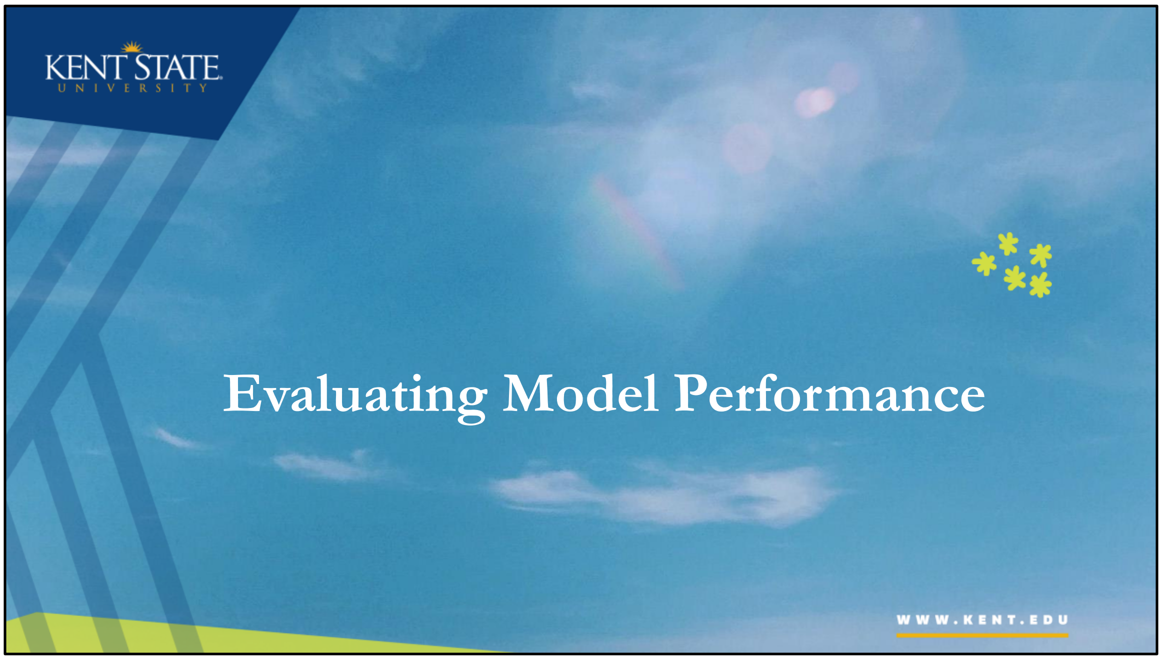
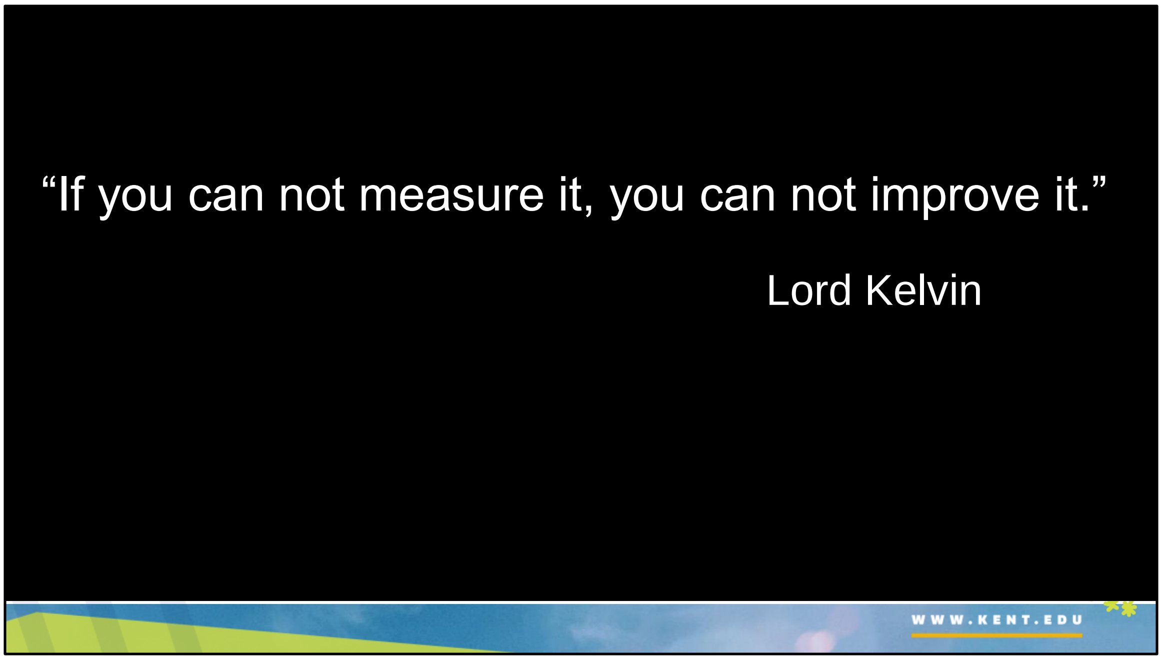


Evaluating Model Performance

KENT STATE UNIVERSITY

WWW.KENT.EDU

In this lecture, we dwell deeper into evaluating model performance. Specifically the division of the sample into training and test sets.

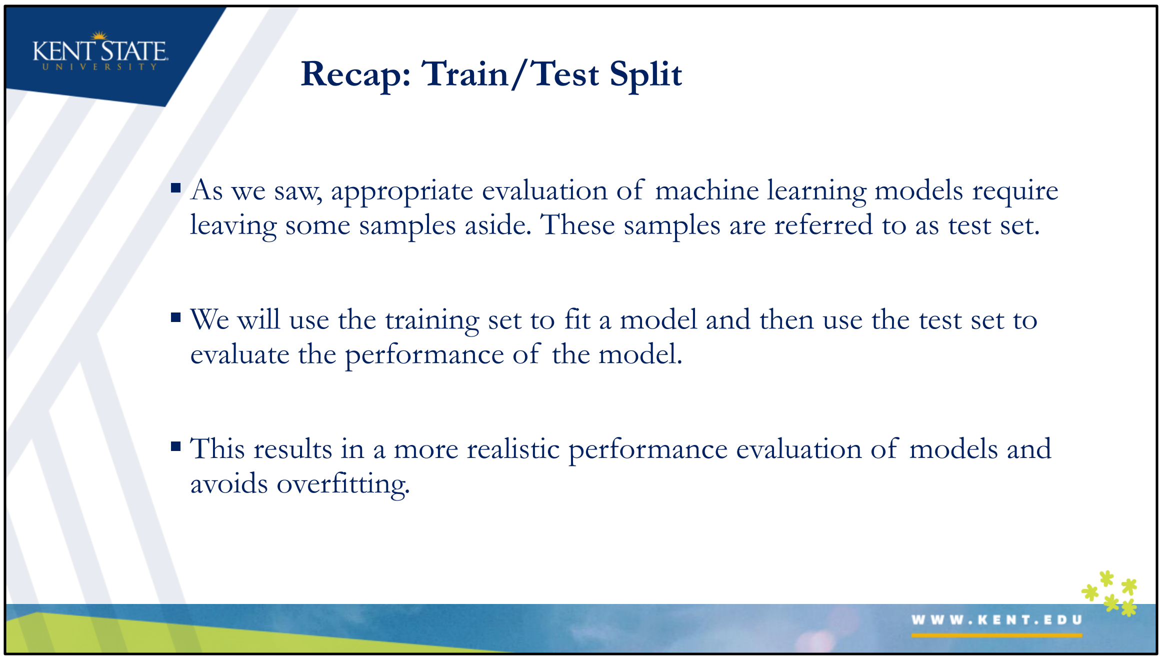


“If you can not measure it, you can not improve it.”

Lord Kelvin

WWW.KENT.EDU

no audio

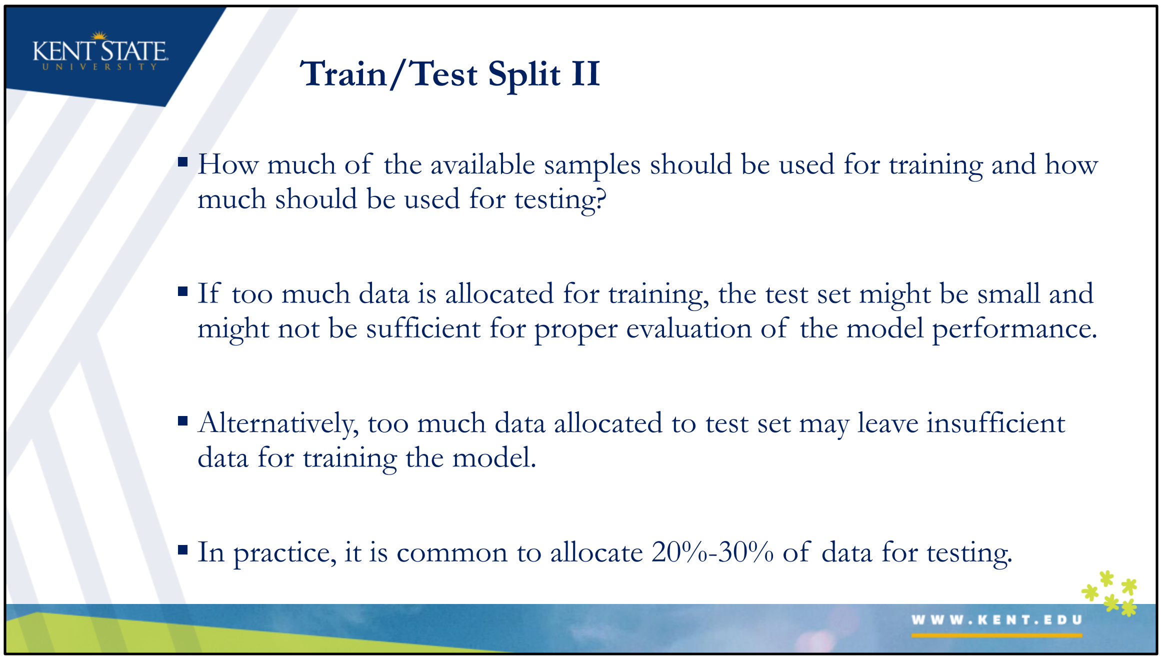


Recap: Train/Test Split

- As we saw, appropriate evaluation of machine learning models require leaving some samples aside. These samples are referred to as test set.
- We will use the training set to fit a model and then use the test set to evaluate the performance of the model.
- This results in a more realistic performance evaluation of models and avoids overfitting.

WWW.KENT.EDU

One of the primary reasons for ML models is prediction. As such, it is important that we choose models that have good prediction performance. As we saw earlier, this can be achieved by the use of test sets. We now examine in greater detail on how to accomplish this.

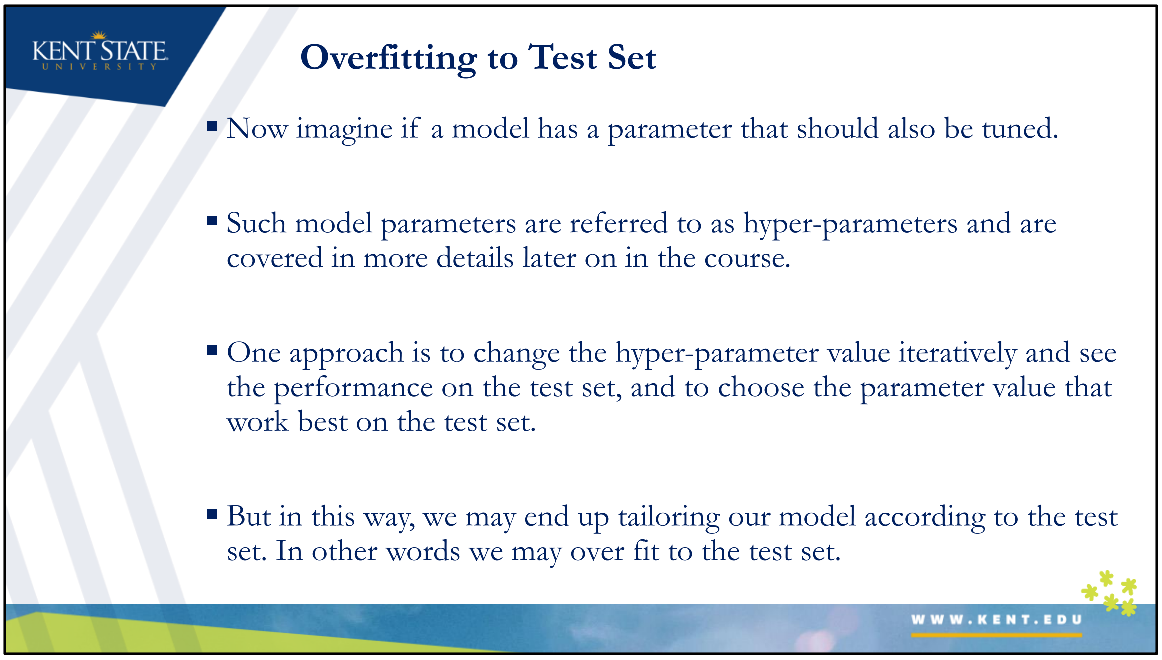

Train/Test Split II

- How much of the available samples should be used for training and how much should be used for testing?
- If too much data is allocated for training, the test set might be small and might not be sufficient for proper evaluation of the model performance.
- Alternatively, too much data allocated to test set may leave insufficient data for training the model.
- In practice, it is common to allocate 20%-30% of data for testing.

WWW.KENT.EDU

Assume that our sample size is large enough so that we can afford to partition it into a training and test set. If that is so, then generally between 20% to 30% of the data is used for testing, while the rest is for training. If the sample size is large enough, then this will prove adequate to both build a good model to determine h the relationship between X and Y, and to be able to predict for unseen samples (test set).

Question: What if the sample size is not large enough?



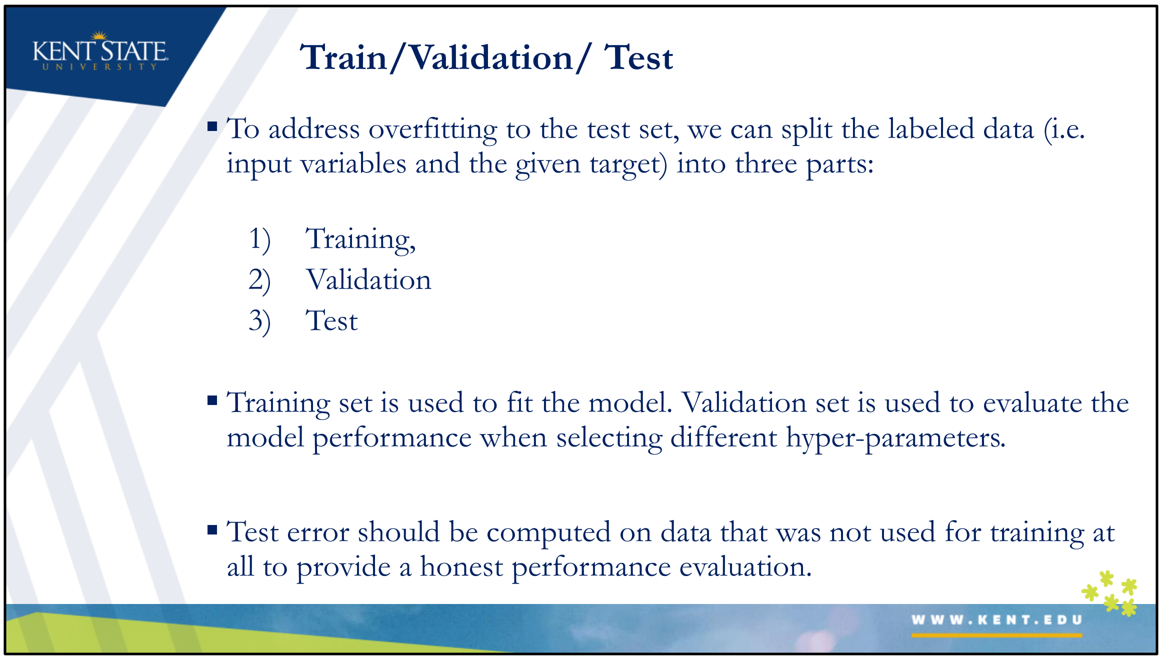
Overfitting to Test Set

- Now imagine if a model has a parameter that should also be tuned.
- Such model parameters are referred to as hyper-parameters and are covered in more details later on in the course.
- One approach is to change the hyper-parameter value iteratively and see the performance on the test set, and to choose the parameter value that work best on the test set.
- But in this way, we may end up tailoring our model according to the test set. In other words we may over fit to the test set.

WWW.KENT.EDU

Remember the purpose in using a test set. That is to see how well our model does on unseen data. One danger in what we have proposed is the fact that by repeatedly checking performance on the test set, we have inadvertently optimized the selection procedure for the test set. This is exactly what we wanted to avoid. Our objective is to see how the model does on unseen data. In practical terms, that means we do not have access to the test set, so there will be no way to optimize our model for the test set. And, this is something that we should not do in any case.

How do we solve this dilemma? We need to determine our model based on its prediction on unseen data, but at the same time, we can’t use data from this unseen sample to optimize our selection procedure.



Train/Validation/ Test

- To address overfitting to the test set, we can split the labeled data (i.e. input variables and the given target) into three parts:

  1) Training,
  2) Validation
  3) Test

- Training set is used to fit the model. Validation set is used to evaluate the model performance when selecting different hyper-parameters.

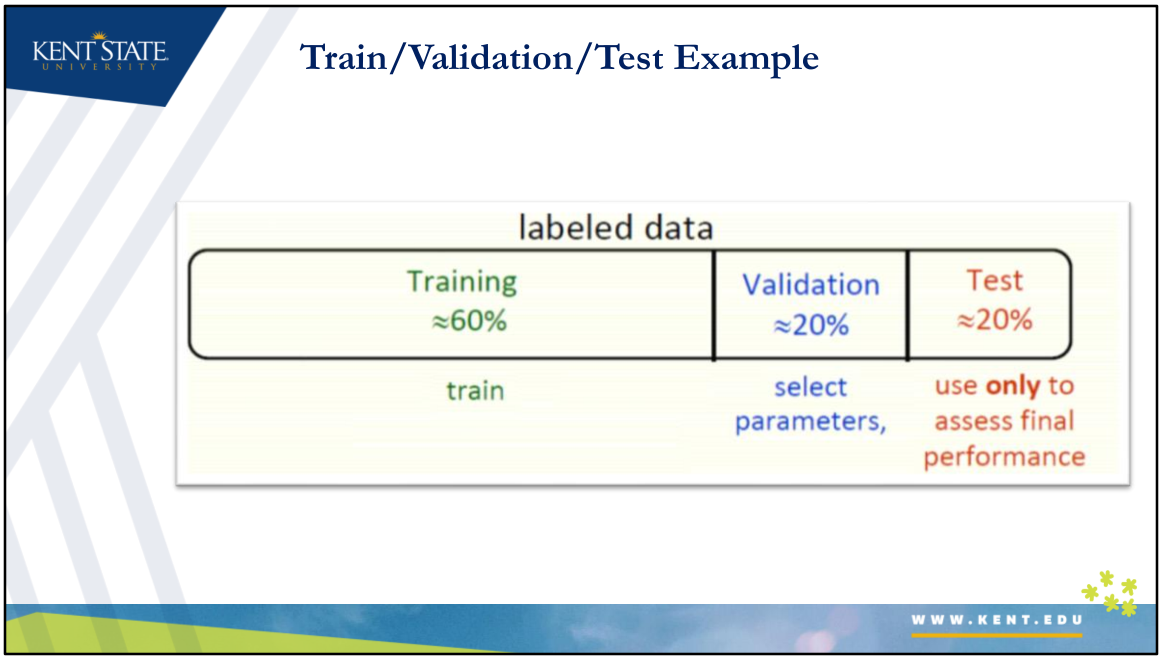
- Test error should be computed on data that was not used for training at all to provide a honest performance evaluation.

WWW.KENT.EDU

The way to address the question on the previous page is to divide the data into three parts: Training, Validation, and Test.

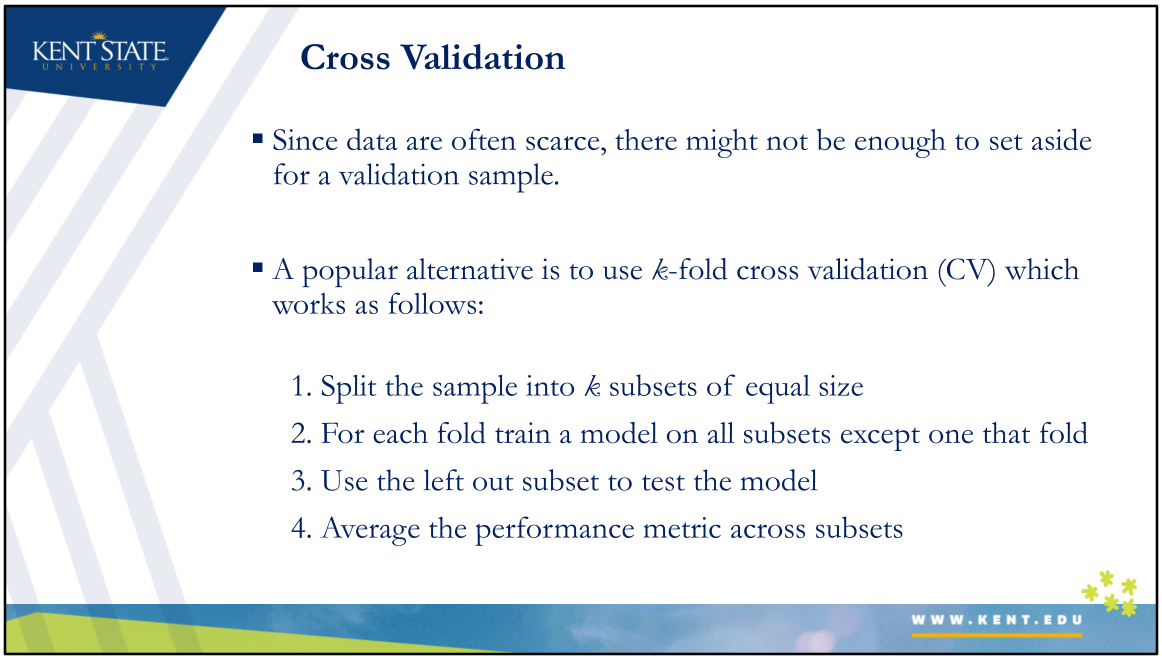
Training serves the same purpose as before. The validation set here performs the role for model selection. This data is not used in training, but serves as a reference point to see how models do on unseen data. As we use the validation set for model selection, it really is not truly unseen. That purpose is now served by our holdout test set, which we use in the end to gauze model performance.

It is common practice that once we have determined the model based on Training and Validation sets, we combine the two sets to get the parameter values of the model. Remember that we don't want to discard any useful data.



Train/Validation/Test Example

Given a large enough sample, it is common practice to split the data as shown in the figure above. What if the sample is not large enough? In this case, we use an approach called k-fold cross validation.

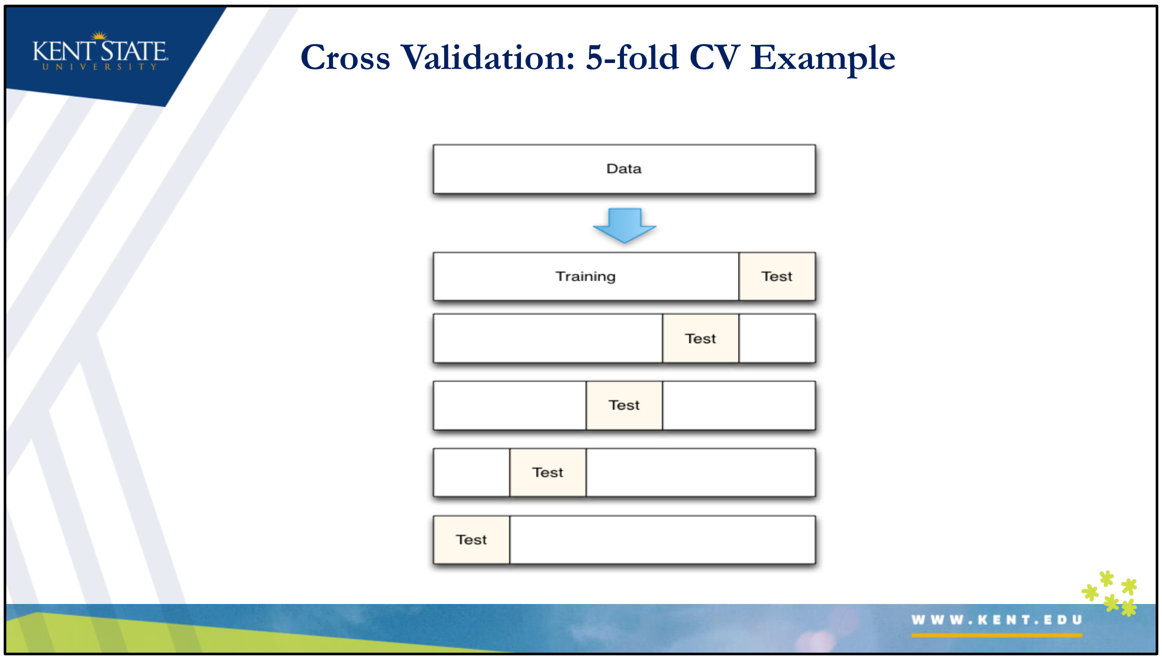


Cross Validation

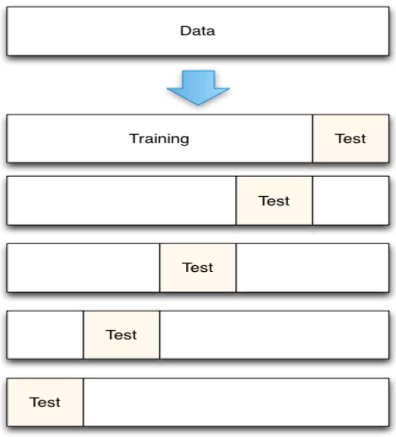
- Since data are often scarce, there might not be enough to set aside for a validation sample.
- A popular alternative is to use *k*-fold cross validation (CV) which works as follows:

  1. Split the sample into *k* subsets of equal size
  2. For each fold train a model on all subsets except one that fold
  3. Use the left out subset to test the model
  4. Average the performance metric across subsets

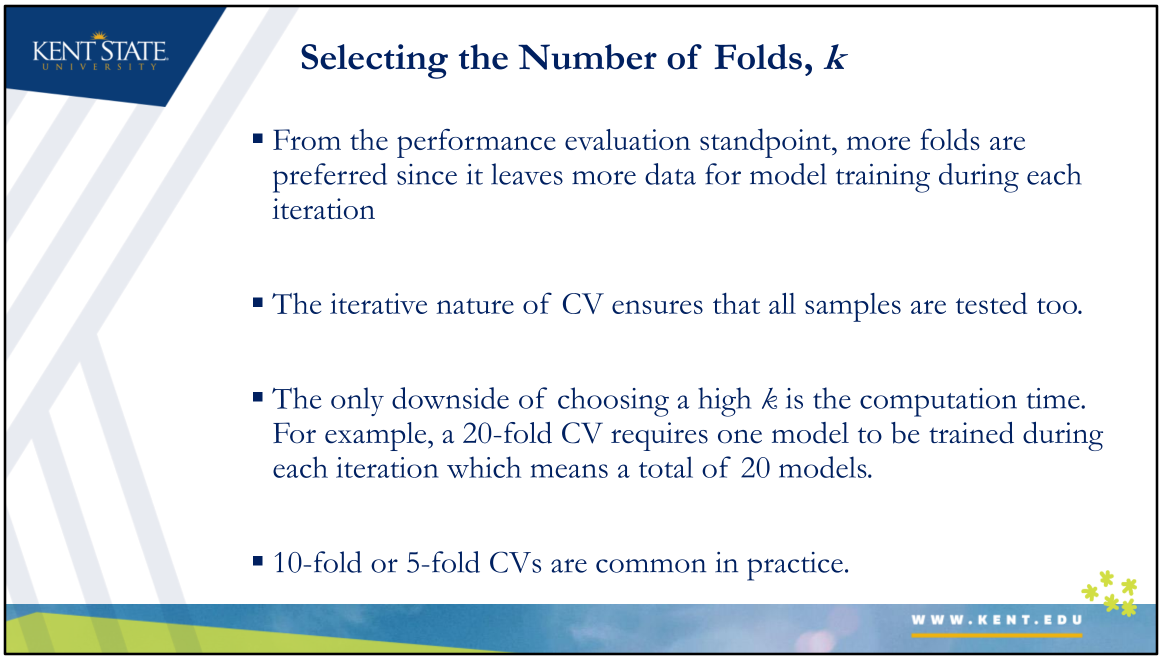
The idea is simple. We divide the original sample into k equal, but non-overlapping, sets. We then use k-1 sets for training, with the kth set as the validation set. For example, if k = 5, then we have five different ways of choose 4 sets. Each time, the left out set becomes the validation set. Here is a graphical illustration.



Cross Validation: 5-fold CV Example



Notice that here the holdout set indicates it is the test set. The holdout set can be used as validation or test, depending on the purpose. Remember that both validation and test sets are unseen data. If we do not use either of them to retune the parameters of the model, then either serves as a true holdout set.

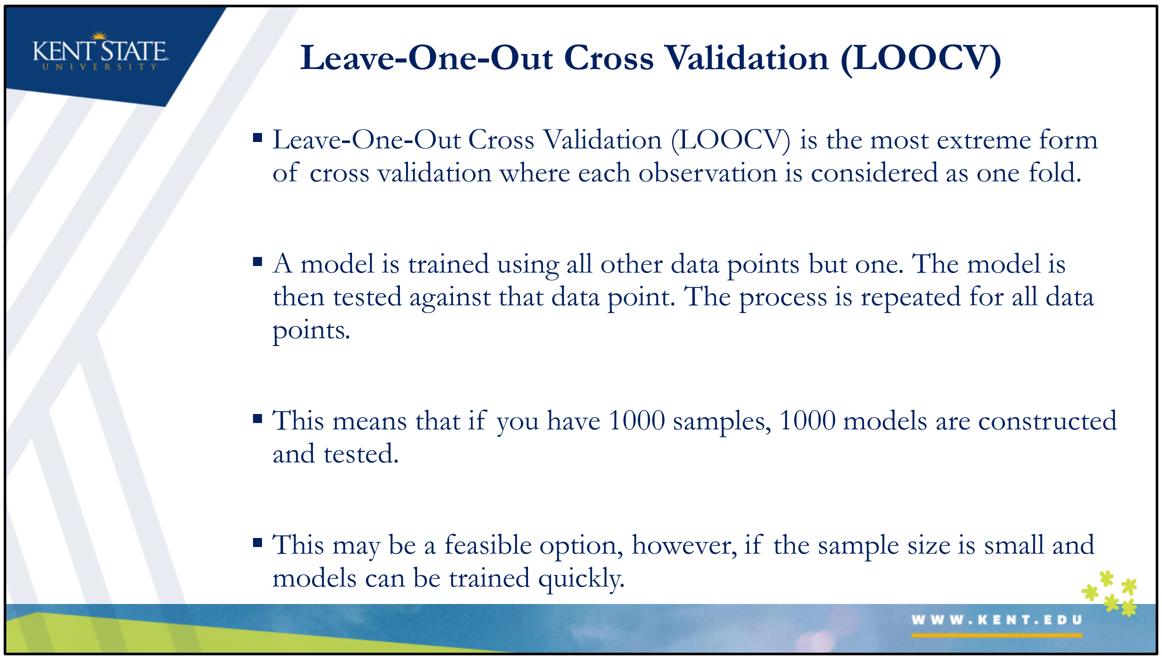


Selecting the Number of Folds, *k*

- From the performance evaluation standpoint, more folds are preferred since it leaves more data for model training during each iteration
- The iterative nature of CV ensures that all samples are tested too.
- The only downside of choosing a high *k* is the computation time. For example, a 20-fold CV requires one model to be trained during each iteration which means a total of 20 models.
- 10-fold or 5-fold CVs are common in practice.

WWW.KENT.EDU

For most practical applications, k=5 works well. It really depends on the size and characteristics of the sample.

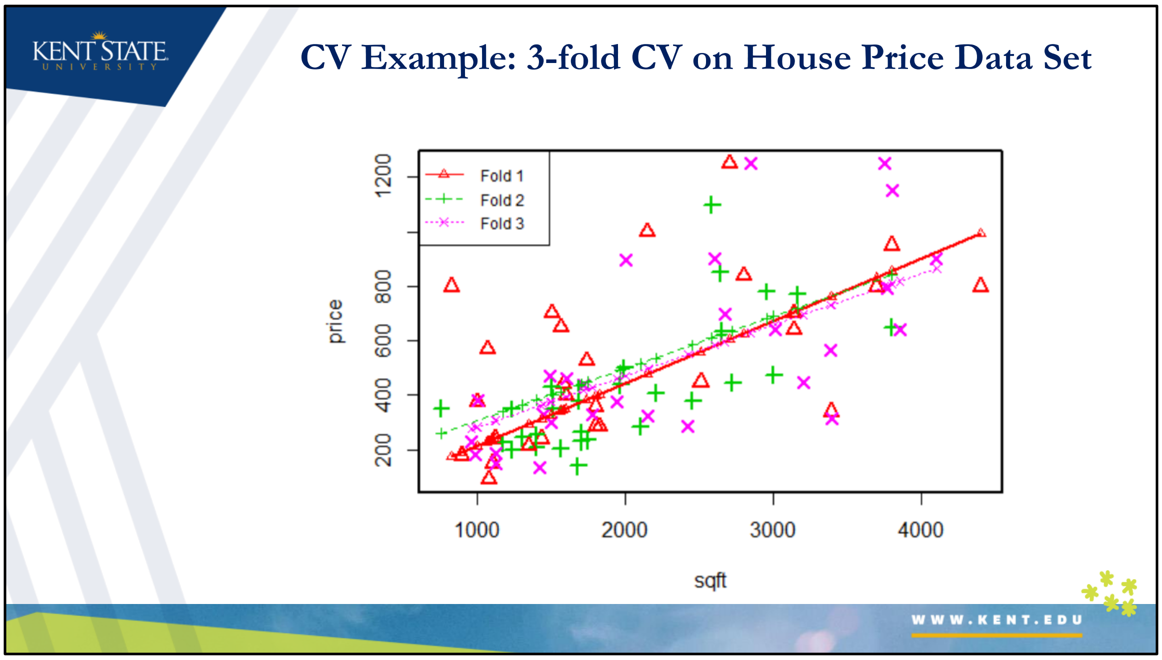


Leave-One-Out Cross Validation (LOOCV)

- Leave-One-Out Cross Validation (LOOCV) is the most extreme form of cross validation where each observation is considered as one fold.
- A model is trained using all other data points but one. The model is then tested against that data point. The process is repeated for all data points.
- This means that if you have 1000 samples, 1000 models are constructed and tested.
- This may be a feasible option, however, if the sample size is small and models can be trained quickly.

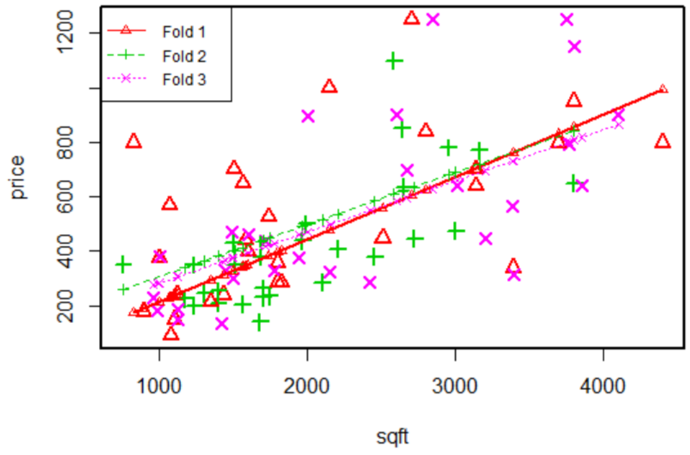
WWW.KENT.EDU

This is an extreme case of the k-fold cross-validation approach. The idea is to construct a model with n-1 data points, where n is the total number of points, and see how it predicts the nth point. This approach is frequently used in regression analysis to determine the effect of a single point on the Model. It is called PRESS. For example, let us assume that one sample point is highly influential. If hat is so, then excluding that point from model construction is likely to give very different results than if you were to include it. This allows us to determine such points in the sample. In any case, other than specific instances like regression, we are unlikely to use this approach much.



KENT STATE UNIVERSITY

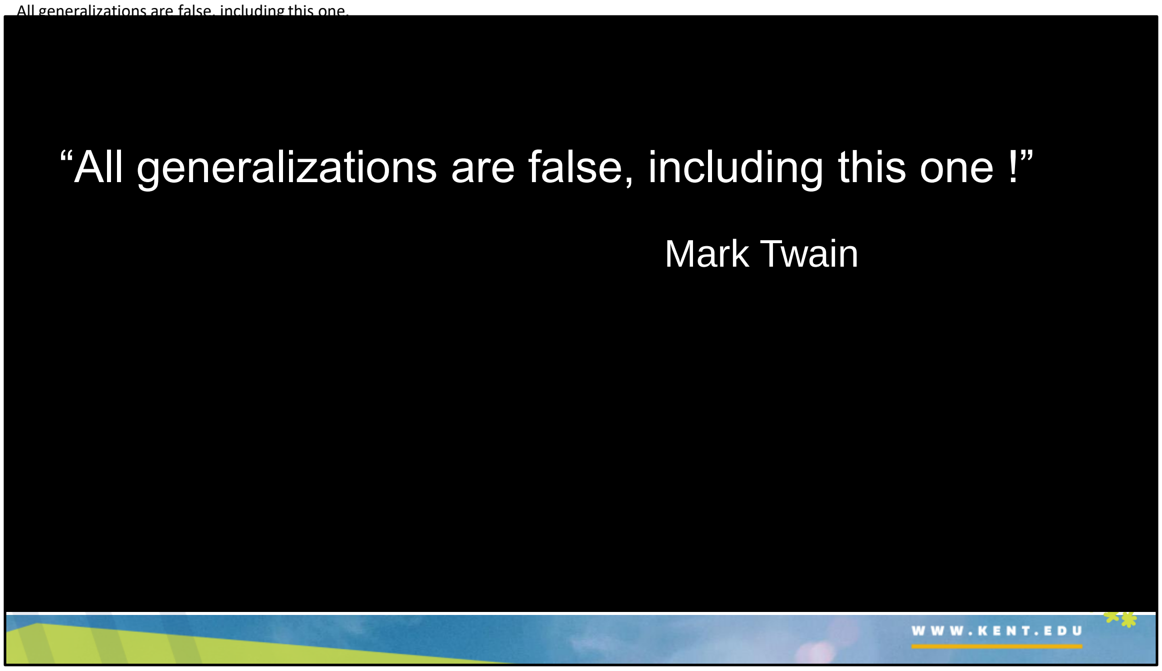
CV Example: 3-fold CV on House Price Data Set



WWW.KENT.EDU

Question: If we use a k-fold cross validation approach, how do we determine our final model?

All generalizations are false, including this one.



"All generalizations are false, including this one !"

Mark Twain

WWW.KENT.EDU

It is important to remember that most generalizations are limited. If you start using ML models for extrapolation, i.e., on data that is outside the range of what you have observed, it is unlikely to do well. For example, a ML model trained on data from children is unlikely to predict well for adults.

This concludes our lecture on cross validation.