# Report

## 1) SVM Classifier

a) The entire dataset was first split into train and test set by randomly picking 80% of the data as training set and the rest as train set. The function SVC of the library sklearn was used to implement the following SVM Classifiers. The confusion matrix and the classification report for C = 1 (default value) for the three kernels is attached below:

i) Linear

```
[[16 13]
 [ 2 40]]
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| b | 0.89 | 0.55 | 0.68 | 29 |
| g | 0.75 | 0.95 | 0.84 | 42 |
| accuracy | | | 0.79 | 71 |
| macro avg | 0.82 | 0.75 | 0.76 | 71 |
| weighted avg | 0.81 | 0.79 | 0.78 | 71 |

ii) Quadratic

```
[[24  5]
 [ 3 39]]
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| b | 0.89 | 0.83 | 0.86 | 29 |
| g | 0.89 | 0.93 | 0.91 | 42 |
| accuracy | | | 0.89 | 71 |
| macro avg | 0.89 | 0.88 | 0.88 | 71 |
| weighted avg | 0.89 | 0.89 | 0.89 | 71 |

iii) Radial Basis Function

```
[[25  4]
 [ 2 40]]
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| b | 0.93 | 0.86 | 0.89 | 29 |
| g | 0.91 | 0.95 | 0.93 | 42 |
| accuracy | | | 0.92 | 71 |
| macro avg | 0.92 | 0.91 | 0.91 | 71 |
| weighted avg | 0.92 | 0.92 | 0.91 | 71 |

b) The most suitable value of the generalization constant C was found using the GridSearchCV() function of the sklearn library. It was found to be C = 10. The confusion matrix and the classification report for C = 10 for the three kernels is attached below:

i) Linear

```
[[17 12]
 [ 4 38]]
              precision    recall  f1-score   support

           b       0.81      0.59      0.68        29
           g       0.76      0.90      0.83        42

    accuracy                           0.77        71
   macro avg       0.78      0.75      0.75        71
weighted avg       0.78      0.77      0.77        71
```

ii) Quadratic

```
[[23  6]
 [ 3 39]]
              precision    recall  f1-score   support

           b       0.88      0.79      0.84        29
           g       0.87      0.93      0.90        42

    accuracy                           0.87        71
   macro avg       0.88      0.86      0.87        71
weighted avg       0.87      0.87      0.87        71
```

iii) Radial Basis Function

```
[[27  2]
 [ 2 40]]
              precision    recall  f1-score   support

           b       0.93      0.93      0.93        29
           g       0.95      0.95      0.95        42

    accuracy                           0.94        71
   macro avg       0.94      0.94      0.94        71
weighted avg       0.94      0.94      0.94        71
```

c) The final train and test accuracy in a comparison table of the classifier comparing the three kernels is attached below:

| | Kernel | Train Accuracy | Test Accuracy |
|---|---|---|---|
| **0** | Linear | 0.953571 | 0.774648 |
| **1** | Quadratic | 0.953571 | 0.873239 |
| **2** | Radial Basis Function | 0.953571 | 0.943662 |

## 2) ANN Classifier

a) The dataset was splitted into two parts with 80:20 ratio. The bigger part was used for training the models and the other part for testing. The function MLPClassifier of the library sklearn was used to implement the following ANN Classifiers.
In part a, we varied the number of hidden layers and number of nodes in hidden layers as follow:

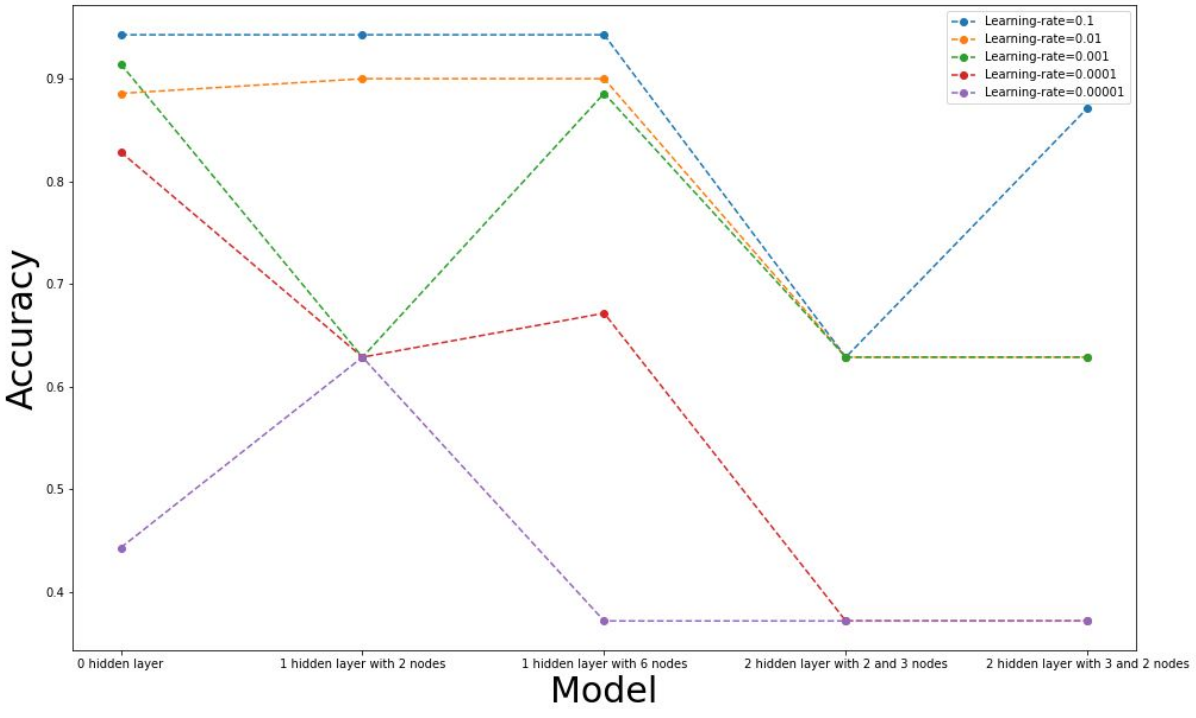
i) 0 hidden layers
ii) 1 hidden layer with 2 nodes
iii) 1 hidden layer with 6 nodes
iv) 2 hidden layer with 2 and 3 nodes
v) 2 hidden layer with 3 and 2 nodes

b) In part b we varied the learning rates as 0.1, 0.01, 0.001, 0.0001, 0.00001 for each of the architectures mentioned in the part a.

c) From the graph-2a and graph-2b we can clearly see that the accuracy is highest for the following models:

1. Hidden Layer = “0 hidden layers” , Learning rate =0.1 , optimiser= “Stochastic gradient descent”, activation=”logistic”, batch_size=”auto”(default), max_iter=2000, random state=0, momentum=0.9(default)
2. Hidden Layer = “1 hidden layer with 2 nodes”, Learning rate =0.1 , optimiser= “Stochastic gradient descent” , activation=”logistic”, batch_size=”auto”(default), max_iter=2000, random state=0, momentum=0.9(default)
3. Hidden Layer = “1 hidden layer with 6 nodes” , Learning rate =0.1 , optimiser= “Stochastic gradient descent”, activation=”logistic”, batch_size=”auto”(default), max_iter=2000, random state=0, momentum=0.9(default)

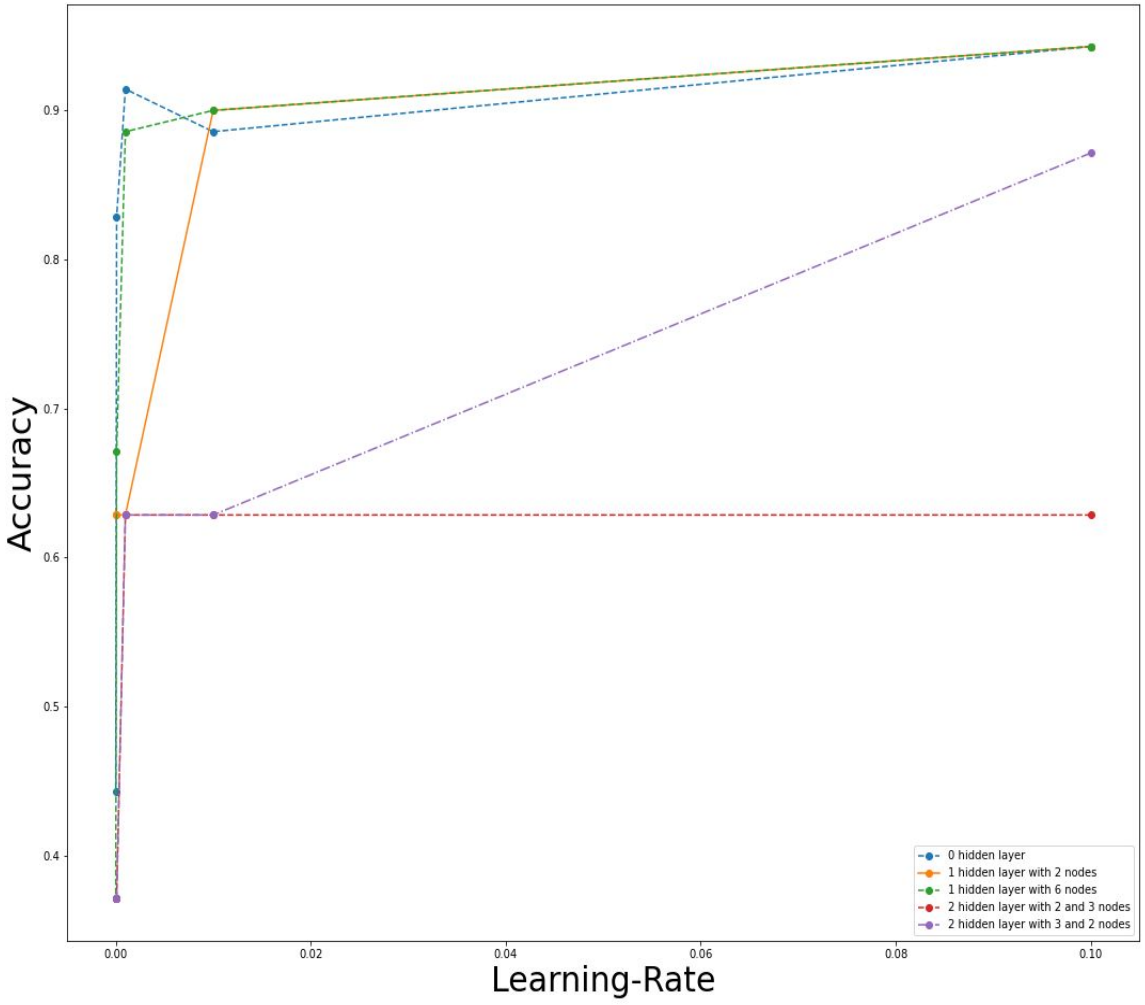
All of the above mentioned models have the same accuracy 94.28% .




**Graph-2(a)**

From the graph-2a ,we can see that the models with Learning rate=0.1 are having the highest accuracy. Also we can observe from the graph that , as the Learning rate becomes lower the accuracy also decreases.

We know that large learning rates result in unstable training and tiny rates result in a failure to train.
-So we can that in our case learning rate=0.1 is just perfect causing the most accuracy. And as the learning rate becomes lower , that results in a failure to train the model .




**Graph-2b**

Another factor that causes the variation of accuracy is the number of hidden layers and the node in each layer .
From the graph-2b we can see that:-
Comparison of accuracies:
(0 hidden layers) ~ (1 hidden layer with 2 nodes) ~ (1 hidden layer with 6 nodes) > (1 hidden layer with 3 and 2 nodes) > (1 hidden layer with 2 and 3 nodes)

So we can see that as the hidden layer sizes increase the accuracy decreases. So clearly the model with two hidden layers overfits the dta. Also the 0 hidden layer model gives the highest accuracy , so we can conclude the data might be linearly separable . Also as data has two outputs: ‘g’ and ‘b’ . So the output is binary. To represent any binary function we don’t need a function greater than 1 hidden layer.