Nagiza F. Samatova, William Hendrix, John Jenkins, Kanchana Padmanabhan, and Arpan Chakraborty

Practical Graph Mining with R

Preface

Graph mining is a growing area of study that aims to discover novel and insightful knowledge from data that is represented as a graph. Graph data is ubiquitous in real-world science, government, and industry domains. Examples include social network graphs, Web graphs, cybersecurity networks, power grid networks, and protein-protein interaction networks. Graphs can model the data that takes many forms ranging from traditional vector data through time-series data, spatial data, sequence data to data with uncertainty. While graphs can represent the broad spectrum of data, they are often used when links, relationships, or interconnections are critical to the domain. For example, in the social science domain, the nodes in a graph are people and the links between them are friendship or professional collaboration relationships, such as those captured by Facebook and LinkedIn, respectively. Extraction of useful knowledge from collaboration graphs could facilitate more effective means of job searches.

The book provides a practical, "do-it-yourself" approach to extracting interesting patterns from graph data. It covers many basic and advanced techniques for graph mining, including but not limited to identification of anomalous or frequently recurring patterns in a graph, discovery of groups or clusters of nodes that share common patterns of attributes and relationships, as well as extraction of patterns that distinguish one category of graphs from another and use of those patterns to predict the category for new graphs.

This book is designed as a primary textbook for advanced undergraduates, graduate students, and researchers focused on computer, information, and computational science. It also provides a handy resource for data analytics practitioners. The book is self-contained, requiring no prerequisite knowledge of data mining, and may serve as a standalone textbook for graph mining or as a supplement to a standard data mining textbook.

Each chapter of the book focuses on a particular graph mining task, such as link analysis, cluster analysis, or classification, presents three representative computational techniques to guide and motivate the reader's study of the topic, and culminates in demonstrating how such techniques could be utilized to solve a real-world application problem(s) using real

data sets. Applications include network intrusion detection, tumor cell diagnostics, face recognition, predictive toxicology, mining metabolic and protein-protein interaction networks, community detection in social networks, and others. These representative techniques and applications were chosen based on availability of open-source software and real data, as the book provides several libraries for the R statistical computing environment to "walk-through" the real use cases. The presented techniques are covered in sufficient mathematical depth. At the same time, chapters include a lot of explanatory examples. This makes the abstract principles of graph mining accessible to people of varying levels of expertise, while still providing a rigorous theoretical foundation that is grounded in reality. Though not every available technique is covered in depth, each chapter includes a brief survey of bibliographic references for those interested in further reading. By presenting a level of depth and breadth in each chapter with an ultimate focus on "hands-on" practical application, the book has something to offer to the student, the researcher, and the practitioner of graph data mining.

There are a number of excellent data mining textbooks available; however, there are a number of key features that set this book apart. First, the book focuses specifically on mining graph data. Mining graph data differs from traditional data mining in a number of critical ways. For example, the topic of classification in data mining is often introduced in relation to vector data; however, these techniques are often unsuitable when applied to graphs, which require an entirely different approach such as the use of graph kernels.

Second, the book grounds its study of graph mining in R, the open-source software environment for statistical computing and graphics (http://www.r-project.org/). R is an easy-to-learn open source software package for statistical data mining with capabilities for interactive, visual, and exploratory data analysis. By incorporating specifically designed R codes and examples directly into the book, we hope to encourage the intrepid reader to follow along and to see how the algorithmic techniques discussed in the book correspond to the process of graph data analysis. Each algorithm in the book is presented with its accompanying R code.

Third, the book is a self-contained, teach-yourself practical approach to graph mining. The book includes all source codes, many worked examples, exercises with solutions, and real-world applications. It develops intuition through the use of easy-to-follow examples, backed up with rigorous, yet accessible, mathematical foundations. All examples can easily be run using the included R packages. The underlying mathematics presented in the book is self-contained. Each algorithmic technique is accompanied by a rigorous and formal explanation of the underlying

mathematics, but all math preliminaries are presented in the text; no prior mathematical knowledge is assumed. The level of mathematical complexity ranges from basic to advanced topics. The book comes with several resources for instructors, including exercises and complete, step-by-step solutions.

Finally, every algorithm and example in the book is accompanied with the snippet of the R code that the readers could actually perform any of the graph mining technique discussed in the book from (or while) reading it. Moreover, each chapter provides one or two real application examples, again with R codes and real data sets, that walks the reader through solving the real problem in an easy way.

We would like to acknowledge and thank the students of the CSC 422/522 Automated Learning and Data Analysis course taught in Fall 2009 at North Carolina State University, who wrote this book under the supervision and guidance of thier instructor, Dr. Nagiza Samatova. Despite the contributed chapter format, a specific effort has been made to unify the presentation of the material across all the chapters, thus making it suitable as a textbook. We would also like to thank the students of the Fall 2011 batch of the same course, where the book was used as the primary textbook. The feedback provided by those students was extremely useful in improving the quality of the book. We would like to thank North Carolina State University, Department of Computer Science for their encouragement of this book; the anonymous reviewers for their insightful comments; the CRC press editorial staff for their constant support and guidance during the publication process. Finally, we would like to thank the funding agencies National Science Foundation and the US Department of Energy for supporting the scientific activity of the various co-authors and co-editors of the book.

Contents

List of Figures

List of Tables

1

Performance Metrics for Graph Mining Tasks

Kanchana Padmanabhan

North Carolina State University

John Jenkins

North Carolina State University

1.1 Introduction

So far, we have detailed, discussed, and analyzed algorithms for several data mining tasks, with a focus on graphic data. Given a set of data and some mining goal (e.g. classification, clustering, etc.), we have a good idea on which algorithms to apply in order to analyze the data. However, this is only one half of the battle. The other half lies in understanding, (1) if the model or algorithm applied is (in)appropriate for the task at hand, and (2) if the model or algorithm produces usable results in the context of the problem. To help differentiate these outcomes, there are a number of well-established *performance metrics* which standardize evaluative criteria.

The task of performance measurement, unlike other areas of data mining, does not help analyze the data or the results directly. Instead, the metrics help analyze how the model or algorithm performed its intended task on the data. For example, if a model was built to perform a classification task, i.e., assign class labels to the data, performance metrics could be used to quantify various measures of model prediction accuracy.

The metrics discussed in this chapter can analyze the results of data mining models and algorithms applied to both vector and graph data.

Thus, in this chapter we will utilize the generic term points to denote input data to a mining model or algorithm.

1.2 Supervised Learning Performance Metrics

When incorporating the use of performance metrics to evaluate supervised learning techniques (e.g. classification) on a data set, the learning process undergoes two phases before it is ready to be used to make predictions on data with unknown class labels. The first phase is called the *training phase*, where the model uses a set of data with known class labels to learn the properties of the data belonging to different classes. The second phase is called the *testing phase* where the model is used to predict the class label of a separate set of points with known class labels, to measure various forms of accuracy. To facilitate these two phases, the set of data with known class labels is partitioned into a *training set* and *test set*, respectively. The testing phase gives us an idea of how the model will work on data with unknown class labels, provided the input dataset is somewhat representative of the data to be classified. Various strategies are used for partitioning the data into test and training sets, which are discussed in Section 1.2.5. The model is typically considered to be good if most predictions made during the testing phase match the testing set's class labels. As in most data mining tasks, "most" is quantified by a threshold, based on the application domain, prior knowledge, or statistical techniques.

1.2.1 Confusion Matrices for Binary Classification

1.2.2 2×2 Confusion Matrix

Definition 1.1 *Confusion matrix [27]*
An $n \times n$ matrix, where n is the number of classes and entry (i, j) represents the number of elements with class label i, but predicted to have class label j.

Confusion matrices provide a visually appealing organization of classification results on the test set, allowing us to make a large number of observations about the model's effectiveness. A particularly important confusion matrix that we will describe in detail is the 2×2 matrix, describing binary classification results. The 2×2 matrix is shown in Figure 1.1. For simplicity, we refer to the two classes as *positive* (+) and

i

Actual Class		Predicted Class: +	Predicted Class: -	
	+	f_{++}	f_{+-}	$C = f_{++} + f_{+-}$
	-	f_{-+}	f_{--}	$D = f_{-+} + f_{--}$
		$A = f_{++} + f_{-+}$	$B = f_{+-} + f_{--}$	$T = f_{++} + f_{-+} + f_{+-} + f_{--}$

ii

Actual Class		Predicted Class: +	Predicted Class: -	
	+	Hits (H)	Misses (M)	C = H+M
	-	False Alarms (FA)	Correct Rejections (CR)	D = FA+CR
		A =H+FA	B = M+CR	T = H+FA+M+CR

iii

Actual Class		Predicted Class: +	Predicted Class: -	
	+	True Positives (TP)	False Negatives (FN)	C = TP+FN
	-	False Positives (FP)	True Negatives (TN)	D = FP+TN
		A = TP+FP	B = FN+TN	T = TP+FP+TN+FN

FIGURE 1.1: 2×2 confusion matrix template for model M. A number of terms are used for the entry values, which are shown in i, ii, and iii. In our text we refer to the representation in Figure i.

(1)

Vertex ID	Actual Class	Predicted Class
1	+	+
2	+	+
3	+	+
4	+	+
5	+	-
6	-	+
7	-	+
8	-	-

(2)

		Predicted Class: +	Predicted Class: -	
Actual Class	+	4	1	C = 5
	-	2	1	D = 3
		A = 6	B = 2	T = 8

FIGURE 1.2: Example of a 2×2 confusion matrix.

negative $(-)$. For example, if the data mining task given is to build a model to predict if the tissue sample is cancerous or not, then a cancerous tissue sample will belong to the positive class and a normal tissue sample will belong to the negative class. The four main entries of the 2×2 confusion matrix are described as follows:

- f_{++} or *True Positives* or *Hits* is the set of all data whose actual class and predicted class label are both $+$.
- f_{+-} or *False Negatives* or *Misses* is the set of all data whose actual class is $+$ but predicted class label is $-$.
- f_{-+} or *False Positives* or *False Alarms* is the set of all data whose actual class is $-$ but predicted class label is $+$.
- f_{--} or *True Negatives* or *Correct Rejections* is the set of all data whose actual class and predicted class label are both $-$.

Based on the confusion matrix, several performance metrics can be derived. This section will discuss those measures and show sample calculations using the example in Figure 1.2. Figure 1.2.1 shows the predictions of a model M when classifying the nodes of a graph $G = (V, E)$ where $|V| = 8$. Figure 1.2.2 summarizes the performance using a 2×2 confusion matrix.

1.2.2.1 Basic Metrics

Accuracy

Accuracy (a) [42, 19] is simply the proportion of correct predictions:

$$a = \frac{f_{++} + f_{--}}{T} \tag{1.1}$$

Accuracy measures the extent to which the model got its predictions right. An ideal model would have an accuracy of 1, i.e., actual and predicted values match on every count. The accuracy calculated using the confusion matrix in Figure 1.2 is:

$$\begin{aligned} a &= \frac{4+1}{8}, \\ &= 0.63 \end{aligned}$$

Error Rate

Error rate (ER) [42] is the converse of accuracy: the proportion of incorrect predictions. The error rate is measured as follows:

$$ER = \frac{f_{+-} + f_{-+}}{T} \tag{1.2}$$

Error rate measures the extent to which the model got its predictions wrong. An ideal model would have an error rate of 0, which means that the model has not made any incorrect predictions. The error rate calculated using the confusion matrix in Figure 1.2 is:

$$\begin{aligned} ER &= \frac{2+1}{8}, \\ &= 0.38 \end{aligned}$$

1.2.2.2 Class-specific metrics

The metrics discussed in the following section describe metrics that quantify a model's performance with respect to a specific class as opposed to a model's overall performance.

Recall

Recall is the number of input instances per class whose class label is correctly predicted by the model M. Recall quantifies the notion of *sensitivity*; How many datapoints that belong to a certain class were also

predicted to belong the same class? This can take any value between 0 and 1, we ideally want it to be closer to 1. There are two specific recall measures defined for a 2×2 matrix.

True Positive Rate
The *true positive rate* (TPR) [42, 19], or *positive recall*, is the proportion of points predicted as positive that are correct:

$$TPR = \frac{f_{++}}{f_{++} + f_{+-}} \tag{1.3}$$

In other words, it is the fraction of all input instances belonging to the $+$ class that were correctly predicted to belong to the $+$ class by model M. The true positive rate calculated using the example confusion matrix in Figure 1.2 is:

$$\begin{aligned} TPR &= \frac{4}{5}, \\ &= 0.80 \end{aligned}$$

True Negative Rate
The *true negative rate* (TNR) [42, 19], or *negative recall*, is similarly the proportion of points predicted as negative that are correct:

$$TNR = \frac{f_{--}}{f_{-+} + f_{--}} \tag{1.4}$$

In other words, it is the fraction of all the instances belonging to the $-$ class that were correctly predicted to belong to the $-$ class by M. The true negative rate calculated using the example confusion matrix in Figure 1.2 is:

$$\begin{aligned} TNR &= \frac{1}{3}, \\ &= 0.33 \end{aligned}$$

Misclassification Rate

Misclassification Rate is analogous to recall, being the proportion of input instances whose class label is predicted incorrectly. Ideally, we want it to be closer to 0.

False Positive Rate

The *false Positive rate* (FPR) [42, 19] is the proportion of predicted negative values that are actually negative:

$$FPR = \frac{f_{-+}}{f_{-+} + f_{--}} \tag{1.5}$$

In other words, it is the fraction of all the instances belonging to the $-$ class that were misclassified as $+$. The false positive rate calculated using the example confusion matrix in Figure 1.2 is:

$$\begin{aligned} FPR &= \frac{2}{3}, \\ &= 0.67 \end{aligned}$$

False Negative Rate
Similarly, the *False Negative* (FNR) [42, 19] is the proportion of predicted positive values that are actually negative:

$$FNR = \frac{f_{+-}}{f_{+-} + f_{++}} \tag{1.6}$$

In other words, it is the fraction of all the instances belonging to the $+$ class that were misclassified as $-$ by M. The false negative rate calculated using the example confusion matrix in Figure 1.2 is:

$$\begin{aligned} FNR &= \frac{1}{5}, \\ &= 0.20 \end{aligned}$$

Precision

The *precision* (P) [42, 19] of class Y ($Y = +$ or $Y = -$) is the proportion of points classified as Y that are actually Y:

$$P^{+} = \frac{f_{++}}{f_{-+} + f_{++}}, \tag{1.7}$$

$$P^{-} = \frac{f_{--}}{f_{+-} + f_{--}} \tag{1.8}$$

It quantifies the notion of *specificity*; How many of the points predicted as a certain class actually belong there? The precision for a positive class and a negative class calculated using the example confusion matrix in

Figure 1.2 is, respectively, as follows:

$$\begin{aligned} P^+ &= \frac{4}{4+2}, \\ &= 0.67, \\ P^- &= \frac{1}{1+1}, \\ &= 0.5 \end{aligned}$$

Advanced Metrics

F-measure

F-measure (F) [30, 35] is the harmonic mean of precision and recall values of a class and provides a way to judge the overall performance of the model for each class. Harmonic mean is one of the Pythagorean means (along with geometric and arithmetic) and is not affected by outliers, unlike the arithmetic mean. The F-measure has a value close to 1 only when both the precision and recall are both close to 1.

$$F^+ = \frac{2 \times P^+ \times TPR}{P^+ + TPR}, \tag{1.9}$$

$$F^- = \frac{2 \times P^- \times TNR}{P^- + TNR} \tag{1.10}$$

The F-measure calculated using the example confusion matrix in Figure 1.2 is:

$$\begin{aligned} F^+ &= \frac{2 \times \frac{2}{3} \times \frac{4}{5}}{\frac{2}{3} + \frac{4}{5}}, \\ &= 0.73, \\ F^- &= \frac{2 \times \frac{1}{2} \times \frac{1}{3}}{\frac{1}{2} + \frac{1}{3}}, \\ &= 0.40 \end{aligned}$$

G-mean

The *G-mean* [29, 40] is the geometric mean of the positive (TPR) and negative (TNR) recall values of the two classes.

$$G\text{-mean} = \sqrt{TPR \times TNR} \tag{1.11}$$

This metric provides a way to judge the overall performance of the model because it provides equal representation for all classes. It is also a better

way to quantify the performance of the model because it is less prone to the effects of the underlying class distribution of the input dataset unlike other measures like accuracy. For example, consider a dataset with 100 points, 95 of which have a positive class label and the remaining 5 have a negative class label. Predicting all points as positive will result in an accuracy value of $A = 0.95$, but in reality the model has completely missed predicting the negative class. However, the G-mean value will account for this because the TPR and TNR of the model are 1 and 0, respectively, leading to a G-mean of 0.

The G-mean calculated for the confusion matrix in Figure 1.2 is:

$$\begin{aligned} G\text{-mean} &= \sqrt{\frac{4}{5} \times \frac{1}{3}}, \\ &= 0.52 \end{aligned}$$

The implementation of these measures is available as part of the PerformanceMetrics package.

```
library(PerformanceMetrics)
data(M)
M
     [,1] [,2]
[1,]    4    1
[2,]    2    1

twoCrossConfusionMatrixMetrics(M)
$Accuray
[1] 0.625

$`Error Rate`
[1] 0.375

$`True Positive Rate`
[1] 0.8

$`True Negative Rate`
[1] 0.3333333

$`False Positive Rate`
[1] 0.6666667

$`False Negative Rate`
[1] 0.2
```

Multi-level Confusion Matrix		Predicted Class				Marginal Sum of Actual Values
		Class 1	Class 2	----	Class N	
Actual Class	Class 1	f_{11}	f_{12}	----	f_{1N}	$\sum_{j=1}^{N} f_{1j}$
	Class 2	f_{21}	f_{22}	----	f_{2N}	$\sum_{j=1}^{N} f_{2j}$
	⋮	⋮	⋮	---	⋮	⋮
	Class N	f_{N1}	f_{N2}	----	f_{NN}	$\sum_{j=1}^{N} f_{Nj}$
Marginal Sum of Predictions		$\sum_{i=1}^{N} f_{i1}$	$\sum_{i=1}^{N} f_{i2}$	----	$\sum_{i=1}^{N} f_{iN}$	$T = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{ij}$

FIGURE 1.3: Multi-level confusion matrix template.

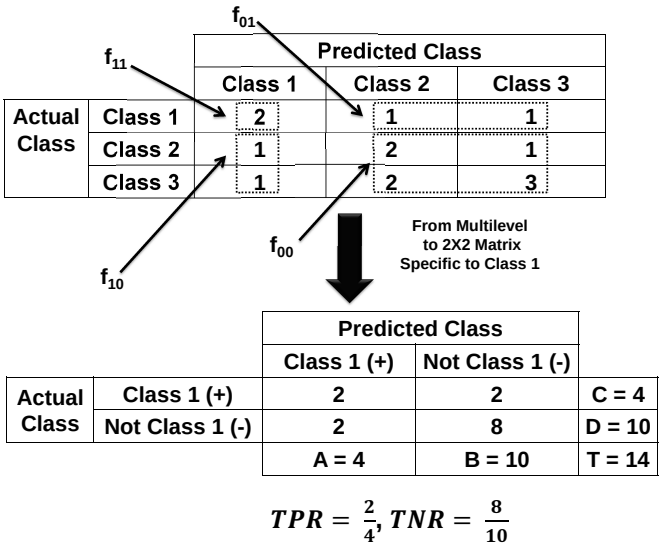
```
$‘Precision+’
[1] 0.6666667

$‘Precision’
[1] 0.5

$‘F-measure+’
[1] 0.7272727

$‘F-measure-’
[1] 0.4

$‘G-Mean’
[1] 0.5163978
```

1.2.3 Multi-level Confusion Matrix

Performance metrics on 2×2 confusion matrices is an important subset for classifier evaluation, but many data mining applications work with more than two classes. For example, the cancer dataset can have three labels: two signifying two types of cancer such as acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL), and one signifying normal tissue. In such cases, the model requires a *multi-level confusion matrix* [47, 1, 45] (see Figure 1.3).

		Predicted Class			Marginal Sum of Actual Values
		Class 1	Class 2	Class 3	
Actual Class	Class 1	2	1	1	4
	Class 2	1	2	1	4
	Class 3	1	2	3	6
Marginal Sum of Predictions		4	5	5	T = 14

FIGURE 1.4: Example of a 3×3 confusion matrix.



$$TPR = \frac{2}{4}, TNR = \frac{8}{10}$$

FIGURE 1.5: Multi-level matrix to a 2×2 conversion.

Many of the measures discussed in the 2×2 confusion matrix, such as accuracy and G-mean, can be extended for confusion matrices of arbitrary number of classes. We can also collapse the multi-level table into a 2×2 matrix for each particular class to compute other measures, such as the number of true "positives." Figure 1.5 provides an example of how to convert a multi-level matrix into a class-specific 2×2 matrix.

However, since problems utilizing multiple classes are prevalent in several application domains, such as biology and climate, measures have been developed that can be calculated for any multi-level matrix. The following measures and examples will be based on the example in Figure 1.4.

Critical Success Index

The critical success index (CSI) [45, 15], also called the *threat score*, is the proportion of the correct predictions of class L of the sum of all predicted values of L and all values of L not correctly predicted.

$$CSI^L = \frac{f_{LL}}{\sum_{i=\text{L or } j=\text{L}} f_{ij}} \tag{1.12}$$

The CSI^{Class1} calculated using the example confusion matrix example in Figure 1.4 is:

$$\begin{aligned} CSI^{Class1} &= \frac{2}{6}, \\ &= 0.33 \end{aligned}$$

Hit Rate

The hit rate (HR) [26, 45], also called the probability of detection, is the success rate of each class L. It is the ratio of the correctly predicted instances of class L to all points with class label L. This is a generalized version of the recall.

$$HR^L = \frac{f_{LL}}{\sum f_{Lj}} \tag{1.13}$$

The HR^{Class1} calculated using the example confusion matrix in Figure 1.4 is:

$$\begin{aligned} HR^{Class1} &= \frac{2}{4}, \\ &= 0.50 \end{aligned}$$

Expected Confusion Matrix		Predicted Class			
		Class 1	Class 2	----	Class N
Actual Class	Class 1	q_{11}	0	----	0
	Class 2	0	q_{22}	----	0
	⋮	⋮	⋮	---	⋮
	Class N	0	0		q_{NN}

FIGURE 1.6: Expected confusion matrix.

Bias

The bias (ϕ) measure [26] helps us understand whether the model is over-predicting or under-predicting a particular class. For each class L, it is the ratio of the total points with class label L to the number of points predicted as L.

$$\phi^L = \frac{\sum f_{Lj}}{\sum f_{iL}} \tag{1.14}$$

Model under-prediction with respect to L corresponds to $\phi^L > 1$, while model over-prediction corresponds to $\phi^L < 1$. ϕ^{Class1} is:

$$\begin{aligned} \phi^{Class2} &= \frac{4}{5}, \\ &= 0.8 \end{aligned}$$

The model M_2 is under-predicting $Class2$.

Mean Absolute Error

Mean Absolute Error (MAE) [24] is the average absolute difference between expected and observed values over all classes. The diagonal multi-level matrix represents the observed values for each class. The expected value is perfect classification (see Figure 1.6), where all values except the diagonal values are zero. If q_{ii} from Figure 1.6 represents the expected values for each class i and f_{ii} from Figure 1.3 represents the observed value for each class i and N is the total number of classes, then the formula for MAE is given as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |f_{ii} - q_{ii}| \tag{1.15}$$

Expected Confusion Matrix		Predicted Class		
		Class 1	Class 2	Class 3
Actual Class	Class 1	4	0	0
	Class 2	0	4	0
	Class 3	0	0	6

Observed Confusion Matrix		Predicted Class		
		Class 1	Class 2	Class 3
Actual Class	Class 1	2	1	1
	Class 2	1	2	1
	Class 3	1	2	3

FIGURE 1.7: An example of expected and observed confusion matrices.

The MAE calculated using the example confusion matrix in Figure 1.7 is:

$$\begin{aligned} MAE &= \frac{2+2+3}{14}, \\ &= 0.5 \end{aligned}$$

Skill Metric

The *skill metric*, as the name suggests, measures the skill or ability of a model to make correct predictions. It takes into consideration the probability of correct predictions by the model and the probability of correct predictions that could have occurred by chance, or in other words, by randomly selecting a class [46]. The concept of chance is necessary because it is important to know if the agreement between the actual and predicted value occurred due to a model's concious decision or due to a random guess. The parameter that denotes random chance is also called the *probability of random agreement*. The term stems from the fact that the actual values and the model predictions are considered two judges, and the chance parameter quantifies the probability that the two judges placed the same point in the same class (reached an agreement) for different unrelated reasons.

Gilbert Skill Score

The *Gilbert Skill Score* (GSS) [45, 15], also called equitable threat score, is similar to CSI but includes the parameter to quantify random chance.

The parameter is called *chance* ($e(\omega)$) and is the product of the frequency of class prediction by the model and the number of input instances that belong to the class. The score is calculated per class. For a 2×2 matrix, it is defined as follows:

$$GSS^L = \frac{f_{LL} - e(\omega)}{\left(\sum_{i=\text{L or } j=\text{L}} f_{ij}\right) - e(\omega)}, \tag{1.16}$$

$$e(\omega) = \frac{A}{T} \times C \tag{1.17}$$

where A, C, and T are defined in Figure 1.2.

The generalized version of $e(\omega)$ that can be applied for multi-level matrices is defined as follows:

$$e(\omega) = \frac{\sum_{j=1}^{N} f_{Lj}}{T} \times \sum_{i=1}^{N} f_{iL} \tag{1.18}$$

GSS^{Class1} of the confusion matrix example in Figure 1.4 is calculated as follows:

$$\begin{aligned} e(\omega) &= \frac{16}{14}, \\ &= \frac{8}{7}, \\ GSS^{Class1} &= \frac{2 - \frac{8}{7}}{6 - \frac{8}{7}}, \\ &= 0.18 \end{aligned}$$

Pi Statistic

The Pi (π) statistic [37] takes into consideration agreements between actual and predicted classes over all classes. For a 2×2 matrix, it is calculated as follows:

$$\pi = \frac{Agree - e(\pi)}{1 - e(\pi)}, \tag{1.19}$$

$$e(\pi) = \left(\frac{(A + C)}{2T}\right)^2 + \left(\frac{(B + D)}{2T}\right)^2 \tag{1.20}$$

where A, B, C, D, and T are defined in Figure 1.2.

Agree is the proportion of times the actual and predicted values agree, or simply *Accuracy*. $e(\pi)$ represents the chance parameter. $e(\pi)$ is calculated by squaring and summing the joint proportion for each class.

The *joint proportion* of each class is the sum of the marginal sum of the actual and predicted values divided by the twice the total number of input instances, which denotes the two dimensions of the multi-level table (actual and predicted). The $e(\pi)$ uses an additive term to account for the collective distribution of classes across the model predictions and actual values.

The reason for including the number of dimensions into the $e(\pi)$ calculation is that π can generalize to comparing more than two dimensions as well; for example, predictions made by three models can be tabulated in a multi-level, 3-dimensional matrix. When calculating the $e(\pi)$ for this matrix, the T in the denominator is multiplied by 3.

The π of the confusion matrix example in Figure 1.2 is calculated as follows:

$$\begin{aligned} e(\pi) &= \left(\frac{(6+5)}{2\times 8}\right)^2 + \left(\frac{(3+2)}{2\times 8}\right)^2, \\ &= 0.57, \\ \pi &= \frac{0.63-0.57}{1-0.57}, \\ &= 0.14 \end{aligned}$$

The generalized version of the $e(\pi)$ that can be applied for multi-level matrices is defined as follows:

$$e(\pi) = \sum_{k=1}^{N}\left(\frac{\sum_{j=1}^{N} f_{kj} + \sum_{i=1}^{N} f_{ik}}{2T}\right)^2 \tag{1.21}$$

Heidke Skill Score (κ Statistic)

Cohen's Kappa (κ) statistic [5], also called Heidke Skill Score (HSS) [20], was derived from the π statistic but the expected agreement by chance uses a multiplicative term as opposed to an additive. The multiplicative terms helps account for the distribution of classes for both the actual values and model predictions. The denominator of $e(\kappa)$ does not include the dimension parameter because $e(\kappa)$ is meant for use only in a two-dimensional multi-level matrix. The κ or HSS for a 2×2 matrix is calculated as follows (see Figure 1.2 for the definition of A, B, C, D, T):

$$e(\kappa) = \frac{(A * C) + (D * B)}{T^2}, \tag{1.22}$$

$$\kappa = \frac{P_{agree} - e(\kappa)}{1 - e(\kappa)} \tag{1.23}$$

P_{agree} here is simply $(f_{--} + f_{++})/T$. The κ of the confusion matrix

example in Figure 1.2 is calculated as follows:

$$\begin{aligned} e(\kappa) &= 0.5625, \\ \kappa &= \frac{0.625 - 0.0.5625}{1 - 0.5625}, \\ \kappa &= 0.143 \end{aligned}$$

The generalized version of κ that can be applied for multi-level matrices is defined as follows. Let ac_i denote that the actual class label is i and pc_i denote that the predicted class label is i:

$$\kappa = \frac{\sum \rho(ac_i, pc_i) - \sum \rho(ac_i)\rho(pc_i)}{1 - \sum \rho(ac_i)\rho(pc_i)}, \tag{1.24}$$

$$\rho(ac_i, pc_i) = \frac{f_{ii}}{T}, \tag{1.25}$$

$$\rho(ac_i) = \frac{\sum f_{ij}}{T}, \tag{1.26}$$

$$\rho(pc_i) = \frac{\sum f_{ji}}{T} \tag{1.27}$$

A score of 1 means that the model is 100% accurate, while a score of 0 means that the model's predictions are completely random. A negative value means that the model's predictions are worse than those occurring at random.

The HSS or κ score of the confusion matrix example in Figure 1.4 is calculated as follows.

$$\begin{aligned} \kappa &= \frac{(\frac{2}{14} + \frac{2}{14} + \frac{3}{14}) - (\frac{4}{14} \times \frac{4}{14} + \frac{4}{14} \times \frac{5}{14} + \frac{6}{14} \times \frac{5}{14})}{1 - (\frac{4}{14} \times \frac{4}{14} + \frac{4}{14} \times \frac{5}{14} + \frac{6}{14} \times \frac{5}{14})}, \\ \kappa &= 0.3 \end{aligned}$$

Alpha-Reliability

The alpha reliability (α) [28, 6] metric is also called the "gold standard" of skill measurements [43]. Unlike the earlier measures, it takes into consideration the amount of disagreement that occurred between the actual and predicted values (D_o) (Error rate) versus a disagreement that could have occurred due to chance (D_e). α for a 2×2 matrix is calculated as follows, where ER represents the error rate:

$$\alpha = 1 - \left(\frac{ER}{e(\alpha)}\right), \tag{1.28}$$

$$e(\alpha) = \frac{(A * C) + (D * B)}{T^2} \tag{1.29}$$

$e(\alpha)$ represents the agreement/disagreement that occurred due to chance and utilizes the same parameter as κ. $\alpha = 1$ means that the model has perfect reliability. A value of 0 indicates a random model, i.e., all the predictions that occurred were due to chance. A value of -1, indicates that disagreements that occurred are worse than those occurred by chance.

α of the confusion matrix example in Figure 1.2 is calculated as follows:

$$\begin{aligned} \alpha &= 1 - \left(8 \times \frac{2+1}{(5\times 6)+(2\times 3)}\right), \\ &= 1 - \left(8 \times \frac{3}{36}\right), \\ &= 0.33 \end{aligned}$$

$e(\alpha)$, applied for multi-level matrices, is defined as follows:

$$ER = \frac{\sum_{i\neq j} f_{ij}}{T}, \tag{1.30}$$

$$e(\alpha) = \sum_{k=1}^{N} \frac{\sum_{j=1}^{N} f_{kj} \sum_{i=1}^{N} f_{ik}}{T} \tag{1.31}$$

It has been shown through various experiments and theory that a reasonable maximum value for probability by chance is 0.5 but the statistics described so far can produce probability by chance values in the range of 0 to 1 [16]. To correct for this fact, Gwet [17] introduced a new method to calculate the probability by chance. It is called $e(\gamma)$ and was defined as follows:

$$e(\gamma) = \beta_1(1-\beta_1) + \beta_2(1-\beta_2), \tag{1.32}$$

$$\beta_1 = \frac{C+A}{2\times T}, \tag{1.33}$$

$$\beta_2 = \frac{B+D}{2\times T} \tag{1.34}$$

Due to the way the confusion matrix table is constructed, we find that $\beta_1 = 1 - \beta_2$. Hence, the equation can also be rewritten as

$$e(\gamma) = 2 \times \beta_1(1-\beta_1) \tag{1.35}$$

$e(\gamma)$ of the confusion matrix in Figure 1.2 is calculated as follows:

$$\begin{aligned} \beta_1 &= \frac{6+5}{2\times 8}, \\ e(\gamma) &= 2 \times 0.69(1-0.69) \\ &= 0.43 \end{aligned}$$

The R implementation of these metrics is available in PerformanceMetrics package.

```
library(PerformanceMetrics)
data(MultiLevelM)
MultiLevelM
     [,1] [,2] [,3]
[1,]    2    1    1
[2,]    1    2    1
[3,]    1    2    3
multilevelConfusionMatrixMetrics(MultiLevelM)
$`Critical Success Index`
[1] 0.3333333 0.2857143 0.3750000

$`Hit Rate`
[1] 0.7142857 0.6428571 0.6428571

$Bias
[1] 1.0 0.8 1.2

$`Mean Absolute Error`
[1] 0.5

$`Gilbert Skill Score`
[1] 0.1764706 0.1025641 0.1463415

$`Pi Statistic`
[1] 0.2432432

$`Heidke Skill Score (kappa)`
[1] 0.2461538

$`Alpha Reliability`
[1] -0.4848485
```

1.2.4 Visual Metrics

Receiver Operating Characteristics

Receiver Operating Characteristics [45, 10, 42, 19, 9] (see Figure 1.8) was part of a field called "Signal Detection Theory" developed during World War II for the analysis of radar images. Radar operators had to decide

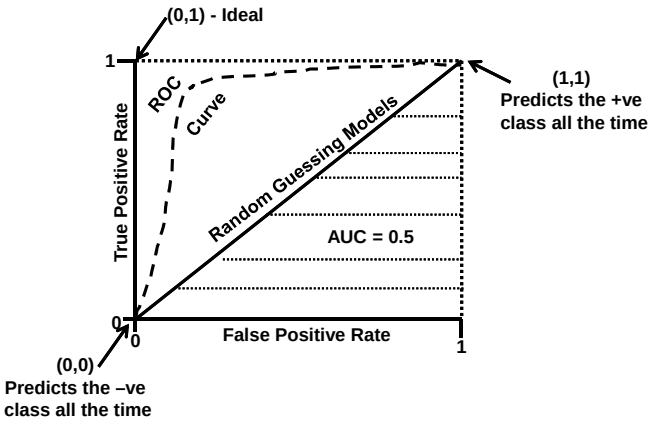


FIGURE 1.8: ROC space and curve.

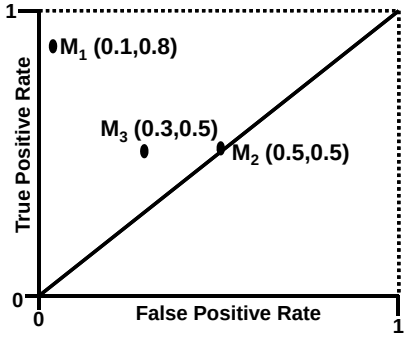


FIGURE 1.9: Plotting the performance of three models, M_1, M_2, and M_3 on the ROC space.

whether a blip on the screen represented an enemy target, a friendly ship, or just noise. Signal detection theory measures the ability of radar receiver operators to make these important distinctions. It was adapted to judge the performance of models that perform predictive data mining.

1.2.4.1 ROC Space

Points in ROC space are built by plotting the *false positive rate* (FPR) along the horizontal axis and *true positive rate* (TPR) along the vertical axis (the true negative and false negative rates can similarly be plotted). *ROC space* is the region constrained by the four points {(0,0), (0,1), (1,0), (1,1)}, corresponding to the minimum and maximum values of both FPR and TPR. We can judge the performance of multiple models by plotting their (FPR, TPR) values on the ROC space. The point $(0, 0)$ represents a model that places all the points in the *negative class*, while point $(1, 1)$ represents a model that places all the points in the *positive class*. The point $(0, 1)$ denotes the ideal scenario of perfect classification. We typically want the model to lie in the upper left region of the space.

Models with (FPR, TPR) in the lower left side region of the ROC space can be termed *conservative* with respect to the positive class. The model's FP rate is low, so it makes very few mistakes but also has a low TP rate, i.e, the model does not place a datum in the positive class unless there is strong evidence. On the other hand, models with (FPR, TPR) in the upper right hand region can be termed *liberal* with respect to the positive class. They assign more points to the *positive* class but also make a large number of misclassifications.

The line $TPR = FPR$, the diagonal across the ROC space, is termed as the *random guessing* line [10]. The model whose (FPR, TPR) lies along this line is termed to be making completely random predictions [11]. For example, a model that predicts the positive class 50% of the time has an (FPR, TPR) value of $(0.5, 0.5)$ [10]. Similarly, if the model predicts the positive class 80% percent of the time, the TPR is 0.8 but its FPR is also 0.8.

The models whose (FPR, TPR) lie below the diagonal are said to be performing worse than random. However, the ROC space separated by the diagonal is symmetric, that is, an (FPR, TPR) pair that lies below the diagonal can be negated to result in model with an (FPR, TPR) point above the diagonal. On negation, the FPR of the original model becomes the TNR of the negated model and the TPR of the original model becomes the FNR of the negated model. The models with the (FPR, TPR) values below the diagonal are said to have learned the properties of the data correctly but are not applying them right [12].

The ROC can also be used for comparing different models that

are trained and tested using the same dataset. The model whose (FPR, TPR) point the furthest in the upper-right direction [11] is chosen. Figure 1.9 shows a ROC space with the performance of three models M_1, M_2, and M_3 on the same data plotted. We see M_1's performance occurs furthest in the upper-right direction and hence is considered the best model.

1.2.4.2 ROC Curve

There are typically two kinds of predictive models: the discrete and probabilistic. Discrete models only provide a prediction as output and the entire set of predictions can be utilized to build a confusion matrix and arrive at a single (FPR, TPR) point. The second kind provide a probability (or confidence) value as output. These models are called *probabilistic*, *ranking*, or *scoring classifiers*. These models can be made discrete by using a threshold of τ and considering all points with *probability* values greater than τ to belong to the positive class and the rest assigned to the negative class. This threshold τ can be varied and for each threshold we obtain a confusion matrix and thereby an (FPR, TPR) point. The set of all (FPR, TPR) points can be plotted in ROC space, and a curve is constructed using those points.

1.2.4.3 Area Under the Curve

The *area under curve* (AUC) [10, 9, 42] provides a scalar value that quantifies the ROC curve. The total area in the ROC space is 1. The area under the diagonal is exactly half that space, or 0.5. We do not want the performance of any model to be less than or equal to 0.5. We can compare two predictive models by comparing the AUC values obtained on the same dataset. The model with the larger AUC is considered the better predictor.

1.2.5 Cross-validation

Cross-validation [42, 7, 47], also called rotation estimation, is a way to analyze how a predictive data mining model will perform on an unknown dataset, i.e., how well the model generalizes. One iteration of cross-validation involves dividing the data with known class labels into two non-intersecting subsets called the *training* and *validation*, or *test*, data. The training data is then utilized to build the model and the validation data is utilized to analyze the model performance.

With the *holdout* [42, 47] method, a random subset of the data, typically two-thirds of the dataset, is utilized for training, with the remaining used for testing. Then, a single round of cross-validation is performed.

However, in order to avoid bias, cross-validation is performed over multiple rounds and performance metrics for each round are averaged to get the performance of the model.

There are several strategies utilized to perform this iterative cross-validation. *k-fold* validation method [42], the most common, is performed by splitting the data into k equal subsets, using $(k-1)$ sets for training and one set for testing. It is repeated until all the k subsets have had a chance to be the validation set.

A special kind of the k-fold validation is the *leave-one-out* method [42]. In this case, one single point is used for testing and the rest for training, and it is repeated until all the points have been part of the test set. It is equivalent to 1-fold validation.

Random subsampling [42] is performed by randomly splitting the data into training and test datasets. The advantage of random subsampling is that unlike k-fold cross-validation, the ratio of training to test set is not dependent on the number of iterations.

1.3 Unsupervised Learning Performance Metrics

Unsupervised learning techniques (e.g., clustering) are utilized to divide the input points into groups that share some common property. The resulting groups also called clusters are analyzed to quantify the overall performance of the unsupervised learning method U. The set of all clusters is denoted as $G = \{g_1, g_2, \ldots, g_W\}$ and the total number of clusters is denoted as $W = |G|$.

There are two kinds of evaluation that can take place: (1) evaluation using *prior* knowledge and (2) evaluation purely on the basis of the resulting clusters and their attributes without external knowledge. Sometimes a third evaluation can be added where a comparative analysis can be performed between two clustering algorithms, but those methods typically use metrics from the former two categories and, hence, are not discussed in a separate section.

1.3.1 Evaluation Using Prior Knowledge

One example of using *prior* knowledge to test the effectiveness of unsupervised learning methods is by considering a dataset D with known class labels, stripping the labels and providing the set as input to an unsupervised leaning algorithm, U. The resulting clusters are then compared with the knowledge priors to judge the performance of U.

		Cluster	
		Same Cluster	**Different Cluster**
Class	**Same Class**	u_{11}	u_{10}
	Different Class	u_{01}	u_{00}

FIGURE 1.10: Contingency table template for *prior* knowledge based evaluation.

		Cluster	
		Same Cluster	**Different Cluster**
Class	**Same Class**	**9**	**4**
	Different Class	**3**	**12**

FIGURE 1.11: Contingency table example for *prior* knowledge based evaluation.

1.3.1.1 Contingency Table

The contingency table [42] is used to evaluate unsupervised learning techniques when some *prior knowledge* about the points is available. *Prior* knowledge P includes points with known class labels or knowledge from literature that tells us which points belong together. The most common *prior* knowledge used is known class labels, and so the contingency table looks like the one in Figure 1.10. To fill in the table, we consider every pair of non-redundant points (n points produce $(n * (n-1))/2$ pairs). For each pair, we check which one of the four boxes it lies in. The number in each box represents the count of all the pairs that belong to that box, namely:

- The pair belongs to the same class in P and it is also placed in the same cluster by U (u_{11});
- The pair belongs to the same class in P but its points are placed in different clusters by U (u_{10});
- The paired points belong to different classes in P but they are placed in the same cluster by U (u_{01}); and
- The paired points belong to the different classes in P and they are placed in different clusters by U (u_{00}).

Similar to the confusion matrix, we ideally want the u_{10} and u_{01} to be 0. The most common measures used to measure the performance of an unsupervised technique using the matching matrix are described below.

Rand Statistic

The *Rand statistic* [22, 18, 34], also called the *simple matching coefficient* [42], is a measure where both placing a pair of points with the same class label in the same cluster and placing a pair of points with different class labels in different clusters are given equal importance, i.e., it accounts for both specificity and sensitivity of the clustering. The value u_{11} accounts for sensitivity; data that should be placed together according to P and are, in fact, placed together by U. The value u_{00} accounts for specificity; data that should be separated according to P and are, in fact, not placed together by U. The formula to calculate the score is as follows:

$$Rand = \frac{u_{11} + u_{00}}{u_{11} + u_{00} + u_{10} + u_{01}} \tag{1.36}$$

The Rand statistic of the contingency matrix example in Figure 1.11 is calculated as follows:

$$\begin{aligned} Rand &= \frac{3}{4}, \\ &= 0.75 \end{aligned}$$

Jaccard Coefficient

The *Jaccard coefficient* (JC) [25, 42], introduced in Chapter **??** on proximity measures, can be utilized when placing a pair of points with the same class label in the same cluster is primarily important. The formula to calculate the score is as follows:

$$JC = \frac{u_{11}}{u_{11} + u_{10} + u_{01}} \tag{1.37}$$

The JC of the confusion matrix example in Figure 1.11 is calculated as follows:

$$\begin{aligned} JC &= \frac{9}{16}, \\ &= 0.56 \end{aligned}$$

The R implementation of these metrics is available in PerformanceMetrics package.

```
1 library(PerformanceMetrics)
2 data(ContingencyTable)
```

		Cluster ID				
		1	2	.....	W	
Class Labels	1	m_{11}	m_{12}	.....	m_{1W}	
	2	m_{21}	m_{22}	.....	m_{2W}	
	⋮	⋮	⋮	.....	⋮	
	V	m_{V1}	m_{V2}	.....	m_{VW}	
Total Elements per Cluster m_j		$\sum_{i=1}^{V} m_{i1}$	$\sum_{i=1}^{V} m_{i2}$	.....	$\sum_{i=1}^{V} m_{iW}$	$T = \sum_{j=1}^{W} m_j$

FIGURE 1.12: Matching matrix template for *prior* knowledge based evaluation.

		Cluster ID			
		1	2	3	
Class Labels	1	2	0	1	
	2	0	2	1	
m_j		2	2	2	T = 6

FIGURE 1.13: Matching matrix example.

```
ContingencyTable
     [,1] [,2]
[1,]    9    4
[2,]    3   12

contingencyTableMetrics(ContingencyTable)
$Rand
[1] 0.75

$‘Jaccard Coefficient’
[1] 0.5625
```

1.3.1.2 Matching Matrix

The matching matrix [48] (very similar to confusion matrix) is a $V \times W$ matrix (Figure 1.12), where V is the number of class labels in P and W is the total number of resulting clusters. Each row of the matrix represents one class label and each column represents a cluster ID. Each m_{ij} entry represents the number of points from class i that are present in cluster g_j. The table is filled based on the *prior* knowledge P and clusters obtained using U. There are a number of metrics that can be used with the matching matrix.

Entropy

Entropy (E) [38, 39, 42] is the measure of impurity that is present in each cluster, i.e., it measures the extent to which a cluster contains data belonging to the same class. For each cluster g_j, the probability that the member of cluster g_j also belongs to class i is computed using the following formula:

$$p_{ij} = \frac{m_{ij}}{m_j}, \tag{1.38}$$

where m_{ij} is the number of points that belong to both class i and cluster g_j and m_j is the size of cluster g_j. Using this information the entropy (E) of each cluster is calculated as follows:

$$E_{g_j} = -\sum_{i=1}^{V} p_{ij} \log_2 p_{ij} \tag{1.39}$$

The range of values this measure E_j can take is between 0 and 1, where 0 denotes a completely homogeneous cluster. The value of $\log_2 0$ is taken as 0 for these calculations. The total entropy (TE) for the entire set of clusters is calculated as the sum of entropies of each individual cluster weighted by the number of elements in each cluster:

$$TE = \sum_{j=1}^{W} \frac{m_j}{M} E_{g_j}, \tag{1.40}$$

where M is the total number of points in the input. This similarly takes values between 0 and 1 with 0 denoting a good clustering.

The entropy calculations for the matching matrix example in Figure 1.13 are as follows:

$$\begin{aligned} E_1 &= 0, \\ E_2 &= 0, \\ E_3 &= 0.5, \\ TE &= 0.33 \end{aligned}$$

The set of clusters in Figure 1.13 has a TE value of 0.33, which is close to 0 and, hence, represents the results of a good clustering algorithm.

Purity

Purity (Pu) [42] is another measure to analyze the homogeneity of the cluster with respect to the class labels. It is calculated as follows:

$$Pu_{g_j} = \max_{i=1 \text{ to } V} p_{ij} \tag{1.41}$$

This measure takes any value in range of $1/V$ to 1, but unlike entropy, a value of 1 indicates a completely homogeneous cluster. The total purity (TPu) for the entire set of clusters is calculated as the sum of purities of each individual cluster weighted by the number of elements in each cluster.

$$TPu = \sum_{j=1}^{W} \frac{m_j}{M} Pu_{g_j} \tag{1.42}$$

The purity calculations for the matching matrix example in Figure 1.13 are as follows:

$$\begin{aligned} Pu_1 &= 1, \\ Pu_2 &= 1, \\ Pu_3 &= 0.5, \\ TPu &= 0.83 \end{aligned}$$

The set of clusters in Figure 1.13 has a TPu value of 0.83, which is close to 1 and, hence, represents the results of a good clustering algorithm.

The R implementation of these metrics is also available in PerformanceMetrics package.

```
library(PerformanceMetrics)
data(MatchingMatrix)
MatchingMatrix
     [,1] [,2] [,3]
[1,]    2    0    1
[2,]    0    2    1
matchingMatrixMetrics(MatchingMatrix)
$‘Entropy per cluster’
[1] 0 0 1

$‘Total Entropy’
[1] 0.3333333
```

Ideal (Based on class label) X	Data Point 1	Data Point 2	Data Point 3
Data Point 1	1	1	0
Data Point 2	1	1	0
Data Point 3	0	0	1

Observed (Based on clusters) Y	Data Point 1	Data Point 2	Data Point 3
Data Point 1	1	0	1
Data Point 2	0	1	0
Data Point 3	1	0	1

FIGURE 1.14: Example of a pair of ideal and observed matrices.

```
$‘Purity per cluster’
[1] 1.0 1.0 0.5

$‘Total Purity’
[1] 0.8333333
```

1.3.1.3 Ideal and Observed Matrices

Given that the number of points is T, *ideal-matrix* ($\mathbf{X}$) [42] is a $T \times T$ matrix, where each cell (i, j) has a 1 if the points i and j belong to the same class and a 0 if they belong to different classes, based on the *prior* knowledge P. The *observed-matrix* ($\mathbf{Y}$) [42] is a $T \times T$ matrix, where a cell (i, j) has a 1 if the points i and j belong to the same cluster and a 0 if they belong to different cluster. An example of these matrices is shown in Figure 1.14.

Each matrix can be wrapped into a vector by iteratively appending each row to the end of the previous row. Using these vectors, several existing similarity measures can be applied. However, there are also measures that can directly calculate the similarity of two matrices of the same rank. One such a method is discussed below.

Mantel Test

The Mantel (r_M) test [31] is a statistical test of the correlation between two matrices of the same rank. The two matrices, $\mathbf{X}$ and $\mathbf{Y}$, in this case, are symmetric and, hence, it is sufficient to analyze lower or upper

diagonals of each matrix; given that the size of both input matrices is $T \times T$, only $\frac{T \times (T-1)}{2}$ entries are considered. The Mantel r_M score is calculated as follows:

$$r_M(\mathbf{X}, \mathbf{Y}) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{x_{ij} - \overline{\mathbf{X}}}{\sigma_{\mathbf{X}}} \times \frac{y_{ij} - \overline{\mathbf{Y}}}{\sigma_{\mathbf{Y}}}, \tag{1.43}$$

$$n = \frac{T \times (T-1)}{2}, \tag{1.44}$$

where $\overline{\mathbf{X}}$ and $\sigma_{\mathbf{X}}$ denote the mean and standard deviation of the elements of $\mathbf{X}$, respectively; a similar notation holds for matrix $\mathbf{Y}$. The correlation measure takes a value between -1 and 1, where 1 represents a perfect match between the ideal and observed matrices, -1 represents a perfect mismatch between the ideal and observed matrices, and a 0 represents no relationship between the ideal and observed matrices.

1.3.2 Evaluation Using Cluster Properties and Attributes

This section describes methods and measures that can be used to analyze the performance of an unsupervised learning method solely using attributes and properties of the resulting clusters.

Cohesion and Separation

With unsupervised learning techniques, the points are often grouped based on some proximity/distance function defined over a set of attributes. The performance of the algorithm in such cases is measured both in terms of cohesion that exists within each group of nodes and the amount of separation across the groups [42]. The cohesion and separation for a set of clusters are defined as follows:

$$cohesion = \frac{\sum_{i=1}^{W} \frac{\sum_{x \in g_i, y \in g_i} proximity(x,y)}{(|g_i|)^2}}{W}, \tag{1.45}$$

$$seperation = 2 \times \frac{\sum_{i=1}^{W} \sum_{j=1}^{W} \frac{\sum_{x \in g_i, y \in g_j, i \neq j} proximity(x,y)}{|g_i| \times |g_j|}}{W \times (W-1)}, \tag{1.46}$$

where $|g_i|$ denotes the number of objects in cluster g_i and *proximity* represents some proximity metric. Both cohesion and separation can take values in the range 0 to 1. We want cohesion to be closer to 1 and separation to be closer to 0 when using proximity metrics. That scenario is reversed if distance metrics are used.

Silhouette Coefficient

The silhouette coefficient (SC) [36, 42], unlike cohesion and separation, is independent of the number of clusters present. For each point d, the silhouette coefficient (SC) is calculated for each point d as follows:

$$SC(d) = \frac{\mu_{inter} - \mu_{intra}}{\max(\mu_{inter}, \mu_{intra})}, \tag{1.47}$$

where μ_{inter} is the average distance between d and every other point not in the same cluster as d, and μ_{intra} is the average distance between d and every other node in the same cluster as d. The range is between -1 and 1, where a value closer to 1 indicates that the point has been assigned to the best possible cluster.

The average of these *per point* coefficients provides the average SC score for the entire clustering (ASC):

$$ASC = \frac{\sum_{j=1}^{T} SC(d_j)}{T} \tag{1.48}$$

Dunn Index

The Dunn index ($Dunn$) [8] is defined as the ratio between the least inter–cluster distance and the largest intra–cluster distance. The index is defined as follows:

$$Dunn = \frac{dist_{min-inter}}{dist_{max-intra}} \tag{1.49}$$

where $dist_{min-inter}$ denotes the smallest distance between two points from different clusters, and $dist_{max-intra}$ denotes the largest distance between two points from the same cluster. The Dunn index can take a value in range 0 and ∞ and should be maximized; it is preferable to maximize the distance between clusters while minimizing the inter–cluster difference, to create a strong coupling between data points and cluster membership.

C-Index

The C-Index (χ_{index}) [23] is defined as follows:

$$\chi_{index} = \frac{K - K_{min}}{K_{max} - K_{min}} \tag{1.50}$$

where K is the sum of distances over all pairs of points from the same cluster, b is the number of those pairs, and K_{min} is the sum of the b^{th}

smallest distances over all pairs of points. K_{max} is the sum of the b^{th} largest distances over all pairs of points. The χ_{index} can take a value in range 0 and 1 and should be minimized. We see that the numerator of χ_{index} will be closer to zero if the pairs within the same cluster are closer. The denominator is for normalization purposes.

1.4 Optimizing Metrics

Performance metrics help assess the quality of the output or quality of some model built and can also provide feedback to build better models or improve the algorithm in an iterative process. However, in some cases, the performance metric becomes part of the optimization function of the data mining algorithm and, hence, we are able to attain a balance between performance and cost (e.g., in terms of CPU time) as we run the algorithm or build the model. Two examples of such metrics are discussed below.

Sum of Squared Errors

Squared sum error (SSE) [42] is typically used in clustering algorithms to measure the quality of the clusters obtained. This parameter takes into consideration the distance between each point in a cluster to its cluster center (centroid or some other chosen representative). This value is small when points are close to their cluster center, indicating a good clustering. Similarly, a large SSE indicates a poor clustering. Thus, clustering algorithms aim to minimize SSE. During an algorithm run, if we find that the SSE of the current iteration is within acceptable limits, we may choose not to proceed with the next iteration even if the algorithm has not converged (i.e., unchanging cluster centers). For d_j, a point in cluster g_i, where m_i is the cluster center of g_i, and W, the total number of clusters, SSE is defined as follows:

$$SSE = \sum_{i=1}^{W} \sum_{d_j \in g_i} dist(m_i, d_j)^2 \tag{1.51}$$

An alternative to the same measure can use *cohesion*, where the underlying data mining technique uses a proximity measure as opposed to a distance and, hence, the objective becomes a maximization function—maximizing the similarity between every data point and its cluster center. Thus, we may choose to stop running the algorithm when the value

rises above a certain threshold. *Total cohesion* (TC) [42] is calculated as follows:

$$TC = \sum_{i=1}^{W} \sum_{d_j \in g_i} simlarity(m_i, d_j) \tag{1.52}$$

Preserved Variability

Preserved variability [42] is typically used in eigenvector-based dimension reduction techniques to quantify the variance preserved by the chosen dimensions. This parameter is the ratio of the sum of eigenvalues of the k out of r chosen dimensions to the sum of eigenvalues of all the dimensions. The objective of the dimension reduction technique is to maximize this parameter. The value of this parameter depends on the number of dimensions chosen: the more included, the higher the value. Choosing all the dimensions will result in the perfect score of 1. However, the idea behind dimension reduction is to reduce the number of features we have to work with. Thus, dimensions are chosen such that the preserved variability, while not 1, is within an acceptable limit (say 95%). Given that the point is represented in r dimensions ($k << r$), the eigenvalues are $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{r-1} \geq \lambda_r$. The preserved variability (PV) is calculated as follows:

$$PV = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{r} \lambda_j} \tag{1.53}$$

1.5 Statistical Significance Techniques

The performance metrics discussed so far in the chapter provide some score to quantify the model's performance. However, this score alone is not sufficient. In addition to the score, a confidence value for the score will add to its credibility. In this section, we discuss two methods to arrive at such a confidence value. The first is the *Monte Carlo* method, which is a more generic method to assign confidence to any score, and the second is more specific method based on binomial experimentation that obtains a confidence interval for the error (or accuracy) of a model.

1.5.1 Monte Carlo Procedure

In data mining tasks, such as clustering, one form of validation is to calculate the pair-wise proximity (or distance) score between the points assigned to the same cluster. A high average proximity score (or low average distance) would then signify a cohesive cluster. Analyzing the proximity score alone does not say much about the cluster. For example, a cluster may have an average cosine similarity score of 0.99. At first glance, this seems like a good score, but it is possible that a randomly chosen set of points of the same size as the cluster also produces 0.99 score. What does this say about our cluster?

To answer this question, we can use the Monte Carlo procedure [32, 49] to assign a confidence (in terms of p-value) to the score obtained for a cluster. Given a cluster g with score η_g calculated using formula f, the empirical p-value for η_g is estimated by randomly sampling J (e.g., $J \approx 1,000$) subsets of size $|g|$ from the input set and calculating the score using f. Let R be the number of subsets from N that have score $\geq \eta_g$. The p-value is estimated as R/J. We can use a cutoff of, say, 0.05 (most commonly used), to filter out insignificant clusters.

The average pair-wise cosine similarity of a cluster g of size 5 is 0.85. The p-value cutoff chosen is 0.05. Through the Monte Carlo procedure, $1,000$ sets ($J = 1,000$) of size 5 are randomly sampled from the input set. Let the number of random clusters with cosine similarity score greater than or equal to 0.85 be 10 ($R = 10$). The empirical p-value quantifying the significance of the score 0.85 for a cluster of size 5 is $R/J = 10/1,000 = 0.01$, which is ≤ 0.05; hence, cluster g has a significant score.

1.5.2 Confidence Interval Estimation

In predictive data mining tasks such as classification, consider an error rate associated with a model on a sample data set S from the set of data instances D. Let H be the number of instances from S that the model predicts incorrectly. Thus, the error rate for the model, ER_s, is H/S, i.e., the probability that an instance from S will be misclassified. However, ER_s may not be the true error rate ER_t, which is the probability that any random instance drawn from D will be misclassified. We need to utilize ER_s to estimate ER_t. While we cannot point out the exact value of ER_t, we can obtain an interval in which ER_t lies, called the *confidence interval*. For this purpose, the prediction task is taken as a binomial experiment, that has two outcomes: a correct prediction and a wrong prediction. Assuming that S is large enough, we assume that S

comes from a normal distribution. The confidence interval for ER_t [42] is estimated as follows:

$$ER_s \pm Z_l \times \sqrt{\frac{ER_s(1-ER_s)}{|S|}}, \tag{1.54}$$

where Z_l value comes from a standard normal distribution table pertaining $l\%$ area under the curve.

Let the total number of points S classified by a model M be 100 and the number of correct classifications C be 50. The error rate of the model M is $\frac{100-50}{100}$, or $\frac{1}{2}$. Given that the $l\%$ is 97.5%, the corresponding Z_l value from the standard normal distribution is 1.96. Substituting into Equation 1.54, we obtain the confidence interval ER_t of the model M as follows:

$$\begin{aligned} \frac{1}{2} &\pm 1.96\sqrt{\frac{\frac{1}{2}(1-\frac{1}{2})}{100}}, \\ 0.5 &\pm 1.96 \times \frac{1}{20}, \\ 0.5 &\pm 0.098 \end{aligned}$$

Confidence interval estimation can also be extended for unsupervised techniques, where the sample error ER_s will signify the probability that a pair of points from S will be misclustered (or misgrouped), i.e., a pair of points belonging to the same class will be placed in different clusters or a pair of points belonging to different classes will be placed in the same cluster. The true error ER_t signifies the misclustering (misgrouping) of any pair of points from D.

1.6 Model Comparison

As we have seen so far in this book and in general, there are many algorithms developed for the purpose of supervised and unsupervised learning, even for the same data mining task. The question is how to compare different models learned by different algorithms? Aside from the ROC curve, there are a number of comparison metrics, which are covered in this section.

Cost Matrix		Predicted Class	
		+	-
Actual Class	+	c_{11}	c_{10}
	-	c_{01}	c_{00}

FIGURE 1.15: Cost matrix template for model comparison.

Cost Matrix		Predicted Class	
		+	-
Actual Class	+	-20	100
	-	45	-10

Confusion Matrix of M_x		Predicted Class	
		+	-
Actual Class	+	4	1
	-	2	1

Confusion Matrix of M_y		Predicted Class	
		+	-
Actual Class	+	3	2
	-	2	1

FIGURE 1.16: Example of a 2×2 cost matrix.

1.6.1 Cost-based Analysis

In the metrics associated with various learning algorithms the hits and a mistakes are treated equally, i.e., they have equal preferability when evaluating models. However, in real-world applications, certain aspects of model performance are considered more important than others. For example: if a person with cancer was diagnosed as cancer-free or vice-versa then the prediction model should be especially penalized. This penalty can be introduced in the form of a cost-matrix and typically applied with contingency or confusion tables. This analysis is typically used to select one model when we have more than one choice through using different algorithms or different parameters to the learning algorithms.

Figure 1.15 shows a cost-matrix. The total cost of the model, defined by the confusion matrix in Figure 1.1, is calculated by the following formula:

$$C_M = \sum_{i,j} c_{ij} \times f_{ij}, \tag{1.55}$$

where c_{ij} is the cost of classifying a datum belonging to class i into class j and f_{ij} is the number of instances where a datum belonging to class i is predicted as class j [42].

Typically, the cost associated with correct classification is a zero or less, while a misclassification is associated with a positive number. Under this circumstance, we want to minimize the cost of the model. Similarly, we can transform the problem to a maximization problem by assigning positive costs to correct classifications and assigning 0 or less costs to misclassifications.

This cost-based analysis is done to perform comparative analysis of several models trained and tested using the same dataset. The cost of each model is calculated using the same cost matrix and the model with the lowest cost is chosen. Given the cost matrix in Figure 1.16 and the confusion matrices of two models M_x and M_y, the total cost of the model M_x is a 100 and M_y is 220 and so purely on the basis of cost, M_x is a better model.

The R implementation of the cost-based analysis is available in PerformanceMetrics package.

```
1 library(PerformanceMetrics)
2 data(Mx)
3 data(My)
4 data(CostMatrix)
5 Mx
6      [,1] [,2]
7 [1,]    4    1
8 [2,]    2    1
```

```
My
     [,1] [,2]
[1,]    3    2
[2,]    2    1

CostMatrix
     [,1] [,2]
[1,]  -20  100
[2,]   45  -10
costAnalysis(Mx,CostMatrix)
$`Cost Of Model'
[1] 100

costAnalysis(My,CostMatrix)
$`Cost Of Model'
[1] 220
```

A similar cost model can also be applied to the unsupervised results (contingency tables), where the cost of placing two data points belonging to the same class into the same cluster and placing two data points belonging to the different classes into the different clusters are given a zero or negative weights and the cost of placing data points belonging to two different classes into the same cluster and points of the same class into two different clusters are given positive weights. The goal is to once again minimize the associated cost.

1.6.2 Comparison of Error Rates

In predictive data mining, given two models M_1 and M_2 with error rates ER_1 and ER_2, many would reflexively choose the model with the smaller error rate. However, assuming that error rate is the most important metric to the user, this choice would only be correct if the models were trained and tested on the same datasets. If the models were obtained on two different datasets D_1 and D_2, then the first step is to identify if the two error rates are significantly different. A statistical method of determining whether the error rates are significantly different can be performed by assuming that the sizes of D_1 and D_2 are sufficiently large to warrant considering ER_1 and ER_2 to be normally distributed. The difference $diff = |ER_1 - ER_2|$ then also follows a normal distribution with true difference ${diff}_t$ and variance σ^2_{diff}. The variance of $diff$ is the sum of the variances of each model under the assumption that the

models are independent. So the variance of $diff$ [42] is given as

$$\sigma^2_{diff} = \frac{ER_1(1-ER_1)}{|D_1|} + \frac{ER_2(1-ER_2)}{|D_2|} \tag{1.56}$$

The confidence interval for the true difference $diff_t$ is calculated as

$$diff_t = diff \pm Z_l \times \sigma_{diff}, \tag{1.57}$$

where Z_l value comes from a standard normal distribution table pertaining $l\%$ area under the curve.

Let M_1 and M_2 be two models that need to be compared. The error rates for the two models are given as $ER_1 = 0.5$ and $ER_2 = 0.7$. The size of the datasets on which the error rates were obtained for M_1 and M_2 were 5 and 1,000, respectively. The $l\%$ is 97.5%. So, $diff = |ER_1 - ER_2| = 0.2$. The variance of $diff$ is calculated as follows:

$$\begin{aligned} \sigma^2_{diff} &= \frac{0.5 \times 0.5}{5} + \frac{0.7 \times 0.2}{1000}, \\ &= 0.05 \end{aligned}$$

The confidence interval for the true difference $diff_t$ is given by:

$$\begin{aligned} diff_t &= 0.2 \pm 1.96 \times \sqrt{0.05}, \\ &= 0.2 \pm 0.44 \end{aligned}$$

In this case, the difference is not significant since the interval 0.2 ± 0.44 contains the value 0.

1.7 Handling the Class Imbalance Problem in Supervised Learning

Class imbalance occurs when some classes are under-represented in the input dataset. The model trained on such a set may not learn to predict these classes well. One way of overcoming this problem is by a process called sampling.

Sampling is a process utilized by predictive data mining tasks to expand the training set by creating additional data-points. This is useful especially in predictive data mining tasks. Class imbalance occurs when some classes are more represented and other classes hardly have any representation.

The most common approach to solve this problem is through the use of random oversampling and random undersampling. Random undersampling randomly samples from the data belonging to the majority class and removes them [4]. Random oversampling randomly selects entities from the minority class and duplicates them. However, it has been shown [4] that random undersampling has the problem of losing important information that could help in the classification process and random oversampling could make the classifier more specific. To address this problem, SMOTE (Synthetic Minority Over-sampling Technique)[3] was introduced. SMOTE generates new samples that lie in between the samples in the minority class and their nearest neighbours.

1.8 Other Issues

Model overfitting [42] occurs in a predictive model when the model has almost no training error but a large test error. This can typically occur due to several reasons: (1) the model built is too detailed with respect to the training data and, hence, it is unable to generalize for cases that it has not seen before, (2) the noise in the training data has prevented it from learning some important characteristics about the data set, and (3) there is a class imbalance, i.e., some classes have very few representatives in the training dataset. The *generalization error* should typically give us a clue if the model is overfitted.

The *generalization error* is defined as the probability that an unseen data would be misclassified by the model and it is estimated using the training error. Typically, an *optimistic approach* [42] is taken, where the training error is considered as a good estimate of the model and is also considered as the generalization error. However, when overfitting occurs this is no longer a good estimate and, hence, changes are made to the *generalization error* estimate so that when model overfitting has occurred, this error value should give a clue about it.

There are two common approaches to dealing with this issue. The first method is the *pessimistic approach* [42] that takes into consideration the training error and contains a penalty term for the model complexity (since more complex models will likely also be overfitted). The second method is the *reduced error pruning* [42], where the training set is divided into two smaller subsets, where one subset is used for training and the other set, known as the validation set, is used for estimating the generalization error. Reduced error pruning is used by error-prevention mechanisms such as cross-validation.

Another way to prevent overfitting is to first analyze the class distribution in the sample data and resolve issues with imbalanced classes (see Section 1.7). The second way is to analyze the data for outliers and filter them out before building the model. The third way is to follow *Occam's razor* [42] also called *principle of parsimony* that states that given two models with the same generalization error, the simpler one is preferable.

Missing attribute values [42] is another issue that needs to be handled in cases of real-world datasets. The main issue occurs when trying to cluster or classify data with missing attribute values. There are methods to estimate the value of the missing data; for example, the mean, median, or the most frequently occurring value for the attribute can all be used to replace the missing value. But they may still be very different from the actual value of the attribute for that specific data.

1.9 Application Domain-Specific Measures

The metrics discussed so far in the chapter have been application independent. In problems that are heavily tied to an application domain, we also want our data mining results to apply domain knowledge to validate the results. In this section, we discuss one use-case for the domain of *biology*.

For example, in Chapter **??** on Cluster Analysis, we discussed the application of the maximal clique enumeration algorithm to identify protein complexes. While the clusters identified perfectly match the formal definition of a protein complex (complete graph), we still need to perform further testing to make sure that the clusters make sense *biologically*. The easiest way is to have a domain expert look at the results and analyze them. But this would be tedious given the fact that the number of maximal cliques enumerated is typically in the order of thousands. A more feasible way would be to apply existing biological domain knowledge to computationally filter out the random/insignificant maximal cliques.

The most popular way of computationally performing *biological validation*, or assessing the results for their biological relevance, is called *functional enrichment analysis*.

Functional Enrichment Analysis

Functional enrichment analysis [2, 21] typically makes use of the hypergeometric test to analyze the clusters identified. This method works on the principle that if a set of proteins represents a protein complex, then

the proteins work together to carry out a single biological function in the cell, i.e., the proteins are functionally homogeneous. In biology, there is an ontology called the *Gene Ontology* (*GO*), that describes almost every possible function a protein can carry out. A term in the ontology is called a *GO* term. Each protein in an organism is associated with a set of terms from this ontology. The list of all proteins in the organism along with the associated GO terms is taken as the *population* ($\mathbb{P}$) for the hypergeometric test. The list of all proteins in the cluster with their corresponding GO terms is taken as the *sample* ($\mathbb{S}$). We now check to see if a particular GO term enriches the cluster, or if the cluster is biased toward this term. Let the GO term we are currently testing be GO_t. The proteins in the population that are annotated with GO_t are the successes in the population ($\mathbb{P}_{su}$). The proteins in the sample (cluster) that are annotated with GO_t are the successes in the sample ($\mathbb{S}_{su}$). A p-value to quantify the biological significance is calculated using the hypergeometric principle as follows:

$$p\text{-value} = \frac{\binom{|\mathbb{P}_{su}|}{|\mathbb{S}_{su}|}\binom{|\mathbb{P}|-|\mathbb{P}_{su}|}{|\mathbb{S}|-|\mathbb{S}_{su}|}}{\binom{|\mathbb{P}|}{|\mathbb{S}|}} \tag{1.58}$$

A threshold is used to determine if a p-value denotes significance or not. A typical value is 0.05, any GO term resulting in p-value below or equal to 0.05 is said to significantly enrich the cluster.

Given a population $\mathbb{P}$ of size 20 and a GO term *GO:0008125*, the number of successes in the population for *GO:0008125* is 10 ($|\mathbb{P}_{su}| = 10$). The size of cluster (a.k.a. sample) $\mathbb{S}$ is 10 and the number of successes for *GO:0008125* in the sample is 5 ($|\mathbb{S}_{su}| = 5$). The p-value quantifying the enrichment of *GO:0008125* in $\mathbb{S}$ is calculated as follows:

$$\begin{aligned} p\text{-value} &= \frac{\binom{10}{5}\binom{20-10}{10-5}}{\binom{20}{10}}, \\ &= 0.34 \end{aligned}$$

Since $0.34 > 0.05$, *GO:0008125* does not significantly enrich the given cluster $\mathbb{S}$.

Bibliographic Notes

There have been variants of the mean absolute error (MAE) that can be found in literature. The *Brier score* [46] closely resembles the mean

absolute error (MAE), but with the Brier score, the difference between the predicted and observed values is squared. Other variants of the mean absolute error are *median absolute error*, *mean percentage error*, and *mean absolute percentage error* [41]. The median absolute error ($MeAE$), as the name suggests, uses the median rather than the mean to avoid the effect of outliers and skewed distributions. The mean percentage error (MPE) calculates the error rate percentage per-class label and averages over the number of class labels. The mean absolute percentage error ($MAPE$) is similar to the MPE except that the error rate that is calculated is taken as an absolute value.

The generalized version of the F-measure is called the F_β [42, 35] and uses a parameter β to examine the tradeoff between recall and precision. β can take a value between 0 and ∞. When β is closer to 0, F_β will be closer to the precision, while larger values of β move F_β closer to recall. The F-measure discussed in this chapter is called F_1, where $\beta = 1$ and precision and recall are weighted equally.

The Pierce Skill Metric [33, 46] is very similar to the Heidke Skill Score [20, 46], differing in how the chance parameter in the denominator is calculated based on the assumption that the predictions are unbiased, i.e., not influenced by the probability of the actual class information. The *odds ratio* [44] is a simple method that measures the ratio of odds of a true positive conditioned on the event occurring to the odds of a false positive conditioned on the event not occurring. The Gandin-Murphy Skill Score [46, 14, 13] is different from the other skill metrics discussed so far in that it assigns a different weight to the joint probability of the actual and predicted class labels.

The set of performance metrics discussed in this chapter is in no way comprehensive, and newer measures are being developed to improve the existing metrics in the area of data mining.

1.10 Exercises

1. Using the information provided in Table 1.1, compare the performance of models M_1 and M_2 using the metrics defined in Section *Basic Metrics*. Which model has a better performance?
2. Using the information provided in Table 1.1 and Table 1.2, perform a cost based analysis to compare models M_1 and M_2. Which model has a better performance? What do you observe from the conclusions drawn in Question 1 and Question 2.

TABLE 1.1: Predictions by models M_1 and M_2 for Question 1

Datapoint	Actual Class	Model M_1	Model M_2
1	+	+	+
2	-	-	+
3	-	+	-
4	+	+	+
5	+	-	-
6	+	+	-
7	-	+	+
8	+	+	+

TABLE 1.2: Cost matrix for Question 2

		Predicted Class	
		Class=+	Class=-
Actual Class	Class=+	0	0
	Class=-	-145	+120

3. Using the information in Table 1.3 for the results of applying the maximal clique enumeration algorithm (overlapping clusters), identify which performance metric(s) can still be used to draw useful conclusions about the algorithm? Apply those metrics and provide your thoughts on the performance of the algorithm.

4. Using the information provided in Table 1.1, decide if the two models M_1 and M_2, have significantly different error rates (given $l\% = 98.9\%$).

5. The SNN-clustering algorithm, when applied to dataset D,

TABLE 1.3: Data for Question 3

Vertex ID	Clusters(s) Membership	Class Membership
1	1,2	+
2	1	*
3	2,3	-
4	3	*
5	2	-
6	4	+
7	4,5	*
8	3,5	+

produces a set of clusters G_1 of equal size, with the average entropy TE equal to 0.001 and he average purity TPu equal to 0.99. Is it safe to conclude that SNN-clustering produces a good clustering for D?

6. Can the hypergeometric test discussed in Section 1.9 be utilized to analyze the *goodness* of an unsupervised technique in the presence of *prior* class label knowledge? If yes, how?

7. Arrive at a single performance measure that utilizes accuracy, error rate, recall (both positive and negative), precision, and misclassification rates to quantify the performance of a model M that predicts only two class labels. The measure's objective function should be maximization.

8. Paddy has built a model M with 99% accurate predictions and 99.99% confidence. Paddy is developing an unsupervised algorithm U and wants to analyze its performance on a dataset D that has no *prior* knowledge associated with it. Paddy's friend Happy suggests that Paddy use M to assign class labels to the elements in D and then use that as a knowledge *prior* to analyze U's performance. In the worst case, the confidence of U's performance analysis will be 99.99%. What questions should Paddy be asking before going this route?

Bibliography

[1] T. Abudawood and P.A. Flach. Learning multi-class theories in ILP. In *Proceedings of the 20th International Conference on Inductive Logic Programming*, pages 6–13, 2011.

[2] E.I. Boyle, S. Weng, J. Gollub, H. Jin, D. Botstein, J.M. Cherry, and G. Sherlock. GO::termfinder-open source software for accessing gene ontology information and finding significantly enriched gene ontology terms associated with a list of genes. *Bioinformatics*, 20:3710 –3715, 2004.

[3] N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[4] B. Chen and J. Hu. Hierarchical multi-label classification incorporating prior information for gene function prediction. In *Proceedings*

of the Intelligent Systems Design and Applications, pages 231–236, 2010.

[5] J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37–46, 1960.

[6] L.J. Cronbach. Coefficient alpha and the internal structure of tests. *Psychometrika*, 16:297–334, 1951.

[7] P.A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice Hall, 1982.

[8] J.C. Dunn. Well separated clusters and optimal fuzzy-partitions. *Journal of Cybernetics*, 4:95–104, 1974.

[9] T. Fawcett. ROC graphs: Notes and practical considerations for researchers. Technical report, HP Laboratories, 2004.

[10] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

[11] T. Fawcett. ROC graphs: Notes and practical considerations for researchers. Technical report, HP Laboratories, 2007.

[12] P. Flach. Repairing concavities in ROC curves. In *Proceedings of the Workshop on Computational Intelligence*, pages 38–44, 2003.

[13] L.S. Gandin and A.H. Murphy. Equitable skill scores for categorical forecasts. *Monthly Weather Review*, 120:361–370, 1992.

[14] J.P. Gerrity. A note on Gandin and Murphy's equitable skill score. *Monthly Weather Review*, 120:2709–2712, 1992.

[15] G. F. Gilbert. Finleys tornado predictions. *American Meteorological Journal*, 1:166–172, 1884.

[16] K. Gwet. *Handbook of inter-rater reliability: How to Measure the Level of Agreement Between 2 Or Multiple Raters*. STATAXIS Publishing Company, 2001.

[17] K. Gwet. Kappa statistic is not satisfactory for assessing the extent of agreement between raters. *Methods*, 1:1–5, 2002.

[18] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17:107–145, 2001.

[19] L. Hamel. *The Encyclopedia of Data Warehousing and Mining*, chapter Model Assessment with ROC Curves. Idea Group Publishers, 2008.

[20] P. Heidke. Berechnung des erfolges und der gute der windstarkvorhersagen im sturmwarnungsdienst. *Geografika Annaler*, 8:301–349, 1926.

[21] D.W. Huang, B.T. Sherman, and R.A. Lempicki. Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources. *Nature Protocols*, 4:44–57, 2008.

[22] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[23] L. Hubert and J. Schultz. Quadratic assignment as a general data-analysis strategy. *British Journal of Mathematical and Statistical Psychology*, 29:190–241, 1976.

[24] R.J. Hyndman and A.B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22:679–688, 2006.

[25] P. Jaccard. Etude comparative de la distribuition florale dans une portion des alpes et des. *Bulletin Societe Vandoise des sciences naturelles*, 37:547–579, 1901.

[26] I. Jolliffe and D. Stephenson. *Forecast Verification - A Practitioner's Guide in Atmospheric Science*. John Wiley & Sons, 2003.

[27] R. Kohavi and F. Provost. Glossary of terms. *Special Issue on Applications of Machine Learning and the Knowledge Discovery Process, Machine Learning*, 30:271–274, 1998.

[28] K. Krippendorff. Computing Krippendorff's alpha reliability. Technical report, University of Pennsylvania, Annenberg School for Communication, 2007.

[29] M. Kubat, R.C. Holte, and S. Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30:195–215, 1998.

[30] D.D. Lewis. A sequential algorithm for training text classifiers: Corrigendum and additional data. *Special Interest Group on Information Retrieval Forum*, 29:13–19, 1995.

[31] N. Mantel. The detection of disease clustering and a generalized regression approach. *Cancer Research*, 27:209220, 1967.

[32] B.V. North, D. Curtis, and P.C. Sham. A note on the calculation of empirical p-values from Monte Carlo procedures. *American Journal of Human Genetics*, 71:439–441, 2002.

[33] C.S. Peirce. The numerical measure of the success of predictions. *Science*, 4:453–454, 1884.

[34] W.M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850, 1971.

[35] C.J. Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 1979.

[36] P. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[37] W.A. Scott. Reliability of content analysis: The case of nominal scale coding. *Public Opinion Quarterly*, 19:321–325, 1955.

[38] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379423, 1948.

[39] C.E. Shannon and W. Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, 1949.

[40] Y. Sun, M.S. Kamel, and Y. Wang. Boosting for learning multiple classes with imbalanced class distribution. In *Proceedings of the 6th International Conference on Data Mining*, pages 592–602, 2006.

[41] D. Swanson, J. Tayman, and T. Bryan. MAPE-R: A rescaled measure of accuracy for cross-sectional subnational population forecasts. *Journal of Population Research*, 28:225–243, 2011.

[42] P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley Longman Publishing, 2005.

[43] J. Taylor and D. Watkinson. Indexing reliability for condition survey data. *The Conservator*, 30:49–62, 2007.

[44] A. Westergren, S. Karlsson, P. Andersson, O. Ohlsson, and I.R. Hallberg. Eating difficulties, need for assisted eating, nutritional status and pressure ulcers in patients admitted for stroke rehabilitation. *Journal of Clinical Nursing*, 10:257–269, 2001.

[45] D.S. Wilks. *Statistical Methods in the Atmospheric Sciences*. Academic Press, 2005.

[46] D.S. Wilks. *International Geophysics, Volume 100 : Statistical Methods in the Atmospheric Sciences*. Academic Press, 2011.

[47] I.H. Witten, E. Frank, and M.A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers, 2011.

[48] K.Y. Yeung and W.L. Ruzzo. Details of the adjusted rand index and clustering algorithms supplement to the paper "an empirical study on principal component analysis for clustering gene expression data". *Science*, 17:763774, 2001.

[49] B. Zhang, B.H. Park, T. Karpinets, and N.F. Samatova. From pull-down data to protein interaction networks and complexes with biological relevance. *Bioinformatics*, 24:979–986, 2008.