UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING

SECB4313-01
BIOINFORMATICS MODELING AND SIMULATION
ASSIGNMENT 2

LECTURER:

DR. AZURAH BTE A SAMAH

GROUP MEMBERS:

AMIR ISKANDAR BIN NORKHAIRULAZADDIN (A20EC0011)

IMAN EHSAN BIN HASSAN (A20EC0048)

Identify 4 Hyperparameters and Propose Two Values for Each

1. The first hyperparameter is the learning rate for the optimizer. The learning rate values that were proposed are 0.01 and 0.1. This is because the learning rate controls how much the model is adjusted with respect to the loss gradient. The smaller values such as 0.01 allow for finer adjustments while larger values such as 0.1 help to enable faster learning.

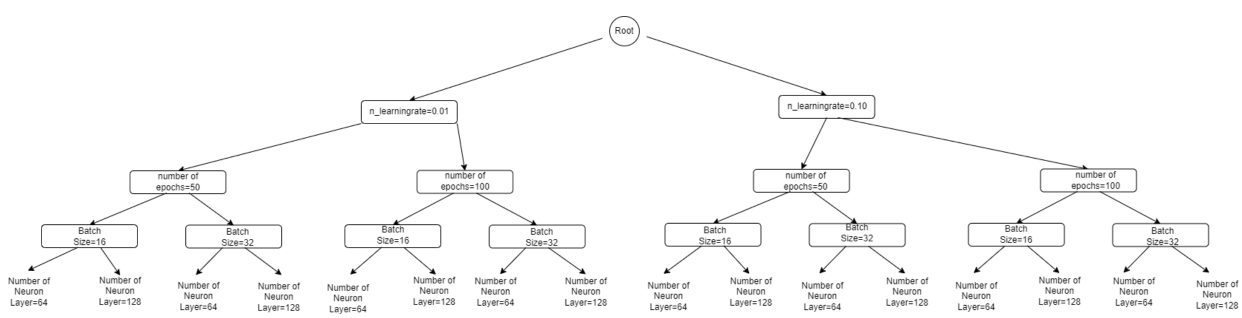
2. The number of epochs are 50 and 100, which defines how many times the learning algorithm will work through the entire training dataset. Increasing epochs can help the model learn more but too many epochs can lead to overfitting.

3. The batch size values are 16 and 32 where the batch size is the number of samples processed before the model is updated. When the smaller batches provide a regular, the noisy estimate of the gradient while larger batches result in a more stable estimate.

4. The number of neurons in hidden layers are 64 and 128 where the number of neurons in the hidden layers affects the model's ability to capture complex patterns.This is because more neurons can capture more detail but can also lead to overfitting and increased computational cost.

Tree Diagram

Below is the tree diagram that displays the proposed hyperparameters and its corresponding values. Each of these branch parameters was used for the model. Then, the accuracy score was calculated for each of the branches and it was tabulated in the next section.



Tabulated Experimental Design

Below is the table which consists of the experiments for which have different combinations of hyperparameters.

Experiment	Learning Rate	Number of Epochs	Batch Size	Neurons in Hidden Layer
0	0.01	50	16	64
1	0.01	50	16	128
2	0.01	50	32	64
3	0.01	50	32	128
4	0.01	100	16	64
5	0.01	100	16	128
6	0.01	100	32	64
7	0.01	100	32	128
8	0.10	50	16	64
9	0.10	50	16	128
10	0.10	50	32	64
11	0.10	50	32	128
12	0.10	100	16	64
13	0.10	100	16	128
14	0.10	100	32	64
15	0.10	100	32	128

Hyperparameter Tuning Simulation

Below is the code for the hyperparameter tuning to print out the best hyperparameter, best score and test accuracy of the model at the end of it.

```
for lr in learning_rates:
    for epoch in epochs:
        for batch_size in batch_sizes:
            for neuron in neurons:
                # Define the Keras model
                model = Sequential()
                model.add(Dense(neuron, input_dim=X_train.shape[1], activation='relu'))
                model.add(Dense(1, activation='sigmoid'))

                # Compile the model
                optimizer = Adam(learning_rate=lr)
                model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])

                # Train the model
                model.fit(X_train, y_train, epochs=epoch, batch_size=batch_size, verbose=0)

                # Evaluate the model
                _, accuracy = model.evaluate(X_test, y_test, verbose=0)

                # Record the results
                results.append({
                    'learning_rate': lr,
                    'epochs': epoch,
                    'batch_size': batch_size,
                    'neurons': neuron,
                    'accuracy': accuracy
                })
```

```
# Find the best hyperparameter combination
best_result = results_df.loc[results_df['accuracy'].idxmax()]
print("Best Hyperparameter Combination:")
print(best_result)
```

	learning_rate	epochs	batch_size	neurons	accuracy
0	0.01	50	16	64	0.868852
1	0.01	50	16	128	0.852459
2	0.01	50	32	64	0.852459
3	0.01	50	32	128	0.836066
4	0.01	100	16	64	0.852459
5	0.01	100	16	128	0.754098
6	0.01	100	32	64	0.868852
7	0.01	100	32	128	0.868852
8	0.10	50	16	64	0.524590
9	0.10	50	16	128	0.524590
10	0.10	50	32	64	0.852459
11	0.10	50	32	128	0.770492
12	0.10	100	16	64	0.524590
13	0.10	100	16	128	0.819672
14	0.10	100	32	64	0.819672
15	0.10	100	32	128	0.524590

```
Best Hyperparameter Combination:
learning_rate      0.010000
epochs            50.000000
batch_size        16.000000
neurons           64.000000
accuracy           0.868852
Name: 0, dtype: float64
```

Tabulated Results of the Proposed Experimental Design

Below is the table which holds the results for the experiments that were conducted from the code above. The accuracy score of each of the models was used as the performance measure. The model which has the highest accuracy has the best hyperparameters.

Experiment	Learning Rate	Number of Epochs	Batch Size	Neurons in Hidden Layer	Accuracy
0	0.01	50	16	64	0.868852
1	0.01	50	16	128	0.852459
2	0.01	50	32	64	0.852459
3	0.01	50	32	128	0.836066
4	0.01	100	16	64	0.852459
5	0.01	100	16	128	0.754098
6	0.01	100	32	64	0.868852
7	0.01	100	32	128	0.868852
8	0.10	50	16	64	0.524590
9	0.10	50	16	128	0.524590
10	0.10	50	32	64	0.852459
11	0.10	50	32	128	0.770492
12	0.10	100	16	64	0.524590
13	0.10	100	16	128	0.819672

14	0.10	100	32	64	0.819672
15	0.10	100	32	128	0.524590

Results Analysis

```
Best Hyperparameter Combination:
learning_rate     0.010000
epochs           50.000000
batch_size       16.000000
neurons          64.000000
accuracy          0.868852
Name: 0, dtype: float64
```

The best combination of hyperparameters is experiment number 0 where the value learning_rate of 0.01, 50 epochs, batch_size of 16 and the number of neurons in hidden layers is 64 which produce the accuracy score of 0.868852.

This is because the learning rate of 0.01 allowed for steady pace as if it is the rate too high, the model can be overshoot in optimal weight. For 50 epochs were sufficient for learning without overfitting. A batch size of 16 introduced good noise into the gradient estimates while helping in enhancing the generalization and for the 64 neurons in the hidden layer offered enough capacity to capture the data's complexity without overfitting.

In conclusion, we can say that these factors contribute to the model's performance and help to have effective configuration.