DS 203: Programming for Data Science - Exercise 4

Swayam Saroj Patel (22B1816)

February 28, 2025

Utility Function Descriptions:

`sample(center: tuple = (0, 0), r1: float = 0, r2: float = 1, num_points: int = 100, pos=0)`

Purpose:
Generates a set of 2D points uniformly distributed within an annulus (ring-shaped region) centered at a given point.

How It Works:

- **Uniform Area Sampling:**
  The function samples values uniformly from $[r_1^2, r_2^2]$ and then takes the square root to obtain the radii. This ensures uniform distribution over the annulus area.
- **Angle Determination:**
  Based on the parameter `pos`, the angle θ is sampled:
  - `pos = -1`: θ is sampled from $[0, \pi]$ (points above the x-axis).
  - `pos = 0`: θ is sampled from $[0, 2\pi]$ (points all around the circle).
  - `pos = 1`: θ is sampled from $[\pi, 2\pi]$ (points below the x-axis).
- **Coordinate Calculation:**
  Converts the polar coordinates (r, θ) to Cartesian coordinates using:
  $$x = r\cos(\theta), \quad y = r\sin(\theta)$$
  and then shifts them by the given `center`.

Returns:
A tuple containing two arrays: the x-coordinates and the y-coordinates of the generated points.

`circles_dataset(n_samples=1000)`

Purpose:
Generates a multi-class dataset with three groups of points that form circular patterns, useful for multi-class classification tasks.

How It Works:

- Three groups of points are generated using the `sample` function:
  1. **Group 1:** Points from an inner circle (radius range 0 to 1.15) labeled as class 0.
  2. **Group 2:** Points from an annular region (radius range 1 to 1.6) with `pos = 1` (below the x-axis) labeled as class 1.
  3. **Group 3:** Points from the same annular region but with `pos = -1` (above the x-axis) labeled as class 2.
- The x and y coordinates from all groups are concatenated and reshaped to form the feature matrix `X`.
- The labels from all groups are combined into the label vector `Y`.

Returns:
A tuple (X, Y) where `X` is a 2D array of point coordinates and `Y` is an array of corresponding class labels.

`moons_dataset(n_samples=1000)`

Purpose:
Generates the two-class "moons" dataset, commonly used to evaluate classification algorithms on non-linearly separable data.

How It Works:

- Internally calls `make_moons` from scikit-learn with a specified noise level (5e-2) to add randomness.

Returns:
The features and labels for the moons dataset.

`visualize(X, Y, size=5)`

Purpose:
Visualizes a 2D dataset by creating a scatter plot, with colors representing different classes.

How It Works:

- Defines a color mapping for the classes (e.g., 0: red, 1: blue, 2: green).
- Uses Matplotlib to plot the points with the specified colors.

Displays:
A scatter plot of the dataset using the provided figure size.

`make_meshgrid(x, y, h=.02)`

Purpose:
Generates a mesh grid spanning the range of the given x and y values, which is used for plotting decision boundaries.

How It Works:

- Calculates the minimum and maximum values for both x and y, extending them by 1 unit.
- Creates arrays using `np.arange` with a step size `h` and then uses `np.meshgrid` to generate the grid.

Returns:
The mesh grid arrays `xx` and `yy`.

`plot_contours(ax, clf, xx, yy, poly=None, ...)`

Purpose:
Plots the decision boundary of a classifier as filled contours over a mesh grid.

How It Works:

- Combines the mesh grid coordinates into a single array of input features.
- If a polynomial transformer (`poly`) is provided, applies the transformation to the grid inputs.
- Uses the classifier (`clf`) to predict classes for each point in the grid.
- Reshapes the predictions to match the grid and plots the decision regions using `ax.contourf`.

Returns:
The contour plot object.

`plot_decision_boundary(model, plot_title, size=8, poly=None)`

Purpose:
Creates a comprehensive plot displaying the decision boundary of a trained model along with the original dataset points.

How It Works:

- Sets up a figure and axis with a specified size.

- Extracts the x and y coordinates from the global dataset `X` and uses `make_meshgrid` to generate a mesh grid.
- Calls `plot_contours` to plot the decision boundary. If a polynomial transformer is provided, it applies the transformation before prediction.
- Overlays the original data points (from `X` and `Y`) using a scatter plot.
- Configures plot labels, title, and legend.

Displays:
The final plot with decision boundaries and data points.

2 - class classifications:

Decision Boundaries:

Logistic regrsssion:

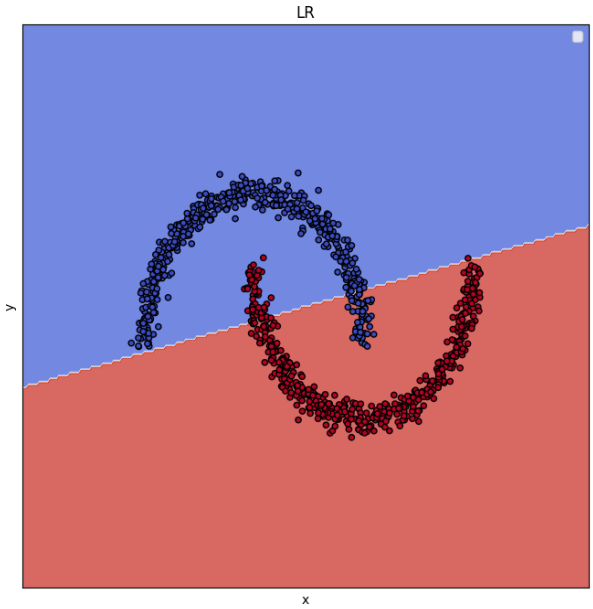

Figure 1: Logistic Regression

SVC – linear:

The **Linear Kernel**: This kernel doesn't transform the data—it attempts to find a straight-line (or, in higher dimensions, a hyperplane) that separates the classes. This kernel is ideal when the data is already linearly separable or nearly so.

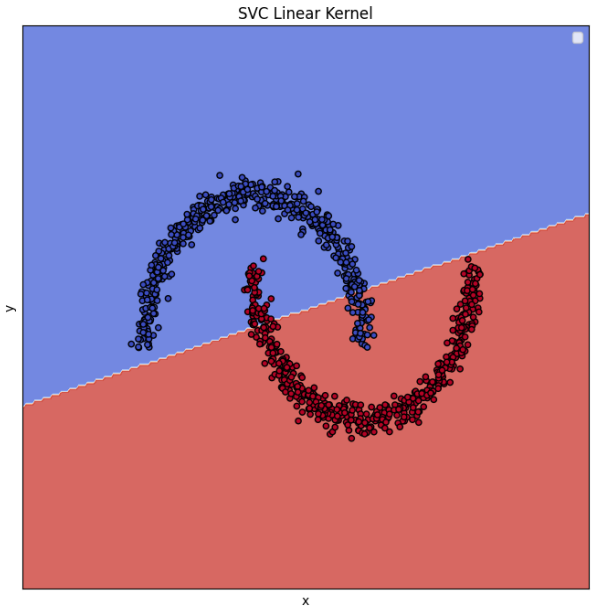

Figure 2: SVC Linear

SVC - rbf:

The **RBF Kernel**: Also known as the Gaussian kernel, maps the data into a higher-dimensional space using a non-linear transformation. This allows the classifier to find a non-linear decision boundary, which is particularly useful when the classes are not linearly separable in the original feature space.

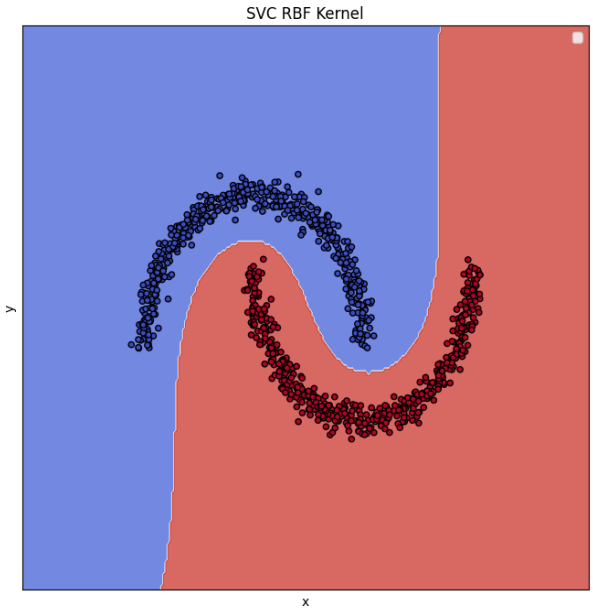

Figure 3: SVC rbf

Random Forest Classifier:

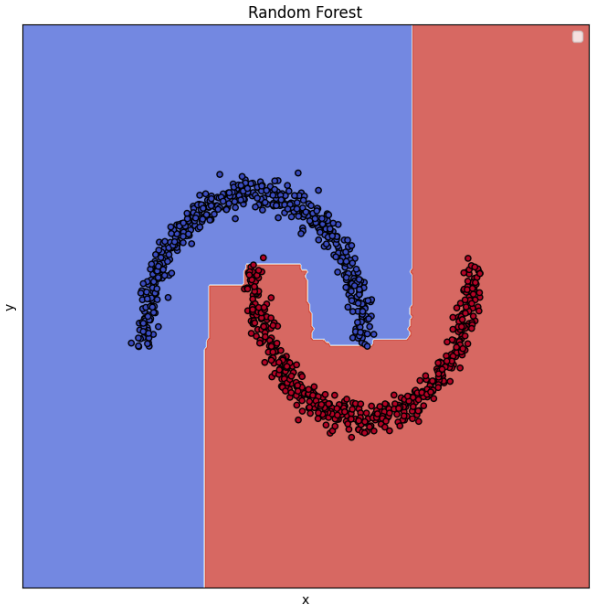

Figure 4: Random Forest Classifier

Neural Network Classifier – with hidden_layer_sizes=(5)

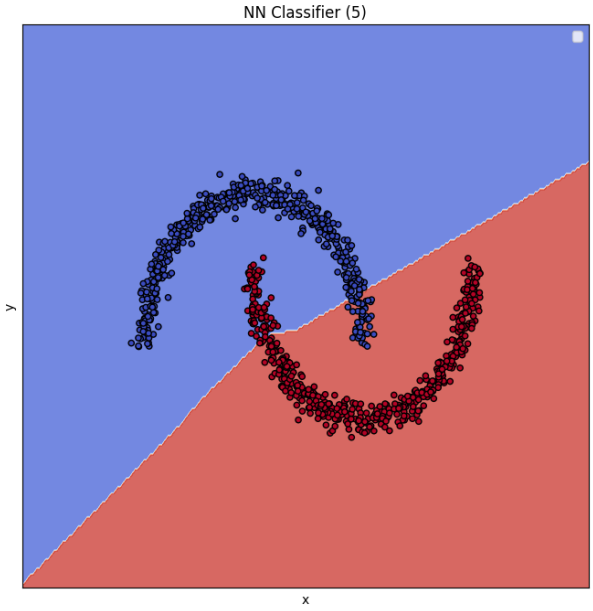

Figure 5: Neural Network Classifier – with hidden_layer_sizes=(5)

Neural Network Classifier – with hidden_layer_sizes=(5,5)

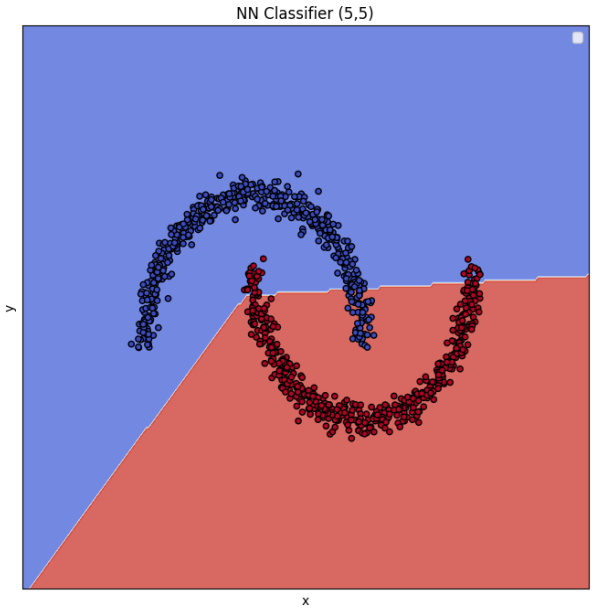

Figure 6: Neural Network Classifier – with hidden_layer_sizes=(5,5)

Metrics:

Below are the confusion matrices for each model:

Logistic Regression

$$\begin{bmatrix} 447 & 53 \\ 54 & 446 \end{bmatrix}$$

SVC (Linear Kernel)

$$\begin{bmatrix} 444 & 56 \\ 55 & 445 \end{bmatrix}$$

SVC (RBF Kernel)

$$\begin{bmatrix} 500 & 0 \\ 0 & 500 \end{bmatrix}$$

Random Forest

$$\begin{bmatrix} 500 & 0 \\ 0 & 500 \end{bmatrix}$$

NN Classifier (5)

$$\begin{bmatrix} 446 & 54 \\ 59 & 441 \end{bmatrix}$$

NN Classifier (5,5)

$$\begin{bmatrix} 446 & 54 \\ 32 & 468 \end{bmatrix}$$

Summary of Classification Metrics

The table below summarizes the accuracy, precision, recall, and F1 score for each model:

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.893	0.8938	0.892	0.8929
SVC (Linear Kernel)	0.889	0.8882	0.890	0.8891
SVC (RBF Kernel)	1.000	1.0000	1.000	1.0000
Random Forest	1.000	1.0000	1.000	1.0000
NN Classifier (5)	0.887	0.8909	0.882	0.8864
NN Classifier (5,5)	0.914	0.8966	0.936	0.9159

Table 1: Summary of Classification Metrics for Different Models

Logistic Regression

Confusion Matrix:

$$\begin{bmatrix} 447 & 53 \\ 54 & 446 \end{bmatrix}$$

Metrics: Accuracy = 0.893, Precision = 0.8938, Recall = 0.892, F1 Score = 0.8929.
Interpretation: Logistic Regression achieves an accuracy of approximately 89.3%. The balanced precision and recall indicate that the model performs well in both identifying true positives and avoiding false positives. This is typical for a linear classifier when the data is nearly linearly separable.

SVC (Linear Kernel)

Confusion Matrix:

$$\begin{bmatrix} 444 & 56 \\ 55 & 445 \end{bmatrix}$$

Metrics: Accuracy = 0.889, Precision = 0.8882, Recall = 0.890, F1 Score = 0.8891.
Interpretation: The SVC with a linear kernel performs similarly to Logistic Regression, with slightly lower accuracy. This suggests that while a linear decision boundary is mostly effective, some points near the boundary are misclassified.

SVC (RBF Kernel)

Confusion Matrix:

$$\begin{bmatrix} 500 & 0 \\ 0 & 500 \end{bmatrix}$$

Metrics: Accuracy = 1.000, Precision = 1.0000, Recall = 1.000, F1 Score = 1.0000.
Interpretation: The SVC using an RBF kernel achieves perfect scores, indicating that the non-linear transformation perfectly captures the underlying data distribution, leading to zero misclassifications.

Random Forest

Confusion Matrix:

$$\begin{bmatrix} 500 & 0 \\ 0 & 500 \end{bmatrix}$$

Metrics: Accuracy = 1.000, Precision = 1.0000, Recall = 1.000, F1 Score = 1.0000.
Interpretation: Random Forest also achieves perfect classification. Its ensemble approach effectively models complex patterns in the data, leading to flawless performance on the test set.

NN Classifier (5)

Confusion Matrix:

$$\begin{bmatrix} 446 & 54 \\ 59 & 441 \end{bmatrix}$$

Metrics: Accuracy = 0.887, Precision = 0.8909, Recall = 0.882, F1 Score = 0.8864.
Interpretation: The Neural Network with one hidden layer (5 neurons) shows an accuracy of approximately 88.7%. Its performance is slightly lower than the linear models, which may indicate that the network's capacity is limited, leading to a few more misclassifications.

NN Classifier (5,5)

Confusion Matrix:

$$\begin{bmatrix} 446 & 54 \\ 32 & 468 \end{bmatrix}$$

Metrics: Accuracy = 0.914, Precision = 0.8966, Recall = 0.936, F1 Score = 0.9159.
Interpretation: The deeper Neural Network with two hidden layers (5 neurons each) improves the performance, achieving an accuracy of approximately 91.4%. The increase in recall and F1 score indicates that the additional layer helps the model better capture non-linear relationships in the data, reducing false negatives significantly.

Hyper-parameters of SVC - rbf:

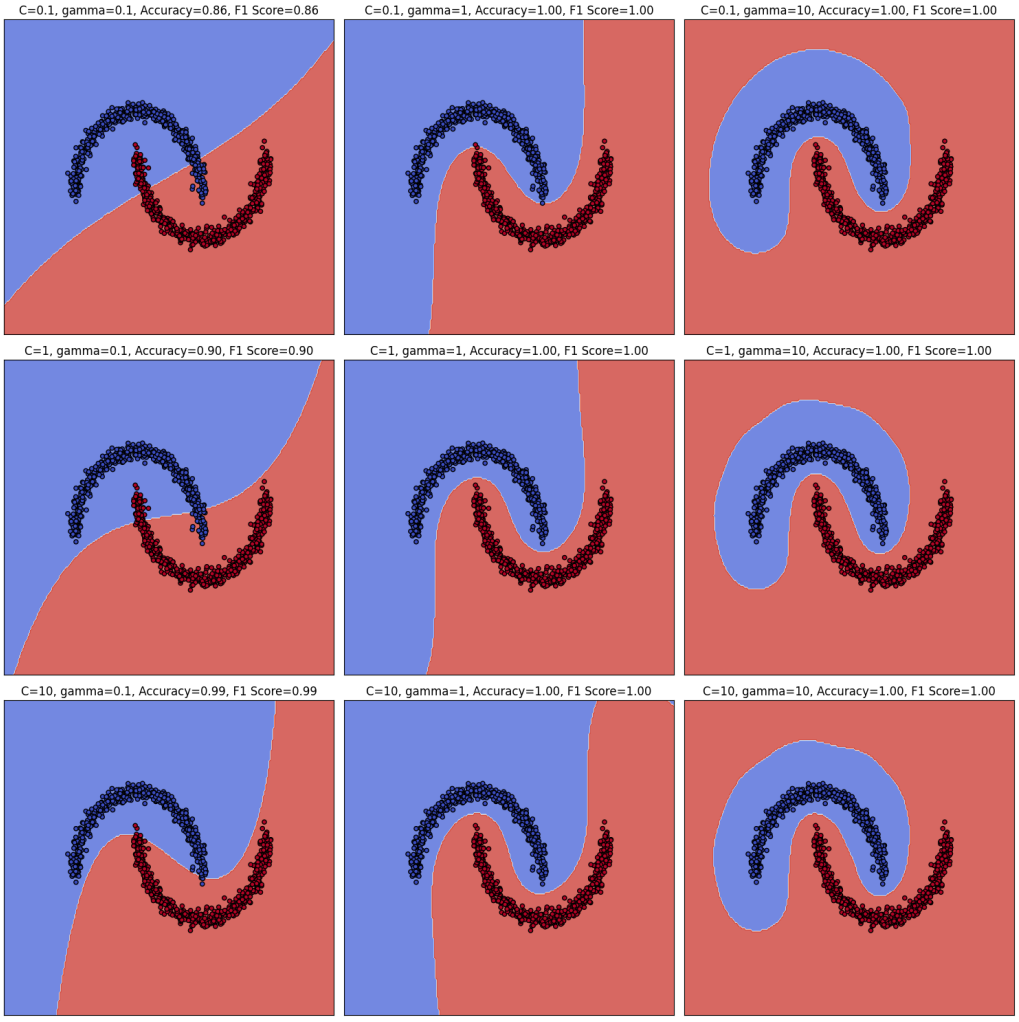

Figure 7: All the boundary graphs

- **Parameter C:** Controls the penalty for misclassification.
  - **Small C**: Places more emphasis on a smoother decision boundary, potentially leading to *underfitting*.
  - **Large C**: Tries to classify all points correctly, potentially leading to a very flexible boundary and *overfitting*.
- **Parameter γ:** Controls the influence of each training example in the RBF kernel.
  - **Small γ**: Each data point has a broad influence, resulting in a smoother boundary and possible *underfitting*.
  - **Large γ**: Each point's influence is very localized, enabling highly curved boundaries and possible *overfitting*.

Observations from the Plots

By examining the nine plots:

- **Underfitting (High Bias):**
  When both C and γ are small (e.g., $C = 0.1, \gamma = 0.1$), the boundary is overly smooth and fails to capture the moons' shape. This yields lower Accuracy/F1 (around 0.86), indicative of underfitting.
- **Better Fit / Possible Overfitting:**
  Increasing C and/or γ leads to more flexible decision boundaries. In many cases (e.g., $C = 0.1, \gamma = 1$ or $C = 1, \gamma = 10$), the model perfectly separates the data, achieving Accuracy and F1 scores of 1.00. Although perfect scores may suggest excellent fit, it can also indicate a risk of overfitting if the boundary is too finely tuned to the training data.

Best Settings Among the Nine Tested

Multiple parameter combinations achieve perfect classification (1.00 Accuracy and 1.00 F1), including:

$$(C = 0.1, \gamma = 1), \quad (C = 0.1, \gamma = 10), \quad (C = 1, \gamma = 1), \quad (C = 1, \gamma = 10), \quad (C = 10, \gamma = 1), \quad (C = 10, \gamma = 10).$$

Since they all yield the same perfect metrics on this dataset, we cannot distinguish them purely by Accuracy/F1. However, a common practice is to choose a simpler (less extreme) model if it already attains perfect scores. Hence, one might prefer a more moderate choice, such as $(C = 1, \gamma = 1)$, which is often less prone to overfitting in general settings.

Overall, the grid search shows how (C, γ) values affect the smoothness or complexity of the decision boundary. Small values can underfit the data, whereas large values can perfectly classify but risk overfitting.

Degree for Logistic Regression:

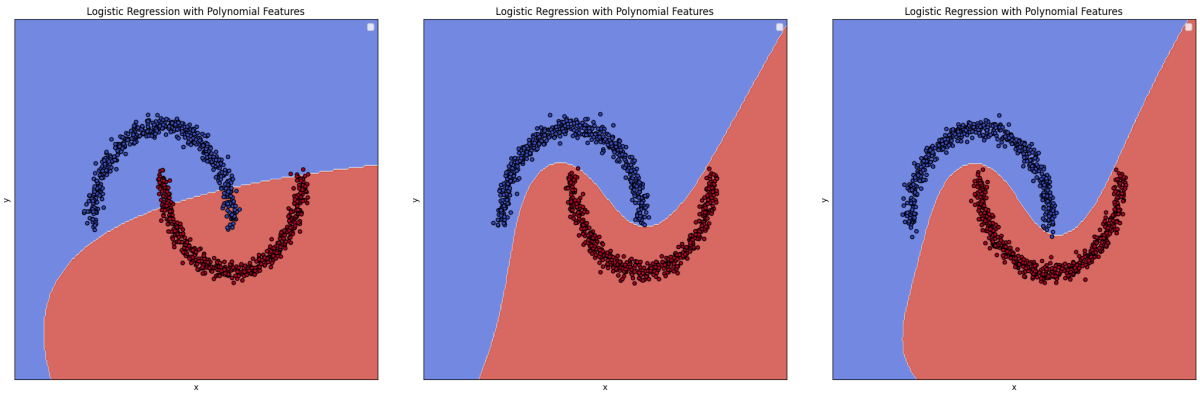

Figure 8: Degree 2 Figure 9: Degree 3 Figure 10: Degree 4

Figure 11: Logistic Regression with various degrees

Degree 2

Performance Metrics:

- Accuracy: 0.881
- Precision: 0.8810
- Recall: 0.881
- F1 Score: 0.8810

Confusion Matrix:

$$\begin{bmatrix} 441 & 59 \\ 60 & 440 \end{bmatrix}$$

Degree 3

Performance Metrics:

- Accuracy: 0.998
- Precision: 0.998
- Recall: 0.998
- F1 Score: 0.998

Confusion Matrix:

$$\begin{bmatrix} 499 & 1 \\ 1 & 499 \end{bmatrix}$$

Degree 4

Performance Metrics:

- Accuracy: 0.999
- Precision: 0.9990
- Recall: 0.999
- F1 Score: 0.9990

Confusion Matrix:

$$\begin{bmatrix} 499 & 1 \\ 0 & 500 \end{bmatrix}$$

The results indicate that when using polynomial features of degree 2, the logistic regression model achieves an accuracy of only 0.881. This suggests that quadratic features are insufficient to capture the non-linear structure present in the dataset.

In contrast, with polynomial features of degree 3, the model's performance dramatically improves, reaching an accuracy of 0.998. The confusion matrix shows almost perfect separation of the classes, with only a single misclassification in each class. A further increase to degree 4 yields only marginal gains (accuracy of 0.999), implying that the cubic features already capture the necessary complexity.

Thus, **degree 3** is identified as the minimum degree required to achieve near-perfect classification performance for this dataset. It introduces cubic and cross terms that effectively capture the underlying non-linear patterns, while avoiding the potential overfitting and increased computational cost associated with higher degrees.

Multi - class classification:

Decisions Boundaries:

SVC - Linear:

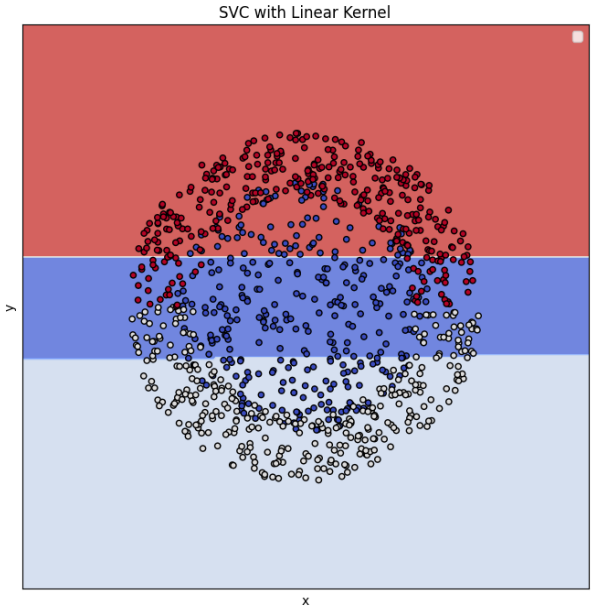

Figure 12: SVC linear

SVC - rbf:

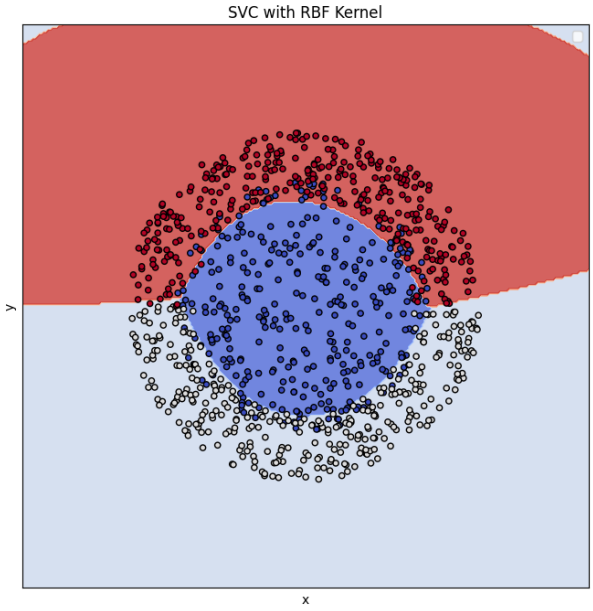

Figure 13: SVC rbf

Decision tree Classifier:

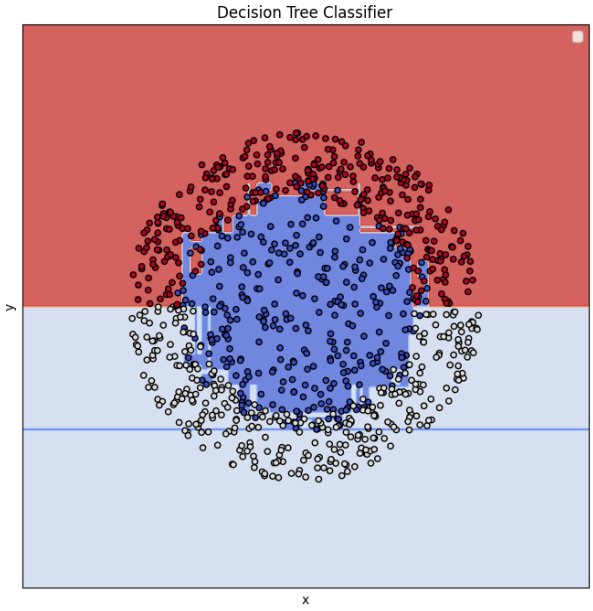

Figure 14: Decision tree

Random Forest Classifier – with min_samples_leaf=1

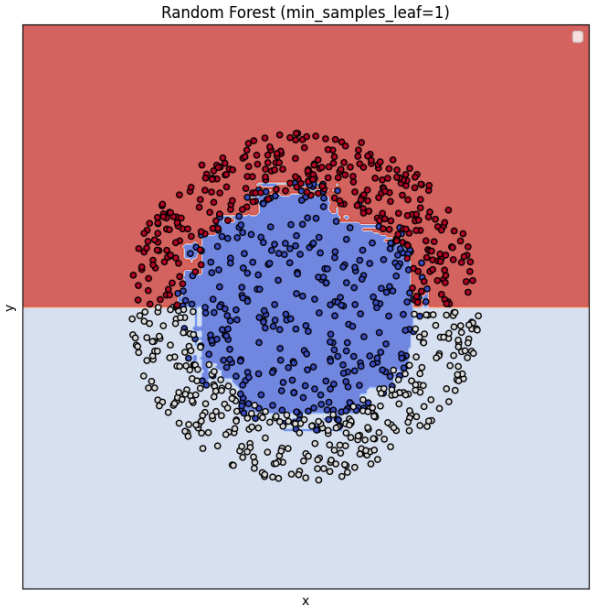

Figure 15: Random Forest Classifier with min_samples_leaf=1

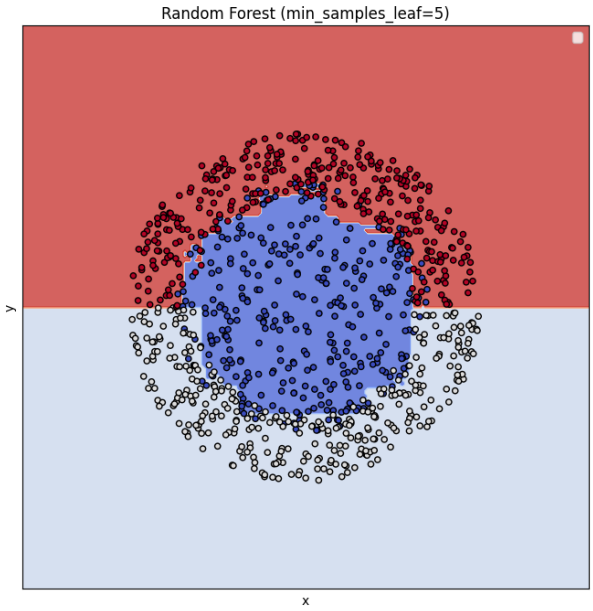

Figure 16: Random Forest Classifier with min_samples_leaf=5

Random Forest Classifier – with min_samples_leaf=5

Neural Network Classifier - with hidden_layer_sizes=(5,5)

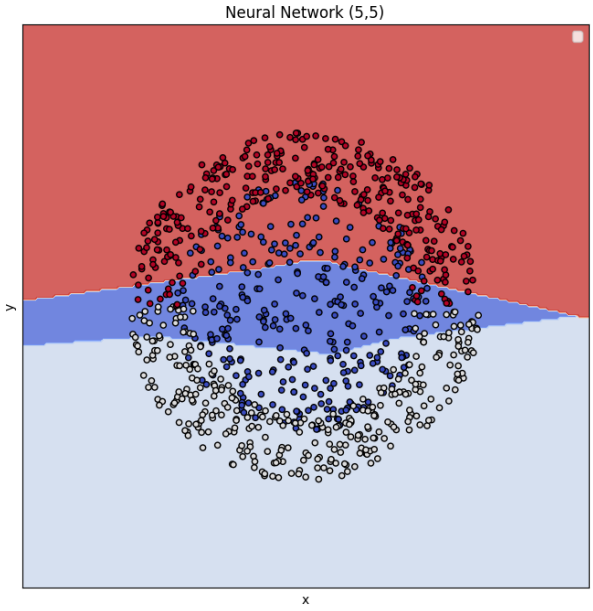

Figure 17: Neural Network Classifier with hidden_layer_sizes=(5,5)

Metrics:

SVC - Linear:

Accuracy: 0.68068

Per-Class Metrics:

Class	Precision	Recall	F1 Score
0	0.522581	0.486486	0.503888
1	0.735211	0.783784	0.758721
2	0.769461	0.771772	0.770615

Averaged Metrics:

Average	Precision	Recall	F1 Score
Micro	0.680681	0.680681	0.680681
Macro	0.675751	0.680681	0.677741
Weighted	0.675751	0.680681	0.677741

SVC - RBF:

Accuracy: 0.90691

Per-Class Metrics:

Class	Precision	Recall	F1 Score
0	0.941176	0.768769	0.846281
1	0.903955	0.960961	0.931587
2	0.884718	0.990991	0.934844

Averaged Metrics:

Average	Precision	Recall	F1 Score
Micro	0.906907	0.906907	0.906907
Macro	0.909950	0.906907	0.904237
Weighted	0.909950	0.906907	0.904237

Decision Tree Classifier:

Accuracy: 1.0

Per-Class Metrics:

Class	Precision	Recall	F1 Score
0	1.0	1.0	1.0
1	1.0	1.0	1.0
2	1.0	1.0	1.0

Averaged Metrics:

Average	Precision	Recall	F1 Score
Micro	1.0	1.0	1.0
Macro	1.0	1.0	1.0
Weighted	1.0	1.0	1.0

Random Forest Classifier (min_samples_leaf=1):

Accuracy: 1.0

Per-Class Metrics:

Class	Precision	Recall	F1 Score
0	1.0	1.0	1.0
1	1.0	1.0	1.0
2	1.0	1.0	1.0

Averaged Metrics:

Average	Precision	Recall	F1 Score
Micro	1.0	1.0	1.0
Macro	1.0	1.0	1.0
Weighted	1.0	1.0	1.0

Random Forest Classifier (min_samples_leaf=5):

Accuracy: 0.94494

Per-Class Metrics:

Class	Precision	Recall	F1 Score
0	0.937107	0.894895	0.915515
1	0.944282	0.966967	0.955490
2	0.952941	0.972973	0.962853

Averaged Metrics:

Average	Precision	Recall	F1 Score
Micro	0.944945	0.944945	0.944945
Macro	0.944777	0.944945	0.944619
Weighted	0.944777	0.944945	0.944619

Neural Network Classifier (5,5):

Accuracy: 0.47848

Per-Class Metrics:

Class	Precision	Recall	F1 Score
0	0.327004	0.465465	0.384139
1	0.615238	0.969970	0.752914
2	0.000000	0.000000	0.000000

Averaged Metrics:

Average	Precision	Recall	F1 Score
Micro	0.478478	0.478478	0.478478
Macro	0.314081	0.478478	0.379018
Weighted	0.314081	0.478478	0.379018

Interpretation:

- **SVC with Linear Kernel:**
  The model achieves an accuracy of 68.07%. The per-class metrics show variability, with class 0 having notably lower precision and recall than classes 1 and 2. The micro, macro, and weighted averages are similar, reflecting moderate overall performance.

- **SVC with RBF Kernel:**
  With an accuracy of 90.69%, the RBF kernel significantly outperforms the linear kernel. It demonstrates high precision and recall across all classes, with slightly lower performance for class 0. The averaged metrics are high, indicating that the non-linear decision boundary is well-suited to the data.

- **Decision Tree Classifier and Random Forest (min_samples_leaf=1):**
  Both models achieve perfect classification with 100% accuracy, precision, recall, and F1 score for every class, indicating that they perfectly capture the underlying structure of the data.

- **Random Forest Classifier (min_samples_leaf=5):**
  This variant achieves slightly lower performance (accuracy of 94.49%) compared to the version with `min_samples_leaf=1`. The performance is strong across all classes, with minor reductions in the metrics, likely due to increased regularization reducing overfitting.

- **Neural Network Classifier (5,5):**
  The neural network with two hidden layers (5,5) performs poorly with an accuracy of 47.85%. In particular, class 2 is completely misclassified (zero precision, recall, and F1 score). The low macro and weighted averages indicate that the model struggles to generalize, possibly due to insufficient network capacity or issues during training.

The SVC with RBF Kernel and the tree-based models (Decision Tree and Random Forest with `min_samples_leaf=1`) show excellent performance with high accuracy and robust per-class metrics. In contrast, the SVC with Linear Kernel exhibits moderate performance, while the Neural Network (5,5) performs poorly. Overall, non-linear models (SVC with RBF and tree-based methods) are much better suited to this dataset, as evidenced by their high classification metrics.

Bonus:

Uniformly sampling points from an annulus is a common task in data generation. A naive approach samples the radius r uniformly between r_1 and r_2, but this does not yield a uniform distribution over the area of the annulus. This is because the area element in polar coordinates is $dA = r\,dr\,d\theta$, and thus larger radii correspond to larger areas. An alternative, *area-corrected* method samples r^2 uniformly and then takes the square root to obtain r. In this document, we describe a methodology to quantitatively compare these two sampling methods.

Methodology:

To quantitatively evaluate the uniformity of the sampled points, we partition the annulus into n concentric rings of equal area. For an annulus with inner radius r_1 and outer radius r_2, the area is proportional to $r_2^2 - r_1^2$. Dividing the interval $[r_1^2, r_2^2]$ into n equal segments yields rings with equal area. The boundaries of these rings are given by:

$$r_k = \sqrt{r_1^2 + k \cdot \Delta}, \quad k = 0, 1, \ldots, n,$$

where $\Delta = \frac{r_2^2 - r_1^2}{n}$. Ideally, each ring should contain approximately $\frac{1}{n}$ (or 0.1 if $n = 10$) of the total number of points if the points are uniformly distributed over the annulus.

We compare two sampling methods:

1. **Naive Sampling:** Sample r uniformly in $[r_1, r_2]$ and θ uniformly in $[0, 2\pi]$.
2. **Area-Corrected Sampling:** Sample u uniformly in $[r_1^2, r_2^2]$ and set $r = \sqrt{u}$; sample θ uniformly in $[0, 2\pi]$.

For each method, we calculate the fraction of points falling into each of the 10 rings and compare these fractions to the ideal value of 0.1.

Results:

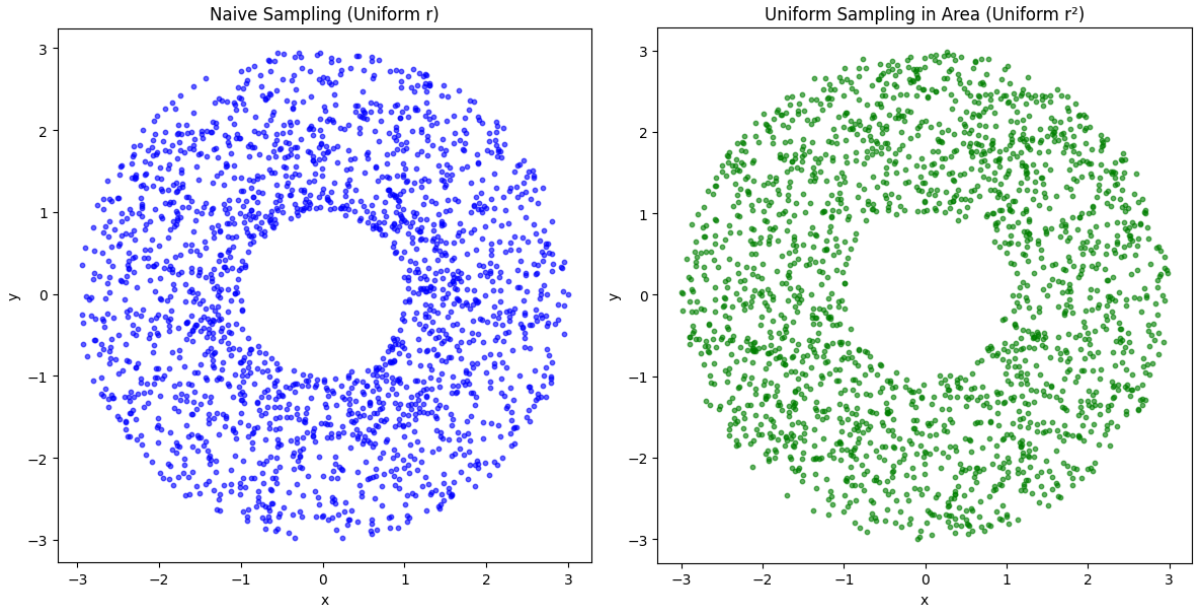

Figure 18: Scatter plot

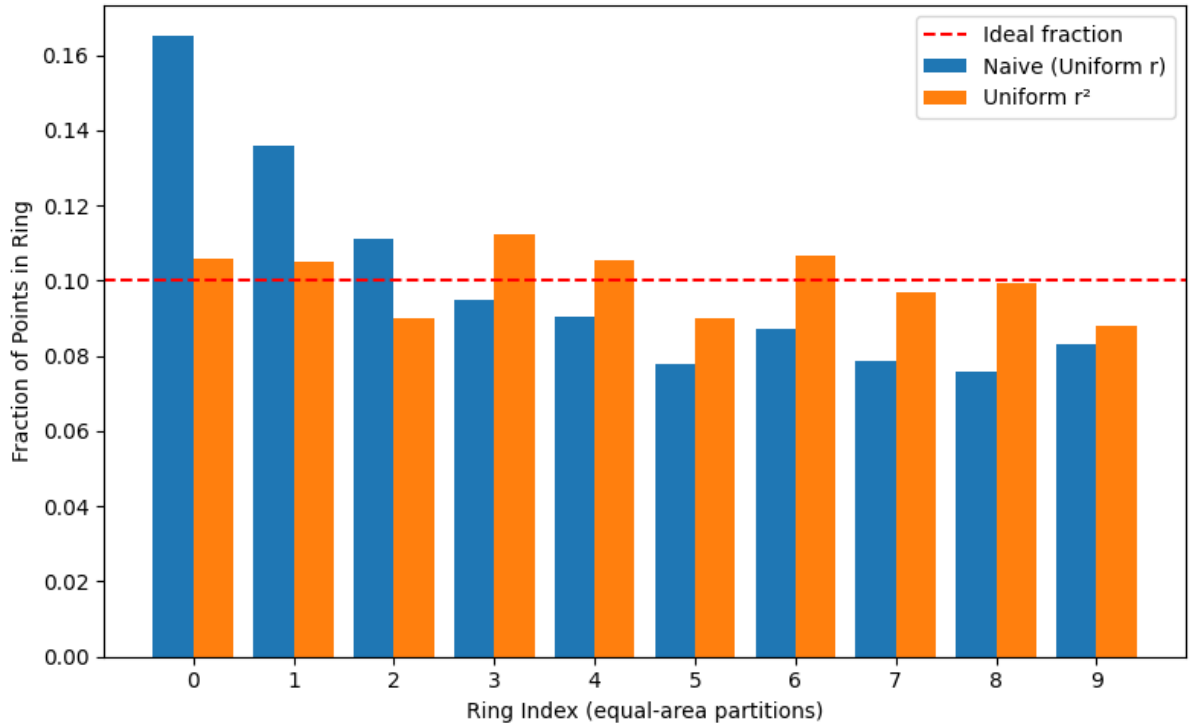

Figure 19: Histogram of the quantized areas

The observed fractions of points per ring for the two methods are as follows:

Via scatter plot Observation:

We can see that in the uninform r sampling the density is more in the inner region than the outer one, roughy suggesting us that the uniform r^2 sampling approach is better in giving uniform outputs.

Naive Sampling (Uniform r)

Ring Fractions: [0.165, 0.136, 0.111, 0.095, 0.0905, 0.078, 0.087, 0.0785, 0.076, 0.083]

Area-Corrected Sampling (Uniform r^2)

Ring Fractions: [0.106, 0.105, 0.09, 0.1125, 0.1055, 0.09, 0.1065, 0.097, 0.0995, 0.088]

Ideal Fraction

For 10 equal-area rings, the ideal fraction per ring is:

$$\frac{1}{10} = 0.1.$$

The naive sampling method, which uniformly samples r, shows significant deviation from the ideal fraction. For example, the first ring contains 16.5% of the points, far exceeding the expected 10%, while other rings are under-represented. This indicates a strong clustering of points near the inner region of the annulus.

In contrast, the area-corrected sampling method (uniform r^2) produces fractions that are much closer to the ideal value of 0.1 across the rings, with only minor variations (ranging from approximately 0.088 to 0.1125). This confirms that sampling uniformly in r^2 yields a more uniform distribution over the area of the annulus.

Thus, by partitioning the annulus into equal-area rings and comparing the fraction of points in each ring, we can quantitatively conclude that the area-corrected sampling method is superior for achieving uniform coverage.

Learning:

- **Classification models and comparing them**: One of the major take away point I would say that I got to know is the correctness of non linear models.
- **Feature Engineering**: Again feature engineering proves its value in making linear models more compatible with complex data and helps in robust execution of the code.
- **Hyperparameter tuning**: This part told me that just selecting the right model in not the end off the road. We can improve it further by tuning its hyperparameters.
- **Random Sampling**: The bonus question should me the importance of how the choice of random function can also bring differences in the study.