3 *Data Mining in a Nutshell*

The purpose of this chapter is to introduce the reader to the main concepts and data mining tools used in business analytics. Methods are described at a non-technical level, focusing on the idea behind the method, how it is used, advantages and limitations, and when the method is likely to be of value to business objectives.

The goal is to transform data into information, and information into insight.

—- Carly Fiorina (1954–)
President of Hewlett Packard, 1999–2005

3.1 What Is Data Mining?

Data mining is a field that combines methods from artificial intelligence, machine learning[1], statistics, and database systems. Machine learning and statistical tools are used for the purpose of learning through experience, in order to improve future performance. In the context of business analytics, data mining is sometimes referred to as "advanced analytics"[2].

We want machines that are able to learn for several reasons. From large amounts of data, hidden relationships and correlations can be extracted. Scenarios such as changing environments highlight the need for machines that can learn how to cope with modifying surroundings. Computer learning algorithms that are not produced by detailed human design but by automatic evolution can accommodate a constant stream of new data and information related to a task.

Data mining focuses on automatically recognizing complex patterns from data, to project likely outcomes. Learning is defined as the acquisition of knowledge or skill through experience. In data mining, we train computational methods

[1] Machine learning is a scientific discipline concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data, such as from sensor data or databases.

[2] We avoid referring to analytics as "simple" or "advanced" as these terms more appropriately describe the usage level of analytics.

to learn from data for the purpose of applying the knowledge to new cases.

The main challenge of data mining techniques is the ability to learn from a finite set of samples (data) and be able to generalize and produce useful output on new cases (scoring).

Within data mining, algorithms are divided into two major types of learning approaches:

Supervised learning: we know the labels or outcomes for a sample of records, and we wish to predict the outcomes of new or future records. In this case, the algorithm is trained to detect patterns that relate inputs and the outcome. This relationship is then used to predict future or new records. An example is predicting the next move in a chess game.

The fundamental approach in supervised learning is based on training the model and then evaluating its performance. Data are therefore typically segmented into three portions:

- **Training data:** the data used for training the data mining algorithm or model
- **Validation data:** used to tweak models and to compare performance across models. The rule of thumb is 80–20 for training and validation data.
- **Test data (or hold-out data):** used to evaluate the final model's performance, based on its ability to perform on new previously "unseen" data.

Unsupervised learning: we have a sample of records, each containing a set of measurements but without any particular outcome of interest. Here the goal is to detect patterns or associations that help find groups of records or relationships between measurements, to create insights about relationships between records or between measurements. An example is Amazon's recommendation system that recommends a set of products based on browsing and purchase information.

3.2 *Predictive Analytics*

This set of tools includes a wide variety of methods and algorithms from statistics and machine learning. We cover a few of the most popular predictive analytics tools. Interested

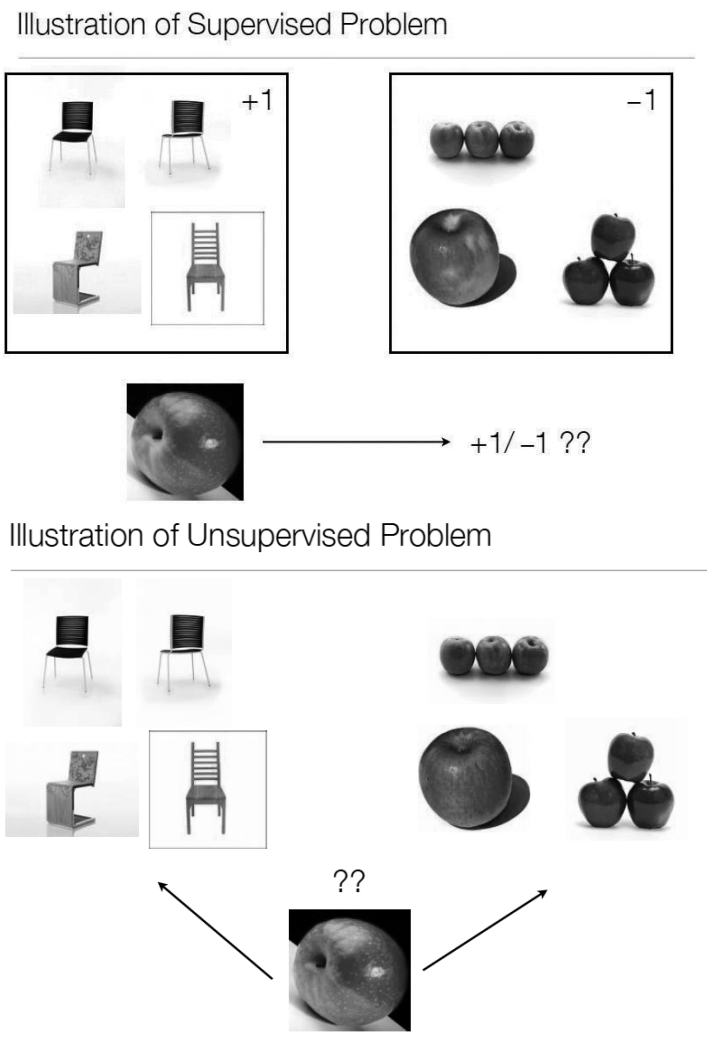


Figure 3.1: The difference between supervised and unsupervised problems. In the supervised learning task we try to classify the object as chair (+1) or an apple (−1). In the unsupervised learning case, we try to measure similarity of an item to other items.

reader can obtain information about further methods or further technical details from more specialized books.

COMPUTERS ARE USELESS. THEY CAN ONLY GIVE YOU ANSWERS.

— Pablo Picasso (1881–1973)

Supervised Learning

In supervised learning, for each record we have a set of input measurements as well as a known target or outcome measurement. For example, in a customer database of mobile phone users, where we are interested in modeling customer churn, a record is a customer. For each customer, input measurements can include demographic information as well as call and billing history. A possible outcome measurement is whether the customer stays with the company for at least a year.

The purpose of supervised learning methods is to find a relationship between the input measurements and the outcome measurement. In the mobile customer churn example, we are looking for a relationship between customer attributes and behavior and their attrition.

Another classic example of a supervised learning task is the prediction of spam (unsolicited email) for the purpose of spam filtering. Each record is an email message, for which we have multiple input measurements such as the sender address, the title, and text length. The outcome of interest is a label of "spam" or "non-spam."

In the above examples of customer churn and spam, the outcome measurement is categorical: whether a customer stays or not, or whether an email is spam or not. This type of outcome is called a *class*. Predicting the outcome is therefore called *classification*.

Supervised learning includes scenarios where the outcome measurement is either categorical or numerical. Some examples of a numerical outcome are predicting the duration of service calls at a call center based on input measurements that are available before the call is taken, or predicting the amount of cash withdrawn in each ATM transaction before the actual amount is keyed in by the customer. When the outcome is numerical, the supervised learning task is called *prediction*[3].

[3] In machine learning, the term used for predicting a numerical outcome is *regression*.

The following supervised learning techniques are used for classification and/or prediction. The various methods, each with strengths and weaknesses, approach the task of detecting potential relationships between the input and outcome measurements differently.

k-Nearest Neighbors (k-NN)

k-nearest neighbors (k-NN) algorithms are useful for both classification and prediction. They can be used to predict categorical and numerical outcomes. The algorithm identifies *k* records in the training set that are most similar to the record to be predicted, in terms of input measurements. These *k* neighbors are then used to generate a prediction of the outcome for the record of interest. If the outcome is categorical, we let the neighbors 'vote' to determine the predicted class of the record of interest. If the outcome is numerical, we simply take an average of the neighbors' outcome measurement to obtain the prediction.

The nearest neighbors approach is what real estate agents tend to instinctively use when pricing a new property. They seek similar properties in terms of size, location and other features and then use these reference properties to price the new property.

Consider the mobile customer churn example for predicting how likely a new customer is to stay with the company for at least one year. The *k*-nearest-neighbors algorithm searches the customer database for a set of *k* customers similar to the to-be-predicted customer in terms of demographic, calling and billing profiles. The algorithm then "considers" the churn behavior of the *k* neighbors and uses the most popular class (churn/no churn) to predict the class of the new customer. If we are interested in a *probability* of churn, the algorithm can compute the percentage of neighbors who churned.

In the call-center call duration example, we want to predict the duration of an incoming call before it begins. The k-NN algorithm searches the historic database for *k* calls with similar features (information available on the caller, call time, etc.). The average call duration of these *k* similar calls is then the predicted duration for the new call.

To illustrate the k-NN algorithm graphically, consider the example of predicting whether an online auction will be competitive or not. A competitive auction is one that receives more than a single bid. Using a set of over 1,000 eBay auctions, we examine two input measurements in each auction:

the seller rating (where higher ratings indicate more experience) and the opening price set by the seller.

The relationship between the auction competitiveness outcome and these two inputs is shown in Figure 3.2. Suppose that we want to predict the outcome for a new auction, given the seller rating and opening price. This new record is denoted by a question mark in the chart. The k-NN algorithm searches for the k nearest auctions. In this case k was chosen to be 7. Among the seven neighbors, five were competitive auctions; the predicted probability of this auction to be competitive is therefore 5/7. If we use a majority rule to generate a classification, then the five competitive auctions are the majority of the seven neighboring auctions, and k-NN classifies the new auction as being competitive.

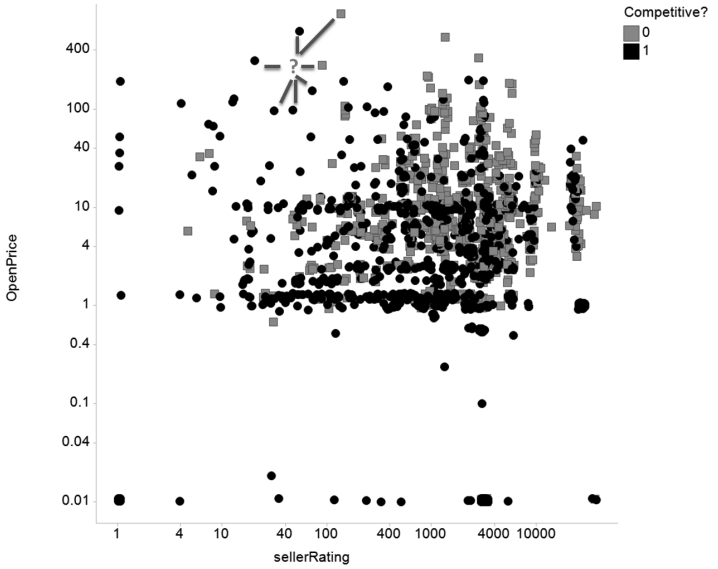


Figure 3.2: Competitive auctions (black circles) and non-competitive auctions (gray squares) as a function of seller rating and opening price in eBay auctions. k-nearest neighbors classifies a new auction's competitiveness based on k auctions with similar seller ratings and opening prices.

A k-nearest neighbors algorithm requires determining two factors: the number of neighbors to use (k) and the definition of similarity between records. The number of neighbors should depend on the nature of the relationship between the input and outcome measurements in terms of its global versus local nature. In a global pattern, the same relationship holds across all input values, whereas in local patterns different relationships exist for different values of the input values.

In the mobile churn example, if churn decreases in age regardless of other demographics or billing features, then we can say that there is a global relationship between churn and age. However, if churn decreases in age only for heavy callers

but increases for low-volume callers, then the relationship between churn and age is local. A small number of neighbors is better for capturing local relationships — only a small set of very close neighbors would be similar to the record of interest — whereas in global relationships a large number of neighbors leads to more precise predictions.

The choice of k is typically done automatically. The algorithm is run multiple times, each time varying the value of k (starting with $k = 2$) and evaluating the predictive accuracy on a validation set. The number of neighbors that produces the most accurate predictions on the validation set is chosen.

Similarity between records can be determined in many ways. Records are compared in terms of their input measurements. The similarity metric most commonly used in k-NN algorithms is Euclidean distance. To measure the distance between two records, we look at each input measurement separately and measure the squared difference between the two records. We then take a sum of all the squared differences across the various input measurements. This is the Euclidean distance between two records.

For example, the Euclidean distance between two auctions is computed by summing up the squared difference between the pair of seller ratings and the squared difference between the pair of opening prices. You may have noticed that computing a Euclidean distance in this way will produce a similarity measure that gives much more weight to input measurements with large scales (such as seller ratings, compared to opening prices). For this reason, it is essential to first normalize the input measurements before computing Euclidean distances. Normalizing can be done in different ways. Two common normalizing approaches are converting all scales to a [0,1] scale or subtracting the mean and dividing by the standard deviation. While similarity between records can be measured in different ways, Euclidean distance is appealing because of its computational efficiency.

In k-NN, computational efficiency is especially important because the algorithm computes the similarity between the to-be-predicted record with each and every record in the training data. Moreover, if we want to predict many new records (such as for a large set of potential customers), the computational task can be heavy.

The Verdict: Among supervised learning methods, k-NN is simple to explain and easy to automate. It can be used for both prediction and classification and is highly data-driven,

i.e., there are no assumptions about the nature of the relationship between the outcome and inputs. While k-NN is simple to explain, it produces "black-box" predictions because it is not clear which inputs contribute to the prediction and to what degree. When transparency is needed, k-NN is not an appealing candidate.

One key requirement of k-NN algorithms is sufficient training data. k-NN must be able to find a sufficient number of close neighbors to produce accurate predictions. Unfortunately, the number of required records increases exponentially in the number of input measurements, a problem called "the curse of dimensionality". Another challenge that KNN faces is computational: the time to find the nearest neighbors in a large training dataset can be prohibitive. While there are various tricks to try to address the curse of dimensionality and the computational burden, these two issues must be considered as inherent challenges within k-NN.

Classification and Regression Trees (CART)

Classification and regression trees are supervised learning algorithms that can be used for both classification ("classification trees") and prediction ("regression trees"). Like k-NN, the idea is to define neighborhoods of similar records, and to use those neighborhoods to produce predictions or classifications for new records. However, the way that trees determine neighborhoods is very different from k-NN. In particular, trees create *rules* that split data into different zones based on input measurements, so that each zone is dominated by records with a similar outcome. In the eBay auctions example, we might have a rule that says "IF the opening price is below $1 AND the seller rating is above 100, THEN the auction is competitive."

To create rules, tree algorithms sequentially split input predictors into two ranges: above and below some value. The algorithm starts by looking for the best split, one that produces two sets of records that are each as homogeneous as possible in terms of the outcome. Finding the best split requires trying different values across all the different input measurements. Once the first split is determined, the algorithm searches for another split, which will further break down the two data groups into three sets. The next step is finding a third split, and so forth.

The splitting process is illustrated graphically in Figure 3.3 for two input measurements, using the eBay auctions example. The first split uses the opening price, and creates two

zones: above and below $1.23. We can see that the lower zone is dominated by competitive auctions, while the upper zone is more balanced, but contains more non-competitive auctions. The second split further separates the high opening price zone (above $1.23) by seller rating, with a high seller rating zone (above 300) and a low seller zone (300 or below). This second split breaks down the upper zone into two strips, one with mostly competitive auctions and the other with mostly non-competitive auctions. The third split separates the "high opening price, high seller rating" zone further, by seller rating (above/below $5).

Displaying the results in a scatter plot with splits as in Figure 3.3 is no longer possible once additional input measurements are introduced. However, there is an alternative powerful chart that can be used to display the resulting splits, in the form of a tree. The tree for this same example is shown in Figure 3.4. Starting from top to bottom, each layer of the tree represents a split, in the order it occurred. The rectangles are called "leaf nodes" or "terminal nodes", and they represent the outcome of the records that fall in that zone. For example, 89% of the auctions with an opening price below $1.23 are competitive (as can be seen in Figure 3.4).

To convert the leaf node probabilities to competitive/non-competitive classifications, we specify a majority threshold. The default threshold of 50% means that a leaf node with fewer than 50% competitive auctions will lead to a non-competitive classification (such as the right-most leaf node in Figure 3.4), whereas a leaf node with 50% or more competitive auctions will lead to a competitive classification. For a numerical outcome, such as call duration, the leaf node label is typically the average outcome of records in that leaf node.

A tree can easily be translated into a set of logical rules that relate the outcome of interest to the input measurements. In our example, using a 50% majority threshold, we have four rules:

1. IF the opening price is below $1.23, THEN the auction is competitive.

2. IF the opening price is above $1.23, AND the seller rating is below 300, THEN the auction is competitive.

3. IF the opening price is above $1.23, AND the seller rating is above 300, AND the opening price is below $5, THEN

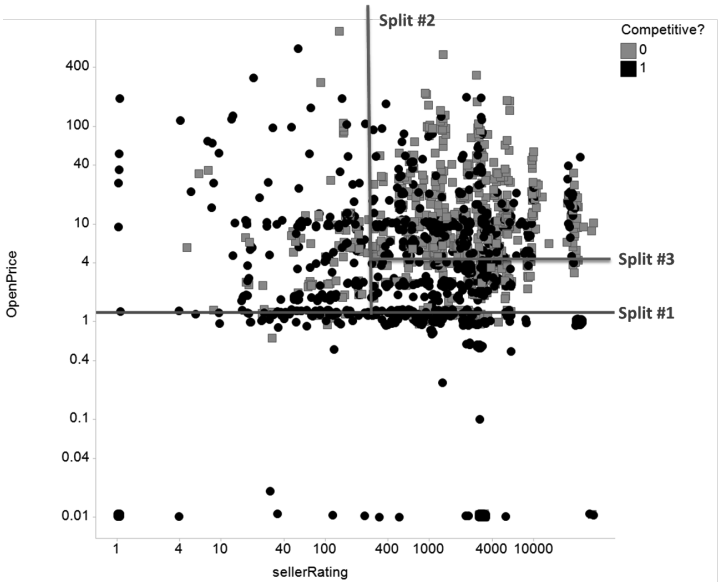


Figure 3.3: Competitive auctions (black circles) and non-competitive auctions (grey squares) as a function of seller rating and opening price in eBay auctions. The classification tree splits the auctions first by opening price (above/below$1.23), then the high opening price zone is split by seller rating above/below 300, and then the high seller rating zone is further split by opening price (above/below $5). Each of the four zones has a majority of auctions with similar outcomes (competitive/non-competitive).

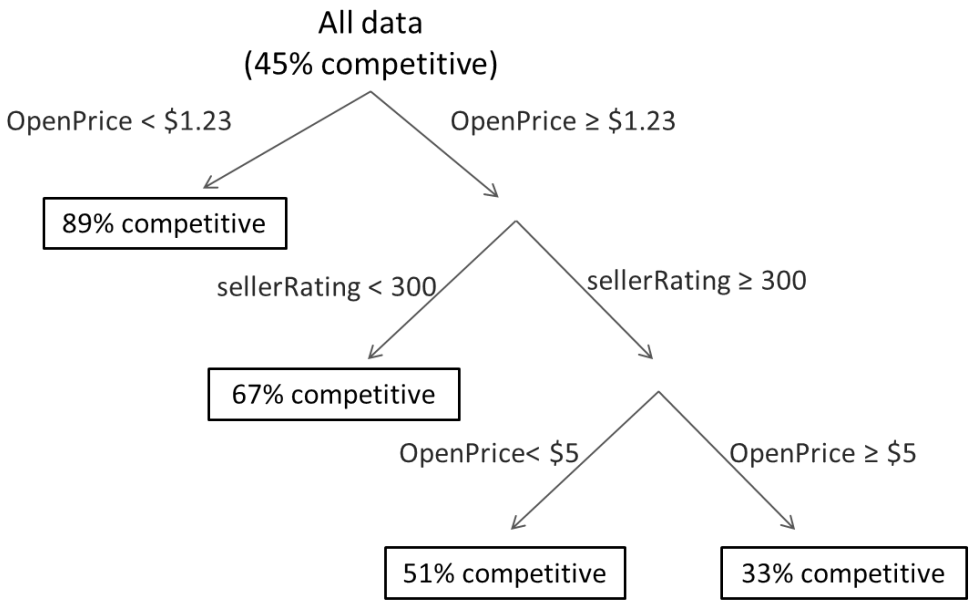


Figure 3.4: Displaying the algorithm splitting results in the form of a tree.

the auction is competitive.

4. IF the opening price is above $1.23, AND the seller rating is above 300, AND the opening price is above $5, THEN the auction is non-competitive.

Obviously, these rules can be further compressed into a smaller set of rules ("IF the opening price is above $5 AND the seller rating is above 300, THEN the auction is non-competitive; otherwise it is competitive").

Using a tree to generate predictions or classifications is straightforward: simply drop a new record at the top of the tree and find out in which leaf node it lands. The leaf node then determines the class or prediction for that record.

The two factors that must be determined in a tree algorithm are the function measuring homogeneity of a set of records in terms of the outcome, and the size of the tree in terms of splits. Typical homogeneity measures are the famous Gini index for classification tasks, and the standard deviation for prediction tasks. Most software packages will have defaults and some allow users to choose between different homogeneity measures.

Determining tree size is an important step. Too many splits result in over-fitting the training data. An extreme example is when each zone contains a single class. This will typically require many splits and the final zone will contain very few records. Most tree algorithms use an automated approach, where validation data are used to determine the tree size, thereby avoiding over-fitting. A tree that produces predictions on the training data that are much more accurate than predictions on the validation data is clearly over-fitting. Algorithms are designed to avoid this scenario.

The Verdict: CART has proven useful in a wide range of business applications. Its biggest strength among data-driven algorithms is transparency and interpretability. The ability to understand the relationship between a prediction and the input measurements is crucial in applications such as insurance underwriting, as well as for increasing the comfort level of users who are not analytics-savvy. Trees are often used in applications with a large number of potential input measurements, for the purpose of ranking the importance of the input measurements.

Trees are highly automated. They do not require any user input, and do not make assumptions about the type of relationship between the outcome and input measurements. Be-

cause of their data-driven nature, they require a large number of records for training, and a separate set for validation. Trees are robust to extreme records, and can be applied to data with missing values. However, a small change in the data can result in a different tree. In terms of computation, trees can be computationally expensive, increasingly so as the number of input measurements increases.

Finally, extensions of trees such as *random forests* and *boosted trees* improve prediction accuracy and stability. These extensions are based on creating multiple trees from the data and combining them to produce better predictions.

Regression Models

CART and k-NN are both data-driven algorithms, where users need not specify the form of the relationship between the outcome and the input measurements. These methods therefore require large amounts of training data, and are subject to over-fitting. A different approach is to model the relationship between the outcome and input measurements via a statistical model of the form:

$$Outcome = \beta_0 + \beta_1 \times Input_1 + \beta_2 \times Input_2 + \ldots$$

We call this type of model a *regression model*.

Consider the example of a car manufacturer interested in predicting the cost of the three-year warranty it provides to customers. The manufacturer has access to a large dataset of historical warranty claims, with detailed information on each claim. The challenge is to determine which input measurements are predictive of warranty cost and how to use this information to predict the warranty cost for new cars. Figure 3.5 illustrates a regression model for modeling the relationship between the warranty cost (outcome) and the vehicle odometer reading (single input measurement)[4]. The straight line denotes the regression equation, and we see that the specific regression formula estimated from the training data is:

$$\text{Warranty Cost} = 174.45 + 0.01 \times \text{Odometer Reading},$$

where cost is in USD and odometer reading is in miles. The regression tells us that the higher the odometer reading, the higher the warranty cost. Moreover, each additional mile on the odometer increases the predicted cost by one cent ($0.01). While it is difficult to display a model of warranty cost on more than a single input measurement using a chart, regression models do scale up to cases of multiple input measurements.

[4] The data for this illustration is a sample from the "Don't Get Kicked" competition on www.kaggle.com.

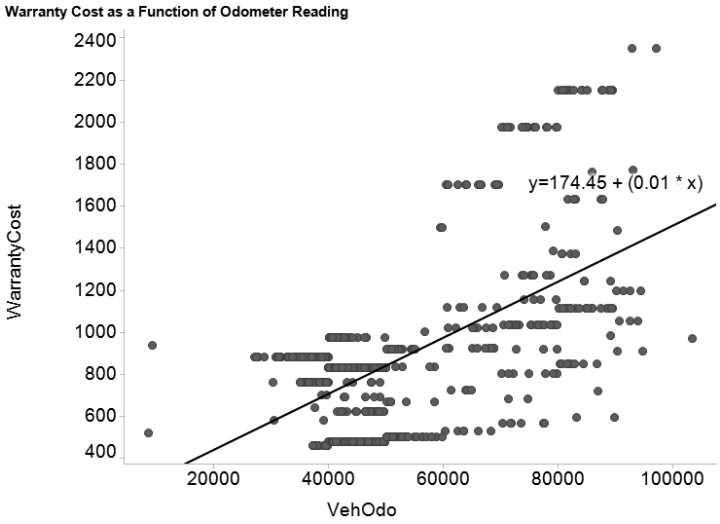


Figure 3.5: A regression model for predicting warranty cost of a vehicle by its odometer reading. Each point corresponds to a vehicle.

Although a linear relationship is a rough approximation of a more complicated relationship, it may be sufficient for producing sufficiently accurate predictions. Moreover, variations of the linear model above can yield input-outcome functions that are non-linear, such as exponential or polynomial. For a categorical outcome with two or more classes, regression formulas are available that link class probability with the inputs. A popular regression model for a categorical outcome is the logistic regression model.

By specifying a mathematical function that relates outcome to input measurements, we significantly reduce the computational burden and use the data more efficiently. Rather than using the data to detect the relationship structure, we specify the structure and use the data only to estimate the function parameters $(\beta_0, \beta_1, \beta_2, \ldots)$.

The Verdict: Regression models are popular predictive tools. They can be used for classification and prediction and can capture linear, non-linear or any pre-specified relationship functions between the outcome and input measurements. In addition to providing predictions, regression models also provide information about the importance of different input measurements through quantifiable coefficients (β values). Like CART, regression models are considered transparent and interpretable. They do not require large amounts of training data. By pre-specifying an approximate formula (such as a linear function) smaller amounts of data are needed. The main weakness of regression models is the

need for users to specify the formula linking the outcome to the inputs.

Ensembles

Given the variety of predictive algorithms, each with its strengths and weaknesses, one approach is to compare the results and choose the algorithm that yields the best results. The algorithm with the most precise predictions is not necessarily the best. Other considerations such as operational costs, future software availability or run time may lead to a different choice of algorithm.

An alternative approach is to combine results from several algorithms. Combining results can be done in different ways. A simple approach is to average the results. For example, for predicting warranty cost of a new vehicle, we can take the average of the cost predictions generated by different algorithms. For a classification task such as churn/no-churn, we can classify a new customer by a majority vote of class predictions by different algorithms.

The averaging or voting operations are easily automated, thereby producing a hybrid algorithm, typically called an ensemble. Ensembles rely on the same principle as diversified financial portfolios: they reduce risk. In the context of prediction, this means that ensembles produce more precise predictions. This phenomenon has been demonstrated in many real world cases. Ensembles played a major role in the million-dollar Netflix Prize contest[5], where teams competed in creating the most accurate predictions of movie preferences by users of the Netflix DVD rental service. Different teams ended up joining forces to create ensemble predictions, which proved more accurate than the individual predictions. The winning "BellKor's Pragmatic Chaos" team combined results from the "BellKor" and "Big Chaos" teams and additional members. In a 2010 article in *Chance magazine*, the Netflix Prize winners described the power of their ensemble approach [2]:

[5] www.netflixprize.com

> An early lesson of the competition was the value of combining sets of predictions from multiple models or algorithms. If two prediction sets achieved similar RMSEs, it was quicker and more effective to simply average the two sets than to try to develop a new model that incorporated the best of each method. Even if the RMSE for one set was much worse than the other, there was almost certainly a linear combination that improved on the better set.

Another way to create ensembles is to split the data into

multiple sub-samples, to run a particular algorithm separately on each sample, and then to combine the results. There are also various ways to combine results. For example, instead of an average we can take a weighted average that gives more weight to more precise predictions. Lastly, some data mining algorithms are themselves ensembles. For example, *random forests* are an ensemble of classification or regression trees.

The Verdict: Ensembles, or combining algorithm results, is a useful way to generate more precise and more robust predictions. The only caveat with the ensemble approach is that it requires more resources than a single algorithm. It requires running each of the algorithms at the production stage and whenever new predictions are needed. Constraints such as run time or future software availability should therefore be carefully considered.

Unsupervised Learning

In unsupervised learning, we have a set of measurements for each record. As in supervised learning, it is important to define what is a record, because different definitions will lead to different data structures. For example, from a bank's database we can extract a dataset of customers, where a set of measurements is available for each customer: demographic information, banking history (number of monthly ATM transactions, teller transactions, loans, etc.). Alternatively, we can extract a dataset of single transactions, where measurements are given on each transaction (time of day, type of transaction, amount, etc.). A third option is to define a record as a teller encounter, where measurements are the types of products or services rendered in that encounter as well as the customer data.

The purpose of unsupervised learning methods is to find relationships either between measurements or between records. Note that in contrast to supervised learning, we do not differentiate between input and outcome measurements. In the bank customer dataset, for instance, we might be interested in discovering different types of customers in terms of their banking history. This popular task is called *customer segmentation*. In the teller encounter example, we might want to learn which products/services tend to go together, for purposes such as providing product recommendations, determining marketing campaigns, etc. Several algorithms are

popular for these various tasks, ranging from *collaborative filtering* methods to *market basket analysis*.

A third relationship of interest is figuring out the amount of information overlap in a large set of measurements, usually in an effort to reduce the number of measurements to a smaller, more manageable one. This process is called *dimension reduction*. In the following sections we describe several popular unsupervised data mining techniques.

Cluster Analysis (Segmentation)

Cluster analysis is an exploratory tool for discovering clusters of similar records. Humans can easily detect similarities and differences between objects in some contexts. For example, we easily distinguish between a cat and a dog, despite the fact that they share many common features (four legs, fur, etc.). Yet, when it comes to many rows of records with multiple measurements, our ability to find similarities and differences between records is challenged.

Clustering algorithms are an automated way of detecting clusters of similar records. Similarity requires defining a metric that compares all the measurements of interest. For example, the similarity between two bank accounts can be measured by comparing the different features of the two accounts such as account type, recent activity, current balance, etc. Various distance metrics, each with advantages and shortcomings, are used to measure the difference between a pair of records.

A popular metric is Euclidean distance, which aggregates the squared difference for each feature (we have discussed this metric earlier in the context of k-NN). For example, bank account #1 has a current balance of $2,500 and the most recent activity is 24 hours ago. Account #2 has a current balance of $3,000 and the most recent activity is 14 hours ago. The Euclidean distance between accounts #1 and #2 is $(2,500-3,000)^2+(24-14)^2=250,100$. Note that using Euclidean distance requires re-scaling (normalizing) all features so that they have the same scale. Otherwise, features that have units with a wide range, such as *current balance*, will dominate the distance.

Figure 3.6 graphically illustrates the notion of clustering. The chart compares 821 vehicle records, each represented by a horizontal line (row). For each vehicle, we have four features (from left to right: odometer reading, acquisition cost,

whether the vehicle was purchased online, and warranty price). The chart uses color to denote the value on each feature, so that darker color corresponds to higher values. We sorted the vehicles by odometer reading, from high to low. On this chart, we can visually detect a few types of vehicles. At the very top are "high-mileage-inexpensive-online-purchased-low-warranty-cost vehicles." At the bottom of the chart we can spot a cluster of vehicles with very low odometer readings, acquisition costs, and warranty costs. Most vehicles were purchased offline (one cluster) and a few online (another cluster).

Note how difficult it is to clearly separate the 821 vehicles into clear-cut clusters based on all four features. For this reason, data mining algorithms are needed. The result of applying a clustering algorithm is shown in Figure 3.7. The algorithm detected four clusters (labeled 10, 6, 8, 9). Clusters 10 and 6 comprise the vehicles that were purchased offline. The main difference between the clusters is the higher warranty cost and odometer readings in cluster 10.

The Verdict: We can think of cluster analysis as a way for compressing rows of a dataset. Cluster analysis is a useful approach for exploring groupings of records, based on a set of measurements. It is especially advantageous in Big Data, where the number of records and the number and variety of measurements is large.

Some clustering algorithms are designed for very large datasets and can be deployed for high-performance implementation. Note that cluster analysis is an exploratory approach. There are no correct or incorrect results. The key question is always whether the resulting clusters offer insights or lead to insightful decisions. Segmenting records into clusters is useful for various reasons and applications, including differentiated customer targeting (customary in marketing campaigns), deploying different policies to different segments (as in credit card scoring), and more.

Dimension Reduction

One feature of Big Data is the large number of measurements for each record. This richness is the result of the proliferation of digitized data collection mechanisms, including scanners, RFID readers, GPS devices and automated recording of online and mobile footprints. While more measurements may increase the richness of data, they also add a lot of noise due to errors and other factors. Moreover, measure-

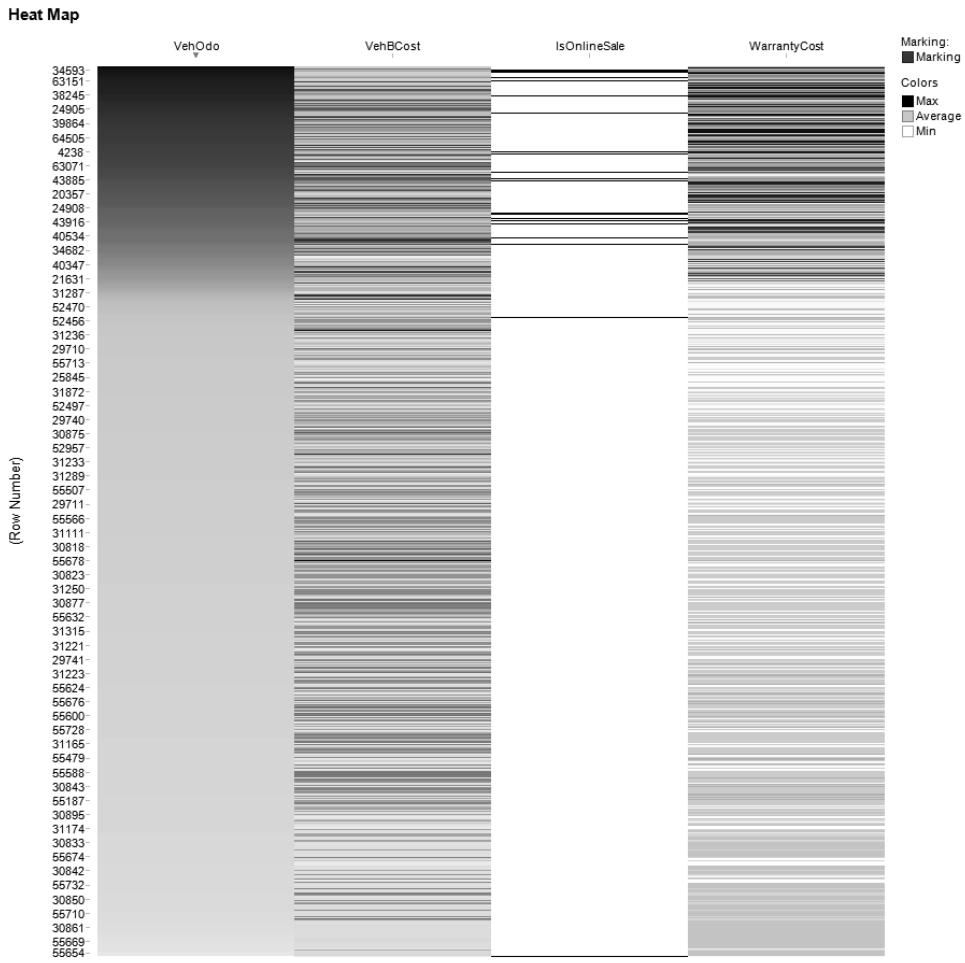


Figure 3.6: Heatmap comparing 821 vehicle records (rows) across four features. Darker color denotes higher values.

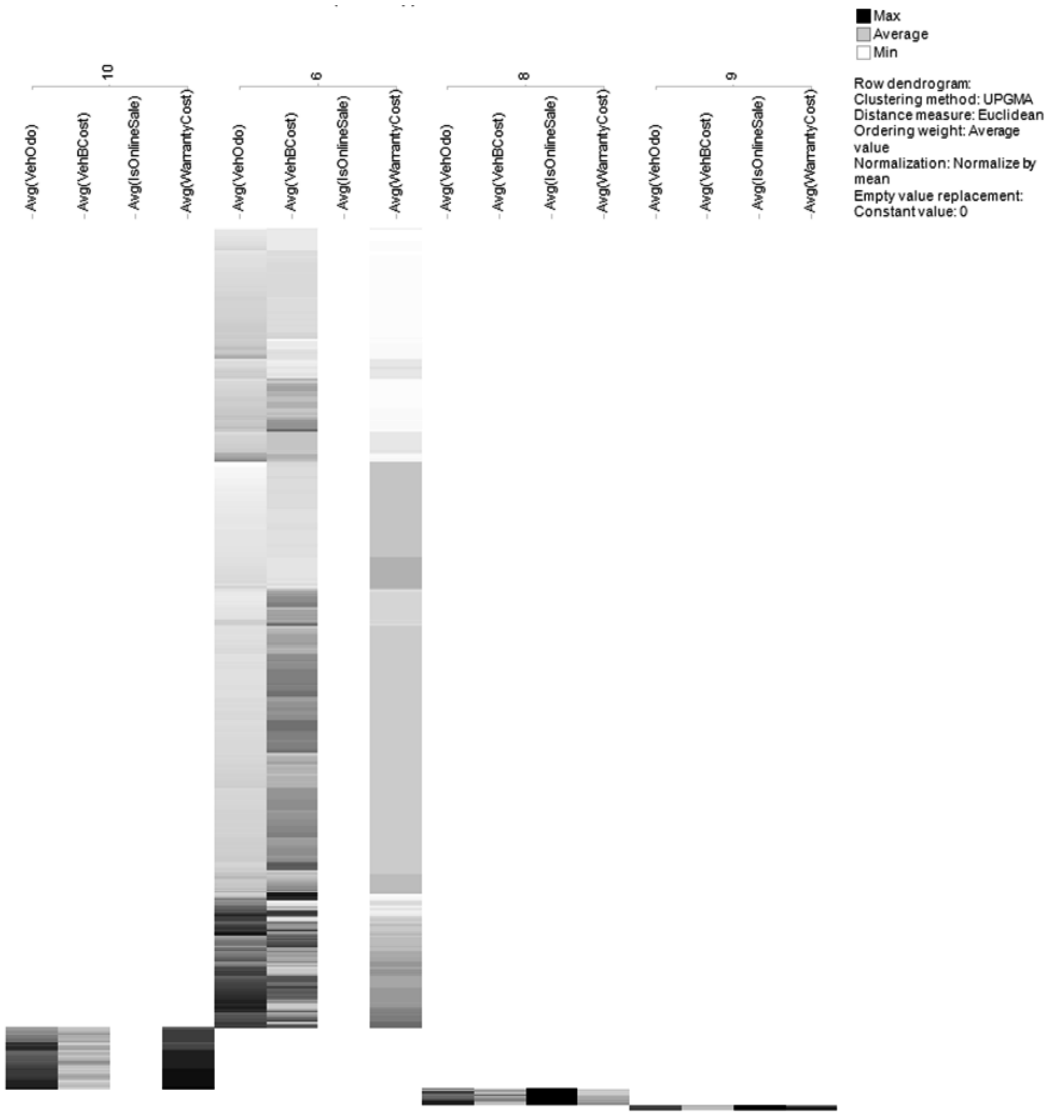


Figure 3.7: Results from a clustering algorithm: The 821 vehicle records are grouped into four clusters.

ments often contain an overlap of information. For example, *age* and *years since opening bank account* are likely to be highly correlated. The challenge is to determine what part of the data contains the information and to separate it out from the noise.

Dimension reduction techniques are aimed at reducing the number of measurements (the dimension of the data) to a smaller set that contains a high ratio of information-to-noise. One approach is to choose a particular set of information-rich measurements and to discard all the other measurements. A different approach is to derive a small set of new measurements, based on the large set of original measurements. This is the basis for popular data mining algorithms such as Principal Components Analysis (PCA), Singular Value Decomposition (SVD), and Factor Analysis.

In PCA, each new measurement, called a *principal component*, is a weighted average of the original measurements. What is special about these principal components is that they contain no information overlap. In other words, the correlation between each pair of principal components is zero. A small number of principal components (typically less than 10) is sufficient to capture most of the variability in the dataset.

For example, in the vehicle data that we described earlier, running PCA (after rescaling the four measurements) indicates that we can capture approximately 70% of the information that is contained in four measurements (odometer reading, acquisition cost, whether the vehicle was purchased online, and warranty price) by considering only two principal components: the average of these four measurements and whether the vehicle was purchased online[6].

The Verdict: We can think of dimension reduction as compressing the columns of a dataset. Dimension reduction is a crucial step in data mining, because datasets typically include tens, hundreds or even thousands of measurements. Domain knowledge can be used to eliminate measurements that are completely irrelevant. However, by eliminating measurements altogether, some useful information might be lost. Methods such as PCA and SVD approach dimension reduction in a different fashion. They help reduce information overlaps between measurements by creating a smaller set of variables that do not have information overlap. The resulting set of variables is smaller than the original set of measurements, and therefore easier to handle and faster to run

[6] The first principal component is approximately the simple average:

$$0.63\mathit{VehOdo} + 0.42\mathit{VehBcost} + 0.30\mathit{IsOnlineSale} + 0.58\mathit{WarrantyCost}.$$

The second principal component is dominated by *IsOnlineSale*, as can be seen from the computation

$$0.02\mathit{VehOdo} + 0.47\mathit{VehBcost} - 0.88\mathit{IsOnlineSale} + 0.08\mathit{WarrantyCost}.$$

in other analyses. However, computing these new variables usually requires access to the entire set of original measurements.

Association Rules (Market-Basket Analysis)

Scanners and online shopping create records of customers' "baskets." Each transaction includes a combination of products that are linked to the same record in a single transaction. Services such as movie rentals and telecom plans similarly have baskets for different customers: a basket of movies rented within the same transaction; a basket of mobile telecom services (such as a monthly 300-minute talk time, unlimited SMS and unlimited data). Similar data structures arise in healthcare: a doctor visit produces a combination of symptom data; a hospitalization produces a combination of medical procedures and tests. Online education produces data on groups of courses taken by a student.

Unlike the data structure that we described earlier, where we have a set of measurements (columns) on a set of records (rows), here different records have different measurements. If we think of baskets as records, then different baskets contain different items. This data structure is illustrated in Table 3.2, which shows an example for a fictitious pizza delivery service.

We see that different orders contain a different number of items. We could potentially restructure the data so that each item on the menu is a column and then we would list the item count for each order. However, when there are many different possible items (many SKUs in a grocery store, many possible symptoms, etc.) the latter data structure will contain mostly zeros. Association rules are an algorithm that is designed especially for such circumstances, i.e., a large set of potential items.

Order No.	Item 1	Item 2	Item 3	Item 4
1	Large Margherita Pizza	2-Liter Pepsi	Small salad	
2	Small Veg Pizza	Small Hawaiian Pizza		
3	Large Margherita Pizza	Large Margherita Pizza	2-Liter Pepsi	2-Liter Pepsi
4	Large salad	Small Veg Pizza		

The purpose of association rules algorithms is to discover relationships between items that co-appear in baskets, in

very large transactional datasets. The algorithm does this by searching for large sets of baskets that contain the same item combinations. The result is a set of statements about which item goes with which other items. More accurately, association rules generate IF/THEN rules such as "IF Tylenol, THEN Kleenex."

In the pizza example, discovering a rule such as "IF Large Margherita THEN 2-Liter Pepsi" can be used for coupon offers, discount bundles, stocking, and designing the menu. In other words, the rules are useful for generating common, impersonal decisions.

The Verdict: Association rules are useful for discovering relationships between measurements or items in large datasets. "Large" refers to the number of records as well as to a large number of potential items. The resulting association rules are easy to understand. However, two notes of caution are in place:

- With a large dataset, association rules will produce a large number of rules. One danger is "discovering" rules that do not generalize and instead reflect combinations that are simply due to chance.

- Discovered rules can be uninteresting or trivial. For example, discovering that bananas are purchased with many other products in grocery stores in the United States is unsurprizing, because bananas are a typical loss item that is offered to lure customers.

While association rules originated from grocery store transactional databases, they can be used in many other contexts and applications, including healthcare, finance, education, telecom, and more. Using association rules requires a large number of baskets and potential items, and stakeholder commitment toward generating common rules.

Collaborative Filtering

Recommender systems are common in almost every domain. They are used for book recommendations on Amazon.com, for music recommendations on personalized radio stations such as LastFM.com and on video sharing sites such as YouTube, for product recommendations in online shopping, and more.

Collaborative filtering is the algorithm behind many online recommender systems. Like association rules, collabora-

tive filtering uses large datasets on baskets that are combinations of products or services that are consumed or purchased together. However, unlike association rules that generate general, impersonal rules, collaborative filtering generates personalized recommendations. The idea is to generate user-specific associations based on information from many other users. The fundamental assumption is that if users rate items similarly, or have similar behaviors (e.g., buying, watching, listening), they will rate or behave similarly on other items [20].

Another difference is that association rules look for popular combinations in the sense that the combinations appear many times in the dataset. In contrast, recommender systems look for combinations that are frequent among baskets that have at least one of the items, regardless of how many such baskets there are. This means that association rules ignore non-popular items while recommender systems do not. These non-popular items are what is known as "the long tail."

Collaborative filtering algorithms search for items that have a large percentage of baskets in common. In the pizza example in Table 3.2, a 2-liter Pepsi and a large pizza Margherita share 100% of the baskets that contain either a 2-liter Pepsi or a large pizza Margherita. Even if these are the only two orders in thousands of orders, this coupling will be captured by the collaborative filtering algorithm. When a new order is placed and one of these two items is mentioned, a recommendation of the second item will automatically be generated.

Another difference between association rules and collaborative filtering is in terms of implementation. Association rule algorithms are run retrospectively on a large transactional dataset, and generate rules. The rules are then used in the decision making process for purposes of generating business rules. In contrast, collaborative filtering is run on a "live" and incrementally growing dataset, and generates real-time recommendations to the end users.

The data used in recommendation systems can be user ratings of different items, which require user input. For example, Netflix solicits movie ratings from its users in order to provide them with recommendations. User preference data may also be implicit, as in purchase or consumption of item combinations. An example is book orders on Amazon.com. When ratings are used, user-based collaborative filtering al-

gorithms search for users who rate a set of items similarly and use this group as "neighbors" to generate recommendations for a new similar user. This is the same idea as the k-NN algorithm (Section 3.2), where the predictors are ratings of items and the outcome is an item not yet rated by the user but rated by neighbors.

Another type of collaborative filtering is item-based. In this case "similarity" refers to items, so that items with similar user ratings or those purchased together are considered similar. This approach is the basis for Amazon.com's recommender system that produces recommendations of the type "Customers who bought this item also bought. . . ."

A major challenge with collaborative filtering is that most users (rows) only have values or ratings on a small number of items. The resulting users-by-items dataset therefore typically has many missing values. This "sparsity" issue poses various challenges. Because similarity is based on common items (which may be relatively rare), results can be unreliable. One approach to dealing with the many missing values is applying dimension reduction methods.

The Verdict: The personalized nature of collaborative filtering recommendations, and the possibility of implementing them in real-time, makes them desirable techniques for online or other real-time recommendation systems. We noted the sparsity issues, for which various solutions are available. Two conceptual limitations to consider are the lack of recommendations for users who are dissimilar from all other users and the reliance of similarity on combinations of behaviors but not on combinations of non-behaviors (such as the absence of certain symptom combinations). In spite of the above, collaborative filtering is considered one of the most successful recommender system techniques.

3.3 *Forecasting*

The term *forecasting* commonly refers to prediction of future values. While it is similar to supervised learning as described in Section 3.2, forecasting has a few distinguishing features. The first characteristic is the type of data. In forecasting, the outcome measurement is usually a series of values over time, called a *time series*. Examples include data on quarterly sales, average daily temperature, hourly website traffic, and annual rainfall. Second, the typical goal is to generate a forecast of

future values of the time series. For example, forecasting demand in the next four quarters or forecasting weekly stockouts.

The main assumption underlying forecasting techniques is that behaviors and patterns observed in the past will continue in the future. For example, an increasing trend is expected to increase further in the future. A seasonal pattern will recur in the future. The equivalent of this assumption in the ordinary supervised learning context is that the existing data are similar to the new data that will be scored/predicted.

While the supervised learning methods can be used for forecasting, the temporal nature of both data and goal requires modifications. Let us consider a few key modifications.

Outcome and Input Measurements

Two forecasting approaches are extrapolation and causal regression. Extrapolation techniques use early measurements of the series as input measurements for later measurements (the outcome). Supervised learning methods such as regression can then be applied to the input and output columns. When the series contains a trend and/or seasonal cycles, special input measurements are created to capture such patterns. For example, to capture a linear trend of an annual demand series, we can model the demand values (the outcome) in a linear regression with the column of year numbers as an input variable.

In addition to the input measurements based on the series' past, additional external information can be included as input variables, similar to ordinary supervised learning

Asur and Huberman [7] created a forecasting model for box-office revenue generated by a movie in its opening weekend. Their input variables are based on tweets (posts on Twitter.com) that referred to the movie prior to its release. To forecast box-office revenue for a particular movie in its opening weekend, they used the "daily tweet ratio" from each of the last seven days as seven input variables.

[7] S. Asur and B. A. Huberman. Predicting the future with social media. *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technolgy*, pages 492–499, 2010.

Training and Holdout Data

In Section 3.2 we introduced the practice of partitioning the data into training, validation and test portions in order to

evaluate the predictive accuracy of a solution on new data. In the case of time series, we cannot use random segmentation, because it will create "holes" in the series. The values in a time series are ordered chronologically and this order is of importance. Therefore, the concept of data partitioning is carried out by splitting the series into an earlier training period and a later validation period (the most recent portion of the series) that is considered the holdout period.

Data mining algorithms are trained on the training period and their performance is evaluated on the holdout period. The holdout period is the future, relative to the training period. Determining where to split the series depends on different aspects of the forecasting goal and the nature of the data, such as the forecast horizon and the types of patterns that the data exhibit. Time series partitioning is illustrated in Figure 3.8.

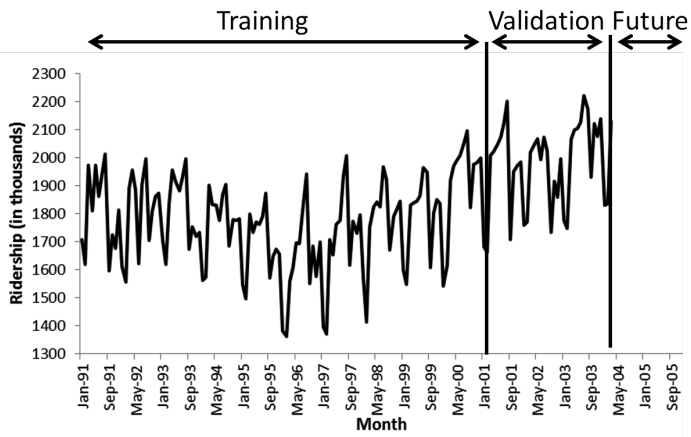


Figure 3.8: Illustration of partitioning a time series into training and holdout periods (from G. Shmueli. *Practical Time Series Forecasting: A Hands-On Guide*. CreateSpace, 2nd edition, 2011, with permission).

Further Forecasting Methods

The temporal nature of a time series usually implies correlation between neighboring values (*autocorrelation*). This also implies that the most recent values in a series carry the freshest information for forecasting future values. There are various techniques that approach the forecasting task by directly capturing this fact. The simplest are smoothing methods, such as the *moving average* and *exponential smoothing*. A moving average produces forecasts by taking the average of the most recent values of the series. Exponential smoothing forecasts are a weighted average of all past values with de-

caying weights into the past. More sophisticated smoothing methods exist for forecasting series with trends and seasonal patterns.

A different approach that is somewhat similar to regression models is based on fitting a formula that directly models the relationship between a value and past values. Methods such as *Autoregressive Integrated Moving Average (ARIMA)* models and state space models are based on this approach. The formula can be quite complicated, and automated procedures are used to find the formula's parameters that generate the best forecasts.

Forecasting Many Series

While forecasting methods have been used for decades, a new aspect of forecasting is the need to forecast a very large number of series, sometimes in real-time. Examples include retail stores and chains that require forecasts for each Stock Keeping Unit (SKU). Websites such as Google Insights for Search produce immediate forecasts for keyword popularity according to the keywords typed by users. Farecast.com (now Bing Travel, by Microsoft `www.bing.com/travel/about/howAirPredictions.do`) produces forecasts for many air travel routes, to determine the best time to purchase air tickets. Forecasting algorithms predict whether the airfare on a certain route will increase/decrease with a certain confidence level.

In this new realm, the role of automation is critical. Algorithms that can be automated and that have low memory and storage requirements are often preferred over alternatives that might produce slightly more accurate forecasts.

The Verdict: Forecasting is an important component of supervised learning, although the temporal nature of the data and the goal require some modifications to non-temporal approaches. In addition, specialized forecasting algorithms exist that take advantage of the correlation between neighboring values in the series. Today's context often requires generating forecasts for a large number of series, often in real-time. Automated forecasting algorithms are therefore at the core of many such systems.

3.4 Optimization

Optimization is an important tool with origins in operations research. Operations research is the discipline of applying analytical techniques to arrive at better decisions. The primary focus of this field is to arrive at optimal or near-optimal solutions for complex decision-making problems. Optimization and simulation (see Section 3.5) are the main two tools. Optimization is aimed at finding a combination of parameters that optimizes a function of interest. For example, in online advertising we might ask what ad location, size, and color lead to maximum click-through rate.

We illustrate the process of optimization through a business example. Consider a manufacturing company that produces different types of furniture. The company would like to determine which combination of items produced in one day results in maximum profit.

As a first step, it is important to understand the landscape we wish to optimize. We have three main components: the function to optimize, the parameters, and the constraints.

Parameters (Variables)

We are interested in determining the combination of furniture item quantities to produce in a day. In this example, we consider four types of items, and plan to produce quantities x_1, x_2, x_3 and x_4 of each item, respectively. Regarding the amount produced of each item, there may be minimal requirements, such as at least one item from each type. For the sake of simplicity, let us assume that there is no required minimal quota of furniture to be produced.

The output of the optimization procedure will include the values for x_1, x_2, x_3 and x_4 that optimize the daily profit.

Costs and Constraints

In our example, we want to capture the different manufactured items as well as their respective costs and profits. Costs include labor (man hours worked) and materials. We list the labor and material requirements in Table 3.1.

We also need to capture the cost and availability of the labor and required materials. These are given in Table 3.2.

	Labor (hrs)	Metal (lbs)	Wood (ft^3)	Price ($)
Chairs	1	1	3	79
Bookcases	3	1	4	125
Bed-frames	2	1	4	109
Desks	2	1	3	94

Table 3.1: The furniture company is able to produce four different items that each have different labor and material costs. The last column lists the selling price of each item.

	Labor (hrs)	Metal (lbs)	Wood (ft^3)
Cost ($)	14	20	11
Availability	225	117	420

Table 3.2: The cost and availability of labor, metal and wood for the production of chairs, bookcases, bed-frames and desks.

Objective Function

The goal stated by the company is maximizing daily profit. Assuming that all of the produced items can be sold, the business question to be addressed is: "How many desks, chairs, bookcases and bed-frames should the company produce per day in order to achieve maximum profit?" To rephrase, how can the company maximize profit given the above constraints?

Computing profit requires computing costs and revenues. For a given set of furniture item quantities x_1, x_2, x_3 and x_4, we can compute costs, revenues and resulting profits. The spreadsheet in Figure 3.9 illustrates this computation. Ordinary cells include numbers, while highlighted cells include formulas (for example, the formula for cell F11 is shown in the function bar).

Optimization Model

The optimization model can be represented with the optimization objective

$$\textbf{Maximize \{ Profit = Total Revenue} - \textbf{Total Cost \}}$$

with the parameters on which the decision will be made on

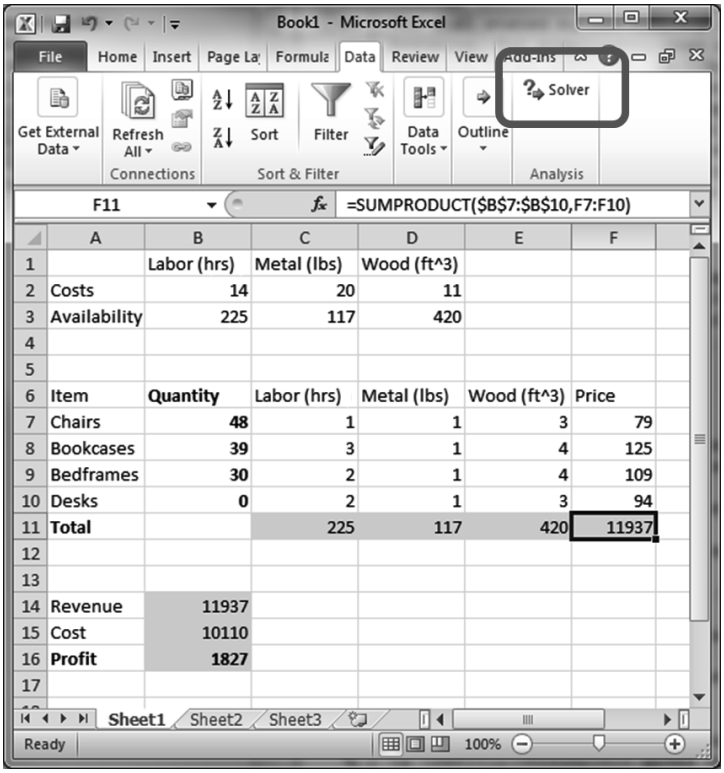
$$\textbf{\#Desks (}x_1\textbf{), \#Chairs (}x_2\textbf{), \#Bookcases (}x_3\textbf{), and \#Bed-frames (}x_4\textbf{)}$$

and the constraints (limitations) to achieve the objective

$$\textbf{Availability of labor, metal, and wood}.$$

This optimization problem can be solved using Excel's Solver add-in[8]. Given the spreadsheet in Figure 3.9, we can specify the objective function, parameters and constraints as shown in Figure 3.10, which yields the following solution

[8] For more examples of using Excel's Solver for optimization problems, see G. Shmueli. *Practical Time Series Forecasting: A Hands-On Guide*. CreateSpace, 2nd edition, 2011.



	A	B	C	D	E	F
1		Labor (hrs)	Metal (lbs)	Wood (ft^3)		
2	Costs	14	20	11		
3	Availability	225	117	420		
4						
5						
6	Item	Quantity	Labor (hrs)	Metal (lbs)	Wood (ft^3)	Price
7	Chairs	48	1	1	3	79
8	Bookcases	39	3	1	4	125
9	Bedframes	30	2	1	4	109
10	Desks	0	2	1	3	94
11	**Total**		225	117	420	11937
12						
13						
14	Revenue	11937				
15	Cost	10110				
16	**Profit**	1827				
17						

Figure 3.9: Spreadsheet with furniture example. Ordinary cells include numbers; highlighted cells include formulas. All formulas are based on the four values in the *Quantity* column.

(shown in the spreadsheet): the maximum achievable daily profit is $1827 by manufacturing 0 desks, 48 chairs, 39 bookcases and 30 bed-frames. Examining the pre-determined constraints, we find that we maximally and exactly utilize the full availability of labor at 225 hours, metal at 117 and wood at 420 with zero daily surplus.

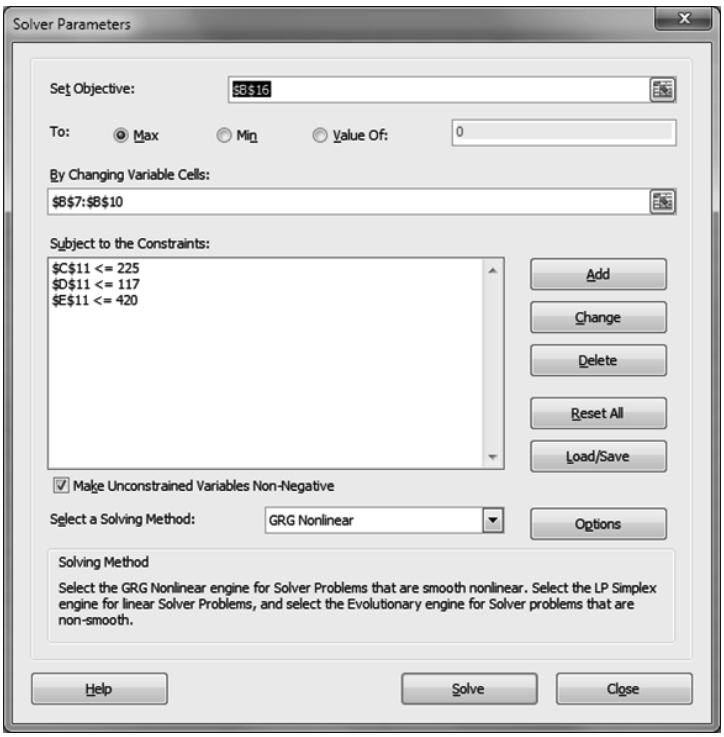


Figure 3.10: Using Excel's Solver add-in to solve the optimization problem. The optimal solution (parameters and profit) is shown in Figure 3.9 in bold.

The optimization model above, together with the information in Tables 3.1 and 3.2, can also be translated into the following optimization code[9]:

[9] The code is written and executed in SAS\OR. *Optmodel* indicates that the type of optimization solver would be automatically selected.

```
proc optmodel;
   /* declare variables */
   var desks >= 0, chairs >= 0, bookcases >= 0,
    bedframes >= 0;

   /* declare constraints */
   con Labor: 2*desks + 1*chairs + 3*bookcases
                     + 2*bedframes <= 225;
   con Metal: 1*desks + 1*chairs + 1*bookcases
                     + 1*bedframes <= 117;
```

```
con Wood: 3*desks + 3*chairs + 4*bookcases
                  + 4*bedframes <= 420;

/* declare objective */
max Profit =
     94*desks + 79*chairs + 125*bookcases + 109*bedframes
  - 14 * (2*desks + 1*chairs + 3*bookcases + 2*bedframes)
  - 20 * (1*desks + 1*chairs + 1*bookcases + 1*bedframes)
  - 11 * (3*desks + 3*chairs + 4*bookcases + 4*bedframes);

expand;
solve;
print desks chairs bookcases bedframes;
quit;
```

The SAS System

The OPTMODEL Procedure

Solution Summary	
Solver	Dual Simplex
Objective Function	Profit
Solution Status	Optimal
Objective Value	1827
Iterations	3
Primal Infeasibility	0
Dual Infeasibility	0
Bound Infeasibility	0

desks	chairs	bookcases	bedframes
0	48	39	30

Figure 3.11: SAS\OR output illustrating the chosen solver, the status of the output, the value of the optimal output as well as the variables values (number of items to produce).

Figure 3.11 gives the output of our scenario, where the system automatically detects the type of solver to be Dual Simplex.

The solution given in Figures 3.9 and 3.11 is optimal; it is not possible to do better given the current information. However, the solution might not be operationable. Assume that our manufacturer is unable to produce zero desks due to a minimum requirement of desks, or that employees are unable to work non-stop and require fixed rest periods, etc. An optimization model will provide the best possible solution (if one is available) based on the information provided. Subsequent information that needs to be taken into account (such as labor rest periods or shift changes) would need to be included in the overall formulation.

Perhaps most importantly, the optimization framework allows for examining multiple scenarios, real and potential. We can use optimization not only to identify the feasibility of current business objectives, but also to identify the outcome if changes are incorporated. For example, we can explore the maximum possible profit and quantity of manufactured furniture items if we increase/decrease the selling price, labor hours and/or the material costs. Optimization gives us the ability to identify the best possible case under varying conditions.

Optimization Methods and Algorithms

As illustrated above, the simplest case of optimization is the maximization or minimization of an objective function by systematically choosing values from within an allowed set and computing the value of the objective function. Optimization problems fall into various categories, depending on the form of the objective function and constraints.

The simplest case is a linear objective function and linear equality or inequality constraints. Our furniture example had exactly this structure: revenue was a linear function of the four item quantities and the constraints were all linear inequalities. The common optimization technique for solving such problems is *Linear Programming (LP)*.

Optimization problems are generally divided into two types of problems, convex and non-convex. Convex problems are those with a single optimal solution. The objective function has a single minimum or maximum point.

Geometrically, a function is convex if every point on a straight line joining two points on the function is also within the region. This is illustrated in Figure 3.12, where any straight line joining two points in the convex curve is within

the region, whereas for the non-convex function we can create a line external to the region by connecting two points.

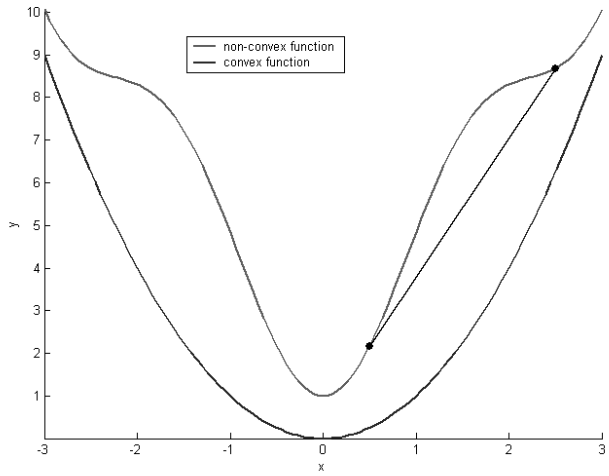


Figure 3.12: A plot of a non-convex (top line) and convex (bottom line) function. The top curve is non-convex because the black straight line segment connecting two points on the curve is below the curve ("outside the region").

In general, the difficulty of solving a problem is affected by its convexity. A variety of optimization methods exists, each designed for particular characteristics of the problem. For example, *convex programming* is used when the objective function is convex or concave (a minimization or maximization goal) and the constraint set is convex. LP is an example of convex programming. *Nonlinear programming* is needed when the objective function and/or the constraints are nonlinear.

A possible constraint is that parameter units are integers. In our furniture example, we might require the item quantities to be integers. *Integer programming* is used when some or all of the parameters are constrained to take on integer values. This is a non-convex case, and in general much more difficult than regular linear programming.

A comprehensive list of different optimization problems and methods is listed in Wikipedia's Mathematical Optimization article[10].

Many computational algorithms exist for solving optimization problems. Examples of popular algorithms include *Branch and Bound, the Expectation-Maximization (EM) Algorithm, the Genetic Algorithm*, and *the Simplex Algorithm*.[11] We note that most commercial software packages are able to automatically select the relevant technique for the defined op-

[10] en.wikipedia.org/wiki/Mathematical_optimization.

[11] For a comprehensive listing see en.wikipedia.org/wiki/Category:Optimization_algorithms_and_methods

timization problem. For a theoretical understanding of optimization methodologies see [12].

[12] Stephen Boyed and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004; R. T. Rockafellar. Lagrange multipliers and optimality. *SIAM Review*, 21(183), 1993; and T. Ui. A note on discrete convexity and local optimality. *Japan J. Indust. Appl. Math.*, 23(21), 2006.

3.5 *Simulation*

Simulation is a tool for scenario building and assessment, which originates in operations research and is used in a wide range of fields and applications. The purpose of simulation is to assess the impact of various factors on outcomes of interest. In particular, simulation is the process of imitating an operation of a process over time. We can then change various process parameters and conditions and see how they affect the process outcomes of interest.

Simulation and optimization are sometimes confused. The main difference is that in *optimization* we search for combinations of inputs that lead to some optimal outcome, whereas *simulation* imitates a prescribed process flow over time — it does not optimize the process.

Let us continue our example of the furniture company. We used optimization to find the optimal combination of furniture items produced daily in order to maximize profit, given various constraints. We can use simulation to assess how the optimal solution (daily production process) will be affected by factors not considered in the optimization. For example, what if we over-estimated the number of work hours? What if a large order comes in? Moreover, although we stated the different costs and requirements as figures (such as 1 hour for producing a chair), there is always some random fluctuation in these figures. For instance, a chair might take anywhere between 45 minutes and 1.5 hours to produce. We therefore may wish to *simulate* the furniture assembly line by capturing the steps required to assemble the different types of furniture. Such a simulation can help us evaluate the outcome under a broader context, taking into account random fluctuation. The simulation can help identify bottlenecks in the process as well as whether the process is sustainable over time.

Monte Carlo Simulation

One approach to incorporating random variability into the simulation is using Monte Carlo experiments. The idea behind this approach is to compute the relationship of interest multiple times, each time varying parameters according to

some random process. By examining the different runs, we get an idea of possible fluctuations in the outcomes of interest and their sensitivity to changes in the parameters. In our furniture example, costs of labor and material were stated as fixed figures. Suppose that the number of hours to produce each of the furniture items can vary within half an hour of the values in Table 3.1 due to causes not under our control (differences in skill level, effects of temperature on workers, etc.). How does this extra variability affect the daily profit? How much variation does it introduce into our setup?

To take into account the above-mentioned random variation in labor time, we replace the values in cells C7:C10 in the spreadsheet shown in Figure 3.9 with a formula that adheres to the type of randomness that we envisage. If we assume that building a chair can take anywhere between half an hour and 1.5 hours, and that any value within this range is equally likely, then we can use the Excel formula *=0.5+RAND()*, where the function RAND() generates a random number between 0 and 1. Figure 3.13 shows the result of a single run. We see that the RAND() functions that were inserted into the labor values lead to a slightly different daily profit. Compared to the original profit of $1,827, the profit in this screenshot is $1,918. However, this is only a single run, representing one possible outcome. In Excel, we can regenerate random numbers by pressing the F9 key. The next step is to aggregate the results from multiple runs, and to examine the variation in profit across the different runs. This is typically done by examining a chart of profit across the different runs. An example is shown in Figure 3.14, where we see the likelihood of obtaining different profit levels.

In short, simulation helps us assess the uncertainty associated with the relationship between the process parameters and the objectives. Monte Carlo simulation does this by adding random variation into the process parameters. In this example, we chose a uniform distribution (function RND()) to represent the randomness structure. However, many other probability distributions can be used instead.

Let us expand on our previous example and assume that the furniture company also provides repair and maintenance services for the items it manufactures. We would now want to know whether the current inventory holding level sufficiently adheres to the company's service agreements, which target a certain customer service level. Suppose that the company requires a 95% service level, where 95% of all spare

C7 =0.5+RAND()

	A	B	C	D	E	F
1		Labor (hrs)	Metal (lbs)	Wood (ft^3)		
2	Costs	14	20	11		
3	Availability	225	117	420		
4						
5						
6	Item	Quantity	Labor (hrs)	Metal (lbs)	Wood (ft^3)	Price
7	Chairs	48	0.72	1	3	79
8	Bookcases	39	2.94	1	4	125
9	Bedframes	30	2.30	1	4	109
10	Desks	0	2.42	1	3	94
11	**Total**		218.50	117	420	11937
12						
13						
14	Revenue	$ 11,937				
15	Cost	$ 10,019				
16	**Profit**	**$ 1,918**				
17						

Figure 3.13: Furniture example with randomness inserted into the labor time (cells C7:C10). The effect can be seen on the revenue, cost, and profit values.

parts are available at all times, while the remaining 5% require special order or are on queue for repair. We simulate the demand for repair parts, assuming that each part is completely repairable.

Let us consider two types of replacement parts: Part A (wood frame) and Part B (metal frame). Part A can be replaced with either another Part A or with a Part B. However, Part B can be replaced only by another Part B.

To evaluate the service level for our simulation, we aim to capture a number of operational considerations (as illustrated in Figure 3.15, left to right):

- Simulate the daily number of furniture items coming in for repair over time, and the respective required part (Part A or Part B).
- Assign the requested part, otherwise place the item into the queue and calculate service level.
- If requesting part A, check availability of part A or B and send the replaced part A for repair.
- If requesting part B, check availability of part B and send the replaced part B for repair.
- Update inventory holding to reflect outgoing parts.

Such a simulation will ascertain the ongoing service level when given the number of parts held in inventory (at some

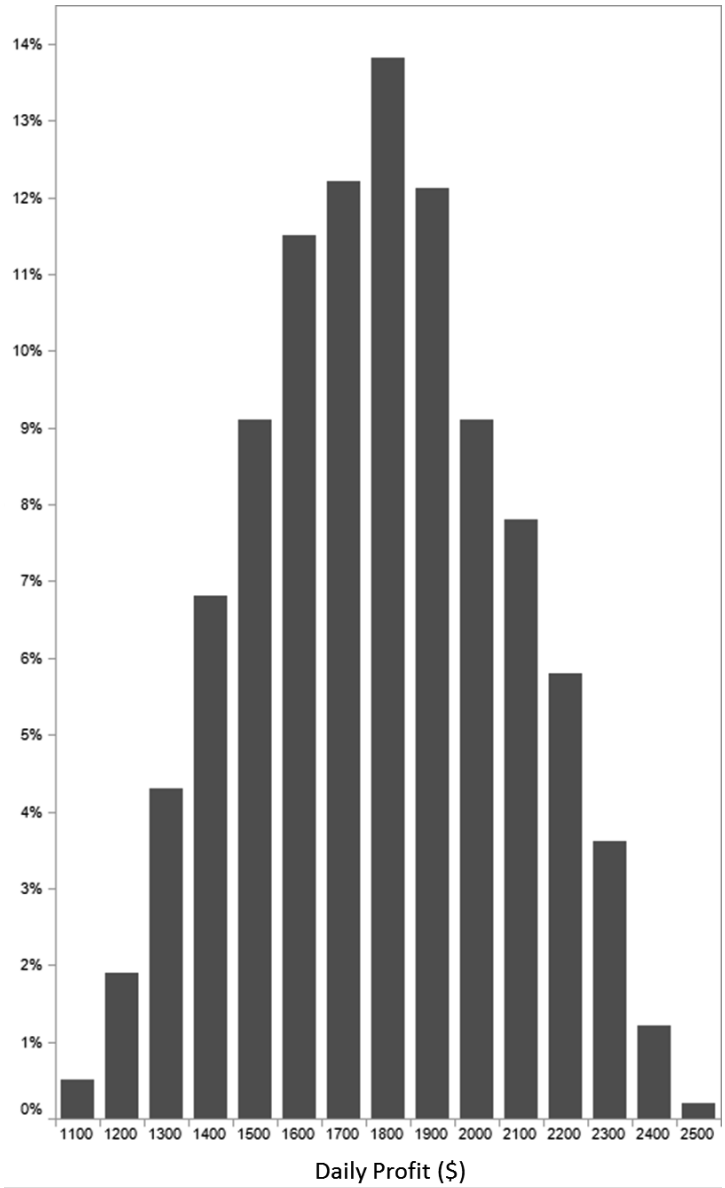


Figure 3.14: Results from 1,000 Monte Carlo runs of furniture example with randomness inserted into the labor time. The y-axis shows the percent of runs obtained for a particular profit range. The peak indicates that a profit in the range of $1,800 is most likely, but values can range between $1,100–2,500.

starting point, as the service level will change as parts are consumed and furniture is repaired), the time taken to repair each item, the service time, etc.

Figure 3.15 presents a schematic view of a recommended holistic operational solution flow. It combines forecasting, optimization and simulation in a manner that is common in practice. Given forecasts, the outcome of interest is optimized while simulating the ongoing operational variability. In this scenario, we try to capture the knowns and to quantify the unknowns. In the context of our furniture example, this means:

1. Forecast the future demand for furniture items.
2. Optimize the production mix, given constraints and forecasted demand.
3. Simulate the system given the forecasted demand, optimal holding level and domain knowledge.

Such a process will take into consideration future requirements, find the best possible configuration and simulate the solution over time.

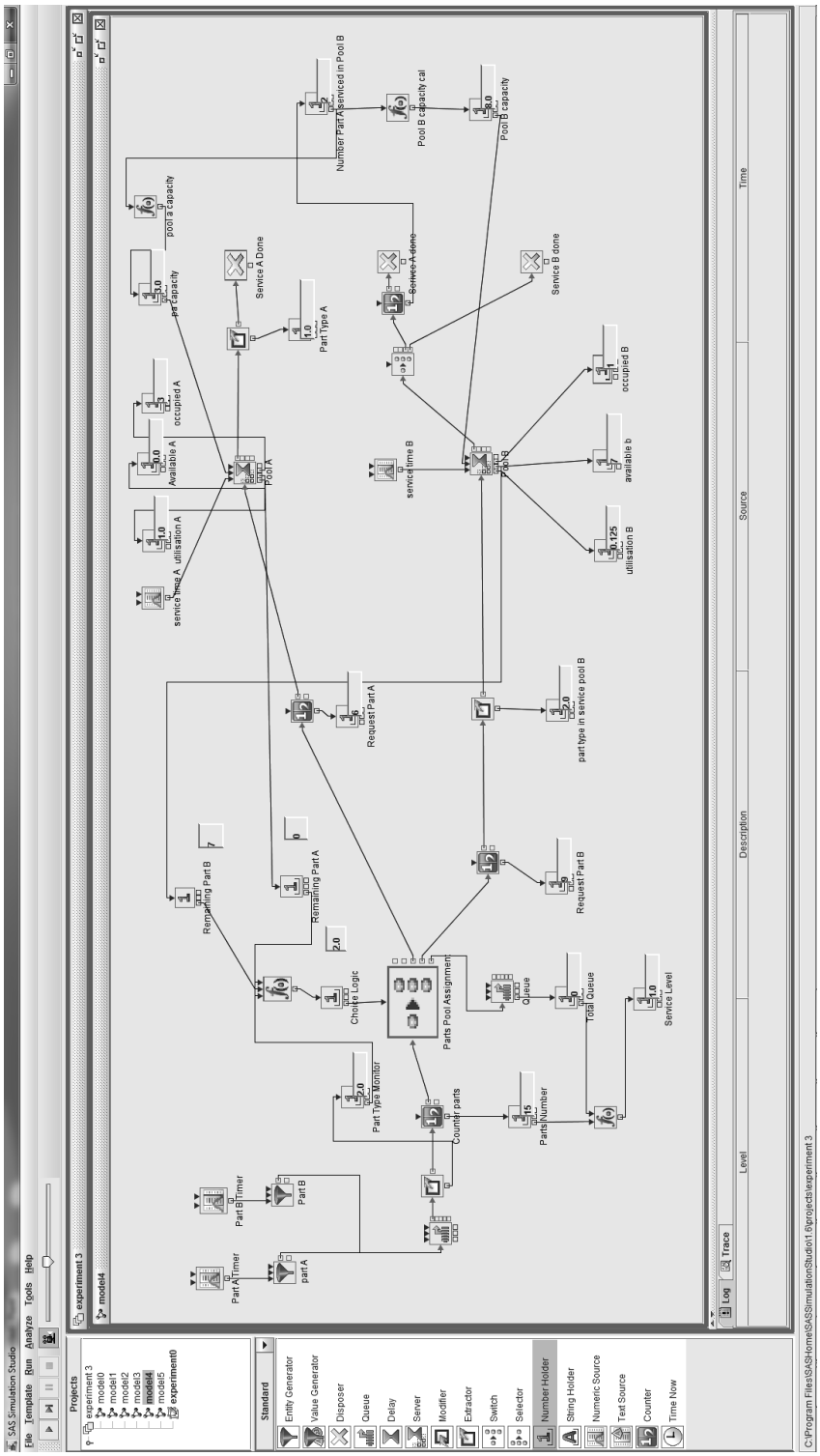


Figure 3.15: Visualization of a simulation flow of repairing furniture with two interchangeable parts.

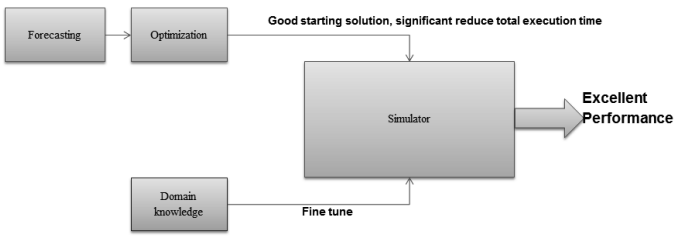


Figure 3.16: Visualisation of a holistic approach towards a complete operational system that incorporates forecasting, optimization and simulation.