**Exercise – 4: DS203-2024-S1**

**Submissions due by: September 9, 2024, 11:55pm**

This exercise is aimed at:

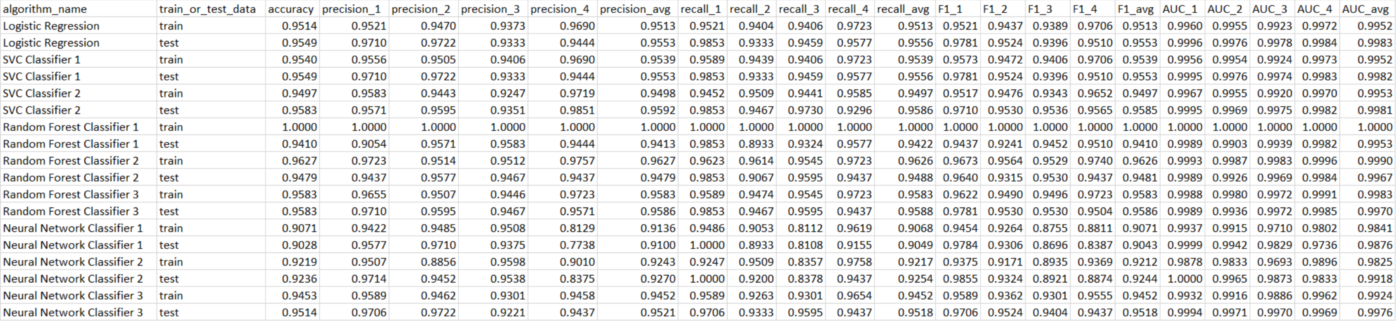
- Getting introduced to and running various **Classification** algorithms on a given data set and understanding their relative characteristics, performance, and advantages.
- Calculating, effectively documenting, and understanding various Classification metrics and developing an approach towards effectively using them.
- Creating and consolidating multiple plots with the aim to compare and contrast the results of the algorithms
- Get introduced to the relevant functions of the Python library: **sklearn**

In this Exercise you will be processing the following datasets:

- Clusters-4-v0.csv
- Clusters-4-v1.csv
- Clusters-4-v2.csv

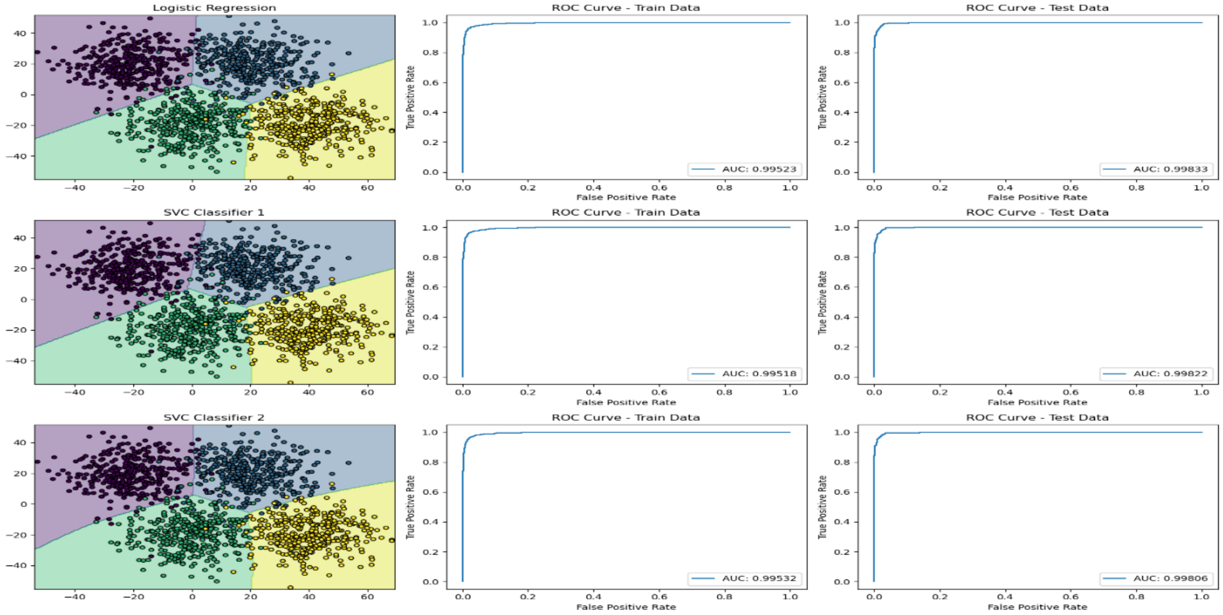
Processing steps:

1. Divide each data set into 'train' and 'test' datasets, once, and use them for all subsequent steps.
2. Review the data using appropriate plots and understand the overall structure of the data. Comment on the data and anticipate how well LogisticRegression will perform on the data.
3. Use the following algorithms / variants to process the datasets:
    a. Logistic Regression
    b. SVC – with 'linear' kernel (what is 'linear'?)
    c. SVC – with 'rbf' kernel (what is 'rbf'?)
    d. Random Forest Classifier – with min_samples_leaf=1
    e. Random Forest Classifier – with min_samples_leaf=3
    f. Random Forest Classifier – with min_samples_leaf=5
    g. Neural Network Classifier – with hidden_layer_sizes=(5)
    h. Neural Network Classifier – with hidden_layer_sizes=(5,5)
    i. Neural Network Classifier – with hidden_layer_sizes=(5,5,5)
    j. Neural Network Classifier – with hidden_layer_sizes=(10)
4. In each of the above cases generate, capture, and save all the results, for all the datasets, into a common csv file – to facilitate analysis later on. The following metrics (for train and test data) should be created: (For example, see the image that follows):
    - Accuracy, Precision (per class), Precision (average), Recall (per class), Recall (average), F1-score (per class), F1-score (average), AUC (per class), AUC (average).
    - (Hint: The following functions may be used: accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, roc_curve)



| algorithm_name | train_or_test_data | accuracy | precision_1 | precision_2 | precision_3 | precision_4 | precision_avg | recall_1 | recall_2 | recall_3 | recall_4 | recall_avg | F1_1 | F1_2 | F1_3 | F1_4 | F1_avg | AUC_1 | AUC_2 | AUC_3 | AUC_4 | AUC_avg |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Logistic Regression | train | 0.9514 | 0.9521 | 0.9470 | 0.9373 | 0.9690 | 0.9513 | 0.9521 | 0.9404 | 0.9406 | 0.9723 | 0.9513 | 0.9521 | 0.9437 | 0.9389 | 0.9706 | 0.9513 | 0.9960 | 0.9955 | 0.9923 | 0.9972 | 0.9952 |
| Logistic Regression | test | 0.9549 | 0.9710 | 0.9722 | 0.9333 | 0.9444 | 0.9553 | 0.9853 | 0.9333 | 0.9459 | 0.9577 | 0.9556 | 0.9781 | 0.9524 | 0.9396 | 0.9510 | 0.9553 | 0.9996 | 0.9976 | 0.9978 | 0.9984 | 0.9983 |
| SVC Classifier 1 | train | 0.9540 | 0.9556 | 0.9505 | 0.9406 | 0.9690 | 0.9539 | 0.9589 | 0.9439 | 0.9406 | 0.9723 | 0.9539 | 0.9573 | 0.9472 | 0.9406 | 0.9706 | 0.9539 | 0.9956 | 0.9954 | 0.9924 | 0.9973 | 0.9952 |
| SVC Classifier 1 | test | 0.9549 | 0.9710 | 0.9722 | 0.9333 | 0.9444 | 0.9553 | 0.9853 | 0.9333 | 0.9459 | 0.9577 | 0.9556 | 0.9781 | 0.9524 | 0.9396 | 0.9510 | 0.9553 | 0.9995 | 0.9976 | 0.9974 | 0.9983 | 0.9982 |
| SVC Classifier 2 | train | 0.9497 | 0.9583 | 0.9443 | 0.9247 | 0.9719 | 0.9498 | 0.9452 | 0.9509 | 0.9441 | 0.9585 | 0.9497 | 0.9517 | 0.9476 | 0.9343 | 0.9652 | 0.9497 | 0.9967 | 0.9955 | 0.9920 | 0.9970 | 0.9953 |
| SVC Classifier 2 | test | 0.9583 | 0.9571 | 0.9595 | 0.9351 | 0.9851 | 0.9592 | 0.9853 | 0.9467 | 0.9730 | 0.9296 | 0.9586 | 0.9710 | 0.9530 | 0.9536 | 0.9565 | 0.9585 | 0.9995 | 0.9969 | 0.9975 | 0.9982 | 0.9981 |
| Random Forest Classifier 1 | train | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| Random Forest Classifier 1 | test | 0.9410 | 0.9054 | 0.9571 | 0.9583 | 0.9444 | 0.9413 | 0.9853 | 0.8933 | 0.9324 | 0.9577 | 0.9422 | 0.9437 | 0.9241 | 0.9452 | 0.9510 | 0.9410 | 0.9989 | 0.9903 | 0.9939 | 0.9982 | 0.9953 |
| Random Forest Classifier 2 | train | 0.9627 | 0.9723 | 0.9514 | 0.9512 | 0.9757 | 0.9627 | 0.9623 | 0.9614 | 0.9545 | 0.9723 | 0.9626 | 0.9673 | 0.9564 | 0.9529 | 0.9740 | 0.9626 | 0.9993 | 0.9987 | 0.9983 | 0.9996 | 0.9990 |
| Random Forest Classifier 2 | test | 0.9479 | 0.9437 | 0.9577 | 0.9467 | 0.9437 | 0.9479 | 0.9853 | 0.9067 | 0.9595 | 0.9437 | 0.9488 | 0.9640 | 0.9315 | 0.9530 | 0.9437 | 0.9481 | 0.9989 | 0.9926 | 0.9969 | 0.9984 | 0.9967 |
| Random Forest Classifier 3 | train | 0.9583 | 0.9655 | 0.9507 | 0.9446 | 0.9723 | 0.9583 | 0.9589 | 0.9474 | 0.9545 | 0.9723 | 0.9583 | 0.9622 | 0.9490 | 0.9496 | 0.9723 | 0.9583 | 0.9988 | 0.9980 | 0.9972 | 0.9991 | 0.9983 |
| Random Forest Classifier 3 | test | 0.9583 | 0.9710 | 0.9595 | 0.9467 | 0.9571 | 0.9586 | 0.9853 | 0.9467 | 0.9595 | 0.9437 | 0.9588 | 0.9781 | 0.9530 | 0.9530 | 0.9504 | 0.9586 | 0.9989 | 0.9936 | 0.9972 | 0.9985 | 0.9970 |
| Neural Network Classifier 1 | train | 0.9071 | 0.9422 | 0.9485 | 0.9508 | 0.8129 | 0.9136 | 0.9486 | 0.9053 | 0.8112 | 0.9619 | 0.9068 | 0.9454 | 0.9264 | 0.8755 | 0.8811 | 0.9071 | 0.9937 | 0.9915 | 0.9710 | 0.9802 | 0.9841 |
| Neural Network Classifier 1 | test | 0.9028 | 0.9577 | 0.9710 | 0.9375 | 0.7738 | 0.9100 | 1.0000 | 0.8933 | 0.8108 | 0.9155 | 0.9049 | 0.9784 | 0.9306 | 0.8696 | 0.8387 | 0.9043 | 0.9999 | 0.9942 | 0.9829 | 0.9736 | 0.9876 |
| Neural Network Classifier 2 | train | 0.9219 | 0.9507 | 0.8856 | 0.9598 | 0.9010 | 0.9243 | 0.9247 | 0.9509 | 0.8357 | 0.9758 | 0.9217 | 0.9375 | 0.9171 | 0.8935 | 0.9369 | 0.9212 | 0.9878 | 0.9833 | 0.9693 | 0.9896 | 0.9825 |
| Neural Network Classifier 2 | test | 0.9236 | 0.9714 | 0.9452 | 0.9538 | 0.8375 | 0.9270 | 1.0000 | 0.9200 | 0.8378 | 0.9437 | 0.9254 | 0.9855 | 0.9324 | 0.8921 | 0.8874 | 0.9244 | 1.0000 | 0.9965 | 0.9873 | 0.9833 | 0.9918 |
| Neural Network Classifier 3 | train | 0.9453 | 0.9589 | 0.9462 | 0.9301 | 0.9458 | 0.9452 | 0.9589 | 0.9263 | 0.9301 | 0.9654 | 0.9452 | 0.9589 | 0.9362 | 0.9301 | 0.9555 | 0.9452 | 0.9932 | 0.9916 | 0.9886 | 0.9962 | 0.9924 |
| Neural Network Classifier 3 | test | 0.9514 | 0.9706 | 0.9722 | 0.9221 | 0.9437 | 0.9521 | 0.9706 | 0.9333 | 0.9595 | 0.9437 | 0.9518 | 0.9706 | 0.9524 | 0.9404 | 0.9437 | 0.9518 | 0.9994 | 0.9971 | 0.9970 | 0.9969 | 0.9976 |

5. For each dataset generate plots like the ones below - to understand the classification boundaries and the overall performance of the classifiers:



6. Compare the metrics within and across the datasets (train and test) and algorithms. In addition to the variations in metrics, include in your report aspects related to classification boundaries, overfitting, etc.
7. List your major learnings from this exercise.
8. Your submissions should include the following:
    a. A PDF document with all the above analyses and comments. Ensure that you include the required figures and Tables (ie. metrics data) in your report, along with the explanations and analysis.
    b. Your Python source file (.py file). Please DO NOT upload Jupyter Notebooks – they get bulky!
    c. Name of the PDF should be **E4-your-roll-number.pdf** and the name of the Python source file should be **E4-your-roll-number.py**
    d. Upload the PDF and the source file to the assignment submission point E4.

**oooOOOooo**