SYNTHETIC DATA CLASSIFICATION:
HALF-MOONS AND TWO-SPIRAL DATASETS

Madhukshara Sarkar (B18735)

Ramakrishna Mission Vivekananda Educational and Research Institute

M.Sc. Big Data Analytics

July 2019

1 INTRODUCTION

The evaluation of the performance of a machine learning method is an important task. It helps to find an appropriate machine learning technique for a certain problem [1]. But data is expensive. Rarely data is available freely. Taking the most typical machine learning tasks, classification and regression, as an example, limited and insufficient samples cause low generalization of machine learning models, which cannot provide reasonable predictions. Data are insufficient either because of sample rarity or because data are impeded to be accessed for privacy concerns or confidential protection. However, it is possible to test the performances of different techniques on synthetic datasets.

In this paper, we evaluate the performances of two common classifiers – Support Vector Machine and Multilayer Perceptron on artificially generated data. To evaluate the methods, we have written python scripts which generate artificial datasets with different characteristics. The scripts allow the user to manipulate some parameters, so that the resulting complexity of datasets vary. The goal was to construct datasets, which would allow to evaluate the performance of the chosen data mining methods.

The rest of the paper is organised as follows: Section 2 describes some theoretical foundations of the classifiers and parameter tuning methods, Section 3 discusses the performance evaluation metrics used, Section 4 presents the result classification of the generated datasets and Section 5 includes final discussion and conclusion.

1.1 DATASET DESCRIPTION

In this study, we work with two artificially generated datasets: the half-moons data and the two-spiral data.

1.1.1 Half-Moons Dataset

The half-moons data is actually a data structured as two intervening half circles. Here each of the moons is a class, i.e. a data point belongs to either of the half circles. We consider 3 different cases by varying the complexity and linearity of these half circles.

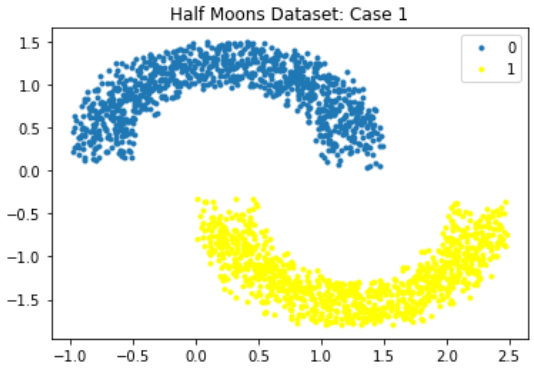


Figure 1(a)

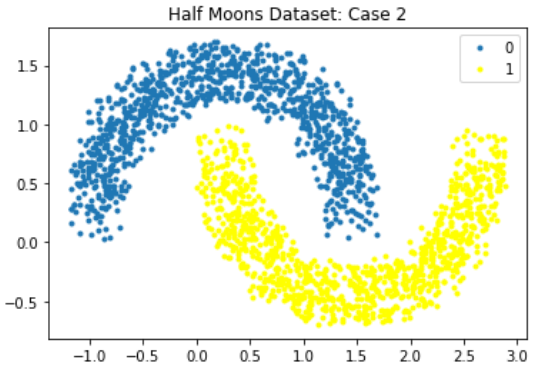


Figure 1(b)

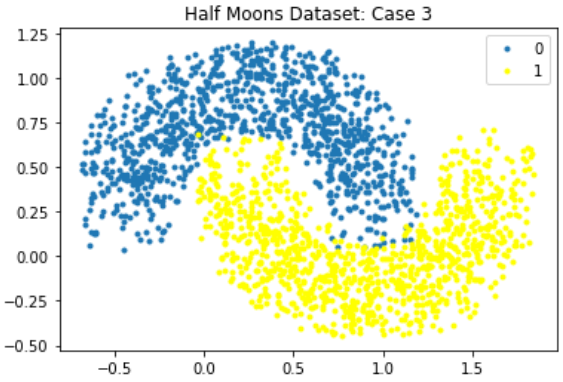


Figure 1(c)

Figure 1(a) shows a half-moon dataset where the two half circles are at significant distance. Figure 1(b) shows a half-moon dataset where the two half circles are somewhat intervening. However, a clear demarcation between the two can be made, i.e. there are no areas of overlap between the data points belonging to the two moons. Figure 1(c) shows a more complex scenario where the two half circles are very close to each other. There are several areas where the data points of the two classes overlap.

The task of the classifiers would be to classify a test data point into one of the half-moons.

1.1.2 Two-Spiral Dataset

The two-spiral dataset was developed by Lang and Witbrock [2]. It contains two different spirals in the same space. Here each of the spirals is a class, i.e. a data point belongs to either of the spirals. We consider 3 different cases by varying the noise of the spirals data.

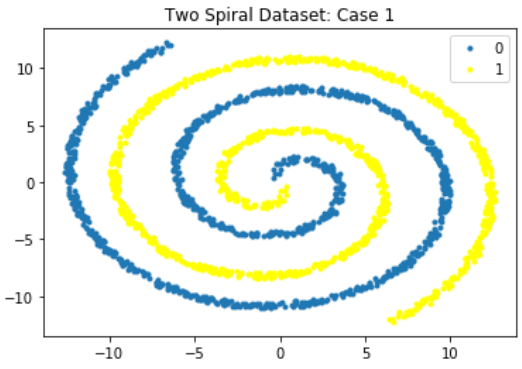


Figure 2(a)

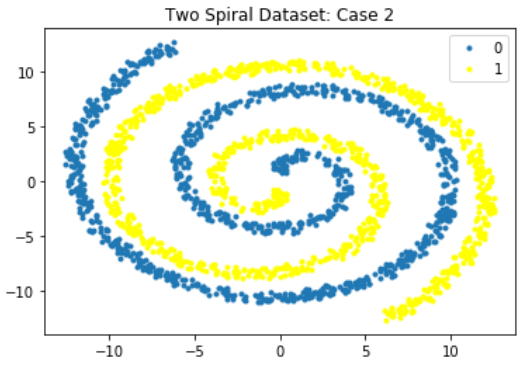


Figure 2(b)

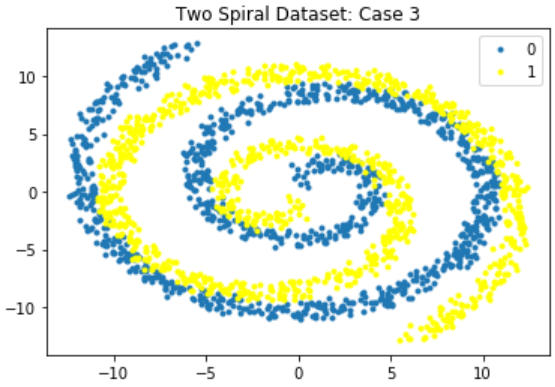


Figure 2(c)

Figure 2(a) shows two clearly distinct spirals. Figure 2(b) shows two spirals with slightly greater noise. However, a clear demarcation between the two can be made, i.e. there are no areas of overlap between the data points belonging to the spirals. Figure 2(c) shows a two-spiral data with much higher noise. There are several areas of overlap among data points belonging to the two different spirals.

The task of the classifiers would be to classify a test data point into one of the spirals.

2 METHODS

2.1 SUPPORT VECTOR MACHINE

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labelled training data, the algorithm outputs an optimal hyperplane which categorizes new examples.

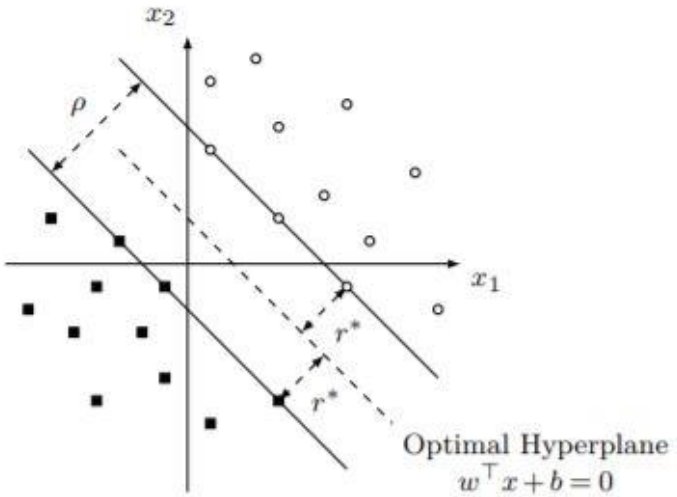


Figure 3: Linearly separable training set with hyperplane and maximum margin ρ [3]

Let $(\overrightarrow{x_i}, y_i)$ be given, where, $\overrightarrow{x_i} \epsilon \mathbb{R}^m$, $\forall i = 1,2, \dots, m$ is m dimensional feature vectors and $y_i \in \{-1,1\}$ is the class label of $\overrightarrow{x_i}, \forall i$, where n is the number of samples in the training set. Let $\vec{w} = [w_1, w_2, \cdots, w_m]$ be the m dimensional weight vector. The feature vector x_i for which $y_i(w^T x_i + b) = 1, \forall\, i = 1,2, \dots, m$, are known as the support vectors. Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

The support vector machines require the solution of the following optimization problem:

$$\min \ \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i$$

$$\text{subject to } y_i(w^T \phi(x_i) + b) \ \geq 1 - \xi_i \ ; \ \xi_i \geq 0$$

2.1.1 Kernel function

A kernel function is based on the inner product between the given data to a feature space of higher (or infinite) dimension. The scalar product is then transformed nonlinear. Here training vectors x_i are mapped to higher (maybe infinite) dimensional space by the function ϕ. SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. $C > 0$ is the penalty parameter of the error term. Furthermore $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$ is called the kernel function [3]. The four basic kernels are:

- Linear: $K(x_i, x_j) = {x_i}^T x_j$
- Polynomial: $K(x_i, x_j) = (\gamma {x_i}^T x_j + r)^d, \gamma > 0$
- Radial basis function (RBF): $K(x_i, x_j) = \exp\left(-\gamma \left\| x_i - x_j \right\|^2\right), \gamma > 0$
- Sigmoid: $K(x_i, x_j) = \tanh(\gamma {x_i}^T x_j + r)$

Here, γ, r and d are kernel parameters.

2.2 MULTILAYER PERCEPTRON

A multilayer perceptron (MLP) is a deep, artificial neural network. It is composed of more than one perceptron. They are composed of an input layer to receive the signal, an output layer that makes a decision or prediction about the input, and in between those two, an arbitrary number of hidden layers that are the true computational engine of the MLP. MLPs with one hidden layer are capable of approximating any continuous function.

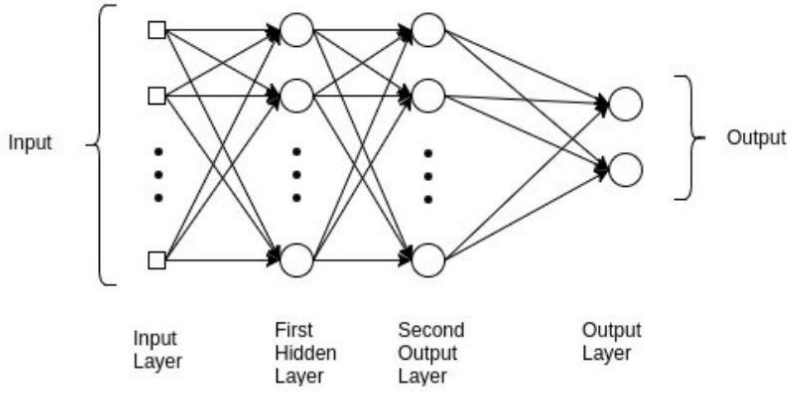


Figure 4: Multilayer Perceptron (MLP) or Feedforward Neural Network (FFNN) with Two Hidden Layers [4]

The input layer takes the external input $x = [x_1, x_2, \dots, x_{n1}]^T$, which is an n_1-dimensional feature vector of the training sample set. The network represents a function from $\mathbb{R}^{n1} \longrightarrow \mathbb{R}^{n2}$ in the L^{th} layer, where L is the number of layers and n_2 denotes the number of nodes in the output layer.

We define,

n_l : number of nodes in l^{th} layer, $\forall\ l = 1,2, \dots, L$

y_i^l : output of i^{th} node in layer l, $\forall\ l = 1,2, \dots, L\ and\ i = 1,2, \dots, n_l$

w_{ij}^l : weight of the connection from i^{th} node in layer l to j^{th} node in layer $l + 1$

S_i^l : net input of i^{th} node in layer l, where net input of a node is weighted sum of the nodes in the previous layer

The nodes in the hidden layers and the output layer calculate the net input and use an activation function to compute the outputs $y_i^l, \forall\ l = 1,2, \dots, L\ and\ i = 1,2, \dots, n_l$.

The output of a node, say j at layer l is computed as

$$y_j^l = f(S_j^l)$$

where $f(\cdot)$ is the activation function.

The objective of the multilayer perceptron algorithm is to learn the weights

$$W = \left\{w_{ij}^l \mid i = 0,1, \dots, n_l; j = 1,2, \dots, n_{l+1}; l = 1,2, \dots, L\right\}$$

using the training samples.

2.2.1 Activation function

The computation of the y for each neuron of the multilayer perceptron requires knowledge of the derivative of the activation function $f(\cdot)$ associated with that neuron. For this

derivative to exist, we require the function $f(\cdot)$ to be continuous [5]. Every activation function (or *non-linearity*) takes a single number and performs a certain fixed mathematical operation on it. Some commonly used activation functions are:

- Logistic function: $f(x) = \frac{1}{1+e^{-x}}$
- tanh (Hyperbolic Tangent function): $f(x) = \tanh(x)$
- ReLU (Rectified Linear Unit function): $f(x) = \max(0, x)$

2.3 Parameter Tuning and Cross Validation Algorithms

In real-life applications, estimators used by practitioners always depend on unknown parameters that have to be chosen, which is called "parameter tuning". For instance, when estimating a density with a histogram (or a kernel estimator), the partition (resp. the bandwidth) has to be specified beforehand. Other instances are LASSO and SVM algorithms: They both depend on a "regularization parameter" that has to be chosen by practitioners. Moreover, the final performance of the estimator crucially depends on that unknown parameter. This tuning step is often performed by use of Cross-Validation (CV) in practice, but without any real guaranty of performance [6].
Cross-validation (CV) is a popular strategy for algorithm selection. It is a statistical method used to estimate the skill of machine learning models. The main idea behind CV is to split data, once or several times, for estimating the risk of each algorithm: Part of data (the training sample) is used for training each algorithm, and the remaining part (the validation sample) is used for estimating the risk of the algorithm. Then, CV selects the algorithm with the smallest estimated risk [7]. The widely used forms of cross-validation are leave one out method and k-fold cross-validation technique. In this study we have made use of the k-fold cross validation technique to obtain the optimum values of the hyper-parameter of the classifiers.

In this study we use the grid search technique in Python Scikit-Learn Module for cross validation purpose.

3 Evaluation Criteria

Evaluation of classification algorithms is one of the key points in any process of data mining. The most commonly tools used in analysing the results of classification algorithms applied is the confusion matrix [8].

The confusion matrix displays the number of correct and incorrect classifications made by the classifier as compared to the actual classifications in the data [8]. A general form of a confusion matrix is presented below:

	Classes Predicted		
Actual Classes		Positive	Negative
	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Here *true positive* (TP) counts the number of data points correctly predicted to the positive class. *False positive* (FP) counts the number of data points that actually belong to the negative class, but predicted as positive (i.e., *falsely predicted as positive*). *False negative* (FN) counts the number of data points that actually belong to the positive class, but predicted as negative (i.e., *falsely predicted as negative*). *True negative* (TN) counts the number of data points correctly predicted to the negative class.
The accuracy is the proportion of the total number of correct predictions and calculated as the ratio between the number of cases correctly classified and the total number of cases.

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$$

The error indicates the proportion of classes classified incorrectly.

$$\text{Error} = 1 - Accuracy$$

The performance of a classifier in classifying the feature vectors can be measured by its precision, recall and f-measure.

The precision gives a measure of the proportion of positive samples which are correctly classified.

$$\text{Precision} = \frac{TP}{TP+FP}$$

Note that, if FP = 0, then Precision = 1 i.e., all negative samples are classified correctly.

The recall gives a measure of the proportion of correct classification amongst all positive classes.

$$\text{Recall} = \frac{TP}{TP+FN}$$

If FN = 0, then Recall = 1 i.e., all positive samples are classified correctly.

F-measure combines precision and sensitivity as the harmonic mean of the two parameters.

$$\text{F-measure} = \frac{2 \times Recall \times Precision}{Recall+Precision}$$

The closer the values of precision and recall, the higher is the f-measure. F-measure becomes 1 when the values of precision and recall are 1 and it becomes 0 when precision is 0, or recall is 0, or both are 0. Thus f-measure lies between 0 and 1. A high f-measure value is desirable for good classification.

The above measures are corresponding to a two class classification problem. But they can be extended in a generalised fashion for an m-class classification problem. There are two conventional methods to evaluate the performance of a classifier aggregated over all classes, namely *macro-averaging* and *micro-averaging*.
Macro-averaging gives equal weight to each class, whereas micro-averaging gives equal weight to each per-document classification decision. Because the f-measure ignores true negatives and its magnitude is mostly determined by the number of true positives, large classes dominate small classes in micro-averaging.

The macro-averaged precision and recall for a set of m classes are computed as follows:

$$\text{Macro-averaged Precision} = \frac{1}{m}\sum_{i=1}^{m}\frac{TP_i}{TP_i + FP_i}$$

$$\text{Macro-averaged Recall} = \frac{1}{m}\sum_{i=1}^{m}\frac{TP_i}{TP_i + FN_i}$$

The micro-averaged precision and recall for a set of m classes are computed as follows:

$$\text{Micro-averaged Precision} = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m} TP_i + \sum_{i=1}^{m} FP_i}$$

$$\text{Micro-averaged Recall} = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m} TP_i + \sum_{i=1}^{m} FN_i}$$

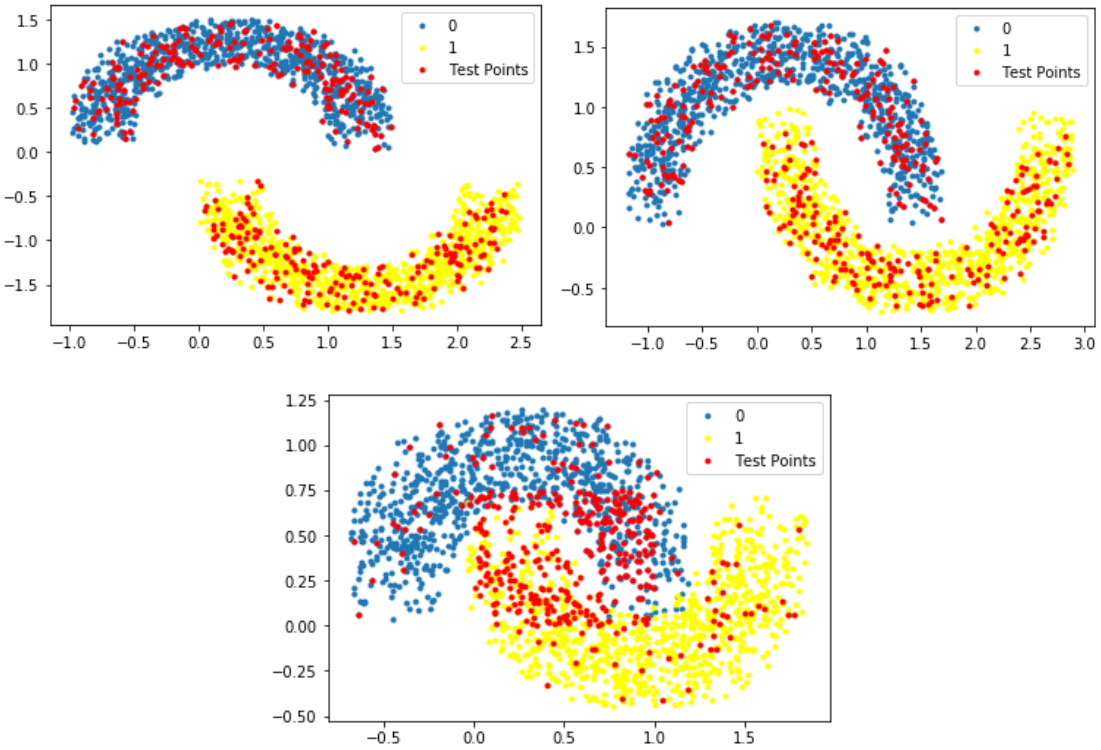
4 Analysis of Results

We generate two artificial data: the half-moons data and the two-spiral data and evaluate the performances of the Support Vector Machine and Multilayer Perceptron on these data.

4.1 Half-Moons Data

We split the data into training and testing set in 80-20% proportions. For the first two cases (see Figures 1(a) and 1(b)) the test samples are randomly selected from all over the dataset. For the third case, i.e. where there is considerable overlap in the data, out of the 20% test samples, 15% are chosen from the area whose 'y' coordinates range in [0.00, 0.75] in figure 1(c), and remaining 5% from the rest of the dataset. The aim of such splitting is to evaluate the performance of the classifiers in the prime area of overlap.

The distribution of the test points within the overall data for each of the three cases is shown below



We shall now see the results of the classifiers.

4.1.1 Support Vector Machine

We apply the SVM classifier on the data for each of the cases using a variety of kernel functions. We perform a 5-fold cross validation to obtain the optimum value of parameter C.

In the grid search, we take, C: [0.001, 0.01, 0.1, 10, 100].

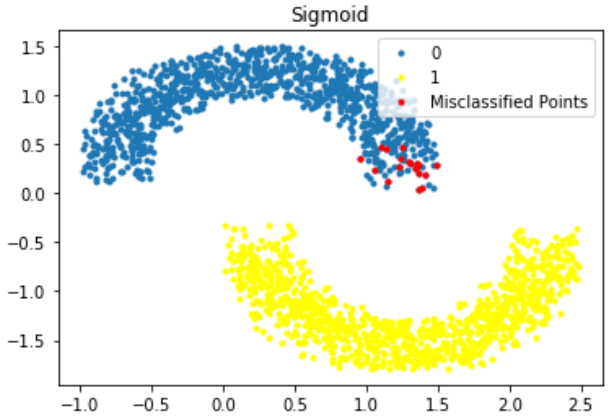
4.1.1.1 Case 1

The results obtained are

Kernel	RBF	Sigmoid	Linear	Polynomial
Optimum C	0.01	0.01	0.01	10
Confusion Matrix	$\begin{bmatrix} 205 & 0 \\ 0 & 195 \end{bmatrix}$	$\begin{bmatrix} 187 & 18 \\ 0 & 195 \end{bmatrix}$	$\begin{bmatrix} 205 & 0 \\ 0 & 195 \end{bmatrix}$	$\begin{bmatrix} 205 & 0 \\ 0 & 195 \end{bmatrix}$
Precision	1.0	0.9577	1.0	1.0
Recall	1.0	0.9561	1.0	1.0
F-Score	**1.0**	0.9549	**1.0**	**1.0**

The RBF, linear and polynomial kernels classify the dataset perfectly.

The data points misclassified by the sigmoid kernel are

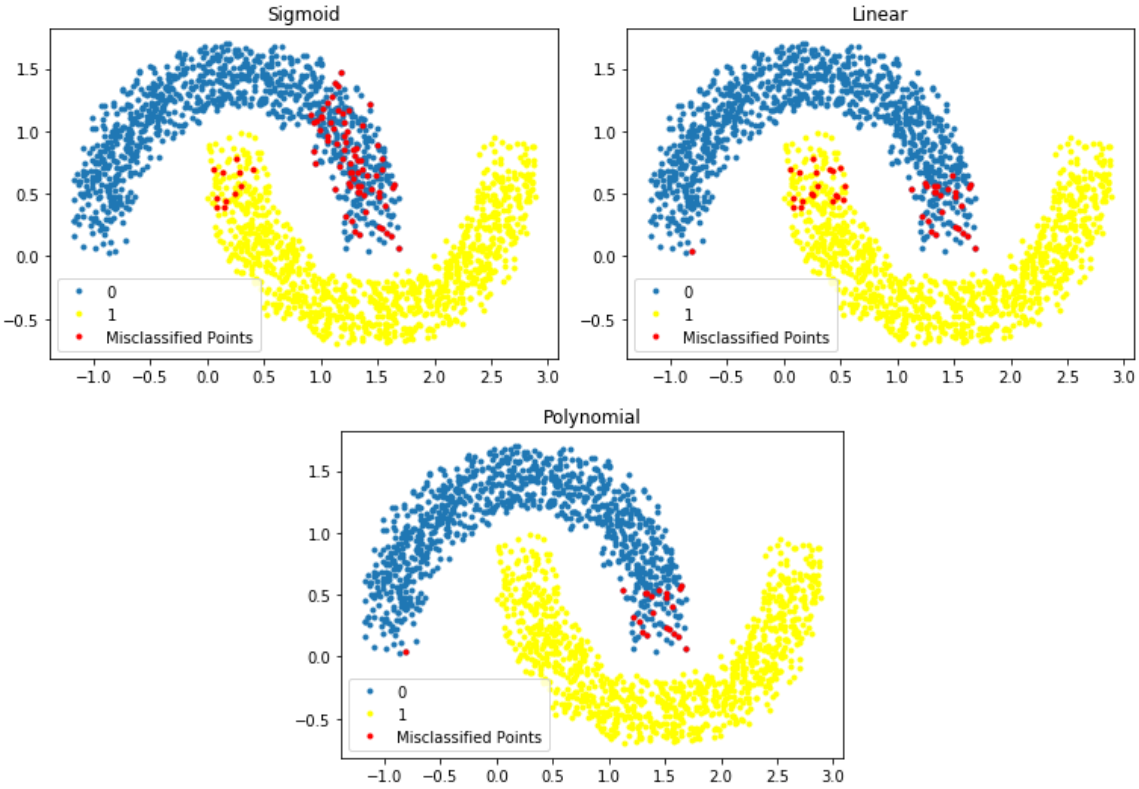


4.1.1.2 Case 2

The results obtained are

Kernel	RBF	Sigmoid	Linear	Polynomial
Optimum C	10	0.01	100	100
Confusion Matrix	$\begin{bmatrix} 205 & 0 \\ 0 & 195 \end{bmatrix}$	$\begin{bmatrix} 134 & 71 \\ 11 & 184 \end{bmatrix}$	$\begin{bmatrix} 179 & 26 \\ 19 & 176 \end{bmatrix}$	$\begin{bmatrix} 184 & 21 \\ 0 & 195 \end{bmatrix}$
Precision	1.0	0.8229	0.8877	0.9514
Recall	1.0	0.7986	0.8879	0.9488
F-Score	**1.0**	0.7917	0.8875	0.9475

The misclassified test samples (for kernels Sigmoid, Linear and Polynomial) are



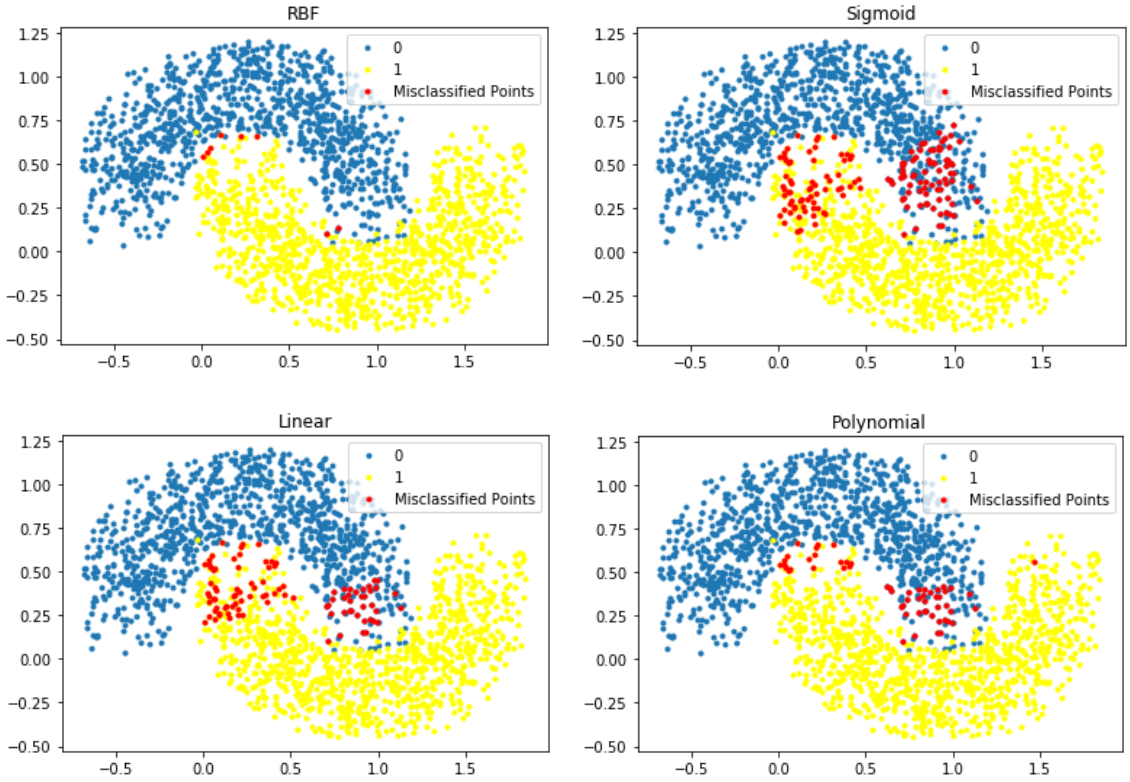
SVM classifies the data perfectly when using RBF kernel. The next best performance is given by the polynomial kernel.

4.1.1.3 Case 3

The results obtained are

Kernel	RBF	Sigmoid	Linear	Polynomial
Optimum C	100	0.01	100	10
Confusion Matrix	$\begin{bmatrix} 201 & 2 \\ 6 & 191 \end{bmatrix}$	$\begin{bmatrix} 126 & 77 \\ 67 & 130 \end{bmatrix}$	$\begin{bmatrix} 167 & 36 \\ 60 & 137 \end{bmatrix}$	$\begin{bmatrix} 166 & 37 \\ 20 & 177 \end{bmatrix}$
Precision	0.9803	0.6404	0.7638	0.8598
Recall	0.9798	0.6403	0.7590	0.8581
F-Score	**0.9799**	0.6399	0.7584	0.8574

The misclassified test samples are



The best performance is given by the RBF kernel.

4.1.2 Multilayer Perceptron

We apply the MLP classifier on the data for each of the cases using a variety of activation functions. We perform a 5-fold cross validation to obtain the optimum value of parameter alpha.

We use 4 layers, meaning the network contains 1 input layer, 2 hidden layers and 1 output layer. The input layer has 2 nodes. The first hidden layer has 3 nodes and the second hidden layer has 2 nodes. Finally, the output layer has 1 node. That is, we build a {2, 3, 2, 1} neural network.

In the grid search, we take, alpha: [0.001, 0.01, 0.1, 10, 100].

4.1.2.1 Case 1

The results obtained are

Activation function	Logistic	tanh	ReLU
Optimum alpha	0.001	0.001	0.01
Confusion Matrix	$\begin{bmatrix} 205 & 0 \\ 0 & 195 \end{bmatrix}$	$\begin{bmatrix} 205 & 0 \\ 0 & 195 \end{bmatrix}$	$\begin{bmatrix} 205 & 0 \\ 0 & 195 \end{bmatrix}$
Precision	1.0	1.0	1.0
Recall	1.0	1.0	1.0
F-Score	**1.0**	**1.0**	**1.0**

There are no misclassified test samples for any of the kernels.

Hence, we can say that MLP performs perfectly for this dataset.

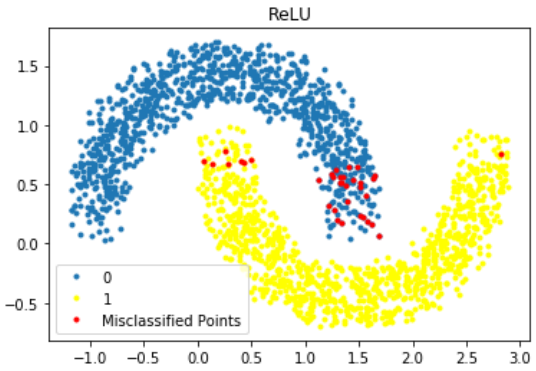
4.1.2.2 Case 2

The results obtained are

Activation function	Logistic	tanh	ReLU
Optimum alpha	0.1	0.1	0.01
Confusion Matrix	$\begin{bmatrix} 205 & 0 \\ 0 & 195 \end{bmatrix}$	$\begin{bmatrix} 205 & 0 \\ 0 & 195 \end{bmatrix}$	$\begin{bmatrix} 178 & 27 \\ 8 & 187 \end{bmatrix}$
Precision	1.0	1.0	0.9154
Recall	1.0	1.0	0.9136
F-Score	**1.0**	**1.0**	0.9125

MLP perfectly classifies the data while applying logistic or tanh function.

The misclassified test samples while applying ReLU activation function are



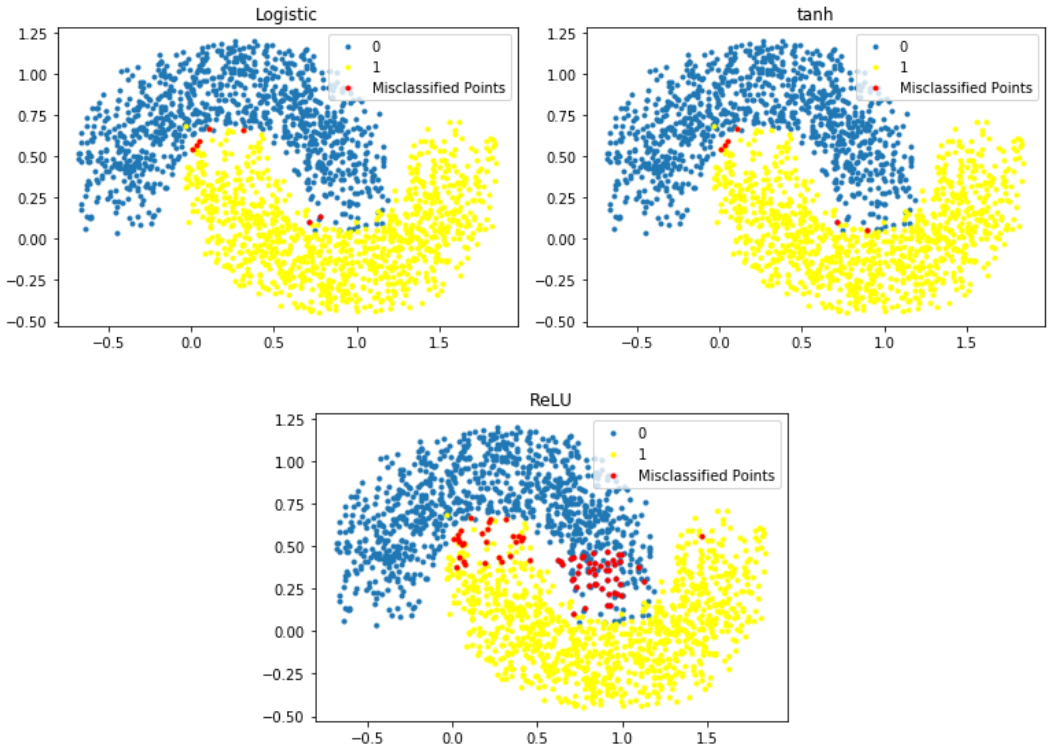
MLP gives perfect classification results while using the sigmoid or tanh activation function.

4.1.2.3 Case 3

The results obtained are

Activation function	Logistic	tanh	ReLU
Optimum alpha	0.1	0.01	0.1
Confusion Matrix	$\begin{bmatrix} 201 & 2 \\ 5 & 192 \end{bmatrix}$	$\begin{bmatrix} 202 & 1 \\ 5 & 192 \end{bmatrix}$	$\begin{bmatrix} 158 & 45 \\ 29 & 168 \end{bmatrix}$
Precision	0.9827	0.9853	0.8168
Recall	0.9824	0.9848	0.8156
F-Score	0.9825	**0.9849**	0.8149

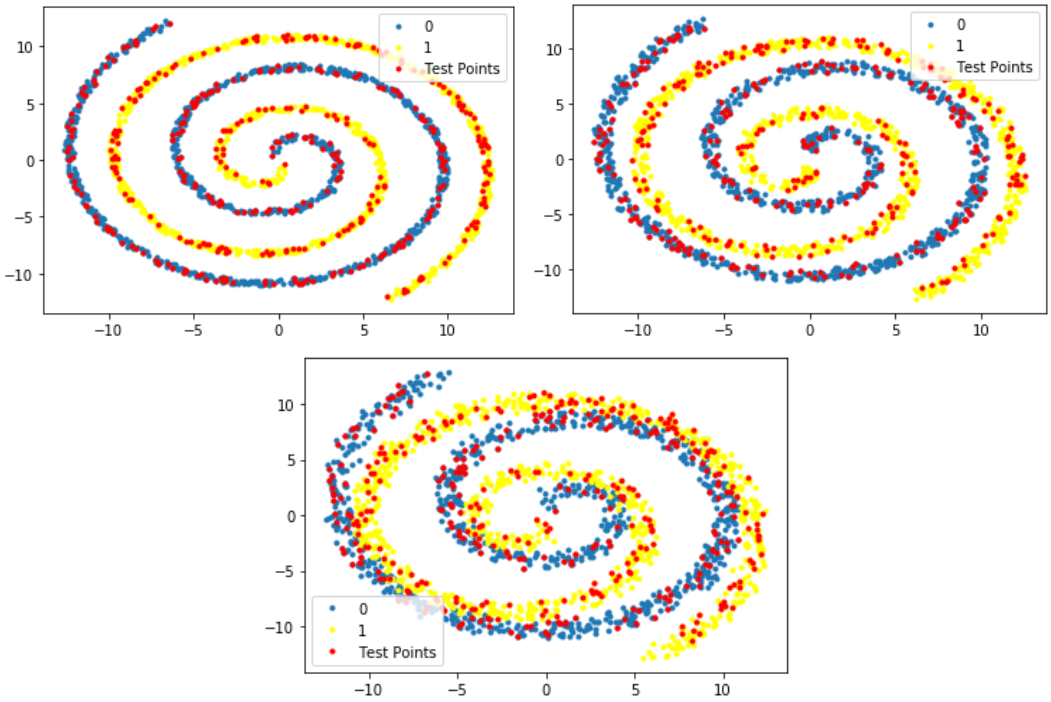
The misclassified test samples are



MLP gives best classification results while using the tanh activation function.

4.2 SPIRAL DATA

We split the data, randomly, into training and testing set in 80-20% proportions. The distribution of the test points within the overall data for each of the three cases is shown below



We shall now see the results of the classifiers.

4.2.1 Support Vector Machine

We apply the SVM classifier on the data for each of the cases using a variety of kernel functions. We perform a 5-fold cross validation to obtain the optimum value of parameter C.

In the grid search, we take, C: [0.001, 0.01, 0.1, 10, 100].

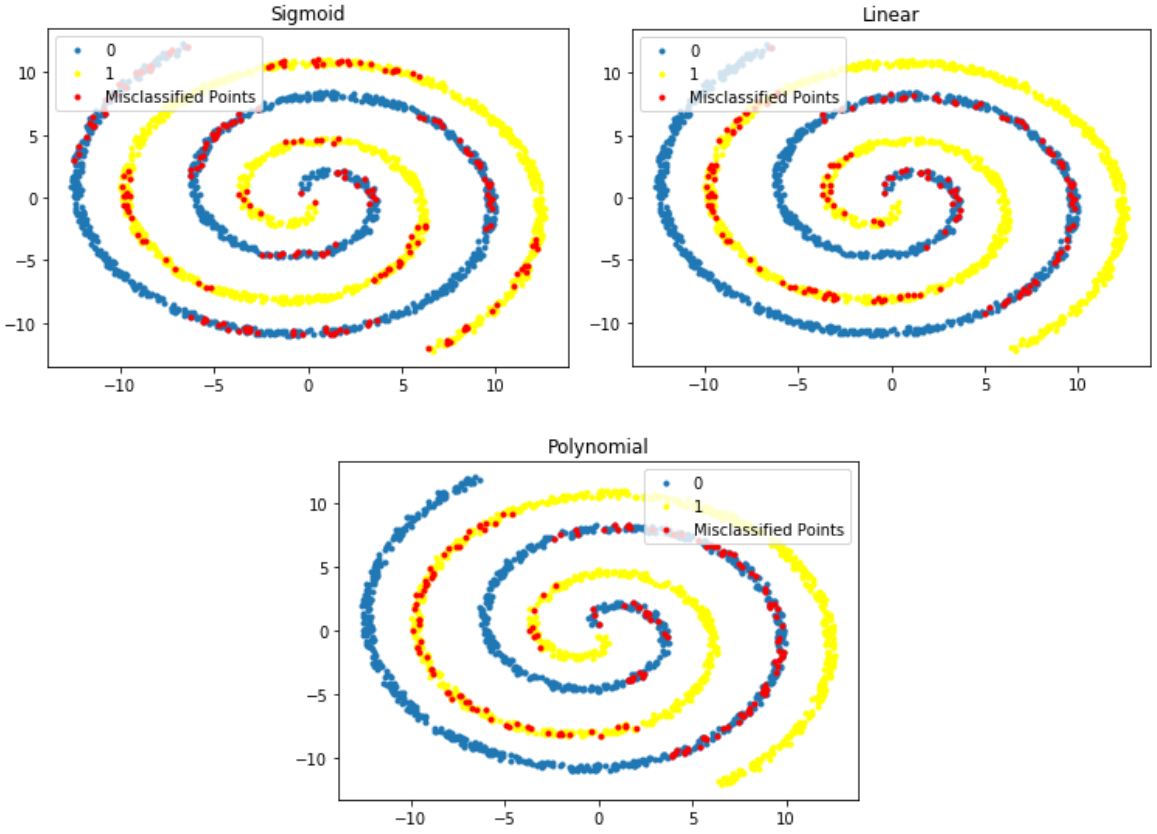
4.2.1.1 Case 1

The results obtained are

Kernel	RBF	Sigmoid	Linear	Polynomial
Optimum C	0.1	0.01	0.1	10
Confusion Matrix	$\begin{bmatrix} 205 & 0 \\ 0 & 195 \end{bmatrix}$	$\begin{bmatrix} 90 & 115 \\ 94 & 101 \end{bmatrix}$	$\begin{bmatrix} 120 & 85 \\ 75 & 120 \end{bmatrix}$	$\begin{bmatrix} 109 & 96 \\ 72 & 123 \end{bmatrix}$
Precision	1.0	0.4784	0.6004	0.5819
Recall	1.0	0.4785	0.6004	0.5812
F-Score	**1.0**	0.4771	0.6	0.5794

SVM using RBF kernel function classifies the data perfectly.

The misclassified test samples are

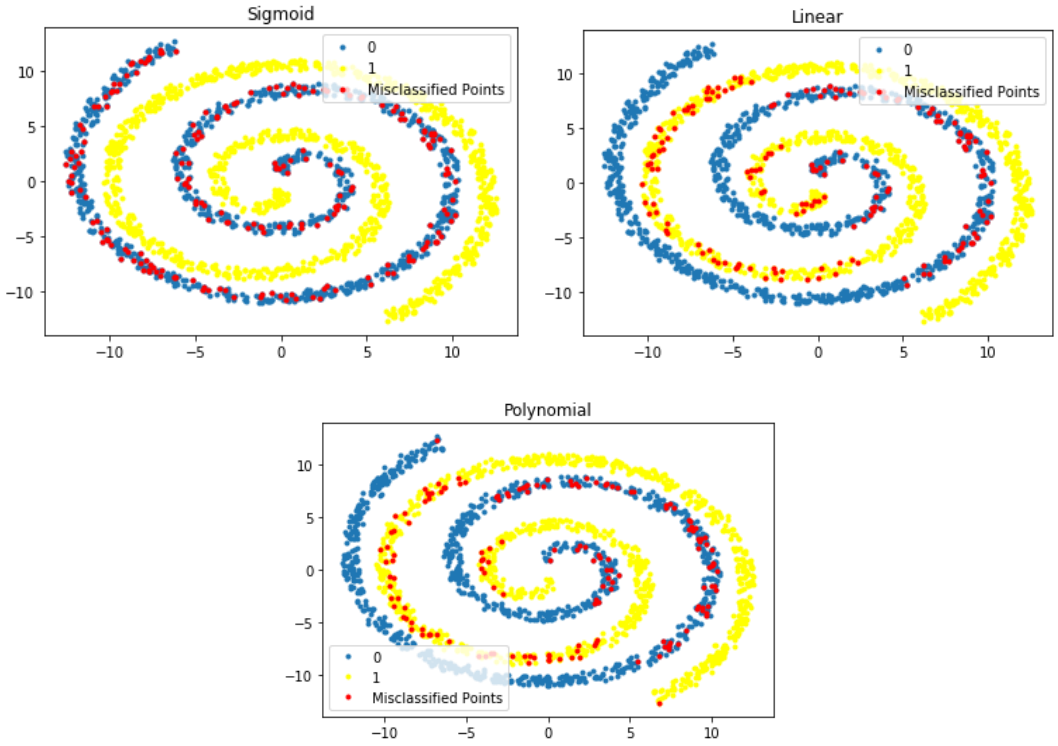


4.2.1.2 Case 2

The results obtained are

Kernel	RBF	Sigmoid	Linear	Polynomial
Optimum C	0.1	0.001	100	0.01
Confusion Matrix	$\begin{bmatrix} 205 & 0 \\ 0 & 195 \end{bmatrix}$	$\begin{bmatrix} 0 & 205 \\ 0 & 195 \end{bmatrix}$	$\begin{bmatrix} 121 & 84 \\ 82 & 113 \end{bmatrix}$	$\begin{bmatrix} 116 & 89 \\ 72 & 123 \end{bmatrix}$
Precision	1.0	0.2438	0.5848	0.5986
Recall	1.0	0.5	0.5849	0.5983
F-Score	**1.0**	0.3277	0.5848	0.5973

The misclassified test samples are



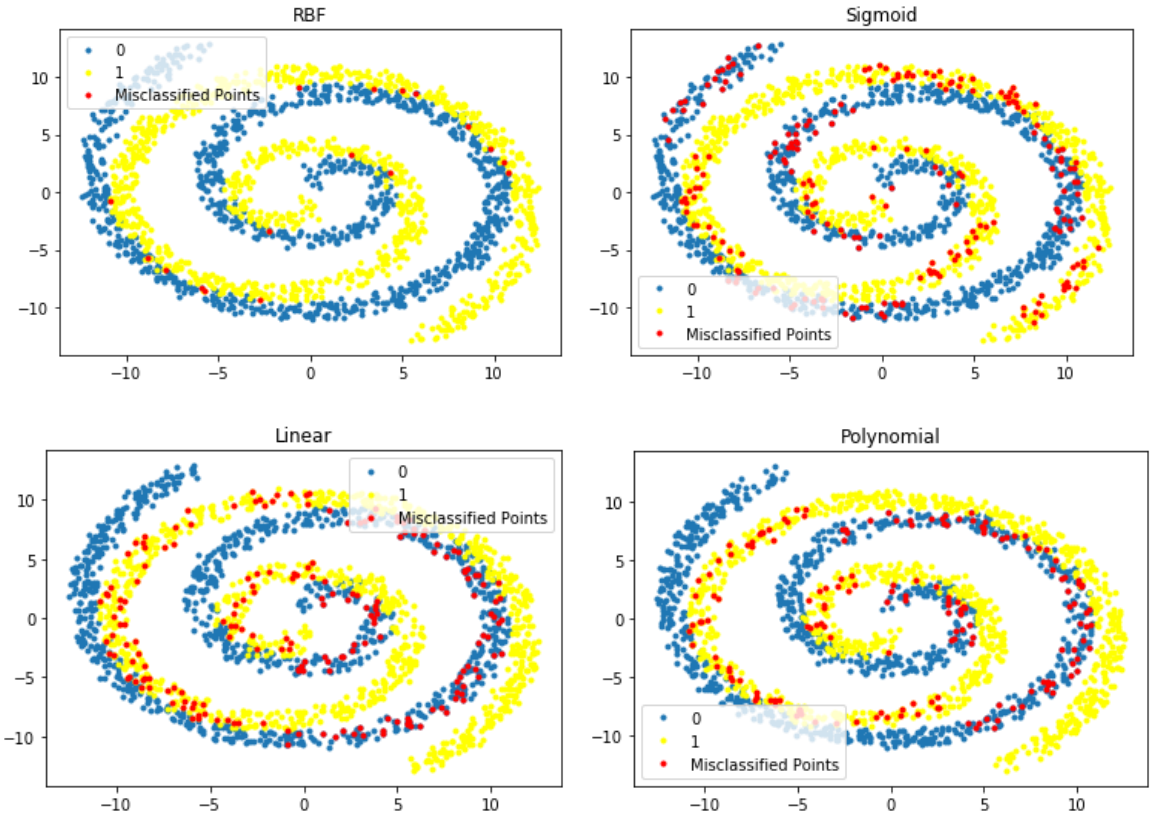
SVM suing RBF classifies the data perfectly.

4.2.1.3 Case 3

The results obtained are

Kernel	RBF	Sigmoid	Linear	Polynomial
Optimum C	10	0.1	0.001	10
Confusion Matrix	$\begin{bmatrix} 197 & 8 \\ 9 & 186 \end{bmatrix}$	$\begin{bmatrix} 98 & 107 \\ 102 & 93 \end{bmatrix}$	$\begin{bmatrix} 107 & 98 \\ 90 & 105 \end{bmatrix}$	$\begin{bmatrix} 111 & 94 \\ 77 & 118 \end{bmatrix}$
Precision	0.9575	0.4775	0.5302	0.5735
Recall	0.9574	0.4775	0.5302	0.5733
F-Score	**0.9575**	0.4774	0.5299	0.5723

The misclassified test samples are



SVM gives best results while using RBF kernel function.

4.2.2 Multilayer Perceptron

We apply the MLP classifier on the data for each of the cases using a variety of activation functions. We perform a 5-fold cross validation to obtain the optimum value of parameter alpha.

We use 4 layers, meaning the network contains 1 input layer, 2 hidden layers and 1 output layer. The input layer has 2 nodes. The first hidden layer has 25 nodes and the second hidden layer has 23 nodes. Finally, the output layer has 1 node. That is, we build a {2, 25, 23, 1} neural network.

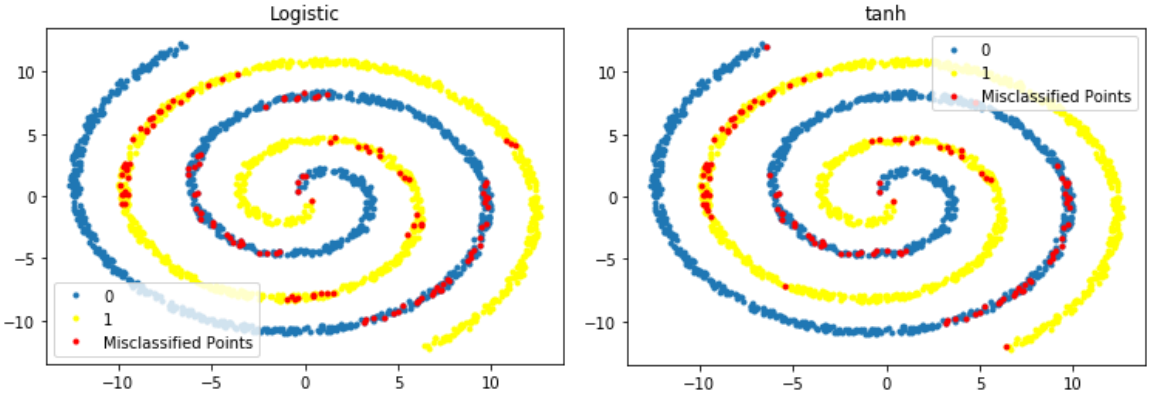
In the grid search, we take, alpha: [0.001, 0.01, 0.1, 10, 100].

4.2.2.1 Case 1

The results obtained are

Activation function	Logistic	tanh	ReLU
Optimum alpha	0.001	10	10
Confusion Matrix	$\begin{bmatrix} 134 & 71 \\ 59 & 136 \end{bmatrix}$	$\begin{bmatrix} 139 & 66 \\ 51 & 144 \end{bmatrix}$	$\begin{bmatrix} 205 & 0 \\ 0 & 195 \end{bmatrix}$
Precision	0.6756	0.7086	1.0
Recall	0.6755	0.7082	1.0
F-Score	0.6749	0.7074	**1.0**

The misclassified test samples are



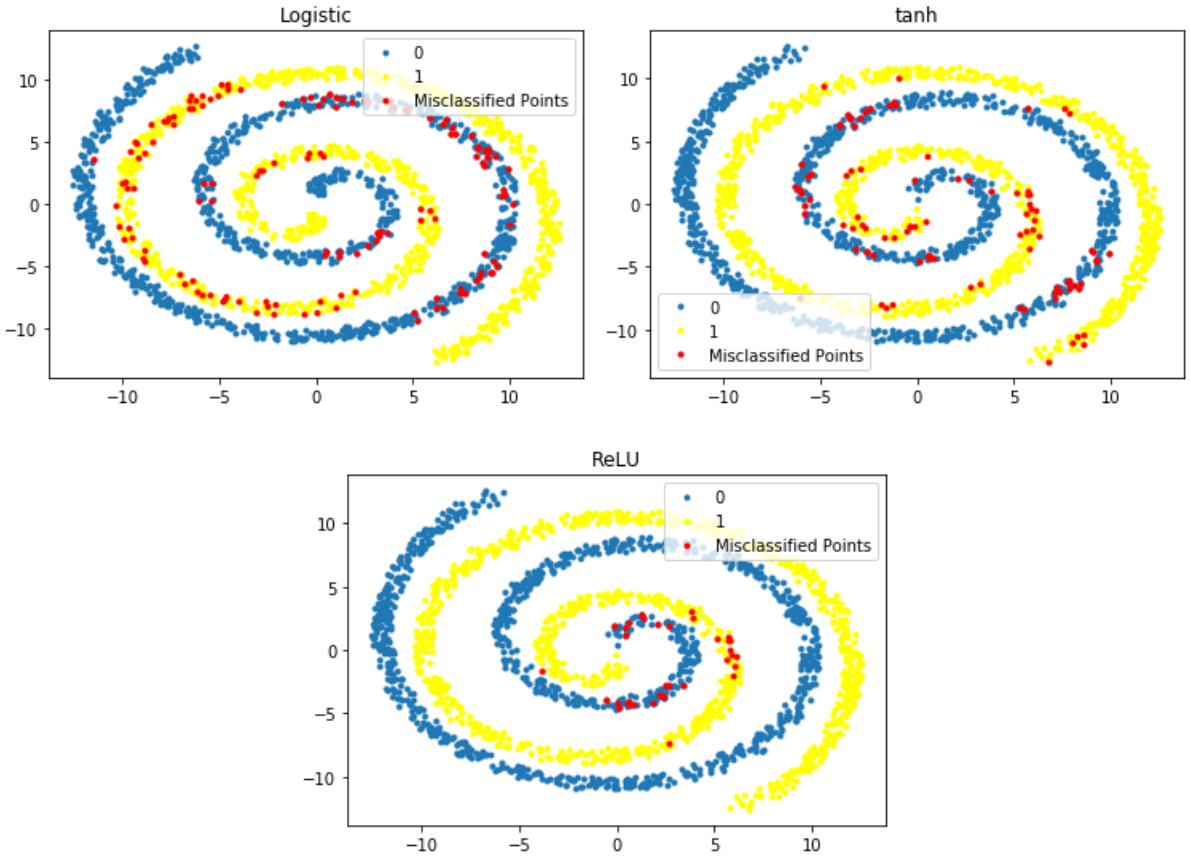
The performance of the MLP classifier is best when using ReLU activation function.

4.2.2.2 Case 2

The results obtained are

Activation function	Logistic	tanh	ReLU
Optimum alpha	0.1	0.01	10
Confusion Matrix	$\begin{bmatrix} 133 & 72 \\ 70 & 125 \end{bmatrix}$	$\begin{bmatrix} 152 & 53 \\ 45 & 150 \end{bmatrix}$	$\begin{bmatrix} 182 & 23 \\ 14 & 281 \end{bmatrix}$
Precision	0.6448	0.7552	0.9079
Recall	0.6449	0.7553	0.9080
F-Score	0.6448	0.7549	**0.9049**

The misclassified test samples are



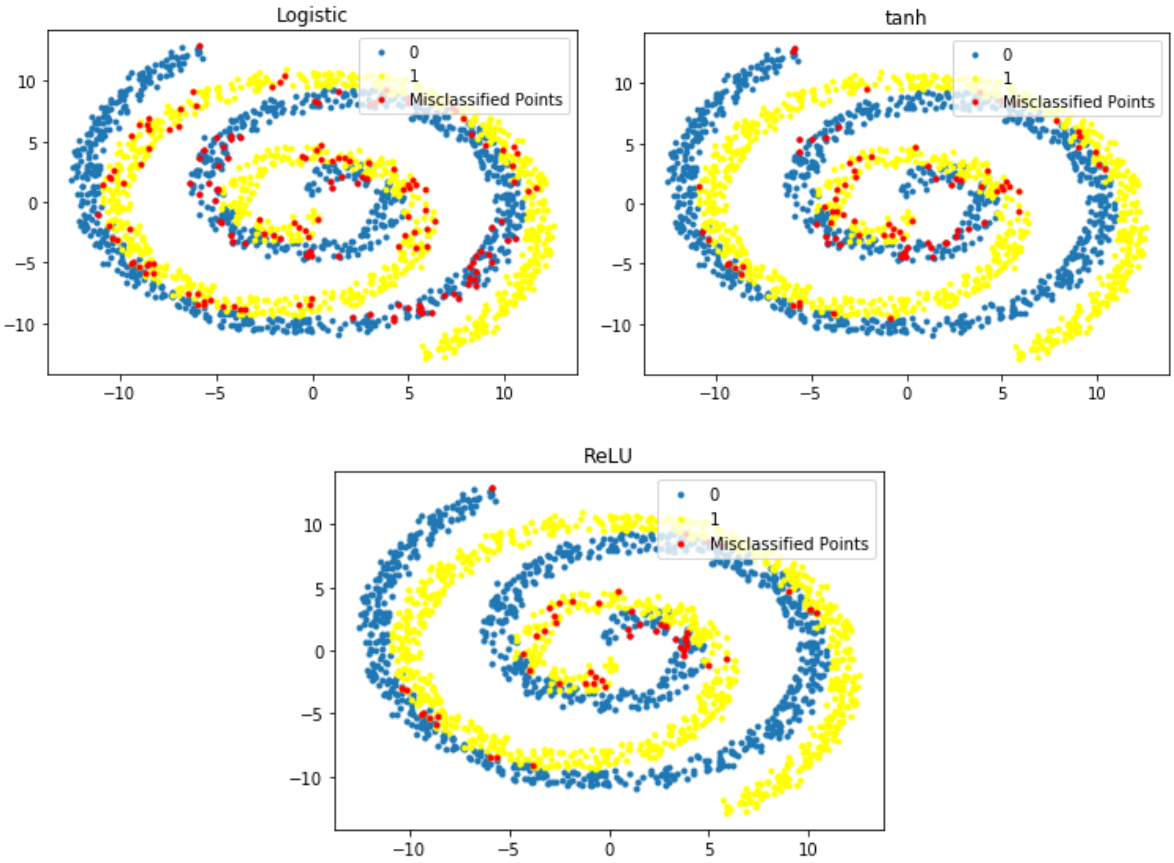
The performance of the MLP classifier is best when using ReLU activation function.

4.2.2.3 Case 3

The results obtained are

Activation function	Logistic	tanh	ReLU
Optimum alpha	0.001	10	10
Confusion Matrix	$\begin{bmatrix} 127 & 78 \\ 87 & 108 \end{bmatrix}$	$\begin{bmatrix} 166 & 39 \\ 48 & 147 \end{bmatrix}$	$\begin{bmatrix} 183 & 22 \\ 30 & 165 \end{bmatrix}$
Precision	0.5870	0.7830	0.8707
Recall	0.5867	0.7818	0.8694
F-Score	0.5865	0.7820	**0.8697**

The misclassified test samples are



The performance of the MLP classifier is best when using ReLU activation function.

5 DISCUSSION AND CONCLUSION

We evaluate the performances of Support Vector Machine and Multilayer Perceptron classifiers for two synthetically generated datasets – the half-moons data and the two-spiral data.

For the half-moons data, in the first case, the data being clearly linearly separable, both the classifiers, irrespective of choice of parameters (kernel functions, activation functions, etc.) give perfect classification. In the second case, when the two moons are slightly more intervening and noisy, SVM performs best while using RBF kernel function and MLP performs perfectly while the choice of activation function is logistic or tanh. For the third case, when the two moons are much more intervened and the data is extremely noisy, i.e. there are several areas of overlap between data points of the two classes, SVM gives good results when the kernel function is RBF; MLP gives good results when the activation function is tanh.

For the two-spiral data, we again explore three different scenarios. For all three cases, we see that, SVM gives best results for choice of kernel function as RBF; and MLP gives best results for choice of function as ReLU.

Thus, we can see that RBF kernel function for SVM is a universally good performer when it comes to classifying complex datasets. Technically if we use the Gaussian kernel, then the method is nonparametric, while if we use the linear or polynomial kernels, the model is parametric. In a way nonparametric model means that the complexity of the model is potentially infinite, its complexity can grow with the data. So asymptotically, if we have unlimited data and very weak assumptions about the problem, a nonparametric method is always better.

MLP classifier gives good results using different activation functions for different datasets. For the half-moons data, the logistic and tanh functions give relatively better results than ReLU. However, the performance of ReLU is still very good. And for the two-spiral dataset, ReLU function outperforms the rest. Thus, on an average the overall performance of ReLU is better than sigmoid or tanh. One probable reason for this could be that the complexity of the half-moons data and that of the two-spiral data is high. Since, sigmoid and tanh have slower convergence than ReLU, the latter classifies the data of a much complex type in lesser time and more effectively. Also, since ReLU function can account for sharp changes, it is more effective than sigmoid or tanh functions.

However, if we study the overall performances of the two classifiers, given the best choice of parameters in each of the cases, the multilayer perceptron and the support vector machine are at par with each other.

6 REFERENCES

[1] Victoria Ivanova and Yaroslav Nalivajko. “Large Synthetic Datasets to Compare Different Data Mining Methods”.

[2] Kevin J. Lang and Michael J. Witbrock. “Learning to Tell Two Spirals Apart”. *Proceedings of the 1988 Conectionist Models Summer School*, ISBN: 0-55860-015-9.

[3] Chih-Wei Hsu, Chih-Jen Lin, and Chih-Chung Chang. “A Practical Guide to Support Vector Classification”. Department of Computer Science National Taiwan University, Taipei 106, Taiwan, 2003.

[4] Dennis W. Ruck, Steven K. Rogers, and Matthew Kabrisky. “Feature Selection using a Multilayer Perceptron”. *Journal of Neural Network Computing*, Volume 2, Number 2, 1990, pp 40-48.

[5] Simon Haykin. *Neural Networks and Learning Machines.* Pearson, Prentice Hall. 3rd ed.

[6] Celisse, Alain. “Parameter Tuning and Cross Validation Algorithms”.

[7] Arlot, Sylvian, Alain Celisse. “A survey of cross-validation procedures for model selection”. *Statistical Surveys* Vol. 4 (2010) ISSN 1935-7516.

[8] Oprea, Cristina. “Performance Evaluation of the Data Mining Classification Methods”. Annals of the „Constantin Brâncuşi” University of Târgu Jiu, Economy Series, Special Issue (2014): *Information society and sustainable development.*