Question 2

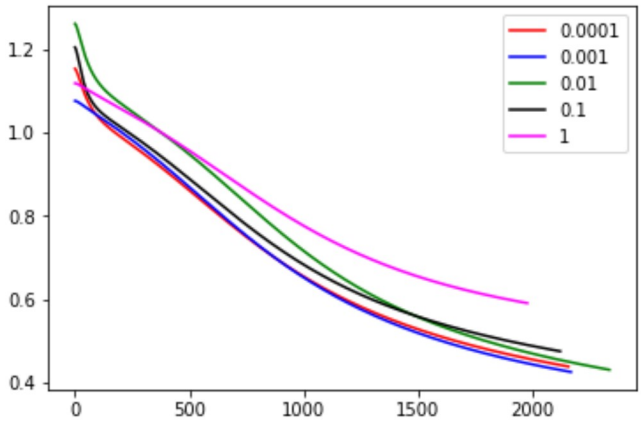
Write Python code to build a neural network with the following details.

- Input data = Iris dataset
- Number of hidden layers = 1
- Number of units in hidden layer = 10
- Number of iterations = 5000
- Learning algorithm = stochastic gradient descent
- Activation = logistic
- Learning rate = 0.0001, 0.001, 0.01, 0.1, 1

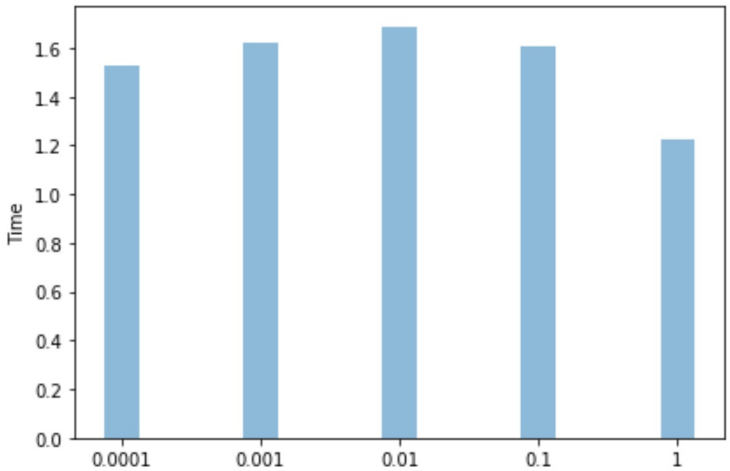
1. Compare the training score for each learning rate.
2. Plot the loss curve for each learning rate.
3. Report execution time for each learning rate as a bar graph. (Use library time and time() method)

Expectations

1. Expected output: (approximately)
    - Training accuracy 0.0001 is xx.xxx
    - Training accuracy 0.001 is xx.xxx
    - Training accuracy 0.01 is xx.xxx
    - Training accuracy 0.1 is xx.xxx
    - Training accuracy 1 is xx.xxx
2. Graph: Training Loss (Actual output may vary)



3. Bar graph: Execution Time (Actual output may vary)



You are expected to modify this notebook and upload the modified file as assignment submission.

PS: Code written within the block will be evaluted. Other code will be ignored.

start code here

end code here

In [ ]:
```
from sklearn import datasets
import numpy as np
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, classification_report, confu
sion_matrix

# Load Iris dataset.

iris = datasets.load_iris()

# Extract all columns except last from the dataset for X values.
# y is the target column.

X = iris["data"][:,:-1]
y = iris["target"]

# Split data into train and test

(X_train, X_test, y_train, y_test) = train_test_split(X, y, stratify=y,
test_size= 0.3)

# normalise the data
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

In [ ]:
```
# Use the library function sklearn.neural_network.MLPClassifier

# Build neural network for each learning rate. (max 10 lines of code) Us
e loop.

# start code here

# end code here
```

In [ ]:
```
# Compare the training  score for each learning rate. (max 2 lines of co
de) Use loop.

# start code here

# end code here
```

In [ ]:
```
# Plot the loss curve for each learning rate. (max 5 lines of code) Use
loop.

# start code here

# end code here
```

In [ ]:
```
# Plot the execution time as bar graph. (max 5 lines of code)

# start code here

# end code here
```