Phase 4: Model Building

In Phase 4, the data science team needs to develop datasets for training, testing, and production purposes. These datasets enable the data scientist to develop the analytical model and train it (“training data”), while holding aside some of the data (“hold-out data” or “test data”) for testing the model. (These topics are addressed in more detail in Chapter 3.) During this process, it is critical to ensure that the training and test datasets are sufficiently robust for the model and analytical techniques. A simple way to think of these datasets is to view the training dataset for conducting the initial experiments and the test sets for validating an approach once the initial experiments and models have been run. In the model building phase, shown in Figure 2-6, an analytical model is developed and fit on the training data and evaluated (scored) against the test data. The phases of model planning and model building can overlap quite a bit, and in practice one can iterate back and forth between the two phases for a while before settling on a final model. Although the modeling techniques and logic required to develop models can be highly complex, the actual duration of this phase can be short compared to the time spent preparing the data and defining the approaches. In general, plan to spend more time preparing and learning the data (Phases 1–2) and crafting a presentation of the findings (Phase 5). Phases 3 and 4 tend to move more quickly, although they are more complex from a conceptual standpoint. As part of this phase, the data science team needs to execute the models defined in Phase 3. During this phase, users run models from analytical software packages, such as R or SAS, on file extracts and small datasets for testing purposes. On a small scale, assess the validity of the model and its results. For instance, determine if the model accounts for most of the data and has robust predictive power. At this point, refine the models to optimize the results, such as by modifying variable inputs or reducing correlated variables where appropriate. In Phase 3, the team may have had some knowledge of correlated variables or problematic data attributes, which will be confirmed or denied once the models are actually executed. When immersed in the details of constructing models and transforming data, many small decisions are often made about the data and the approach for the modeling. These details can be easily forgotten once the project is completed. Therefore, it is vital to record the results and logic of the model during this phase. In addition, one must take care to record any operating assumptions that were made in the modeling process regarding the data or the context. Creating robust models that are suitable to a specific situation requires thoughtful consideration to ensure the models being developed ultimately meet the objectives outlined in Phase 1. Questions to consider include these: ● Does the model appear valid and accurate on the test data? ● Does the model output/behavior make sense to the domain experts? That is, does it appear as if the model is giving answers that make sense in this context? ● Do the parameter values of the fitted model make sense in the context of the domain? ● Is the model sufficiently accurate to meet the goal? ● Does the model avoid intolerable mistakes? Depending on context, false positives may be more serious or less serious than false negatives, for instance. (False positives and false negatives are discussed further in Chapter 3 and Chapter 7, “Advanced Analytical Theory and Methods: Classification.”) ● Are more data or more inputs needed? Do any of the inputs need to be transformed or eliminated? ● Will the kind of model chosen support the runtime requirements? ● Is a different form of the model required to address the business problem? If so, go back to the model planning phase and revise the modeling approach. Once the data science team can evaluate either if the model is sufficiently robust to solve the problem or if the team has failed, it can move to the next phase in the Data Analytics Lifecycle.

2.5.1 Common Tools for the Model Building Phase

There are many tools available to assist in this phase, focused primarily on statistical analysis or data mining software. Common tools in this space include, but are not limited to, the following: ● Commercial Tools: ● SAS Enterprise Miner [17] allows users to run predictive and descriptive models based on large volumes of data from across the enterprise. It interoperates with other large data stores, has many partnerships, and is built for enterprise-level computing and analytics. ● SPSS Modeler [18] (provided by IBM and now called IBM SPSS Modeler) offers methods to explore and analyze data through a GUI. ● Matlab [19] provides a high-level language for performing a variety of data analytics, algorithms, and data exploration. ● Alpine Miner [11] provides a GUI front end for users to develop analytic workflows and interact with Big Data tools and platforms on the back end. ● STATISTICA [20] and Mathematica [21] are also popular and well-regarded data mining and analytics tools. ● Free or Open Source tools: ● R and PL/R [14] R was described earlier in the model planning phase, and PL/R is a procedural language for PostgreSQL with R. Using this approach means that R commands can be executed in database. This technique provides higher performance and is more scalable than running R in memory. ● Octave [22], a free software programming language for computational modeling, has some of the functionality of Matlab. Because it is freely available, Octave is used in major universities when teaching machine learning. ● WEKA [23] is a free data mining software package with an analytic workbench. The functions created in WEKA can be executed within Java code. ● Python is a programming language that provides toolkits for machine learning and analysis, such as scikit-learn, numpy, scipy, pandas, and related data visualization using matplotlib. ● SQL in-database implementations, such as MADlib [24], provide an alterative to in-memory desktop analytical tools. MADlib provides an open-source machine learning library of algorithms that can be executed in-database, for PostgreSQL or Greenplum.