

# CS 203: Software Tools and Techniques for AI

## Assignment 1

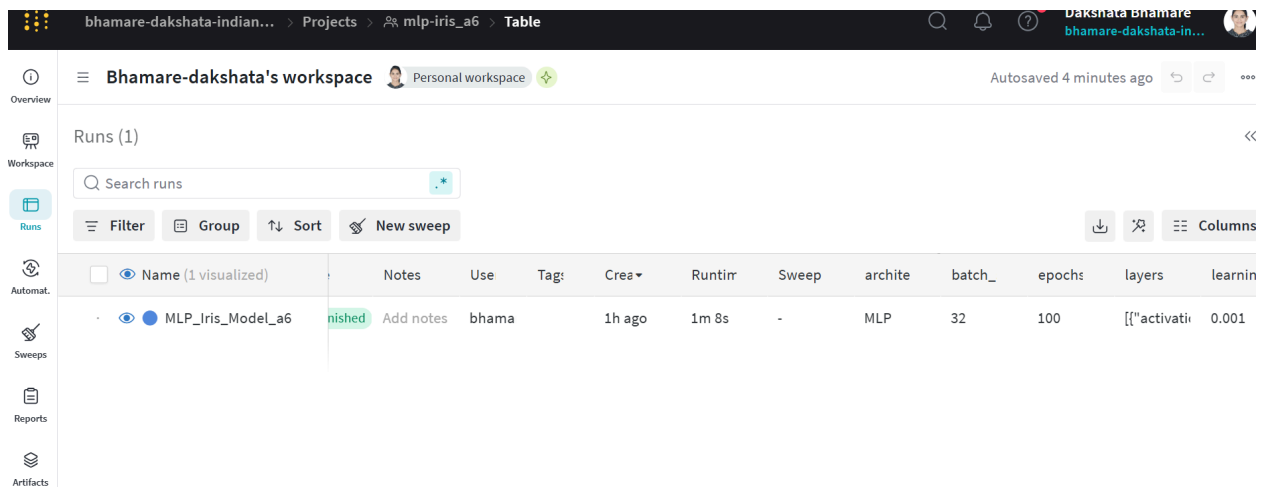
Chinmay Pendse (23110245)  
Bhamare Dakshata(23210027)

GitHub repository: Assignment\_6

### Section 1: Screenshots of the W&B dashboard displaying:

- Model architecture.
- Hyperparameters.
- Logged metrics.
- Final evaluation results.
- Confusion matrix visualization.
- Training and validation loss curves.

- 1) This screenshot displays the structure of your neural network, including the layers, number of epochs and hyperparameters. It helps visualize how data flows through the model.



- 2) Other evaluation results, such as precision, recall, and f1- score, are shown in the next screenshot.

This screenshot shows the MLflow interface for a workspace named 'Bhamare-dakshata's workspace'. It displays a table of runs for the model 'MLP\_Iris\_Model\_a6'. The table includes columns for accuracy, best epoch, best validation loss, epoch number, f1 score, final cost, loss, precision, recall, validation accuracy, and validation loss. The run 'MLP\_Iris\_Model\_a6' is highlighted, showing a validation accuracy of 0.9333 and a validation loss of 0.21823.

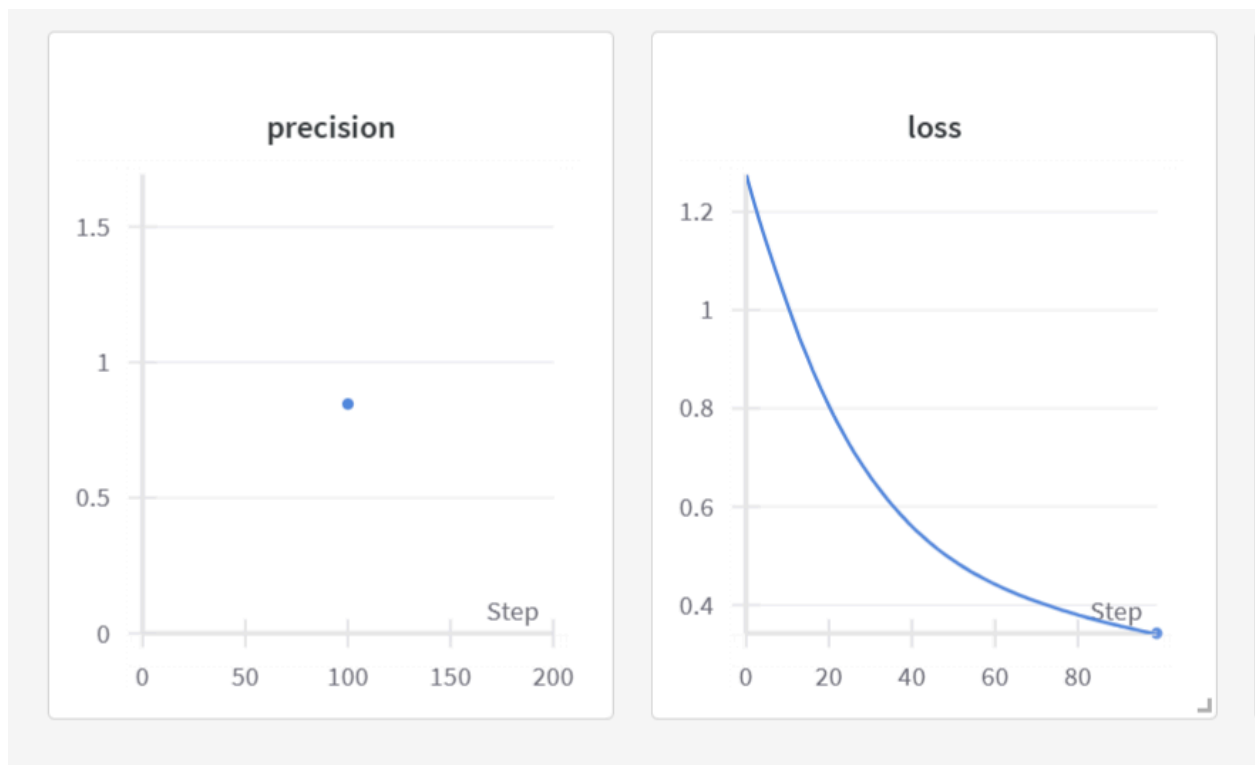
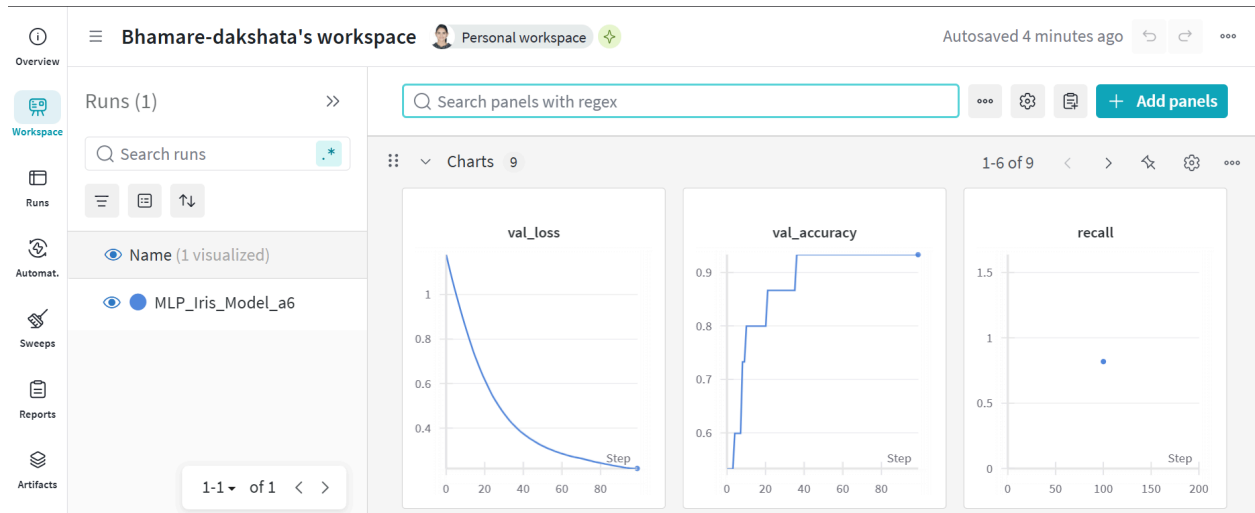
Name (1 visualized)	accuracy	best_epoch	best_val_loss	epoch	f1_score	final_cost	loss	precision	recall	val_accuracy	val_loss
MLP_Iris_Model_a6	0.8	99	0.21823	99	0.775	[[12,0,0],[12,0,0]]	0.34224	0.84615	0.81818	0.93333	0.21823

3) Next, the Screenshot captures logged matrices showing validation loss and accuracy.

This screenshot shows the MLflow interface for the same workspace, displaying a log of training metrics for the model 'MLP\_Iris\_Model\_a6'. The log shows the progress of training over 100 epochs, with metrics including accuracy, loss, validation accuracy, and validation loss. The log is filtered to show the last 100 epochs, with the first 10 epochs (0-9) being the most recent.

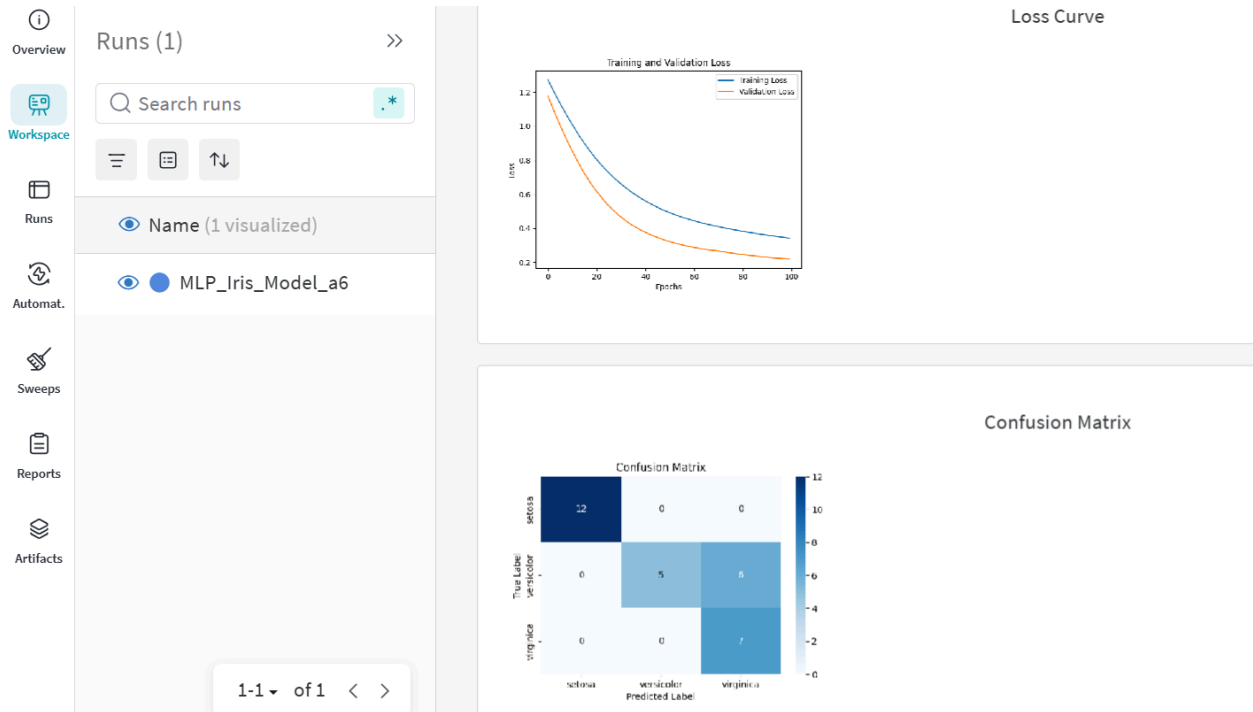
Epoch	Accuracy	Loss	Val Accuracy	Val Loss
234	0.8422	0.3690	0.9333	0.2365
235	0.8422	0.3690	0.9333	0.2365
236	0.8422	0.3690	0.9333	0.2365
237	0.8422	0.3690	0.9333	0.2365
238	0.8422	0.3690	0.9333	0.2365
239	0.8422	0.3690	0.9333	0.2365
240	0.8422	0.3690	0.9333	0.2365
241	0.8422	0.3690	0.9333	0.2365
242	0.8422	0.3690	0.9333	0.2365
243	0.8422	0.3690	0.9333	0.2365
244	0.8422	0.3690	0.9333	0.2365
245	0.8422	0.3690	0.9333	0.2365
246	0.8422	0.3690	0.9333	0.2365
247	0.8422	0.3690	0.9333	0.2365
248	0.8422	0.3690	0.9333	0.2365
249	0.8422	0.3690	0.9333	0.2365
250	0.8422	0.3690	0.9333	0.2365
251	0.8422	0.3690	0.9333	0.2365
252	0.8422	0.3690	0.9333	0.2365
253	0.8422	0.3690	0.9333	0.2365
254	0.8422	0.3690	0.9333	0.2365
255	0.8422	0.3690	0.9333	0.2365
256	0.8422	0.3690	0.9333	0.2365
257	0.8422	0.3690	0.9333	0.2365
258	0.8422	0.3690	0.9333	0.2365
259	0.8422	0.3690	0.9333	0.2365

4) The next two Screenshots capture graphs for accuracy, precision, recall and loss.



5) **Confusion Matrix Visualization** – shows the model's classification performance, highlighting correctly and incorrectly classified instances for each category.

**Training and Validation Loss Curves** – show the progression of training and validation loss over epochs, helping diagnose underfitting, overfitting, or convergence issues.



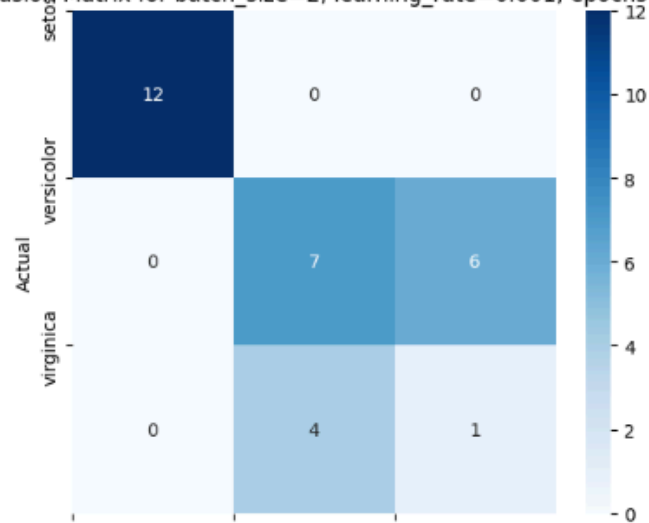
## Section 2: Hyperparameter Optimization and Automated Hyperparameter Search

### 1. Task 1

In this task, we trained the model by generating a parameter grid that stores all the values given in the assignment. Then, using the Keras module, we trained our model to classify the three using different epochs, learning rates and batch sizes. The confusion matrix for all of the cases is as follows.

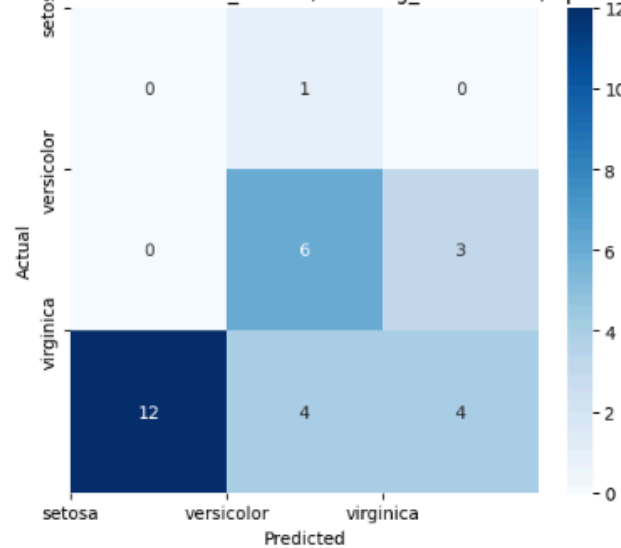
```
53/53 ————— 2s 13ms/step - accuracy: 0.5972 - loss: 1.1253 - val_accuracy: 0.6000 - val_loss: 0.9946  
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file fo  
1/1 ————— 0s 155ms/step
```

Confusion Matrix for batch\_size=2, learning\_rate=0.001, epochs=1



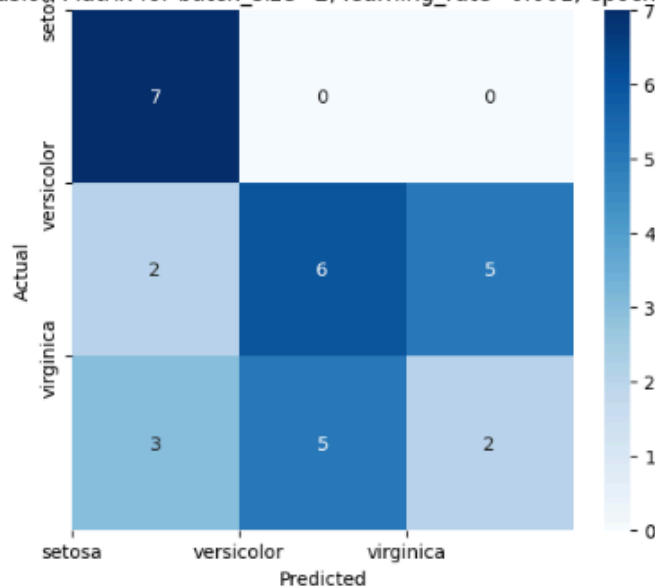
Training with: batch\_size=2, learning\_rate=1e-05, epochs=1  
 53/53 ————— 4s 21ms/step - accuracy: 0.3445 - loss: 1.5097 - val\_accuracy: 0.2667 - val\_loss: 1.3821  
 WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save\_model(model)`. This file format  
 1/1 ————— 0s 190ms/step

Confusion Matrix for batch\_size=2, learning\_rate=1e-05, epochs=1



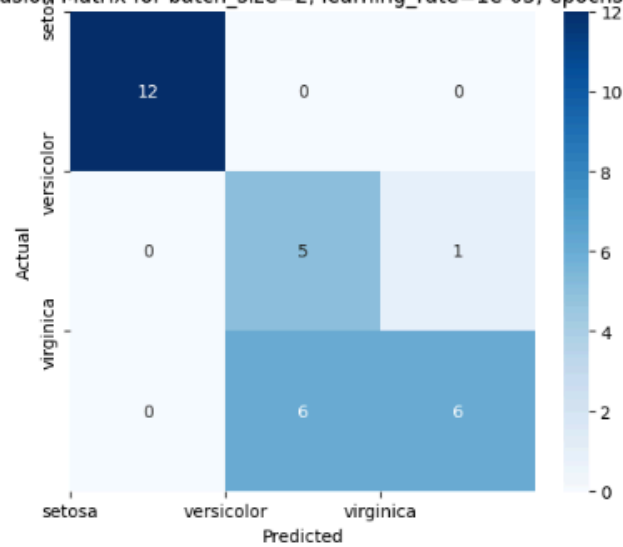
Training with: batch\_size=2, learning\_rate=0.001, epochs=3  
 53/53 ————— 2s 13ms/step - accuracy: 0.1074 - loss: 1.3483 - val\_accuracy: 0.3333 - val\_loss: 1.1058  
 WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save\_model(model)`. This file  
 1/1 ————— 0s 85ms/step

Confusion Matrix for batch\_size=2, learning\_rate=0.001, epochs=3

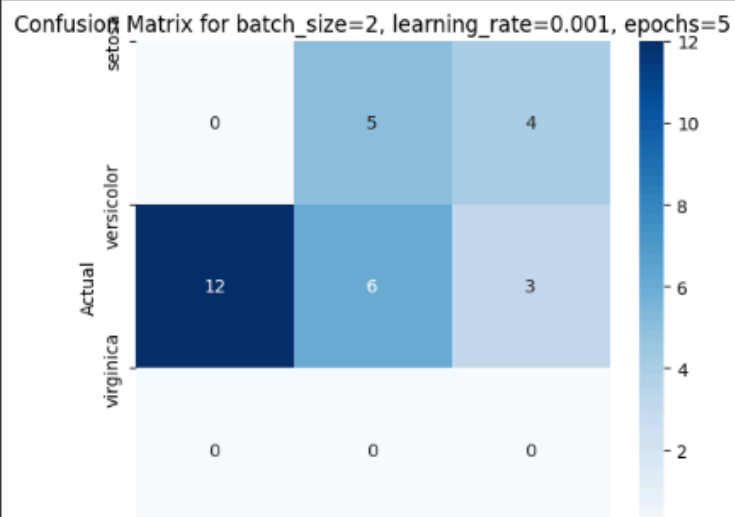


```
53/53 — 2s 10ms/step - accuracy: 0.6201 - loss: 0.9331 - val_accuracy: 0.8000 - val_loss: 0.7872
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format
WARNING:tensorflow:5 out of the last 5 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_di
1/1 — 0s 87ms/step
```

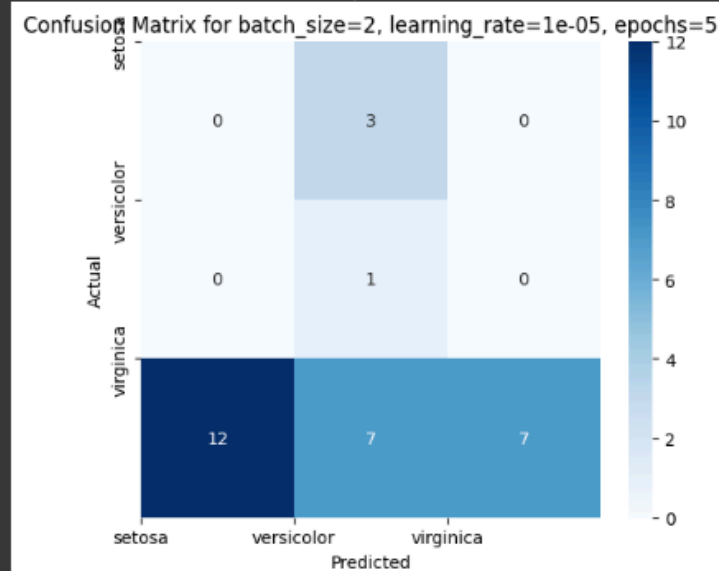
Confusion Matrix for batch\_size=2, learning\_rate=1e-05, epochs=3



Training with: batch\_size=2, learning\_rate=0.001, epochs=5  
 53/53 — 1s 6ms/step - accuracy: 0.2456 - loss: 1.7193 - val\_accuracy: 0.0667 - val\_loss: 1.6039  
 WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save\_model(model)`. This file format is deprecated, and will be removed in a future version of TensorFlow.  
 WARNING:tensorflow:6 out of the last 6 calls to <function TensorFlowTrainer.make\_predict\_function.<locals>.one\_step\_on\_data\_d  
 1/1 — 0s 70ms/step

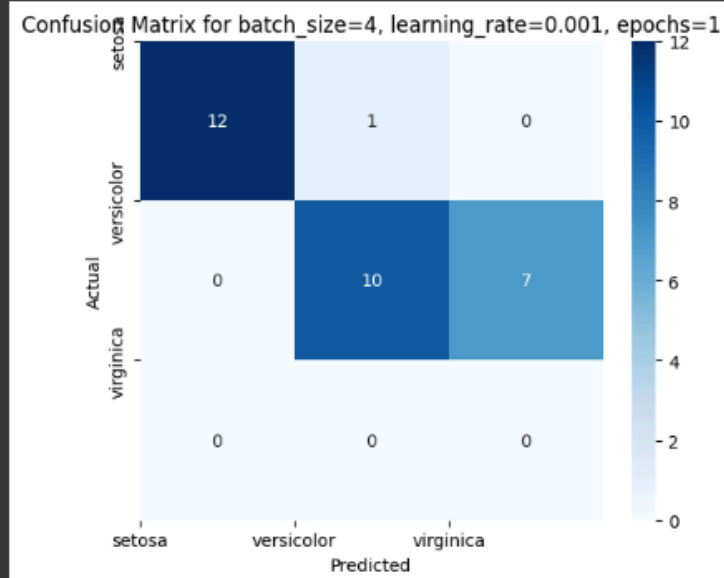


Training with: batch\_size=2, learning\_rate=1e-05, epochs=5  
 53/53 — 1s 7ms/step - accuracy: 0.1055 - loss: 1.2576 - val\_accuracy: 0.2000 - val\_loss: 1.0719  
 WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save\_model(model)`. This file format is deprecated, and will be removed in a future version of TensorFlow.  
 1/1 — 0s 59ms/step

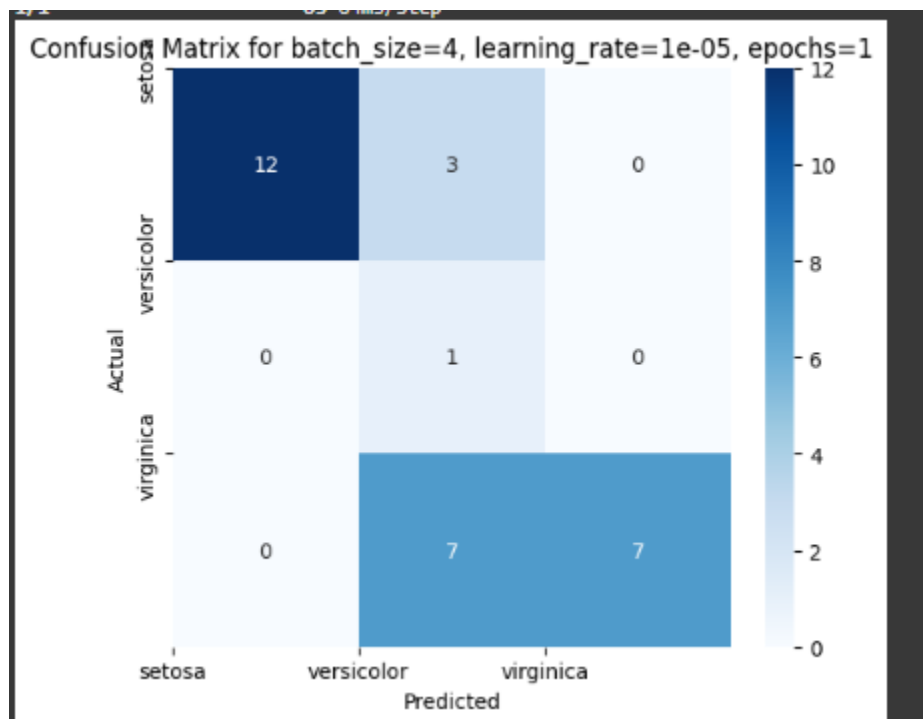


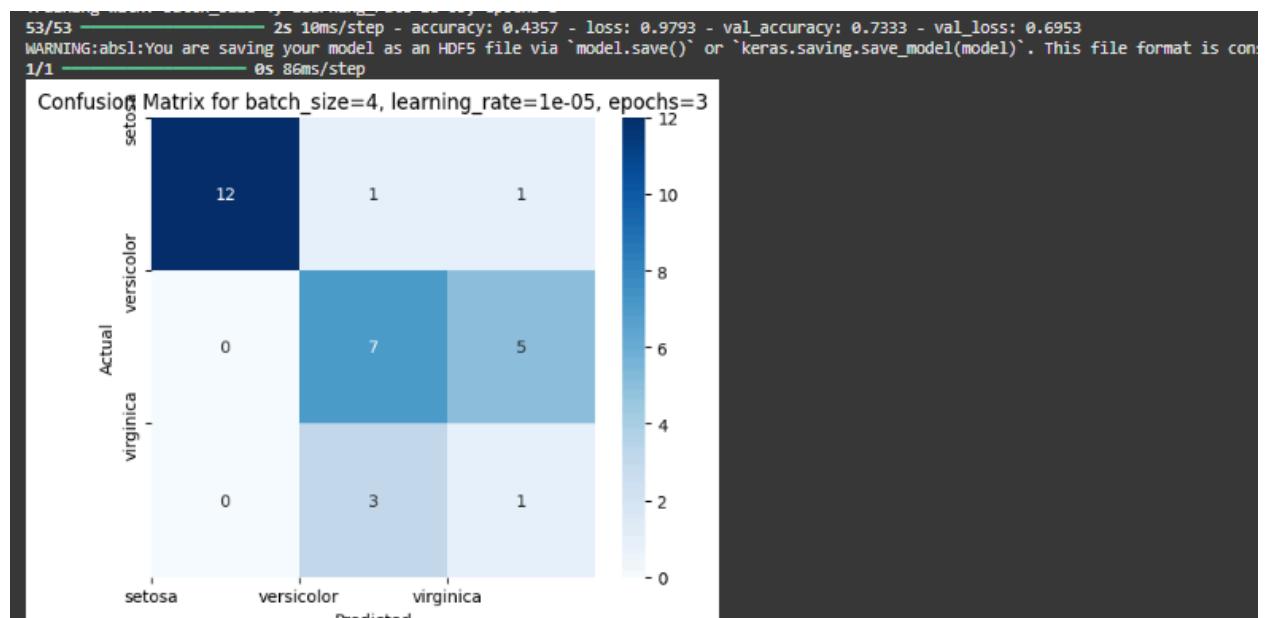
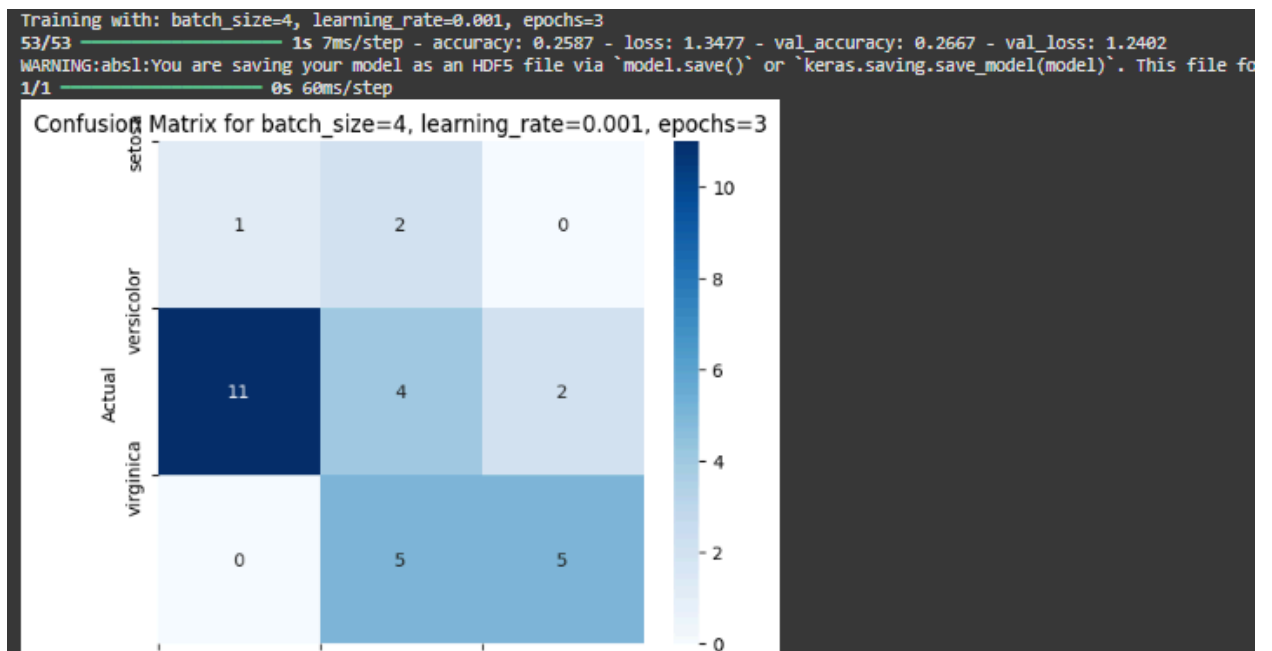


Training with: batch\_size=4, learning\_rate=0.001, epochs=1  
 53/53 1s 6ms/step - accuracy: 0.5408 - loss: 1.2154 - val\_accuracy: 0.6000 - val\_loss: 1.0161  
 WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save\_model(model)`. This file format is deprecated. It will be removed in a future version of TensorFlow. Please use the Keras format instead.  
 1/1 0s 63ms/step

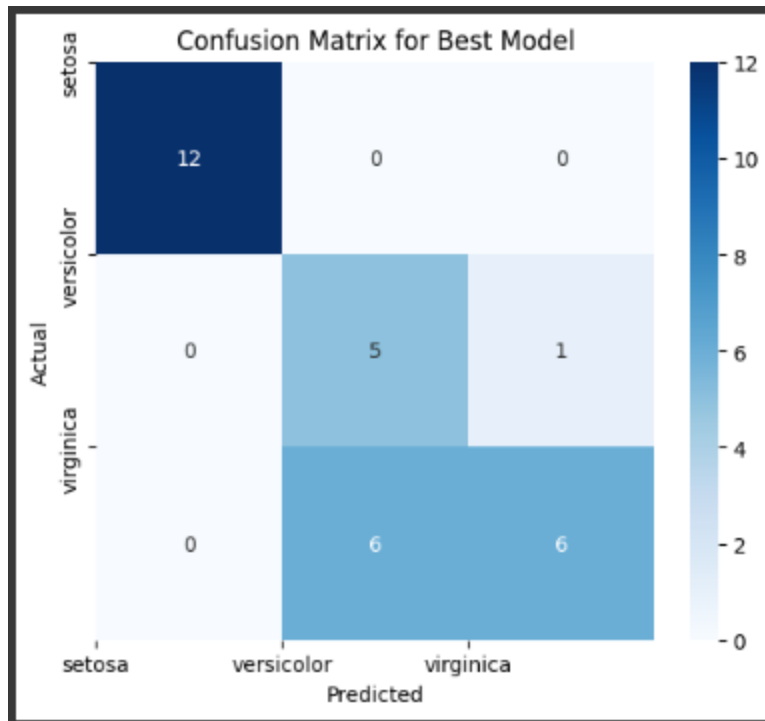


Training with: batch\_size=4, learning\_rate=1e-05, epochs=1  
 53/53 1s 6ms/step - accuracy: 0.4956 - loss: 0.9257 - val\_accuracy: 0.8667 - val\_loss: 0.6306  
 WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save\_model(model)`. This file format is deprecated. It will be removed in a future version of TensorFlow. Please use the Keras format instead.  
 1/1 0s 6ms/step





Now, from all this, we have the best possible model. Based on the accuracies and the losses, we compared all of them and found the best model.



## Sample Predictions

Sample Predictions:

Input: [-1.25747488 0.21701605 -1.32533157 -1.40568508]  
Prediction: 0, Truth: 0

Input: [-0.53399618 0.93659559 -1.38370397 -1.13511325]  
Prediction: 0, Truth: 0

Input: [-0.05167705 -0.74242333 0.01723376 -0.05282593]  
Prediction: 1, Truth: 1

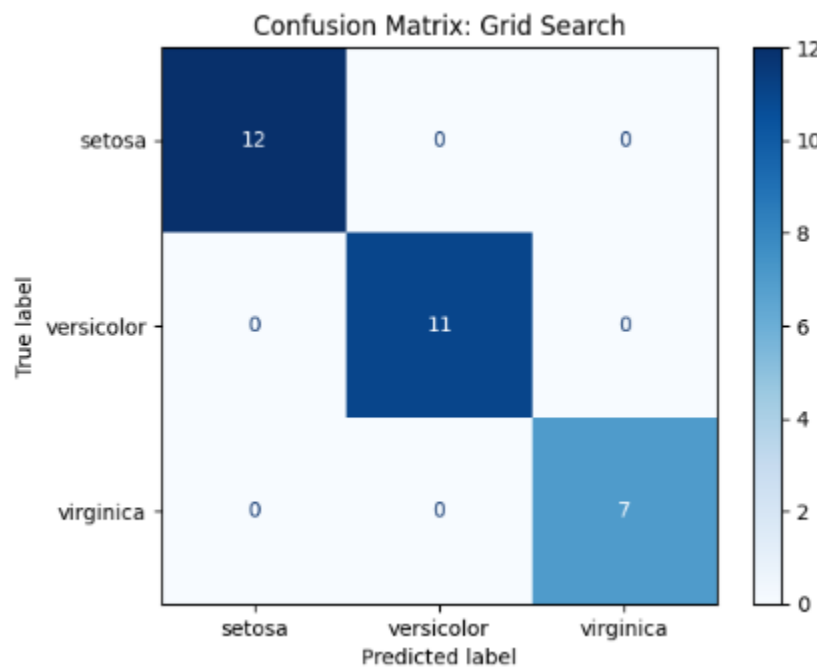
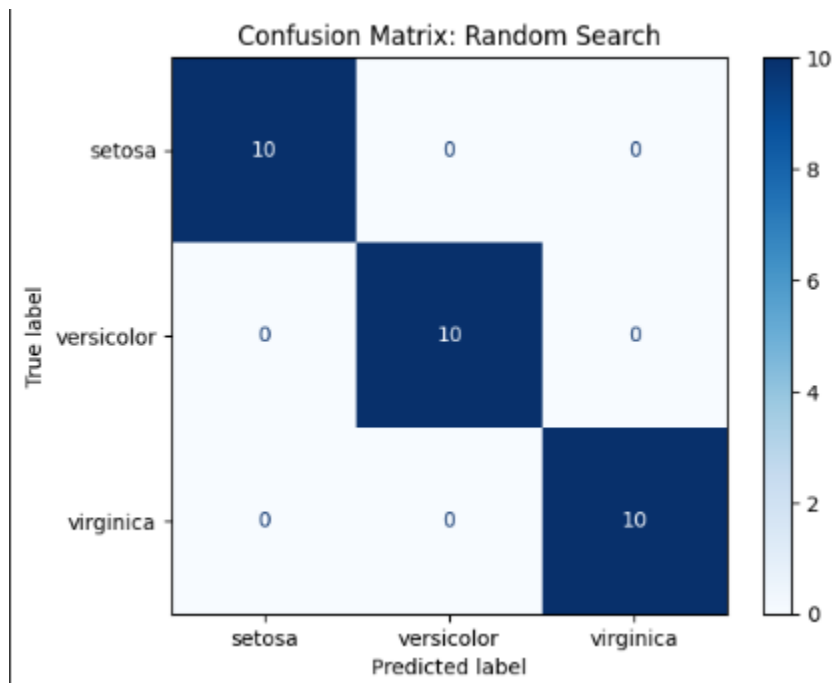
Input: [-1.49863445 1.41631528 -1.675566 -1.40568508]  
Prediction: 0, Truth: 0

Input: [-0.53399618 2.13589481 -1.50044878 -1.13511325]  
Prediction: 0, Truth: 0

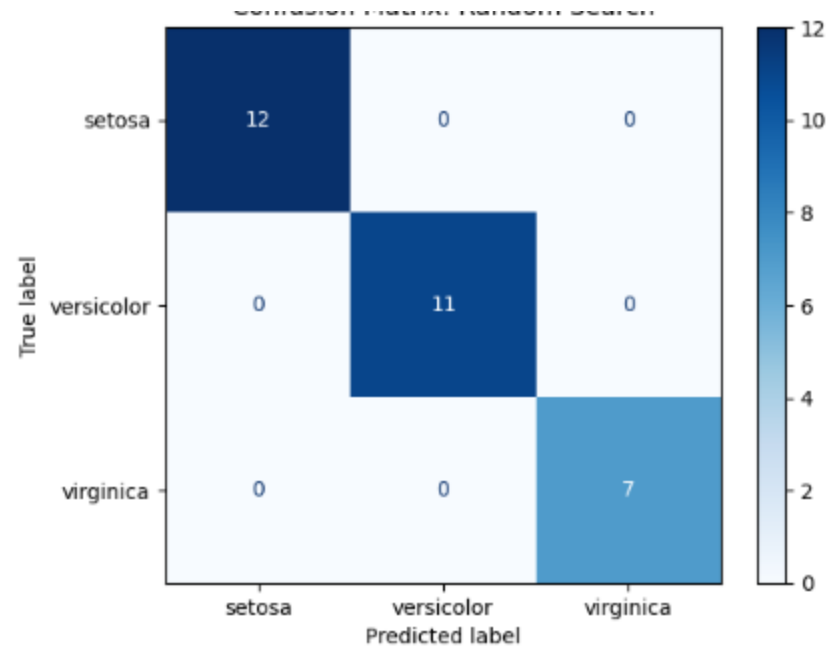
## 2. Task 2

For this task, we have used Autogloun TabularPredicto, where we manually define the hyperparameters `search_method` to `grid`, `random`, `auto` (for Bayesian (as the

documentation mentions that auto refers to Bayesian)), and hyperband. Then, we get the following results.



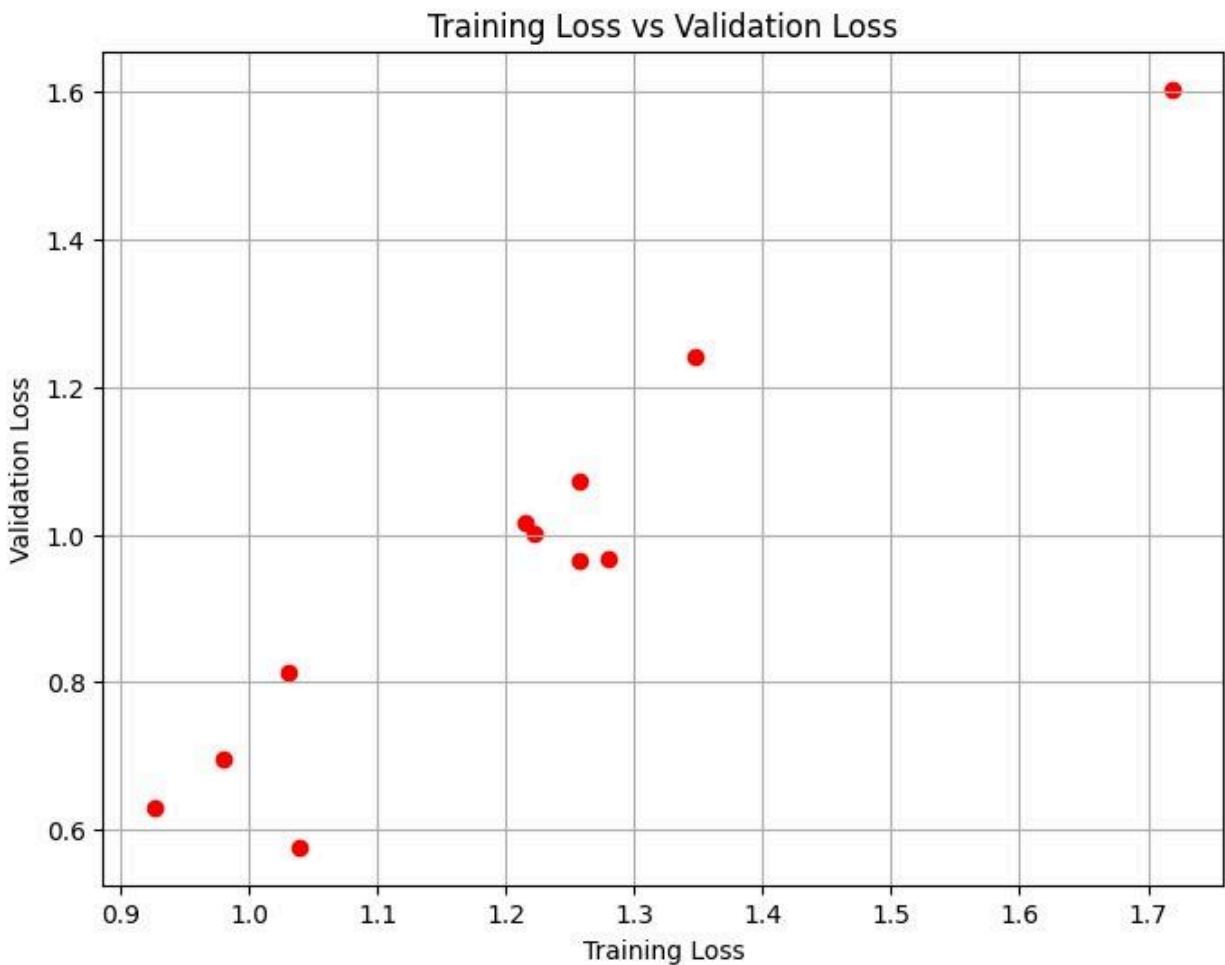
Hyperband Search



As per the theory, the accuracies follow the order of  
Hyperband > Grid > Random

However, since our dataset is small, we are getting 100 % accuracy everywhere, or this might be because the model is overfitting.

From the output which we got during the training, We manually listed down the losses and plotted them to obtain this



From this, we understand the difference between manual and automated tuning

### **Manual Tuning**

- We need to adjust hyperparameters manually based on intuition
- Simple but slow and inefficient for complex models.
- Requires trial-and-error runs

### **Automated Search**

- Uses algorithms (e.g., bayesian, grid, or random search) to find the best hyperparameters.
- Faster and avoids human bias
- Works well for deep learning or large models where manual tuning is impractical.

