$$\text{Interest rate} = 10\,\% - \frac{6 * \text{Credit Score}}{1,000}\,\%$$

Using this model we can calculate the interest rate for a specified credit score of the borrower. Linear regression is one of the simplest models to get us started in model building and more complex models are discussed later in this book. In reality, the rate calculation involves a few dozen input variables and also takes into account the nonlinear relationship between variables.

2.3.3 Evaluation of the Model

The model generated in the form of an equation is generalized and synthesized from seven training records. We can substitute the credit score in the equation and see if the model estimates the interest rate for each of the seven training records. The estimation may not be exactly the same as the values in the training records. We do not want a model to memorize and output the same values that are in the training records. The phenomenon of a model memorizing the training data is called *overfitting*, which will be explored in Chapter 4 Classification. An overfitted model just memorizes the training records and will underperform on real production data. We want the model to generalize or *learn* the relationship between credit score and interest rate. To evaluate this relationship, the validation or test data set, which was not previously used in building the model, is used for evaluation, as shown in Table 2.5.

Table 2.5 provides the three testing records where the interest rate is known; these records were not used to build the model. The actual value of the interest rate can be compared against the predicted value using the model and thus the prediction error can be calculated. As long as the error is acceptable, this model can be used for deployment. The error rate can be used to compare this model with other models developed from different algorithms like neural networks or Bayesian models, etc.

2.3.4 Ensemble Modeling

Ensemble modeling is a process where multiple diverse models are created to predict an outcome, either by using many different modeling algorithms or using different training data sets. The ensemble model then aggregates the prediction

Table 2.5 Evaluation of Test Data Set

Borrower	Credit Score (X)	Interest Rate (Y)	Model Predicted (Y)	Model Error
04	700	6.40%	6.11%	-0.29%
07	750	5.90%	5.81%	-0.09%
10	825	5.70%	5.37%	-0.33%

of each base model and results in once final prediction for the unseen data. The motivation for using ensemble models is to reduce the generalization error of the prediction. As long as the base models are diverse and *independent*, the prediction error of the model decreases when the ensemble approach is used. The approach seeks the wisdom of crowds in making a prediction. Even though the ensemble model has multiple base models within the model, it acts and performs as a single model. Most of the practical data mining solutions utilize ensemble modeling techniques. Chapter 4 Classification covers the approaches of different ensemble modeling techniques and their implementation in detail.

At the end of the modeling stage of the data mining process, we have (1) analyzed the business question, (2) sourced the data relevant to answer the question, (3) picked a data mining technique to answer the question, (4) picked a data mining algorithm and prepared the data to suit the algorithm, (5) split the data into training and test data sets, (6) built a generalized model from the training data set, and (7) validated the model against the test data set. This model now can be used to predict the target variable based on an input variable of unseen data. This answers the business question on prediction. Now, the model needs to be deployed, for example by integrating the model in the production loan approval process of an enterprise.

2.4 APPLICATION

Deployment or application is the stage at which the model becomes production ready or "live." In business applications, the results of the data mining, either the model for predictive tasks or the learning framework for association rules or clustering, need to be assimilated into the business process—usually in software applications. The model deployment stage leads to some key considerations: assessing model readiness, technical integration, response time, model maintenance, and assimilation.

2.4.1 Production Readiness

The production readiness part of the deployment determines the critical qualities required for the deployment objective. Let's consider two distinct use cases: determining whether a consumer qualifies for a loan account with a commercial leading institution and determining the groupings of customers for an enterprise.

The consumer credit approval process is a real-time endeavor. Either through a consumer-facing website or through a specialized application for frontline agents, the credit decisions and terms need to be provided in real time as soon as prospective customers provide relevant information. It is seen as a competitive advantage to provide a quick decision while also providing accurate results in the interest of customer and the company. The decision-making model

In summary, the key advantages of SVM are

- **Flexibility in application:** SVMs have been applied for activities from image processing to fraud detection to text mining.
- **Robustness:** Small changes in data does not require expensive remodeling.
- **Overfitting resistance:** The boundary of classes within data sets can be adequately described usually by only a few support vectors.

These advantages have to be balanced with the somewhat high computational costs of SVMs.

4.7 ENSEMBLE LEARNERS

In supervised data mining, the objective is to build a model that can explain the relationship between inputs and output. The model can be considered a hypothesis that can map new input data to predicted output. For a given training set, multiple hypotheses can explain the relationship with varying degrees of accuracy. While it is difficult to find the exact hypothesis from an infinite hypothesis space, we would like the modeling process to find the hypothesis that can *best* explain the relationship with least error.

Ensemble methods or learners optimize the hypothesis-finding problem by employing an array of individual prediction models and then combining them to form an aggregate hypothesis or model. These methods provide a technique for generating a better hypothesis by combining multiple hypotheses into one. Since a single hypothesis can be locally optimal or overfit a particular training set, combining multiple models can improve the accuracy by forcing a meta-hypothesis solution. It can be shown that in certain conditions this combined predictive power is better than the predictive power of individual models. Since different methods often capture different features of the solution space as part of any one model, the ensembles of models have emerged as the most important technique for many practical classification problems.

4.7.1 Wisdom of the Crowd

Ensemble models have a set of base models that accept the same inputs and predict the outcome individually. Then the outputs from all of these base models are combined, usually by voting, to form an ensemble output. This approach is similar to decision making by a committee or a board. The method of improving accuracy by drawing together the prediction of multiple models is also called *meta learning*. We see this similar decision-making methodology in higher courts of justice, corporate boards, and

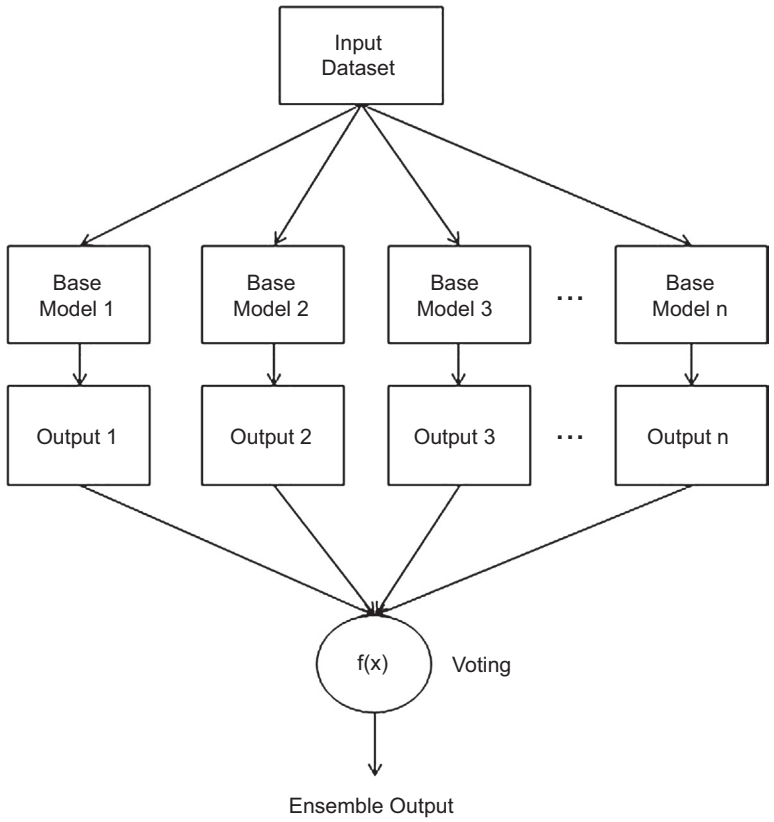

FIGURE 4.59
Ensemble model.

various committees in legislative bodies. While individual members have biases and options, the thinking here is collective decision making is better than one individual's assessment. Ensemble methods are used to improve the error rate and overcome the modeling bias of individual models. They can produce one strong learner by combining many weak learners. Figure 4.59 provides the framework of ensemble models.

The final step of aggregating the prediction is usually done by voting. The predicted class with more votes from the base learners is the output of the combined ensemble model. Base models predict the outcome with varied degrees of accuracy. Hence, we can also weight the vote by the accuracy rate of individual models, which causes base models with higher accuracy to have higher representation in the final aggregation than models with lower accuracy rate (Dietterich, 2007).

PREDICTING DROUGHT

Drought is a period of time where a region experiences far less than average water supply. With the onset of climate change, there has been an increase in frequency and duration of drought conditions in many parts of the world. Immediate drought is caused by the development of high-pressure regions, which inhibits the formation of clouds, which results in low precipitation and lower humidity. Predicting drought conditions in a region is a very challenging task. There is no clear start and end point for draught duration. There are too many variables that impact the climate patterns that lead to drought conditions. Hence, there is no strong model to predict drought well ahead of time (Predicting Drought, 2013). Predicting drought seasons in advance would provide time for regional administrations to mitigate the consequences of the drought.

Droughts involve a myriad factors including groundwater level, air stream flow, soil moisture, topology, and large-scale global weather patterns like El Nino and La Nina (Patel, 2012). With thousands of attributes and many unknown variables that influence the conditions for drought, there is no "silver bullet" massive model for predicting when drought is going to hit a region with a high degree of accuracy. What we have is many different "weak" models that use some of the thousands of attributes available, which make predictions marginally better than pure chance. These weak models may provide different drought predictions for the same region and time, based on the diverse input variables for each model. We can summarize the prediction by combining the predictions of individual models and take a vote. Ensemble models provide a systemic method to combine many weak models into one better model. Most of the data mining models deployed in production applications are ensemble models. Ensemble models greatly reduce generalization errors and improve the accuracy of the overall prediction, if certain conditions are met.

4.7.2 How it Works

Let's take an example of a hypothetical corporate boardroom with three board members. Assume that individually each board member makes wrong decisions about 20% of time. The board needs to make a yes/no decision for a major project proposal. If all board members make consistent unanimous decision every time, then the error rate of the board as a whole is 20%. But, if each board member's decisions are *independent* and if their outcomes are not correlated, the board makes an error only when more than *two board members make an error* at the same time. The board makes an error only when the majority of its members make an error. We can calculate the error rate of the board using the binomial distribution.

In binomial distribution, the probability of k successes in n independent trials each with a success rate of p is given by a probability mass function:

$$P(k) = \left(\frac{n}{k}\right) p^k (1-p)^{n-k} \tag{4.22}$$

P(Board wrong) = P($k \geq 2$) = P(2 members wrong) + P(3 members wrong)

$$P(\text{Board wrong}) = \binom{n}{3} p^k (1-p)^{n-k} + \binom{n}{2} p^k (1-p)^{n-k}$$

$$= \binom{3}{3} 0.2^3 (1-0.2)^0 + \binom{3}{2} 0.2^2 (1-0.2)^1$$

$$= 0.008 + 0.096$$

$$= 0.104$$

$$= 10.4\%$$

In this example, the error rate of the board (10.4%) is *less* than the error rate of the individuals (20%)! We therefore see the impact of collective decision making. A generic formula for calculating error rate for the ensemble is given by

$$P(\text{ensemble wrong}) = P(k \geq \text{round}(n/2)) = \sum_{k=0}^{n} \binom{n}{k} p^k (1-p)^{n-k}$$

However, some important criteria to note are:

- Each member of the ensemble should be independent.
- The individual model error rate should be less than 50% for binary classifiers.

If the error rate of the base classifier is more than 50%, its prediction power is worse than pure chance and hence, it is not a good model to begin with. Achieving the first criterion of independence amongst the base classifier is difficult. However there are a few techniques available to make base models as diverse as possible. In the board analogy, having a board with diverse and independent members makes statistical sense. Of course, they all have to make right decisions more than half the time.

Achieving the Conditions for Ensemble Modeling

We will be able to take advantage of combined decision-making power only if the base models are good to begin with. While meta learners can form a strong learner from several weak learners, those weak learners should be better than random guessing. Because all the models are developed based on the same training set, the diversity and independence condition of the model is difficult to accomplish. While complete independence of the base models cannot be achieved, we can take steps to promote independence by changing the training sets for each base model, varying the input attributes, building different classes of modeling techniques and algorithms, and changing the modeling parameters to build the base models. To achieve the diversity in the

base models, we can alter the conditions in which the base model is built. The most commonly used conditions are:

- **Different model algorithms**: The same training set can be used to build different classifiers, such as decision trees using multiple algorithms, naïve Bayesian, k-nearest neighbors, artificial neural networks, etc. The inherent characteristics of these models will be different, which yields different error rates and a diverse base model set.
- **Parameters within the models:** Changing the parameters like depth of the tree, gain ratio, and maximum split for decision tree model can produce multiple decision trees. The same training set can be used to build all the base models.
- **Changing the training record set:** Since the training data is the key contributor to the error in a model, changing the training set to build the base model is one effective method for building multiple independent base models. A training set can be divided into multiple sets and each set can be used to build one base model. However, this technique requires a sufficiently large training set and is seldom used. Instead, we can sample training data with replacement from a data set and repeat the same process for other base models.
- **Changing the attribute set:** Similar to changing the training data where a sample of records are used for the building of each base model, we can sample the attributes for each base model. This technique works if the training data have a large number of attributes.

In the next few sections, we will be reviewing specific approaches to building ensemble models based on the above techniques on promoting independence among base models. There are some limitations in using ensemble models. If different algorithms are used for the base models, they impose different restrictions on the type of input data that can be used. Hence it could create a superset of restrictions to inputs for an ensemble model.

4.7.3 How to Implement

In data mining tools, ensemble modeling operators can be found in meta learning or ensemble learning groupings. In RapidMiner, since ensemble modeling is used in the context of predicting, all the operators are located in Modeling > Classification and Regression > Meta Modeling. The process of building ensemble models is very similar to that of building any classification models like decision tress or neural networks. Please refer to previous classification algorithms for steps to develop individual classification processes and models in RapidMiner. There are a few options to choose for implementing meta modeling in RapidMiner. In the next few pages we will review the implementation of ensemble modeling with simple voting and a couple of other techniques to make the base models independent by altering examples for the training set.

Ensemble by Voting

Implementing an ensemble classifier starts with building a simple base classification process. For this example, we can build a decision tree process with the Iris data set as shown in Section 4.1 Decision Trees. The standard decision tree process involves data retrieval and a decision tree model, followed by applying the model to an unseen test data set sourced from the Iris data set and using a performance evaluation operator. To make it an ensemble model, the *Decision Tree* operator has to be replaced with the *Vote* operator from the meta learning folder. All other operators will remain the same. The ensemble process will look similar to the process shown in Figure 4.60.

The *Vote* operator is an ensemble learner that houses multiple base models inside the *inner subprocess* (Mierswa et al., 2006). The model output from the vote process behaves like any other classification model and it can be applied in any scenario where a decision tree can be used. In the apply model phase, the predicted classes are tallied up amongst all the base classifiers and the class with highest number of votes is the predicted class for the ensemble model.

On double-clicking the nested *Vote* meta modeling operator, we can add multiple base classification models inside the *Vote* operator. All these models accept the same training set and provide an individual base model as output. In this example we have added three models: decision tree, k-NN and naïve Bayes. Figure 4.61 shows the inner subprocess of the *Vote* meta modeling operator. The act of tallying

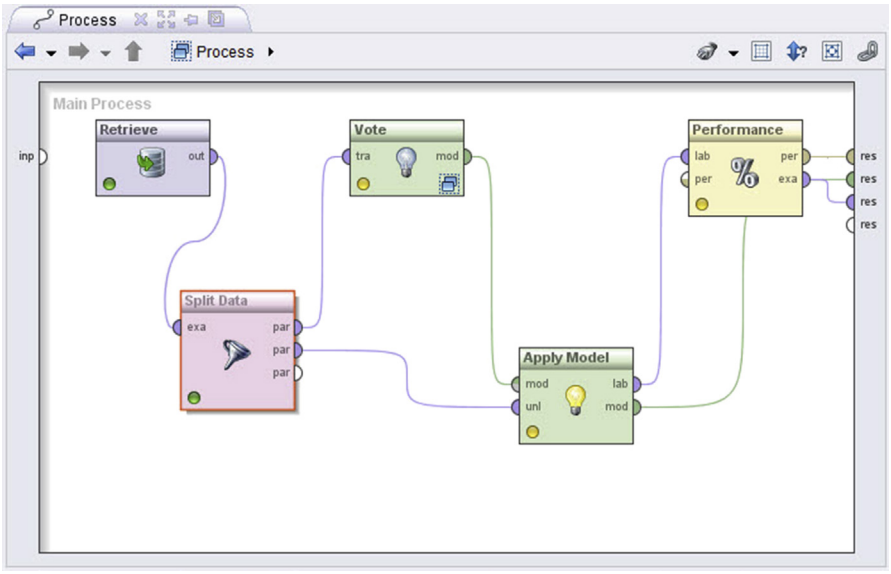

FIGURE 4.60

Data mining process using ensemble model.

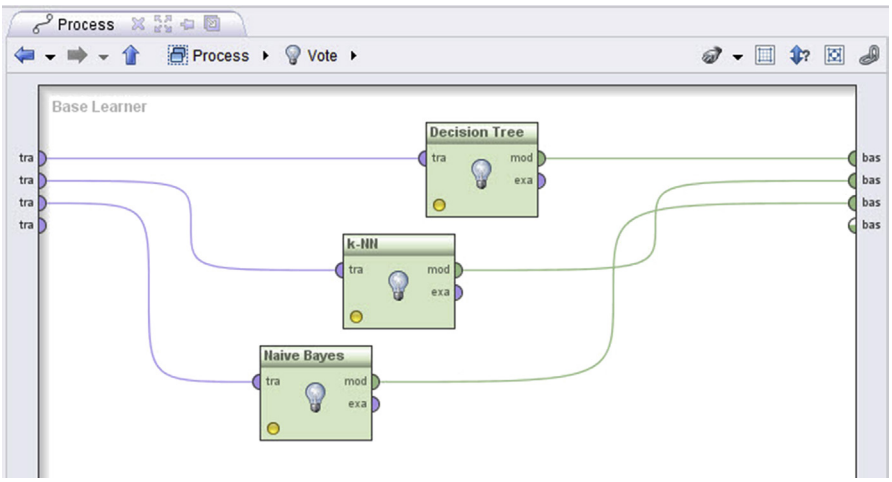

FIGURE 4.61
Subprocess inside the Vote operator.

all the predictions of these base learners and providing the majority prediction is the job of the meta model—the *Vote* modeling operator. This is the aggregation step in ensemble modeling and in RapidMiner it is called a stacking model. A stacking model is built into the *Vote* operator and is not visible on the screen.

The ensemble process with the *Vote* meta model can be saved and executed. Once the process is executed, the output panel of the performance vector is no different than a normal performance vector. Since this process has a meta model, the model panel in the results window exhibits new information, as shown in Figure 4.62. The model window shows all the individual base models and one stacking model, which is the combination of all the base models. The *Vote* meta model is simple to use wherever an individual base model could have been used independently. The limitation of the model is that all the base learners use the same training data set and different base models impose restrictions on what data types they can accept.

Bootstrap Aggregating or Bagging

Bagging is a technique where base models are developed by changing the training set for every base model. In a given training set T of n records, m training sets are developed each with n records, by sampling with replacement. Each training set T_1, T_2, T_3, …, T_m will have the same record count of n as the original training set T. Because they are sampled with replacement, they can contain duplicate records. This is called *bootstrapping*. Each sampled training set is then used for a base model preparation. Through bootstrapping, we have a set of m base models and the prediction of each model is aggregated for an ensemble model. This combination of bootstrapping and aggregating is called *bagging*.

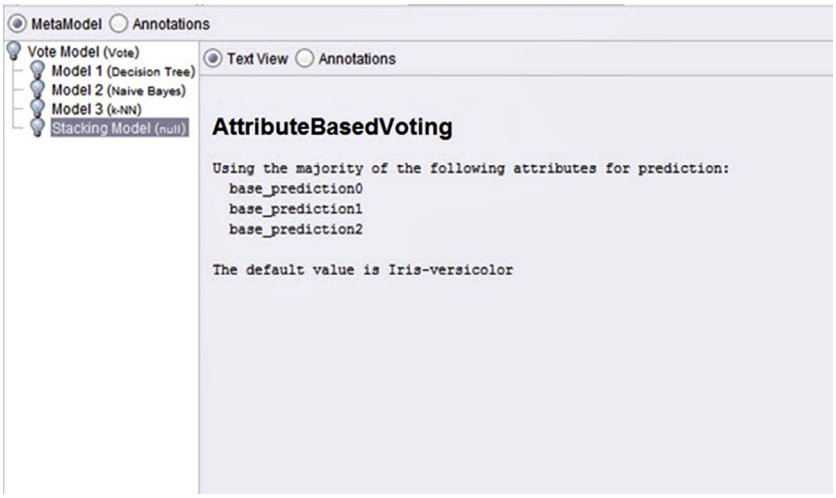

FIGURE 4.62
Output of ensemble model based on voting.

On average, each base training set T_i contains about 63% unique training records as compared to the original training set T. Sampling with replacement of n records contains $1 - (1 - 1/n)^n$ unique records. When n is sufficiently large we get $1 - 1/e = 63.2\%$ unique records on average. The rest of the data contains duplicates from already sampled data. The process of bagging improves the stability of unstable models. Unstable models like decision trees and neural network are highly dependent even on slight changes in the training data. Because a bagging ensemble combines multiple hypotheses of the same data, the new aggregate hypothesis helps neutralize these training data variations.

Implementation

The *Bagging* operator is available in the meta learning folder: Modeling > Classification and Regression > Meta modeling > Bagging. Like the *Vote* meta operator, *Bagging* is a nested operator with an inner subprocess. Unlike the vote process, bagging has only one model in the inner subprocess. Multiple base models are generated by changing the training data set. The *Bagging* operator has two parameters.

- **Sample ratio:** Indicates the fraction of records used for training.
- **Iterations (m):** Number of base modes that need to be generated.

Figure 4.63 shows the RapidMiner process for the *Bagging* operator. Figure 4.64 shows the inner subprocess for the *Bagging* operator with one model specification. Internally multiple base models are generated based on iterations (m) configured in the Bagging parameter. In this example, we are using a decision tree model for the inner subprocess.

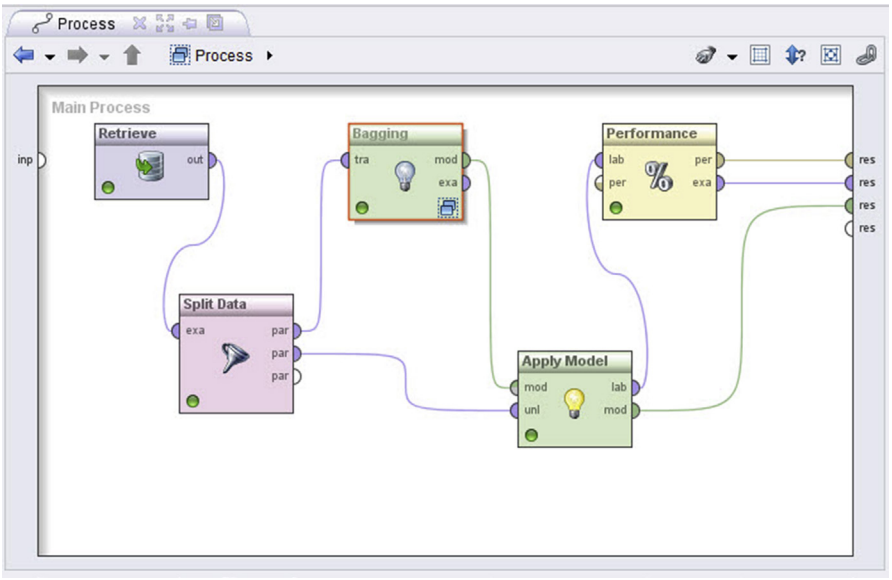

FIGURE 4.63
Ensemble process using bagging.

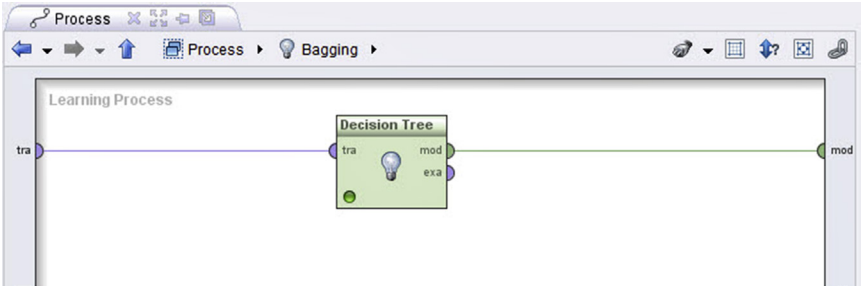

FIGURE 4.64
Bagging subprocess.

The RapidMiner process for bagging can be saved and executed. Similar to the *Vote* meta model, the *Bagging* meta model acts as one model with multiple base models inside. The results window shows the practiced example set, performance vector, and bagging model description. In the results window, we can examine all *m* (in this case 10) models that are developed based on *m* iterations of the training set. The base model results are aggregated using simple voting. Bagging is particularly useful when there is anomaly in the training data set that impacts the individual model significantly. Bagging provides a useful framework where the same data mining algorithm is used for all base

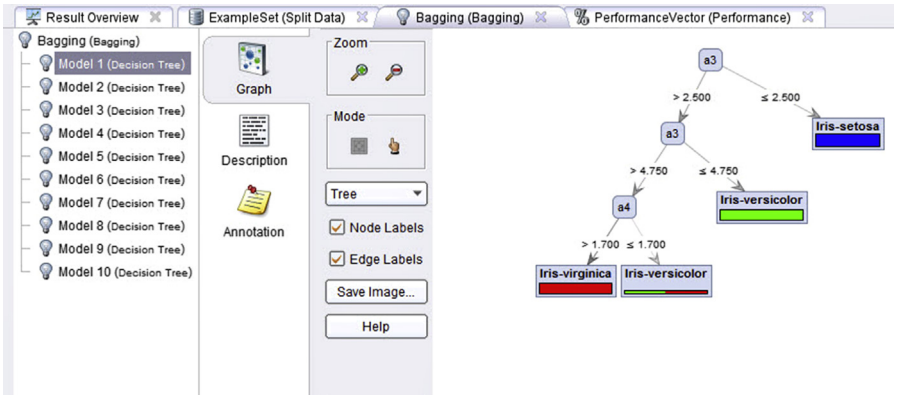

FIGURE 4.65
Output of bagging models.

learners. However, each base model differs because the training data used by the base learners are different. Since each base model explores a different solution space, the performance of the ensemble model would be better than that of base models. Figure 4.65 shows the model output of *Bagging* meta model with constituent decision trees.

Boosting

Boosting offers another approach to building an ensemble model by manipulating training data similar to bagging. As with bagging, it provides a solution to combine many weak learners into one strong learner, by minimizing bias or variance due to training records. Unlike bagging, boosting trains the base models in sequence one by one and assigns weights for all training records. The boosting process concentrates on the training records that are hard to classify and overrepresents them in the training set for the next iteration.

The boosting model is built by an iterative and sequential process where a base model is built and tested with all of the training data, and based on the outcome, the next base model is developed. To start with, all training records have equal weight. The weight of the record is used for the sampling distribution for selection with replacement. A training sample is selected based on the weights and used for model building. Then the model is used for testing with the whole training set. Incorrectly classified records are assigned a higher weight and correctly classified records are assigned a low weight, so hard-to-classify records have a higher propensity of selection for the next round. The training sample for the next round will be most likely filled with incorrectly classified records from the previous round. Hence the next model will focus on the hard-to-classify data space.

Boosting assigns the weight for each training record and has to adaptively change the weight based on difficulty of classification. This results in an ensemble of base learners specialized in classifying both easy-to-classify and hard-to-classify records. When applying the model, all base learners are combined through a simple voting aggregation.

AdaBoost

AdaBoost is one of the most popular implementations of the boosting ensemble approach. It is adaptive because it assigns weights for base models (α) based on the accuracy of the model, and changes weights of the training records (w) based on the accuracy of the prediction. Here is the framework of the AdaBoost ensemble model with m base classifiers and n training records ((x_1,y_1), (x_2,y_2), …, (x_n,y_n)). Following are the steps involved in AdaBoost:

1. Each training record is assigned an uniform weight $w_i = 1/n$.
2. Training records are sampled and the first base classifier $b_k(x)$ is built.
3. The error rate for the base classifier can be calculated by Equation 4.23:

$$e_k = \sum_{k=1}^{n} w_i * I(b_k(x_i) \neq y_i) \tag{4.23}$$

where I(x) = 1 when the prediction is right and 0 when the prediction is incorrect.

4. The weight of the classifier can be calculated as $\alpha_k = \ln(1 - e_k)/e_k$. If the model has a low error rate, then the weight of the classifier is high and vice versa.
5. Next, the weights of all training records are updated by

$$w_{k+1}(i+1) = w_k(i) * e^{(\alpha_k F(bk(xi) \neq yi))}$$

where F(x) = –1 if the prediction is right and F(x) = 1 if the prediction is wrong.

Hence, the AdaBoost model updates the weights based on the prediction and the error rate of the base classifier. If the error rate is more than 50%, the record weight is not updated and reverted back to the next round.

AdaBoost Model in RapidMiner

The *AdaBoost* operator is available in the meta learning folder: Modeling > Classification and Regression > Meta modeling > AdaBoost. The operator functions similar to Bagging and has an inner subprocess. The number of iterations or base models is a configurable parameter for the *AdaBoost* operator. Figure 4.66 shows the AdaBoost data mining process. This example uses the Iris data set with the *Split Data* operator for generating training and test data sets. The output of the AdaBoost model is applied to the test set and the performance is evaluated by the *Performance* operator.

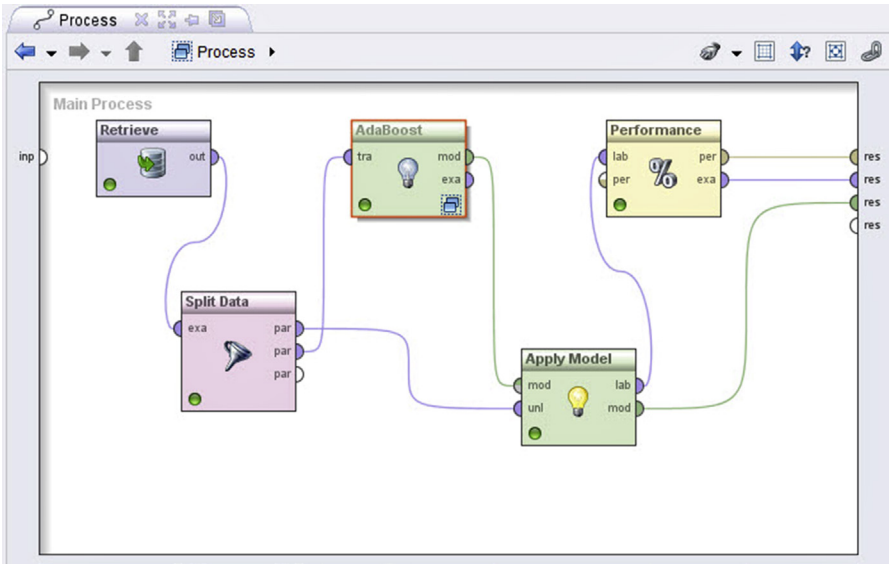

FIGURE 4.66
Data mining process using AdaBoost.

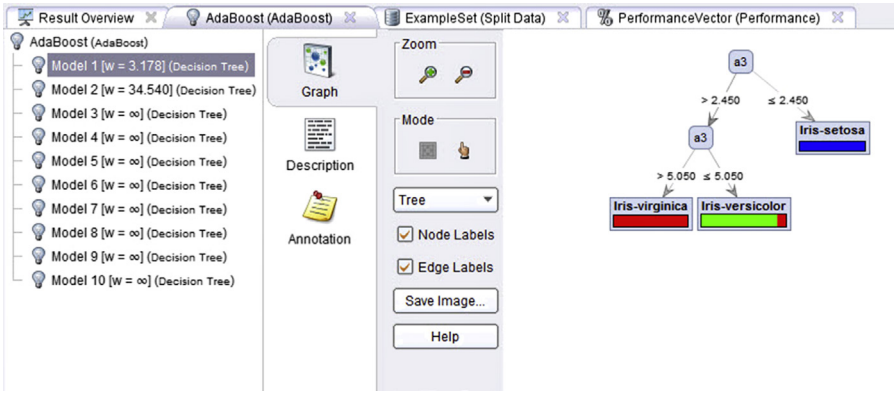

FIGURE 4.67
Output of AdaBoost model.

The number of iterations used in the AdaBoost is three, which is specified in the parameter. In the inner process, the model type can be specified. In this example the decision tree model is used. The completed RapidMiner process is saved and executed. The result window has the output ensemble model, base models, and the predicted records. The model window shows the decision trees for the base classifiers. Figure 4.67 shows the result output for the AdaBoost model.

Random Forest

Recall that in the bagging technique, for every iteration, a sample of training records is considered for building the model. The random forest technique uses a similar concept to the one used in bagging. When deciding on splitting each node in a decision tree, the random forest only considers a random subset of all the attributes in the training set. To reduce the generalization error, the algorithm is randomized in two levels, training record selection and attribute selection, in the inner working of each base classifier. The random forests concept was first put forward by Leo Breiman and Adele Cutler (Breiman, 2001).

In general, the model works using the following steps. If there are n training records with m attributes, and let k be the number of trees in the forest; then for each tree:

1. An n random sample is selected with replacement. This step is similar to bagging.
2. A number D is selected, where D << m. D determines the number of attributes to be considered for node splitting.
3. A decision tree is started. For each node, instead of considering all m attributes for the best split, a random number D attributes are considered. This step is repeated for every node.
4. As in any ensemble, the greater the diversity of the base trees, the lower the error of the ensemble.

Once all the trees in the forest are built, for every new record, all the trees predict a class and vote for the class with equal weights. The most predicted class by the base trees is the prediction of the forest (Gashler et al., 2008).

Implementation

The *Random Forest* operator is available in Modeling > Classification and Regression > Tree Induction > Random Forest. It works similarly to the other ensemble models where the user needs to specify the number of base trees to be built. Since the inner base model is always a decision tree, there is no explicit inner subprocess specification. Bagging or boosting ensemble models require explicit inner subprocess specification. All the tree-specific parameters like leaf size, depth, and split criterion can be specified in the *Random Forest* operator. The key parameter that specifies the number of base trees is *Number of Trees*. Figure 4.68 shows the RapidMiner process with the Iris data set, the *Random Forest* modeling operator, and the *Apply Model* operator. For this example, the number of base trees is specified as 10. The process looks and functions similarly to a simple decision tree classifier.

Once the process is executed, the results window shows the model, predicted output, and performance vector. Similar to other meta model output, the

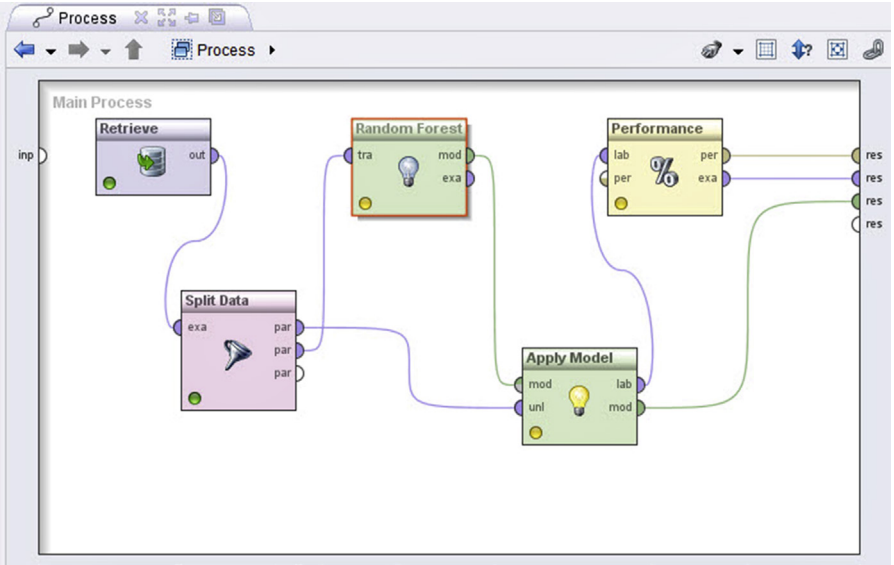

FIGURE 4.68
Data mining process using the Random Forest operator.

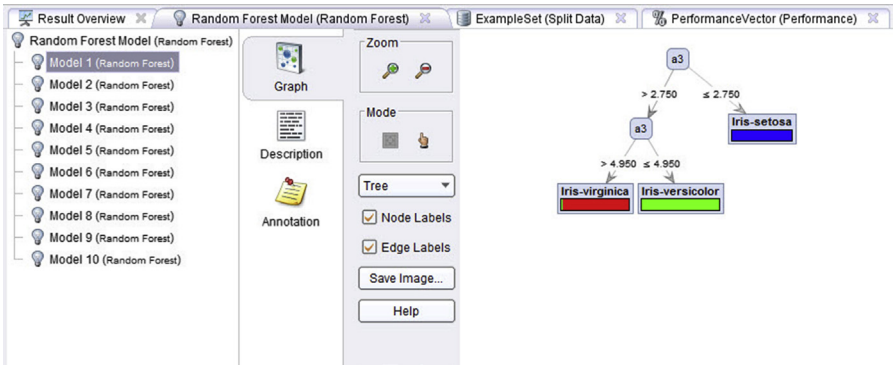

FIGURE 4.69
Output of Random Forest models.

Random Forest model shows the trees for all base classifiers. Figure 4.69 shows the model output for the *Random Forest* operator. Notice that the nodes are different in each tree. Since the attribute selection for each node is randomized, each base tree is different. Thus the Random Forest models strive to reduce the generalization error of the decision tree model. The Random Forest models are very useful as baseline ensemble models for comparative purposes.

4.7.4 Conclusion

Most of the data mining models developed for production applications are built on ensemble models. They are used in wide range of applications, including political forecasting (Montgomery et al., 2012), weather pattern modeling, media recommendation, web page ranking (Baradaran Hashemi et al., 2010), etc. Since many algorithms approach the problem of modeling the relationship between input and output differently, it makes sense to aggregate the prediction of a diverse set of approaches. Ensemble modeling reduces the generalization error that arises due to overfitting the training data set. The four ensemble techniques discussed provide fundamental methods of developing a cohort of base models by choosing different algorithms, changing parameters, changing training records, sampling, and changing attributes. All these techniques can be combined in one ensemble model. There is no one approach for ensemble modeling; all the techniques discussed in this chapter were proven to perform better than base models as long as they are diverse (Polikar, 2006). The wisdom of crowds makes sense in data mining as long as "group thinking" is controlled by promoting independence amongst base models.

REFERENCES

Altman, N. S. (1992). An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician*, *46*(3), 175–185.

Baradaran Hashemi, H., Yazdani, N., Shakery, A., & Pakdaman Naeini, M. (2010). Application of ensemble models in web ranking. *2010 5th International Symposium on Telecommunications*, 726–731. http://dx.doi.org/10.1109/ISTEL.2010.5734118.

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*, 5–32.

Brieman, L. F. (1984). *Classification and Regression Trees*. Chapman and Hall.

Cohen, W. W. (1995). Fast Effective Rule Induction. *Machine Learning: Proceedings of the Twelfth International Conference*.

Cortes, C. A. (1995). Support Vector Networks. *Machine Learning*, 273–297.

Cover, T. A. (1991). Entropy, Relative Information, and Mutual Information. In T. A. Cover (Ed.), *Elements of Information Theory* (pp. 12–49). John Wiley and Sons.

Dietterich, T. G. (2007). *Ensemble Methods in Machine Learning*. Retrieved from http://www.eecs.wsu.edu/~holder/courses/CptS570/fall07/papers/Dietterich00.pdf.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Human Genetics*, *7*, 179–188. http://dx.doi.org/10.1111/j.1469-1809.1936.tb02137.x.

Fletcher, R. (1987). *Practical Methods of Optimization*. New York: John Wiley.

Gashler, M., Giraud-Carrier, C., & Martinez, T. (2008). Decision Tree Ensemble: Small Heterogeneous Is Better Than Large Homogeneous. *2008 Seventh International Conference on Machine Learning and Applications*, 900–905. http://dx.doi.org/10.1109/ICMLA.2008.154.

Grabusts, P. (2011). The Choice of Metrics for Clustering Algorithms. *Proceedings of the 8th International Scientific and Practical Conference*, *II*(1), 70–76.

Haapanen, R., Lehtinen, K., Miettinen, J., Bauer, M. E., & Ek, A. R. (2001). Progress in adapting k-NN methods for forest mapping and estimation using the new annual Forest Inventory and Analysis data. In Third Annual Forest Inventory and Analysis Symposium (p. 87).