thypcme9z

February 19, 2024

1 DS 203 -E4 ASSIGNMENT

2 Siddharth Verma 22B2153

Importing Required Libraries and Data

[30]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from matplotlib.colors import ListedColormap
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score, roc_auc_score, roc_curve
from sklearn.preprocessing import LabelBinarizer
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
from sklearn.metrics import RocCurveDisplay
```

[31]:
```
data1=pd.read_csv(r"clusters-4-v0.csv")
data2=pd.read_csv(r"clusters-4-v1.csv")
data3=pd.read_csv(r"clusters-4-v2.csv")
```

[32]:
```
data1,data2,data3
```

[32]:
```
(            x1         x2  y
 0   -19.205039  15.905880  1
 1    -8.081249  30.322485  1
 2   -21.284312  22.407210  1
 3   -14.661466  25.668522  1
 4    -6.461822  17.406663  1
```

```
…          …         … ..
1435  42.626596 -29.882691  4
1436  43.465376 -23.001958  4
1437  46.358862 -29.557741  4
1438  42.809519 -26.078532  4
1439  48.566917 -16.958004  4

[1440 rows x 3 columns],
             x1         x2  y
0    -12.304702   3.499240  1
1    -21.302900  17.983794  1
2     -6.320254  29.639092  1
3      2.259775  26.227155  1
4    -14.777150  19.536615  1
…          …         … ..
1435  50.450656 -11.950103  4
1436  49.692856 -17.175831  4
1437  59.579815 -24.594350  4
1438  31.231241 -24.288964  4
1439  47.030269  -7.566840  4

[1440 rows x 3 columns],
             x1         x2  y
0      2.695298 -11.500760  1
1     -6.302900   2.983794  1
2      8.679746  14.639092  1
3     17.259775  11.227155  1
4      0.222850   4.536615  1
…          …         … ..
1435  65.450656 -11.950103  4
1436  64.692856 -17.175831  4
1437  74.579815 -24.594350  4
1438  46.231241 -24.288964  4
1439  62.030269  -7.566840  4

[1440 rows x 3 columns])
```

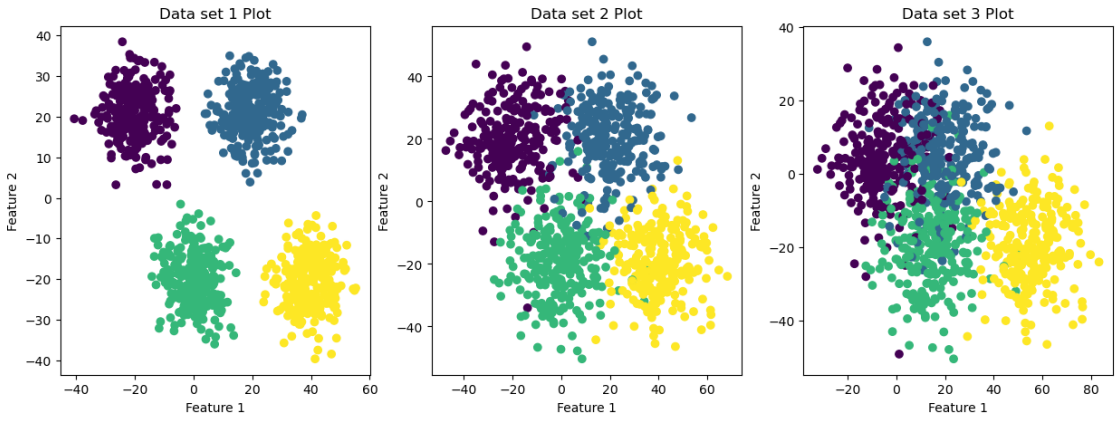
3 Making the Training and Testing DataSets from overall Dataset and use them for all subsequent processing

[33]:
```
X1=data1[['x1','x2']]
y1=data1['y']
X2=data2[['x1','x2']]
y2=data2['y']
X3=data3[['x1','x2']]
y3=data3['y']
```

```
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.3,␣
 ↪random_state=42)
X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size=0.3,␣
 ↪random_state=42)
X3_train, X3_test, y3_train, y3_test = train_test_split(X3, y3, test_size=0.3,␣
 ↪random_state=42)
```

4 Plotting the data using Matplotlib and Guessing Possible Pattern

[34]:
```
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
# Scatter plot 1
axes[0].scatter(X1_train['x1'], X1_train['x2'], c=y1_train)
axes[0].set_xlabel('Feature 1')
axes[0].set_ylabel('Feature 2')
axes[0].set_title('Data set 1 Plot')
# Scatter plot 2
axes[1].scatter(X2_train['x1'], X2_train['x2'], c=y2_train)
axes[1].set_xlabel('Feature 1')
axes[1].set_ylabel('Feature 2')
axes[1].set_title('Data set 2 Plot')
# Scatter plot 3
axes[2].scatter(X3_train['x1'], X3_train['x2'], c=y3_train)
axes[2].set_xlabel('Feature 1')
axes[2].set_ylabel('Feature 2')
axes[2].set_title('Data set 3 Plot')
plt.show()
```



Each of the four clusters in the datasets reveals notable differences in their distinguishability. The initial dataset showcases clusters with a distinct demarcation, making it easy to differentiate be-

tween them. However, in the second dataset, the clusters exhibit a diminished level of distinctiveness compared to the first dataset, creating a more challenging scenario for differentiation.

Dataset 3 presents a unique challenge as its clusters display substantial overlap. This overlap poses a formidable challenge in accurately identifying and delineating individual clusters, particularly when dealing with instances involving the intersection of the four clusters.

5 We have Used the following Algorithms/Variants to process the datasets

- Logistic Regression
- SVC with linear kernel
- SVC with rbf kernel
- Random Forest Classifier with min_samples_leaf=1
- Random Forest Classifier with min_samples_leaf=3
- Random Forest Classifier with min_samples_leaf=5
- Neural Network Classifier with hidden_layer_sizes=(5)
- Neural Network Classifier with hidden_layer_sizes=(5,5)
- Neural Network Classifier with hidden_layer_sizes=(5,5,5)
- Neural Network Classifier with hidden_layer_sizes=(10)

6 The following metrics were generated, captured, and saved into a CSV file for all algorithms on both training and test datasets:

- Train Accuracy
- Train Precision (per class and average)
- Train Recall (per class and average)
- Train F1-score (per class and average)
- Train AUC (per class and average)
- Test Accuracy
- Test Precision (per class and average)
- Test Recall (per class and average)
- Test F1-score (per class and average)
- Test AUC (per class and average)

[42]:
```
# Define the algorithms with probability=True
algorithms = [
    ('Logistic Regression', LogisticRegression()),
```

```
    ('SVC (Linear Kernel)', SVC(kernel='linear', probability=True)),
    ('SVC (RBF Kernel)', SVC(kernel='rbf', probability=True)),
    ('Random Forest Classifier (min_samples_leaf=1)',␣
 ↪RandomForestClassifier(min_samples_leaf=1)),
    ('Random Forest Classifier (min_samples_leaf=3)',␣
 ↪RandomForestClassifier(min_samples_leaf=3)),
    ('Random Forest Classifier (min_samples_leaf=5)',␣
 ↪RandomForestClassifier(min_samples_leaf=5)),
    ('Neural Network Classifier (hidden_layer_sizes=(5,))',␣
 ↪MLPClassifier(hidden_layer_sizes=(5,), max_iter=1000)),
    ('Neural Network Classifier (hidden_layer_sizes=(5,5))',␣
 ↪MLPClassifier(hidden_layer_sizes=(5, 5), max_iter=1000)),
    ('Neural Network Classifier (hidden_layer_sizes=(5,5,5))',␣
 ↪MLPClassifier(hidden_layer_sizes=(5, 5, 5), max_iter=1000)),
    ('Neural Network Classifier (hidden_layer_sizes=(10,))',␣
 ↪MLPClassifier(hidden_layer_sizes=(10,), max_iter=1000))
]

# Define the number of datasets (n)
n_datasets = 3

# Initialize a list to store the results
all_results = []

# Iterate over each dataset
for dataset_index in range(1, n_datasets + 1):
    X_train = globals()[f"X{dataset_index}_train"]
    X_test = globals()[f"X{dataset_index}_test"]
    y_train = globals()[f"y{dataset_index}_train"]
    y_test = globals()[f"y{dataset_index}_test"]

    # Initialize a list to store the results for the current dataset
    results = []

    # Iterate over each algorithm
    for name, model in algorithms:
        # Fit the model to the training data
        model.fit(X_train, y_train)

        # Predictions on training data
        y_train_pred = model.predict(X_train)

        # Predictions on testing data
        y_test_pred = model.predict(X_test)

        # Calculate evaluation metrics for training data
```

```
        train_accuracy = accuracy_score(y_train, y_train_pred)
        train_precision = precision_score(y_train, y_train_pred, average=None)
        train_precision_avg = precision_score(y_train, y_train_pred,␣
↪average='weighted')
        train_recall = recall_score(y_train, y_train_pred, average=None)
        train_recall_avg = recall_score(y_train, y_train_pred,␣
↪average='weighted')
        train_f1 = f1_score(y_train, y_train_pred, average=None)
        train_f1_avg = f1_score(y_train, y_train_pred, average='weighted')
        train_auc = roc_auc_score(y_train, model.predict_proba(X_train),␣
↪average='macro', multi_class='ovr')
        train_auc_avg = roc_auc_score(y_train, model.predict_proba(X_train),␣
↪average='macro', multi_class='ovr')


        # Calculate evaluation metrics for testing data
        test_accuracy = accuracy_score(y_test, y_test_pred)
        test_precision = precision_score(y_test, y_test_pred, average=None)
        test_precision_avg = precision_score(y_test, y_test_pred,␣
↪average='weighted')
        test_recall = recall_score(y_test, y_test_pred, average=None)
        test_recall_avg = recall_score(y_test, y_test_pred, average='weighted')
        test_f1 = f1_score(y_test, y_test_pred, average=None)
        test_f1_avg = f1_score(y_test, y_test_pred, average='weighted')
        test_auc = roc_auc_score(y_test, model.predict_proba(X_test),␣
↪average='macro', multi_class='ovr')
        test_auc_avg = roc_auc_score(y_test, model.predict_proba(X_test),␣
↪average='macro', multi_class='ovr')

        # Store the results
        results.append({
            'Algorithm': name,
            'Train Accuracy': train_accuracy,
            'Train Precision': train_precision,
            'Train Precision (Avg)': train_precision_avg,
            'Train Recall': train_recall,
            'Train Recall (Avg)': train_recall_avg,
            'Train F1-score': train_f1,
            'Train F1-score (Avg)': train_f1_avg,
            'Train AUC': train_auc,
            'Train AUC (Avg)': train_auc_avg,
            'Test Accuracy': test_accuracy,
            'Test Precision': test_precision,
            'Test Precision (Avg)': test_precision_avg,
            'Test Recall': test_recall,
            'Test Recall (Avg)': test_recall_avg,
```

```
            'Test F1-score': test_f1,
            'Test F1-score (Avg)': test_f1_avg,
            'Test AUC': test_auc,
            'Test AUC (Avg)': test_auc_avg
        })

    # Create a DataFrame from the results for the current dataset
    results_df = pd.DataFrame(results)

    # Save the results to a CSV file for the current dataset
    results_df.to_csv(f'algorithm_metrics_dataset_{dataset_index}.csv',␣
↪index=False)

    # Append the results for the current dataset to the overall results list
    all_results.append(results_df)
```

7 METRICS RESULTS WE GOT

7.1 Data set 1

[53]:
```
show1=pd.read_csv('algorithm_metrics_dataset_1.csv')
show1
```

[53]:

	Algorithm	Train Accuracy	Train Precision	Train Precision (Avg)
0	Logistic Regression	1.000000	[1. 1. 1. 1.]	1.000000
1	SVC (Linear Kernel)	1.000000	[1. 1. 1. 1.]	1.000000
2	SVC (RBF Kernel)	1.000000	[1. 1. 1. 1.]	1.000000
3	Random Forest Classifier (min_samples_leaf=1)	1.000000	[1. 1. 1. 1.]	1.000000
4	Random Forest Classifier (min_samples_leaf=3)	1.000000	[1. 1. 1. 1.]	1.000000
5	Random Forest Classifier (min_samples_leaf=5)	1.000000	[1. 1. 1. 1.]	1.000000
6	Neural Network Classifier (hidden_layer_sizes=…	0.998016	[0.99606299 1. 1. 0.99609375]	0.998024
7	Neural Network Classifier (hidden_layer_sizes=…	0.997024	[1. 1. 0.99606299 0.9921875]	0.997028
8	Neural Network Classifier (hidden_layer_sizes=…	0.998016	[0.99606299 1. 1. 0.99609375]	0.998024
9	Neural Network Classifier (hidden_layer_sizes=…	0.999008	[1. 1. 1. 0.99609375]	0.999012

	Train Recall	Train Recall (Avg)
0	[1. 1. 1. 1.]	1.000000
1	[1. 1. 1. 1.]	1.000000
2	[1. 1. 1. 1.]	1.000000
3	[1. 1. 1. 1.]	1.000000
4	[1. 1. 1. 1.]	1.000000
5	[1. 1. 1. 1.]	1.000000
6	[1. 1. 0.99215686 1.]	0.998016
7	[1. 1. 0.99215686 0.99607843]	0.997024
8	[1. 1. 0.99215686 1.]	0.998016
9	[1. 1. 0.99607843 1.]	0.999008

	Train F1-score	Train F1-score (Avg)
0	[1. 1. 1. 1.]	1.000000
1	[1. 1. 1. 1.]	1.000000
2	[1. 1. 1. 1.]	1.000000
3	[1. 1. 1. 1.]	1.000000
4	[1. 1. 1. 1.]	1.000000
5	[1. 1. 1. 1.]	1.000000
6	[0.99802761 1. 0.99606299 0.99804305]	0.998014
7	[1. 1. 0.99410609 0.99412916]	0.997024
8	[0.99802761 1. 0.99606299 0.99804305]	0.998014
9	[1. 1. 0.99803536 0.99804305]	0.999008

	Train AUC	Train AUC (Avg)	Test Accuracy
0	1.000000	1.000000	1.000000
1	1.000000	1.000000	1.000000
2	1.000000	1.000000	1.000000
3	1.000000	1.000000	1.000000
4	1.000000	1.000000	1.000000
5	1.000000	1.000000	1.000000
6	0.999992	0.999992	0.993056
7	0.999992	0.999992	0.993056
8	0.999997	0.999997	0.993056
9	1.000000	1.000000	0.990741

	Test Precision	Test Precision (Avg)
0	[1. 1. 1. 1.]	1.000000
1	[1. 1. 1. 1.]	1.000000
2	[1. 1. 1. 1.]	1.000000
3	[1. 1. 1. 1.]	1.000000
4	[1. 1. 1. 1.]	1.000000
5	[1. 1. 1. 1.]	1.000000
6	[1. 1. 1. 0.97222222]	0.993248
7	[1. 1. 1. 0.97222222]	0.993248
8	[1. 1. 1. 0.97222222]	0.993248

```
9  [1.         0.99137931 1.         0.97222222]              0.990954

                                    Test Recall  Test Recall (Avg)  \
0                                  [1. 1. 1. 1.]           1.000000
1                                  [1. 1. 1. 1.]           1.000000
2                                  [1. 1. 1. 1.]           1.000000
3                                  [1. 1. 1. 1.]           1.000000
4                                  [1. 1. 1. 1.]           1.000000
5                                  [1. 1. 1. 1.]           1.000000
6  [1.         1.         0.97142857 1.        ]           0.993056
7  [1.         1.         0.97142857 1.        ]           0.993056
8  [1.         1.         0.97142857 1.        ]           0.993056
9  [0.99065421 1.         0.97142857 1.        ]           0.990741

                                   Test F1-score  Test F1-score (Avg)  \
0                                  [1. 1. 1. 1.]             1.000000
1                                  [1. 1. 1. 1.]             1.000000
2                                  [1. 1. 1. 1.]             1.000000
3                                  [1. 1. 1. 1.]             1.000000
4                                  [1. 1. 1. 1.]             1.000000
5                                  [1. 1. 1. 1.]             1.000000
6  [1.         1.         0.98550725 0.98591549]             0.993054
7  [1.         1.         0.98550725 0.98591549]             0.993054
8  [1.         1.         0.98550725 0.98591549]             0.993054
9  [0.99530516 0.995671   0.98550725 0.98591549]             0.990739

   Test AUC  Test AUC (Avg)
0  1.000000        1.000000
1  1.000000        1.000000
2  1.000000        1.000000
3  1.000000        1.000000
4  1.000000        1.000000
5  1.000000        1.000000
6  1.000000        1.000000
7  0.999949        0.999949
8  0.999985        0.999985
9  0.999993        0.999993
```

7.2 Data set 2

[56]:
```
show2=pd.read_csv('algorithm_metrics_dataset_2.csv')
show2
```

[56]:
```
                                Algorithm  Train Accuracy  \
0                     Logistic Regression        0.947421
1                     SVC (Linear Kernel)        0.949405
2                        SVC (RBF Kernel)        0.946429
```

```
3      Random Forest Classifier (min_samples_leaf=1)        1.000000
4      Random Forest Classifier (min_samples_leaf=3)        0.962302
5      Random Forest Classifier (min_samples_leaf=5)        0.954365
6  Neural Network Classifier (hidden_layer_sizes=…        0.939484
7  Neural Network Classifier (hidden_layer_sizes=…        0.942460
8  Neural Network Classifier (hidden_layer_sizes=…        0.950397
9  Neural Network Classifier (hidden_layer_sizes=…        0.937500

                                  Train Precision  Train Precision (Avg)  \
0  [0.9488189  0.94628099 0.9296875  0.96484375]                0.947416
1  [0.94921875 0.95798319 0.92664093 0.96470588]                0.949555
2  [0.9561753  0.94262295 0.91954023 0.96825397]                0.946669
3                                [1. 1. 1. 1.]                1.000000
4  [0.96825397 0.96680498 0.94208494 0.97265625]                0.962395
5  [0.96414343 0.95435685 0.93436293 0.96498054]                0.954443
6  [0.93846154 0.96137339 0.92125984 0.93869732]                0.939738
7  [0.94921875 0.94605809 0.921875   0.95294118]                0.942475
8  [0.94573643 0.95780591 0.93410853 0.96470588]                0.950527
9  [0.95238095 0.91935484 0.93548387 0.94230769]                0.937531

                                   Train Recall  Train Recall (Avg)  \
0  [0.95256917 0.93469388 0.93333333 0.96862745]            0.947421
1  [0.96047431 0.93061224 0.94117647 0.96470588]            0.949405
2  [0.9486166  0.93877551 0.94117647 0.95686275]            0.946429
3                                [1. 1. 1. 1.]            1.000000
4  [0.96442688 0.95102041 0.95686275 0.97647059]            0.962302
5  [0.95652174 0.93877551 0.94901961 0.97254902]            0.954365
6  [0.96442688 0.91428571 0.91764706 0.96078431]            0.939484
7  [0.96047431 0.93061224 0.9254902  0.95294118]            0.942460
8  [0.96442688 0.92653061 0.94509804 0.96470588]            0.950397
9  [0.9486166  0.93061224 0.90980392 0.96078431]            0.937500

                                  Train F1-score  Train F1-score (Avg)  \
0  [0.95069034 0.94045175 0.93150685 0.9667319 ]              0.947407
1  [0.95481336 0.94409938 0.93385214 0.96470588]              0.949409
2  [0.95238095 0.9406953  0.93023256 0.96252465]              0.946504
3                                [1. 1. 1. 1.]              1.000000
4  [0.96633663 0.95884774 0.94941634 0.97455969]              0.962316
5  [0.96031746 0.94650206 0.94163424 0.96875   ]              0.954366
6  [0.95126706 0.93723849 0.9194499  0.9496124 ]              0.939390
7  [0.95481336 0.9382716  0.92367906 0.95294118]              0.942443
8  [0.95499022 0.94190871 0.93957115 0.96470588]              0.950368
9  [0.95049505 0.92494929 0.92246521 0.95145631]              0.937438

   Train AUC  Train AUC (Avg)  Test Accuracy  \
0   0.994674         0.994674       0.962963
1   0.994362         0.994362       0.965278
```

```
2   0.994850          0.994850       0.962963
3   1.000000          1.000000       0.953704
4   0.998724          0.998724       0.965278
5   0.997960          0.997960       0.958333
6   0.993534          0.993534       0.956019
7   0.993539          0.993539       0.960648
8   0.993482          0.993482       0.949074
9   0.994007          0.994007       0.956019

                                  Test Precision  Test Precision (Avg)  \
0  [0.97196262 0.96460177 0.95283019 0.96226415]              0.962996
1  [0.97222222 0.97321429 0.93636364 0.98039216]              0.965756
2  [0.98095238 0.95652174 0.92857143 0.99      ]              0.963916
3  [0.94594595 0.96363636 0.95238095 0.95283019]              0.953893
4  [0.97196262 0.96491228 0.95327103 0.97115385]              0.965346
5  [0.96296296 0.96396396 0.95327103 0.95283019]              0.958411
6  [0.97196262 0.98165138 0.93518519 0.93518519]              0.956664
7  [0.97222222 0.97321429 0.94392523 0.95238095]              0.960786
8  [0.9537037  0.96396396 0.90990991 0.97058824]              0.949895
9  [0.97196262 0.95652174 0.94339623 0.95192308]              0.956038

                                     Test Recall  Test Recall (Avg)  \
0  [0.97196262 0.94782609 0.96190476 0.97142857]           0.962963
1  [0.98130841 0.94782609 0.98095238 0.95238095]           0.965278
2  [0.96261682 0.95652174 0.99047619 0.94285714]           0.962963
3  [0.98130841 0.92173913 0.95238095 0.96190476]           0.953704
4  [0.97196262 0.95652174 0.97142857 0.96190476]           0.965278
5  [0.97196262 0.93043478 0.97142857 0.96190476]           0.958333
6  [0.97196262 0.93043478 0.96190476 0.96190476]           0.956019
7  [0.98130841 0.94782609 0.96190476 0.95238095]           0.960648
8  [0.96261682 0.93043478 0.96190476 0.94285714]           0.949074
9  [0.97196262 0.95652174 0.95238095 0.94285714]           0.956019

                                    Test F1-score  Test F1-score (Avg)  \
0  [0.97196262 0.95614035 0.95734597 0.96682464]             0.962949
1  [0.97674419 0.96035242 0.95813953 0.96618357]             0.965292
2  [0.97169811 0.95652174 0.95852535 0.96585366]             0.963036
3  [0.96330275 0.94222222 0.95238095 0.95734597]             0.953589
4  [0.97196262 0.96069869 0.96226415 0.96650718]             0.965281
5  [0.96744186 0.94690265 0.96226415 0.95734597]             0.958262
6  [0.97196262 0.95535714 0.94835681 0.94835681]             0.956067
7  [0.97674419 0.96035242 0.95283019 0.95238095]             0.960647
8  [0.95813953 0.94690265 0.93518519 0.95652174]             0.949176
9  [0.97196262 0.95652174 0.9478673  0.94736842]             0.956018

   Test AUC  Test AUC (Avg)
0  0.998513        0.998513
```

```
1  0.998327        0.998327
2  0.998356        0.998356
3  0.996871        0.996871
4  0.997541        0.997541
5  0.997400        0.997400
6  0.997582        0.997582
7  0.997181        0.997181
8  0.996697        0.996697
9  0.997816        0.997816
```

7.3 Data set 3

[55]:
```
show3=pd.read_csv('algorithm_metrics_dataset_3.csv')
show3
```

[55]:
```
                                          Algorithm  Train Accuracy  \
0                              Logistic Regression        0.864087
1                              SVC (Linear Kernel)        0.863095
2                                 SVC (RBF Kernel)        0.859127
3      Random Forest Classifier (min_samples_leaf=1)        1.000000
4      Random Forest Classifier (min_samples_leaf=3)        0.907738
5      Random Forest Classifier (min_samples_leaf=5)        0.883929
6  Neural Network Classifier (hidden_layer_sizes=…        0.861111
7  Neural Network Classifier (hidden_layer_sizes=…        0.818452
8  Neural Network Classifier (hidden_layer_sizes=…        0.858135
9  Neural Network Classifier (hidden_layer_sizes=…        0.847222

                                 Train Precision  Train Precision (Avg)  \
0  [0.86746988 0.79423868 0.83984375 0.95      ]                0.863560
1  [0.87704918 0.79508197 0.82824427 0.9496124 ]                0.863137
2  [0.87551867 0.80672269 0.80514706 0.94941634]                0.859689
3                              [1. 1. 1. 1.]                    1.000000
4  [0.92276423 0.86122449 0.87739464 0.96875   ]                0.907963
5  [0.88571429 0.82520325 0.85769231 0.96498054]                0.883970
6  [0.87242798 0.7556391  0.86919831 0.95038168]                0.862945
7  [0.89189189 0.79620853 0.76923077 0.82450331]                0.820557
8  [0.87449393 0.82017544 0.80597015 0.92830189]                0.857569
9  [0.8531746  0.78423237 0.82926829 0.91449814]                0.845883

                                    Train Recall  Train Recall (Avg)  \
0  [0.85375494 0.7877551  0.84313725 0.96862745]            0.864087
1  [0.8458498  0.79183673 0.85098039 0.96078431]            0.863095
2  [0.83399209 0.78367347 0.85882353 0.95686275]            0.859127
3                              [1. 1. 1. 1.]                1.000000
4  [0.8972332  0.86122449 0.89803922 0.97254902]            0.907738
5  [0.85770751 0.82857143 0.8745098  0.97254902]            0.883929
6  [0.83794466 0.82040816 0.80784314 0.97647059]            0.861111
```

```
7  [0.7826087  0.68571429 0.82352941 0.97647059]            0.818452
8  [0.85375494 0.76326531 0.84705882 0.96470588]            0.858135
9  [0.84980237 0.77142857 0.8        0.96470588]            0.847222

                                  Train F1-score  Train F1-score (Avg)  \
0  [0.86055777 0.79098361 0.84148728 0.9592233 ]              0.863783
1  [0.861167   0.79345603 0.83945841 0.95516569]              0.862997
2  [0.85425101 0.79503106 0.83111954 0.953125  ]              0.859018
3                                [1. 1. 1. 1.]              1.000000
4  [0.90981964 0.86122449 0.8875969  0.97064579]              0.907774
5  [0.87148594 0.82688391 0.86601942 0.96875   ]              0.883868
6  [0.85483871 0.78669276 0.83739837 0.96324952]              0.861289
7  [0.83368421 0.73684211 0.79545455 0.8940754 ]              0.815753
8  [0.864      0.79069767 0.82600382 0.94615385]              0.857354
9  [0.85148515 0.77777778 0.81437126 0.9389313 ]              0.846303

   Train AUC  Train AUC (Avg)  Test Accuracy  \
0   0.967657         0.967657       0.872685
1   0.967301         0.967301       0.868056
2   0.967365         0.967365       0.861111
3   1.000000         1.000000       0.854167
4   0.992559         0.992559       0.863426
5   0.987641         0.987641       0.858796
6   0.961511         0.961511       0.863426
7   0.960057         0.960057       0.787037
8   0.965551         0.965551       0.851852
9   0.965861         0.965861       0.833333

                                  Test Precision  Test Precision (Avg)  \
0  [0.82300885 0.89247312 0.82352941 0.96261682]              0.875560
1  [0.81981982 0.89247312 0.8        0.98058252]              0.873417
2  [0.81481481 0.88297872 0.78740157 0.98058252]              0.866588
3  [0.8125     0.85106383 0.8173913  0.93693694]              0.854200
4  [0.82142857 0.88043478 0.82051282 0.93693694]              0.864989
5  [0.81818182 0.87096774 0.80327869 0.95327103]              0.861446
6  [0.8245614  0.83838384 0.8440367  0.94545455]              0.862358
7  [0.83505155 0.85365854 0.72727273 0.76515152]              0.796819
8  [0.82300885 0.88636364 0.81355932 0.89380531]              0.854785
9  [0.80357143 0.85869565 0.80701754 0.86842105]              0.834845

                                     Test Recall  Test Recall (Avg)  \
0  [0.86915888 0.72173913 0.93333333 0.98095238]           0.872685
1  [0.85046729 0.72173913 0.95238095 0.96190476]           0.868056
2  [0.82242991 0.72173913 0.95238095 0.96190476]           0.861111
3  [0.85046729 0.69565217 0.8952381  0.99047619]           0.854167
4  [0.85981308 0.70434783 0.91428571 0.99047619]           0.863426
5  [0.8411215  0.70434783 0.93333333 0.97142857]           0.858796
```

```
6  [0.87850467 0.72173913 0.87619048 0.99047619]           0.863426
7  [0.75700935 0.60869565 0.83809524 0.96190476]           0.787037
8  [0.86915888 0.67826087 0.91428571 0.96190476]           0.851852
9  [0.8411215  0.68695652 0.87619048 0.94285714]           0.833333

                                  Test F1-score  Test F1-score (Avg)  \
0  [0.84545455 0.79807692 0.875      0.97169811]             0.870708
1  [0.83486239 0.79807692 0.86956522 0.97115385]             0.866631
2  [0.81860465 0.79425837 0.86206897 0.97115385]             0.859766
3  [0.83105023 0.76555024 0.85454545 0.96296296]             0.851387
4  [0.84018265 0.7826087  0.86486486 0.96296296]             0.860698
5  [0.82949309 0.77884615 0.86343612 0.96226415]             0.856531
6  [0.85067873 0.77570093 0.85981308 0.96744186]             0.861319
7  [0.79411765 0.7106599  0.77876106 0.85232068]             0.782315
8  [0.84545455 0.76847291 0.86098655 0.9266055 ]             0.848461
9  [0.82191781 0.76328502 0.84018265 0.90410959]             0.830726

   Test AUC  Test AUC (Avg)
0  0.978609        0.978609
1  0.978444        0.978444
2  0.979173        0.979173
3  0.966482        0.966482
4  0.971475        0.971475
5  0.972779        0.972779
6  0.975062        0.975062
7  0.963616        0.963616
8  0.975768        0.975768
9  0.971093        0.971093
```

8 Countour Plot for Data Sets

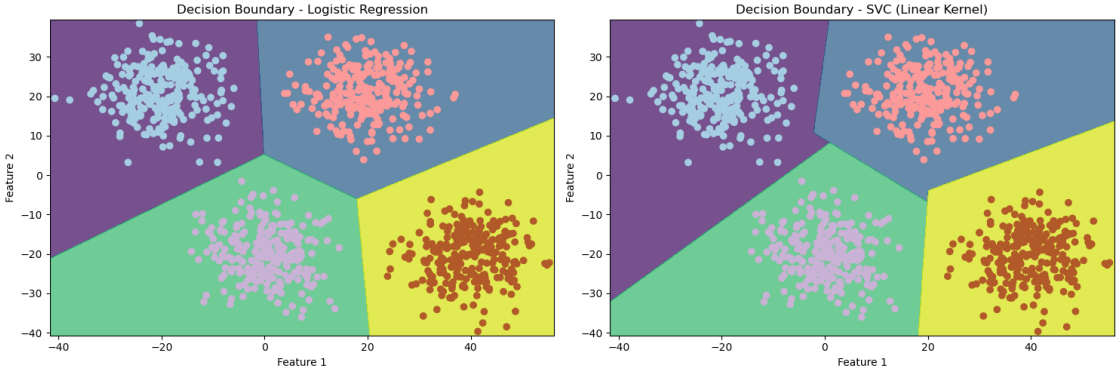
8.1 Data Set 1

[59]:
```
# Define meshgrid for decision boundary plot
X_=np.arange(start=X1_train['x1'].min()-1, stop=X1_train['x1'].max()+1, step=0.
 ↪07)
Y_=np.arange(start=X1_train['x2'].min()-1, stop=X1_train['x2'].max()+1, step=0.
 ↪07)
xx,yy=np.meshgrid(X_,Y_)
# Define classifiers
classifiers = [
('Logistic Regression', LogisticRegression()),
('SVC (Linear Kernel)', SVC(kernel='linear', probability=True)),
('SVC (RBF Kernel)', SVC(kernel='rbf', probability=True)),
('Random Forest Classifier (min_samples_leaf=1)',␣
 ↪RandomForestClassifier(min_samples_leaf=1)),
```

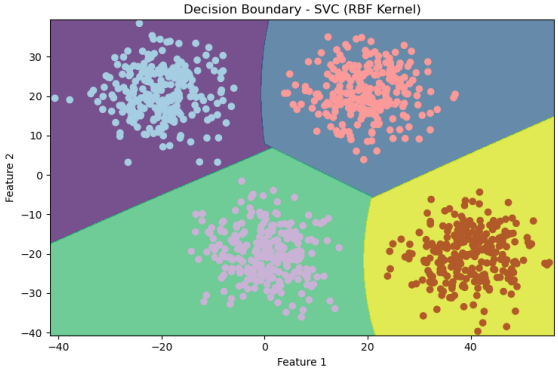
```
('Random Forest Classifier (min_samples_leaf=3)',␣
 ↪RandomForestClassifier(min_samples_leaf=3)),
('Random Forest Classifier (min_samples_leaf=5)',␣
 ↪RandomForestClassifier(min_samples_leaf=5)),
('Neural Network Classifier (hidden_layer_sizes=(5,))',␣
 ↪MLPClassifier(hidden_layer_sizes=(5,), max_iter=1000)),
('Neural Network Classifier (hidden_layer_sizes=(5,5))',␣
 ↪MLPClassifier(hidden_layer_sizes=(5, 5), max_iter=1000)),
('Neural Network Classifier (hidden_layer_sizes=(5,5,5))',␣
 ↪MLPClassifier(hidden_layer_sizes=(5, 5, 5), max_iter=1000)),
('Neural Network Classifier (hidden_layer_sizes=(10,))',␣
 ↪MLPClassifier(hidden_layer_sizes=(10,), max_iter=1000))
]
fig, axes = plt.subplots(5, 2, figsize=(15, 25))
# Flatten axes for easier iteration
axes = axes.flatten()

# Plot decision boundary and ROC curve for each classifier
for ax, (name, classifier) in zip(axes, classifiers):
    classifier.fit(X1_train, y1_train)
    # Plot decision boundary
    Z = classifier.predict(np.array([xx.ravel(), yy.ravel()]).T)
    Z = Z.reshape(xx.shape)
    ax.contourf(xx, yy, Z, alpha=0.75)
    ax.scatter(X1_train.iloc[:, [0]], X1_train.iloc[:, [1]], c=y1_train,␣
 ↪cmap=plt.cm.Paired)
    ax.set_xlabel('Feature 1')
    ax.set_ylabel('Feature 2')
    ax.set_title(f'Decision Boundary - {name}')

plt.tight_layout()
plt.show()
```

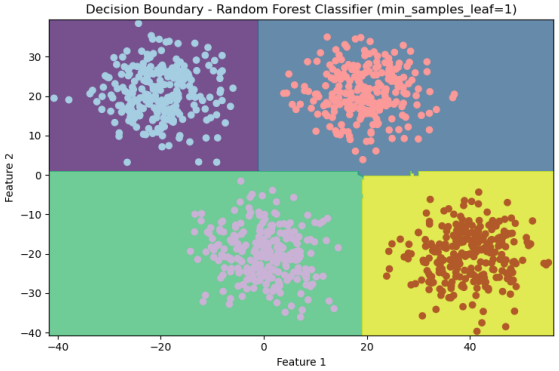
```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but LogisticRegression was fitted with
feature names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but RandomForestClassifier was fitted with
feature names
  warnings.warn(
```

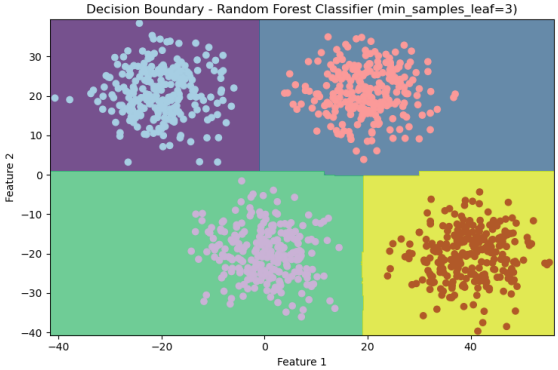
```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but RandomForestClassifier was fitted with
feature names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but RandomForestClassifier was fitted with
feature names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but MLPClassifier was fitted with feature
names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but MLPClassifier was fitted with feature
names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but MLPClassifier was fitted with feature
names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but MLPClassifier was fitted with feature
names
  warnings.warn(
```

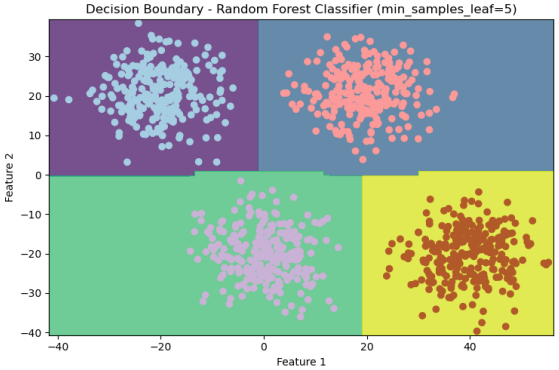

Decision Boundary - Logistic Regression
Feature 1
Feature 2

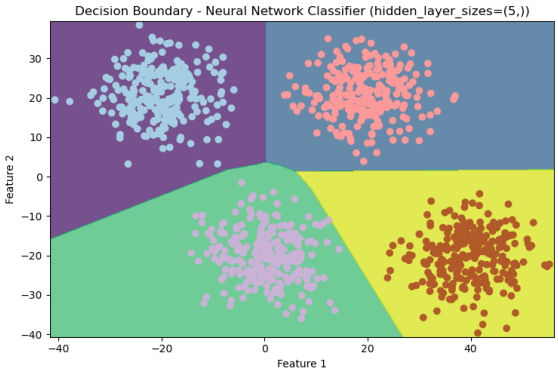
Decision Boundary - SVC (Linear Kernel)
Feature 1
Feature 2

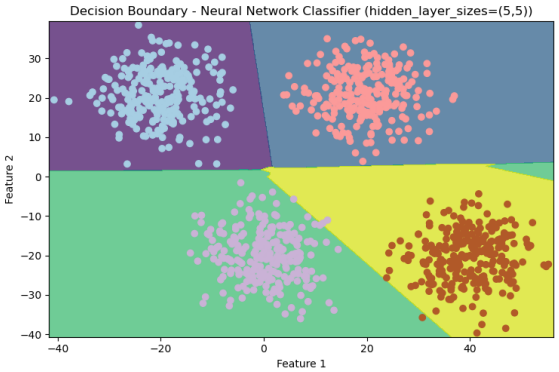

Decision Boundary - SVC (RBF Kernel)
Feature 1
Feature 2

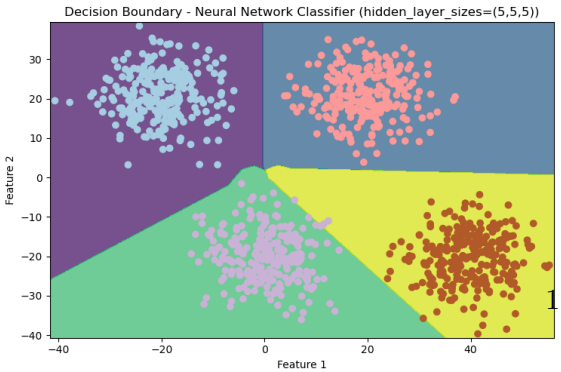

Decision Boundary - Random Forest Classifier (min_samples_leaf=1)
Feature 1
Feature 2

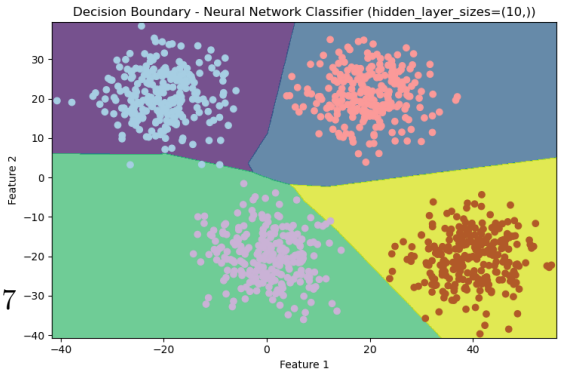

Decision Boundary - Random Forest Classifier (min_samples_leaf=3)
Feature 1
Feature 2


Decision Boundary - Random Forest Classifier (min_samples_leaf=5)
Feature 1
Feature 2


Decision Boundary - Neural Network Classifier (hidden_layer_sizes=(5,))
Feature 1
Feature 2


Decision Boundary - Neural Network Classifier (hidden_layer_sizes=(5,5))
Feature 1
Feature 2


Decision Boundary - Neural Network Classifier (hidden_layer_sizes=(5,5,5))
Feature 1
Feature 2

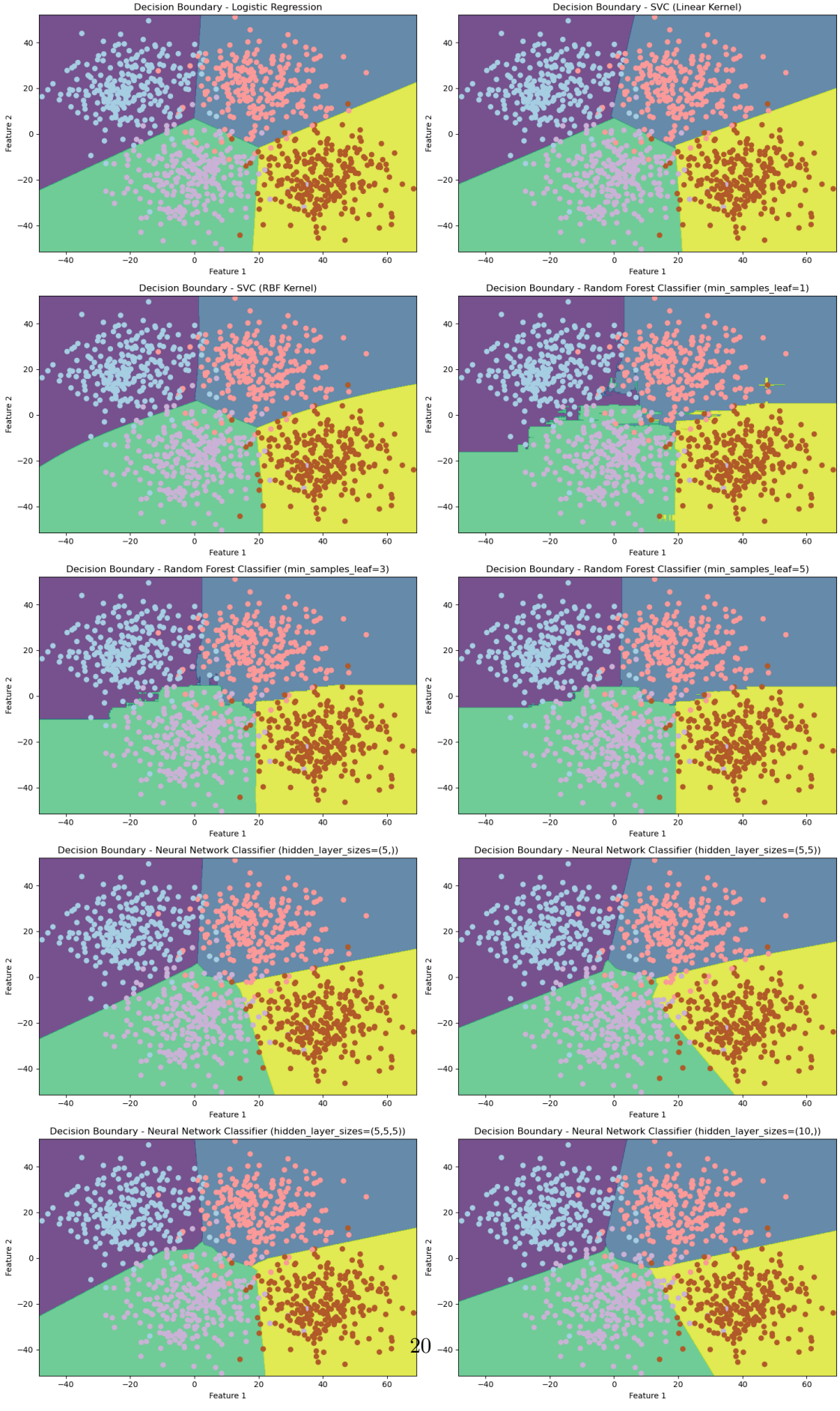

Decision Boundary - Neural Network Classifier (hidden_layer_sizes=(10,))
Feature 1
Feature 2

9 Data Set 2

[60]:
```
# Define meshgrid for decision boundary plot
X_=np.arange(start=X2_train['x1'].min()-1, stop=X2_train['x1'].max()+1, step=0.
 ↪07)
Y_=np.arange(start=X2_train['x2'].min()-1, stop=X2_train['x2'].max()+1, step=0.
 ↪07)
xx,yy=np.meshgrid(X_,Y_)
# Define classifiers
classifiers = [
('Logistic Regression', LogisticRegression()),
('SVC (Linear Kernel)', SVC(kernel='linear', probability=True)),
('SVC (RBF Kernel)', SVC(kernel='rbf', probability=True)),
('Random Forest Classifier (min_samples_leaf=1)',␣
 ↪RandomForestClassifier(min_samples_leaf=1)),
('Random Forest Classifier (min_samples_leaf=3)',␣
 ↪RandomForestClassifier(min_samples_leaf=3)),
('Random Forest Classifier (min_samples_leaf=5)',␣
 ↪RandomForestClassifier(min_samples_leaf=5)),
('Neural Network Classifier (hidden_layer_sizes=(5,))',␣
 ↪MLPClassifier(hidden_layer_sizes=(5,), max_iter=1000)),
('Neural Network Classifier (hidden_layer_sizes=(5,5))',␣
 ↪MLPClassifier(hidden_layer_sizes=(5, 5), max_iter=1000)),
('Neural Network Classifier (hidden_layer_sizes=(5,5,5))',␣
 ↪MLPClassifier(hidden_layer_sizes=(5, 5, 5), max_iter=1000)),
('Neural Network Classifier (hidden_layer_sizes=(10,))',␣
 ↪MLPClassifier(hidden_layer_sizes=(10,), max_iter=1000))
]
fig, axes = plt.subplots(5, 2, figsize=(15, 25))
# Flatten axes for easier iteration
axes = axes.flatten()

# Plot decision boundary and ROC curve for each classifier
for ax, (name, classifier) in zip(axes, classifiers):
    classifier.fit(X2_train, y2_train)
    # Plot decision boundary
    Z = classifier.predict(np.array([xx.ravel(), yy.ravel()]).T)
    Z = Z.reshape(xx.shape)
    ax.contourf(xx, yy, Z, alpha=0.75)
    ax.scatter(X2_train.iloc[:, [0]], X2_train.iloc[:, [1]], c=y2_train,␣
 ↪cmap=plt.cm.Paired)
    ax.set_xlabel('Feature 1')
    ax.set_ylabel('Feature 2')
```

```python
    ax.set_title(f'Decision Boundary - {name}')

plt.tight_layout()
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but LogisticRegression was fitted with
feature names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but RandomForestClassifier was fitted with
feature names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but RandomForestClassifier was fitted with
feature names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but RandomForestClassifier was fitted with
feature names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but MLPClassifier was fitted with feature
names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but MLPClassifier was fitted with feature
names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but MLPClassifier was fitted with feature
names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but MLPClassifier was fitted with feature
names
  warnings.warn(
```

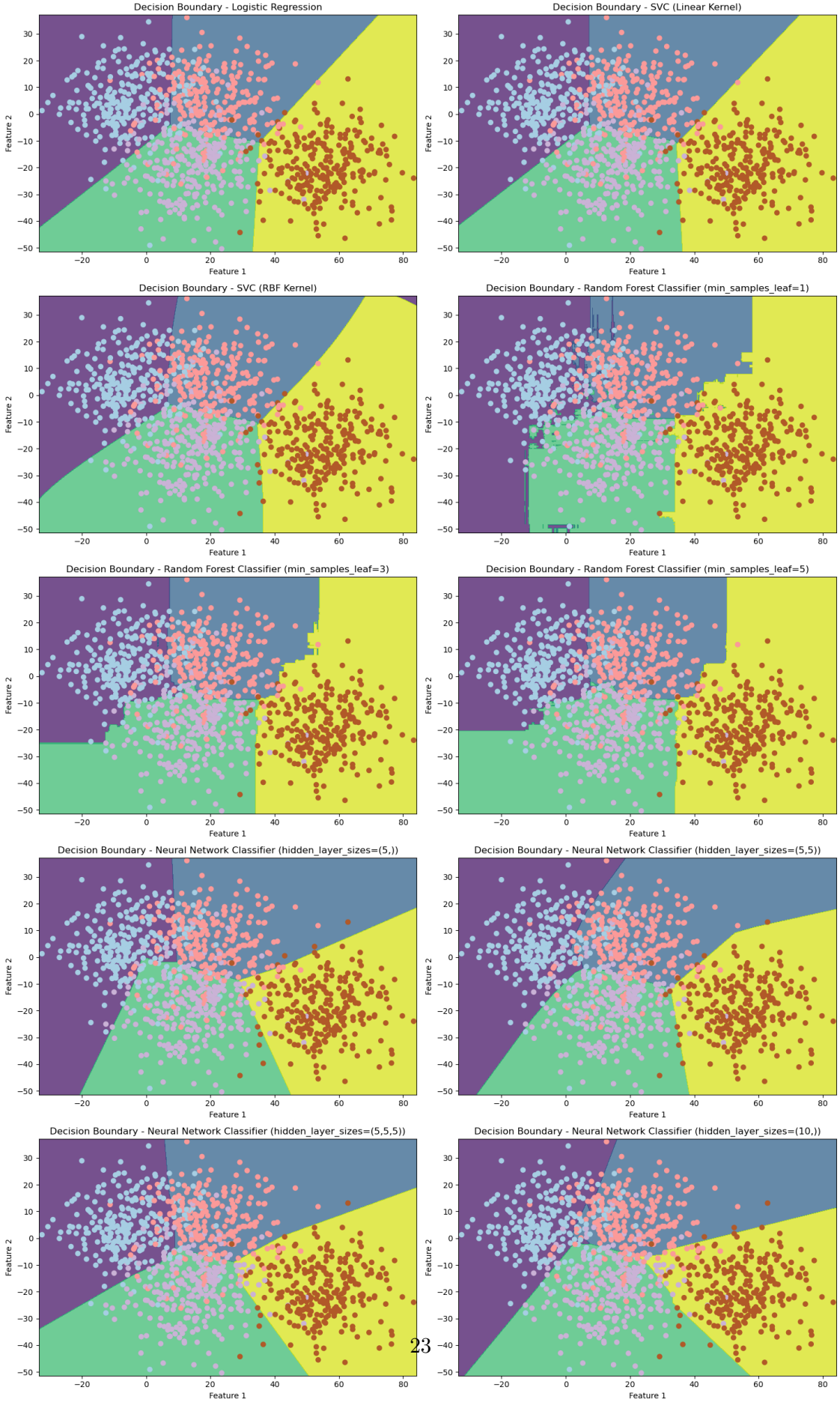

Decision Boundary - Logistic Regression
Decision Boundary - SVC (Linear Kernel)
Decision Boundary - SVC (RBF Kernel)
Decision Boundary - Random Forest Classifier (min_samples_leaf=1)
Decision Boundary - Random Forest Classifier (min_samples_leaf=3)
Decision Boundary - Random Forest Classifier (min_samples_leaf=5)
Decision Boundary - Neural Network Classifier (hidden_layer_sizes=(5,))
Decision Boundary - Neural Network Classifier (hidden_layer_sizes=(5,5))
Decision Boundary - Neural Network Classifier (hidden_layer_sizes=(5,5,5))
Decision Boundary - Neural Network Classifier (hidden_layer_sizes=(10,))
Feature 1
Feature 2

10 Data Set 3

[61]:
```
# Define meshgrid for decision boundary plot
X_=np.arange(start=X3_train['x1'].min()-1, stop=X3_train['x1'].max()+1, step=0.
 ↪07)
Y_=np.arange(start=X3_train['x2'].min()-1, stop=X3_train['x2'].max()+1, step=0.
 ↪07)
xx,yy=np.meshgrid(X_,Y_)
# Define classifiers
classifiers = [
('Logistic Regression', LogisticRegression()),
('SVC (Linear Kernel)', SVC(kernel='linear', probability=True)),
('SVC (RBF Kernel)', SVC(kernel='rbf', probability=True)),
('Random Forest Classifier (min_samples_leaf=1)',␣
 ↪RandomForestClassifier(min_samples_leaf=1)),
('Random Forest Classifier (min_samples_leaf=3)',␣
 ↪RandomForestClassifier(min_samples_leaf=3)),
('Random Forest Classifier (min_samples_leaf=5)',␣
 ↪RandomForestClassifier(min_samples_leaf=5)),
('Neural Network Classifier (hidden_layer_sizes=(5,))',␣
 ↪MLPClassifier(hidden_layer_sizes=(5,), max_iter=1000)),
('Neural Network Classifier (hidden_layer_sizes=(5,5))',␣
 ↪MLPClassifier(hidden_layer_sizes=(5, 5), max_iter=1000)),
('Neural Network Classifier (hidden_layer_sizes=(5,5,5))',␣
 ↪MLPClassifier(hidden_layer_sizes=(5, 5, 5), max_iter=1000)),
('Neural Network Classifier (hidden_layer_sizes=(10,))',␣
 ↪MLPClassifier(hidden_layer_sizes=(10,), max_iter=1000))
]
fig, axes = plt.subplots(5, 2, figsize=(15, 25))
# Flatten axes for easier iteration
axes = axes.flatten()

# Plot decision boundary and ROC curve for each classifier
for ax, (name, classifier) in zip(axes, classifiers):
    classifier.fit(X3_train, y3_train)
    # Plot decision boundary
    Z = classifier.predict(np.array([xx.ravel(), yy.ravel()]).T)
    Z = Z.reshape(xx.shape)
    ax.contourf(xx, yy, Z, alpha=0.75)
    ax.scatter(X3_train.iloc[:, [0]], X3_train.iloc[:, [1]], c=y3_train,␣
 ↪cmap=plt.cm.Paired)
    ax.set_xlabel('Feature 1')
    ax.set_ylabel('Feature 2')
```

```python
    ax.set_title(f'Decision Boundary - {name}')

plt.tight_layout()
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but LogisticRegression was fitted with
feature names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but RandomForestClassifier was fitted with
feature names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but RandomForestClassifier was fitted with
feature names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but RandomForestClassifier was fitted with
feature names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but MLPClassifier was fitted with feature
names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but MLPClassifier was fitted with feature
names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but MLPClassifier was fitted with feature
names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but MLPClassifier was fitted with feature
names
  warnings.warn(
```


Decision Boundary - Logistic Regression
Decision Boundary - SVC (Linear Kernel)
Decision Boundary - SVC (RBF Kernel)
Decision Boundary - Random Forest Classifier (min_samples_leaf=1)
Decision Boundary - Random Forest Classifier (min_samples_leaf=3)
Decision Boundary - Random Forest Classifier (min_samples_leaf=5)
Decision Boundary - Neural Network Classifier (hidden_layer_sizes=(5,))
Decision Boundary - Neural Network Classifier (hidden_layer_sizes=(5,5))
Decision Boundary - Neural Network Classifier (hidden_layer_sizes=(5,5,5))
Decision Boundary - Neural Network Classifier (hidden_layer_sizes=(10,))
Feature 1
Feature 2

11 Model Performance Analysis

11.1 About

In this analysis, we evaluate the performance of various classification algorithms on three different datasets (Dataset 1, Dataset 2, and Dataset 3). The evaluation is based on multiple metrics, including accuracy, precision, recall, F1-score, and AUC.

11.1.1 Metric Definitions

Metric	Definition	Dataset 1 Characteristics	Dataset 2 Characteristics	Dataset 3 Characteristics
Accuracy	Overall correctness of the model's predictions	Easy to distinguish clusters	One cluster less distinguishable	Overlapping clusters, difficult to distinguish
Precision	Correctness of positive predictions made by the model	High precision for all clusters	Lower precision for one cluster	Varied precision due to overlap
Precision (per class)	Ratio of correctly predicted instances of a class to all instances predicted as that class	High precision for each class	Lower precision for one class	Varied precision for each class
Precision (average)	Average of precision values across all classes	High average precision	Lower average precision	Varied average precision
Recall	Ability of the model to correctly identify instances of a class	High recall for all clusters	Lower recall for one cluster	Varied recall due to overlap
Recall (per class)	Ratio of correctly predicted instances of a class to all instances of that class	High recall for each class	Lower recall for one class	Varied recall for each class
Recall (average)	Average of recall values across all classes	High average recall	Lower average recall	Varied average recall
F1-score	Harmonic mean of precision and recall, providing a balance between the two	High F1-score for all clusters	Lower F1-score for one cluster	Varied F1-score due to overlap
F1-score (per class)	Balance between precision and recall for each class	High F1-score for each class	Lower F1-score for one class	Varied F1-score for each class
F1-score (average)	Average of F1-score values across all classes	High average F1-score	Lower average F1-score	Varied average F1-score

Metric	Definition	Dataset 1 Characteristics	Dataset 2 Characteristics	Dataset 3 Characteristics
AUC	Area Under the Curve, measures the ability of the model to distinguish between classes	High AUC for all clusters	Lower AUC for one cluster	Varied AUC due to overlap
AUC (per class)	AUC calculated for each class separately	High AUC for each class	Lower AUC for one class	Varied AUC for each class
AUC (average)	Average of AUC values across all classes	High average AUC	Lower average AUC	Varied average AUC

11.2 Dataset Analysis

11.2.1 Dataset 1

- All algorithms demonstrate perfect performance on the train dataset, achieving an accuracy, precision, recall, F1-score, and AUC of 1. This indicates that the models perfectly fit the training data and can classify instances without errors and maintain their high performance without overfitting.

11.2.2 Dataset 2

- In some cases, particularly with the Random Forest Classifier (min_samples_leaf=1), the models achieve perfect accuracy on the training data but slightly lower accuracy on the test data, indicating potential overfitting.
- The Logistic Regression, SVC with Linear Kernel, and SVC with RBF Kernel consistently achieve high accuracy and precision scores on both the train and test datasets, indicating robust performance in correctly classifying instances across different classes.
- The Random Forest Classifier with min_samples_leaf=3 also demonstrates strong recall and F1-score values, suggesting its effectiveness in correctly identifying positive instances and achieving a balance between precision and recall.

11.2.3 Dataset 3

- In some cases, particularly with the Random Forest Classifier (min_samples_leaf=1), the models achieve perfect accuracy on the training data but slightly lower accuracy on the test data, indicating potential overfitting.
- Overall, the accuracy of the models ranges from approximately 86% to 100% on the training data and from approximately 83% to 96% on the test data.
- The performance of algorithms such as Logistic Regression, SVC with linear and RBF kernels, and Random Forest Classifier with min_samples_leaf=3 generally exhibit balanced performance across different metrics. The neural network classifiers with different hidden layer sizes also demonstrate competitive performance, although they may require careful tuning of hyperparameters to optimize performance.

11.3 Major Learnings

- The performance of classification algorithms varied across different datasets and metrics. Certain algorithms demonstrated superior performance on specific datasets, highlighting the significance of selecting appropriate algorithms based on the unique characteristics of the data.
- Overfitting was observed in some cases, emphasizing the importance of employing regularization techniques or adjusting model hyperparameters to improve generalization and prevent overfitting.
- The choice of evaluation metrics played a crucial role in assessing various aspects of model performance. Metrics such as precision, recall, and overall accuracy provided valuable insights into the effectiveness of the classification models and their ability to correctly classify instances across different classes.

[ ]: