Note: Several of you only reported accuracy - again, especially in multi-class problems with imbalanced classes, I wanted to remind you to look at the confusion matrix. The full list of possible performance metrics can be found here: https://keras.io/api/metrics/.
There were only 2 data points that were of class "ISLAND" in the test set that were mis-classified, which wouldn't affect the accuracy, but could be problematic if it's an important category. A classic example would be if we're predicting whether a loan will be fully paid, partially paid, paid late, or default, while default is a rare event (often <1%), it is the most costly to the lender and the most important to get right.

Q&As

Q: Why can't we use a single logistic regression classifier for multi-class classification, e.g. if we had 10 classes, then if the output is between 0-0.1, it's class 1, etc.?

> **A:** Binary classification models like logistic regression and SVM do not support multi-class classification natively and require meta-strategies. Remember that logistic regression draws a linear decision boundary that most cleanly separates two classes. If we do what you suggest, we're drawing one line, e.g. between numbers 5 and 6 and assuming that the further away from the boundary, it is a different class, with rather arbitrary parallel decision boundaries (i.e. another line between 1 and 2, 2 and 3, etc.). Not only is this essentially creating multiple one vs. rest boundaries, but it's also poorly designed. You can't assume that the same decision boundary that separates numbers 5 and 6 would work to separate numbers 1 and 2.
>
> There are some good visualizations and associated math in the Princeton lecture on this topic:
> https://www.cs.princeton.edu/courses/archive/spring16/cos495/slides/ML_basics_lecture7_multiclass.pdf

Q: I encountered problems when I set the batch size to very large - why is that?

> **A:** You can't set your batch size to be larger than your data set. Example: if you have 1000 training examples, and your batch size is 500, then it will take 2 iterations to complete 1 epoch. Too large of a batch size will lead to poor generalization.
>
> The higher the batch size, the more memory space you'll need, so that could also cause an issue, but that's unlikely to have been a problem in this data set since it's pretty small.

Q: Is it true that batch sizes should be a power of 2?

> **A:** It's not a hard rule - your batch size can be as large as your data set and GPU permit. It has nothing to do with neural network's training or performance, but it's supposed to help with processing because cores (tensor cores, CUDA cores, whatever) are typically laid out and released in configurations that favour powers of 2. See one explanation here: https://datascience.stackexchange.com/questions/20179/what-is-the-advantage-of-keeping-batch-size-a-power-of-2.

Q: I was wondering if there's any advantage of using a single wide hidden layer rather than multiple narrow hidden layers (say 1 hidden layer with 100 neurons than 4 hidden layers with 25) and vice versa. The way I see it, a disadvantage of using only a single hidden layer is that we can only use a single activation function.

A: In general, deeper layers will be more flexible and therefore model complex relationships better. The more layers we add, the more powerful the model and the larger the tendency to either: be very difficult to train, overfit the training data completely, be difficult/time-consuming to train, and the greater the level of regularisation that is likely required to obtain a reasonable validation metric on unseen data. The particular data set we used was not challenging / big enough to show the strengths of DNNs and was prone to over-fitting, but in many real-life use cases for DNNs, you'll want more hidden layers.

Mathematical proof of it can be found here: https://arxiv.org/pdf/1602.04485.pdf. Also see p. 201 of Deep Learning textbook (Goodfellow et al.): "Empirically, greater depth does seem to result in better generalization for a wide variety of tasks. […] This suggests that using deep architectures does indeed express a useful prior over the space of functions the model learns." Highly recommend the textbook for an in-depth understanding of deep learning techniques.

While you can use some intuition and heuristics based on the size and complexity of the data, DNN parameters are best found through experimentation and trial-and-error.