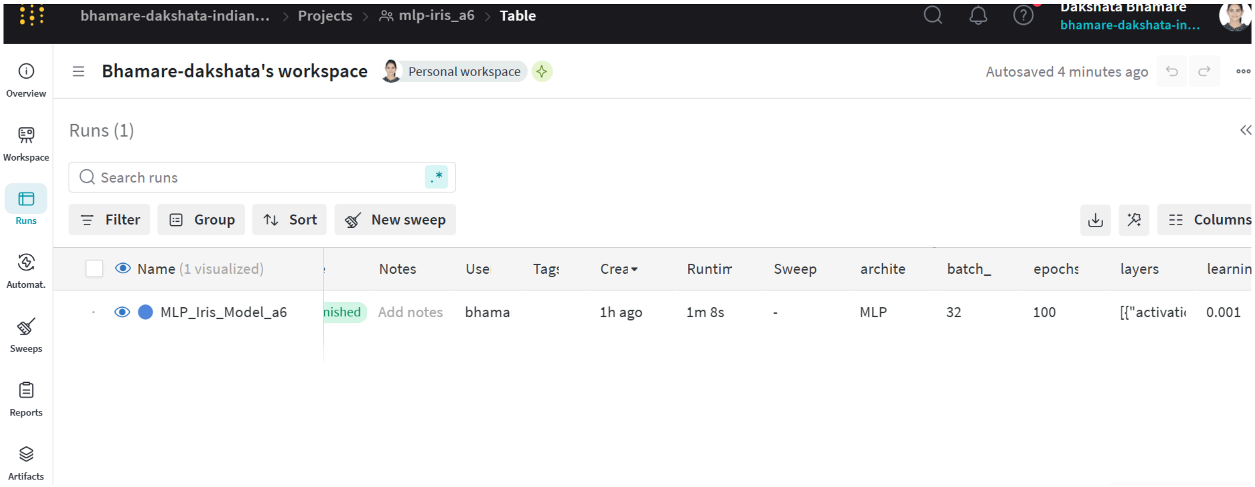
CS 203: Software Tools and Techniques for AI
Assignment 6

Chinmay Pendse (23110245)
Bhamare Dakshata(23210027)

GitHub repository: github

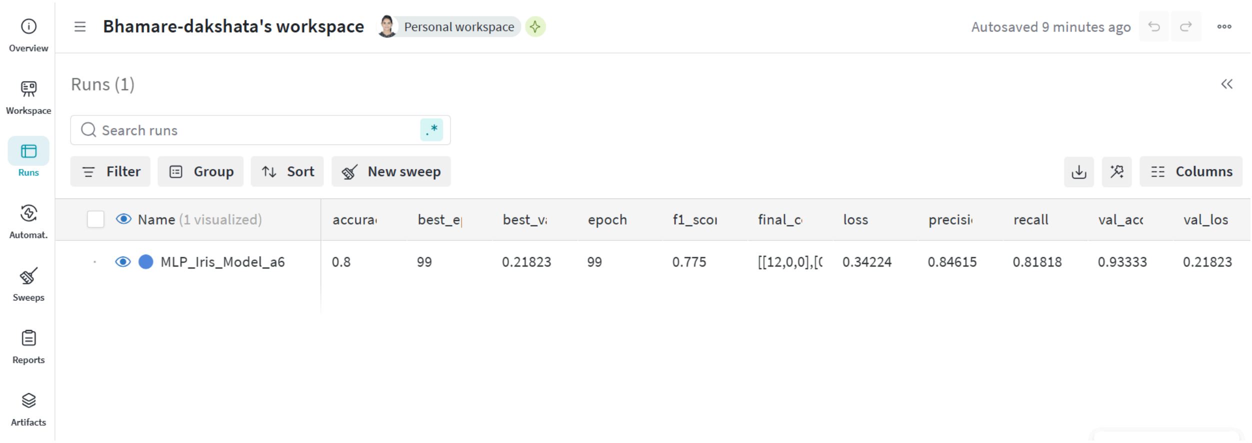
Section 1: Screenshots of the W&B dashboard displaying Model architecture., hyperparameters, Logged metrics, Final evaluation results, Confusion matrix visualization, Training and validation loss curves.

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



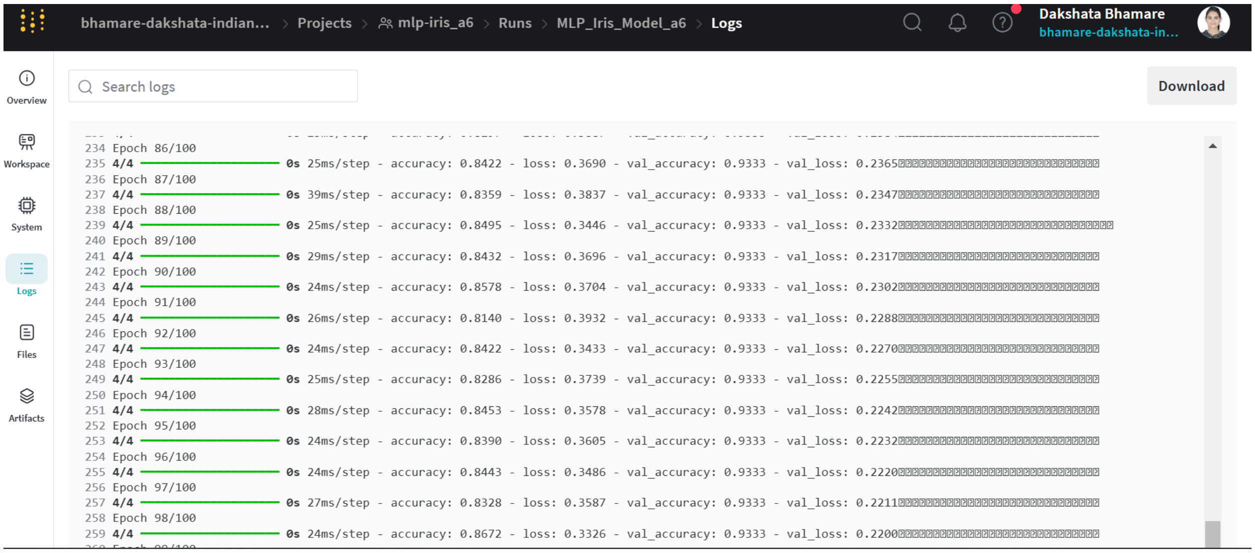
Name (1 visualized)		Notes	Use	Tag:	Crea	Runtir	Sweep	archite	batch_	epochs	layers	learnin
MLP_Iris_Model_a6	nished	Add notes	bhama		1h ago	1m 8s	-	MLP	32	100	[{"activati	0.001

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



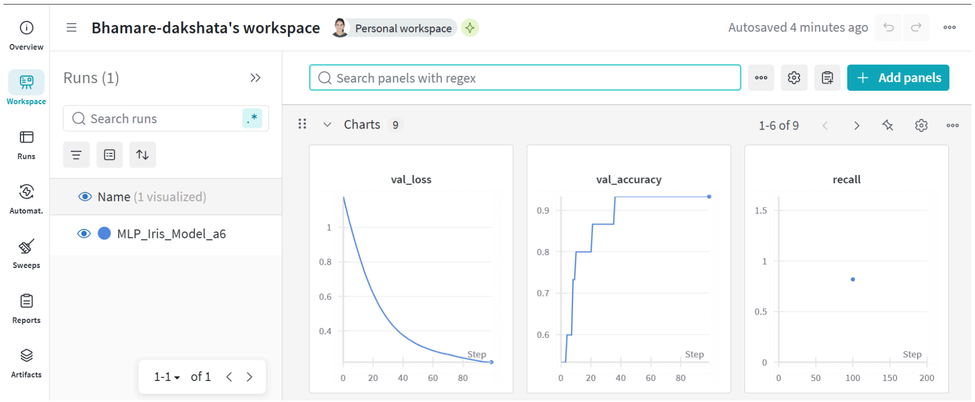
Name (1 visualized)	accura	best_e	best_v	epoch	f1_scoi	final_c	loss	precisi	recall	val_acc	val_los
MLP_Iris_Model_a6	0.8	99	0.21823	99	0.775	[[12,0,0],[(	0.34224	0.84615	0.81818	0.93333	0.21823

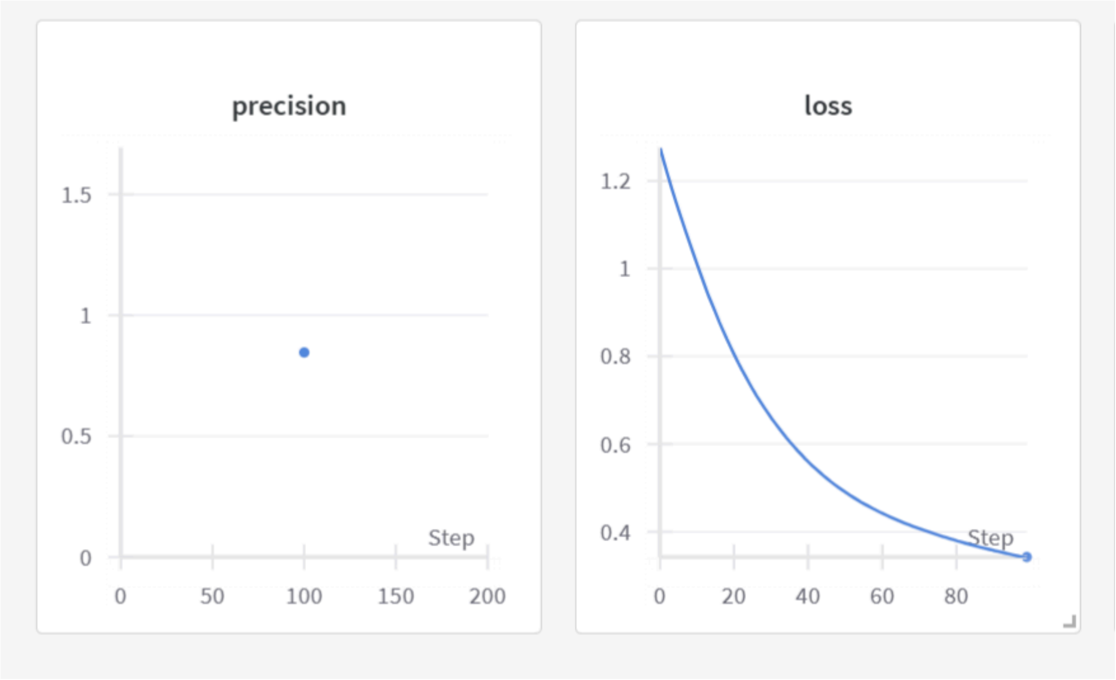
3) Next, the Screenshot captures logged matrics showing validation loss and accuracy.



```
Epoch 86/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.8422 - loss: 0.3690 - val_accuracy: 0.9333 - val_loss: 0.2365
Epoch 87/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - accuracy: 0.8359 - loss: 0.3837 - val_accuracy: 0.9333 - val_loss: 0.2347
Epoch 88/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.8495 - loss: 0.3446 - val_accuracy: 0.9333 - val_loss: 0.2332
Epoch 89/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.8432 - loss: 0.3696 - val_accuracy: 0.9333 - val_loss: 0.2317
Epoch 90/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.8578 - loss: 0.3704 - val_accuracy: 0.9333 - val_loss: 0.2302
Epoch 91/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.8140 - loss: 0.3932 - val_accuracy: 0.9333 - val_loss: 0.2288
Epoch 92/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.8422 - loss: 0.3433 - val_accuracy: 0.9333 - val_loss: 0.2270
Epoch 93/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.8286 - loss: 0.3739 - val_accuracy: 0.9333 - val_loss: 0.2255
Epoch 94/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.8453 - loss: 0.3578 - val_accuracy: 0.9333 - val_loss: 0.2242
Epoch 95/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.8390 - loss: 0.3605 - val_accuracy: 0.9333 - val_loss: 0.2232
Epoch 96/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.8443 - loss: 0.3486 - val_accuracy: 0.9333 - val_loss: 0.2220
Epoch 97/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - accuracy: 0.8328 - loss: 0.3587 - val_accuracy: 0.9333 - val_loss: 0.2211
Epoch 98/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.8672 - loss: 0.3326 - val_accuracy: 0.9333 - val_loss: 0.2200
```

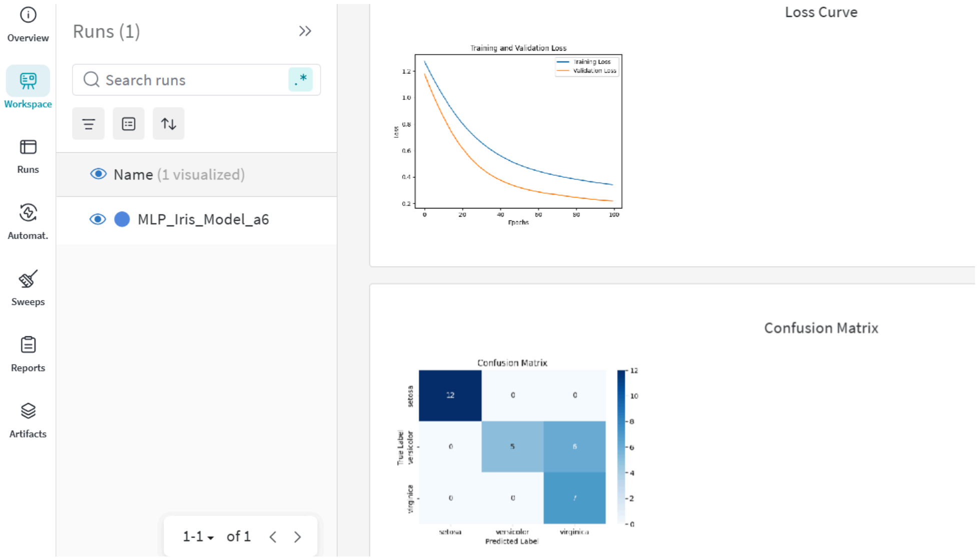
4) The next two Screenshots capture graphs for accuracy, precision, recall and loss.





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.

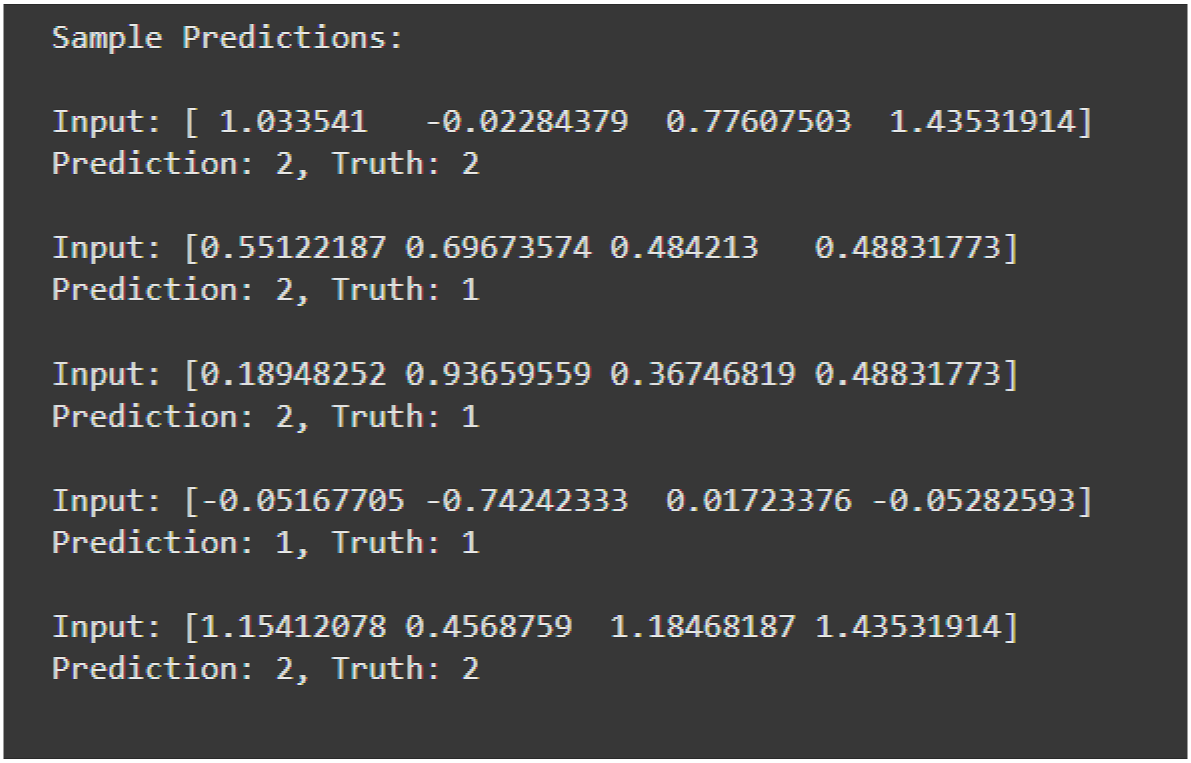


	setosa	versicolor	virginica
setosa	12	0	0
versicolor	0	5	6
virginica	0	0	7

Section 2: Hyperparameter Optimization and Automated Hyperparameter Search

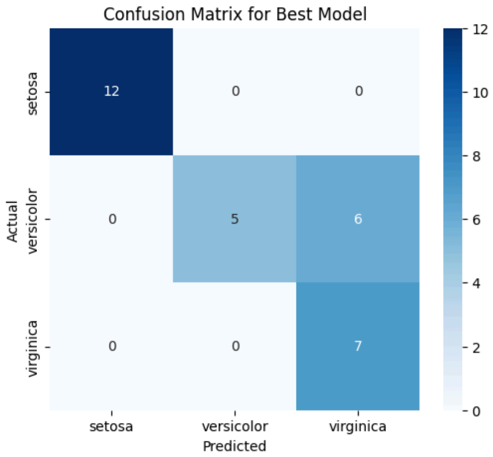
1. **Task 1:** In this task, we trained the model by generating a parameter grid that stores all the values given in the assignment. Then, using the Keras module, we trained our model to classify the three using different epochs, learning rates and batch sizes.

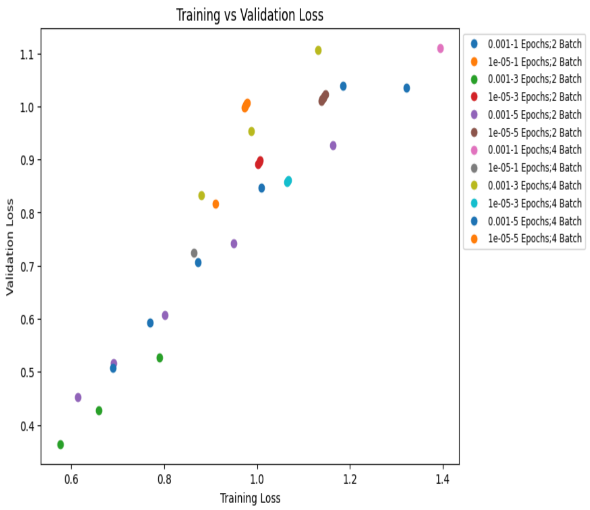
- Inputs, prediction, and truth values for five samples from the test set.



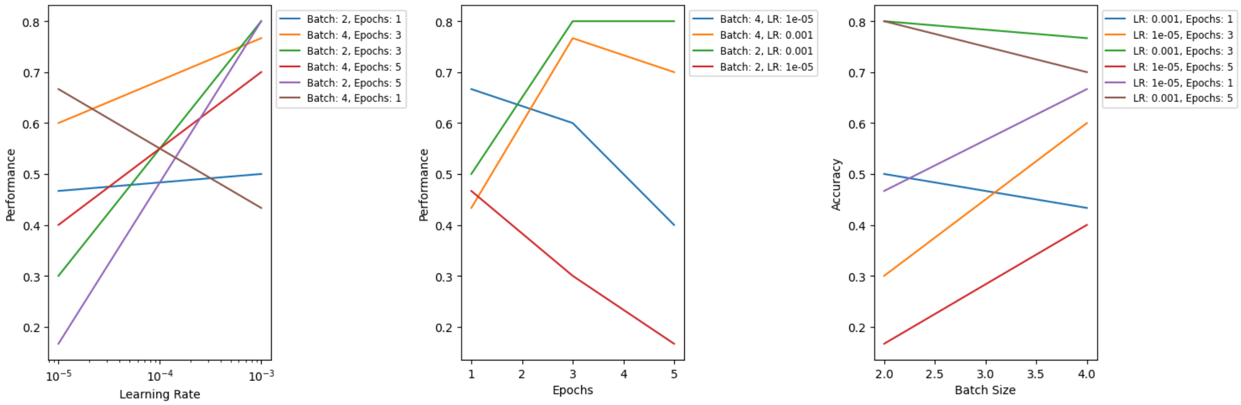
```
Sample Predictions:

Input: [ 1.033541   -0.02284379  0.77607503  1.43531914]
Prediction: 2, Truth: 2

Input: [0.55122187 0.69673574 0.484213   0.48831773]
Prediction: 2, Truth: 1

Input: [0.18948252 0.93659559 0.36746819 0.48831773]
Prediction: 2, Truth: 1

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

Input: [1.15412078 0.4568759  1.18468187 1.43531914]
Prediction: 2, Truth: 2
```

The confusion matrix for all of the cases we got. 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. From the output we got during the training, this is a scatter plot for training vs validation loss.

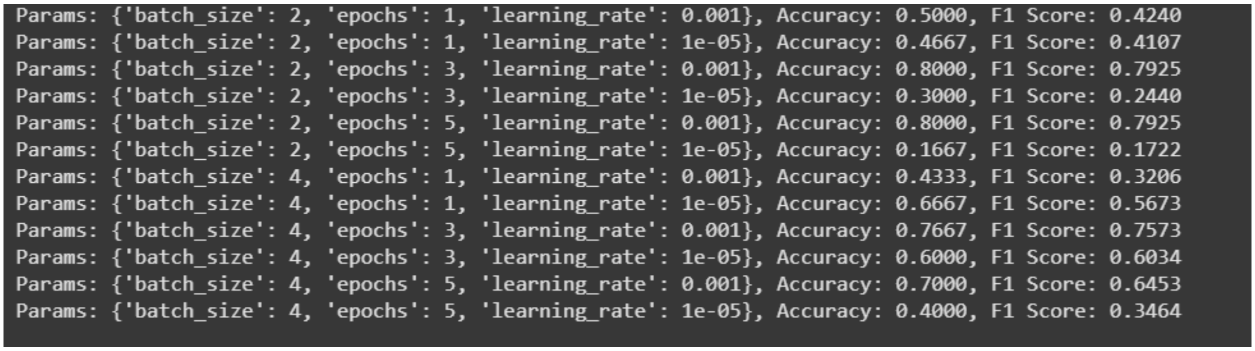




- A relation between the hyperparameters and their impact on the performance.



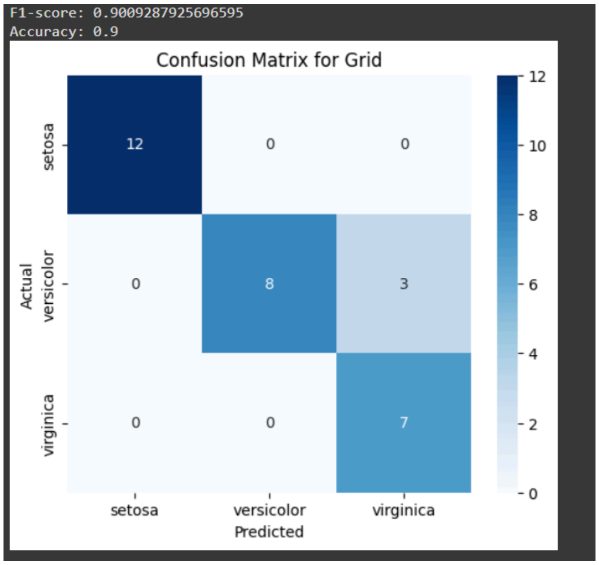
- performance for each hyperparameter combination over accuracy and F1.



```
Params: {'batch_size': 2, 'epochs': 1, 'learning_rate': 0.001}, Accuracy: 0.5000, F1 Score: 0.4240
Params: {'batch_size': 2, 'epochs': 1, 'learning_rate': 1e-05}, Accuracy: 0.4667, F1 Score: 0.4107
Params: {'batch_size': 2, 'epochs': 3, 'learning_rate': 0.001}, Accuracy: 0.8000, F1 Score: 0.7925
Params: {'batch_size': 2, 'epochs': 3, 'learning_rate': 1e-05}, Accuracy: 0.3000, F1 Score: 0.2440
Params: {'batch_size': 2, 'epochs': 5, 'learning_rate': 0.001}, Accuracy: 0.8000, F1 Score: 0.7925
Params: {'batch_size': 2, 'epochs': 5, 'learning_rate': 1e-05}, Accuracy: 0.1667, F1 Score: 0.1722
Params: {'batch_size': 4, 'epochs': 1, 'learning_rate': 0.001}, Accuracy: 0.4333, F1 Score: 0.3206
Params: {'batch_size': 4, 'epochs': 1, 'learning_rate': 1e-05}, Accuracy: 0.6667, F1 Score: 0.5673
Params: {'batch_size': 4, 'epochs': 3, 'learning_rate': 0.001}, Accuracy: 0.7667, F1 Score: 0.7573
Params: {'batch_size': 4, 'epochs': 3, 'learning_rate': 1e-05}, Accuracy: 0.6000, F1 Score: 0.6034
Params: {'batch_size': 4, 'epochs': 5, 'learning_rate': 0.001}, Accuracy: 0.7000, F1 Score: 0.6453
Params: {'batch_size': 4, 'epochs': 5, 'learning_rate': 1e-05}, Accuracy: 0.4000, F1 Score: 0.3464
```

2. Task 2

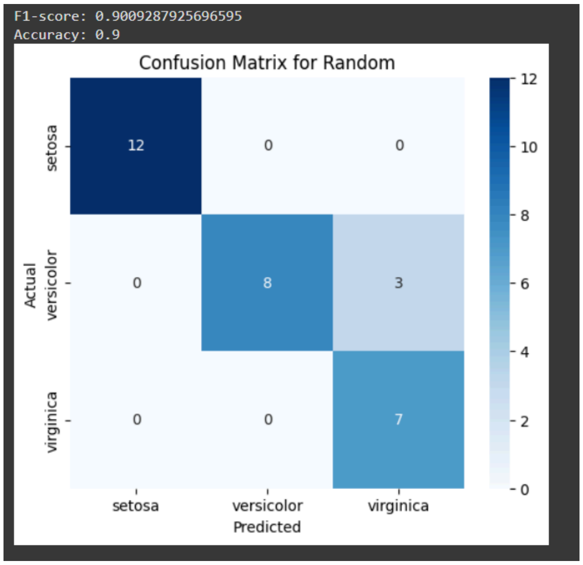
For this task, we have used Autogloun TabularPredictor, where we manually define the hyperparameters search_method to grid, random, auto (for Bayesian (as the documentation mentions that auto refers to Bayesian)), and hyperband. Then, we get the following results.



```
F1-score: 0.9009287925696595
Accuracy: 0.9
```

Confusion Matrix for Grid

Actual \ Predicted	setosa	versicolor	virginica
setosa	12	0	0
versicolor	0	8	3
virginica	0	0	7

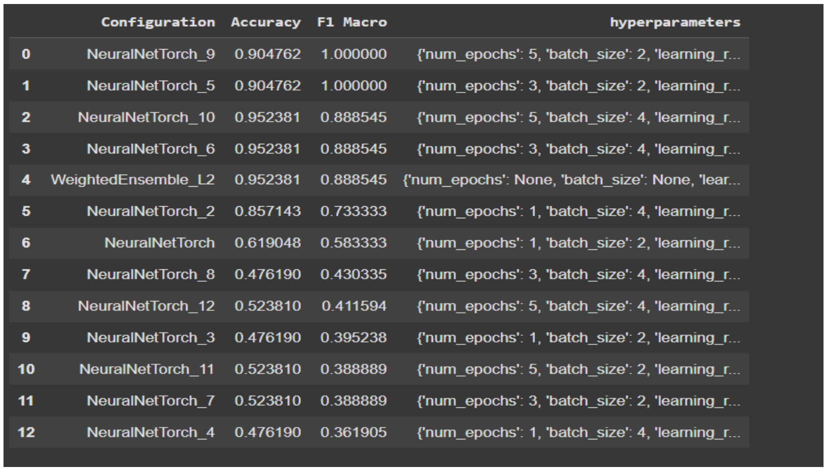


```
F1-score: 0.9009287925696595
Accuracy: 0.9
```

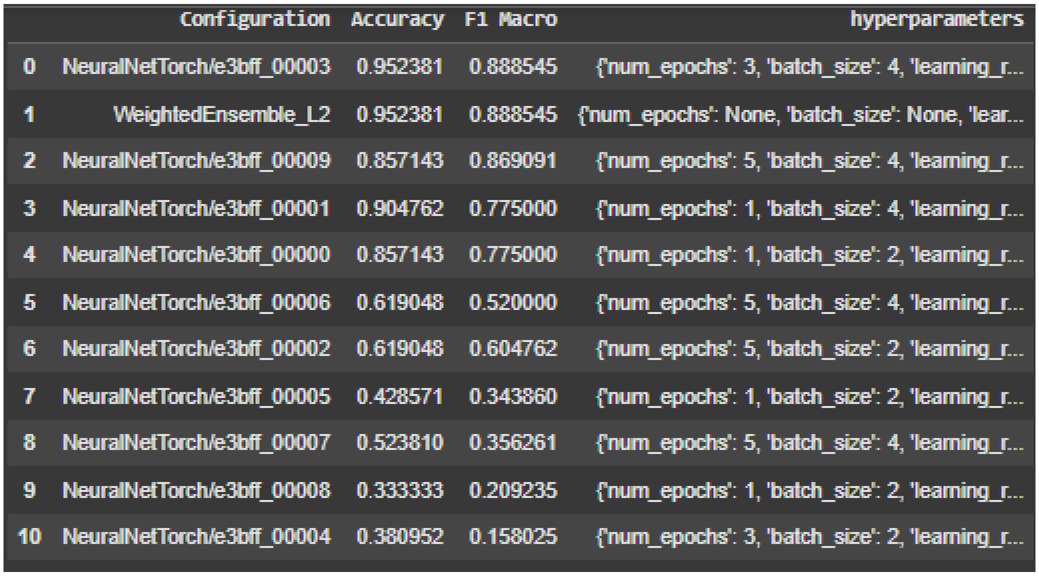
Confusion Matrix for Random

Actual \ Predicted	setosa	versicolor	virginica
setosa	12	0	0
versicolor	0	8	3
virginica	0	0	7

This confusion matrix we got after using Grid and Random search over the defined hyperparameters.

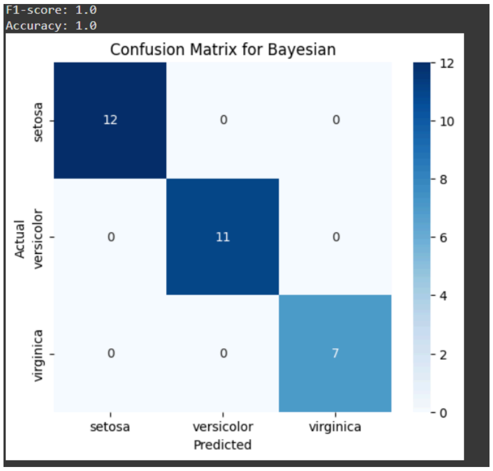


	Configuration	Accuracy	F1 Macro	hyperparameters
0	NeuralNetTorch_9	0.904762	1.000000	{'num_epochs': 5, 'batch_size': 2, 'learning_r...
1	NeuralNetTorch_5	0.904762	1.000000	{'num_epochs': 3, 'batch_size': 2, 'learning_r...
2	NeuralNetTorch_10	0.952381	0.888545	{'num_epochs': 5, 'batch_size': 4, 'learning_r...
3	NeuralNetTorch_6	0.952381	0.888545	{'num_epochs': 3, 'batch_size': 4, 'learning_r...
4	WeightedEnsemble_L2	0.952381	0.888545	{'num_epochs': None, 'batch_size': None, 'lear...
5	NeuralNetTorch_2	0.857143	0.733333	{'num_epochs': 1, 'batch_size': 4, 'learning_r...
6	NeuralNetTorch	0.619048	0.583333	{'num_epochs': 1, 'batch_size': 2, 'learning_r...
7	NeuralNetTorch_8	0.476190	0.430335	{'num_epochs': 3, 'batch_size': 4, 'learning_r...
8	NeuralNetTorch_12	0.523810	0.411594	{'num_epochs': 5, 'batch_size': 4, 'learning_r...
9	NeuralNetTorch_3	0.476190	0.395238	{'num_epochs': 1, 'batch_size': 2, 'learning_r...
10	NeuralNetTorch_11	0.523810	0.388889	{'num_epochs': 5, 'batch_size': 2, 'learning_r...
11	NeuralNetTorch_7	0.523810	0.388889	{'num_epochs': 3, 'batch_size': 2, 'learning_r...
12	NeuralNetTorch_4	0.476190	0.361905	{'num_epochs': 1, 'batch_size': 4, 'learning_r...

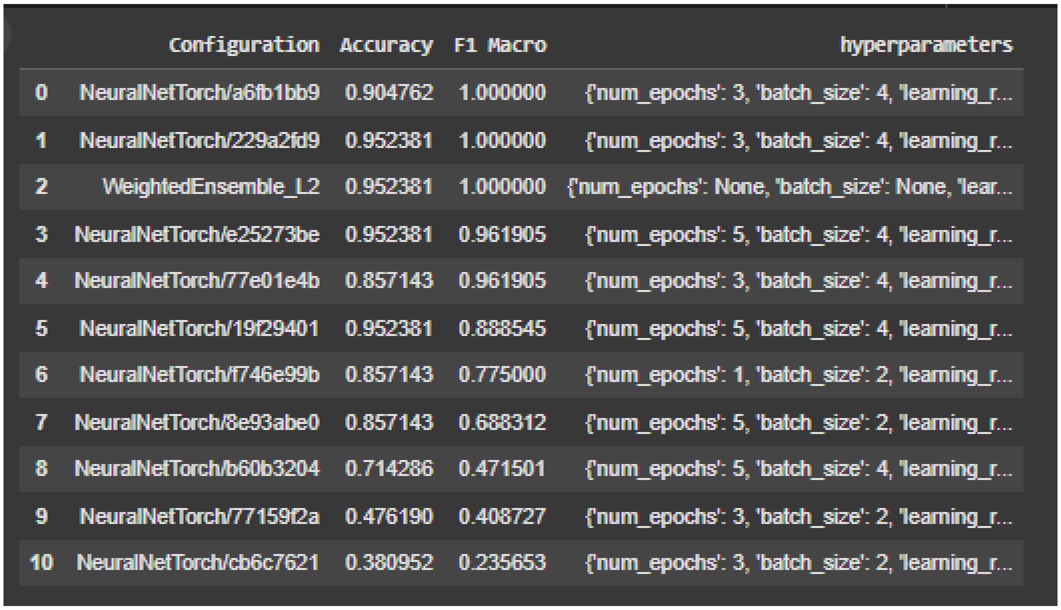


	Configuration	Accuracy	F1 Macro	hyperparameters
0	NeuralNetTorch/e3bff_00003	0.952381	0.888545	{'num_epochs': 3, 'batch_size': 4, 'learning_r...
1	WeightedEnsemble_L2	0.952381	0.888545	{'num_epochs': None, 'batch_size': None, 'lear...
2	NeuralNetTorch/e3bff_00009	0.857143	0.869091	{'num_epochs': 5, 'batch_size': 4, 'learning_r...
3	NeuralNetTorch/e3bff_00001	0.904762	0.775000	{'num_epochs': 1, 'batch_size': 4, 'learning_r...
4	NeuralNetTorch/e3bff_00000	0.857143	0.775000	{'num_epochs': 1, 'batch_size': 2, 'learning_r...
5	NeuralNetTorch/e3bff_00006	0.619048	0.520000	{'num_epochs': 5, 'batch_size': 4, 'learning_r...
6	NeuralNetTorch/e3bff_00002	0.619048	0.604762	{'num_epochs': 5, 'batch_size': 2, 'learning_r...
7	NeuralNetTorch/e3bff_00005	0.428571	0.343860	{'num_epochs': 1, 'batch_size': 2, 'learning_r...
8	NeuralNetTorch/e3bff_00007	0.523810	0.356261	{'num_epochs': 5, 'batch_size': 4, 'learning_r...
9	NeuralNetTorch/e3bff_00008	0.333333	0.209235	{'num_epochs': 1, 'batch_size': 2, 'learning_r...
10	NeuralNetTorch/e3bff_00004	0.380952	0.158025	{'num_epochs': 3, 'batch_size': 2, 'learning_r...

This is a data frame for random and grid searches.



```
F1-score: 1.0
Accuracy: 1.0
```



	Configuration	Accuracy	F1 Macro	hyperparameters
0	NeuralNetTorch/a6fb1bb9	0.904762	1.000000	{'num_epochs': 3, 'batch_size': 4, 'learning_r...
1	NeuralNetTorch/229a2fd9	0.952381	1.000000	{'num_epochs': 3, 'batch_size': 4, 'learning_r...
2	WeightedEnsemble_L2	0.952381	1.000000	{'num_epochs': None, 'batch_size': None, 'lear...
3	NeuralNetTorch/e25273be	0.952381	0.961905	{'num_epochs': 5, 'batch_size': 4, 'learning_r...
4	NeuralNetTorch/77e01e4b	0.857143	0.961905	{'num_epochs': 3, 'batch_size': 4, 'learning_r...
5	NeuralNetTorch/19f29401	0.952381	0.888545	{'num_epochs': 5, 'batch_size': 4, 'learning_r...
6	NeuralNetTorch/f746e99b	0.857143	0.775000	{'num_epochs': 1, 'batch_size': 2, 'learning_r...
7	NeuralNetTorch/8e93abe0	0.857143	0.688312	{'num_epochs': 5, 'batch_size': 2, 'learning_r...
8	NeuralNetTorch/b60b3204	0.714286	0.471501	{'num_epochs': 5, 'batch_size': 4, 'learning_r...
9	NeuralNetTorch/77159f2a	0.476190	0.408727	{'num_epochs': 3, 'batch_size': 2, 'learning_r...
10	NeuralNetTorch/cb6c7621	0.380952	0.235653	{'num_epochs': 3, 'batch_size': 2, 'learning_r...

We got 100 % accuracy for the Bayesian search, maybe because the dataset is so small or the model is overfitting.
We tried our best to get the table for the Hyperband too, However due to versions issues it was not possible and several errors were being displayed during the analysis training of the same.
As per the theory, the accuracies follow the order of
Hyperband/ Bayesian > Grid > Random
However, since our dataset is small, we are getting 100 % accuracy for Bayesin, which might be because the model is overfitting.

Our Table was as follows

Method	Accuaracy	F1 score
Grid	0.9	0.900928
Random	0.9	0.900928
Bayesian	1	1
Hyperband		

Also, looking at the accuracies table, our task is the same.

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.

We think for deep learning or large-scale ML tasks, automated search is the better approach because it optimizes performance without excessive trial and error. However, manual tuning might still be better for small models or when computational resources are limited.