Homework – 3
Name: Chetanraj Kadam
SUID: 250256631

Problem Statement:

Compare the performance (quality and computational effort) of LVQ2, Backprop, and SVM on a classification problem with at least ten input attributes, e.g., chosen from the UCI database. How sensitive to algorithm parameters are the results of each algorithm?

Report:

- For comparison of these 3 algorithms I'm using "breast-cancer-wisconsin" dataset which has 10 input attributes and 2 output class labels. I have done some data preprocessing in the code such had to impute missing values with median and transform output labels from 2, 4 to 0, 1. Total records in dataset are 699.

- Each of these algorithms – Backpropagation, SVM and LVQ2 implemented separately on the same dataset with variations on algorithm parameters such as for backpropagation activation function changed, for SVM different kernel type and for LVQ different learning rate. These results are shown below and compared against each other.

- To compare Backpropagation, SVM and LVQ2 against each other we can use metrics such as accuracy and time taken by model to learn from training data. Below results shows best result achieved from each algorithm:

Metrics	**Backpropagation**	**SVM**	**LVQ2**
Accuracy	95.71	96.43	93.57
Time	7.8439	0.1087	4.1011

So from overall result for this particular dataset we can observe that in terms of accuracy SVM is better that Backpropagation and both are better than LVQ2. Whereas in terms of time taken by algorithm to learn from training data we can see SVM is far better and takes much less time than LVQ2 and backpropagation which takes more time.

Individual algorithm results with parameter comparison are shown as follows:

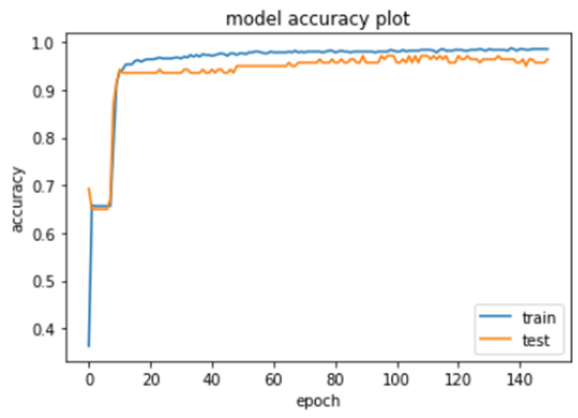
Backpropagation:

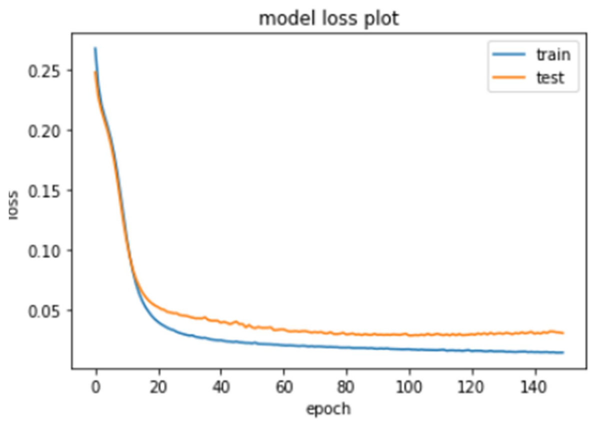
- I have implemented backpropagation using Keras neural network models. Keras layers use backpropagation algorithm internally.
- To compare sensitivity of result to algorithm parameters I have used different activation function and compared result.

Activation - Sigmoid

```
#Create Neural network model
model = Sequential()
model.add(Dense(9,input_dim = 9,activation='sigmoid'))
model.add(Dense(9,activation='sigmoid'))
model.add(Dense(1,activation='sigmoid'))
```

```
140/140 [==============================] - 0s 21us/step

acc: 96.43%

loss: 3.07%

Time taken: 7.995630502700806
```

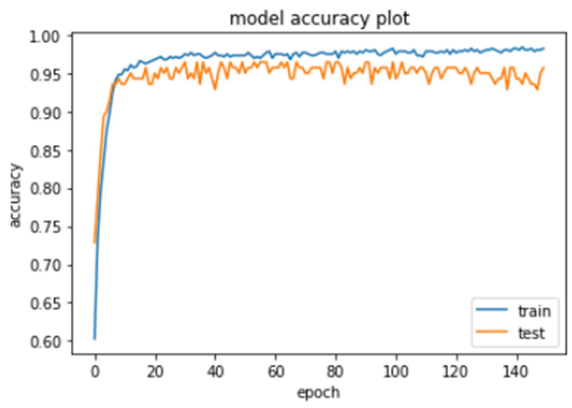


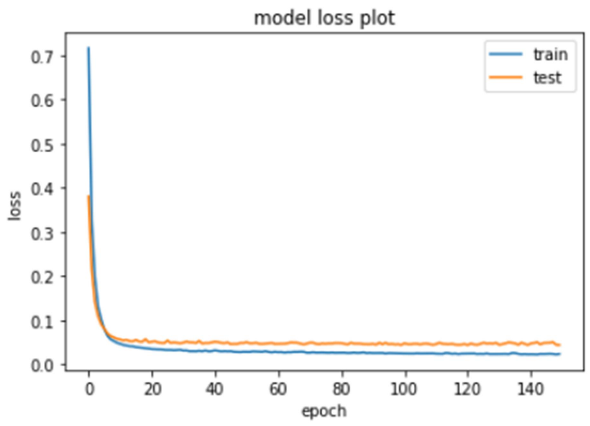


Activation - Relu

```
model1 = Sequential()
model1.add(Dense(9,input_dim = 9,activation='relu'))
model1.add(Dense(9,activation='relu'))
model1.add(Dense(1))
```

```
140/140 [==============================] - 0s 28us/step

acc: 95.71%

loss: 4.36%

Time taken: 7.843999624252319
```





So from results of both activation function we can observe that accuracy for sigmoid is 96.43% whereas accuracy for Relu is 95.71%. This drop in accuracy is because sigmoid activation function works better with binary classification than Relu. There is not much change in other parameters such as loss and time taken to learn.

SVM:

- To implement SVM I have used sklearn python library. To compare performance between different implementations of SVM based on following parameters –

  kernel : string, optional (default=’rbf’)

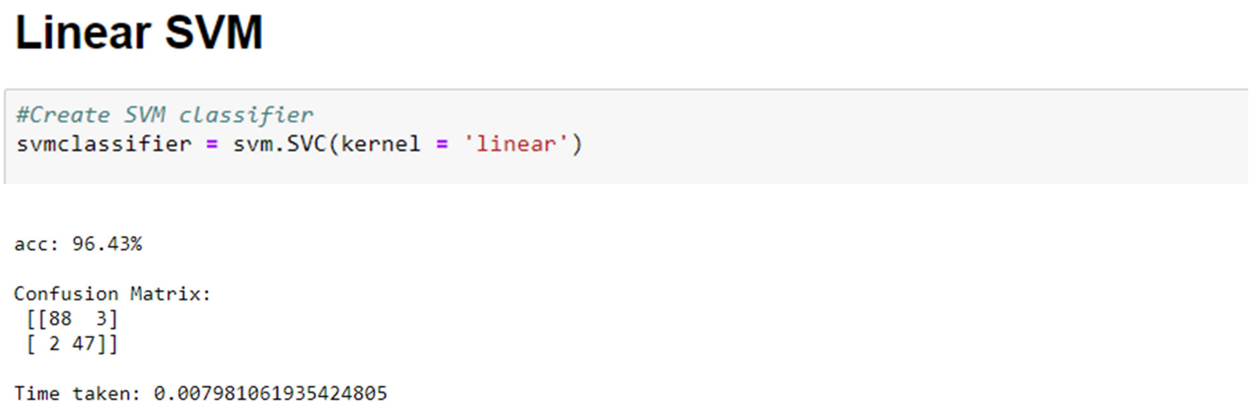
  Specifies the kernel type to be used in the algorithm.

  C : float, optional (default=1.0)

  Penalty parameter C of the error term.

  gamma : float, optional (default=’auto’)

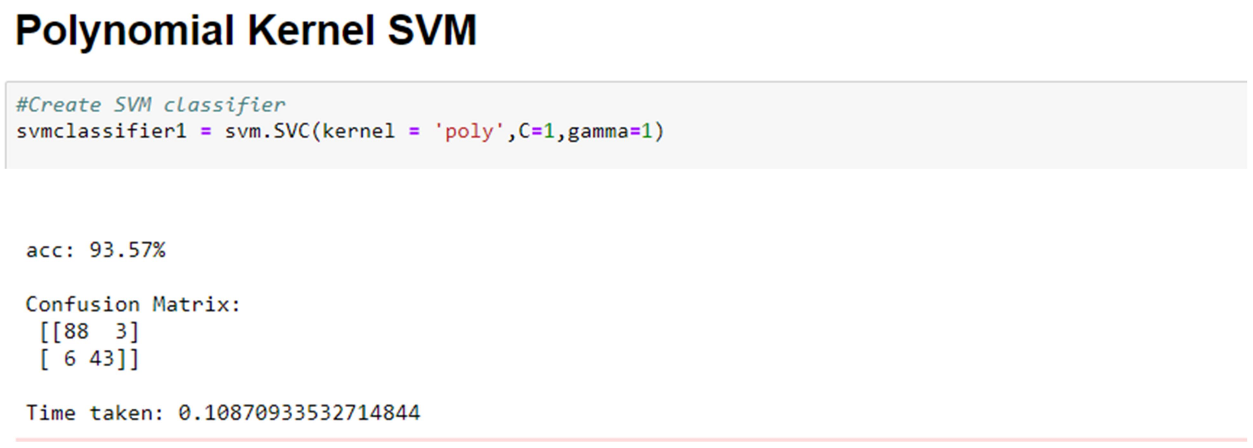
  Kernel coefficient for ‘rbf’, ‘poly’ and ‘sigmoid’.



Linear SVM

```
#Create SVM classifier
svmclassifier = svm.SVC(kernel = 'linear')
```

```
acc: 96.43%

Confusion Matrix:
 [[88  3]
 [ 2 47]]

Time taken: 0.007981061935424805
```

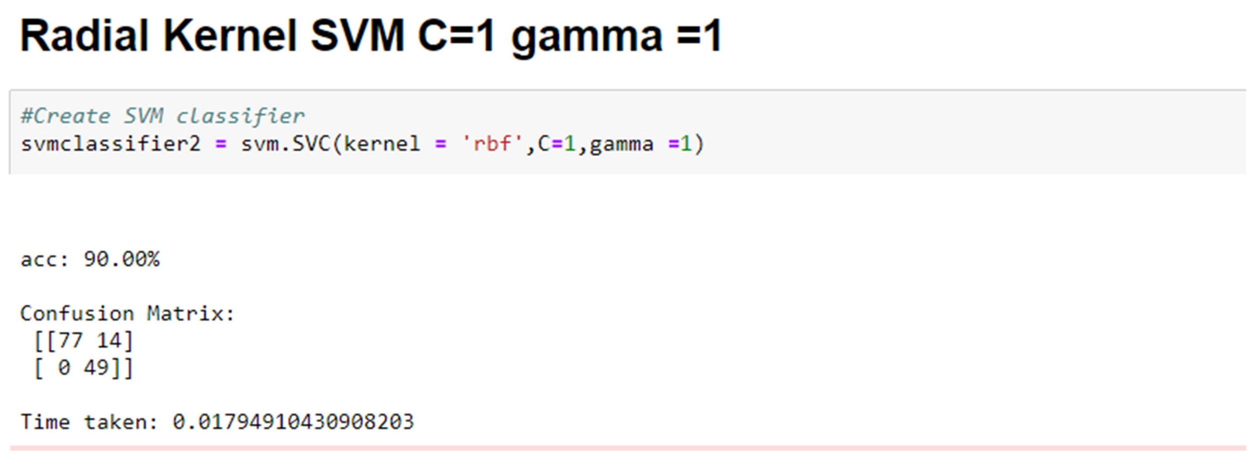


Polynomial Kernel SVM

```
#Create SVM classifier
svmclassifier1 = svm.SVC(kernel = 'poly',C=1,gamma=1)
```

```
acc: 93.57%

Confusion Matrix:
 [[88  3]
 [ 6 43]]

Time taken: 0.10870933532714844
```

We can see here accuracy has dropped from 96.43% to 93.57% with polynomial Kernel SVM.

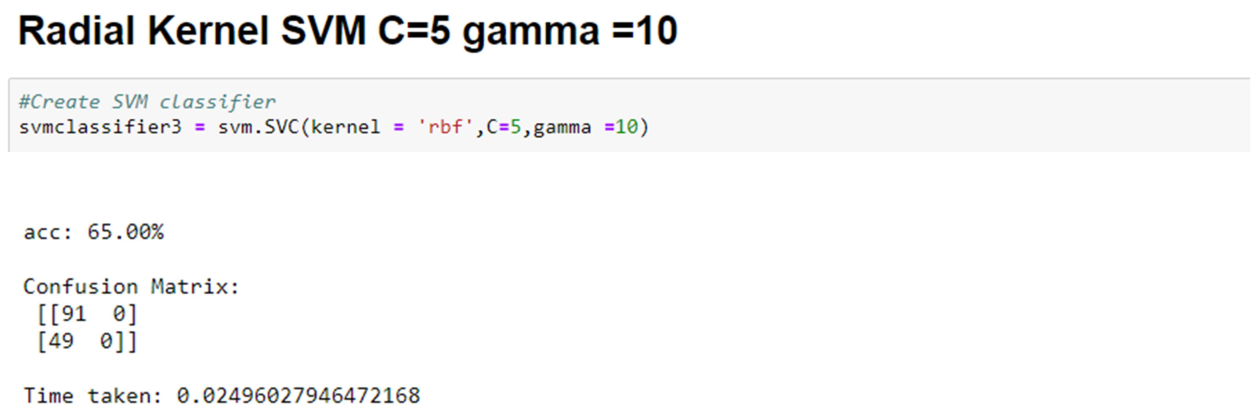


Radial Kernel SVM C=1 gamma =1

```
#Create SVM classifier
svmclassifier2 = svm.SVC(kernel = 'rbf',C=1,gamma =1)
```

```
acc: 90.00%

Confusion Matrix:
 [[77 14]
 [ 0 49]]

Time taken: 0.01794910430908203
```

With Radial Kernel SVM accuracy again dropped to 90.00%.



Radial Kernel SVM C=5 gamma =10

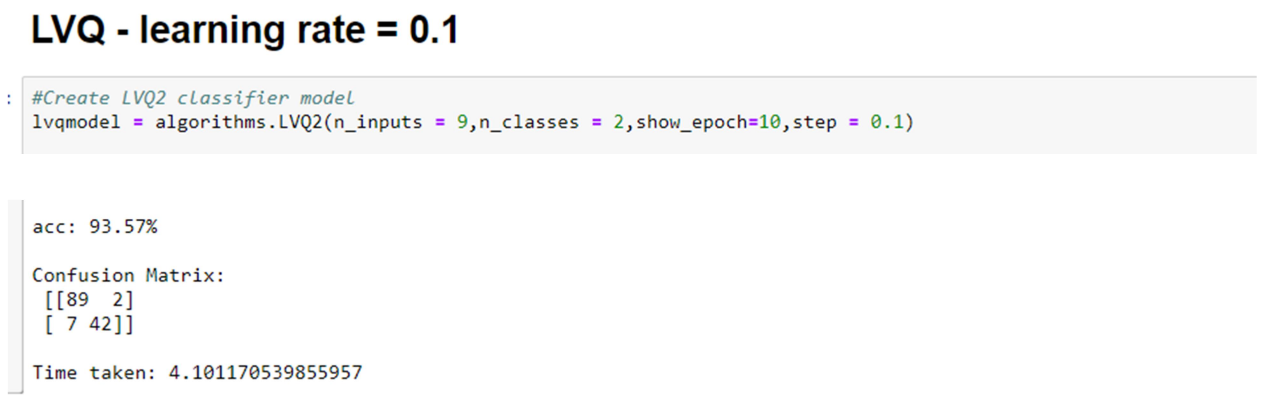
```
#Create SVM classifier
svmclassifier3 = svm.SVC(kernel = 'rbf',C=5,gamma =10)
```

```
acc: 65.00%

Confusion Matrix:
 [[91  0]
 [49  0]]

Time taken: 0.02496027946472168
```

From this we can observe that with increase of kernel coefficient gamma here accuracy is significantly dropping. So it is clear that kernel coefficient has huge impact in SVM learning.

LVQ2:

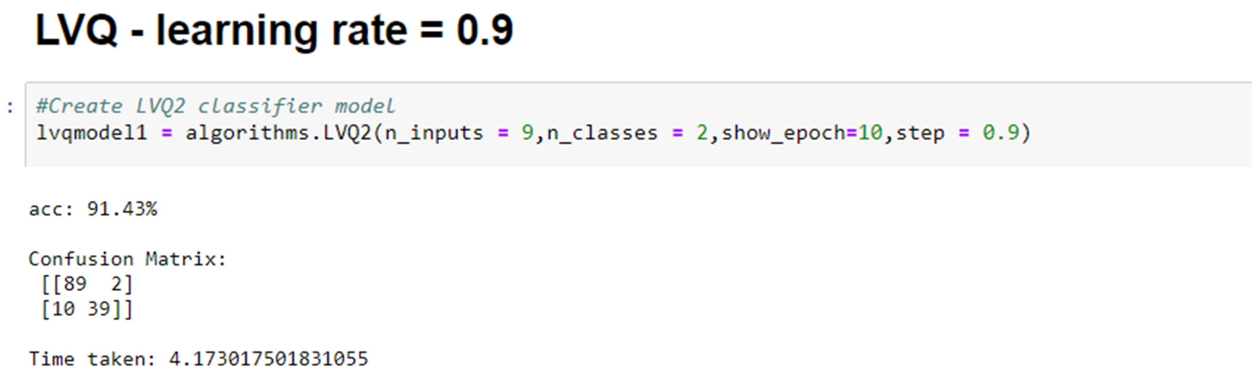
- To implement LVQ2 I have used Neupy python library. To compare sensitivity of LVQ2 to different parameters we can see the impact of learning rate in the process of learning of LVQ2.



LVQ - learning rate = 0.1

```
#Create LVQ2 classifier model
lvqmodel = algorithms.LVQ2(n_inputs = 9,n_classes = 2,show_epoch=10,step = 0.1)
```

```
acc: 93.57%

Confusion Matrix:
 [[89  2]
 [ 7 42]]

Time taken: 4.101170539855957
```



LVQ - learning rate = 0.9

```
#Create LVQ2 classifier model
lvqmodel1 = algorithms.LVQ2(n_inputs = 9,n_classes = 2,show_epoch=10,step = 0.9)
```

```
acc: 91.43%

Confusion Matrix:
 [[89  2]
 [10 39]]

Time taken: 4.173017501831055
```

So from this result we can conclude that as we increase learning rate here accuracy is goint to decrease by some margin. With learning rate of 0.1 accuracy was 93.57% and then it is dropped to 91.43% for the learning rate of 0.9.