Answer the following questions:

1. What is the activation function that you will choose for the output layer? Justify your answer briefly.

The activation function for the output layer is identity function because we are having a regression problem. If a classification activation function such as Sigmoid is used, the output layer will produce the final value in the limited range of value.

2. How many neurons should there be in the output layer? Why?

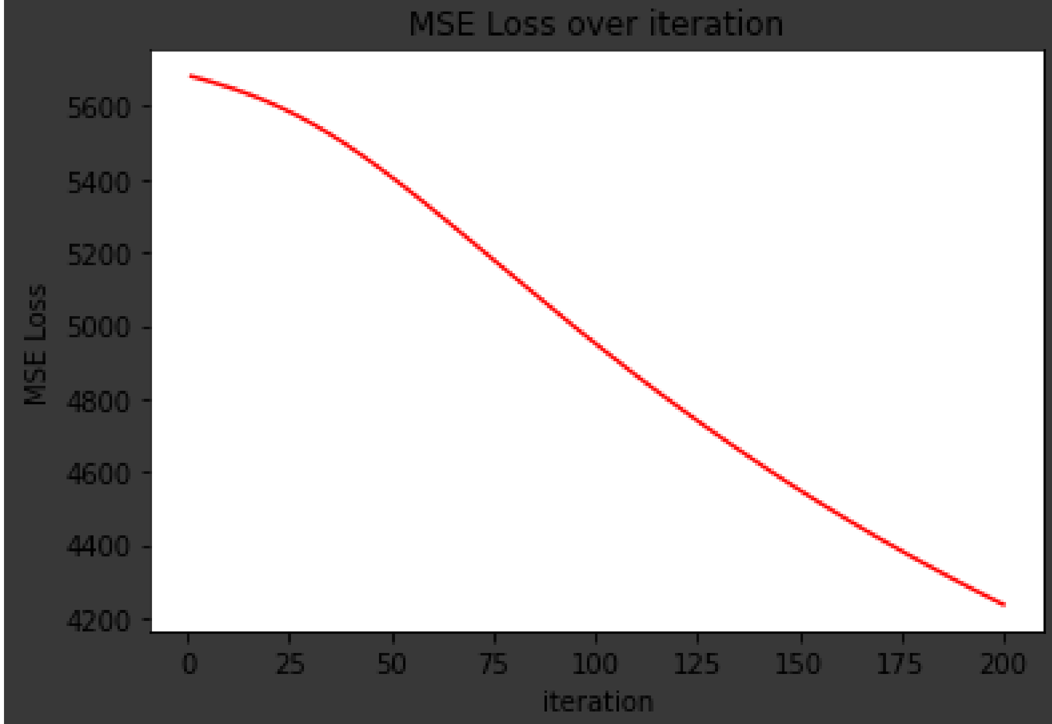
There will be 1 neuron in the output layer since we need to predict the same size of y matrix.

3. Report the average MSE loss.

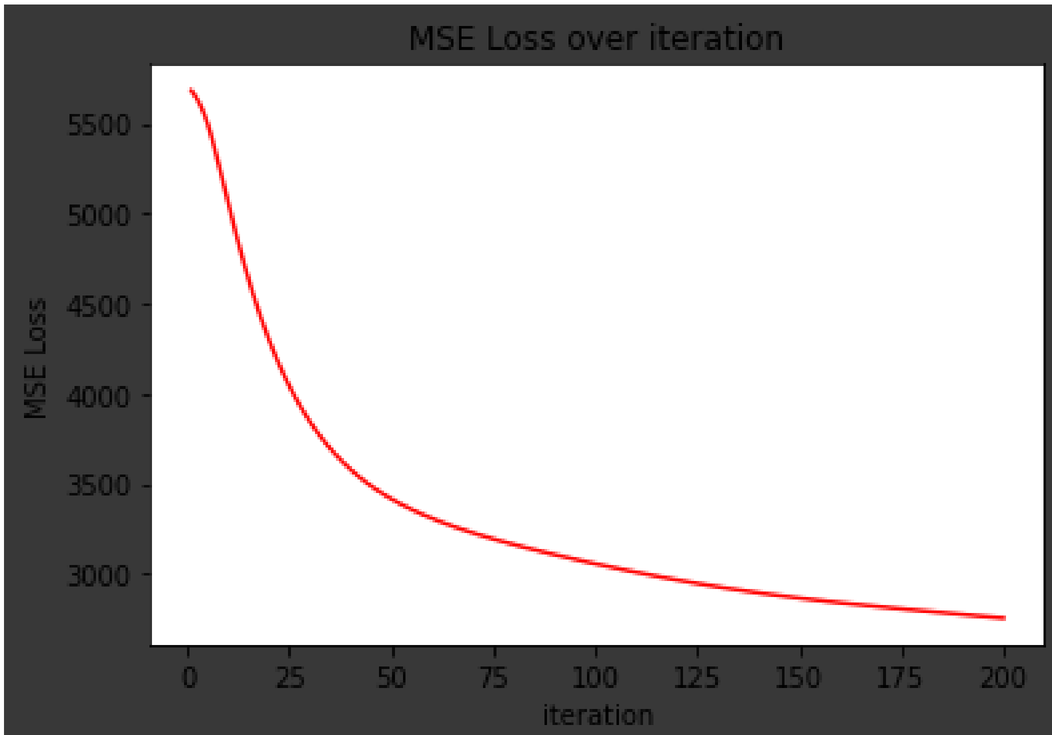
Average MSE loss on training dataset: 302.9964838443765
Average MSE loss on testing dataset: 1767.3839154766

4. Plot the loss as a function of the number of iterations.

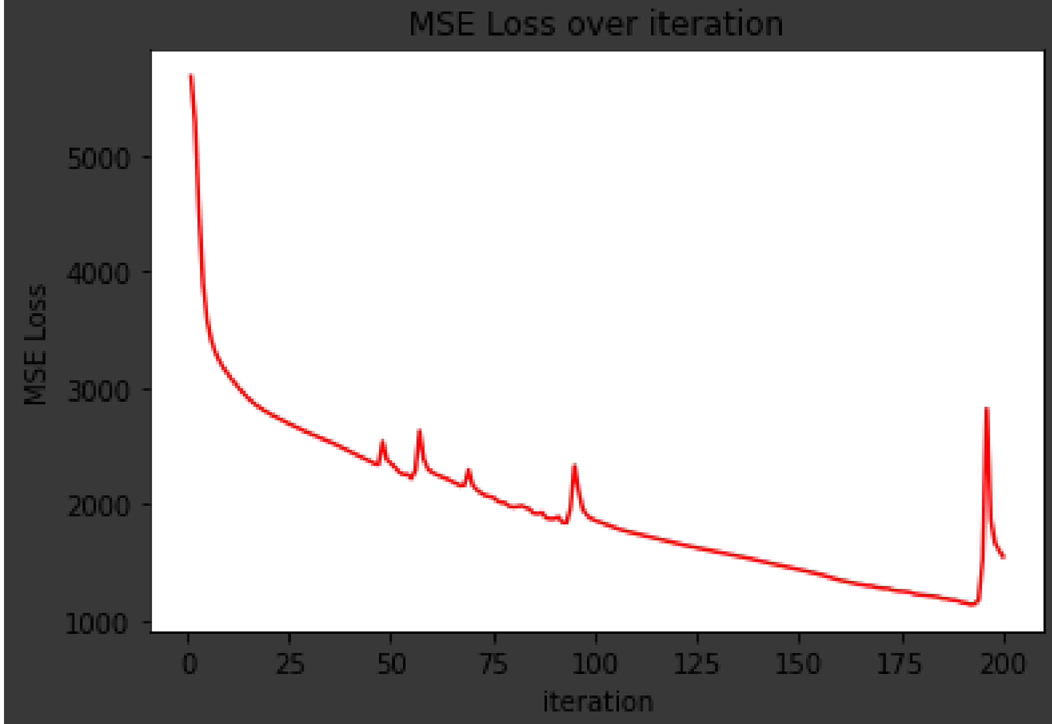
```
-------------------------------------------------------------
```



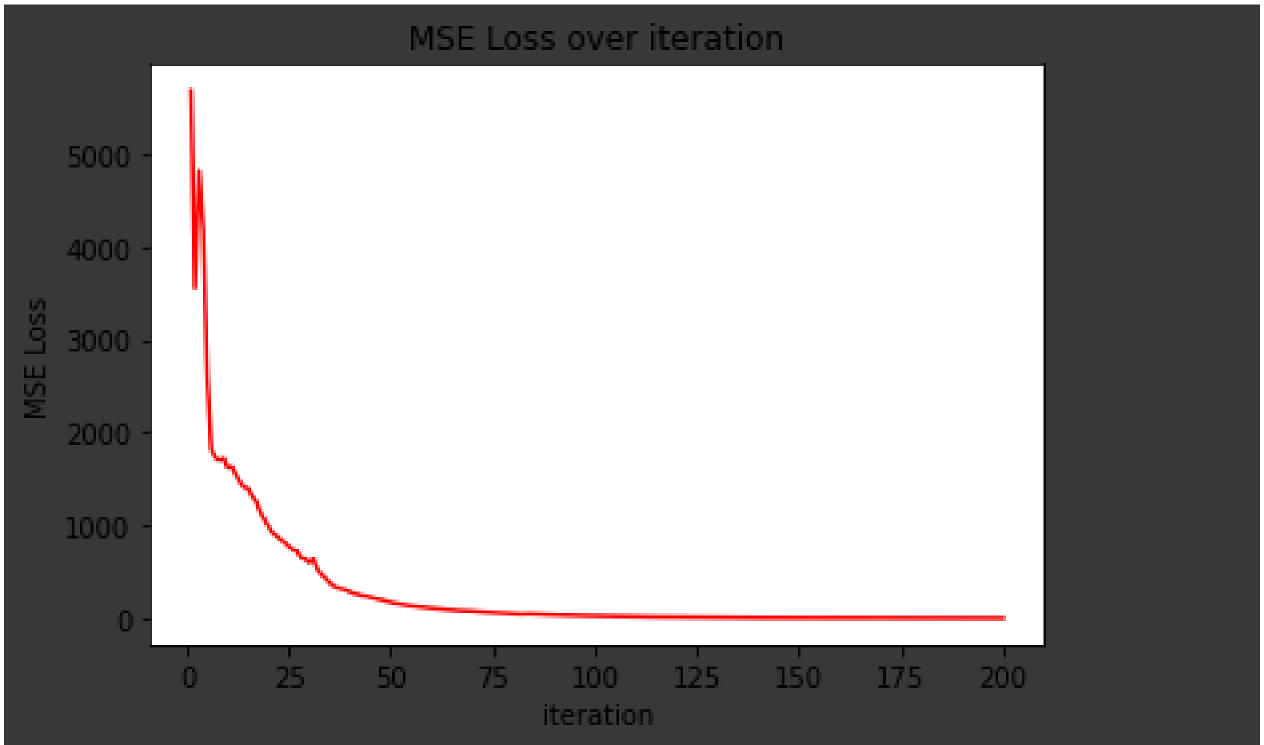
```
Average MSE loss on training dataset:  4966.687667877474
Average MSE loss on testing dataset:  4214.9667999013
learning rate:  0.001
-------------------------------------------------------------
```



```
Average MSE loss on training dataset:  3300.8508777303055
Average MSE loss on testing dataset:  3506.4493416002
learning rate:  0.01
-------------------------------------------------------------
```

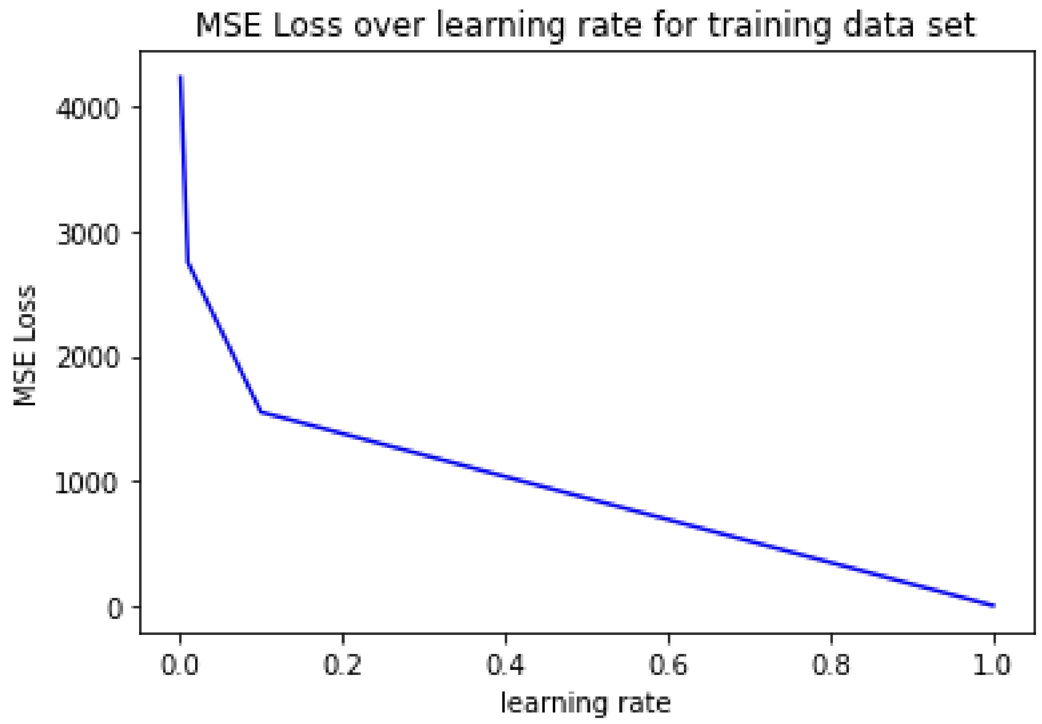


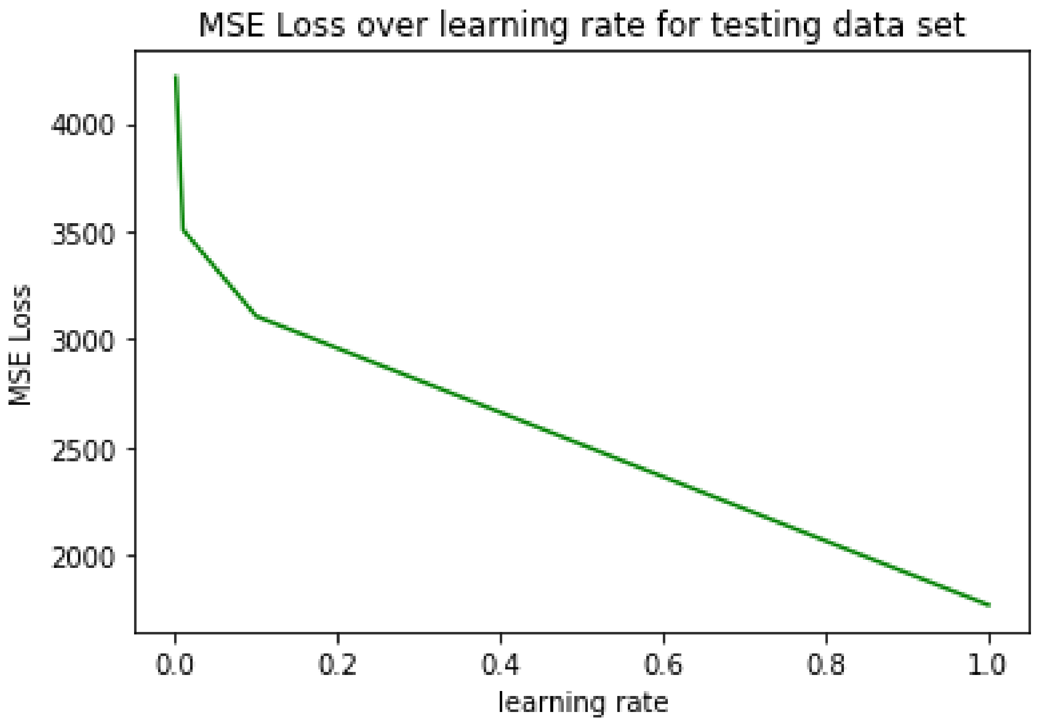
```
Average MSE loss on training dataset:  1989.0494326314613
Average MSE loss on testing dataset:  3106.7066921219
learning rate:  0.1
-------------------------------------------------------------
```



```
Average MSE loss on training dataset:  302.9964838443765
Average MSE loss on testing dataset:  1767.3839154766
learning rate:  1
```

5. What is the effect of the learning rate on the training process? Vary the learning rate to be between 0.001 and 1.0 and plot the resulting MSE loss as a function of learning rate.





6. What is the effect of the number of neurons in the hidden layer? To answer this question, you will need to consider and answer the following:

a. You will need to vary the number of neurons from 1 to 10. Does the update rule need to be changed/derived again? Why or why not?

The update rule does not need to be changed because the number of the neuron in hidden layer would not affect the update rule in the sense of refactoring the formula.

b. Report your observations by reporting the final loss along with a brief (2-3 lines) description.

```
Final MSE loss on training dataset: 1.0979625648 in hidden layer number 1
Final MSE loss on training dataset: 2.084290546 in hidden layer number 2
Final MSE loss on training dataset: 0.5602454605 in hidden layer number 3
Final MSE loss on training dataset: 850847841.5310996 in hidden layer number 4
Final MSE loss on training dataset: 1.0394267085795246e+145 in hidden layer number 5
Final MSE loss on training dataset: 1.1271581636765647e+142 in hidden layer number 6
Final MSE loss on training dataset: 1.7060057229619167e+184 in hidden layer number 7
Final MSE loss on training dataset: 2.1543523323553116e+246 in hidden layer number 8
Final MSE loss on training dataset: 5.686202313520691e+266 in hidden layer number 9
Final MSE loss on training dataset: 4.3223600630628663e+269 in hidden layer number 10
```

By the observation from the result, we can find that with 3 neurons in the hidden layer gives the best MSE loss value out of 10 test cases. The loss value increases rapidly when the number of neurons in the hidden layer increase.

7. What is the effect of the activation functions in the network? Explore two different activation functions other than sigmoid such as tanh, linear, or ReLU.

a. Will you need to change the update rule?

Yes, it needs to change the updated rule to try other activation functions.

b. What is the change that you need to make to achieve this experiment?

It needs to change the activation function and its derivative for calculating a1 and dz1.

c. Report your observations by reporting the final loss along with a brief (2-3 lines) description.

```
Activation function is tanh
Final MSE loss on training dataset: nan in hidden layer number 1
Final MSE loss on training dataset: nan in hidden layer number 2
Final MSE loss on training dataset: nan in hidden layer number 3
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:9: RuntimeWarning: overflow encountered in
exp
  if __name__ == '__main__':
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:9: RuntimeWarning: invalid value
encountered in true_divide
  if __name__ == '__main__':
Final MSE loss on training dataset: nan in hidden layer number 4
Final MSE loss on training dataset: nan in hidden layer number 5
Final MSE loss on training dataset: nan in hidden layer number 6
Final MSE loss on training dataset: nan in hidden layer number 7
Final MSE loss on training dataset: nan in hidden layer number 8
Final MSE loss on training dataset: nan in hidden layer number 9
Final MSE loss on training dataset: nan in hidden layer number 10
```

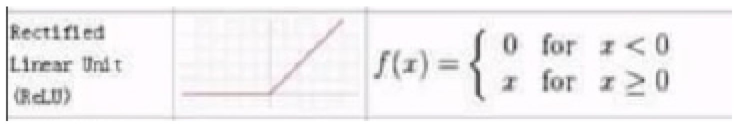
The Tanh function gives “nan” values for the MSE loss because the value z1 is too big to calculate exp^z1. Obviously, tanh wasn’t a good candidate for an activation function for the regression problem.

$$\tanh(a) = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}}$$

Exp of some number can grow exponentially which makes numpy to return nan.

```
Activation function is ReLU
Final MSE loss on training dataset: nan in hidden layer number 1
Final MSE loss on training dataset: nan in hidden layer number 2
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:23: RuntimeWarning: overflow encountered
in square
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:41: RuntimeWarning: invalid value
encountered in multiply
Final MSE loss on training dataset: nan in hidden layer number 3
Final MSE loss on training dataset: nan in hidden layer number 4
Final MSE loss on training dataset: nan in hidden layer number 5
Final MSE loss on training dataset: nan in hidden layer number 6
Final MSE loss on training dataset: nan in hidden layer number 7
Final MSE loss on training dataset: nan in hidden layer number 8
Final MSE loss on training dataset: nan in hidden layer number 9
Final MSE loss on training dataset: nan in hidden layer number 10
```

The ReLU function gives "nan" value for the MSE loss as well because the value z1 is too big to calculate numpy calculation. It is clear evidence that the ReLU function is not applicable for a regression problem



Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$

X can skyrocket.