Part1) In part1 both for information gain and gain ratio criterion, the accuracy results in %100, I think this is happening because the dataset is small. I print the tree paths that is followed in the execution that shows:

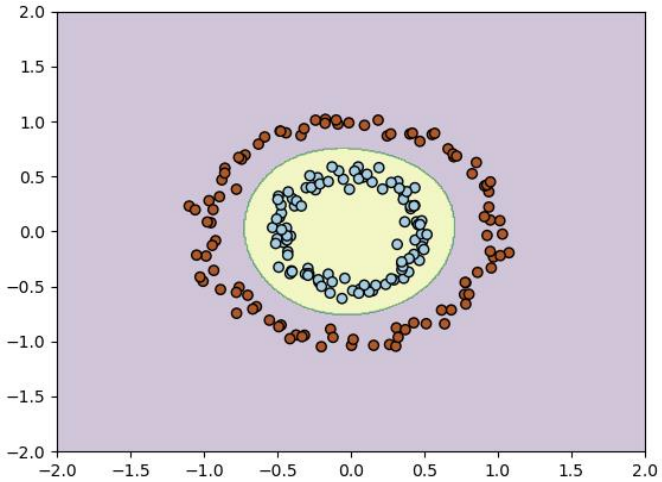
```
['hot', 'sunny', 'high', 'false']
Outlook
{'sunny': <ID3.TreeNode object at 0x000001D641B5AA90>, 'rain': <ID3.TreeNode object at 0x000001D641B5AAF0>, 'overcast': <ID3.TreeLeafNode object at 0x000001D641B5AB50>}
```

The first row is the current sample, the second row is the attribute that is selected for that node and third row is the corresponding nodes according to the possible values of that attribute.

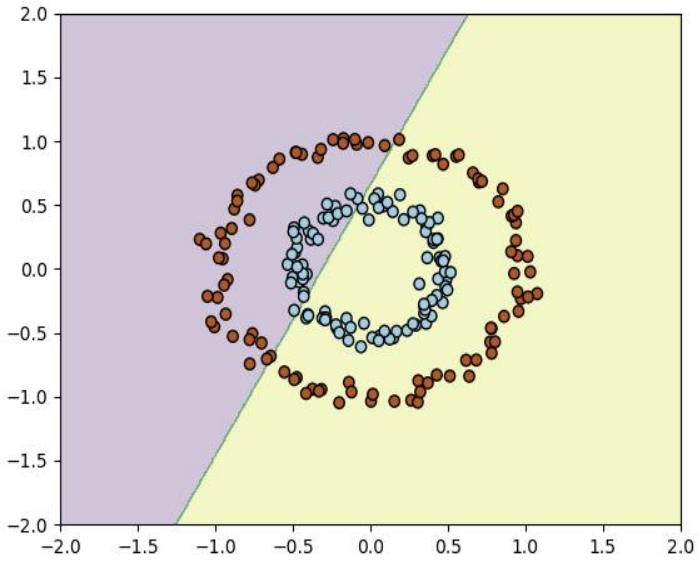
Part2)

Dataset1:

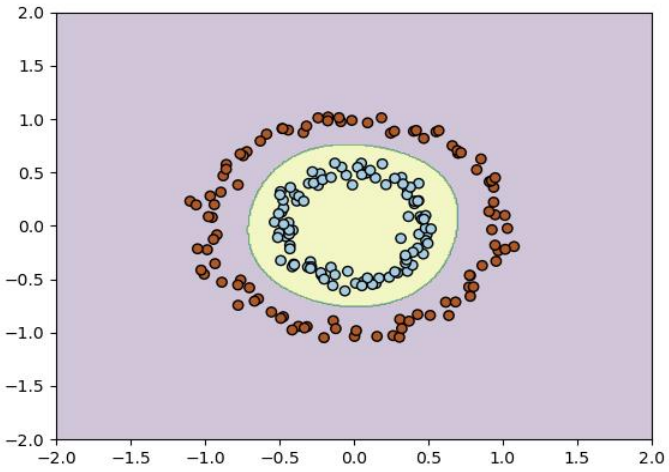
Hyperparameter Conf = C: 1 kernel: rbf



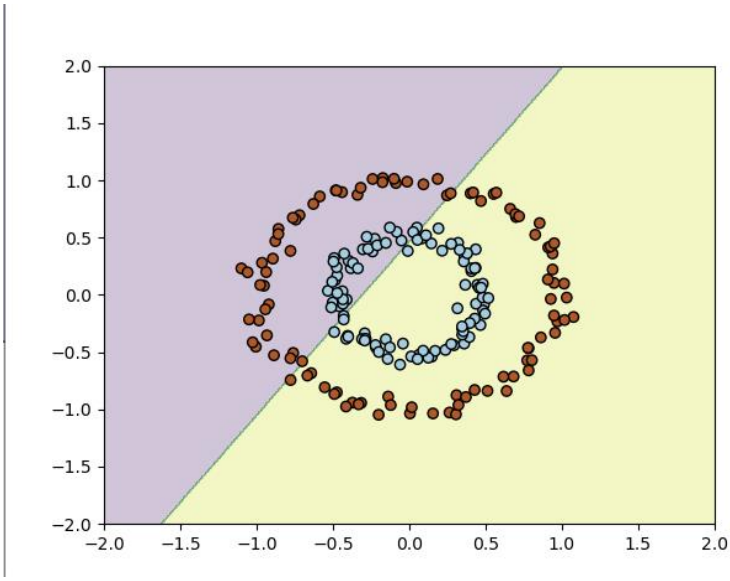
Hyperparameter Conf = C: 1 kernel: linear



Hyperparameter Conf = C: 10 kernel: rbf



Hyperparameter Conf = C: 10 kernel: linear



Dataset2)

PARAMETER CONFIGURATION	MEAN SCORES	STD SCORES	CONFIDENCE INTERVAL
'svm__C': 1, 'svm__kernel': 'rbf'	0.91866667	0.05856051	[0.9092950332503998, 0.9280383000829334]
'svm__C': 1, 'svm__kernel': 'sigmoid'	0.81866667	0.08436956	[0.8051647254509527, 0.8321686078823806]
'svm__C': 10, 'svm__kernel': 'rbf'	0.948	0.05702241	[0.9388745134112486, 0.9571254865887513]
'svm__C': 10, 'svm__kernel': ‘sigmoid’	0.77866667	0.08266667	[0.7654372447314216, 0.7918960886019116]

```
Best Parameter :  {'svm__C': 10, 'svm__kernel': 'rbf'}
97.33333333333334
```

Best Hypermarater configuration is 'svm__C': 10, 'svm__kernel': 'rbf' with a accuracy score 97.3.

PART3)

```
KNN MEAN F1 SCORE 0.684796772820725
KNN MEAN ACCURACY SCORE  0.684796772820725
SVM MEAN F1 SCORE 0.7102030173886462
SVM MEAN ACCURACY SCORE 0.7102030173886462
DEC TREE MEAN F1 SCORE 0.673791755827684
DEC TREE MEAN ACCURACY SCORE 0.673791755827684
RAND FOREST MEAN F1 SCORE 0.7519969070867275
RAND FOREST MEAN ACCURACY SCORE 0.7519969070867275
KNN config:  ('euclidean', 2) Mean test score:  0.6327885385104554 Conf Interval= +- 0.0013902270242902323
KNN config:  ('euclidean', 4) Mean test score:  0.688441402012494 Conf Interval= +- 0.0013573379635035128
SVM config:  (0.1, 'poly') Mean test score:  0.704716118654846 Conf Interval= +- 0.0002069361288921877
SVM config:  (0.1, 'rbf') Mean test score:  0.699996259304979 Conf Interval= +- 3.997870333368013e-05
DEC TREE config:  gini Mean test score:  0.6763549171436054 Conf Interval= +- 0.0014143007606178283
DEC TREE config:  entropy Mean test score:  0.6787794860285042 Conf Interval= +- 0.0014515659237474296
RANDOM FOREST config:  gini Mean test score:  0.7510095836606442 Conf Interval= +- 0.0009581312587039665
RANDOM FOREST config:  entropy Mean test score:  0.7524338308457711 Conf Interval= +- 0.0009810766158541153
```

In the upper part there are the final overall accuracy and f1 scores of the best hyperparameter configuration for each model. Below there are the mean test and confidence interval scores for each hyperparameter configuration for each model.

Decision Tree Feature Importance:

1 0.1521110597401312

2 0.09345603517985202

3 0.0490406643072764

4 0.07535286552679687

5 0.1805762902958859

6 0.036962787847838685

7 0.04379640180295388

8 0.024223320107681575

9 0.024979488512885954

10 0.034591708133214584

11 0.04210263833245199

12 0.04019984899314354

13 0.09474942196139756

14 0.02314447474076234

15 0.013457177742892025

16 0.01965095023799597

17 0.025461271532700093

18 0.015032483893028099

19 0.007936507936507933

20 0.0031746031746031733

The 1st and 5th attribute seems to be the most important attributes which are “Status of existing checking account” and “Credit amount” respectively.

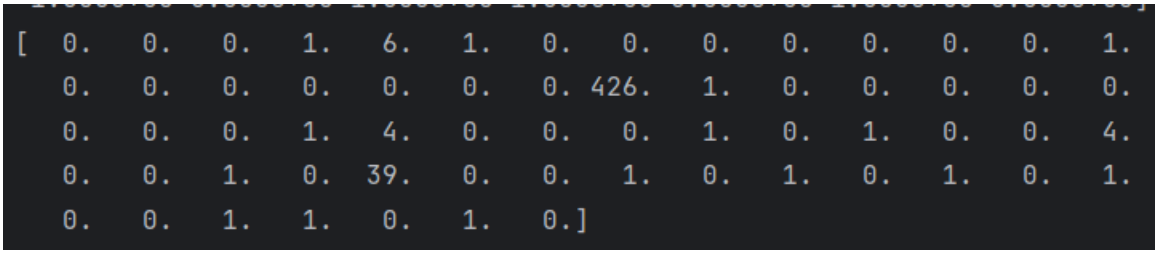
SVM support vectors :

There are a total of 294 + 312 support vectors so it is impossible to fit them here but I deep dived and looked at them in the terminal and see a similarity beetween them. Also there are 63 feature names encountered in fitting. Kernel is “rbf”.

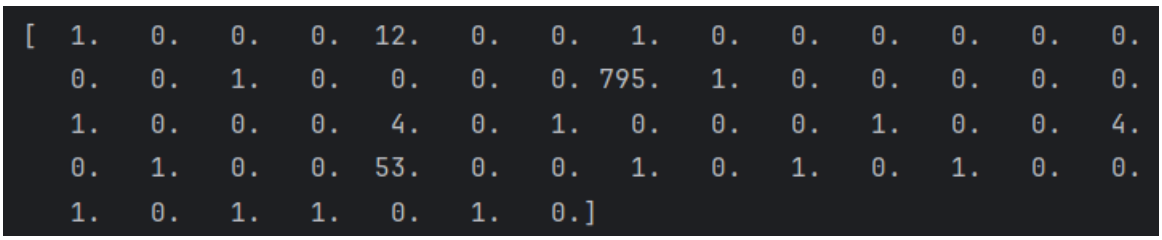
```
Support Vectors for Positive Class:
[[0. 0. 0. ... 0. 1. 0.]
 [1. 0. 0. ... 0. 1. 0.]
 [0. 0. 0. ... 1. 1. 0.]
 ...
 [1. 0. 0. ... 0. 1. 0.]
 [1. 0. 0. ... 1. 1. 0.]
 [0. 1. 0. ... 0. 1. 0.]]

Support Vectors for Negative Class:
[[0. 1. 0. ... 0. 1. 0.]
 [1. 0. 0. ... 0. 1. 0.]
 [0. 1. 0. ... 0. 1. 0.]
 ...
 [0. 0. 0. ... 1. 1. 0.]
 [1. 0. 0. ... 0. 1. 0.]
 [1. 0. 0. ... 1. 1. 0.]]

 Feature Names seen during fit:
63

# of Positive class Supoort vectors: 294

# of Negative class Supoort vectors: 312
```

Commonalities:

Positive class Support Vector example:



Negative class Support Vector example:



In some support vectors for both classes, the 22th value is very high and this result repeats frequently in the support vector results.