CHAPTER

Other Adaptations

9

Vineeth N. Balasubramanian[*], Prasanth Lade[†], Hemanth Venkateswara[†], Evgueni Smirnov[‡] and Sethuraman Panchanathan[†]

[*]*Department of Computer Science and Engineering, Indian Institute of Technology, Hyderabad, India*

[†]*Center for Cognitive Ubiquitous Computing, Arizona State University, AZ, USA*

[‡]*Department of Knowledge Engineering, Maastricht University, Maastricht, The Netherlands*

CHAPTER OUTLINE HEAD

9.1 Introduction

The previous chapters demonstrated how the Conformal Predictions (CP) framework has been adapted to traditional machine learning problems including active learning, feature selection, anomaly detection, change detection, model selection, and quality estimation. In this last chapter of the Adaptations section, we describe three other extensions of the CP framework, each of which is nontraditional in its own way. The task of obtaining a reliability value for the classification of a data instance has been the focus of a number of studies [16,60,85,157,315,327,365,391].

Conformal Prediction for Reliable Machine Learning. http://dx.doi.org/10.1016/B978-0-12-398537-8.00009-2

In Sections 9.2 and 9.3, we describe two methods that use the idea of a meta classifier to associate reliability values with output predictions from a base classifier. In particular, in Section 9.2, we describe the Metaconformal Predictors proposed by Smirnov et al. [328], where a base classifier is combined with a metaclassifier that is trained on metadata generated from the data instances and the classification results of the base classifier to associate reliability values on the classification of data instances. In Section 9.3, we describe the Single-Stacking Conformal Predictors proposed by Smirnov et al. in [329] where an ensemble classifier consisting of the base classifier and the meta classifier is constructed to compute reliability values on the classification outputs. The difference between the metaconformal and the single-stacking approaches is the manner in which the metadata are constructed and the way in which the reliability values are estimated.

In Section 9.4, we describe the application of conformal predictors to online time series analysis as proposed by Dashevskiy and Luo [70]. As mentioned earlier, the emphvalidity property of the CP framework makes it an attractive prediction tool for real-world applications involving machine learning algorithms (see Chapter 1). However, this property relies on the exchangeability assumption, which is not generally associated with time series data. Dashevskiy and Luo [70] proposed different methods to transform time series data in order to apply the CP framework to derive conformal prediction intervals using regression models. We briefly describe the overall idea, while the details of this methodology have been presented in Chapter 12.

9.2 Metaconformal Predictors

Smirnov et al. [328] noted that there are settings when it may not be possible to define a suitable nonconformity measure for a classifier (for instance, when the algorithmic details of a classifier are not known such as when a human expert is the classifier). For such cases, Smirnov proposed the Metaconformal Predictor. We begin our discussion with a description of classifier performance metrics, as in [328].

9.2.1 Classifier Performance Metrics

In order to construct reliable classifiers, metrics are needed to compare classifier performance. In this section, some of the standard metrics that are used to measure classifier performance are outlined. Given a binary classifier $h \in \mathbf{H}$, and an example space of test instances, a confusion matrix is constructed as defined in Figure 9.1. The matrix gives a count of the classified test instances based on the hypothesized class labels and the real class labels. These counts are true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). The confusion matrix is extended to include counts for unclassified instances in case of a reliable classifier. It has entries for unclassified positives (UP) and unclassified negatives (UN). These counts are used to derive metrics to measure classifier performance. The basic metrics are: true positive rate (TPr), false positive rate (FPr), true negative rate (TNr), and

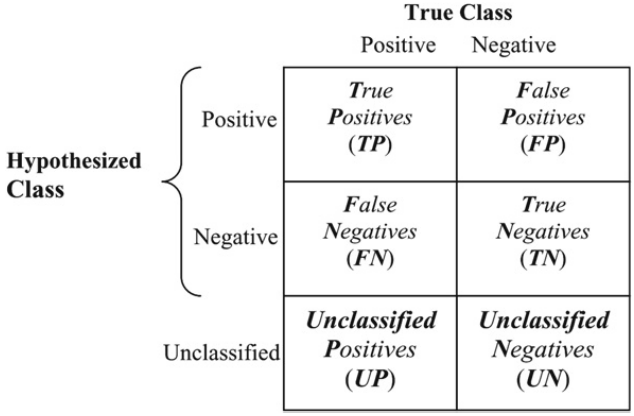

FIGURE 9.1

Confusion matrix for a binary classifier (as in [328]).

false negative rate (FNr).

$$TPr = \frac{TP}{TP + FN} \quad FPr = \frac{FP}{FP + TN}$$
$$FNr = \frac{FN}{TP + FN} \quad TNr = \frac{TN}{FP + TN}$$

The *accuracy rate* A, and *precision* P, are defined for the positive class as:

$$A = \frac{TP + TN}{TP + FP + FN + TN} \quad P = \frac{TP}{TP + FP} \tag{9.1}$$

In addition, the *rejection rate* R is defined as follow. The *coverage rate* is given as one minus rejection rate.

$$R = \frac{UP + UN}{TP + FP + FN + TN + UP + UN} \tag{9.2}$$

9.2.2 Metaclassifiers and Metaconformal Predictors

Given the best classifier $h \in \mathbf{H}$ for a problem, p-values need to be calculated for every test instance to estimate the reliability of the label assigned to the test instance. Smirnov et al. [328] pointed out the need for a metaclassifier, when a classifier h, such as a human expert or a decision rule-based system, may not be sufficient to obtain p-values using the conformal prediction framework. In such cases, a metaclassifier $m \in \mathbf{M}$ (where $\mathbf{M}$ is a space of classifiers conducive to use with the conformal prediction framework in our case) is trained, and estimates the correctness of each instance classification of h. The p-values for the test instances obtained using m can be considered as the p-values by h. The combined classifier is denoted as $h : m$. The base classifier is trained using the training data and the metaclassifier is trained using

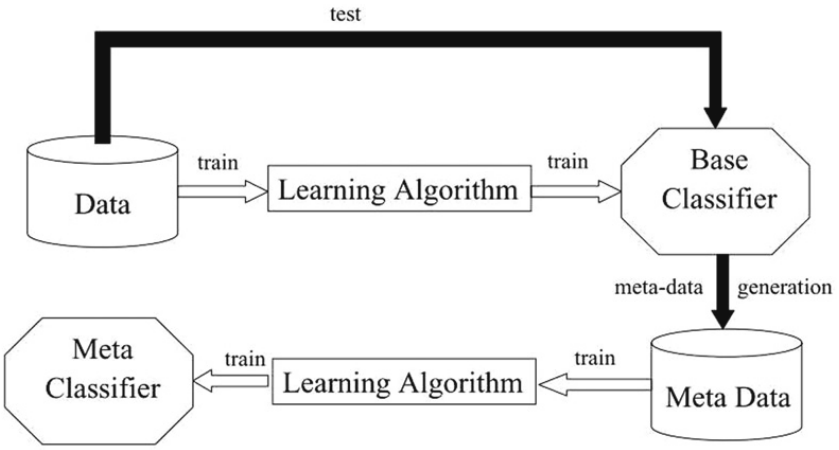

FIGURE 9.2

Combined Classifier($h : m$) which consists of the Base Classifier(h) and Metaclassifier(m).

metadata. Figure 9.2 depicts a combined classifier. Different metaclassifiers differ in the way the metadata is generated.

The base classifier h is trained on data $\mathbf{Z}$. The metaclassifier m is trained on metadata $\mathbf{Z}'$ that is generated from $\mathbf{Z}$ in the following manner. The metadata $\mathbf{Z}'$ is defined on metainstance space $\mathbf{X}'$ and metalabel space $\mathbf{Y}'$. While $\mathbf{X}'$ coincides with $\mathbf{X}$, $\mathbf{Y}'$ consists of two class labels: a "positive metaclass" that indicates reliable classification and a "negative metaclass" that indicates unreliable classification. k-fold cross validation is deployed to estimate $\mathbf{Z}'$ from $\mathbf{Z}$ [86], as described later.

The data $\mathbf{Z}$ is divided into k equally sized folds $\mathbf{F}_i, i \in [1, k]$. For every $\mathbf{F}_j$, $j \in [1, k]$, all the folds $\mathbf{F}_i$ with $i \neq j$ are combined into $\mathbf{Z}_j$. The base classifier h is trained on $\mathbf{Z}_j$ and tested with $\mathbf{F}_j$. The metadata is obtained using the test instances in $\mathbf{F}_j$. If an instance $(x_i, y_i) \in \mathbf{F}_j$ was classified correctly, an instance (x_i, y_i') is created to be placed in $\mathbf{Z}'$ where the value of y' is "positive metaclass." If on the other hand (x_i, y_i) was incorrectly classified by the base classifier, y' is "negative metaclass." In this way, all the elements in fold $\mathbf{F}_j$ are placed into $\mathbf{Z}'$. This process is repeated for all the folds $j \in [1, k]$. Once the metadata is complete, the metaclassifier m is trained using the data $\mathbf{Z}'$. The metaclassifier is usually a nonlinear classifier that is adaptable to the CP framework because the metadata is generally not linearly separable.

The combined classifier is denoted as $h : m$. This classifier assigns to an instance $x \in \mathbf{X}$ a class label $y \in \mathbf{Y}$ predicted by the classifier h, if m decides that the classification is reliable (i.e., m predicts a "positive metaclass"); else, the instance x is left unclassified. Hence, it is possible that not all instances $x \in \mathbf{X}$ are classified. The rejection rate is estimated for the combined classifier $h : m$ as $R_{h:m}$, and is equal to the proportion of instances to which m assigns "negative metaclass" (as shown by Smirnov et al. in [328]).

$$\frac{TN_m + FN_m}{TP_m + FP_m + FN_m + TN_m}.$$

The authors also showed that the accuracy $A_{h:m}$ of the combined classifier $h : m$ is equal to the precision P_m of the metaclassifier, measured with respect to the "positive metaclass" (replicated in Theorem 9.1).

Theorem 9.1. *Given a base classifier h, a metaclassifier m, and a combined classifier $h : m$, the accuracy rate $A_{h:m}$ of the classifier $h : m$ equals the precision rate P_m of the classifier m.*

Proof. Given the combined classifier $h : m$ it follows that:

$$TP_{h:m} + TN_{h:m} = (TP_h + TN_h)\frac{TP_m}{TP_m + FN_m} \tag{9.3}$$

$$FP_{h:m} + FN_{h:m} = (FP_h + FN_h)\frac{FP_m}{FP_m + TN_m} \tag{9.4}$$

From the confusion matrices of the base classifier h and the metaclassifier m, we have:

$$TP_h + TN_h = TP_m + FN_m \tag{9.5}$$

$$FP_h + FN_h = FP_m + TN_m \tag{9.6}$$

Substituting for $TP_h + TN_h$ in Eq. (9.3) and $FP_h + FN_h$ in Eq. (9.4), we get:

$$TP_{h:m} + TN_{h:m} = TP_m \tag{9.7}$$

$$FP_{h:m} + FN_{h:m} = FP_m \tag{9.8}$$

The values for $TP_{h:m} + TN_{h:m}$ and $FP_{h:m} + FN_{h:m}$ as in Eqs. (9.7) and (9.8) are substituted in the formula for accuracy $A_{h:m}$ given in (9.1) to get:

$$A_{h:m} = \frac{TP_{h:m} + TN_{h:m}}{TP_{h:m} + TN_{h:m} + FP_{h:m} + FN_{h:m}} \tag{9.9}$$

$$= \frac{TP_m}{TP_m + FP_m} \tag{9.10}$$

The last expression is the precision rate P_m for the metaclassifier m for the "positive metaclass." It can therefore be concluded that $A_{h:m} = P_m$. □

In Theorem 9.1 it was observed that the accuracy of the base classifier does not influence the accuracy of the combined classifier $h : m$. The accuracy of the combined classifier $h : m$ is equal to the precision of the metaclassifier m. To increase the accuracy of the combined classifier, P_m needs to be maximized.

Interpreting the p-values

Unlike in a standard conformal prediction setting, the p-values of the combined classifier $h : m$ need to be interpreted. It has been assumed that the base classifier h is not based on the conformal prediction framework and is therefore not capable of providing p-values as output for instance classifications. Example classifiers of h are

human experts, decision rules, and such [141,231]. On the other hand, the metaclassifier is conducive to use with the conformal prediction framework and is capable of providing p-values as outputs for the "positive metaclass" and the "negative metaclass." Example classifiers include nearest-neighbor classifiers [280], Support Vector Machines [297], and other classifiers listed in earlier chapters. Given this setting, an instance $x \in \mathbf{X}$ is classified by the combined classifier $h : m$ as follows.

The base classifier h assigns a label $y \in \mathbf{Y}$ to the instance x. The meta classifier m acts upon the instance x and estimates the p-value p_p for the "positive metaclass" and the p-value p_n for the "negative meta class." The "positive metaclass" indicates that the assigned label y is correct and the "negative metaclass" indicates that the label y is incorrect. Based upon this understanding, two assumptions are arrived at:

(A1) The p-value p_p of the "positive meta class" can be considered as an approximation of the p-value p_y of the class y assigned to x.

(A2) The p-value p_n of the "negative meta class" can be considered as the sum of the p-values of the all classes $\mathbf{Y}\backslash\{y\}$ when y is assigned to instance x.

These intuitive assumptions (A1) and (A2) are the basis for how the combined classifier $h : m$ is interpreted. The score $\frac{p_p}{p_n}$ is considered to decide if the particular instance should be classified (as in [188]). A reliable threshold T is determined on the score $\frac{p_p}{p_n}$ to decide if a classification made by h on an instance x is reliable. If the score is greater than the threshold, the classification of x is reliable; otherwise x is left unclassified. The threshold T imposes a certain accuracy on the instances that $h : m$ can classify and the rejection of the combined classifier.

Generalized performance

Smirnov et al. [328] further estimated a threshold T such that the combined classifier $h : m$ has a predefined target accuracy rate $At_{h:m}$ on the instances that $h : m$ can classify. We showed earlier that the p-values of the combined classifier $h : m$ equal the p-values of the conformity-based metaclassifier. By Theorem 9.1, the accuracy $A_{h:m}$ of the combined classifier is equal to the precision rate P_m of the metaclassifier m. A threshold T on the p-values of m is obtained such that the precision rate P_m is equal to the target accuracy rate $At_{h:m}$. To build a conformity-based metaclassifier with precision rate Pt_m, the authors in [328] identified a reliability threshold T using the following steps (similar to [351]):

1. Construct the Receiver Operating Characteristic (ROC) convex hull, called ROCCH, for the metaclassifier m using the score ratio $\frac{p_p}{p_n}$. Each threshold value for the score ratio yields a single value on the ROC curve. Use k-fold cross validation as in Section 9.2.2.
2. Construct the iso-precision line with the target precision Pt_M given by the equation relating TP_r and FP_r; $TPr_M = \frac{Pt_M}{1-Pt_M}\frac{N_M}{P_M}FPr_M$, where N_M is the number of "negative metaclass" instances and P_M is the number of "positive metaclass" instances. This line represents classifiers with precision rate Pt_M.

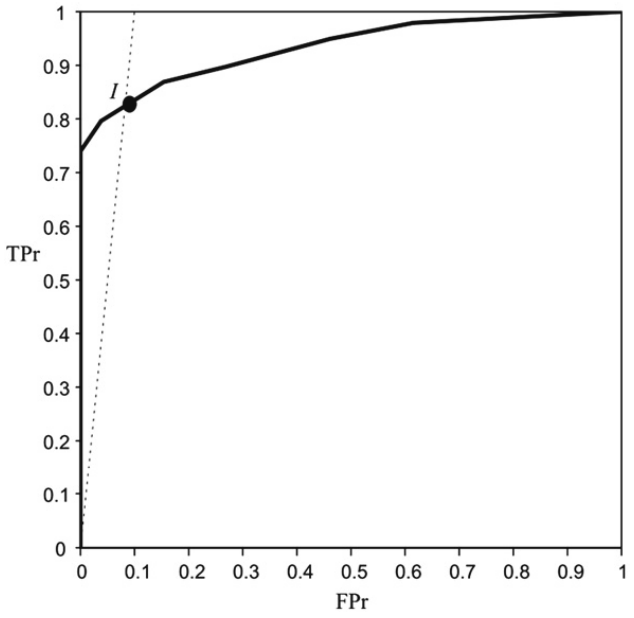

FIGURE 9.3
ROCCH for a conformity-based nearest neighbor classifier [280], trained on metadata from a naive Bayes classifier on the Wisconsin Breast Cancer data [8]. The iso-precision line is for a precision rate of 0.9. Figure as in [328].

3. Find the intersection I of the iso-precision line with the convex hull. The value of the ratio $\frac{p_p}{p_n}$ at the intersection point is the reliability threshold T (see Figure 9.3).

According to [351], the precision rate P_M of the conformity-based metaclassifier will now be equal to the target precision rate Pt_M. From Theorem 9.1, the accuracy $A_{h:m}$ of the combined classifier $h : m$ on the classified instances will be Pt_M, which is equal to $At_{h:m}$, the target accuracy. Also, evidently, the accuracy $A_{h:m}$ is maximized when the precision P_m is maximized. The precision is maximized for the iso-precision line with slope equal to the slope of the line segment of the ROCCH starting from the origin. The highest point of this segment maximizes the number of covered metainstances (see point (0, 0.74) of ROCCH in Figure 9.3). Thus, at this point, by Theorem 2, the accuracy rate of the combined classifier $h : m$ is maximized while the rejection rate is minimized.

9.2.3 Experiments

For completeness, we now present the experimental results obtained by Smirnov et al. in [328] using the 12 UCI data multisets [8]. The base classifier was chosen to be naive Bayes (NB) [231]. The metaclassifier was chosen to be a conformity-based nearest neighbor classifier (called the CNN) [280]. The metaclassifier provides the p-values and is nonlinear. The resulting combined classifier is denoted by $NB : CNN$.

Table 9.1 Rejection rates for classifiers *NB* : *CNN* and *CNN* for accuracy rate at 1.0.

Data Set	$R_{NB:CNN}$	$A_{NB:CNN}$
Annealing	0.57*	0.88
Audiology	0.84*	0.99
Wisconsin breast cancer	0.29	0.31
Glass	0.82*	0.88
Hepatitis	0.71	0.65
Heart-Statlog	0.94	0.85*
Ionosphere	0.70	0.55
Iris	0.29	0.13*
Lymphography	0.80*	0.99
Soybean	0.87*	0.90
Vote	0.52	0.47*
Zoo	0.23	0.14*

* *Indicates rejection rates that are statistically lower according to a paired-t test on significance level of 0.05 (as presented in* [328] *).*

Ten-fold cross validation was used to create the metadata for $NB : CNN$. The metaconformity approach was used to estimate the p-values of $NB : CNN$.

The combined classifier was experimented on 12 UCI data multisets [8] (listed in Table 9.1). At each value of the reliability threshold T in the range $[0, +\infty)$, the score $\frac{p_p}{p_n}$ was estimated. If the score was greater than the threshold for a particular instance, the classification was considered reliable, otherwise the instance was left unclassified. At each value of T, 10-fold cross validation was used to estimate the accuracy/rejection graphs for the different data sets. The results are presented in Figure 9.4. Each point on the curves represents a $NB : CNN$ classifier with a particular threshold $T \in [0, +\infty)$. The leftmost points refer to $T = 0$, the subsequent points refer to increasing T values with the rightmost points depicting T approaching $+\infty$. The rejection rates for the combined classifier $NB : CNN$ and the metaclassifier CNN for accuracy 1.0 are presented in Table 9.1. From the data in Table 9.1, it is seen that each of these classifiers is significantly better for five data multisets. This indicates that the classifiers are good and different, which validates assumptions (A1) and (A2) that the classifiers approximate the p-values very well.

The ROC procedure was used to construct classifiers with predefined accuracies for all the UCI data multisets. The ROCCH was constructed using 10-fold cross validation similar to the manner in which metadata was generated (see Section 9.2.2). The reliability threshold T was estimated for a target accuracy of $At = 1.0$. The accuracy and rejection rates for the combined classifier $NB : CNN$ for accuracy $At = 1.0$ are depicted in Table 9.2. The deviation in accuracy was 0.3. This was attributed to the instability in the 10-fold cross validation during metadata generation and/or the size of the training (meta) data. The largest deviations were observed

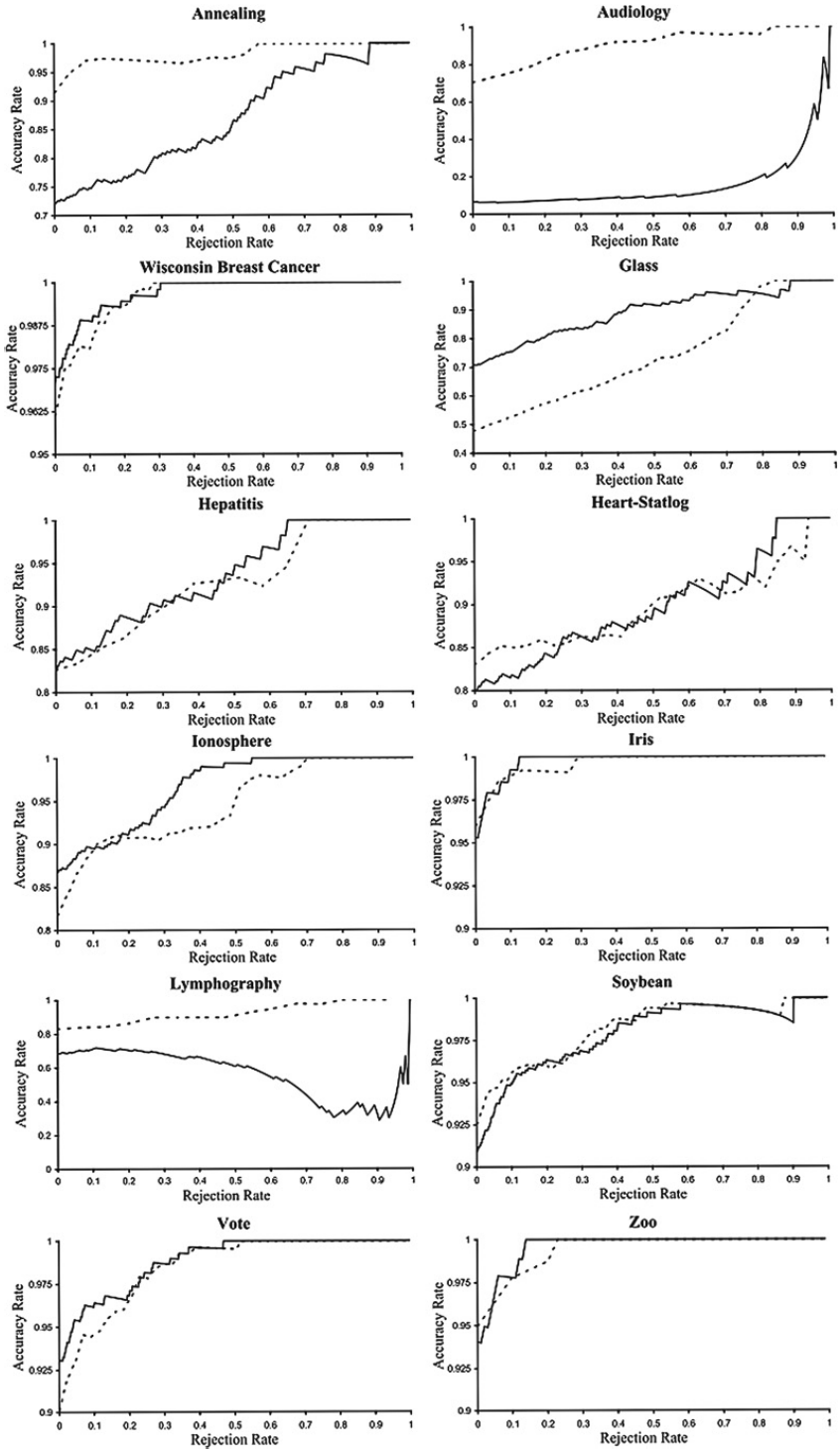

FIGURE 9.4

The accuracy/rejection graphs for *NB*:*CNN* (dashed line) and *CNN* (bold line) (as presented in [328]).

Table 9.2 Accuracy and Rejection rates for the *NB* : *CNN* classifier generated using the ROC method with target accuracy $At = 1.0$ (as presented in [328]).

Data Set	$R_{NB:CNN}$	$A_{NB:CNN}$
Annealing	0.65	1.00
Audiology	0.83	0.97
Wisconsin. breast cancer	0.29	1.00
Glass	0.87	1.00
Hepatitis	0.71	0.98
Heart-Statlog	0.88	0.97
Ionosphere	0.72	1.00
Iris	0.31	0.99
Lymphography	0.79	0.98
Soybean	0.72	1.00
Vote	0.49	1.00
Zoo	0.18	0.99

for data with size less than 300 instances: Audiology, Hepatitis, Heart-Statlog, Iris, Lymphography, and Zoo. We infer from the results that the ROC-based strategy is applicable in practice and produces accurate results when the data size is reasonably large.

9.3 Single-Stacking Conformal Predictors

The metaconformal approach works well in a binary class setting. For a multi class problem, the meta conformal approach does not provide the p-value estimates for all the class labels $y \in \mathbf{Y}$. Smirnov et al. [329] proposed the Single-Stacking Conformal Predictors which employ a stacking ensemble to create a combined classifier to overcome the limitations of metaconformal predictors. In this section, we describe the Single-Stacking Conformal Predictors and illustrate their performance on the MYCAREVENT dataset as in [329]. The MYCAREVENT project seeks to find the reliable estimate on a car status to provide roadside assistance.

9.3.1 Metaconformity versus Single Stacking

In the metaconformal approach, a base classifier $h \in \mathbf{H}$ was combined with a meta-classifier $m \in \mathbf{M}$ to create a combined classifier $h : m$. The combined classifier is capable of estimating p-values for the classified instances. There is however, one shortcoming with the combined classifier. If the classifier m assigns a class label y to an instance x, the associated p-value of class y is set to p_0, the p-value of the metaclass "correct classification." The sum of the p-values of the remaining classes $\mathbf{Y}\backslash\{y\}$ is set to p_1, the p-value of the meta class "incorrect classification." The p-value of every class $y \in \mathbf{Y}$ cannot be estimated using the metaconformal approach. If $p_0 < p_1$,

the class with the highest p-value cannot be estimated. To overcome this shortcoming, the Single-Stacking Conformal Predictor was introduced in [329]. This approach employs a stacking ensemble consisting of a base classifier h and a metaclassifier m (with a suitable nonconformity measure) and yields p-values for the instance classifications of the base classifier h. While the base classifier h and the metaclassifier m are similar to the base classifier and metaclassifier in the metaconformal approach, the difference lies in the manner in which metadata is created and the way in which the class labels are estimated by the metaclassifier. The single stacking approach also needs the base classifier to output the class probability distribution. We now describe the Single-Stacking Conformal Predictor.

9.3.2 Single-Stacking Conformal Predictor

Like the metaconformal predictor, the single-stacking predictor also consists of a base classifier $h \in \mathbf{H}$ and a metaclassifier $m \in \mathbf{M}$. The key idea of the combined classifier is to employ a stacking ensemble [383] of a base classifier h and a metaclassifier m. Figure 9.5 depicts a single-stacking classifier. The p-values of the metapredictions are considered as the p-values of the instance classifications of the base classifier. The metadata belongs to space $\mathbf{X}'$ defined by the attributes of the input space $\mathbf{X}$ concatenated with $|\mathbf{Y}|$ elements from the class probability distribution as estimated by the base classifier h. The meta class $\mathbf{Y}'$ coincides with the class set $\mathbf{Y}$.

The metadata $\mathbf{Z}'$ are formed in $\mathbf{X}' \times \mathbf{Y}'$ using k-fold cross validation. A labeled instance $(x_i', y_i) \in \mathbf{Z}'$ is formed from the labeled instance $(x_i, y_i) \in \mathbf{Z}$ such that x_i' is the concatenation of x_i and the class probability distribution computed by h on x_i. The metaclassifier m is then trained on this metadata $\mathbf{Z}'$. The single-stacking ensemble

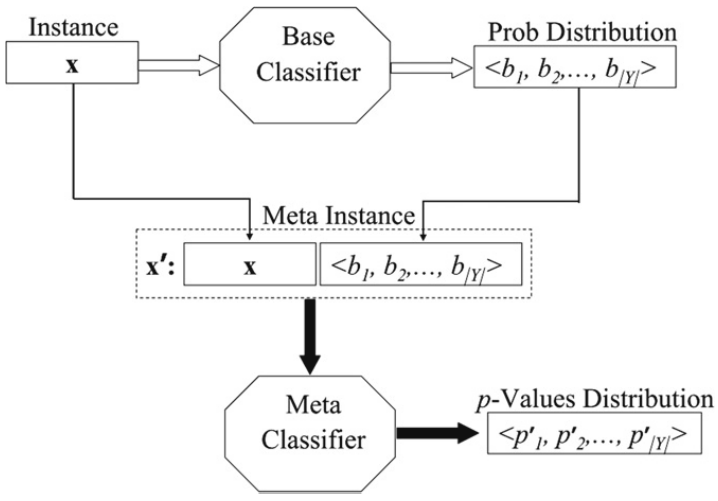

FIGURE 9.5

Single-Stacking Conformity Approach (as in [329]).

classifier consisting of the base classifier h and the meta-classifier m is denoted as $SST_{h;m}$. Given an instance x, the $SST_{h;m}$ classifies it as follows: The base classifier h provides a class probability distribution $\langle b_1, b_2, \ldots, b_{|\mathbf{Y}|}\rangle$ for x. The instance x and the distribution $\langle b_1, b_2, \ldots, b_{|\mathbf{Y}|}\rangle$ are concatenated to form the meta-instance x'. Since the meta-classifier m is associated with a nonconformity score, the class probability distribution of m is a p-values distribution $\langle p'_1, p'_2, \ldots, p'_{|\mathbf{Y}|}\rangle$ consisting of p-values for the classes in $\mathbf{Y}$ obtained using the CP framework. The single-stacking conformity approach approximates for the base classifier h the p-value p_i for each class $y \in \mathbf{Y}$ with the p-value p'_i of the metaclassifier m.

9.3.3 Experiments

Once again, for completeness, we present the results of applying the Single-Stacking Conformal Predictor to the problem of roadside assistance as presented in [329]. These experiments were based on data obtained from the MYCAREVENT project, which has historical patrol car data of previously diagnosed faults and symptoms provided by RAC (Royal Automobile Club, a UK-based motor organization) derived from call center operator conversations. The data consists of four discrete attributes: *Brand* (40 discrete values), *Model* (229 discrete values), *Primary Fault* (35 discrete values), *Secondary Fault* (80 discrete values), and the class attribute *Status*. The class attribute *Status* has three values (class labels):

1. *Fixed*: The problem is solved by roadside assistance and the car can continue its journey safely (3366 instances).
2. *Required Tow*: The car needs to be towed to the workshop (1077 instances).
3. *Other*: Some parts of the problem cannot be solved by roadside assistance but the car is able to get to the workshop on its own (1477 instances).

The experiment employed four standard classifiers: the C4.5 decision tree learner ($C4.5$) [282], the k-nearest neighbor classifier (NN), naive Bayes classifier (NB) [83], and the CP framework-based nearest neighbor classifier (called $TCMNN$) [280]. $C4.5$, NN, and NB were used as independent classifiers and as base classifiers. $TCMNN$ was used as an independent classifier and as a metaclassifier in conformity ensembles. The metaconformal approach was experimented using the following classifiers $MCT(C4.5 : TCMNN)$, $MCT(NN : TCMNN)$, and $MCT(NB : TCMNN)$. The single-stacking ensemble was experimented using the following classifiers $SST(C4.5 : TCMNN)$, $SST(NN : TCMNN)$, and $SST(NB : TCMNN)$. The classifiers' parameters were empirically obtained for maximum classifier performance. The classification probabilities of $C4.5$, NN, and NB were interpreted as the reliability values for the classifiers. For the $TCMNN$ and the ensembles, the p-values that were generated were used as the reliability values.

Many experiments were conducted by varying the reliability threshold r in the range [0,1]. When the reliability value of a classification was above r, the classification was considered reliable, otherwise the instance was left unclassified. For each value

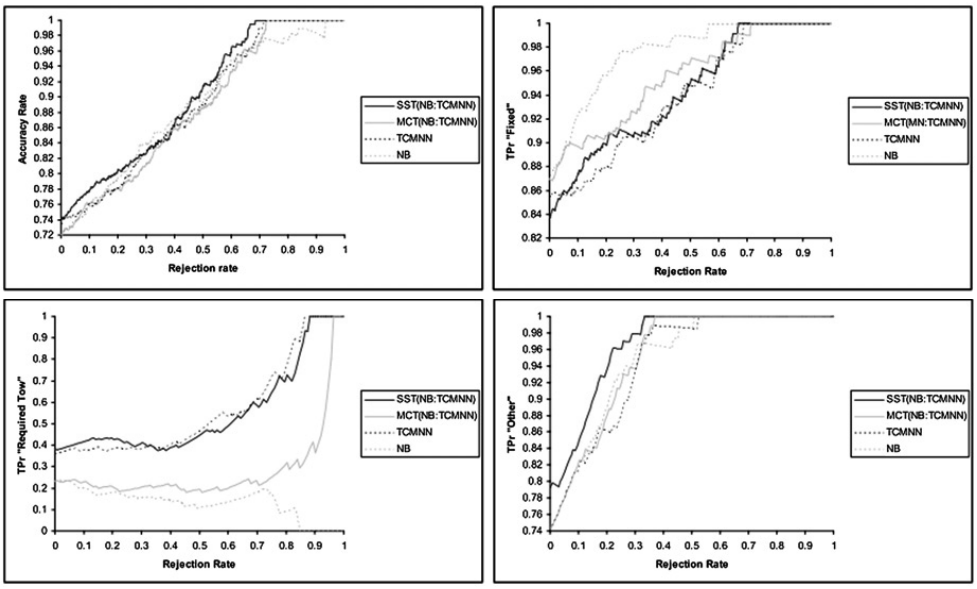

FIGURE 9.6

Accuracy/rejection and *TPr*/rejection graphs for *NB*, *TCMNN*, *SST*(*NB* : *TCMNN*), *MCT* (*NB* : *TCMNN*) (as presented in [329]).

of r, the following were evaluated using 10-fold cross validation: rejection rate (proportion of unclassified instances), accuracy rate (on the classified instances), rejection rate per class (proportion of unclassified instances per class), and true positive rate per class TPr (on the classified instances). Figures 9.6, 9.7, and 9.8 provide the results of the experiments as accuracy/rejection and TPr/rejection graphs for each of the three independent classifiers, NB, NN, and $C4.5$[1] [98]. To enable comparison between the classifiers, the rejection rates for accuracies of 1.0 and TPr of 1.0 per class were extracted from the graph and are presented in Table 9.3.

It can be observed that the accuracy/rejection and TPr/rejection rates of the $TCMNN$, SST ensembles, and the MCT ensembles dominate those of NB, NN, and $C4.5$ classifiers. This leads to the following conclusions:

- The classification probabilities of NB, NN, and $C4.5$ are estimates of the classification reliability values. They fail for the minority classes "Required Tow" and "Other." They can be used for the majority class "Fixed," as the last two columns of Table 9.3 indicate.
- The SST ensembles, MCT ensembles, and the $TCMNN$ provide good classification reliability values for all the classes.

[1]The accuracy/rejection (TPr/rejection) graph of the "always right" classifier is determined by the $\langle(0,1),(1,1)\rangle$ segment. If two classifiers have the same accuracy rate, the classifier with lower rejection rate is preferred.

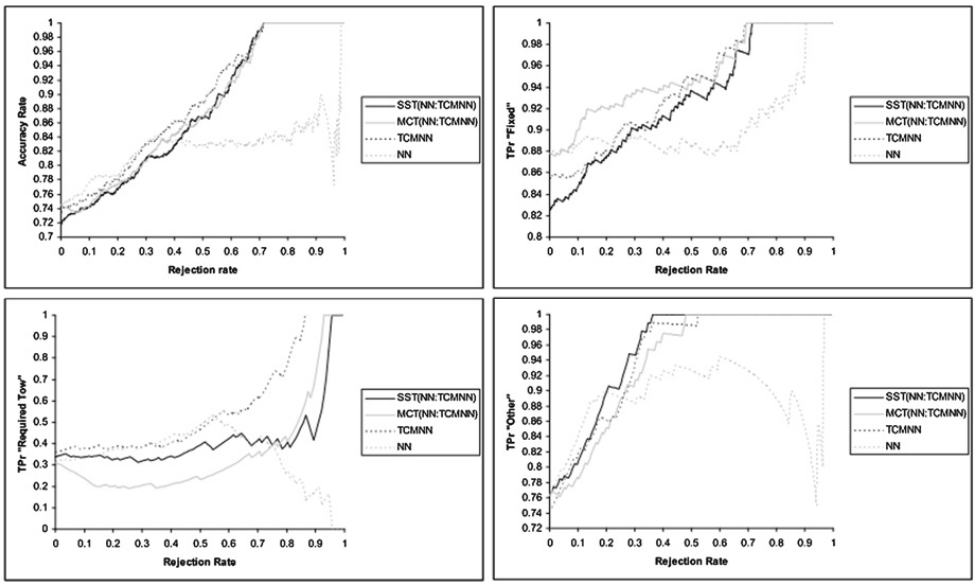

FIGURE 9.7

Accuracy/rejection and *Tpr*/rejection graphs for *NN*, *TCMNN*, *SST*(*NN* : *TCMNN*), *MCT* (*NN* : *TCMNN*) (as presented in [329]).

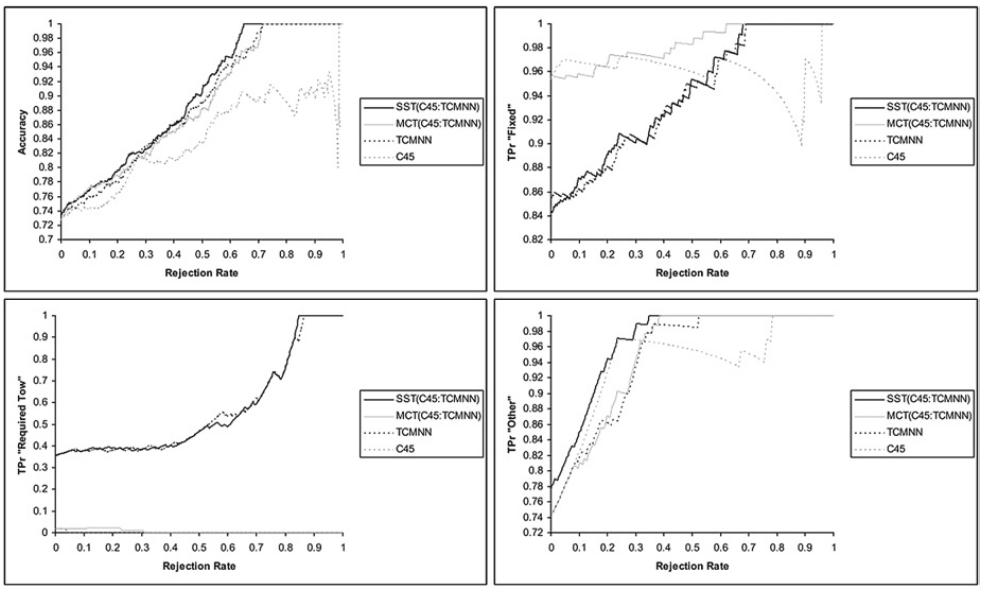

FIGURE 9.8

Accuracy/rejection and *Tpr*/rejection graphs for *C*4.5, *TCMNN*, *SST*(*C*4.5 : *TCMNN*), *MCT*(*C*4.5 : *TCMNN*) (as presented in [329]).

Table 9.3 Rejection rates for accuracy rate 1.0 and *TPr* rate of 1.0. R is the rejection rate for accuracy rate 1.0. R_F is the rejection rate for *TPr* rate 1.0 for "Fixed" class. R_R is the rejection rate for *TPr* rate 1.0 for "Required Tow" class. R_O is the rejection rate for *TPr* rate 1.0 for "Other" class. Undefined rejection rates are denoted by "–".

Classifiers	R	R_F	R_R	R_O
NB	0.93	**0.56**	–	0.51
NN	0.98	0.90	–	0.97
*C*4.5	0.98	0.95	–	0.78
TCMNN	0.72	0.69	0.93	0.52
MCT(*NB* : *TCMNN*)	0.72	0.71	0.96	0.37
MCT(*NN* : *TCMNN*)	0.71	0.69	0.92	0.48
MCT(*C*4.5 : *TCMNN*)	0.71	0.62	–	0.38
SST(*NB* : *TCMNN*)	0.68	0.67	0.88	**0.34**
SST(*NN* : *TCMNN*)	0.71	0.71	0.95	0.36
SST(*C*4.5 : *TCMNN*)	**0.64**	0.68	**0.84**	**0.34**

The SST ensembles outperform the MCT ensembles and the $TCMNN$ on the accuracy graphs as indicated in Table 9.3. For the majority class "Fixed," the $MCT(C4.5 : TCMNN)$ outperforms both the SST ensembles and the $TCMNN$. For the minority classes, however, the SST ensembles perform better than the MCT ensembles and the $TCMNN$. The best classifier is the $SST(C4.5 : TCMNN)$ ensemble. For an accuracy rate of 1.0, its rejection rate is below those of MCT ensembles and $TCMNN$, whose values are at 0.07 and 0.08, respectively. Although the rejection rate of $MCT(C4.5 : TCMNN)$ is below that of $SST(C4.5 : TCMNN)$ for the "Fixed" class, $SST(C4.5 : TCMNN)$ has lower rejection rates for the minority classes "Required Tow" and "Other."

9.4 Conformal Predictors for Time Series Analysis

A time series is a sequence of data points $a_1, \ldots, a_n$ where the indices $1, \ldots, n$ indicate the time or order in which the point has been observed. When applying conformal predictors to time series data, an important consideration is the exchangeability assumption that is necessary for the validity property of the CP framework to be true (see Chapter 1). The observations from a time series are dependent on each other; that is, the order of observations is important and they cannot be assumed to be exchangeable. Hence, in order to apply the CP framework to time series data, the dependency between observations needs to be minimized through suitable data/feature transformations. Dashevskiy and Luo [70] proposed different methods to transform time series data in order to apply the CP framework to derive conformal prediction intervals using regression models. We briefly describe the overall idea of their methodology in this section, while the details of their algorithm and experimental results are substantiated in Chapter 12.

9.4.1 Time Series Analysis Methods

The prediction problem for a time series is defined as the prediction of the current output of a process (that generates the series) using past observations of the process. In addition to typically used methods in machine learning, such as K-Nearest Neighbors (KNN) and Ridge Regression (RR), other stochastic models such as the Auto Regressive Moving Average (ARMA) and Fractional Auto Regressive Integrated Moving Average (FARIMA) models have also been used for time series analysis. In addition, the Aggregating algorithm (AA), which gives a prediction by aggregating the decisions given by various models, has also been used. Since ARMA, FARIMA, and AA models are not familiar models, we briefly describe them next.

Autoregressive moving average (ARMA)

In the Autoregressive Moving Average (ARMA) model [270], the current observation is modeled as a linear combination of the past observations and also as a linear combination of a set of normally distributed random variables. An ARMA process $\{X_t\}$ is defined as:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + W_t + \theta_1 W_{t-1} + \cdots + \theta_q W_{t-q}, \quad (9.11)$$

where $\phi_p \neq 0, \theta \neq 0$ and $\{W_t\} \sim \text{WN}(0, \sigma^2)$ where WN is white noise (i.e., a normal distribution with mean 0 and variance σ^2). An ARMA process is defined by two parameters p and q, where p defines the order of dependency between current and past data and q defines the dependency between the current observation and a set of normally distributed variables. To make a prediction, the parameters θ, ϕ, and σ^2 are updated at each time step and X_t is calculated using a recursive algorithm (see [270] for details).

Fractional auto regressive integrated moving average

Introduced in [126], the Fractional Auto Regressive Integrated Moving Average (FARIMA) method is used to model processes with long-range dependence such as network traffic data. A FARIMA process $\{X_t\}$ is defined as:

$$\phi(B)X_t = \theta(B)(1 - B)^{-d}\epsilon_t, \quad (9.12)$$

where $\phi(B) = 1 + \phi_1 B + \cdots + \phi_p B^p$ and $\theta(B) = 1 + \theta_1 B + \cdots + \phi_q B^q$. p and q exactly correspond to the parameters used in the ARMA model. A FARIMA model is thus parametrized by (p, q, d), where p and q are nonnegative and d is a real number such that $(-1/2) < d < (1/2)$. (For more details of FARIMA, please see [28,126].)

Aggregating algorithm

The Aggregating Algorithm (AA) was first introduced in [358] and is used to aggregate the information given by different experts. When there are multiple experts, we need an algorithm that can utilize all the expert advice and come up with an overall prediction. AA calculates the final outcome such that the error incurred by comparing it with the

true value is less than the maximum of the errors given by all experts. In time series analysis, each regression model can be considered as an expert that predicts the current observation using the past ones. At each time step, the errors from each model are calculated by comparing the predictions with the true value and the expert models are weighted accordingly. A model with high error rate will get smaller weight, and a model with a low error rate will be assigned a greater weight. The AA calculates the final outcome using the weights and the predictions of different models (see [358] for more details).

9.4.2 Conformal Predictors for Time Series Analysis: Methodology

As time series data is not inherently exchangeable, we cannot theoretically guarantee that conformal predictors will assure calibrated error rates in time series analysis. Since the dependency between observations limits the exchangeability assumption, we can relax the dependency assumption. Instead of assuming that every observation is dependent on all its previous observations, we assume that it depends only on observations within a given a lag $T \in N$, which we call the *window*. Considering (without any loss of generality) only time series of type $a_1, a_2, a_3, \ldots$ where $a_i \in R^K$ for any finite dimension K, the objective for our analysis is to predict a_i given $a_1, \ldots, a_{i-1}$. In order to apply the CP framework for regression (see Section 1.7), the underlying regression algorithms may require each data point to be of the form $z_i = (x_i, y_i)$, that is, an (object, label) pair. Hence, the data a_i is transformed into z_i in two different ways where one is exchangeable and the other is not.

To create data that is not exchangeable or *dependent* we use the following rule:

$$\forall T + 1 \leq i \leq n : z_i = (x_i, y_i) := ((a_{i-T}, \ldots, a_{i-1}), a_i),$$

where n is the length of the time series. For example, if $n = 6$ and $T = 2$, the new transformed data will be $\{z_1, z_2, z_3, z_4\} = \{((a_1, a_2), a_3), ((a_2, a_3), a_4), ((a_3, a_4), a_5), ((a_4, a_5), a_6)\}$. We observe that the order of the data is important because there is an overlap of data between z_1 and z_2, and so on. Even though conformal predictors can now be applied to the data, this transformation may not assure theoretical guarantee for validity. Following is another rule that transforms the time series into an exchangeable or *independent* sequence:

$$\begin{aligned}\forall 0 \leq i \leq \left[\frac{n}{T+1}\right] - 1 : z_i \\ = (x_i, y_i) := ((a_{n-i(T+1)-T}, \ldots, a_{n-i(T+1)-1}), a_{n-i(T+1)}),\end{aligned}$$

where $[k]$ denotes the integer part of k. Considering the earlier example where $n = 6$ and $T = 2$, this rule gives $\{z_1, z_2\} = \{((a_4, a_5), a_6), ((a_1, a_2), a_3)\}$. Thus this transformation generates data points that are *independent* and thus, are exchangeable.

Once the data is transformed into (object, label) pairs, regression models are used to predict a new observation. If an ARMA process is used, the data need not be

transformed to (object, label) pairs. As in regression, the nonconformity measure $\alpha(y)$, for a new point in the time series is the error of prediction $|\hat{y} - y|$, where $\hat{y}$ is the estimate and y is the actual value of the observation (see Section 1.7. It should be noted that the nonconformity score of a point y, $\alpha(y)$ depends only on the current observation and not on any other observations.

Algorithm 10 Conformal Prediction Interval calculation for Time series data based on one nearest neighbor

Input: Parameters ϵ and y_1, y_2

$x_2 := y_1$

$\alpha_2(y) := 0$

for n = 3, 4, 5, ... **do**

$x_n := y_{n-1}$

$\hat{y}_n := y_{\arg\min_{i, 2 \le i \le n-1}\{|x_i - x_n|\}}$

$p := \max\{r \in R : \frac{\#\{\alpha_i \ge r, 2 \le i \le n-1\}}{n-2} \ge \epsilon\}$

Output $\Gamma_n := [\hat{y}_n - p, \hat{y}_n + p]$ as prediction for y_n

Get y_n

$\alpha_n(y) := |y_n - \hat{y}_n|$

end for

Algorithm 10 shows the steps involved in computing conformal prediction intervals for time series data, as described by Dashevskiy and Luo [70] for the one nearest neighbor regression method. The maximum error r is selected as the range of the prediction interval such that a significant number of points have errors greater than r. The significance level ϵ is given as a parameter to the algorithm. The prediction interval is given by $[\hat{y}_n - p, \hat{y}_n + p]$. The calculation of the estimate $\hat{y}$ changes in Algorithm 10 depending on whether the underlying model is based on ridge regression or ARMA, and the rest of the steps remain the same. More details of the methodology, experimental results, and analysis are presented in Chapter 12.

9.5 Conclusions

In conclusion, the theoretical guarantees on validity provided by the conformal prediction framework (in the online setting), along with its general applicability to all classification and regression methods, render it suitable to be incorporated in various machine learning settings. While the use of conformal prediction in traditional machine learning settings such as active learning, anomaly detection, and change detection were discussed in earlier chapters, this chapter presented its adaptation to other seemingly nontraditional machine learning settings such as reliability estimation and time series analysis. The experimental results presented in this chapter also support the potential use of conformal predictors in varied machine learning settings.

Acknowledgments

This chapter is based upon work supported by the US National Science Foundation under Grant No. 1116360. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the US National Science Foundation.