PRACTICAL 11

Plotting of Training and Validations Logs Using Matplotlib

ACTIVITY - Submission Required!

In this exercise, we will be creating a model to predict whether the person is suffering from heart disease or not. After training the model, we will plot the training and validation logs to visualize the performance of the training process.

The dataset for this tutorial can be downloaded from: Click here

Perform the following steps to complete this exercise:

1. Open a new Jupyter Notebook file and save it inside Google Drive.
2. Import the required libraries from the Keras and Pandas.

```
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
```

3. Load the **heart.csv** and assign to **df** variable.

```
df = ....................
```

4. Display the top and bottom 5 rows from the dataset.

```
df = ....................
```

5. Separate the features and labels. You can use **iloc** function of Pandas library.

```
X = ....................
y = ....................
```

6. Use **train_test_split** function from the Pandas module to split the dataset into train and test sets. Use 20% and 80% as test and train ratio respectively.

```
....................
....................
```

7. Build a model using the following layers:
    a. **Input Layer**: 13 neurons
    b. **First Hidden Layer**: 8 neurons
    c. **Second Hidden Layer**: 12 neurons
    d. **Third Hidden Layer**: 14 neurons
    e. **Output Layer**: 1 neurons with sigmoid activation function

8. Compile the model using **adam** as the optimizer, **binary cross entropy** as the loss function and **accuracy** as the **metrics**.
9. Fit the model using the code:

```
history = model.fit(X_train, y_train, epochs = 100, batch_size = 8,
validation_data = (X_test, y_test))
```

10. Plot the training and validation loss.

```
loss_train = history.history['loss']
loss_val = history.history['val_loss']
epochs = np.arange(1, 100)
plt.plot(loss_train, 'g')
plt.plot(loss_val, 'b')
plt.xticks(np.arange(0, N, 5))
plt.title('Training and Validation Loss')
plt.xlabel('Number of Epochs')
plt.ylabel('Loss')
plt.legend(['train','validation'])
plt.show()
```

11. Plot the training and validation accuracy.

```
acc_train = history.history['accuracy']
acc_val = history.history['val_accuracy']
epochs = np.arange(1, 100)
plt.plot(acc_train, 'g')
plt.plot(acc_val, 'b')
plt.xticks(np.arange(0, N, 5))
plt.title('Training and Validation Accuracy')
plt.xlabel('Number of Epochs')
plt.ylabel('Loss')
plt.legend(['train','validation'])
plt.show()
```

Activity

1. Explain the output generated in steps 10 and 11.
2. Retrain the model for 50 epochs and plot the training and validation graphs. You may have to edit the graph to plot the x-axis starting from 0 to 50.

****