Design choices

Overall principles for both datasets

- Robust scaler used to minimise effect of outliers, i.e. $X \leftarrow \frac{X - \tilde{X}}{IQR}$ where $\tilde{X} \equiv \text{median}$ and $IQR \equiv \text{interquartile range}$.
- K = 5 folds used for cross validation with shuffle=True. Accuracy/loss curves helped guide the design process (models show little variation, models converge nicely, etc…)
- Binary_crossentropy as loss function
- batch_size = 128 chosen to make model robust and smooth gradients
- Epochs chosen to point where val_accuracy maximised

Dataset1

Model:

- One Dense(32, softmax) hidden layer
- **Optimizer:** Adam with learning_rate = 0.002
- epochs = 250
- **Total trainable params:** 290

Decisions:

- Due to the simplicity of the dataset, models with 0 hidden layers were tried. These could not exceed 90% test accuracy.
- To improve learning beyond 91%, a large unit-hidden layer was chosen, with a slightly higher learning rate and a batch size to smooth out gradients

Dataset2

Why is dataset2 harder?

- Dataset2 has skewed attributes with many large outliers, in particular Ball Radius and Spring Constant.
- Dataset2 has multicollinearity, and in general the correlation matrix does not show strong correlations with the target variable for any of the attributes
- The dataset comes sorted by target values, thus requires shuffling

Model:

- Three Dense hidden layers, 16, 8, 16 with biases 0.7, 0.5, 0.3. All softmax activations
- **Optimizer:** Adam with default parameters
- epochs = 600
- **Total trainable params:** 506

Decisions:

- An initial attempt to remove the multicollinearity and skewness used MinMaxScaler, and then set Ball Radius and Spring Constant to 0. This proved ineffective, even when different bias / kernel initialisers were used.
- BatchNormalization layers added between dense layers to prevent ‘stacking’ of gradients, and also to decrease learning time.
- Bias added to the layers to improve learning