Answers to Assessed Quiz 1

Question 1
When collecting data for machine learning, which of the following does *not* need to be considered?
Group of answer choices
Overfitting/underfitting (81%)
Range of the values (6%)
Outliers (9%)
Quantity of the data (4%)

- Overfitting and underfitting are properties of how well a model can fit the data and are primarily about the model and whether it is suitable choice for the data and task. All other aspects are important considerations for collecting data that will provide good training and avoid bias.

Question 2
What indicates underfitting in a model?
Group of answer choices
Low testing set error (0%)
High training set error (96%)
High testing set error (0%)
Low training set error (4%)

- A model that is underfitting cannot do a good job on the training set, and hence has high errors there. It may also have high testing set errors, but that does not define whether it is underfitting or just not generalising, which could be overfitting. Only the training set error can be used to define underfitting.

Question 3
In a typical machine learning workflow, how do we select the best model out of several alternatives?
Group of answer choices
It has the smallest gap between the mean errors in the training and validation sets. (22%)
It has the smallest mean error in the validation set. (70%)
It has the smallest variance in the validation set. (6%)
It has the smallest mean error in the training set. (1%)

- We always select the best model on the basis of the validation set performance, and that is defined by the smallest mean error (or, more exactly, the smallest loss function value). It is also nice to have a small gap between the training and validation set errors (or loss function values) as that indicates that it is generalising well, but if another method has a better validation set error then we would still prefer that, even with some degree of overfitting, as it is still delivering better performance on the unseen data (the validation set).

Question 4
Which of the following steps is *not* a necessary step in best practice for applying machine learning?
Group of answer choices

Choosing some candidate models (0%)
Artificially generating data for the model(s) (85%)
Data exploration (9%)
Problem formulation (6%)

- When following best practice it is necessary to formulate the problem clearly (which will help with choice of method and loss function), and to explore the data (to remove problems, but also to get a sense of what constitutes high and low values for our performance measures), and to choose some candidate models (for obvious reasons). Sometimes we may also generate artificial data but for some problems it isn't really possible (e.g. determining whether a person has an unusual disease or not, as knowing what is and is not an acceptable value for a "healthy" person is very difficult). So the artificial data step is not necessary, but is used sometimes.

Question 5

If the training error is close to zero when performing a regression. Which of the following statements is true?
Group of answer choices
There is not enough information to predict anything about the test error. (75%)
The model is overfitting and the test error will be high. (15%)
The model is generalising well and will lead to a low test error. (9%)
The test error will also be close to zero. (1%)

- When the training error is near zero then it is possible that the validation and test errors could be low or high, depending on if the model is generalising really well or overfitting, respectively. Either situation is possible and there is not enough information given here to know anything about the validation or test set performance. Also note that "close to zero" is a subjective statement and may, or may not, indicate that the performance is good or that the error is "low" with respect to the requirements of this task. So don't assume too much from such words unless it is clear that the values are being assessed with respect to the current dataset and task requirements.

Question 6

When running 5-fold cross-validation it is important that:
Group of answer choices
The same 5-fold cross-validation is used for every method being compared (60%)
Any MinMax or StandardScaler scaling is done before the cross-validation (18%)
The data is initially split as 80%/20% for training and test sets (12%)
The standard deviation of the error across folds is not bigger than the mean error (10%)

- There is no requirement or assessment of the standard deviation of the errors – it might have some interpretive value, but is would not normally be compared to anything like the mean value.
- The initial split is often 80%/20% but that it not a requirement, and other splits can be perfectly valid.
- Running pre-processing (such as scaling) before the cross-validation is *wrong* as this would necessarily need to fit to the set of data (to get min/max or mean/stddev) *before* the splits into training and validation *within* the cross-validation method. Hence this would, for each fold, be using some information for the pre-processing that has come from the validation set for that fold, and that would then be a source of

bias. What needs to be done for K-fold cross-validation (or anything that subsequently creates training/validation splits) is to ensure that the pre-processing *is put into a pipeline* (if it requires fitting – and some pre-processing options do not, as we will see later for image pre-processing in deep learning) so that the pre-processing elements only ever see training data when fitting their internal parameters.

- Hence the only correct answer here is that the same folds are used for every method that is being compared. For instance, if you were comparing two methods (a DecisionTree and a KNN regression) each with a range of hyperparameter settings, then it is *essential* (in order to avoid bias) that the *same folds* are used for every evaluation – that means every hyperparameter setting and each different method needs to use the same 5 sets of training and validation data, as otherwise you are not fairly comparing methods/hyperparameter-settings.

Question 7

Which of the following is *not* a mistake to avoid?
Group of answer choices
Having a very small test set in order to provide more training data
Tuning hyperparameters to minimise the training error (31%)
Doing a model comparison between 2 models, selecting the best model, and then later on deciding to add another type of model but still using the same validation and test sets (48%)
Fitting and applying the StandardScaler pre-processing to each dataset separately at the start of the model comparison, after doing imputation (13%)
Increasing regularisation in order to fix underfitting (7%)

- In the standard workflow we tune hyperparameters by minimising the validation error, not the training error, so it would be a mistake to minimise the training error for hyperparameter tuning.
- Scaling or any pre-processing that involves fitting internal parameters (min/max, mean/stddev, etc.) must *not* be done on anything except the training set. Hence it is a mistake to fit it to other datasets, or to fit and apply it to a dataset that later on will be split into training and validation set (such as K-Fold cross-validation). Instead, these pre-processing steps should be put into a pipeline so that they are only ever fit to training data, just like the model.
- Increasing regularisation will normally make an underfitting model even worse, as if you notice underfitting in a model that contains regularisation you should *decrease* the regularisation, to see if the extra flexibility that this allows will help the model fit better.
- All models that are being compared need to use the same validation and training datasets, in order to be fair. If a model is added after some initial round of comparisons then it should be treated in the same way as if it were included in the initial set of models. This means that it should use the same validation and training sets, then the best model is evaluated on the test set to measure the unbiased generalisation error. It is still unbiased as long as the test set performance is *never* used to choose between models. So if model B was the best model in the first round and then other models were added and it turned out that model E was now the best model (as defined on the basis of the best performance on the *validation* set) then model E can be evaluated on the test set, even if model B had been evaluated before. *However*, if model E is worse on the test set than model B then you *cannot* turn around and choose model B on this basis – you need to act as if the test set performance for model B had never been measured. If model E is better than model B on the validation set then this means that model E is the best choice in our

framework. Its performance on the test set then defines its true generalisation error (or, more correctly, an unbiased *estimate* of its generalisation error). In short, always use the validation performance to choose the best model, and then use the test set to get an unbiased generalisation performance estimate, but never use the test set to compare anything.

Question 8

Your colleague runs a GridSearchCV and finds that the best performing method has a validation error of 3.4 and a training error of 3.1, while the second best method has a validation error of 3.5 and a training error of 0.2. They say that it would be better to choose the second best method because the training error is so much better. Which of these would be the best response?
Group of answer choices
Say that there isn't enough information and so the two methods should be evaluated on the test set to be sure (16%)
Disagree, because it is only the validation error that counts (33%)
Agree that this would be a good decision, because the validation error might be a noisier estimate compared to the training error, which uses more data and so will be more reliable (7%)
Disagree, because the second best method is overfitting (43%)

- It is important to remember two things when looking at performance/error numbers: (i) these are noisy estimates, not perfectly accurate values; and (ii) whether numbers are high/low depends on the nature of the task, the available data, and the requirements of the user/client. So it is very hard to know whether these numbers represent high or low values, and hence whether the difference is important. However, regardless of this we only ever use the *validation* set performance to compare models/hyperparameter-choices. It does not matter if there is a big difference in training and validation performance when it comes to comparing them – we always take the one with the lowest best performance. So we don't actually care if a method might be overfitting or not in terms of finding the best in our model selection. Furthermore, we don't know if the difference in these numbers is important or not, so if they are both well under the desired performance criteria then it would not be overfitting anyway. We might be able to improve things by noting this overfitting, but for any particular choice we will always go with the model with the best validation performance. In addition, we never compare methods on the test set, and despite the fact that the methods might be noisy, we still accept the validation results and would not choose another method where the performance was worse on the validation data.

Question 9

What would be an appropriate data scaling for the following feature:

	0
count	100.000000
mean	2.435695
std	6.667815
min	0.295457
25%	1.372246
50%	1.809204
75%	2.191614
max	68.200000

Group of answer choices
MinMax (13%)
SimpleImputer (4%)
None (24%)
StandardScaler (58%)

- The key thing to notice here is that the maximum value (68.2) is a long way away from the rest of the distribution (75th percentile is 2.19, and min is 0.295). So using MinMax would be bad, as it would squeeze most values into a narrow range near zero. We want the set of values to generally span 0 to 1, or -1 to +1, and so doing no scaling would leave over 50% of the data above 1.8, which is not well matched to our desired ranges. Imputation is about filling in missing values, not fixing the range by scaling, and so the correct answer here is to use the StandardScaler (even though most data ends up between -0.3 and 0, this is still better than more than 75% of the data being greater than 1.0). A much better solution in practice would be to use the RobustScaler, as that would be insensitive to this outlier, whereas the StandardScaler would still end up compressing the values into a somewhat narrow range near zero (but nowhere near as narrow as MinMax would do). Furthermore, in practice we'd probably investigate the source of this outlier value and decide whether it represented a true but unusual case (and keep it) or an erroneous measurement/record (and delete it).
- Since I was informed that some people had interpreted "None" as "None of the other options" rather than "No scaling" (where the latter was intended), I have treated "None" as a correct answer as well.

Question 10

Which of the following statements is true?
Group of answer choices
Grid Search is the best way to do model comparisons because it tests all possible combinations of hyperparameter settings (13%)
If the error on the test set is less than that on the validation set then the test set is poorly chosen and another random split should be used and the training/selection process repeated. (4%)
The validation error is always higher than the training error (6%)
Comparing different types of models (e.g. ridge regression and decision tree) as well as different hyperparameter settings for each requires two separate validation sets, one for each model type, and then a final test set (39%)
A k-fold cross-validation changes the definition of the training, validation and test sets for each fold (7%)
The validation error for the selected model is always biased, which is why a test set error is also needed (30%)

- Even though the validation error for each model gives us an unbiased estimate, the probability distribution for the method than maximises performance on the validation set will be skewed and hence biased. For example, imagine that everyone in a classroom is asked to toss a coin and keep tossing until they get heads, and then record how many tosses it took until they got heads. Then, take the student that got the highest number and ask them to do it again. Would you expect the second number to be higher or lower than the first? Almost always it will be lower, because you've selected a particularly unusual outcome and then it is difficult to repeat and so the second measurement is usually lower, or put another way, the first measurement is biased, as is fairly obvious in this coin tossing experiment, but holds true for us or any selection process. Thus the test set measurement is the only unbiased measurement that we have of the *selected* model.

- Two entirely different models (e.g. ridge regression and decision tree) or two instances of the same model with different hyperparameter settings, even if the difference is tiny (e.g. regularisation of 0.1 or 0.08 for ridge regression) should be treated in the same way. Any change in model needs to be evaluated in the same way, with the same training and validation datasets in order to have a fair comparison. So we definitely do not use different validation sets for different models, as then they are not strictly comparable.
- Grid Search will compare all possible combinations of *your chosen* set of hyperparameter values. However, this set is rarely complete, as any hyperparameter that takes a real value cannot be completely tested (e.g. regularisation can be anything from 0 to infinity, and even if you cap the maximum value, you still could have 0.9 and 0.99 and 0.999 and …) so we very rarely can do *all possible* combinations. Even if there are limited values and several hyperparameters, this can become infeasible in practice (e.g. 6 hyper-parameters each with 3 different settings would still be 3^6 = 729 combinations, and that might take too long to run).
- K-fold CV changes the training and validation sets for each fold but the test set must remain separate and not part of the CV. Otherwise the test set data would be part of many of the training sets during the K-fold loop, and hence the ML methods would have had access to the test set data before the final test, meaning that you would get a *biased* estimate for the final performance on the test set. Only keeping the test set fixed and separate from the K-fold CV ensures that you get an unbiased estimate at the end.
- Validation error is often higher than the training error, but they are both noisy estimates (based on a finite set of samples) and hence you can get lucky sets where the validation error is less than the training error. There is no guarantee that one is higher than the other, but on *average* we expect this to be true.
- Both the validation error and test error are noisy estimates, so it is very possible that the test error could be less than the validation error, even though the validation error for the selected model is biased. However, that only gives us an expectation that this will be true *on average* across different datasets and tasks, not that any particular task and dataset will guarantee anything. So if the test error is less than the validation error then that’s fine. We must not decide to repeat the process on the basis of some criterion not being met by the test set performance – we can *never* make a decision based on the test set performance without then causing bias in our process. So we just accept the performance measurement on the test set, regardless of what it looks like. The only exception to this would be if something very odd or bad was found in the test set, in which case excluding this (or using imputation) and then measuring again is fine, as long as the identification of the problem value(s) was based on something other than the fact that they gave a bad error value – there must be other reasons employed or else you get into the situation of introducing bias again.