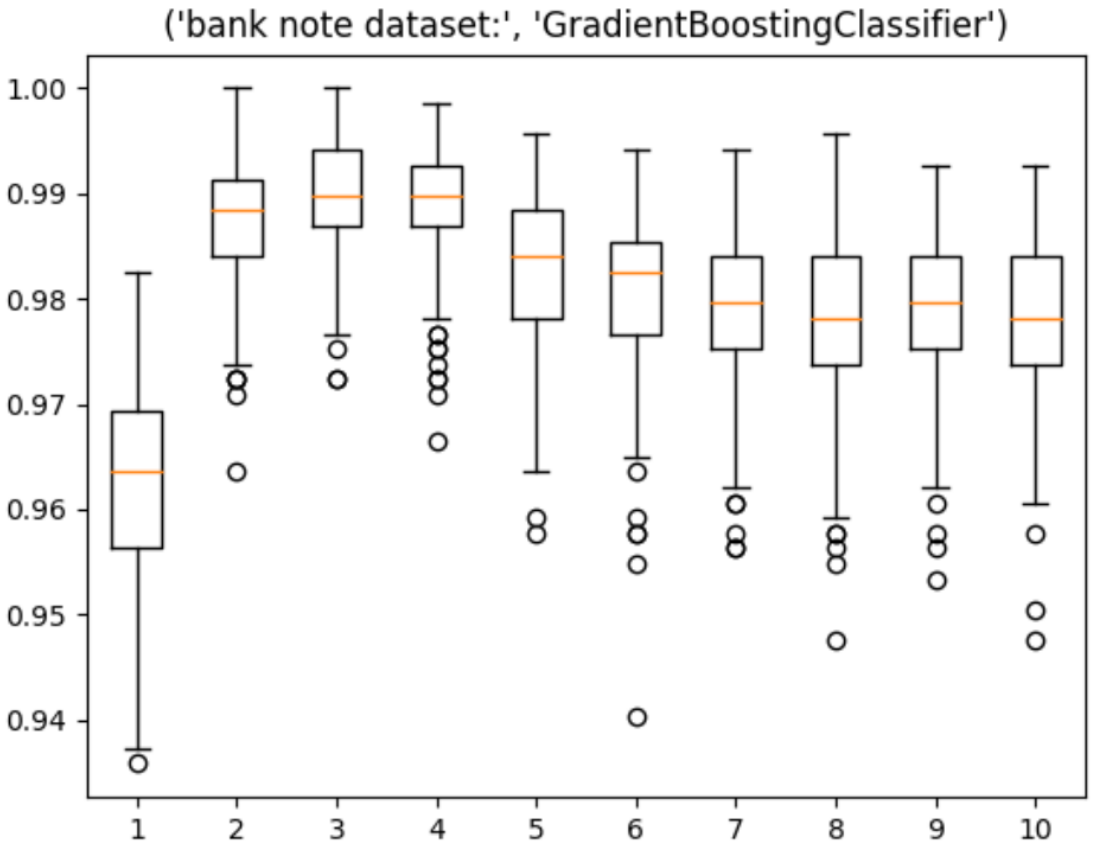



```
Decision Tree Classifier for myfakedataset
The best average value of myfakedataset is 0.7010666666666666 2
The highest value for the control parameter is 1
```

From the above boxplot, the performance for the fake data is bad with average accuracies for 10 values are between 40 to 80 percent which means the performance is also unstable.

GradientBoostingClassifier:

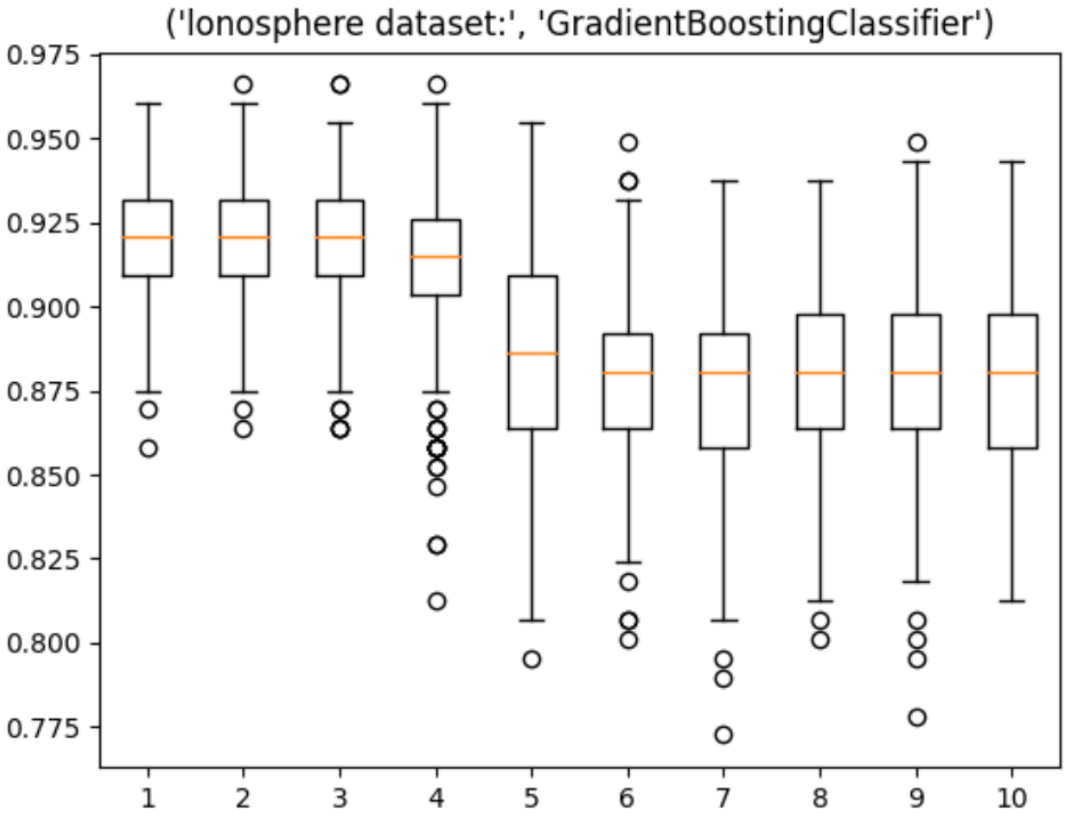
Bank Note dataset:




```
GradientBoostingClassifier for bank note dataset:
The best average value of bank note dataset: is 0.9901457725947522 2
The highest value for the control parameter is 3
```

From the above boxplot, the performance is good, with average accuracies for the last 8 values are between 96 to 99 percent. When value = 3, the performance is the best as it has the best average accuracy.The value= 1 situation has the worst performance as they have lowest accuracy.

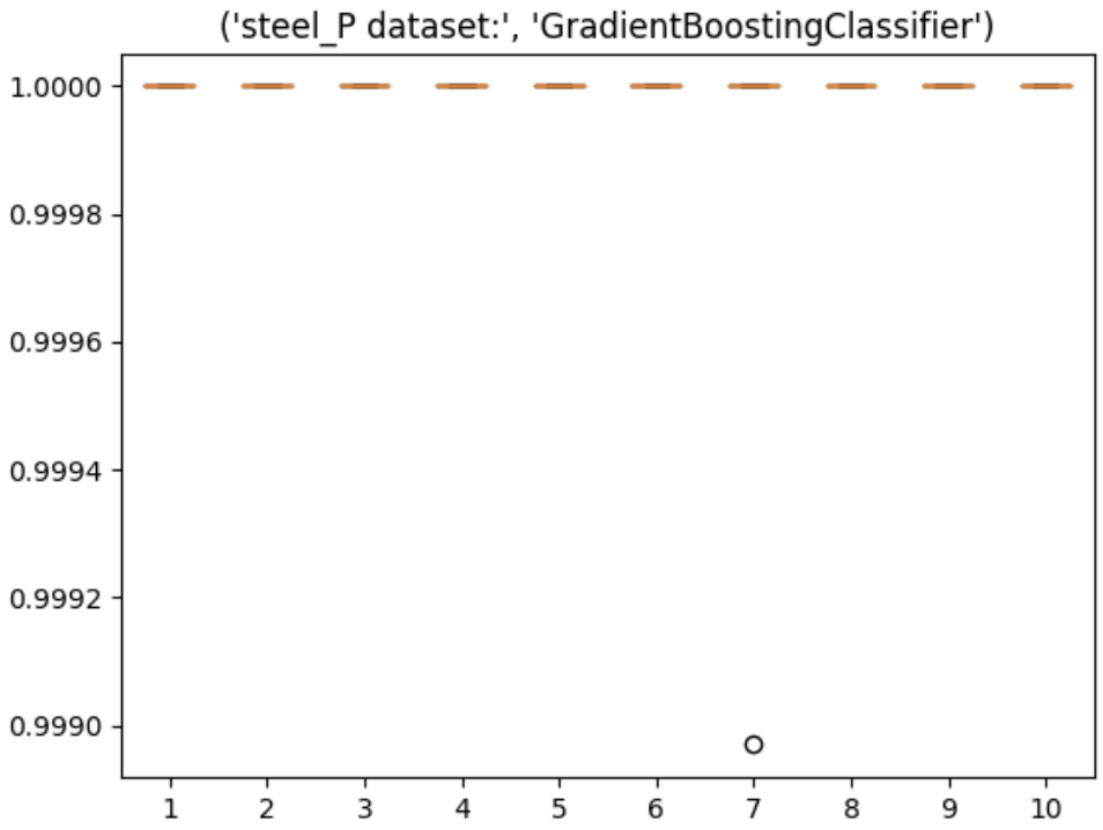
Lonosphere dataset:




```
GradientBoostingClassifier for lonosphere dataset:
The best average value of lonosphere dataset: is 0.921155303030303 2
The highest value for the control parameter is 2
```

From the above boxplot, the performance is good. The first four values have better performance compared to others which have around 92 percent average accuracy rate.When value = 2, the performance is the best as it has the best average accuracy.The value= 7 situation has the worst performance as they have lowest accuracy.

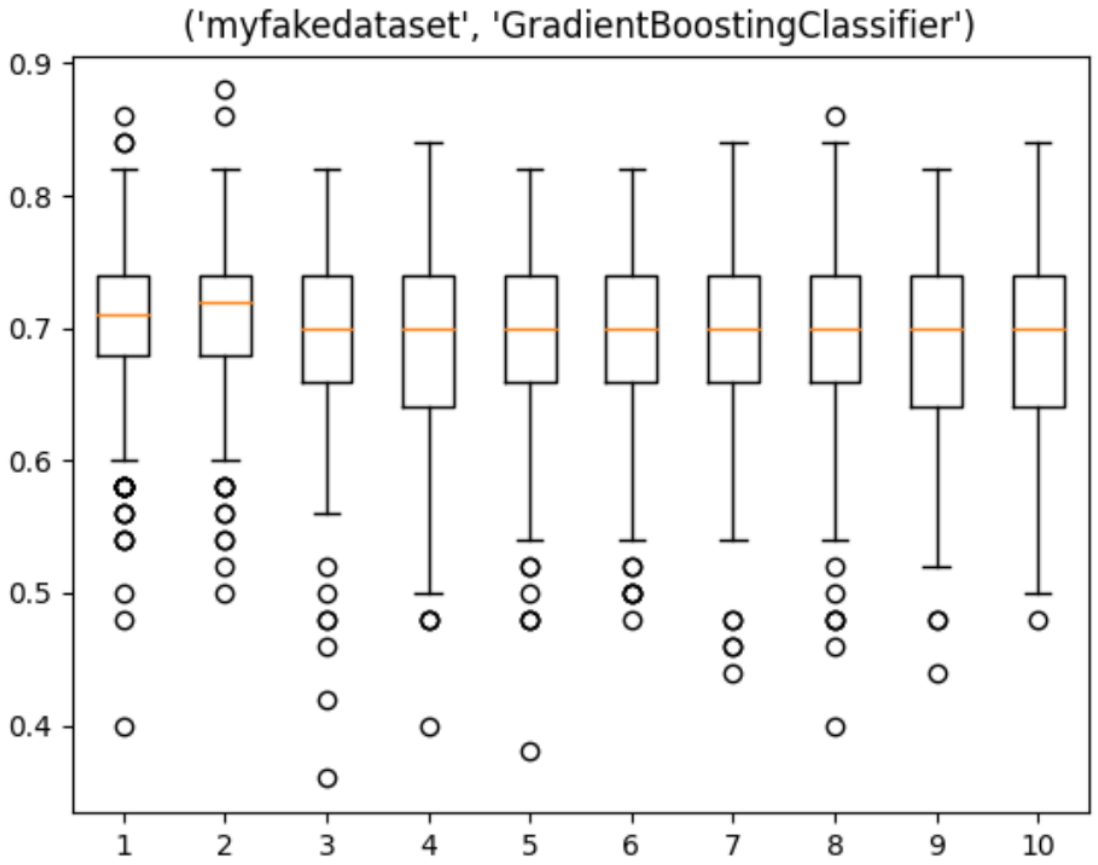
Steel_P dataset:




```
GradientBoostingClassifier for steel_P dataset:
The best average value of steel_P dataset: is 1.0 2
The highest value for the control parameter is 1
```

From the above boxplot, all values have achieved 100 percent accuracy.

myfakedata:

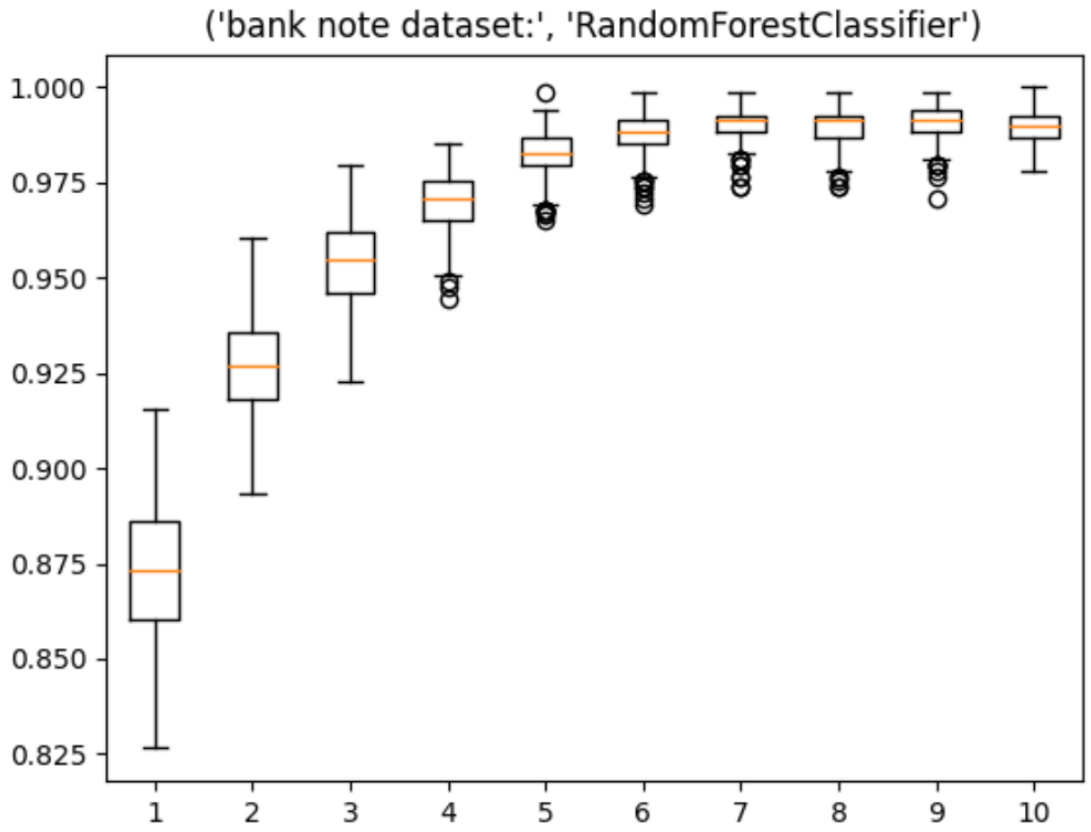



```
GradientBoostingClassifier for myfakedataset
The best average value of myfakedataset is 0.7116 2
The highest value for the control parameter is 2
```

From the above boxplot, the performance for the fake data is bad with accuracies for 10 values are between 40 to 80 percent which means the performance is also unstable. The max depth value with 1 and 2 performs better overally.

RandomForestClassifier:

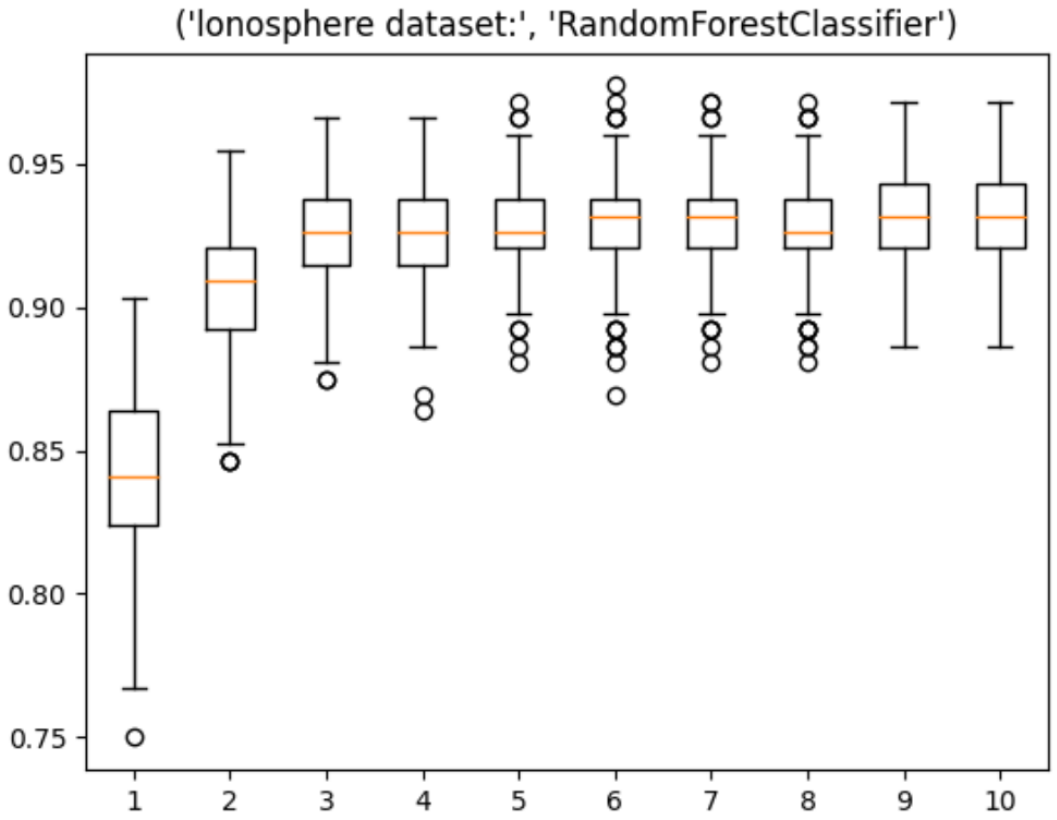
Bank Note dataset:




```
RandomForestClassifier for bank note dataset:
The best average value of bank note dataset: is 0.9907871720116618 2
The highest value for the control parameter is 9
```

From the above boxplot, the performance is good. We can see as the max depth increases the performance will increase when max depth values are small.When value = 9, the performance is the best as it has the best average accuracy.The value= 1 situation has the worst performance as they have lowest accuracy.

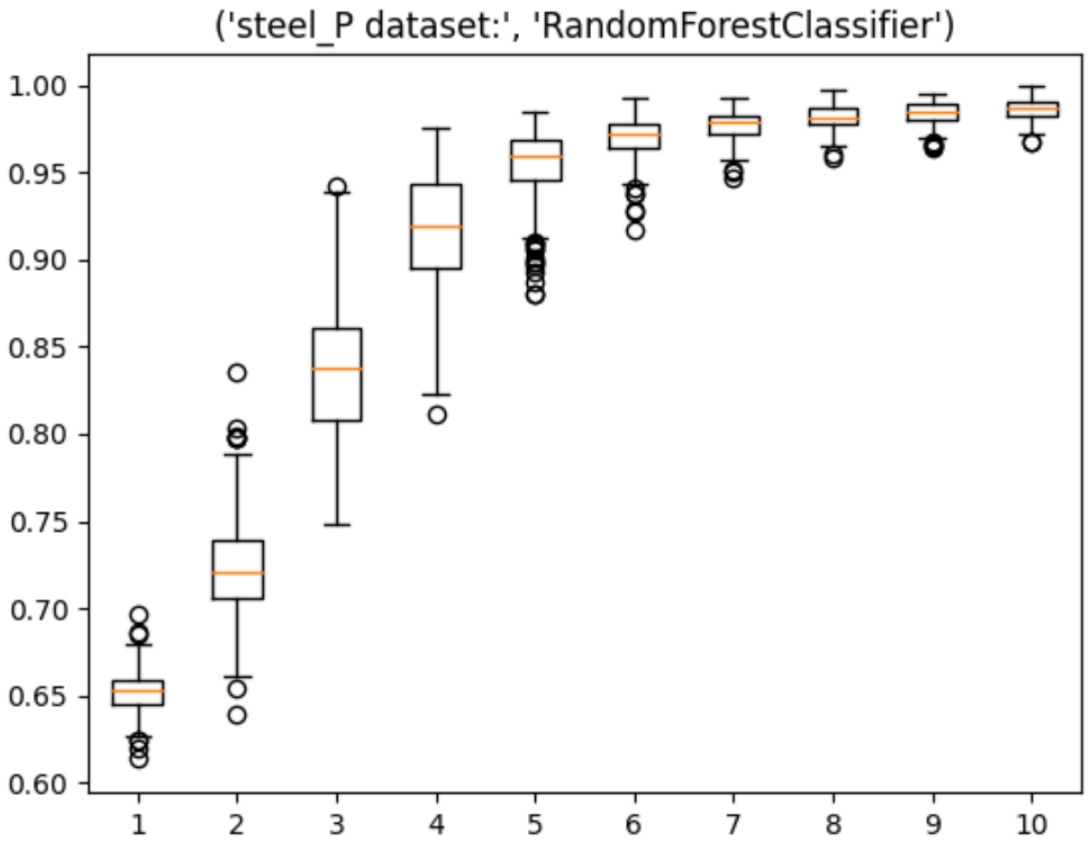
Lonosphere dataset:




```
RandomForestClassifier for lonosphere dataset:
The best average value of lonosphere dataset: is 0.9304166666666667 2
The highest value for the control parameter is 10
```

From the above boxplot, the performance is good. We can see as the max depth increases the performance will increase when max depth values are small. The performance between max depth = 4 to max depth =10 are similar . When value = 10, the performance is the best as it has the best average accuracy.The value= 1 situation has the worst performance as they have lowest accuracy.

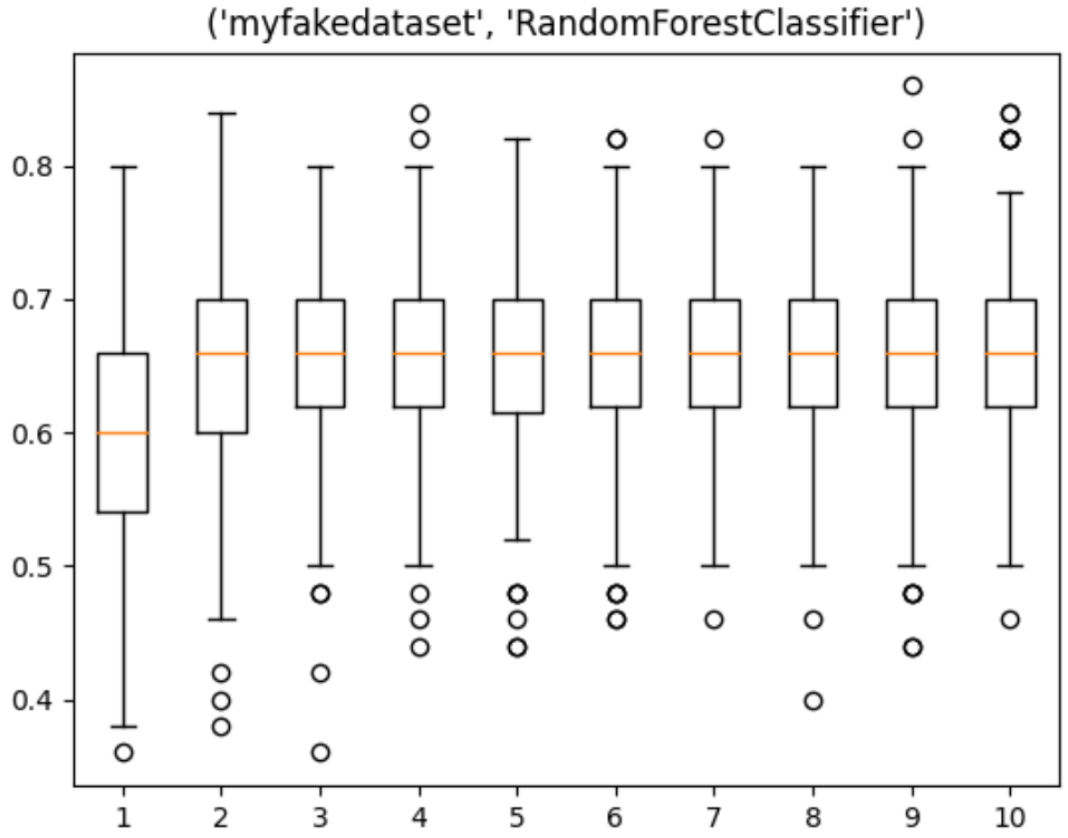
Steel_P dataset:




```
RandomForestClassifier for steel_P dataset:
The best average value of steel_P dataset: is 0.986306213525575 2
The highest value for the control parameter is 10
```

From the above boxplot, the performance is good. We can see as the max depth increases the performance will increase when max depth values are small(less than 6). The performance between max depth = 6 to max depth =10 are similar . When value = 10, the performance is the best as it has the best average accuracy.The value= 1 situation has the worst performance as they have lowest accuracy.

myfakedata:

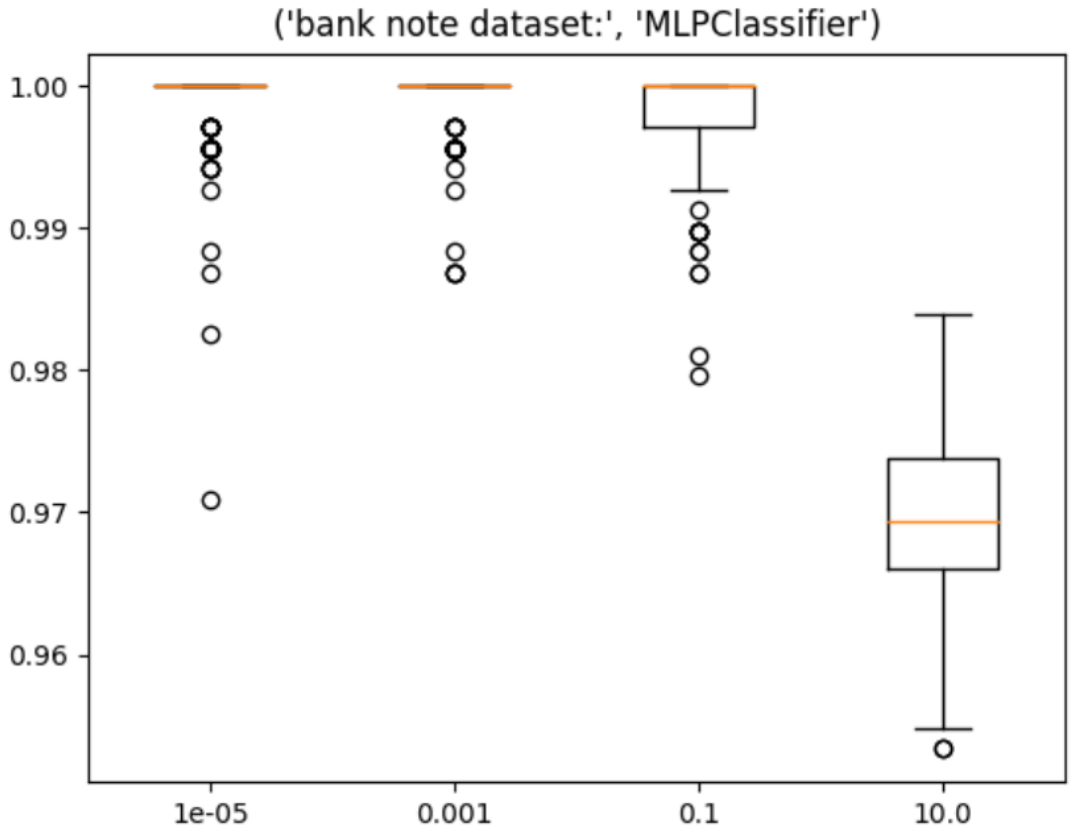



```
RandomForestClassifier for myfakedataset
The best average value of myfakedataset is 0.6627333333333333 2
The highest value for the control parameter is 10
```

From the above boxplot, the performance for the fake data is bad with accuracies for 10 values are between 40 to 80 percent which means the performance is also unstable. The max depth value with 1 performs worse compared to others.

MLPClassifier:

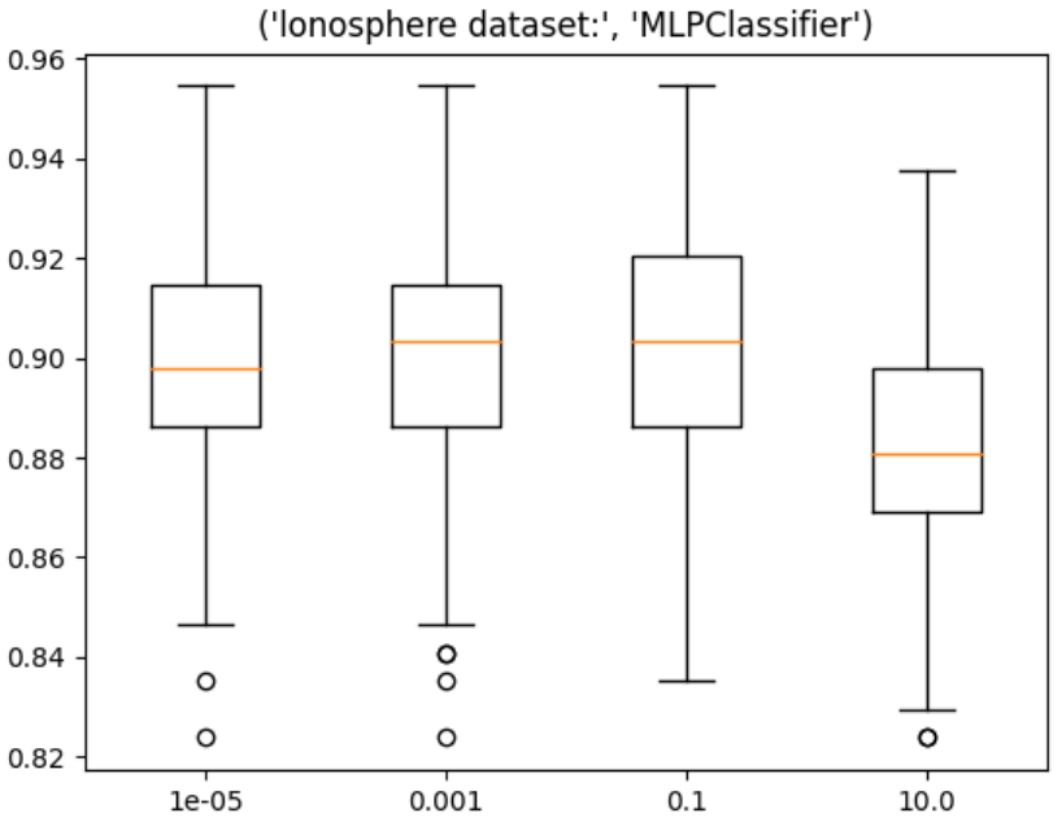
Bank Note dataset:




```
MLPClassifier of bank note dataset:
The best average value of bank note dataset: is 0.9992857142857143
The highest value for the control parameter is 0.001
```

From the above boxplot, the performance is good. We can see as when controller parameters are small the performance is clearly better . When value = 0.001, the performance is the best as it has the best average accuracy. When value = 1e-5 the performance is also good but it has more outlinears compared to 0.001. The value= 10.0 situation has the worst performance as they have lowest accuracy.

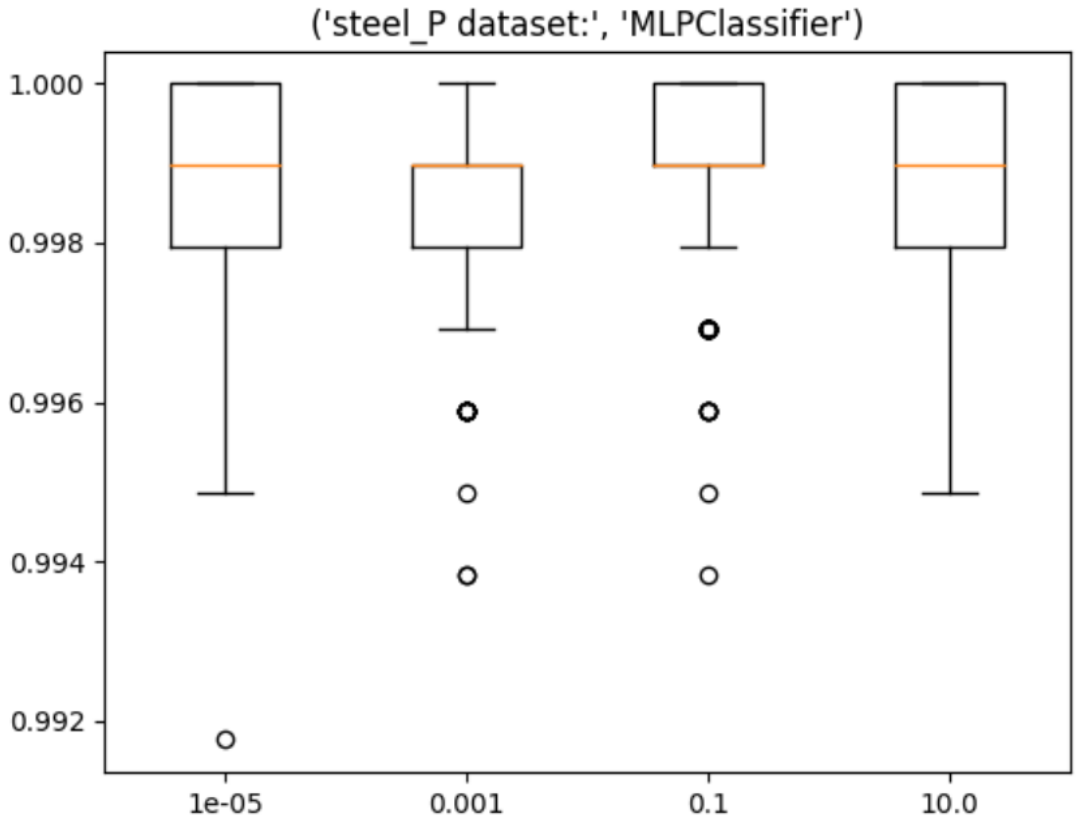
Lonosphere dataset:




```
MLPClassifier of lonosphere dataset:
The best average value of lonosphere dataset: is 0.9013257575757576
The highest value for the control parameter is 0.1
```

From the above boxplot, the performance is good. We can see when controller parameters differences are not obvious . When value = 0.01, the performance is the best as it has the best average accuracy. When value = 1e-5 and 0.001 the performance is also similar. The value= 10.0 situation has the worst performance as they have lowest accuracy.

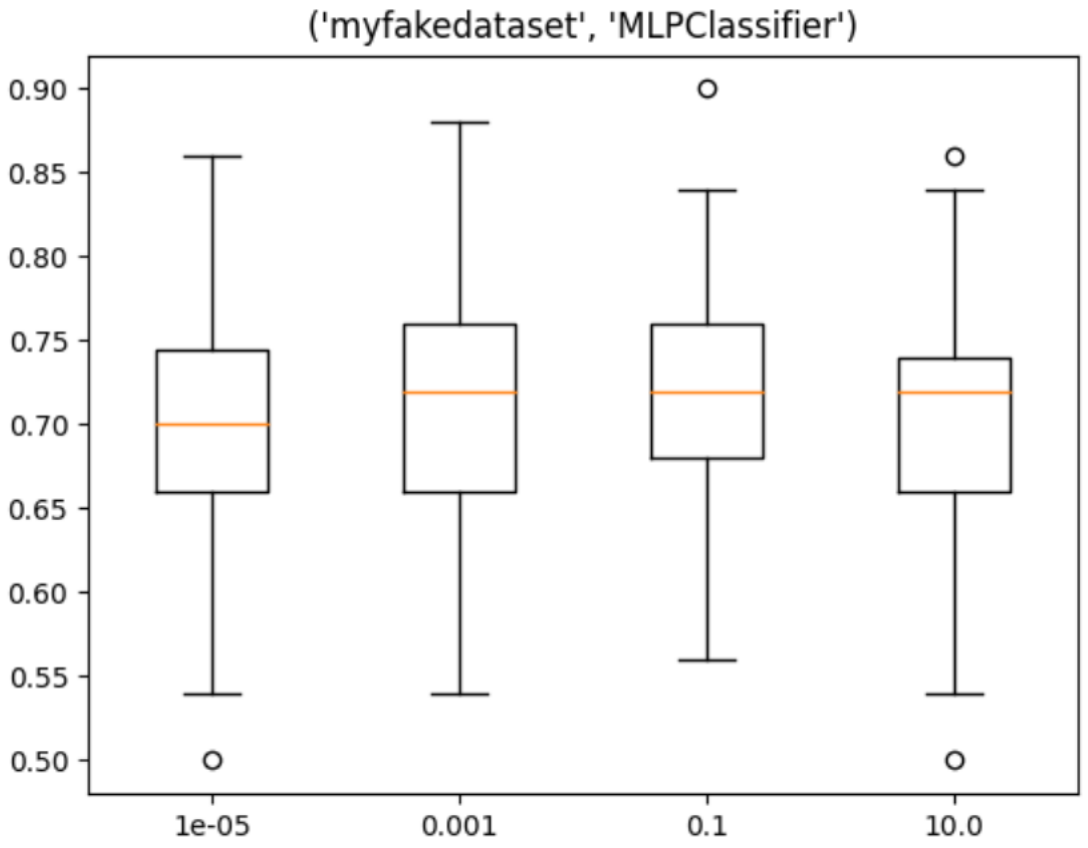
Steel_P dataset:




```
MLPClassifier of steel_P dataset:
The best average value of steel_P dataset: is 0.998918640576725
The highest value for the control parameter is 10.0
```

From the above boxplot, the performance is good. We can see when controller parameters differences are not obvious except for when value = 0.001 and 0.1 there are more outlinears.

myfakedata:




```
MLPClassifier of myfakedataset
The best average value of myfakedataset is 0.7161333333333333
The highest value for the control parameter is 0.1
```

From the above boxplot, the performance for the fake data is bad with accuracies for 4 values are between 50 to 85 percent which means the performance is also unstable.

Table I with the best average value :

	Bank Note dataset	**Lonosphere dataset**	**Steel_P dataset**	**myfakedata**
KNeighborsClassifier	0.9984	0.87858	0.9814	0.63022
GaussianNB	0.83977	0.88475	0.9973	0.6024
LogisticRegression	0.9847	0.8760	0.9997	0.6676
DecisionTreeClassifier	0.97824	0.88331	1.0	0.70106

GradientBoostingClassifier	0.99014	0.92115	1.0	0.7116
RandomForestClassifier	0.9907	0.9304	0.9863	0.66273
MLPClassifier	0.99928	0.90132	0.9989	0.7161

Table II with the best value for the control parameter

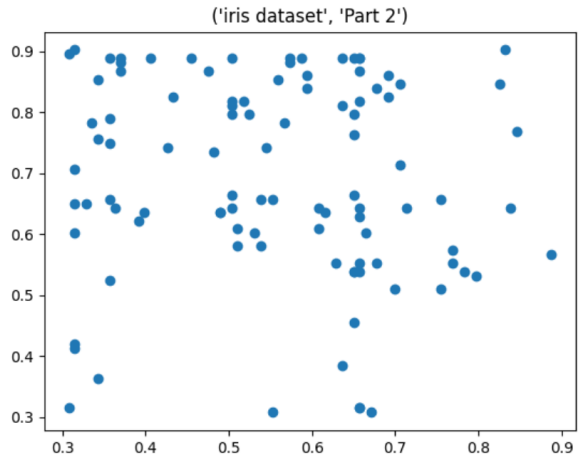
	Bank Note dataset	**Lonosphere dataset**	**Steel_P dataset**	**myfakedata**
KNeighborsClassifier	2	2	1	5
GaussianNB	1e-09	1e-05	1e-09	1e-01
LogisticRegression	5.0	0.5	5.0	0.5
DecisionTreeClassifier	9	2	6,7,8,9,10	1
GradientBoostingClassifier	3	2	1,2,3,4,5,6,7,8,9,10	2
RandomForestClassifier	9	10	10	10
MLPClassifier	0.001	0.1	1e-5,0.001,0.1,10	0.1

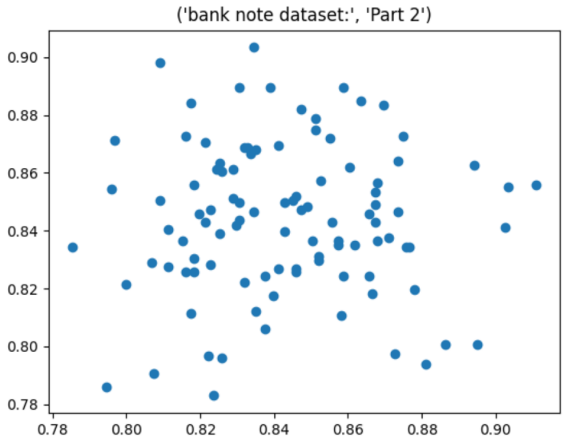
Overall Result Summary && Unexpected Performances:

For all models applied to datasets, most of the models achieved a pretty good average accuracy rate except when models are training myfakedata by looking at table 1. This is understandable because fake datas is randomly generated by numpy.random so there are

less connections between features and targets and there are less similarities when comparing and training features.
Comparing each classifier, the best average accuracies are similar but the actual performances are different.
For K Neighbors Classifier, Out of four datasets , two of the datasets' best value for control parameters is 2 but we cannot but we cannot give a conclusion that value 2 is better than others as differences of the control parameters are not very obvious, the table 2 also shows that. But when the control parameter value is equal to 5, there are obviously more outlinears as it indicates when value = 5 , the performance is less stable.
For GaussianNB Classifiers, the differences of the control parameters are not obvious at all, basically all performances for different values are similar from the box plot and tables, so we may give a conclusion that the "var_smoothing" controller may not influence the result that much. One unexpected point is that in the steel_P dataset box plot, when var_smoothing = 0.1, the difference between the first quartile and the third quartile is very large. This is worth mentioning because it may indicate that there might be potential troubles when var_smoothing =0.1.
For Logistic Regression, except for the banknote dataset,the differences of the control parameters are not very obvious as well. But for the banknote dataset, the performance seems to be increased when the C parameter increases.
For Decision Tree Classifier , when the max depth value is small if the max-depth increase the performance will also increase but when max-depth is big enough, the influence of the max-depth parameter will decrease.This is why when max-depth value between 5 and 10, the performances are similar.This can be seen by looking at table 2, all the best values are large numbers except for Lonosphere dataset but if we look the box plot carefully the performance between 2 and 10 are similar but when max depth is 1, the performance is bad so it doesn’t violate max-depth value should be larger conclusion.
For the Gradient Boosting Classifier, it is like the other way round of decision tree classifiers, the performance seems to decrease when max-depth increases. But this is also not absolute, because there are many unexpected things when using gradient boosting classifiers. The first thing is that in the banknote dataset boxplot, the performance is extremely low when value =1 which violates the performance decreases when max-depth increases rule. The second unexpected point is that in the steel_P dataset , the accuracy rate reaches 100 percent for all different parameter values and I think this may be because the classifier fits the dataset very well.
For the Random Forest Classifier, its performance is very likely to the Decision Tree classifier as when the max depth value is below five if the max-depth increase the performance will also increase but when max-depth is big enough, the influence of the max-depth parameter will decrease. This can be seen by looking at table 2, all the best values are large numbers.
For MLPClassfier, the performance seems to be better when the alpha value is small , when the alpha value is equal to 10 , the performance will decrease immediately. This can be seen by looking at table 2, all the best values are small numbers.

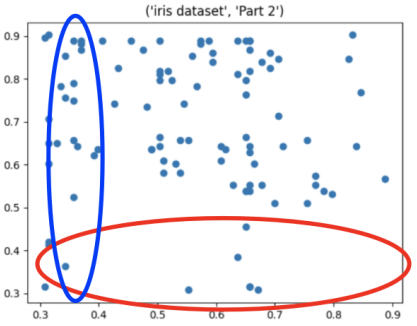
Part two: clustering



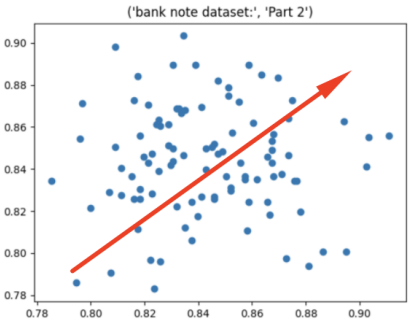


write a paragraph explaining whether they are the same or different, and why

For the two datasets , they have all been randomly split by 100 times and x axis indicates the performance without clustered column and y axis indicates the performance with clustered column.
Comparing this to the scatter plot, we can tell they are different.
First of all, let's have a look at the iris dataset. If we look at the graph on the right hand side, we can see there are a lot of points in the blue circle and only few points in the red circle. Overally, there are more points above in the diagonal than below it. This indicates that clustered columns included have better performances than without cluster columns included.



Then we look at the bank note data set, we can see there is a going up trend and the amount of points above and below in the diagonal are pretty much equal. This means the clustered performance and the non-clustered performance are similar. So the clustering doesn't have much influence when dealing with bank note dataset and it can be caused by the k value isn't quite right for this dataset.



Overally, they are different.