Q1:

1. Layers:

(a) Use 2 hidden layers with 20 nodes in each layer.

(b) Use 2 hidden layers with 100 nodes in each layer.

(c) Use 5 hidden layers with 20 nodes in each layer.

(d) Use 5 hidden layers with 100 nodes in each layer.

2. Activation Function:

(a) Use ‘relu’.

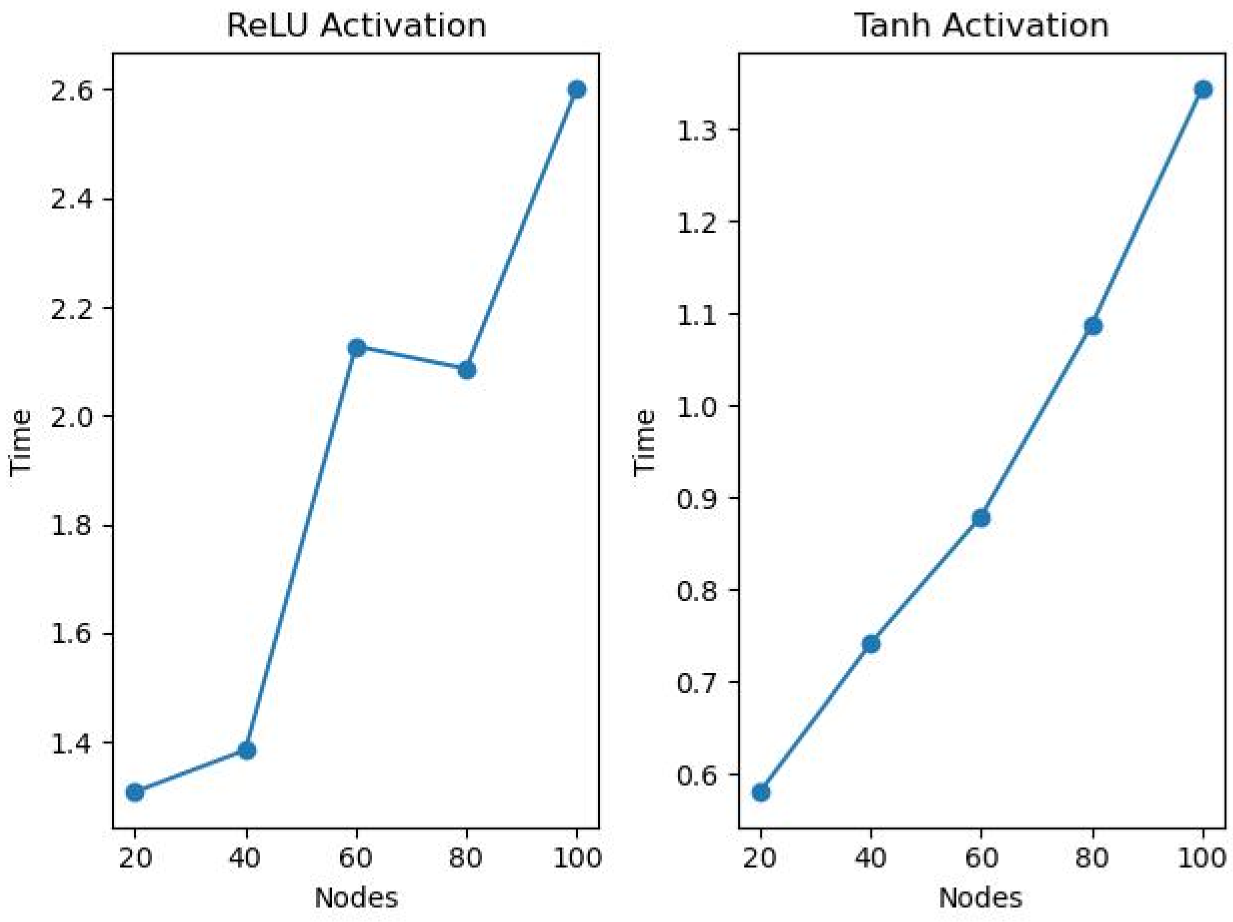
(b) Use ‘tanh’.

```
Hidden_layers: 2 ; Nodes: 20 ; Activation: relu
recall    = 0.9092 ±0.0085
precision = 0.9100 ±0.0085
f1        = 0.9092 ±0.0084
Time 1.2439209938049316

Hidden_layers: 2 ; Nodes: 20 ; Activation: tanh
recall    = 0.9130 ±0.0102
precision = 0.9141 ±0.0104
f1        = 0.9131 ±0.0102
Time 0.5853209972381592

Hidden_layers: 2 ; Nodes: 100 ; Activation: relu
recall    = 0.9397 ±0.0081
precision = 0.9405 ±0.0078
f1        = 0.9397 ±0.0081
Time 2.7366137981414793

Hidden_layers: 2 ; Nodes: 100 ; Activation: tanh
recall    = 0.9352 ±0.0085
precision = 0.9357 ±0.0084
f1        = 0.9351 ±0.0085
Time 1.4506761074066161

Hidden_layers: 5 ; Nodes: 20 ; Activation: relu
recall    = 0.9075 ±0.0057
```

```
precision = 0.9084 ±0.0053
f1        = 0.9075 ±0.0058
Time 2.93609414100647

Hidden_layers: 5 ; Nodes: 20 ; Activation: tanh
recall    = 0.9065 ±0.0092
precision = 0.9077 ±0.0090
f1        = 0.9066 ±0.0091
Time 1.8800164222717286

Hidden_layers: 5 ; Nodes: 100 ; Activation: relu
recall    = 0.9378 ±0.0073
precision = 0.9384 ±0.0072
f1        = 0.9379 ±0.0073
Time 6.551486778259277

Hidden_layers: 5 ; Nodes: 100 ; Activation: tanh
recall    = 0.9397 ±0.0070
precision = 0.9404 ±0.0070
f1        = 0.9397 ±0.0070
Time 4.2029359340667725
```

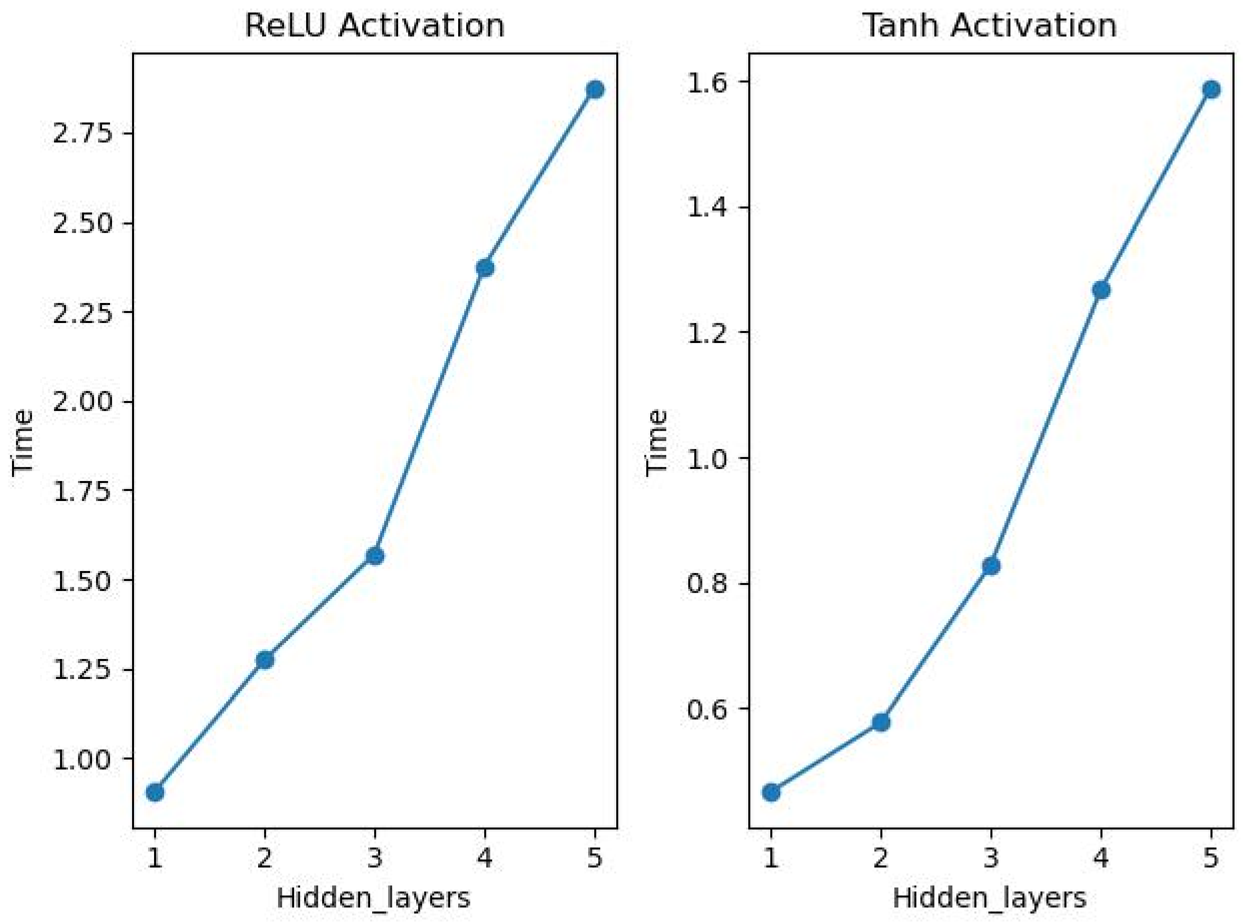
Q1: 1. The time it takes to fit each model as size of layers increase for various layer sizes (recommend number of layers = 2 and size of layers = [20, 40, 60, 80, 100]). Provide graphs.

| | hidden_layers | nodes | activation | time |
|---|---|---|---|---|
| 0 | 2 | 20 | relu | 1.307363 |
| 1 | 2 | 20 | tanh | 0.580583 |
| 2 | 2 | 40 | relu | 1.384251 |
| 3 | 2 | 40 | tanh | 0.741307 |
| 4 | 2 | 60 | relu | 2.127945 |
| 5 | 2 | 60 | tanh | 0.879589 |
| 6 | 2 | 80 | relu | 2.086441 |
| 7 | 2 | 80 | tanh | 1.087524 |
| 8 | 2 | 100 | relu | 2.602205 |
| 9 | 2 | 100 | tanh | 1.344310 |



Q1: 2. The time it takes to fit each model as number of layers increase for various numbers of layers (recommend number of layers = [1, 2, 3, 4, 5] and size of layers = 20). Provide graphs.

| | hidden_layers | nodes | activation | time |
|---|---|---|---|---|
| 0 | 1 | 20 | relu | 0.906844 |
| 1 | 1 | 20 | tanh | 0.466802 |
| 2 | 2 | 20 | relu | 1.274438 |
| 3 | 2 | 20 | tanh | 0.577103 |
| 4 | 3 | 20 | relu | 1.569927 |
| 5 | 3 | 20 | tanh | 0.828048 |
| 6 | 4 | 20 | relu | 2.377049 |
| 7 | 4 | 20 | tanh | 1.266640 |
| 8 | 5 | 20 | relu | 2.873670 |
| 9 | 5 | 20 | tanh | 1.587789 |



Q1：3. What type of growth do you see for each? Why?
The time exponential growth with an increase in the number of layers with tanh activation and relu activation at beginning. Linear growth with increase in size of laters.

For the relu activation graph, the time growth appears to be linear with layers. This means that as the number of nodes increases, the time taken increases at a constant rate. This could be because the ReLU function is computationally efficient as it only needs to determine if an input is above or below zero.

For the tanh Activation graph, the time growth appears to be exponential with layers. This means that as the number of nodes increases, the time taken increases at an accelerating rate. The tanh function is more computationally expensive than ReLU as it involves more complex mathematical operations (exponential functions), which could explain the exponential growth in time as the number of nodes increases.

Q1：4. What happens in simple/smallish network if you stop using MinMaxScaler. Why?
A warning appears if a MinMaxScaler is not used for a simple network. This is because the data falls outside of the optimal range for the activation functions.

Question 2: In the scikit learn package, the default value for the learning rate of an MLP is 0.001. Explain what will happen to the classification result if we set this parameter to 0.5 and why.
If the learning parameter is set too high, the gradient descent method will not converge on the optimal value; instead, it will pass the minimum and fluctuate between non-optimal values. Gradient descent's learning rate can be viewed as the step size. An algorithm may "jump over" the minima we are attempting to obtain if the learning rate is too high, resulting in oscillations around the minimum or maybe outright divergence. This is because while a greater learning rate produces larger steps, it also increases the chance of exceeding the minimum.