3.1 Learning Response: Modeling Data

Jason “Scott” Person

20250414

1. Explain why and how we split the data into train, validation, and test sets.
   1. What is the purpose of each?
   2. What proportions are typically used for each?
   3. Recommended: Include relevant screenshots of Jedamski’s visuals.

Figure 1 below is from Jedamski’s presentation. It illustrated dividing the full data set into three parts:

* Training – used to create models
* Validation – once one or more models are created, this set is used to evaluate and tune the models and to aid in selection of the best model
* Testing – used to perform final evaluation of the model to determine that performance meets expectations.

A blue and black rectangle with white text

Description automatically generated

Figure 1: Dividing Data Set

The full dataset is typically divided into 60/20/20 proportions respectively. There is complete coverage of the full dataset in the subsets and no duplication. In other words, this his is selection without replacement.

Figure 2 below is also taken from Jedamski’s presentation. Here you can see the blue circles representing the usage of each of the data subsets in the model training and evaluation process. Note the tree evaluations/validation/testing stages and the iteration that is possible with each one.

A diagram of data processing

Description automatically generated

Figure 2: Train, Evaluate, and Test Process

1. Look up the homepage and Wikipedia page for the scikit-learn Python library. Then in 50-100 words, define it and describe its major features. Finally, provide a link to both its homepage and its Wikipedia page.

scikit-learn is one the most popular machine learning libraires for Python. It provides a free, open-source library for implementing common machine learning tasks. Features include utility methods commonly used by data-scientists, many data-science algorithms, a consistent API, and declarative syntax for implementing data and machine-learning pipelines (*scikit-learn*, 2025). Algorithms are available for classification, regression, and clustering as well as dimensionality reduction, model selection, and data preprocessing (scikit-learn developers, 2025).

URLs for the Wikipedia page and homepage for scikit-learn are in the References section below.

1. Explain cross-validation, k-fold cross-validation, and their purpose and benefits.

Cross-validation uses splitting and sampling to assess the performance of various models. The various types of cross-validation are primarily differentiated by the process for splitting the data. The most common method k-fold cross-validation involves splitting the training set into k subsets (without replacement). One of the subsets is set aside for evaluation and the other k-1 subsets are used to train the model. The model is then tested using the set-aside subset. This is repeated k times with each subset being used as the test set in one iteration. Figure 3 below shows the k-fold cross validation. Here 10,000 records were divided into five subsets with the fifth being used as the test set. In subsequent iterations, the red box indicating the test set would move to each of the other subsets in turn. The accuracy of the model is recorded for each iteration and then aggregated to calculate the mean accuracy of the general model type. This provides an overall expected accuracy for that model type, which is used to find the best model type and hyperparameters. The model is then retrained using the complete training set and the optimal hyperparameters identified during cross-validation.

Cross-validation is a useful process for identifying the best model type to use, determining the expected accuracy of the final model, and ensuring the optimal selection of hyperparameters to maximize accuracy without overfitting.

A diagram of a test set

Description automatically generated

Figure 3: k-fold cross validation

1. Define and explain the three evaluation metrics typically used in machine learning: Accuracy, Precision, and Recall. Include the formula for each.

Accuracy, precision, and recall are measures for evaluating classification models. They are defined as follows:

Accuracy – the percent classified or predicted correctly.

(Bruce, Bruce, & Gedeck, 2020, p. 220)

Precision – the percent of true positives of those identified as positive

(Bruce, Bruce, & Gedeck, 2020, p. 223)

Recall – the proportion of true positives correctly identified by the model

(Bruce, Bruce, & Gedeck, 2020, p. 223)

1. Use an outline to name and describe the steps and their purposes in the *process* Jedamski outlines. Then *be sure* to add a screenshot of his visual for this process.

Figure 4 below illustrates the modeling process. Here are the high-level steps:

1. Perform exploratory data analysis on the full dataset to understand, clean, and optimize the dataset for modeling.
2. Divide the dataset into three subsets
   1. Training (60%) – used to train the model
   2. Validation (20%) – used for initial validation and to aid in tuning
   3. Testing (20%) – used for final testing of the model
3. Divide the training set into five subsets.
4. Create models
   1. Run Fivefold cross-validation to get the best model types
   2. Refit the best models on full training set
   3. Evaluate the models on the validation set and select the best model
   4. Evaluate the best model on the test data set

A diagram of data processing

Description automatically generated

Figure 4: Model Training Process

1. Describe the central ideas of the bias/variance tradeoff.

The bias/variance tradeoff is largely a function of model complexity. Generally low model complexity leads to high bias error and low variance error and high model complexity reverses this. The goal is to minimize the aggregate error (the sum of bias and variance) such that we have a model that is as complex as it needs to be, but no more complex.

See Figure 5 below. High bias results from models that are two simple to capture the patterns in the data. An example would be using a linear model on data that is exponential. High variance results from models that are overly complex. These models could be going so far as to “memorize” the observations. Such a model would perform very well on training data but then give poor results on the training set and real-world data. An example of this would be a decision tree where the number of levels approaches the number of observations.

A diagram of a model complex

Description automatically generated

Figure 5: Bias/Variance Tradeoff

1. Explain the meaning of *under*fitting in data modeling and the problems this creates.

Underfitting is when the model is overly simplified and fails to capture the complexity of the data. Underfitting is characterized as having high bias and low variance. See the left-hand side of Figure 5 above. The model would exhibit poor results in both training and evaluation. At the extreme, a model of y = x would provide extremely high bias with zero variance and would have very little predictive value.

1. Explain the meaning of *over*fitting in data modeling and the problems it creates.

Overfitting, the opposite of underfitting, occurs when the model is overly complex. Such a model would be on the right-hand side of the Figure 5 graph. Note the low bias, but high variance. An overfit model will be very accurate against training data but will exhibit unpredictable and (hopefully) bad results against test data. An example of an overfit model is a complex polynomial function that essentially encodes the training set but then fails when tested against data that differs significantly.

1. Describe the meaning of *optimal tradeoff* in relationship to complexity, underfitting, and overfitting.

The goal of modeling is to minimize the overall error. Jedamski presents that total error is the sum of bias, variance, and irreducible error. With the last component being constant, to get the best model we want to minimize the sum of bias and variance. This occurs at a point where neither bias nor variance are at their minimums, but their sum is. See Figure 5 where the notional optimum model is roughly in the middle of the graph.

1. Describe *hyperparameter tuning* and its role in data modeling.
   1. Include a description of the difference between a parameter and a *hyper*parameter.
   2. How does hyperparameter tuning relate to underfitting or overfitting a decision tree?

The video and other resources explain that parameters are internal to the model while hyperparameters are external but impactful on the training of the model. An example that I’ve used to describe these in the past is in a linear model, a coefficient is a parameter, and the number of coefficients is a hyperparameter. Hyperparameters define things about the model and parameters are part of the model.

Hyperparameters define the model complexity. This relates directly to underfitting and overfitting. As discussed above if the model is too complex it will overfit if not complex enough it will underfit.

1. Describe *regularization* and its role in data modeling. *Do not worry too much about the gritty details of this*. Focus on the basic meaning and these two questions:
   1. What is its goal?
   2. How does Occam’s razor apply?

Regularization provides limits (referred to as guardrails in the video) that constrain the complexity of a model. A loss function is a way of measuring how well a machine learning model is performing. Regularization will add a penalty to the loss value of a model based on its complexity. The more complex the model the greater the penalty. For a more complex model to be acceptable regularization forces it to be significantly more accurate than a simpler model with less of a penalty.

In the context of artificial intelligence, Occam’s razor suggests that the simplest model that fits the data is usually the best. Finding the simplest model – the model that is complex enough to fit the data, but no more complex, is exactly what regularization is intended to do. By penalizing complexity, regularization favors simplicity.

References:

Bruce, P., Bruce, A., & Gedeck, P. (2020). *Practical statistics for data scientists: 50+ essential concepts using R and Python* (2nd ed.). O’Reilly Media.

scikit-learn. (2025, April 3). In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Scikit-learn&oldid=1283740733

scikit-learn developers. (2025). *scikit-learn: Machine learning in Python (Version 1.6.1)*. https://scikit-learn.org/stable/index.html