Activity 2

Classification with SVM, BP and MLR

- **Git Repository**

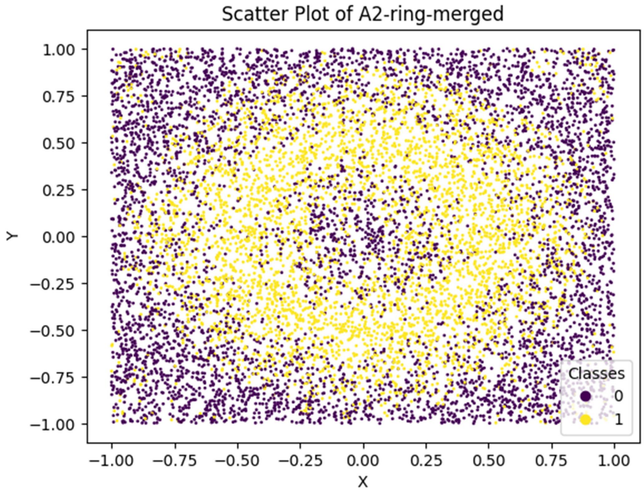
  https://github.com/YoussefEzz/Classification-SVM-BP-and-LR

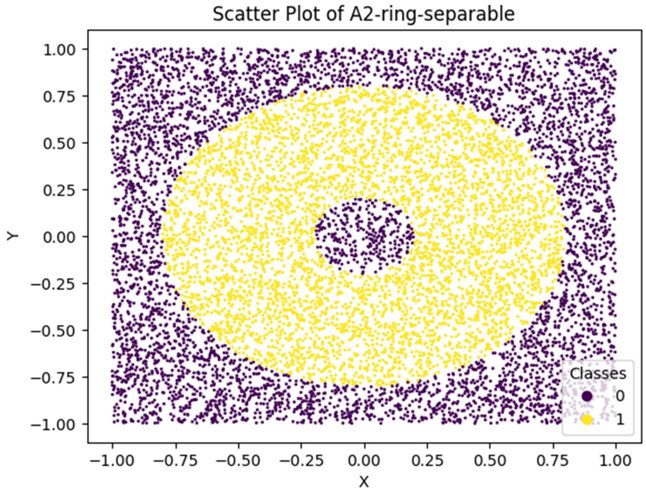
- **Part 1 : Selecting and analyzing the datasets**

  Since we do not want to give a priori more importance to some of the input variables w.r.t. the others, we should scale all of them to the same range of variation.

  1. **Ring datasets A2-ring-merged.txt and A2-ring-separable.txt analyzed in Ring_Datasets.ipynb**

     since the two input variables(call them x and y) lie in the same range [-1.0, 1.0], No pre-processing is needed. Both Training sets have the same two input feature values but with different output values(class labels) such that the plot of A2-ring-merged shows that the class of points are emerged but the plot of A2-ring-separable shows that the class of points are separable





  2. **Bank Dataset bank-additional.csv analyzed in Bank_Datasets.ipynb**

     the dataset contains 20 input variables and one output variable(y) ,so total 21 columns.

     - 10 numerical columns
       - 5 integer columns e.g. (age, duration)
       - 5 float columns e.g. (emp.var.rate, cons.price.idx)
     - 11 categorical values
       - 9 nominal columns(No particular order) e.g. (marital, education)
       - 2 ordinal columns(some ordered) e.g. (month, day_of_week)

     As per https://www.kaggle.com/code/pythonafroz/categorical-to-numerical-encoding-methods

Unique values for each column

```
Unique values for column 'age':
[30 39 25 38 47 32 41 31 35 36 29 27 44 46 45 50 55 40 28 34 33 51 48 20
 76 56 24 58 60 37 52 42 49 54 59 57 43 53 75 82 71 21 22 23 26 81 61 67
 73 18 64 74 77 86 85 63 88 78 72 68 80 66 19 62 65 69 70]

Unique values for column 'job':
['blue-collar' 'services' 'admin.' 'entrepreneur' 'self-employed'
 'technician' 'management' 'student' 'retired' 'housemaid' 'unemployed'
 'unknown']

Unique values for column 'marital':
['married' 'single' 'divorced' 'unknown']

Unique values for column 'education':
['basic.9y' 'high.school' 'university.degree' 'professional.course'
 'basic.6y' 'basic.4y' 'unknown' 'illiterate']

Unique values for column 'default':
['no' 'unknown' 'yes']

Unique values for column 'housing':
['yes' 'no' 'unknown']

Unique values for column 'loan':
['no' 'unknown' 'yes']
...

Unique values for column 'y':
['no' 'yes']
```

datatype for each column

```
df.dtypes
[34] ✓ 0.0s
...
age                 int64
job                object
marital            object
education          object
default            object
housing            object
loan               object
contact            object
month              object
day_of_week        object
duration            int64
campaign            int64
pdays               int64
previous            int64
poutcome           object
emp.var.rate      float64
cons.price.idx    float64
cons.conf.idx     float64
euribor3m         float64
nr.employed       float64
y                  object
```

to preprocess the data

a) Drop rows with missing information tagged as “unknown” in any column
b) categorical columns : encode as ordinal using category_encoders

```
# encode 10 input categorical features

    encoder = ce.OrdinalEncoder(cols=['marital', 'job', 'education', 'default', 'housing',
'loan', 'contact', 'month', 'day_of_week', 'poutcome'])

    df_normalized = encoder.fit_transform(df_normalized)
# ... decode to view and test
    df_normalized_reversed = encoder.inverse_transform(df_normalized)
```

c) numerical float columns : encode by scaling from -1 to 1

```
# encode 5 input numerical float features by scaling from -1 to 1
    scaler = MinMaxScaler(feature_range=(-1, 1))
    columns_to_scale = ['emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m',
'nr.employed']
    data_to_scale = df_normalized[columns_to_scale]
    scaler.fit(data_to_scale)
    scaled_data = scaler.transform(data_to_scale)
    df_normalized[columns_to_scale] = scaled_data
```

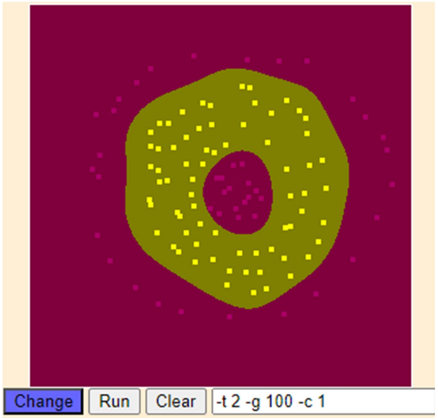
d) output class label column : encode by replacing yes with 1 and no with 0

```
# encode output as 1 for yes and 0 for no
    df_normalized["y"].replace({"yes":1 ,"no":0 } , inplace=True)
```

- **Part 2 : Classification problem**

1. **SVM classification model for Ring Dataset in SVM_ring_separable.py**

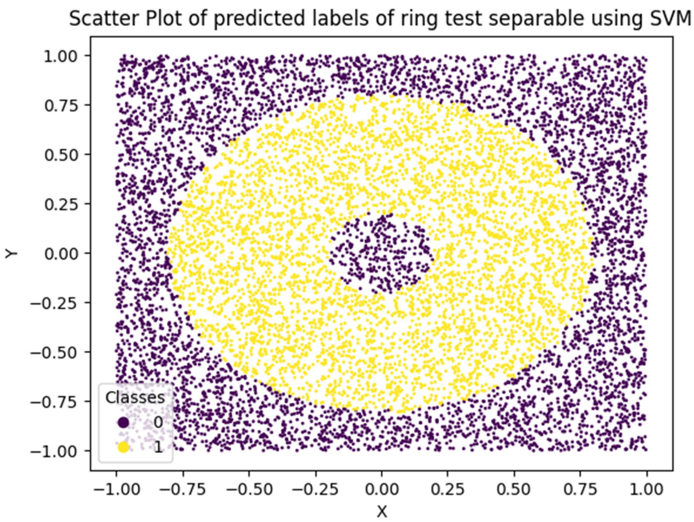
Libsvm library is used to train the model using the training set of **A2-ring-separable.txt** with the default parameter **-t** for kernel type 2 -- radial basis function: exp(-gamma*|u-v|^2), but gamma variable **-g** had to be tuned to be 50 – 200 (default 1/num_features) to maintain accuracy above 99%.

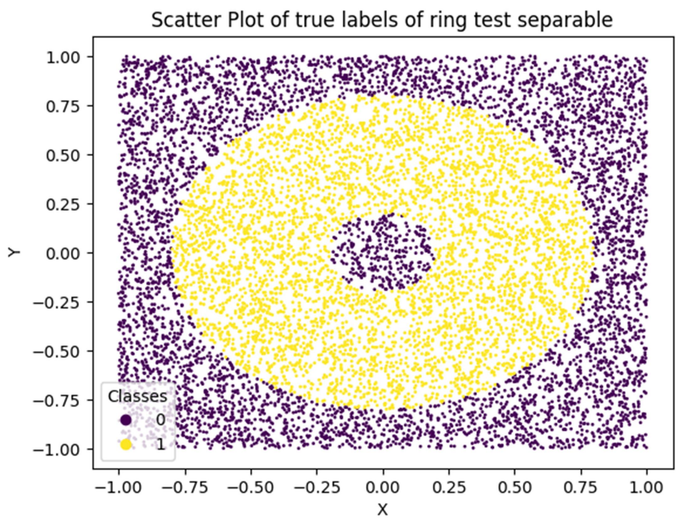


Graphic Interface applet on LiBsvm website https://www.csie.ntu.edu.tw/~cjlin/libsvm/ helped select the value of gamma to obtain below results.

```
optimization finished, #iter = 18591
nu = 0.008091
obj = -5064.815069, rho = -0.165202
nSV = 1438, nBSV = 23
Total nSV = 1438
Accuracy = 99.38% (9938/10000) (classification)
```

Comparison between predicted labels of test set obtained in **SVM_ring_separable.py** and true labels obtained in **Ring_Datasets.ipynb**

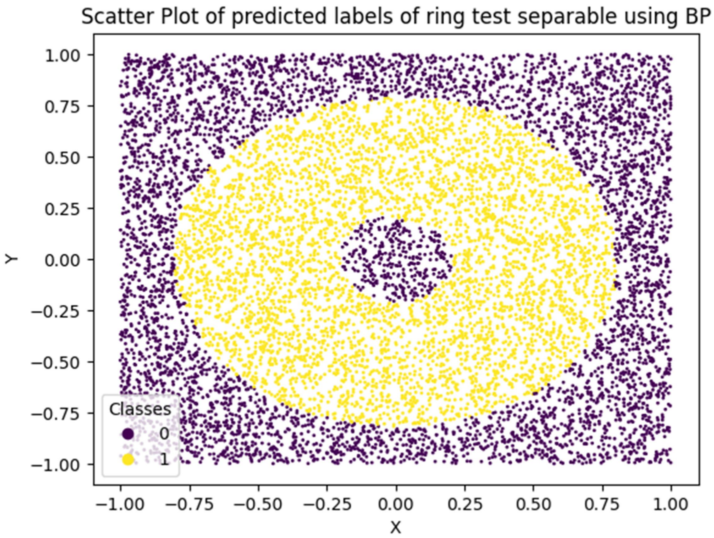




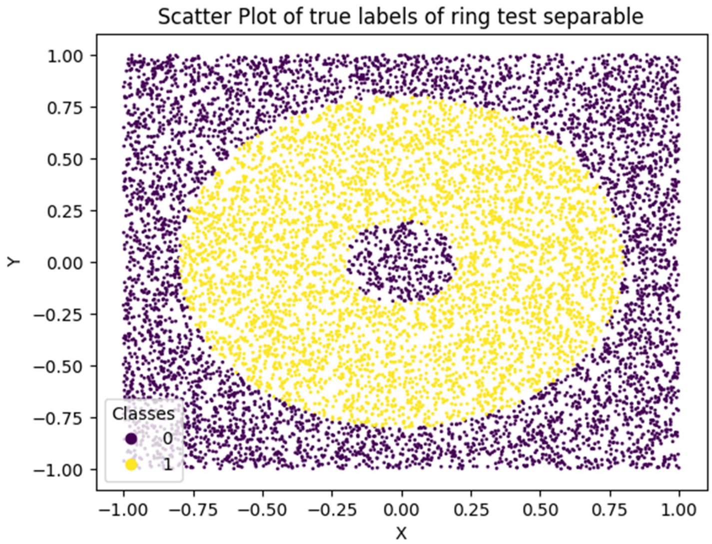
2. **BP classification model for Ring Dataset in SVM_ring_separable.ipynb**

TensorFlow library is used to train the model using the training set of **A2-ring-separable.txt** with parameters :

- 1 input layer 2 * 10000
- 3 hidden layers with sizes 100, 50 and 25
- 1 output layer with size 1
- Learning rate 0.01
- Momentum 0.9
- Number of epocs 100



Scatter Plot of predicted labels of ring test separable using BP
Classes
0
1
X
Y



Scatter Plot of true labels of ring test separable
Classes
0
1
X
Y

3.