General description of my subtask3 code

My code for subtask 3 is very similar to that of subtask 2.

My code for subtask 3 utilizes four functions which are described as follows:

1. create_model_subtask3 functions very similarly to create_model, but it takes in the kernel_initializer string and the dropout float and uses those to set the kernel_initailizer and the dropout rate, respectively.
2. evaluate_subtask3 functions very similarly to evaluate, but it takes in the optimizerSTR and uses it to set the optimizer of the model when evaluating it.
3. createModelsAndEvaluate_subtask3 takes in all the lists of advanced hyper parameters, loops through each to produce all combinations of advanced hyper parameters (48 combinations), produces a model from that combination, evaluates the accuracy of that model, and then adds those particular parameters and their associated accuracy, as a list, to a list of lists called “rows”. Once all combinations and their associated accuracies have been added to “rows”, “rows” is returned.
4. hyperparameter_ranking_subtask3 takes in the list of lists called “rows” produced by createModelsAndEvaluate, produces a list of column names, produces a pandas data frame from the rows and then sorts all the rows in descending order based on the “Accuracies” column. As the data frame is sorted the indexes of each row is reassigned to be the descending ordered position (row 0 has the highest accuracy, etc).

createModelsAndEvaluate_subtask3 and hyperparameter_ranking_subtask3 were separated so that if there are changes that need to be made to the sorting, they can be made without having to reproduce and reevaluate every model.

When createModelsAndEvaluate_subtask3 is called, the result is also immediately saved to a .csv so that the results can be loaded later, without having to reproduce and reevaluate every model.

What is the meaning of “dropout rate”? Explain in 2 sentences how dropout is done in the training process.

The dropout rate sets the frequency with which the dropout layer randomly sets input units to 0. The purpose of this is to help prevent overfitting by preventing the neural network from relying too heavily on any particular neuron.

Does an increased dropout rate have any impact on performance? Is it consistent?

Increasing the dropout rate seemed to have a deteriorating effect on accuracy. The worst 16 models all had the highest dropout rate. While the best 5 models all had dropout rates of 0. However this trend is not 100% consistent because models ranked 6th best onward up to 32nd best have a mixture of dropout rates of 0 and 0.2.

Explain the results you found with ‘zero kernel initializer’. Would you have gotten the same result if you used “-1” instead of “0” for the kernel initializer?

All node weights are initialized to a value of 0. These models performed rather poorly. The results likely would not have been the same had the weights been initialized to -1 instead.

Explain the different results obtained with different Optimizers. Is there any winner? And if there is a winner, does it outperform the other optimizers in all the cases?
If we exclude models having a dropout of 0.5 or a kernel initializer of zeros (both of which severely crippled the model's performance) and thereby only examine the best 24 models then we can compare only viable models. In this comparison Adam narrowly takes first place with an average ranking of 5.83 (indexing the best model at 0), followed by RMSprop with an average ranking of 6, with SGD taking third place with an average ranking of 13.66, and Adagrad coming in last with an average ranking of 20.5.
While Adam is the winner, it does not outperform the other optimizers in all cases, and this is reflected in how narrow Adam's lead is over RMSprop in the comparison of their average rankings.