CHAPTER

MACHINE LEARNING IN TRANSPORTATION DATA ANALYTICS

12

Parth Bhavsar[1], Ilya Safro[2], Nidhal Bouaynaya[1], Robi Polikar[1], and Dimah Dera[1]
[1]Rowan University, Glassboro, NJ, United States [2]Clemson University, Clemson, SC, United States

12.1 INTRODUCTION

Machine learning is a collection of methods that enable computers to automate data-driven model building and programming through a systematic discovery of statistically significant patterns in the available data. While machine learning methods are gaining popularity, the first attempt to develop a machine that mimics the behavior of a living creature was conducted by Thomas Ross in 1930s [1]. In, 1959 Arthur Samuel defined machine learning as a *"Field of study that gives computers the ability to learn without being explicitly programmed"* [2]. While the demonstration by Thomas Ross, then a student at the University of Washington and his professor Stevenson Smith, included a Robot Rat that can find a way through artificial maze [1], the study presented by Arthur Samuel included methods to program a computer *"to behave in a way which, if done by human beings or animals, would be described as involving the process of learning."* With the evolution of computing and communication technologies, it became possible to utilize these machine learning algorithms to identify increasingly complex and hidden patterns in the data. Furthermore, it is now possible to develop models that can automatically adapt to bigger and complex data sets and help decision makers to estimate impacts of multiple plausible scenarios in a real time.

The transportation system is evolving from a technology-driven independent system to a data-driven integrated system of systems. For example, researchers are focusing on improving existing Intelligent Transportation Systems (ITS) applications and developing new ITS applications that rely on quality and size of the data [3]. With the increased availability of data, it is now possible to identify patterns such as flow of traffic in real time and behavior of an individual driver in various traffic flow conditions to significantly improve efficiency of existing transportation system operations and predict future trends. For example, providing real-time decision support for incident management can help emergency responders in saving lives as well as reducing incident recovery time. Various algorithms for self-driving cars are another example of machine learning that already begins to significantly affect the transportation system. In this case, the car (a machine) collects data through various sensors and takes driving decisions to provide safe and efficient travel experience to passengers. In both cases, machine learning methods search through several data sets and utilize complex algorithms to identify patterns, take decisions, and/or predict future trends.

Machine learning includes several methods and algorithms, some of them were developed before the term "machine learning" was defined and even today researchers are improving existing

Data Analytics for Intelligent Transportation Systems. DOI: http://dx.doi.org/10.1016/B978-0-12-809715-1.00012-2

methods and developing innovative and efficient methods. It is beyond the scope of this book to provide in-depth review of these techniques. This chapter provides brief overview of selected data preprocessing and machine learning methods for ITS applications.

12.2 MACHINE LEARNING METHODS

Machine learning methods can be characterized based on the type of "learning." There exist several basic types of learning methods, such as: (1) supervised learning where previously labeled data is used to guide the learning process; (2) unsupervised learning, where only unlabeled data is used; (3) semi-supervised learning, which uses both labeled and unlabeled data, and (4) reinforcement learning, where the learning process is guided by a series of feedback/reward cycles.

12.2.1 SUPERVISED LEARNING

Supervised learning method trains a function (or algorithm) to compute output variables based on a given data in which both input and output variables are present. For example, for a given highway, input parameters can be volume (i.e., number of vehicles per hour), current time, and age of the driver, and corresponding output parameter can be an average traffic speed. The learning algorithm utilizes this information for automated training of a function (or algorithm) that computes the speed from a given input. Often, the goal of a learning process is to find a function that minimizes a risk of prediction error that is expressed as a difference between the real and computed output values when tested on a given data set. In such cases, the learning process can be controlled by predetermined acceptable error threshold. The supervised learning process can be thought of as a collection of comments provided by a driving instructor during a lesson in which the instructor explains what should be done (output variables) in different situations (input variables). These comments are adapted by a student driver and turned into a driver behavior. The predetermined thresholds can be thought of as the standards provided by external examiner such as standards published by the Department of Motor Vehicles to pass the driving exam. In this case, the student driver knows the standard way to drive (i.e., actual output) and steps to achieve it (i.e., actual inputs) before he or she starts driving lessons. For the student driver, it becomes an iterative process to achieve acceptable performance. In every iteration, the student driver makes mistakes that are corrected by the driving instructor (i.e., training the new student driver). This iterative process ends when the student successfully gets driving license. Here, we discuss two big categories of supervised learning methods, namely, classification and regression. For example, given the speed information of individual vehicles for a highway section, the problem can be defined in the following ways:

1. Estimating how many drivers are speeding based on the speed limit provided for the highway
2. Estimating an average speed of the highway in future based on the past data.

In the first case, because the solution of the problem relies on classifying the data between users who are speeding vs users who are driving below speed limit, the problem can be thought of as classification problem. In the second case the solution includes mapping past data to estimate average speed of the highway section in future and it can be thought of as regression function.

12.2.1.1 Classification

For a classification problem, the goal of the machine learning algorithm is to categorize or classify given inputs based on the training data set. The training data set in a classification problem includes set of input−output pairs categorized in classes. Many classification problems are binary, i.e., only two classes such as True and False are involved. For example, the individual vehicle's speed data over time can be classified into "speeding" and "not-speeding." Another example of classification is categorical classification, e.g., volume and speed data over time for a highway segment can be classified into level of service "A," "B," "C," "D," "E," and "F." When a new set of observations is presented to a trained classification algorithm, the algorithm categorizes each observation into set of predetermined classes. Further details and selected classification methods are provided in subsequent sections.

12.2.1.2 Regression

For a regression problem, the goal of the machine learning algorithm is to develop a relationship between outputs and inputs using a continuous function to help machines understand how outputs are changing for given inputs. The regression problems can also be envisioned as prediction problems. For example, given the historic information about volume and speed for a given highway, the output can be the average speed of the highway for a next time period. The relationship between output variables and input variables can be defined by various mathematical functions such as linear, nonlinear, and logistic.

To summarize, supervised learning depends on availability of historic data. It is important to note that the data must include input and corresponding known output values in order to train the model. While classification methods are used when the output is of categorical nature, the regression methods are used for the continuous output.

12.2.2 UNSUPERVISED LEARNING

Unsupervised learning methods depend only on the underlying unlabeled data to identify hidden patterns of data instead of inferring models for known input−output pairs. Consider the same student-driver example, the learning process in this case can be thought of as the student driver with no theoretical instructions for a perfect driving and he/she is driving a vehicle without the driving instructor. Without the presence of correct driving and a driving instructor, the student-driver is forced to drive a vehicle by observing other drivers and deducing the correct pattern of driving. It is important to note that, the perception of "correct driving pattern" may vary for each student driver. Clustering and association are two popular families of methods for unsupervised learning problems.

12.2.2.1 Clustering

Clustering methods focus on grouping data in multiple clusters based on similarity between data points. Usually, clustering methods rely on mathematical models to identify similarities between unlabeled data points. The similarities between data points are identified by various methods such as Euclidean distance. Consider an example of a transportation engineer with a closed circuit television (CCTV) recording of peak hour traffic data for a highway segment without control information

such as speed limit of the section. The engineer is trying to identify aggressive drivers, slow drivers, and normal drivers. The engineer's goal is to find clusters such as aggressive drivers, slow drivers, and normal drivers by observing their driving pattern data such as an acceleration and deceleration. In this case, it is important to note that the logic rules of such clusters are defined by the engineer based on his/her own domain expertise.

12.2.2.2 Association

Association method focuses on identifying a particular trend (or trends) in the given data set that represents major data patterns or, the so-called significant association rules that connect data patterns with each other [4]. For example, given crash data of a highway section, finding an association between age of the drivers involved in the crash, blood-alcohol level of the driver at the time of crash, and time of the day can provide critical information to plan sobriety checkpoint locations and times to reduce crash as well as fatalities. For the student-driver example, the association method can be thought of as the student-driver associating "normal driver behavior" with certain age group and speed range.

In summary, unsupervised learning tries to identify complex patterns based on the logic provided in the algorithm.

12.3 UNDERSTANDING DATA

It is important to note that in both supervised and unsupervised learning, the quality, type, and size of the data are significant factors that affect accuracy, efficiency, and robustness of the machine learning algorithm. While the goal of any machine learning application is to capture reality and model uncertainty, the learned model does not usually represent a real world but the reality presented by the data set.

The flowchart in Fig. 12.1 presents a typical step-by-step approach for a machine learning algorithm development process. As depicted in the figure, for any machine learning application, the data preprocessing and learning depends on how real-world issues are defined and data being collected.

12.3.1 PROBLEM DEFINITION

Problem definition is an important first step for any machine learning application that directly affects all following key steps until the model is computed. Below are the basic questions that one has to ask to define a problem and apply appropriate machine learning method.

- What are the input and output variables to be considered by a learning system?
- Are all of the variables of the same importance? Can we find (possibly nonlinear) weighting scheme for variables that associates each variable with some importance factor?
- What types of machine learning method are we interested in (such classification, clustering, and regression)? Do the problem and data belong to the class of (un-, semi-) supervised learning methods?
- What size and type of the data will be used in learning system?

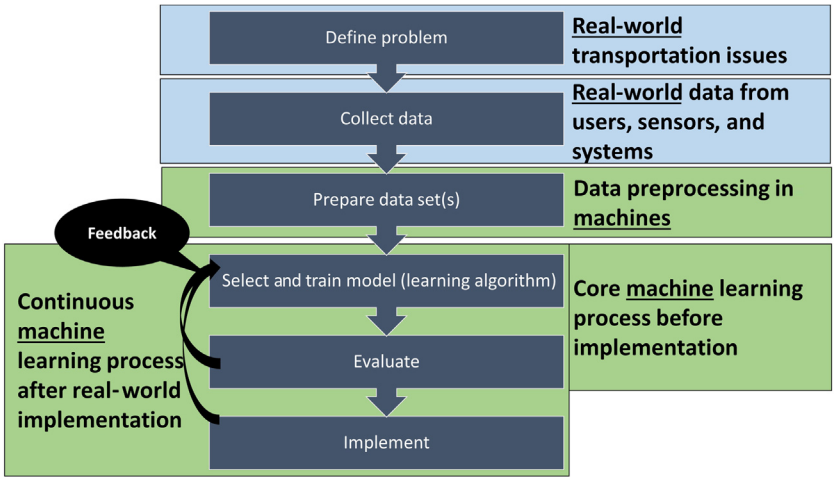

FIGURE 12.1

Machine learning algorithm development approach.

The problem identification and definition is a complex process that depends on several factors such as general perception of the users and needs identified by decision makers. The decision orientated transportation planning approach is one of the methods to identify, define, and prioritize transportation planning and engineering issues [5]. For the transportation data analytics applications, the problem definition provides a much needed information on the required output and possible input parameters. However, a further investigation is required to justify that the solution of the problem requires machine learning approach. According to Tarassenko [6], to identify, whether a given problem requires machine learning solution or not, it must satisfy following three criteria.

1. Given a set of input variables x and output variables y, there must be some logical evidence that a mapping between input and output variables exists such that $y = f(x)$.
2. The form of the function f in above relationship is unknown, i.e., there is no explicit algorithm or set of mathematical equations to describe the solution of the problem.
3. There must exist data to define input and output variables to train and test the learning algorithm.

12.3.2 DATA COLLECTION

The problem definition provides an information about desired output for the application; however, identifying inputs for the problem depends on several parameters. For example, to provide an accurate travel time information for a given highway section, the desired output should be an average speed of the section in near future. However, this information does not provide what input variables should be considered. Hence, the first step of data collection is to develop the list of feasible input−output variables. While there is no set of rules to develop this list, for a transportation engineer, the fundamental knowledge of transportation system, statistics about users, and their behavior

plays an important role. One of the ways to identify input variables is to develop consensus among subject matter experts via in-person or online survey. For example, Delphi survey method can be utilized to identify list of input variables [7]. In this method, the experts are presented with a general survey questions in first round followed by personalized/customized questions in second round onward to refine the opinions obtained in the first round. From the second round onward, the participants review summarized responses of the previous round before answering questions [7]. This approach ensures compilation of list of significant input variables for a given output. In addition, this approach can also be used to reduce the number of input variables as well as to provide weightage/priority to each input variable for data preprocessing.

The second step of data collection is to understand how much data is sufficient. There are no specific rules to answer this question. While researchers have developed certain criteria that are algorithm specific [8], a general approach is to select the size of representative data that captures sufficient variability of the real world for the learning algorithm to be successful most of the time. Typically, availability of resources, time, and educated guess play an important role. However, as presented in Fig. 12.1, it is possible to develop a feedback algorithm that evaluates and trains learning algorithm after the real-world implementation. This approach, if executed successfully, provides a sustainable and long-term solution to overcome limited data variability and inaccuracy issues. As a general rule of thumb, it is always better to have more data than less because there exist a variety of methods that help to prioritize the information and reduce a part of it as statistically insignificant and not influential. However, these methods come with a price of increased running time and complexity of the entire system.

12.3.3 DATA FUSION

The efficiency and efficacy of machine learning applications depend on the quality and variety of data sources considered by the learning algorithm. Data fusion methods strategically integrate and combine multiple data sets into a single, cohesive, and structured data set, extract hidden underlying knowledge, and help in improving prediction accuracy than what is possible using any of the individual data sets. In the geospatial domain, data fusion can be envisioned as combining multiple maps with various data sources. For example, combining location map of recent highway crashes with pavement condition map can help identify potential impacts of deteriorating pavements on highway crashes. Providing ranking and/or priority to each data set or each input variable of each data set is one of the widely used methods to develop a structured data set. This can be achieved by decision-making algorithms [9] or identifying weights through a survey process as explained in Section 3.3.2. Fig. 12.2 presents an example of a next generation roadway asset management system wherein several data sources can be combined using data fusion and analytics methods to predict the performance of pavements over time.

As presented in this example, the long-term pavement performance (LTPP) data which is a generalized nationwide data set can be combined with the location specific laboratory results and real-time data collected by various sensors. In this case, while the real-time data will receive higher weightage compared to LTPP data, the construction method and other cost-related data sets will provide a different perspective to address variability in inputs and outputs. Finally, the successful system will have the ability to predict benefit-to-cost ratio for various alternatives to aid decision

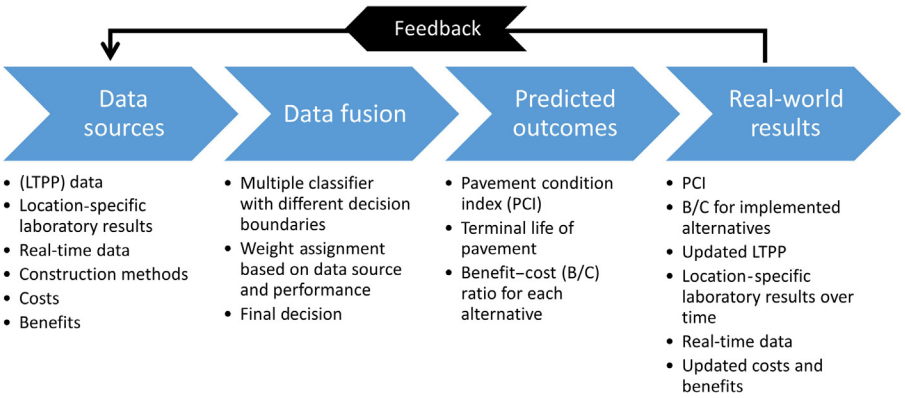

FIGURE 12.2

Intelligent pavement management system.

makers, and the real-world results will be utilized as feedback to improve the performance of the system via continuous machine learning process.

Based on the data sources for input variables and their relation with output variables, the data fusion methods can be classified in the following categories [10]. Detailed review of data fusion methods are provided in Ref. [11].

- *Complementary*: Various input variables in various data source can be identified as complementary when the input variables in each data source provide partial information about the solution and/or output. For example, for a given highway facility, speed information from individual cars and road side unit location on the highway are complementary; combining this information can improve the accuracy of speed prediction for a given facility.
- *Redundant*: When two input variables from two different data sources are providing same information. For example, the radar unit and CCTV camera located on the road side providing average speed information for the same location of the highway.
- *Cooperative*: When the input variables from different data sources can be combined to develop new dummy variable (i.e., information) to improve the accuracy. For example, average speed information collected from the roadway can be combined with CCTV images to estimate density of the given facility.

12.3.4 DATA PREPROCESSING

Data preprocessing is an essential step in which the goal is to remove noise from the raw data and convert it to the form in which potential amount of numerical errors in complex mathematical computations will be minimized. Noise in any data set represents data patterns with errors such as measurement error and errors due to noncalibrated data collection devices, that may significantly affect learning process. Numerical error in this type of scientific computations is one of the most typical pitfalls [12]. Examples include truncation and rounding errors, sensitivity, conditioning, and machine precision problems. Various filtering methods can be employed to reduce or remove the noise [13].

Spatial or graphical plotting of data can often provide visual clues regarding outliers and their impact on other variables. Removing these outliers [14] may help develop better machine learning models that fit the data. When removing outliers, thorough understanding of the problem and fundamental knowledge of the variables involved is required to provide better quality of the final product.

An important step in the data preprocessing is normalization which is a process of conditioning the data within certain boundary to reduce redundancy, eliminate numerical problems, and improve interpretability of the results. It can be done with respect to the mean and variance of the input−output variables such that the normalized data have zero mean and unit variance as provided in the equation below:

$$normalized\_x_i = \frac{x_i - \mu_x}{\sigma_x}$$

where x_i is the ith data component for input (or output) variable x, and μ_x and σ_x are the mean and standard deviation of variable x, respectively.

Variables can also be normalized by rescaling with respect to new minimum and maximum values, i.e.,

$$rescaled\_x_i = \frac{(x_i - x_{imin})}{(x_{imax} - x_{imin})} * (new_{imax} - new_{imin}) + new_{imin}$$

where, x_{imin} and x_{imax} are minimum and maximum values for ith component of x, and new_{imax} respectively, and new_{imin} are the desired maximum and minimum values of the ith component of x, respectively.

If the data values significantly differ in orders of magnitude, one may consider using logarithm transformation as a tool for normalization. Other commonly used normalization methods include square root transformation, and standardization of distributions such as Gaussian, and Poisson distributions [15].

12.4 MACHINE LEARNING ALGORITHMS FOR DATA ANALYTICS

The type of machine learning algorithms may vary from linear regression and classification to complex neuro-fuzzy systems. The following section presents selected popular machine learning algorithms that can be found implemented in a variety of open-source and commercial products.

12.4.1 REGRESSION METHODS

Given a target variable (e.g., an average speed on highway), which up to measurement errors, depends on one or several input variables (e.g., volume), regression describes the nature of dependence between the target and input variables and quantifies the error variance by finding a fitting function that maps the input variables to the target (i.e., output).

Mathematically, the training data is described as the target variables (such as speed) $s_i, i = 1, \ldots, n$ and corresponding input variables (such as volume) v_i, where each input variable can be represented as a vector of information. The general regression model is modeled by by $s_i = f(v_i) + \varepsilon_i$, where ε_i is the regression error. Fig. 12.3 shows examples of linear and nonlinear models.

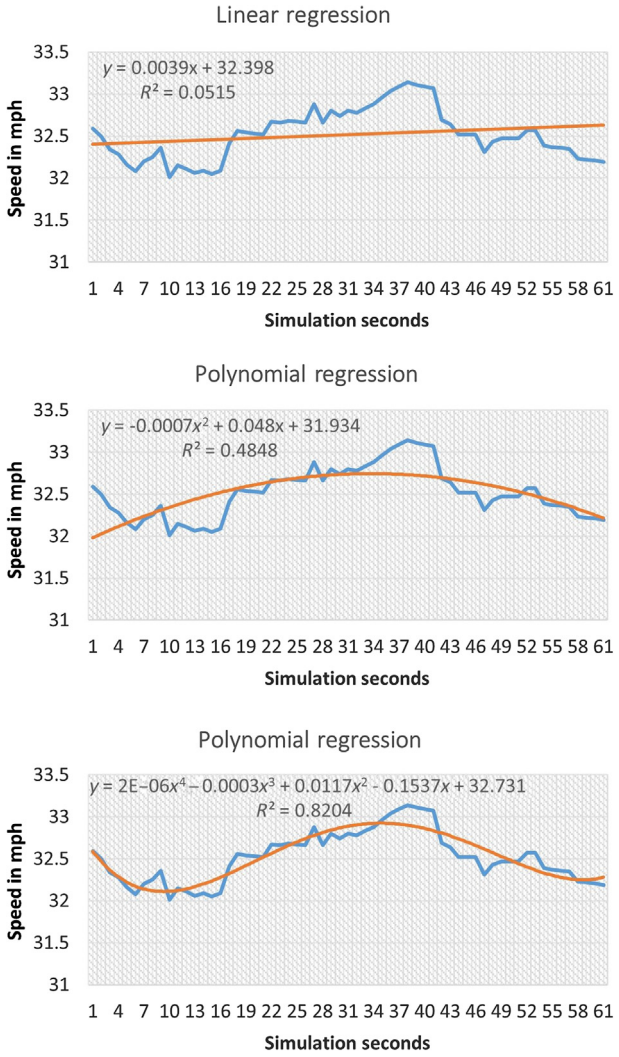

FIGURE 12.3

Examples of regression models.

Linear regression: Consider the following one-variable linear model $f(v_i) = w_0 + w_1 v_i$, i.e.,

$$s_i = w_0 + w_1 v_i + \varepsilon_i$$

where the unknown parameters w_0 and w_1 are called *regression coefficients* or *weights*. For given regression coefficients, we calculate a goodness of fit metric, for instance, the *residual sum of squares* (RSS) as:

$$RSS(w_0, w_1) = \sum_{i=1}^{n} (s_i - [w_0 + w_1 v_i])^2$$

To find the best linear fit, we minimize RSS over all possible w_0, w_1. Taking the derivative of $RSS(w_0, w_1)$ with respect to w_0 and w_1, we find that the optimal regression coefficients are given by:

$$w_1^* = \frac{\left(\sum_{i=1}^n v_i\right)\left(\sum_{i=1}^n s_i\right) - n\sum_{i=1}^n v_i s_i}{\left(\sum_{i=1}^n v_i\right)^2 - n\sum_{i=1}^n v_i^2}$$

and

$$w_0^* = \overline{s} - w_1^* \overline{s}$$

where $\overline{v} = \frac{1}{n}\sum_{i=1}^n v_i$ and $\overline{s} = \frac{1}{n}\sum_{i=1}^n s_i$. For more general models, the optimal regression coefficients may not have a closed form solution. Numerical optimization methods, such as gradient descent, may then be used.

Polynomial regression: In this model, the function f is generalized by a polynomial of degree p by:

$$s_i = w_0 + w_1 v_i + w_2 v_i^2 + \ldots w_p v_i^p + \varepsilon_i$$

When the order of the polynomial p is equal to 1, we find the linear regression model described earlier. A *quadratic model* is described with $p = 2$. Polynomial models are used in such complex data as weather forecasting, flu monitoring, and demand forecasting. More generally, we can consider a generic basis expansion:

$$s_i = w_0 h_0(v_i) + w_1 h_1(v_i) + w_2 h_2(v_i) + \ldots w_p h_p(v_i) + \varepsilon_i$$

Multivariate regression: In this model of f, multiple features are introduced, where each feature is a function of either a single input or multiple inputs. In multivariate regression, the output s is still a scalar but the input is a d-dimensional vector $\boldsymbol{v} = (\boldsymbol{v}[0],\boldsymbol{v}[1],\boldsymbol{v}[2],\ldots,\boldsymbol{v}[d-1])^{\boldsymbol{T}}$.

The multivariable linear model then becomes:

$$s_i = w_0 \boldsymbol{v_i}[0] + w_1 \boldsymbol{v_i}[1] + w_2 \boldsymbol{v_i}[2] + \ldots + w_{d-1} \boldsymbol{v_i}[d-1] + \varepsilon_i$$

In vector form, we have:

$$s_i = \boldsymbol{v_i^T w} + \varepsilon_i$$

where $\boldsymbol{w} = (w_0, w_1, w_2, \ldots, w_{d-1})^{\boldsymbol{T}}$ is the unknown parameter vector.

More generally, the generic basis model becomes

$$s_i = w_0 h_0(\boldsymbol{v_i}) + w_1 h_1(\boldsymbol{v_i}) + w_2 h_2(\boldsymbol{v_i}) + \ldots + w_q h_q(\boldsymbol{v_i}) + \varepsilon_i = \sum_{j=0}^{q} w_j h_j(\boldsymbol{v_i}) + \varepsilon_i$$

The RSS for multivariable regression can be computed as:

$$RSS(\boldsymbol{w}) = \sum_{i=1}^{n} (s_i - \boldsymbol{h^T}(\boldsymbol{v_i})\boldsymbol{w})^2$$

where $\boldsymbol{h^T}(\boldsymbol{v_i}) = (h_0(\boldsymbol{v_i}), h_1(\boldsymbol{v_i}), \ldots, h_q(\boldsymbol{v_i}))$ and $\boldsymbol{w} = (w_0, w_1, w_2, \ldots, w_q)^{\boldsymbol{T}}$. In matrix form,

$$RSS(\boldsymbol{w}) = (\boldsymbol{y} - \boldsymbol{H}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{H}\boldsymbol{w})$$

where $\boldsymbol{s} = (s_1, s_2, \ldots s_n)^{\boldsymbol{T}}$ and $\boldsymbol{H}$ is the matrix such that its columns are formed by the $\boldsymbol{h}(\boldsymbol{v_i})$ vectors. To find the optimal weights, we proceed as in the one-variable case by taking the gradient of the RSS and setting it to zero,

$$\nabla RSS(\boldsymbol{w}) = \nabla[(\boldsymbol{s} - \boldsymbol{H}\boldsymbol{w})^T(\boldsymbol{s} - \boldsymbol{H}\boldsymbol{w})] = -2\boldsymbol{H}^T(\boldsymbol{s} - \boldsymbol{H}\boldsymbol{w}) = 0$$
$$\Rightarrow \boldsymbol{w}^* = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{s}$$

The matrix $\boldsymbol{H}^T\boldsymbol{H}$ is invertible if and only if $\boldsymbol{H}$ is full rank. Since the matrix $\boldsymbol{H}^T\boldsymbol{H}$ is square of dimension q, the computational complexity of finding its inverse is of the order of $O(q^3)$. Instead of finding the inverse explicitly, which may be numerically unstable for large or ill-conditioned matrices, we can resort to numerical optimization techniques, such as gradient descent:

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} + \eta \boldsymbol{H}^{\boldsymbol{T}}\left(\boldsymbol{s} - \boldsymbol{H}\boldsymbol{w}^{(\boldsymbol{k})}\right)$$

where η is an appropriately chosen step size.

How to choose a suitable regression model? The general procedure to choose an appropriate model is summarized as follows:

1. Define a goodness-of-fit measure or equivalently a loss function $L(y, f(x))$, e.g., RSS.
2. Compute an average residual error in data set $\frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i))$.
3. Plot the average training error versus the model complexity.

The error decreases as the model becomes more complex (e.g., considering higher degrees in a polynomial fit). A very small error does not necessarily lead to good predictions, unless the training data includes all the possible features that describe the target variables. A very small fitting error may actually lead to *overfitting*, which means that the model is actually fitting the noise or fluctuations in the data. Therefore, in assessing training error vs model complexity, we look for the inflexion point at which the error rate decreases. Other typical problems of high-degree polynomial fitting include creation of various outliers (such as points of local maximum and minimum), and bad behavior of f in the boundary of manifold given by $\{v_i\}$.

The regression model in the Fig. 12.3 is developed using the speed data of an individual vehicle over time. The data is generated using a traffic micro-simulation software. The "x" axis presents change in speed over time (i.e., speed at every simulation second). Each chart is an example of different regression model that tries to map change in speed over time and can be used to estimate future speed. For each, model presented in the figure, "y" is the target speed, "x" is the time in seconds. R^2 in the figure is a measure of goodness-of-fit of the model or accuracy of the model.

12.4.2 DECISION TREES

Decision tree is a nonparametric method that has a structure similar to tree or flowchart and can be utilized for classification problems [16–19]. The decision tree starts with a primary question for a given problem that must be answered to solve the problem. This question is then broken down to possible solutions that may or may not have definite answer. Each possible solution is then examined by considering the result which may or may not require another decision. If the result of a possible solution requires another decision, the process continues with identifying possible outcomes for the new decision and considering results of each of the outcomes. The process ends when all

possible decisions and outcomes are considered resulting in a tree-like structure in which the logical sequence of decisions ultimately leads to original decision (i.e., primary question/solution of a given problem). For a classification problem, the process follows the structure of a tree where a most informative attribute to classify the data is broken down into hierarchical branches such that the next question to be asked depends on the answer of the current question.

Fig. 12.4 illustrates a simple decision tree in which the problem is to classify interstate data into driver behavior (slow, normal, aggressive) based on the speed thresholds. One of the primary benefits of decision trees is that they are easy to interpret. As illustrated in the figure, from interstate traffic data if we know speed and travel mode for any driver, we can classify that driver into slow, normal, or aggressive driver. For example, if the driver is driving a heavy vehicle with an average speed greater than 60 mph, he or she can be classified as an aggressive driver.

Breiman et al. [20] provided a general framework to construct classification and regression decision trees using the following steps given the training data set with correct labels:

1. Find the most informative attribute and the corresponding threshold to split the parent node. At first, the parent node is the root node.
2. Find the next most informative attributes, for each value of the parent node, along with their threshold to split the current node.
3. If all remaining data points at the end of the current split are of the same label, then the node becomes a leaf node with the associated label.
4. If not, either stop splitting and assign a label to the leaf node, accepting an imperfect decision, or select another attribute and continue splitting the data (and growing the tree) [20].

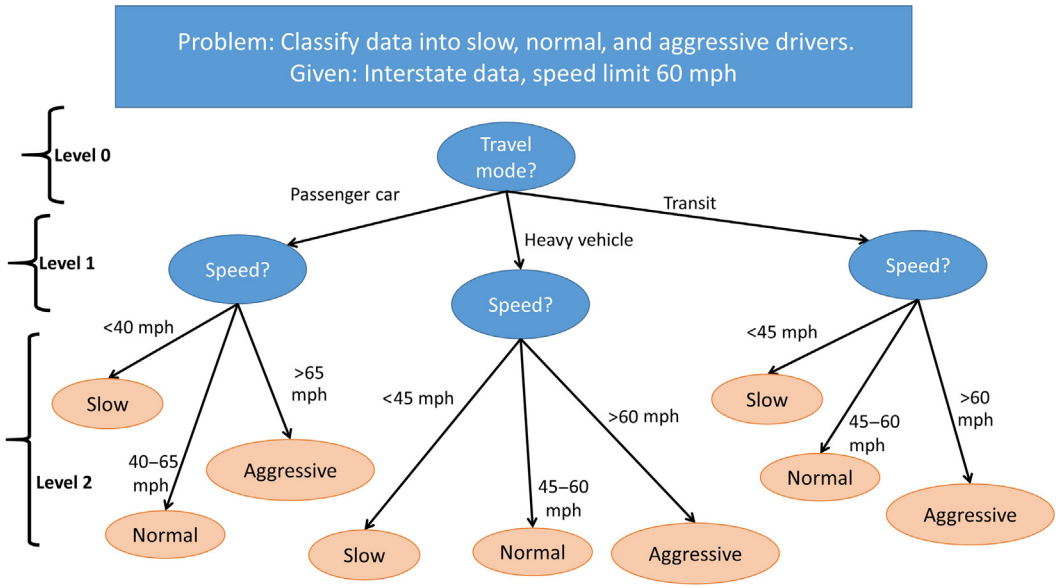

FIGURE 12.4

A simple decision tree that performs classification.

Furthermore, following criteria must be considered to construct the decision tree [18].

1. Attribute test at each level and/or node. As shown in Fig. 12.4 level 1 tests the speed attribute for the data.
2. Number of splits/branching factor. In the above example, the second level is divided into three branches for each node.
3. Stopping criterion.

For a classification decision tree, the level of impurity is measured to evaluate performance of the tree. If the decision tree classifies all data patterns into classes that they actually belong, the splits between class and branches are considered pure. [16]. The impurity between two branches or classes can be computed based on several methods such as entropy-based impurity, Gini impurity, and misclassification impurity [18]. Following equation provides entropy-based impurity.

$$i_{Entropy}(N) = -\sum_{j} P(w_j)\log P(w_j)$$

where $i_{Entropy}(N)$ is the entropy of node N and $P(w_j)$ is the probability of class w_j patterns that arrive at node N [18].

12.4.3 NEURAL NETWORKS

Neural networks (NNs) or artificial neural networks (ANNs) are designed to mimic the functions and architecture of the nervous system [21]. First, introduced in 1943 by McCulloch and Pitts [21], ANNs have gained significant popularity in machine learning and data analytics realm. NNs are extremely powerful tool that have been applied in several transportation applications [22,23]. Similar to the nervous system, the fundamental unit of the ANN is a neuron that utilizes "transfer function" to calculate output for a given input [22]. As illustrated in Fig. 12.5, these neurons are connected to form a network through which data flows (i.e., input layer to other procession layers to output layer). The connections are weighted connections that scale the data flow while transfer functions in each neuron map inputs with outputs.

The general relationship between x inputs and y output can be given as:

$$y_m = f\left(B_m + \sum_{i=1}^{n} W_{im}x_i\right)$$

where y is the output, x_i is ith input from input layer X with n input variable, B_m is the bias for the neuron, W_{im} is connection weight from ith neuron of input layer to mth neuron. The transfer function is typically a nonlinear function such as radial basis or sigmoid.

The structure of ANN typically has three layers: input, hidden, and output. As presented in Fig. 12.6, the hidden layer connects input and output layers with extra set of neurons. For each mth neuron in the hidden layer H, the output is given by:

$$H_m = f\left(B_{hm} + \sum_{i=1}^{n} W_{im}x_i\right)$$

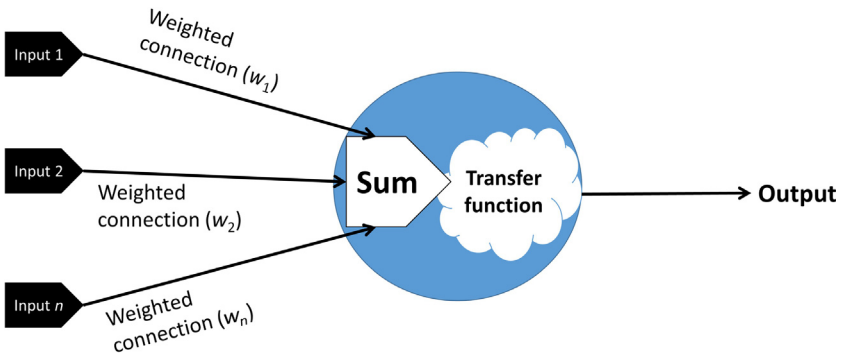

FIGURE 12.5

Neuron: a fundamental processing unit of ANN.

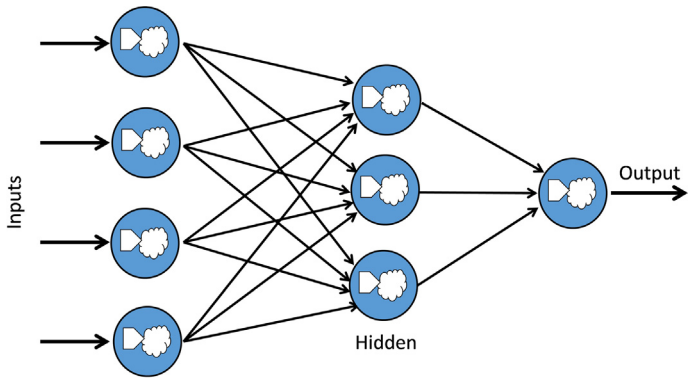

FIGURE 12.6

Typical ANN structure.

where H_m is the output of mth neuron of hidden layer H, B_{hm} is the bias for the neuron in the hidden layer, x_i is ith input from input layer X with nth input variable, and W_{im} is connection weight from ith neuron of input layer to mth neuron of hidden layer.

The output from hidden layer H becomes an input for the next layer, if there is only one hidden layer, H_m becomes an input for the output layer. Thus, prediction from output neuron is given by:

$$P_k = f\left(B_0 + \sum_{i=1}^{n} W_{ik} H_i\right)$$

where P_k is the predicted output from kth neuron of output layer P, B_0 is the bias for the neuron, H_i is ith input from hidden layer H with n hidden neurons, and W_{ik} is connection weight from ith neuron of hidden layer to kth neuron of output layer.

It is evident from these equations that weights, that connect input to hidden and hidden to output layers, are the backbone of the ANN architecture upon which the performance of the algorithm is dependent. The learning (i.e., training) of the ANN algorithm consists of determining the weights using various method. One of the popular methods which is based on gradient descent error minimization is known as backpropagation learning rule [18,19,24]. The backpropagation neural network (BPNN) propagates the error from output layer to input layer, and improves the accuracy by changing the weights and biases. The weights are typically adjusted after each training epoch to minimize the error between target and predicted output. The error function of the network is given in following equation:

$$E = \frac{1}{2}\sum_{i=1}^{n}(T_i - P_i)^2$$

where T_i is the target for ith input pattern and P_i is the predicted output from the network for the same input pattern.

Multilayer perceptron (MLP) NN is implemented in Matlab and the main Matlab functions that we might consider are:

- Feedforwardnet, which creates the network architecture.
- Configure, which sets up the parameters of the network.
- Train, which trains the network.
- Sim, which simulates the network, by computing the outputs for a given test data.
- Perform, which computes the performance of the network test data whose labels are known.

12.4.4 SUPPORT VECTOR MACHINE

Support vector machines (SVMs) are risk-based supervised learning methods that classify data patterns by identifying a boundary with maximum margin between data points of each class [25]. The general idea is to find a function with a set error margin that maps input variables to output variable such that the predicted output does not deviate from the actual output more than the set error margin (Fig. 12.7).

Given the set of labeled data points (input−output pairs) $S=\{(x_i, y_i)\}_{i=1}^{n}$, where $y_i \in \{-1, +1\}$ is the class label of high-dimensional point x_i, i.e., $(x_i, y_i) \in R^{d+1}$, and d and n are the numbers of features and labeled data points, respectively. In binary classification problem, points labeled with $+1$, and -1 belong to classes $\mathbf{C}^+$, and $\mathbf{C}^-$, respectively, i.e., $S = \mathbf{C}^+ \cup \mathbf{C}^-$. The optimal classifier (also known as *soft margin SVM*) is determined by the parameters w and b through solving the convex optimization problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i\left(w^T x_i + b\right) \geq 1 - \xi_i, \quad i = 1, \ldots, n \\ & \xi_i \geq 0, \quad i = 1, \ldots, n \end{aligned}$$

with corresponding linear prediction function $f(x) = wx + b$.

The magnitude of penalization for misclassification is controlled by the parameter C and slack variables ξ_i.

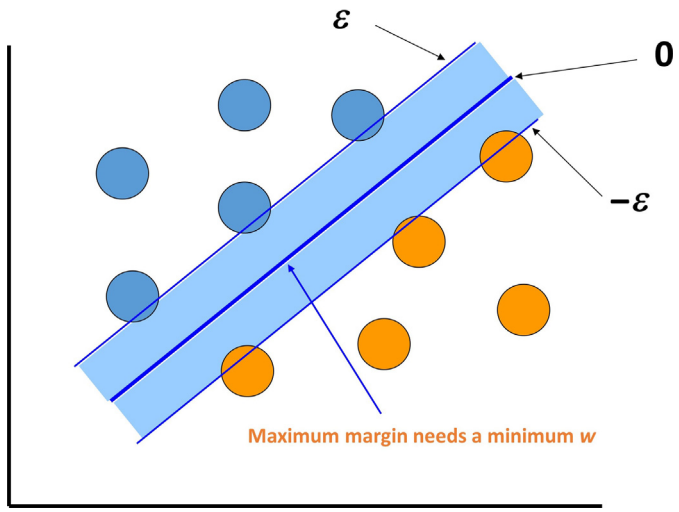

FIGURE 12.7

SVM concept.

In various difficult cases (that are often observed in practice), the data is not separable into two classes by a hyperplane determined by w. A common approach to cope with this problem is to map S to high-dimensional space and find a hyperplane there. Many existing implementations of SVM solve the Lagrangian dual problem [26] instead of the primal formulation above due to its fast and reliable convergence.

When using SVM, one has to be very careful about the types of data, algorithm, and implementation. An imbalanced data (the sizes of $\mathbf{C}^+$, and $\mathbf{C}^-$ can substantially differ from each other) is one of the frequently observed potential problems of SVM models. Correct classification of small class points often becomes more important than classification of others (e.g., in healthcare domain, the number of sick people is less than healthy). In such cases, one should look for a special SVM model that addresses this issue, e.g., weighted SVM that introduces different misclassification penalties for different classes. Determining a correct type of mapping to high-dimensional space can also be extremely important.

Training a typical SVM model becomes time consuming when the number of points or dimensionality is big. Solvers for Lagrangian dual problem typically scale between $O(dn^2)$ to $O(dn^3)$. In order to train classifiers for large-scale data sets, one should consider using strategies such as parallel implementations [27], memory efficient solvers [28], and hierarchical approaches [29]. In Matlab, SVM is implemented with function fitcsvm($\boldsymbol{X; y}$), where $\boldsymbol{X}$ is the matrix of the training observations and $\boldsymbol{y}$ is a vector of the training labels. We can specify the kernel function, kernel scale parameter, initial estimates of Lagrange multipliers, and other parameters.

12.4.5 CLUSTERING

One of the most frequent problems in data analysis is how to partition the data in the absence of labeled examples such that similar data points will belong to the same partitions (or clusters). Two

main classes of such data partitioning algorithms, namely *strict* and *fuzzy* clustering, differ in a type of mapping of a data point to set of partitions. In strict clustering, a point can belong to one partition only, while in fuzzy clustering, a point can appear in several partitions. When choosing a clustering algorithm, one has to understand two key properties of the clustering model, namely,

- what similarity measure does the algorithm use,
- what are the criteria for separation between clusters does the algorithm model and optimize.

Depending on the problem and the algorithm, the number of clusters may or may not be one of the inputs. Typical similarity measures between data points include Euclidian distance, cosine similarity, Hamming distance, mutual information, and Kullback−Leibler divergence. The separation criteria include (among many others) maximization of similarities inside clusters, minimization of similarities between different clusters, and minimization of the distance between cluster elements and cluster centers.

One of the most popular clustering algorithms that has a variety of versions with and without predefined number of clusters as well as a requirement of being strict or fuzzy is called k-means clustering. We describe its basic strict version with a given k (number of clusters).

Given a data set $S=\{x_i\}_{i=1}^{n}$ of size n, and a desired number of clusters k, the goal is to find a many-to-one mapping of S to clusters $C=\{C_1, \ldots, C_k\}$ such that the sum of square distances between data points and centers of their clusters is minimized, namely,

$$\text{minimize overall possible mappings of } S \text{ to } C=\{C_1,\ldots,C_k\} \sum_{j=1}^{k} \sum_{x\in C_j} \|x-\mu_j\|^2$$

where μ_j is the arithmetic mean (or centroid) of all points in cluster C_i. In general, finding an optimal mapping of S to C is computationally difficult problem with no polynomial time algorithm known. However, in practice, several fast approximation and heuristic algorithms can find a good mapping reasonably fast. We refer the reader to Ref. [30] for details on this and other clustering algorithms.

12.4.6 EVALUATION

Evaluating performance of any machine learning model is one of the necessary steps. A fair process of evaluation has to test the model on a data set that was not utilized for training. For this purpose, before any machine learning algorithm is applied to develop a model, the available data set is separated into two sets, namely, training, and testing data set. The machine learning algorithm is trained using the training data set while the testing data set is utilized to evaluate the efficacy (i.e., true performance) of the resulting model. One has to take the following items into account:

- How much data is required for testing and training steps? While it is important to provide enough data variability to train the algorithm, the testing data must provide true performance evaluation.
- What would be a good split of data for testing and training? Since the testing data is removed, the training data may not represent variability that is required for successful learning algorithm.
- Similarly, the testing data may provide results that are “too good to be true” or inaccurate even though the learning algorithms have sufficient training data.

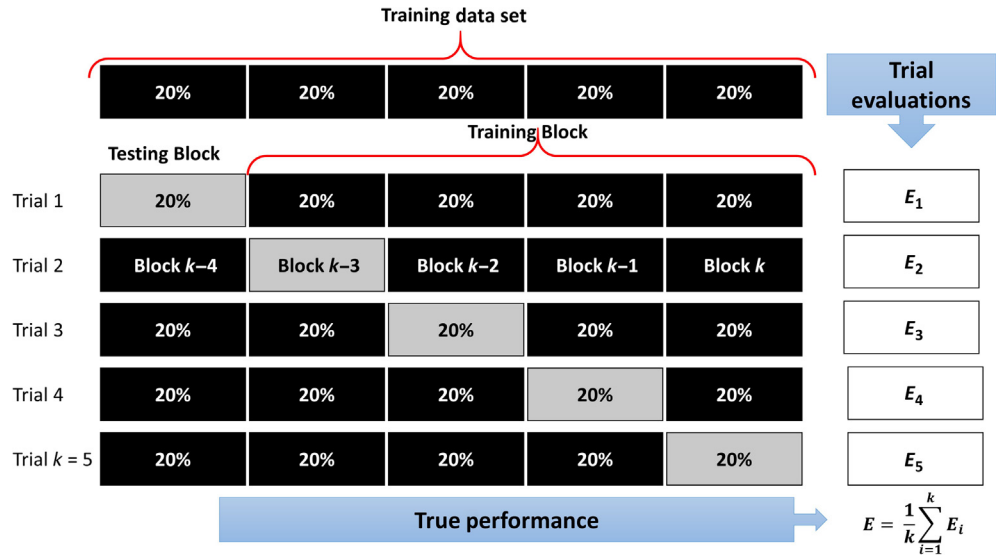

FIGURE 12.8

Cross validation approach, an example for $k = 5$.

The k-fold cross validation is one of approaches widely accepted and used in machine learning systems to estimate the true performance. In this approach, the entire available data set is divided into k (usually equal size) blocks of data. Of these blocks, one block is used for testing and the remaining are used for training. This procedure is repeated k times, using a different block for testing to obtain k models. The average performance of obtained model tests is considered as the true performance of the algorithm. The process is illustrated in Fig. 12.8.

This approach has several important advantages.

- All data is utilized for training and testing, but never at the same time.
- Repeating the procedure k times reduces the probability of having an unusually lucky or unlucky training/testing data.
- The performance can be statistically validated using two-tailed z-test ($k > 30$), or a two-tailed t-test ($k < 30$).

12.5 AN EXAMPLE

The following example presents performance of three machine learning methods in predicting speed of the roadway. The traffic data was generated using traffic micro-simulation software VISSIM and provided in an excel file ("Data.xlsx"). The Matlab code for this experiment is provided in the Appendix section. Machine learning algorithms (1) SVM, (2) decision tree, and (3) ANN were used to predict speed. The data was divided into training and testing patterns. For

(A)

Predicted speed by SVM

True speed	Predicted speed
32.59	32.612
32.5	32.556
32.16	32.364
32.2	32.366
32.36	32.502
32.01	32.305
32.15	32.405
32.05	32.33
32.09	32.352
32.41	32.528

Prediction accuracy 0.9234

Prediction error 0.0766

Predicted speed by decision tree

True speed	Predicted speed
32.59	32.49
32.5	32.49
32.16	31.83
32.2	32.232
32.36	32.255
32.01	32.095
32.15	32.112
32.05	32.095
32.09	32.095
32.41	32.558

Prediction accuracy 0.7085

Prediction error 0.2915

Predicted speed by MLP NN

True speed	Predicted speed
32.59	32.584
32.5	32.516
32.16	32.159
32.2	32.252
32.36	32.32
32.01	31.99
32.15	32.135
32.05	32.028
32.09	32.062
32.41	32.368

Prediction accuracy 0.9983

Prediction error 0.0017

(B)

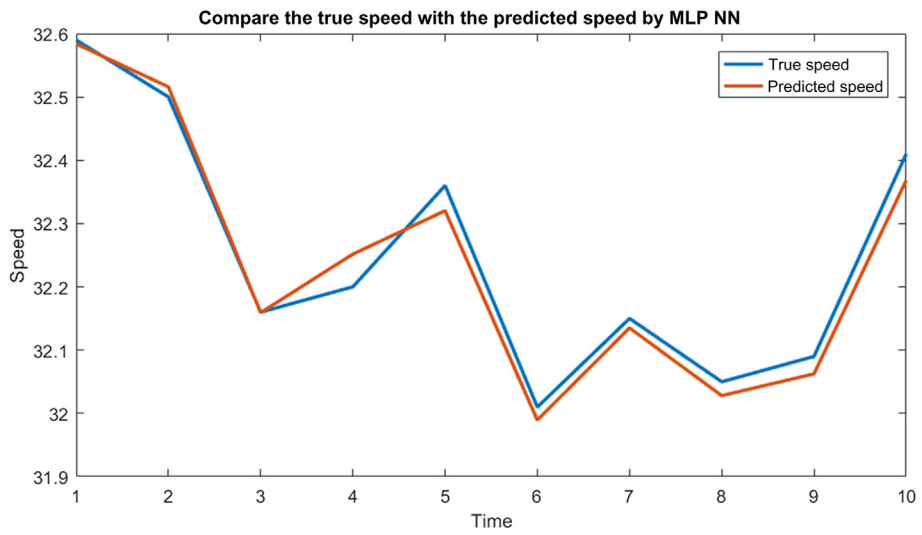

FIGURE 12.9

Performance of machine learning algorithms.

each learning algorithm, the input variables were volume (vehicle/hour) and density (vehicle/mi), and output variable was speed (miles/hour). Fig. 12.9 presents performance of each algorithm.

As presented in these charts, the ANN model performed better than other two models. However, a quick look at the data patterns in the Excel worksheet suggests that the data does not represent uncertainties/variability as the real-world speed prediction model will experience during the day. The histogram of speed developed from the available data is presented in Fig. 12.10. This histogram suggests that speed is varying between 31 and 34 mph. While this

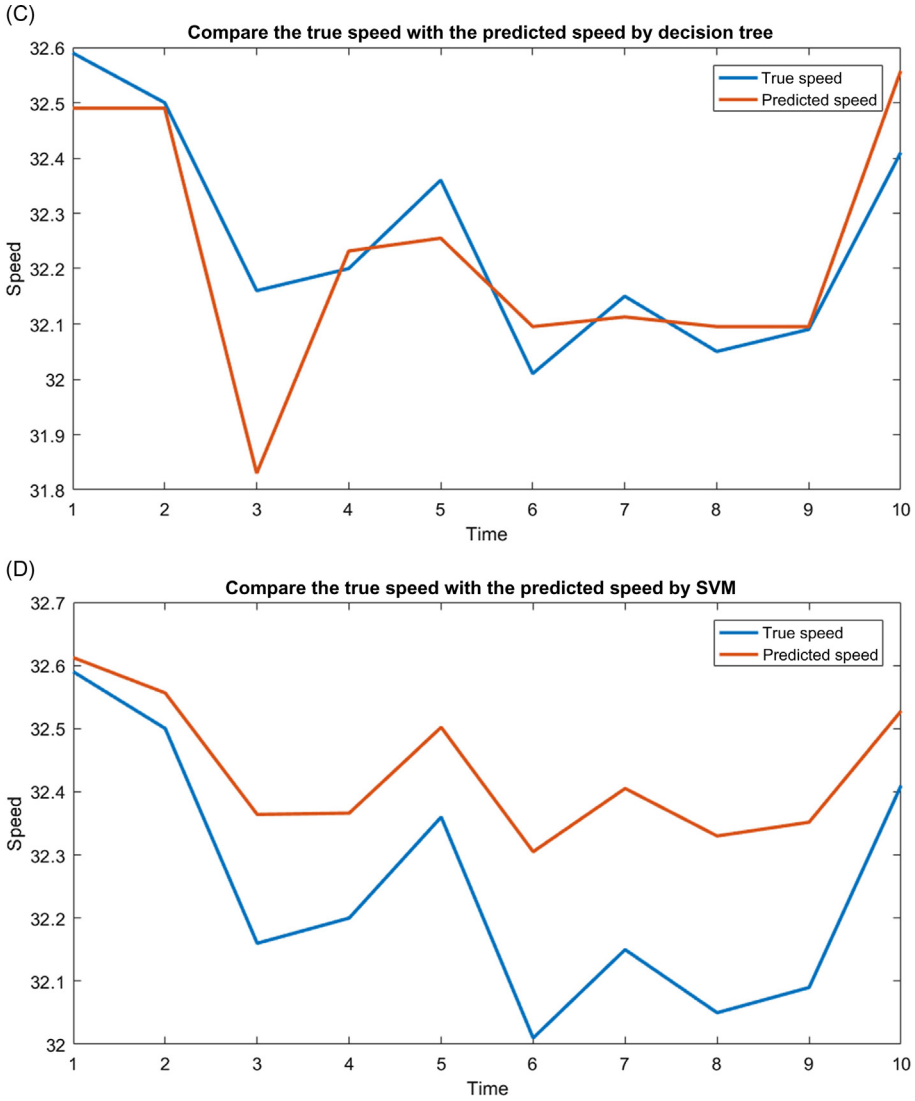

FIGURE 12.9

(*Continued*)

model will be very accurate, it may not perform well when target is outside this range. This suggests that even though the Excel worksheet has 600 data patterns for machine learning algorithm, the final application may require more experience (i.e., training and data) before implementation.

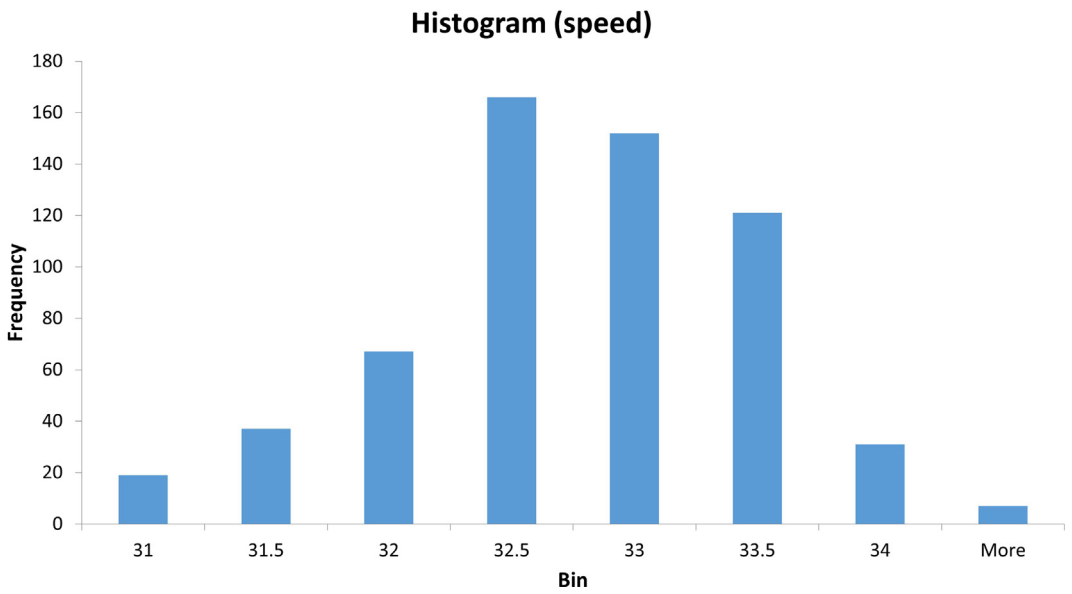

FIGURE 12.10

Distribution of output.

12.6 SUMMARY

This chapter introduced machine learning methods for data analytics in transportation. Machine learning methods' learning algorithm(s) is(are) being utilized and data being presented to the learning algorithm(s). While machine learning methods can significantly improve data analytics for the transportation system, careful consideration must be given to problem definition and understanding the available testing and training data set(s).

12.7 QUESTIONS AND SOLUTIONS

1. What is the difference between supervised and unsupervised learning methods?
2. What is the difference between linear and polynomial regressions?
3. What are the essential steps of data preprocessing before applying a machine learning method?
4. What approach can be used for a correct validation of the true performance of the learned model?
5. Select a transportation data set with at least one output parameter and three or more input parameters.
   a. Split data set into training (80% of total data patterns) and testing (20% of total data patterns)

b. Use one of the regression methods and an NN approach and develop two different prediction models.
c. Use only training data set for cross validation method and calculate true performance of both models. DO NOT use testing data set.
d. Evaluate both models with testing data set.
e. Compare results obtained in c and d.
f. Present your conclusion, observations, and recommendations.

REFERENCES

[1] T. Ross, The synthesis of intelligence—its implications, Psychol. Rev. 45 (2) (1938) 185.
[2] A.L. Samuel, Some studies in machine learning using the game of checkers, IBM J. Res. Develop. 3 (3) (1959) 210–229.
[3] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, C. Chen, Data-driven intelligent transportation systems: a survey, IEEE Trans. Intell. Transport. Syst. 12 (4) (2011) 1624–1639.
[4] R. Agrawal, T. Imieliński, A. Swami, Mining association rules between sets of items in large databases, ACM SIGMOD Record, No. 22, ACM, New York, NY, 1993, pp. 207–216.
[5] Meyer, M., and E. Miller. Urban transportation planning: a decision-oriented approach, 2001.
[6] L. Tarassenko, Guide to Neural Computing Applications, Butterworth-Heinemann, Oxford, UK, 1998.
[7] F. Hasson, S. Keeney, H. McKenna, Research guidelines for the Delphi survey technique, J. Adv. Nurs. 32 (4) (2000) 1008–1015.
[8] A.K. Jain, R.P.W. Duin, J. Mao, Statistical pattern recognition: a review, IEEE Trans. Pattern. Anal. Mach. Intell. 22 (1) (2000) 4–37.
[9] R. Polikar, Ensemble based systems in decision making, IEEE Circuits Syst. Mag. 6 (3) (2006) 21–45.
[10] H.F. Durrant-Whyte, Sensor models and multisensor integration, Int. J. Robot. Res. 7 (6) (1988) 97–113.
[11] F. Castanedo, A review of data fusion techniques, Sci. World J. 2013 (2013).
[12] M.T. Heath, Scientific Computing, McGraw-Hill, New York, NY, 2002.
[13] F. Marczak, C. Buisson, New filtering method for trajectory measurement errors and its comparison with existing methods, Transport. Res. Record J. Transport. Res. Board (2315) (2012) 35–46.
[14] P.J. Rousseeuw, A.M. Leroy, Robust Regression and Outlier Detection, John Wiley & Sons, New York, NY, 2005.
[15] J. Friedman, T. Hastie, R. Tibshirani, The Elements of Statistical Learning, Springer, Berlin, 2001.
[16] Mogha, P., N. Sharma, and S. Sharma. Big Data. IJRIT Int. J. Res. Infor. Technol.—White Paper, 2013.
[17] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, et al., Top 10 algorithms in data mining, Knowl. Inform. Syst. 14 (1) (2008) 1–37.
[18] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, John Wiley & Sons, New York, NY, 2012.
[19] E. Alpaydin, Introduction to Machine Learning, MIT Press, 2014.
[20] L. Breiman, J. Friedman, C.J. Stone, R.A. Olshen, Classification and Regression Trees, CRC press, 1984.
[21] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (4) (1943) 115–133.
[22] M. Dougherty, A review of neural networks applied to transport, Transport. Res. Part C Emerg. Technol. 3 (4) (1995) 247–260.
[23] M.G. Karlaftis, E.I. Vlahogianni, Statistical methods versus neural networks in transportation research: differences, similarities and some insights, Transport. Res. Part C Emerg. Technol. 19 (3) (2011) 387–399.

[24] P.J. Werbos, Backpropagation through time: what it does and how to do it, Proc. IEEE 78 (10) (1990) 1550–1560.
[25] V. Vapnik, The Nature of Statistical Learning Theory, Springer Science & Business Media, New York, NY, 2013.
[26] R. Fletcher, Practical Methods of Optimization, John Wiley & Sons, New York, NY, 2013.
[27] K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, H. Cui, et al., Parallelizing support vector machines on distributed computers, Adv. Neural. Inf. Process. Syst. (2008) 257–264.
[28] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Transac. Intell. Syst. Technol. 2 (3) (2011) 27.
[29] T. Razzaghi, I. Safro, Scalable multilevel support vector machines, Proc. Comp. Sci. 51 (2015) 2683–2687.
[30] C.C. Aggarwal, C.K. Reddy, Data Clustering: Algorithms and Applications, CRC Press, 2013.

APPENDIX

```
%--- Code for matlab example--- excel file is provided in XXXXXXX
filename=(['Data','.xlsx']);
A = xlsread(filename);
X=zeros(601,2);
X(:,1)=A(:,8);
X(:,2)=A(:,6);
Y=A(:,7);

K=2;
c = cvpartition(Y,'KFold',K);
for k=1:K

  X_training=X(c.training(k),:);
  X_test=X(c.test(k),:);
  Y_training=Y(c.training(k));
  Y_test=Y(c.test(k));
  Ytest = Y_test';
  Xtrain=X_training';
  Ytrain=Y_training';
  Xtest=X_test';
  %-------------------------[Support vector machine]-----------
  svm_mdl = fitrsvm(X_training,Y_training,'Standardize',true);
  svm_yfit = predict(svm_mdl,X_test);
  svm_compare= abs(Y_test - svm_yfit)< 0.5;
  svm_accuracy(k) = sum(sum(svm_compare)) / numel(svm_yfit) ;
  %------------------------[Decision tree]------------

  tree_mdl = fitrtree(X_training(:,:),Y_training);
  tree_yfit = predict(tree_mdl,X_test(:,:));
  tree_compare= abs(Y_test - tree_yfit)< 0.5;
  tree_accuracy(k) = sum(sum(tree_compare)) / numel(tree_yfit) ;
```

```
    %------------------[Artificial neural network]------------------

    MLP_hiddenLayerSize = 20;
    MLP_net = feedforwardnet(MLP_hiddenLayerSize, 'trainlm');
    MLP_net.inputs{1}.processFcns={};
    MLP_net.outputs{2}.processFcns={};
    MLP_net.divideFcn = 'dividetrain';
    MLP_net.trainParam.epochs = 200; % Maximum number of epochs to train
    MLP_net.trainParam.goal = 0.01; % Performance goal
    MLP_net.trainParam.max_fail = 60; % Maximum validation failures
    MLP_net.trainParam.mc = 0.9; % Momentum constant
    MLP_net.trainParam.min_grad = 1e-10; % Minimum performance gradient
    MLP_net.trainParam.show = 10; % Epochs between displays
    MLP_net.trainParam.showCommandLine = 0; % Generate command-line output
    MLP_net.trainParam.showWindow = 1; % Show training GUI
    MLP_net.trainParam.time = inf; % Maximum time to train in seconds
    MLP_net.trainParam.Lr = 0.001; % Learninig Rate
    MLP_net.layers{1}.transferFcn = 'logsig';
    MLP_net.layers{2}.transferFcn = 'purelin';
    %train the network
    [MLP_net,train_record] = train(MLP_net,Xtrain,Ytrain);
    Network_output=MLP_net(Xtest);
    MLP_Comparision=abs(Network_output-Ytest)<0.5;
    MLP_Accuracy(k)=(sum(sum(MLP_Comparision))/numel(Network_output));

end
% -----[code to generate SVM figure]------
figure;
x=1:sum(Y_test);
plot(x(1:10),Y_test(1:10),x(1:10),svm_yfit(1:10),'LineWidth',2)
legend('True Speed','Predicted Speed');
title('Compare the True Speed with the Predicted Speed by SVM');
ylabel('Speed')
xlabel('Time')
values = table(Y_test,svm_yfit,'VariableNames',{'True_Speed','Predicted_Speed'});
values(1:10,:)

ts = tinv([0.025 0.975],K-1);
svm_accuracy=sum(svm_accuracy)/K;
svm_performance_CI = ts(1,2)*std(svm_accuracy)/sqrt(K);
svm_error=1-svm_accuracy;
svm_error_CI= ts(1,2)*std(svm_error)/sqrt(K);

disp(svm_accuracy);
disp(svm_error);
% -----[code to generate Decision Tree figure]------
```

```
figure;
x=1:sum(Y_test);
plot(x(1:10),Y_test(1:10),x(1:10),tree_yfit(1:10),'LineWidth',2)
legend('True Speed','Predicted Speed');
title('Compare the True Speed with the Predicted Speed by Decision Tree');
ylabel('Speed')
xlabel('Time')
values = table(Y_test,tree_yfit,'VariableNames',{'True_Speed','Predicted_Speed'});
values(1:10,:)

ts = tinv([0.025 0.975],K-1);
tree_accuracy=sum(tree_accuracy)/K;
tree_performance_CI = ts(1,2)*std(tree_accuracy)/sqrt(K);
tree_error=1-tree_accuracy;
tree_error_CI= ts(1,2)*std(tree_error)/sqrt(K);

disp(tree_accuracy);
disp(tree_error);

% -----[code to generate ANN figure]------
figure;
x=1:sum(Y_test);
plot(x(1:10),Ytest(1:10),x(1:10),Network_output(1:10),'LineWidth',2)
legend('True Speed','Predicted Speed');
title('Compare the True Speed with the Predicted Speed by MLP Neural Network');
ylabel('Speed')
xlabel('Time')
values=table(Ytest',Network_output','VariableNames',
{'True_Speed','Predicted_Speed'});
values(1:10,:)

ts = tinv([0.025 0.975],K-1);
MLP_Accuracy=sum(MLP_Accuracy)/K;
MLP_performance_CI = ts(1,2)*std(MLP_Accuracy)/sqrt(K);
MLP_error=1-MLP_Accuracy;
MLP_error_CI= ts(1,2)*std(MLP_error)/sqrt(K);

disp(MLP_Accuracy);
disp(MLP_error);
```