Chapter

1

Introduction to Machine Learning

THE AWS CERTIFIED MACHINE LEARNING (ML) ENGINEER ASSOCIATE EXAM OBJECTIVES COVERED IN THIS CHAPTER MAY INCLUDE, BUT ARE NOT LIMITED TO THE FOLLOWING:

✔ **Domain 2: ML Model Development**

- 2.1 Choose a modeling approach
- 2.2 Train and refine models

Machine learning (ML) has become ubiquitous in our digital world. Whether you need to book a flight, visit your doctor, make an online purchase, pay a bill, or check the weather forecast, behind the scenes any of these actions has started (or is a part of) a process that collects large amounts of data, processes the data, and performs some ML task.

ML is a branch of artificial intelligence (AI) that enables systems to learn and improve from experience without being explicitly programmed.[1] By analyzing large datasets, ML algorithms can identify patterns, make decisions, and predict outcomes.

The integration of ML into various domains has revolutionized industries by enhancing efficiency, accuracy, and decision-making capabilities.

This chapter will provide the ML foundations you need to know for the exam. To better understand ML, we need to set the context where ML originated, which is AI.

Understanding Artificial Intelligence

AI is a branch of computer science whose main focus is to develop systems capable of performing tasks that typically require human intelligence. These tasks include (but are not limited to) recognizing speech, making decisions, solving problems, identifying patterns, and understanding languages. AI systems leverage techniques such as ML, natural language processing (NLP), deep learning, and computer vision to simulate cognitive functions like learning, reasoning, and self-correction. The field of AI is rapidly evolving, with applications spanning various industries, from healthcare and finance to autonomous vehicles and smart cities. Understanding AI involves exploring its fundamental concepts, its historical development, and the ethical implications of its widespread adoption.

AI systems ingest data, such as human-level knowledge, and emulate natural intelligence. ML is a subset of AI, where data and algorithms continuously improve the training model to help achieve higher-quality output predictions. Deep learning is a subset of ML. It is an approach to realizing ML that relies on a layered architecture, simulating the human brain to identify data patterns and train the model.

[1] An early ML pioneer and computer scientist, Arthur Samuel, coined the term *machine learning* in 1959 to denote the "field of study that gives computers the ability to learn without being explicitly programmed." Thirty-eight years later, another computer scientist, Tom Mitchell, defined machine learning as "the study of computer algorithms that allow computer programs to automatically improve through experience."

FIGURE 1.1 Deep learning, machine learning, and AI.

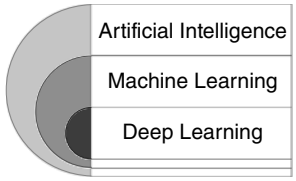


Figure 1.1 illustrates this hierarchy. It all starts with data and how data can be used to extract relevant information, which ultimately produces knowledge.

Data, Information, and Knowledge

Data, information, and knowledge form the foundation of understanding and applying AI. *Data* is the raw, unprocessed facts and figures collected from various sources. When data is organized and processed, it transforms into *information*, which provides context and meaning. *Knowledge* is derived from the synthesis of information, enabling comprehension and informed decision-making.

In AI, data serves as the essential input that fuels ML algorithms, enabling the training of models to recognize patterns and make predictions. Information, structured from the data, helps refine these models by offering insights and context. Ultimately, knowledge empowers AI systems to emulate human-like reasoning, enhancing their ability to perform complex tasks, adapt to new situations, and provide valuable solutions across various domains.

Data

Data is defined as a value or set of values representing a specific concept or concepts. Data is the most critical asset of the digital age, fueling advancements in technology and shaping the way we understand and interact with the world. To harness its full potential, it's essential to recognize the different classes of data: structured, semi-structured, and unstructured. Each class has unique characteristics and applications, particularly in the context of AI.

Understanding these classes of data and their peculiarities is crucial for leveraging AI's full potential. Structured data provides the foundation for organized analysis, semi-structured data offers flexibility in handling complex information, and unstructured data unlocks a wealth of untapped insights. Together, they enable AI systems to learn, adapt, and deliver transformative solutions across various domains. Let's review each class in more detail.

Structured Data

Structured data is highly organized and easily searchable in databases. This type of data adheres to a fixed schema, meaning it follows a predefined format with specific fields and records. For example, a spreadsheet containing customer information, such as names, addresses, and purchase histories, is structured data. It allows for efficient querying and analysis, making it invaluable for business intelligence and data-driven decision-making.

These are examples of structured data:

- Relational databases (e.g., SQL databases)
- Spreadsheets (e.g., Excel files)
- Financial records (e.g., transaction logs)

Semi-Structured Data

Semi-structured data lacks the rigid structure of structured data but still contains some organizational elements, such as tags or markers, that provide context and hierarchy. This class of data is often used for storing and transmitting complex information that doesn't fit neatly into a table format. Semi-structured data is more flexible than structured data, allowing for greater adaptability in handling diverse data types.

These are examples of semi-structured data:

- JavaScript Object Notation (JSON) files
- Extensible Markup Language (XML) documents
- Email messages (headers, body text, attachments)

Unstructured Data

Unstructured data is the most common and diverse type of data, encompassing information that doesn't follow a predefined format or schema. This class of data is often rich in content but challenging to analyze and search. Unstructured data requires advanced AI techniques, such as NLP and computer vision, to extract meaningful insights.

These are examples of unstructured data:

- Text files (e.g., Word documents)
- Multimedia files (e.g., images, videos)
- Social media content (e.g., tweets, posts)

Information

Information bridges the gap between raw data and actionable knowledge. Derived from data, information provides a semantic element in the form of context and meaning, resulting in a transformation of disparate facts and figures into coherent, useful insights.

Information can be understood as data that has been processed, organized, or structured in a way that adds context and relevance. Unlike raw data, which is often unorganized and lacks inherent meaning, information is data presented in a format that is understandable and useful to its recipient(s). For example, a list of temperatures recorded at various times of the day is mere data; when these temperatures are organized into a table showing daily weather patterns, they become information.

The key characteristics of information are accuracy, relevance, completeness, and timeliness:

Accuracy Information must be precise and free from errors to be valuable. Inaccurate information can lead to misguided decisions and outcomes.

Relevance Information should be pertinent to the context or problem at hand. Irrelevant information, no matter how accurate, serves little purpose.

Completeness Information must be comprehensive enough to provide a clear understanding without ambiguity. Incomplete information can result in incorrect conclusions.

Timeliness Information is most useful when it is available at the right time. Outdated information can be as detrimental as inaccurate information.

In AI, information plays a critical role in training and refining algorithms. AI systems rely on vast amounts of data to learn and make predictions. This data is processed into information that adds context and aids in pattern recognition and decision-making.

For instance, in NLP, raw text data is processed to extract meaningful information, such as sentiment analysis or language translation. Similarly, in computer vision, images are analyzed to identify objects and patterns, turning visual data into actionable information.

Knowledge

Knowledge represents a higher level of understanding that goes beyond mere data and information. It encompasses the insights, experiences, and contextual understanding that enable individuals and systems to make informed decisions, solve problems, and innovate.

Knowledge is derived from the synthesis and application of information. It involves recognizing patterns, understanding relationships, and drawing conclusions based on experience and context. Unlike data, which is raw and unprocessed, or information, which is organized and meaningful, knowledge embodies a deeper comprehension that guides action and thought.

The key characteristics of knowledge are contextuality, applicability, basis in experience, and dynamism.

Contextuality Knowledge is deeply rooted in context. It involves understanding not just the facts but also the circumstances and nuances that surround them.

Applicability Knowledge is practical. It involves the ability to apply information to real-world situations, making it actionable and relevant.

Experience-based Knowledge is often gained through experience. It encompasses lessons learned, insights gained, and the wisdom accumulated over time.

Dynamism Knowledge is ever-evolving. As new information becomes available and experiences accumulate, knowledge grows and adapts.

In AI, knowledge is crucial for developing systems that can derive inferences, learn, and adapt. AI systems rely on vast amounts of data and information to build knowledge bases that enable them to perform complex tasks and make intelligent decisions.

For example, expert systems in AI were designed to emulate human expertise in specific domains. These systems utilized knowledge bases that contained rules, facts, and relationships, allowing them to provide recommendations and solve problems. Similarly, ML algorithms use data and information to build models that capture knowledge about patterns and trends, enabling predictive analytics and decision-making.

Understanding Machine Learning

ML is a subset of AI that allows computers to learn from existing information, *without being explicitly programmed*, and apply that learning to perform other similar tasks.

Without explicit programming, the machine learns from data and information it feeds. The machine picks up patterns, trends, or essential features from previous data and makes predictions on new data. This aspect of ML, i.e., a machine's ability to learn *without being explicitly programmed*, is important and emphasizes the key difference between ML and classical programming.

Let's say you are tasked by your manager to build a program that translates text from English to Italian. At a very high level, you could pursue a classical programming approach by manually coding rules and exceptions for lexicon, grammar, syntax, and vocabulary. This approach would be extremely complex and not scalable. Instead, with ML, you could leverage models that excel at understanding context, idioms, and nuances in both languages, which are essential for accurate translation. These ML models can continuously improve their translation capabilities with more data, adapting to new expressions, idioms, and evolving language usage.

A real-life ML application is in recommendation systems. For example, Amazon uses ML to recommend books to users based on users' book categories selection and purchase history. Likewise, Spotify uses ML to recommend songs to users based on previously purchased songs and genres to their liking. In both cases, Amazon and Spotify use large amounts of data to train their ML models and derive meaningful inferences in the form or recommendations.

In the upcoming sections, the fundamental concepts of ML will be introduced. These concepts will form the basis you need to master each exam objective. Let's start with the ML lifecycle.

ML Lifecycle

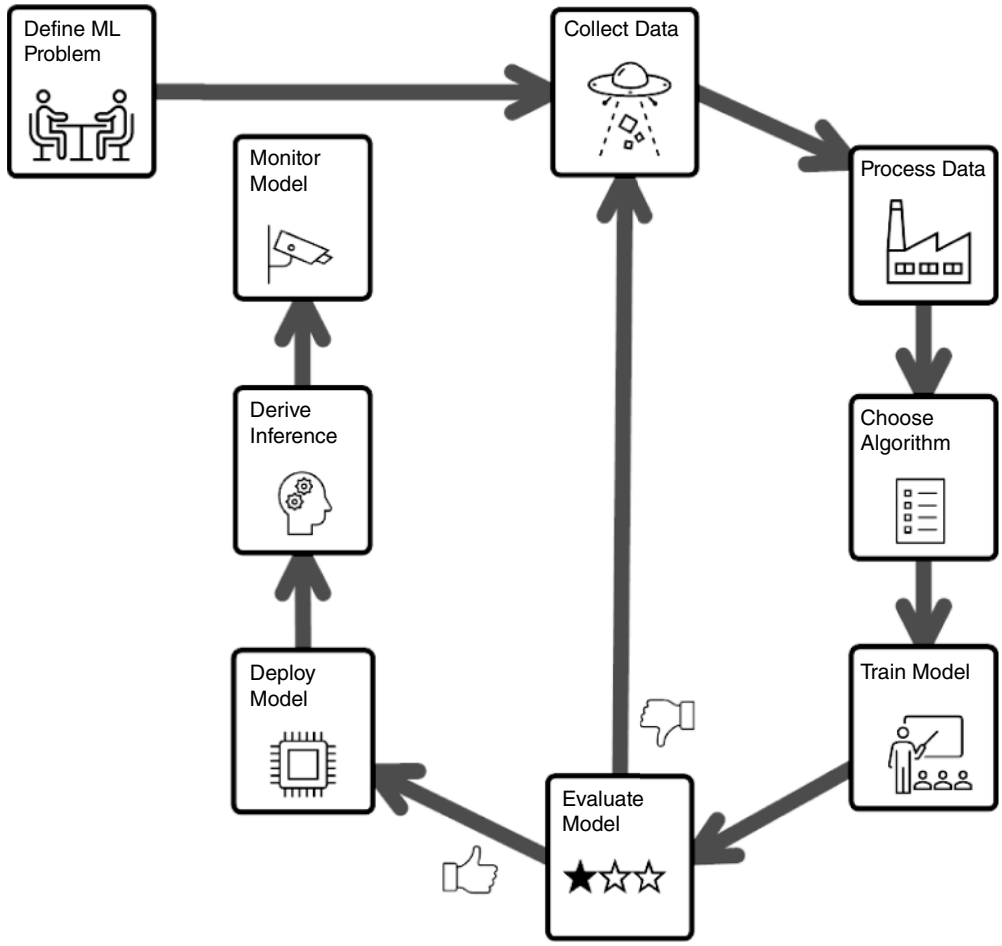
First, you must have a holistic view of the entire ML lifecycle. Because ML is deeply grounded in science, it all starts by questions like "What is the business problem we are trying to solve?" and, most importantly, "How does machine learning address this business need?"

Figure 1.2 illustrates the ML lifecycle. These steps are generally followed in all ML projects, regardless of the cloud provider or the tools in use. Let's review each step.

Define ML Problem

Defining a ML problem is the critical first step in the development of any ML project. It involves understanding and clearly articulating the specific business challenge or question that needs to be addressed using data and ML techniques. This phase includes identifying the problem's scope, the desired outcomes, and the feasibility of applying ML solutions. Crucially, it requires a detailed analysis of the available data, understanding the business

FIGURE 1.2 Machine learning lifecycle.



context, and determining the performance metrics that will be used to evaluate success. By establishing a clear and well-defined problem statement, you can ensure that your efforts are focused and aligned with strategic goals, leading to more effective and impactful ML solutions.

What sets an ML problem apart from other problems in computer science/engineering is its reliance on data-driven approaches and statistical models to make predictions or decisions. Traditional programming relies on explicit instructions and algorithms to solve problems, whereas ML leverages patterns and relationships within the data to infer solutions. This shift from *deterministic* algorithms to *probabilistic models* means that ML problems often involve uncertainty and require continuous learning and adaptation from new data, making them

uniquely dynamic and challenging. This aspect of continuous learning and adaptation from new data is reflected in the cycle represented in Figure 1.2.

Collect Data

The "Collect Data" phase involves gathering the raw data needed for training and testing models. The types of data collected should align with the specific ML problem being addressed. For instance, if the goal is to predict housing prices, numerical data like square footage and year built would be essential. For a customer segmentation task, categorical data like customer demographics and purchase behavior would be relevant. Ensuring that the collected data is relevant and representative of the problem domain sets the foundation for effective model building. Initial efforts should focus on acquiring comprehensive and high-quality data, as this will significantly impact the overall success of the project.

Process Data

The collected data is in a raw state as a result of being ingested from different data sources. In this format, the dataset is not suitable for modeling. First, the dataset may contain missing or noisy data. Second, different data may be in different ranges. Third, not all collected data may be relevant to the model's ability to derive accurate predictions (as you will learn later, also called the target variable, or label). Last, the dataset must be split in three sets: the training, validation, and test datasets.

As a result, the collected data must be adequately transformed to be suitable for model training and evaluation. The "Process Data" step includes normalizing, scaling, and encoding data to ensure consistency and comparability across features. Feature engineering, on the other hand, involves creating new features or modifying existing ones to enhance the model's predictive power. Techniques such as binning numerical data, creating interaction terms, and extracting datetime features are commonly used. The goal of this phase is to clean, refine, and enrich the data, making it ready for effective model training and evaluation.

Choose Algorithm

In the "Choose Algorithm" phase of the ML lifecycle, the focus shifts to selecting the most appropriate algorithm to address the defined problem. This decision is critical because different algorithms are tailored to different types of tasks, such as classification, regression, clustering, or reinforcement learning. Factors like the nature of the data, the complexity of the problem, and the desired outcomes play a significant role in this selection process. For instance, linear regression might be chosen for a straightforward predictive analysis, whereas a deep neural network could be more suitable for complex image recognition tasks.

This phase involves evaluating the strengths and limitations of various algorithms, considering aspects such as accuracy, interpretability, training time, and scalability. It may also include experimenting with multiple algorithms and using cross-validation techniques to compare their performance on the processed data. The goal is to identify the algorithm that

offers the best balance of performance and feasibility, ensuring that the model can effectively learn from the data and make accurate predictions. This informed selection sets the stage for the subsequent phases, where the chosen algorithm will be trained, validated, and fine-tuned to achieve optimal results.

Train Model

The "Train Model" phase is a pivotal stage in the ML lifecycle, following the selection of an algorithm. During this phase, the chosen algorithm is applied to the training data to identify patterns and learn the underlying relationships. This process involves feeding the data into the model and iteratively adjusting the model's parameters to minimize errors, assess bias-variance trade-offs, and improve predictive accuracy. Various techniques, such as gradient descent and backpropagation, are employed to optimize the model's performance.

The goal of training the model is to create a robust and accurate predictive model that generalizes well to new, unseen data. This phase is critical because it determines the model's capability to make reliable predictions and drive decision-making in real-world applications. The effectiveness of this phase hinges on the quality and representativeness of the training data, as well as the proper tuning of hyperparameters to avoid overfitting or underfitting.

Evaluate Model

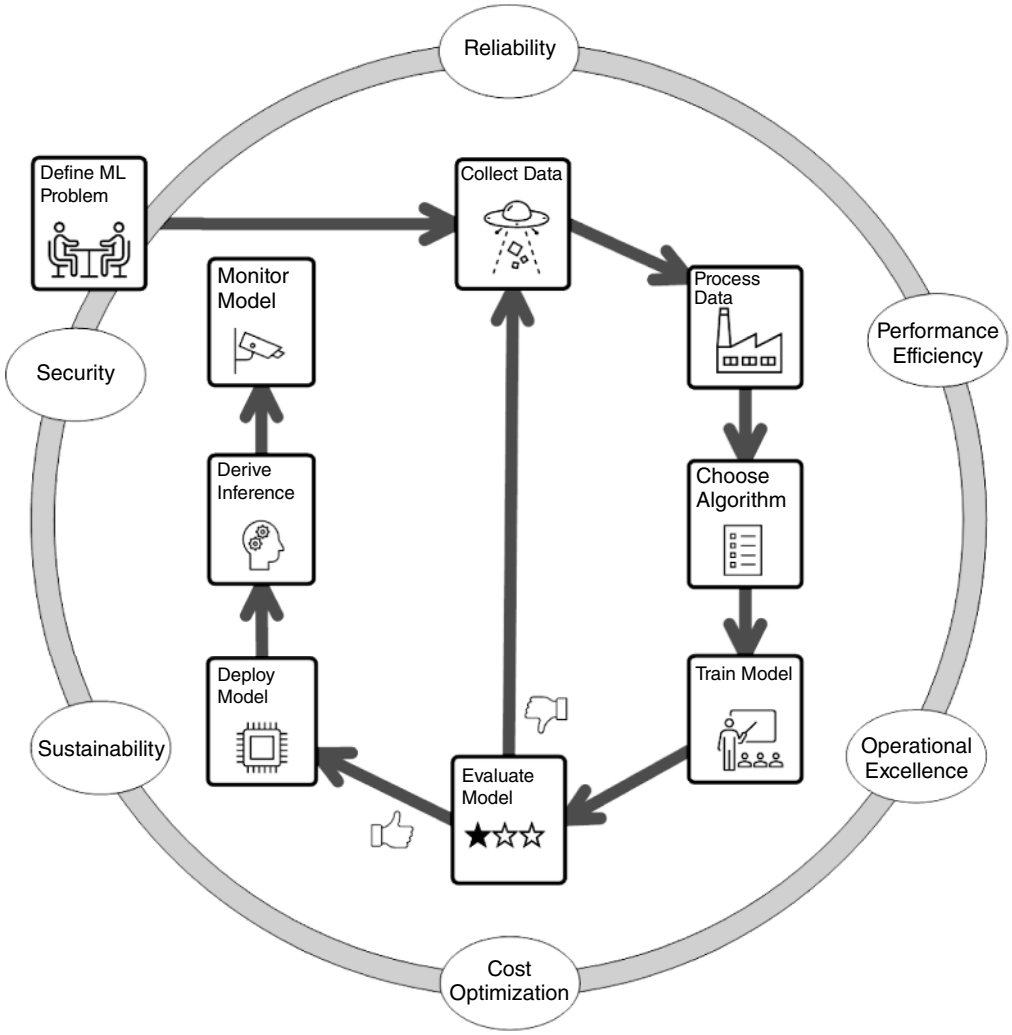
After training, the model is ready for the evaluation phase, whose goal is to assess its performance on a separate *validation dataset* and to ensure that it generalizes well to new, unseen data. Key performance metrics such as accuracy, precision, recall, F1 score, and mean squared error are calculated to gauge the model's effectiveness. This evaluation helps identify any issues like *overfitting* or *underfitting* and provides insights into areas where the model might need improvement.

Evaluation also includes visualizing results through confusion matrices, receiver operating characteristic (ROC) curves, and other diagnostic plots to understand the model's strengths and weaknesses. Based on these evaluations, iterative refinement and tuning of the model may be conducted to enhance its predictive performance. This phase ensures that the model meets the desired criteria and is ready for deployment in real-world applications.

Deploy Model

After you've trained, tuned, and evaluated your model, you can deploy it into production and let your ML application consume it through an endpoint to make predictions. The deployment phase is all about making the trained and validated model available for use in real-world ML applications. It encompasses setting up an infrastructure to host the model, ensuring scalability, reliability, security, sustainability, and cost-effectiveness, which are all pillars of the well-architected ML framework. Figure 1.3 illustrates how each pillar maps to each phase.

FIGURE 1.3 Well-architected ML lifecycle.



The "Deploy Model" phase in AWS leverages Amazon SageMaker to seamlessly transition from evaluation to deployment. This phase involves using specific SageMaker services tailored to deployment needs.

For real-time predictions, *Amazon SageMaker Real-Time Inference* endpoints are ideal, offering low-latency responses suitable for applications requiring immediate predictions, such as interactive web apps or chatbots. For scenarios where batch processing is sufficient, *Amazon SageMaker Batch Transform* provides an efficient way to handle large datasets, making predictions in a cost-effective manner. Additionally, for workloads with fluctuating

traffic, *Amazon SageMaker Serverless Inference* dynamically scales to meet demand without managing servers.

These Amazon SageMaker services ensure that the model remains highly available and performant in production. The deployment phase focuses on making the trained model accessible and usable in real-world applications, ensuring that it delivers value by providing accurate and timely predictions.

Derive Inference

The "Derive Inference" phase, following model deployment, involves using the model to generate predictions or insights from new data in real time or batch mode. This phase is where the model delivers tangible value by applying its learned patterns to make decisions, classify data, or forecast outcomes. In practice, this means feeding the model live data inputs and obtaining actionable outputs that can drive business strategies, optimize operations, or enhance user experiences. The success of this phase hinges on the model's accuracy and relevance, ensuring that it consistently provides reliable and insightful inferences to inform decision-making.

Monitor Model

Deployment is more than just putting the model into operation; it's about maintaining its effectiveness over time. Continuous monitoring is essential to detect any performance degradation or drift in data patterns. This monitoring phase ensures that the model remains accurate and relevant, providing consistent value in its designated tasks.

In AWS, this can be accomplished using *Amazon SageMaker Model Monitor*, which automatically detects data and prediction quality issues. Amazon SageMaker Model Monitor continuously tracks key metrics like accuracy, latency, and data drift, and it generates alerts when deviations occur. Additionally, *Amazon CloudWatch* is used to collect and analyze logs, set alarms, and visualize performance metrics in real time. This proactive monitoring approach helps maintain the model's effectiveness, enabling timely interventions and retraining to address any performance degradation.

By carefully monitoring model deployments, organizations can leverage their ML models to drive impactful business decisions and enhance operational efficiency.

ML Concepts

During this study, we will use many terms related to ML. Therefore, it is essential that you understand this jargon. The key concepts you need to know for the exam are discussed in this section. These concepts will be covered in detail in the upcoming chapters, as your knowledge progresses.

Features

Features—also known as independent variables—are the inputs that the model learns from during training. The model uses the values of these features to make predictions.

Features are derived from observations or measurements recorded in the dataset. Each observation represents a single instance or data point, and the features are the attributes or variables that describe these instances. For example, if you're observing houses, each house (observation) might have features like the number of bedrooms, square footage, and year built.

Target Variable

The *target variable*—also known as the *dependent variable*—is the outcome the model is trying to predict.

In a supervised learning context, the target variable is known beforehand and is used to train the model. Put differently, the entire dataset you feed the model includes features data (independent variables) and target variables data (dependent variables). The model will use this dataset to learn during training.

Target variables are identified during the first step of the ML lifecycle, i.e., "Define ML Problem."

For example, if you're developing a model to predict house prices, the target variable would be the price of the house.

In ML, the terms *target variable, dependent variable*, and *label* are often used interchangeably. They all refer to the output variable that the model aims to predict based on the input features. Whether you're doing regression or classification, this is the variable you're trying to forecast or classify.

Optimization Problem

Optimization problems are central to ML. At its core, ML involves finding the best parameters for a model to minimize error and make accurate predictions. Whether it's through gradient descent, regularization techniques, or hyperparameter tuning, solving optimization problems is essential to training effective models.

Examples of optimization algorithms include gradient descent, stochastic gradient descent (SGD), and mini-batch gradient descent.

For example, consider training a neural network for image classification (neural networks will be introduced in the upcoming sections in this chapter). To achieve the highest accuracy, we use gradient descent to minimize the loss function, which measures the difference between the predicted and actual labels. The optimization algorithm iteratively adjusts the model's weights and biases to reduce this loss, improving the model's performance. Another example of an optimization problem is hyperparameter tuning in ML models. Hyperparameters, like learning rate, batch size, and number of layers in a neural network, significantly impact the model's performance. Finding the optimal combination of these hyperparameters involves searching through a vast space of possible values. Techniques such as grid search, random search, and Bayesian optimization automate this process,

systematically exploring different hyperparameter configurations and evaluating their performance on a validation set. These optimization methods help identify the best settings, leading to improved model performance and generalization.

Objective Function

An *objective function*, also known as a *loss function* or *cost function*, is a key concept in ML that defines the goal of the model training process. It's the function that the algorithm aims to minimize (or maximize) during training. By evaluating how well the model's predictions match the actual outcomes, the objective function measures the error or loss. Algorithms use this feedback to adjust the model's parameters iteratively, striving to reduce the error and improve accuracy. Common examples include mean squared error for regression tasks and cross-entropy loss for classification tasks. Understanding and selecting the right objective function is crucial for guiding the learning process effectively.

ML Algorithms vs. ML Models

Throughout this study, we will use the terms *ML models* and *ML algorithms*. These are two distinct concepts, and it is important to understand the differences between the two.

An ML algorithm is the step-by-step list of instructions provided in the form of code, which runs on a particular dataset to analyze data and identify patterns. This algorithm is analogous to general programming code. For example, it could be the geometrical average of a set of numbers. Similarly, in ML, an algorithm can be applied to learn the statistics of a given dataset or to apply current statistics to predict future data.

On the other hand, an ML model is the result of applying an algorithm to a dataset. It represents the learned patterns and relationships within the data in the form of a set of parameters. Once trained, the model can be used to make predictions or decisions based on new, unseen data.

Think of the ML algorithm as the recipe and the ML model as the finished dish created from the recipe.

Mathematically, the model can be thought of as a function that maps input features to output predictions. The learning process is all about discovering this function by finding the optimal parameters that minimize the difference between the predicted outputs and the actual target values. So, in essence, the ML algorithm helps identify and refine this mathematical function, transforming data into meaningful insights.

ML Algorithm Classification

A general way to classify ML algorithms is by learning type: *supervised learning*, *unsupervised learning*, and *reinforcement learning*. Let's deep dive into each type:

Supervised learning Supervised learning involves training a ML model using a labeled dataset, which means that each training example is paired with an output label: i.e., the correct prediction. The model learns to map inputs to the correct output by adjusting its parameters to minimize the error between its predictions and the actual labels. Common algorithms include linear regression, logistic regression, and support vector machines. Applications of supervised learning include email spam detection, image classification, and medical diagnosis.

Unsupervised learning Unsupervised learning deals with unlabeled data. The goal here is to uncover hidden patterns or intrinsic structures within the data. Without predefined labels, the model tries to learn the underlying distribution or clustering of the data. Common algorithms include k-means clustering, hierarchical clustering, and principal component analysis (PCA). Applications of unsupervised learning include market basket analysis, customer segmentation, and anomaly detection.

Reinforcement learning Reinforcement learning is inspired by behavioral psychology and involves training an agent to make decisions by rewarding desired behaviors and punishing undesired ones. The agent learns to maximize cumulative rewards over time by interacting with its environment and receiving feedback. Key components include the policy, reward signal, and value function. Applications of reinforcement learning include game playing (like AlphaGo), robotics, and autonomous driving.

Another way to classify ML algorithms is by predictive task. You need to understand the following two categories for the exam. More details will be provided in Chapter 4, where Amazon SageMaker's built-in algorithms will be covered in detail, with examples, code, and plots to show you how they work.

Classification Algorithms that predict a discrete category, like "apple" or "orange" (e.g., logistic regression, decision trees)

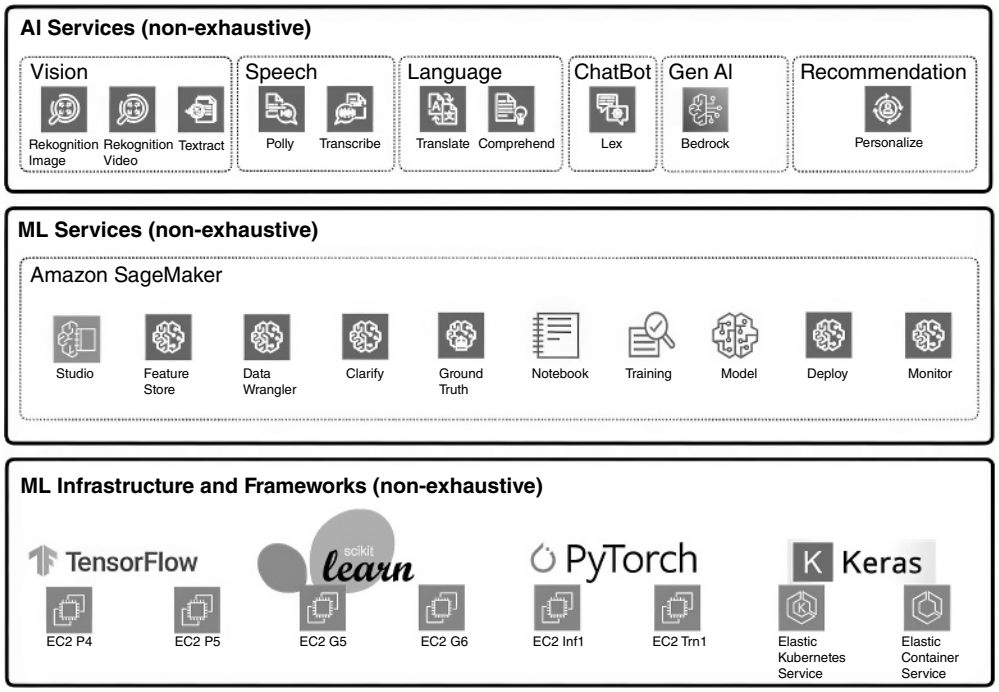
Regression Algorithms that predict a continuous value, like price or temperature (e.g., linear regression, support vector regression)

Differences Between ML and AI

AI and ML are often used interchangeably, but they represent distinct concepts within the field of computer science. AI is the broader concept, encompassing any technique that enables machines to simulate human intelligence, including reasoning, learning, problem-solving, and decision-making. AI can be rule-based systems, expert systems, or even simple automation tasks, aiming to replicate various aspects of human cognition.

ML, a subset of AI, specifically focuses on the ability of machines to learn from data and improve its performance over time without being explicitly programmed. ML involves training algorithms on datasets to identify patterns and make predictions. It includes techniques like supervised learning, unsupervised learning, and reinforcement learning, each serving different purposes in teaching machines to become smarter.

FIGURE 1.4 AWS ML stack.



For your workloads in AWS, you can choose from three different levels of ML and AI capabilities: ML Infrastructure and Frameworks, ML Services, and AI Services. This is called the *AWS ML stack*, as illustrated in Figure 1.4.

The AWS ML stack is a comprehensive suite of tools and services designed to support every stage of the ML lifecycle. At the base is the ML Infrastructure and Frameworks layer, which provides powerful computing resources like EC2 instances (P4, P5, G5, G6, Inf1, and Trn1) optimized for ML, as well as preconfigured environments like AWS Deep Learning AMIs that support popular ML frameworks such as TensorFlow, PyTorch, and Keras.

Moving up, the ML Services layer centers around Amazon SageMaker, a fully managed service that streamlines the processes of building, training, and deploying ML models. Amazon SageMaker is a family of services that offer features such as data labeling, model tuning, model evaluation, and deployment, significantly reducing the complexity and effort required for ML projects. You will learn the specific Amazon SageMaker services for each task in the upcoming chapters.

At the top, the AI Services layer provides *ready-to-use* AI capabilities that developers can easily integrate into their applications without ML or deep learning expertise. These AI services use prebuilt algorithms, use prebuilt models (already trained by AWS), and support a

wide range of use cases: Amazon Rekognition for image and video analysis, Amazon Polly for text-to-speech conversion, Amazon Textract for extracting text and data from documents, Amazon Translate for real-time language translation, Amazon Transcribe for converting speech to text, and Amazon Bedrock for building and scaling generative AI applications. These services enable businesses to quickly implement sophisticated AI functionalities, enhancing their applications with powerful and scalable intelligence.

Understanding Deep Learning

Deep learning (DL) is a specialized subset of ML that focuses on algorithms inspired by the structure and function of the brain, known as *neural networks*. Whereas ML encompasses a wide array of algorithms such as regression, classification, and clustering, DL distinguishes itself through its use of deep neural networks with many layers. These networks have revolutionized fields by automatically learning to represent data in ways that are often more accurate and effective than traditional ML methods.

Introduction to Neural Networks

Neural networks are the backbone of DL. They are composed of interconnected compute units called *neurons* that work together to process and learn from data. Each neuron receives input, processes it through an *activation function*, and passes the output to the next layer of neurons. This structure allows neural networks to learn complex patterns and representations in data, making them powerful tools for tasks like image and speech recognition.

Structure of a Neural Network

Neural networks are composed of interconnected neurons, which are grouped in layers. Let's dive deeper into each of these components.

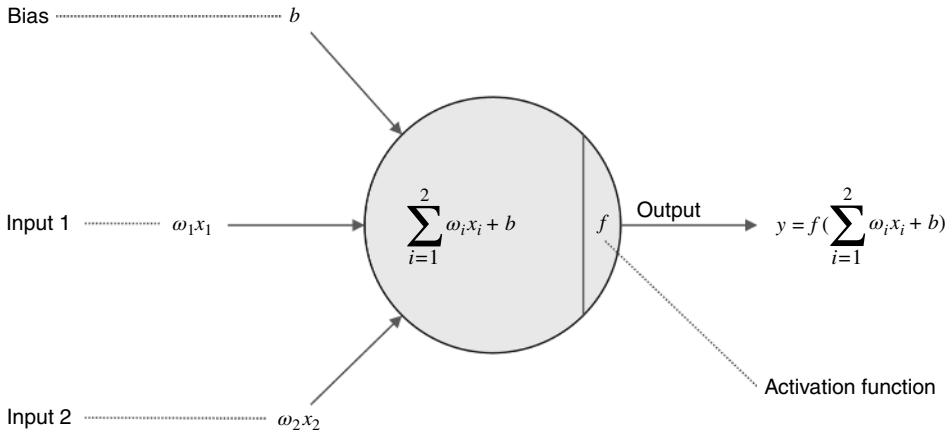
Neuron

A neuron, also known as a *node* or a *perceptron* in the context of artificial neural networks, is the fundamental building block of neural networks. It simulates the function of a biological neuron found in the human brain, albeit in a simplified and mathematical form.

The neuron's mathematical operations enable it to transform input features into meaningful predictions. By adjusting weights and biases through training, the neuron learns to make accurate predictions based on patterns and relationships in the data.

The components of a neuron are represented in Figure 1.5. Let's go over each.

FIGURE 1.5 Mathematical representation of a neuron.



Inputs These are the data features or signals that the neuron receives. Each input represents a specific observation for that feature. In Figure 1.5, the neuron receives two inputs, x_1 and x_2.

Weights Weights are values that determine the importance or contribution of each input to the neuron's output. During training, the network adjusts these weights to improve accuracy. In Figure 1.5, the input x_1 uses the weight ω_1, and the input x_2 uses the weight ω_2.

Bias The bias is an additional parameter that allows the activation function to shift. It helps the neuron make adjustments to the weighted sum before applying the activation function. In Figure 1.5, the bias is denoted with the symbol b.

Weighted sum With reference to Figure 1.5, the neuron calculates a weighted sum of the inputs and the bias: $\sum_{i=1}^{2} \omega_i x_i + b$.

Activation function The activation function f introduces nonlinearity into the model. It takes the weighted sum and produces the neuron's output. Common activation functions include sigmoid, Rectified Linear Unit (ReLU), and Hyperbolic Tangent (Tanh). In a neural network, the choice of activation function for each neuron is made by the data scientists or ML engineers designing the model and is informed by experimentation to determine the most effective configuration for the specific task at hand.

Output The output y is the final result produced by the neuron after applying the activation function to the weighted sum. The neuron sends its output to the next layer of neurons or to the final output layer.

Input Layer

The input layer is the first layer of the neural network. Its primary function is to receive the initial data that will be processed by the network. Each neuron in the input layer corresponds to a feature in the dataset.

For example, in a housing price prediction model, the features might be the size of the house, the number of bedrooms, the location, etc. Each of these features is an input to the input layer.

Hidden Layers

Hidden layers perform complex computations on the input data to detect patterns and relationships that aren't immediately obvious. The network can have one or more hidden layers. Each hidden layer consists of multiple neurons, and the outputs of the neurons in one layer become the inputs for the neurons in the subsequent layer. This process continues through all the hidden layers until it reaches the output layer. Neurons in hidden layers use activation functions (like ReLU, Sigmoid, and Tanh) to introduce nonlinearity. This allows the network to capture complex relationships in the data.

Continuing with our housing price prediction model, hidden layers might learn to identify patterns like the relationship between the size of the house and the number of bedrooms or the influence of the location on the price.

Output Layer

The output layer is the final layer of the neural network. Its primary function is to produce the final prediction or classification based on the processed data from the hidden layers. The output layer consists of neurons, each representing a possible outcome or class. The choice of activation function for each neuron in the output layer depends on the type of problem. Regression problems are characterized by continuous value predictions. For regression a linear activation function or no activation function might be used. Classification problems are characterized by discrete value predictions. For classification problems, a softmax or sigmoid activation function is commonly used to produce probabilities for each class.

In our housing price prediction model, the output layer might have a single neuron that produces the predicted price.

How Neural Networks Work

In a complex neural network, the generated model is a combination of weights and biases for each neuron. Here's a bit more detail on how this process works.

A neural network generates a model through the sophisticated interplay of weights and biases within its neurons, enhanced by the *backpropagation* algorithm. Backpropagation is the key algorithm in training a neuron. During the training phase, input data is fed into the network, where each neuron calculates a weighted sum of its inputs. This sum, combined

with a bias, is passed through an activation function to produce the neuron's output. The weights determine the influence each input (feature) has on the output (prediction), and the bias shifts the activation function to better fit the data. By adjusting these weights and biases through backpropagation, the network learns to map inputs to outputs accurately. Backpropagation calculates the error at the output layer using a *loss function* and then propagates this error backward through the network. By computing the gradient of the loss function with respect to the weights and biases, it uses optimization algorithms like gradient descent to update these parameters, thereby minimizing the error and improving predictive accuracy.

As the network trains, it progressively fine-tunes the weights and biases across all its neurons, effectively capturing complex patterns in the data. Each layer of neurons transforms the input data through a series of weighted sums, biases, and activation functions, with backpropagation ensuring these transformations lead to minimized error. The final layer's output is the model's prediction, whether it's a classification label or a continuous value. This combination of numerous neurons, each with its own set of weights and biases, facilitated by backpropagation, creates a powerful model capable of handling a wide range of tasks, from image recognition to NLP. The resulting model is a finely tuned machine, shaped by the continuous adjustment of weights and biases, enabling it to make accurate and reliable predictions.

Weights and biases are parameters that the model learns during training. They are adjusted through optimization processes like gradient descent and backpropagation to minimize the loss function and improve the model's predictions. Hyperparameters, on the other hand, are set *before* the training process begins and are not learned from the data. They control the training process and the structure of the model. Examples of hyperparameters include learning rate, number of epochs, number of hidden layers and units, batch size, and regularization parameters (to prevent overfitting by adding constraints to the model). More details will be provided in the "Hyperparameter Tuning" section in Chapter 5.

Neural Networks Types

Neural networks come in various types, each designed to tackle specific tasks and data structures. Artificial neural networks (ANNs), deep neural networks (DNNs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs) are some of the most prominent types you need to know for the exam. These different neural network architectures enable diverse applications across various domains, from image and video recognition to NLP and sequential data analysis.

Artificial Neural Networks

Artificial neural networks (ANNs) are the simplest form of neural networks, consisting of an input layer, one hidden layer, and an output layer. These layers are fully connected, meaning each neuron in one layer is connected to every neuron in the next layer. ANNs are capable of learning from data by adjusting the weights of these connections to minimize error. Despite their simplicity, ANNs can perform a wide range of tasks, from classification to regression.

Deep Neural Networks

Deep neural networks (DNNs) build upon the concept of ANNs by adding more hidden layers; hence the term *deep*. This depth allows them to model more complex patterns in the data, which is essential for solving sophisticated tasks. Each layer in a DNN learns to extract increasingly abstract features from the data, culminating in a powerful representation that can be used for tasks like object detection and natural language understanding. Figure 1.6 illustrates a DNN with three inputs, two hidden layers, and one output.

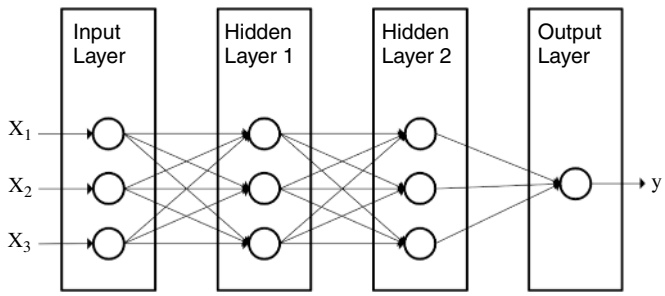
Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of DNN specifically designed for processing grid-like data, such as images. Because images are composed of pixels arranged in a grid format—where each pixel represents a specific color or intensity—CNNs utilize convolutional layers that apply filters to the input data, effectively capturing spatial hierarchies and patterns. This architecture makes CNNs exceptionally good at tasks like image classification, object detection, and even playing complex games.

Recurrent Neural Networks

Recurrent neural networks (RNNs) are another type of DNN tailored for sequential data, such as time series or natural language. Unlike traditional neural networks, RNNs have connections that form directed cycles, allowing them to maintain a memory of previous inputs. This makes them highly effective for tasks that involve sequences, such as language modeling, speech recognition, and machine translation. Variants like long short-term

FIGURE 1.6 Deep neural network with two hidden layers.



memory (LSTM) networks and gated recurrent units (GRUs) address some of the limitations of standard RNNs by better capturing long-term dependencies in the data.

Differences Between DL and ML

ML is a broad field encompassing various techniques that enable machines to learn from data and make predictions without being explicitly programmed. It includes algorithms like linear regression, decision trees, and clustering, often requiring manual feature engineering and preprocessing.

On the other hand, DL is a specialized subset of ML and uses neural networks with multiple layers (deep neural networks) to automatically learn representations from raw data and generate predictions from the data.

Whereas traditional ML models might rely on handcrafted features, DL models excel at processing large volumes of unstructured data, such as images and text, by extracting features autonomously. This depth and complexity allow DL models to achieve higher accuracy in tasks like image recognition and NLP, at the expense of significant amount of computational resources and longer training times.

As an AWS ML engineer, your job is to determine the trade-offs between the quality of your models and the costs related to training, deployment, and maintenance.

Case Studies

The following two case studies highlight how AWS's powerful compute solutions can enhance DL applications across different industries.

Case Study 1: Mobileye's Autonomous Driving Technology

Mobileye, a driving automation technology provider, develops self-driving technology and advanced driver-assistance systems using cameras, computer chips, and software.

Application Mobileye uses Amazon EC2 DL1 instances to train DL models for computer vision tasks like object detection, tracking, and segmentation. These models are critical for building autonomous driving solutions that adapt to changing road conditions.

Impact By leveraging AWS, Mobileye improved price performance by 40%, accelerated its development cycle, and managed more than 250 production workloads daily. This allowed Mobileye to deploy more efficient and cost-effective autonomous driving solutions.

For more information, visit:

`https://aws.amazon.com/solutions/case-studies/mobileye-ec2-dl1-case-study`

Case Study 2: Leidos' Healthcare ML Applications

Leidos, a science and technology solutions leader, builds ML applications to help public and private health organizations accelerate diagnosis and treatment.

Application Leidos migrated its ML workloads to Amazon EC2 DL1 instances, which are powered by Gaudi accelerators from Habana Labs. This migration improved performance and reduced compute costs for NLP and computer vision use cases.

Impact Leidos achieved 66% cost savings on model training, increased precision scores to 95–97%, and cut model training time from 8 hours to less than 1 hour for about 2,200 cases a day. This significantly improved patient outcomes and streamlined claim processing.

For more information, visit:

`https://aws.amazon.com/solutions/case-studies/leidos-case-study`

Summary

In this chapter, you learned how ML is a subset of AI that focuses on developing algorithms and statistical models that enable computers to learn and make predictions without explicit programming.

The distinctive element of ML when compared with traditional programming is the ability of ML models to learn from data and improve their performance over time without being explicitly programmed for specific tasks, allowing them to adapt and generalize to new use cases and unseen data more effectively.

The ML lifecycle was introduced with a description of each phase, from the definition of an ML problem to the deployment and monitoring of an ML model. You learned how features are the input of an ML algorithm, whereas the target variable is the output the algorithm is trying to predict. The concept of objective (loss) function was also introduced as a means to measure how well the model's predictions match the actual outcomes. Emphasis was provided on how ML algorithms differ from ML models, the former defining the specific procedures or rules used to train models by adjusting parameters based on input data, and the latter representing the outcome of this training process, encapsulating learned patterns and making predictions on new data.

You learned how AWS offers a wide spectrum of AI and ML services logically organized in the AWS ML stack. This representation includes at the lowest level compute infrastructure specifically designed and built for optimized ML applications, in the middle level the Amazon SageMaker ecosystem of services, and at the highest level a broad selection of fully managed AI services for a variety of different use cases.

At the end of the chapter, an introduction to deep learning—as a specialized branch of ML—was provided. DL uses neural networks to automatically extract high-level features from raw data, making them particularly effective for tasks such as image and speech recognition, and NLP. DL models, with their ability to learn intricate patterns and representations, have significantly advanced the capabilities of AI systems. Thus, whereas AI encompasses the broader goal of creating intelligent agents, ML and DL provide the tools and methods to achieve these objectives, with DL pushing the boundaries of what's possible in terms of complexity and accuracy.

Exam Essentials

Know the difference between data, information, and knowledge. Data is a collection of raw, unprocessed facts and figures without context or meaning. Information is data that has been processed, interpreted, and given context, making it useful for decision-making. Information is characterized by accuracy, relevance, completeness, and timeliness. Knowledge is information that has been assimilated, understood, and interpreted through experience, leading to insights, wisdom, or expertise.

Know the difference between structured, semi-structured, and unstructured data. Structured data is organized in accordance to a predefined schema and is easily searchable. Examples include Excel files and relational databases. Semi-structured data does not conform to a strict schema but still contains tags or markers to separate semantic elements, providing a degree of organization (e.g., JSON, XML documents). Unstructured data is often rich in information but requires advanced tools, such as natural language processing and ML, to extract meaningful insights because it lacks a predefined structure or schema.

Understand the machine learning lifecycle. ML is an iterative process, where models are continuously refined and improved based on new data and feedback, enabling the system to adapt and enhance its performance over time. The phases of the ML lifecycle are define ML problem, collect data, process data, choose algorithm, train model, evaluate model, deploy model, derive inference, and monitor model.

Know the difference between supervised, unsupervised, and reinforcement ML algorithms. With supervised ML algorithms, models are trained to predict outcomes using labeled data, where each data point is tagged with a corresponding label or outcome (i.e., the correct prediction). In contrast, unsupervised ML algorithms work with unlabeled data. The objective is to identify hidden patterns, structures, or relationships within the data. Reinforcement ML algorithms force agents to learn how to predict outcomes through trial and error.

Know the difference between an ML algorithm and an ML model. An ML algorithm is a set of rules or procedures used to create and train a model. It defines how the model will learn from the input data by adjusting its parameters. Examples include linear regression, logistic regression, decision trees, and k-Nearest Neighbors (k-NN). An ML model is the output generated *after* the training process using an ML algorithm. It represents the learned patterns and relationships derived from the training data. These patterns and relationships are codified in the form of a mathematical function, which essentially expresses in an unambiguous way the learned model. Once trained, the model can make predictions or classifications on new, unseen data.

Understand what a neural network is. A neural network is a computational model used in DL that simulates the human brain, consisting of interconnected neurons in layers. It processes input data, adjusts weights and biases through learning algorithms, and produces output to recognize patterns and make predictions.

Review Questions

1. Which of the following is *not* an example of semi-structured data?
   - **A.** NoSQL database
   - **B.** JSON document
   - **C.** XML document
   - **D.** PDF document
2. Which phase immediately precedes the model evaluation in the ML lifecycle?
   - **A.** Derive inference
   - **B.** Gradient descent
   - **C.** Deploy model
   - **D.** Train model
3. What are the inputs to a machine learning algorithm?
   - **A.** Target variables
   - **B.** Weights
   - **C.** Bias
   - **D.** Features
4. Which of the following is *not* a type of machine learning algorithm?
   - **A.** Supervised learning
   - **B.** Gradient descent
   - **C.** Unsupervised learning
   - **D.** Reinforcement learning
5. What distinguishes a machine learning model from a machine learning algorithm?
   - **A.** The model is a set of rules, and the algorithm is the learned patterns.
   - **B.** The model performs predictions, and the algorithm defines how it learns.
   - **C.** The model is the learning process, and the algorithm is the final output.
   - **D.** The model is a hyperparameter, and the algorithm is an optimization technique.
6. In the context of machine learning, what is a hyperparameter?
   - **A.** A parameter that the model learns from the data
   - **B.** A parameter that is set before the training process begins
   - **C.** A variable that holds the final output of the model
   - **D.** A metric used to evaluate the model's performance

7. What is the primary purpose of using activation functions in neural networks?
   - **A.** To initialize the weights
   - **B.** To introduce nonlinearity into the model
   - **C.** To reduce the dimensionality of the data
   - **D.** To perform gradient descent optimization

8. What is the primary function of backpropagation in a neural network?
   - **A.** To forward propagate the input data through the network
   - **B.** To update the weights and biases of each neuron to minimize error
   - **C.** To visualize the data
   - **D.** To remove outliers from the dataset

9. Which type of neural network is best suited for processing sequential data, such as time series or text?
   - **A.** Convolutional neural network (CNN)
   - **B.** Recurrent neural network (RNN)
   - **C.** Artificial neural network (ANN)
   - **D.** Deep neural network (DNN)

10. What is a key difference between deep learning and machine learning?
    - **A.** DL algorithms require less data than ML algorithms to perform well.
    - **B.** DL relies on neural networks with multiple layers to automatically extract features, whereas ML often requires manual feature engineering.
    - **C.** ML algorithms are always more accurate than DL algorithms.
    - **D.** DL algorithms cannot be used for image and speech recognition tasks.