**Activity 2**

**Classification with SVM, BP and MLR**

* **Git Repository**

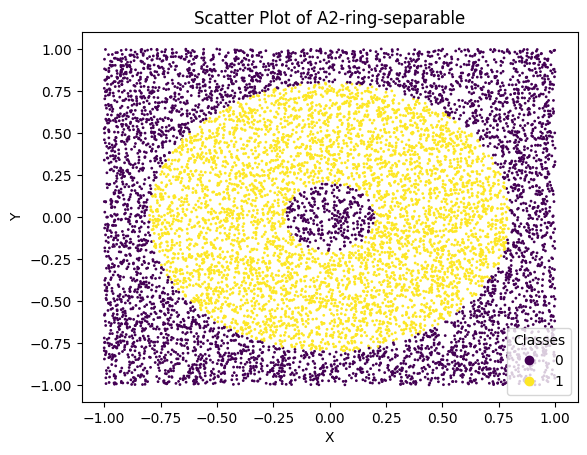
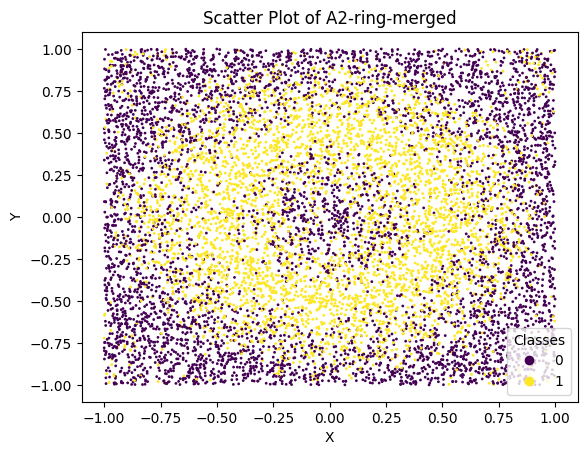
[https://github.com/YoussefEzz/Classification-SVM-BP-and-LR](https://github.com/YoussefEzz/Prediction-BP-and-LR)

* **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

****

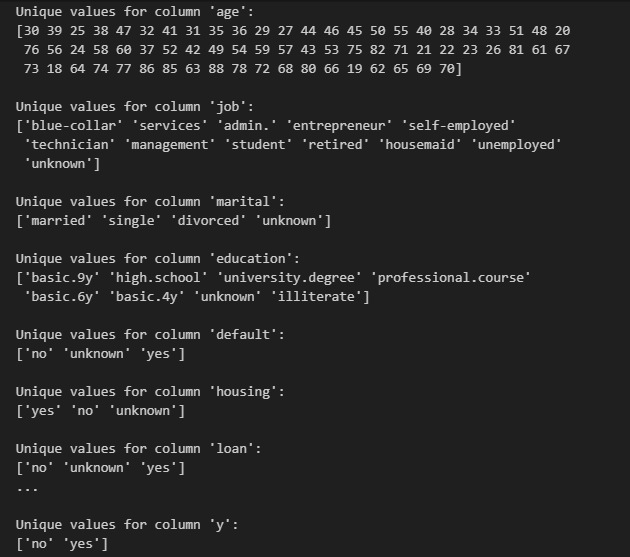
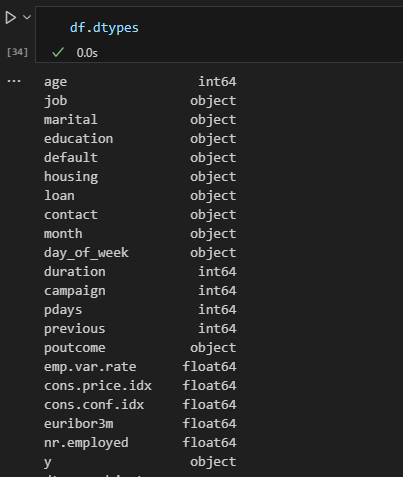
1. **Bank Dataset bank-additional.csv analyzed in Bank\_Datasets.ipynb**

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 datatype for each column



to preprocess the data

1. Drop rows with missing information tagged as “unknown” in any column
2. 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)

1. 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)

scaled\_df = pd.DataFrame(scaled\_data, columns=columns\_to\_scale)

df\_normalized[columns\_to\_scale] = scaled\_df

1. 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)