**PYTHON – MACHINE LEARNING**

**Introduction**- Giving computers the ability to learn from Data

Machine learning evolved as a subfield of artificial intelligence that involved the development of self-learning algorithms to gain knowledge from that data to make predictions. Instead of requiring humans to manually derive rules and build models from analyzing large amounts of data, machine learning offers a more efficient alternative for capturing the knowledge in data to gradually improve the performance of predictive models and make data-driven decisions. Not only is machine learning becoming increasingly important in computer science research, but it also plays an ever-greater role in our everyday life.

**Three different types of machine Learning –**

Diagram

Description automatically generatedThe main goal in **supervised learning** is to learn a model from labeled training data that allows us to make predictions about unseen or future data. Here, the term supervised refers to a set of samples where the desired output signals (labels) are already known.

Diagram

Description automatically generatedThe example of **e-mail spam filtering**, we can train a model using a supervised machine learning algorithm on a corpus of labeled e-mail, e-mail that are correctly marked as spam or not-spam, to predict whether a new e-mail belongs to either of the two categories. A supervised learning task with discrete class labels, such as in the previous e-mail spam-filtering example, is also called a classification task.

* Another subcategory of supervised learning is regression, where the outcome signal is a continuous value:

**Classification for predicting class labels**

In machine learning, Classification, as the name suggests, classifies data into different parts/classes/groups. It is used to predict from which dataset the input data belongs to.

Classification is a subcategory of supervised learning where the goal is to predict the categorical class labels of new instances based on past observations. Those class labels are discrete, unordered values that can be understood as the group memberships of the instances. The previously mentioned example of e-mail-spam detection represents a typical example of a **binary classification task,** where the machine learning algorithm learns a set of rules to distinguish between two possible classes: spam and non-spam e-mail.

**Multi-class classification-** the set of class labels does not have to be of a binary nature. The predictive model learned by a supervised learning algorithm can assign any class label that was presented in the training dataset to a new, unlabeled instance. A typical **example** of a multi-class classification task is handwritten character recognition. Here, we could collect a training dataset that consists of multiple handwritten examples of each letter in the alphabet. Now, if a user provides a new handwritten character via an input device, our predictive model will be able to predict the correct letter in the alphabet with certain accuracy. However, our machine learning system would be unable to correctly recognize any of the digits zero to nine, for example, if they were not part of our training dataset.

Diagram

Description automatically generatedA white paper with writing on it

Description automatically generated with low confidence

Chart, scatter chart

Description automatically generatedGraphical user interface, text, email

Description automatically generated

The following figure illustrates the concept of a binary classification task given 30 training samples: 15 training samples are labeled as negative class (circles) and 15 training samples are labeled as positive class (plus signs). In this scenario, our dataset is two-dimensional, which means that each sample has two values associated with it: 1 x and 2 x. Now, we can use a supervised machine learning algorithm to learn a rule—the decision boundary represented as a black dashed line—that can separate those two classes and classify new data into each of those two categories given its 1 x and 2 x values:

Chart, scatter chart

Description automatically generated**Regression for predicting continuous outcomes:** A second type of supervised learning is the prediction of continuous outcomes, which is also called regression analysis. In regression analysis, we are given several predictor (explanatory) variables and a continuous response variable (outcome), and we try to find a relationship between those variables that allows us to predict an outcome. For example, let's assume that we are interested in predicting the Math SAT scores of our students. If there is a relationship between the time spent studying for the test and the final scores, we could use it as training data to learn a model that uses the study time to predict the test scores of future students who are planning to take this test.

The **following figure** illustrates the concept of linear regression. Given a predictor variable x and a response variable y, we fit a straight line to this data that minimizes the distance—most commonly the average squared distance—between the sample points and the fitted line. We can now use the intercept and slope learned from this data to predict the outcome variable of new data:

**Reinforcement Learning:**

Diagram

Description automatically generatedThe goal is to develop a system (agent) that improves its performance based on interactions with the environment. Since the information about the current state of the environment typically also includes a so-called reward signal, we can think of reinforcement learning as a field related to supervised learning. However, in reinforcement learning this feedback is not the correct ground truth label or value, but a measure of how well the action was measured by a reward function. Through the interaction with the environment, an agent can then use reinforcement learning to learn a series of actions that maximizes this reward via an exploratory trial-and-error approach or deliberative planning.

A **popular example** of reinforcement learning is a chess engine. Here, the agent decides upon a series of moves depending on the state of the board (the environment), and the reward can be defined as win or lose at the end of the game

**Difference between supervised and reinforcement Learning**

In **supervised** learning, we know the right answer beforehand when we train our model, and in **reinforcement learning**, we define a measure of reward for actions by the agent. In **unsupervised learning**, however, we are dealing with unlabeled data or data of unknown structure. Using unsupervised learning techniques, we can explore the structure of our data to extract meaningful information without the guidance of a known outcome variable or reward function.

**Unsupervised Learning**

**Finding subgroups with clustering**

Shape, circle

Description automatically generatedClustering is an exploratory data analysis technique that allows us to organize a pile of information into meaningful subgroups (clusters) without having any prior knowledge of their group memberships. Each cluster that may arise during the analysis defines a group of objects that share a certain degree of similarity but are more dissimilar to objects in other clusters, which is why clustering is also sometimes called "unsupervised classification." Clustering is a great technique for structuring information and deriving meaningful relationships among data**, for example**, it allows marketers to discover customer groups based on their interests to develop distinct marketing programs.

The figure below illustrates how clustering can be applied to organizing unlabeled data into three distinct groups based on the similarity of their features 1 x and 2 x:

**Dimensionality Reduction for data compression**

Diagram

Description automatically generated with medium confidenceAnother subfield of unsupervised learning is dimensionality reduction. Often, we are working with data of high dimensionality—each observation comes with a high number of measurements—that can present a challenge for limited storage space and the computational performance of machine learning algorithms. Unsupervised dimensionality reduction is a commonly used approach in feature preprocessing to remove noise from data, which can also degrade the predictive performance of certain algorithms and compress the data onto a smaller dimensional subspace while retaining most of the relevant information.

Sometimes, dimensionality reduction can also be useful for visualizing data—**for example**, a high-dimensional feature set can be projected onto one-, two-, or three-dimensional feature spaces to visualize it via 3D- or 2D-scatterplots or histograms. The figure below shows an example where non-linear dimensionality reduction was applied to compress a 3D Swiss Roll onto a new 2D feature subspace

**Basic Terminologies and notations:**

**Iris Dataset**

A picture containing schematic

Description automatically generatedDiagram

Description automatically generated

We can represent the iris dataset in terms of matrix and vectors to make notation and implementation efficient. (Linear algebra)

Graphical user interface, text, application

Description automatically generated

**Roadmap for building machine learning systems**

The diagram below shows a typical workflow diagram for using machine learning in predictive modeling, which we will discuss in the following subsections:

Diagram

Description automatically generated

**Preprocessing – getting data into shape**

Raw data rarely comes in the form and shape that is necessary for the optimal performance of a learning algorithm. Thus, the preprocessing of the data is one of the most crucial steps in any machine learning application. If we take the Iris flower dataset from the previous section as an example, we could think of the raw data as a series of flower images from which we want to extract meaningful features. Useful features could be the color, the hue, the intensity of the flowers, the height, and the flower lengths and widths. Many machine learning algorithms also require that the selected features are on the same scale for optimal performance, which is often achieved by transforming the features in the range [0, 1] or a standard normal distribution with zero mean and unit variance.

Some of the selected features may be highly correlated and therefore redundant to a certain degree. In those cases, dimensionality reduction techniques are useful for compressing the features onto a lower dimensional subspace. Reducing the dimensionality of our feature space has the advantage that less storage space is required, and the learning algorithm can run much faster.

To determine whether our machine learning algorithm not only performs well on the training set but also generalizes well to new data, we also want to randomly divide the dataset into a separate training and test set. We use the training set to train and optimize our machine learning model, while we keep the test set until the very end to evaluate the final model.

**Training and selecting a predicative model**

For example, each classification algorithm has its inherent biases, and no single classification model enjoys superiority if we don't make any assumptions about the task. In practice, it is therefore essential to compare at least a handful of different algorithms to train and select the best performing model. But before we can compare different models, we first must decide upon a metric to measure performance. One commonly used metric is classification accuracy, which is defined as the proportion of correctly classified instances.

**How do we know which model performs well on the final test dataset and real-world data if we don't use this test set for the model selection but keep it for the final model evaluation?**

To address the issue embedded in this question, different cross-validation techniques can be used where the training dataset is further divided into training and validation subsets to estimate the generalization performance of the model. Finally, we also cannot expect that the default parameters of the different learning algorithms provided by software libraries are optimal for our specific problem task. Therefore, we will make frequent use of **hyperparameter optimization techniques** that help us to fine-tune the performance of our model in later chapters. Intuitively, we can think of those hyperparameters as parameters that **are not learned from the data but represent the knobs of a model** that we can turn to improve its performance, which will become much clearer in later chapters when we see actual examples.

**Evaluating models and predicting unseen data instances:**

After we have selected a model that has been fitted on the training dataset, we can use the test dataset to estimate how well it performs on this unseen data to estimate the generalization error. If we are satisfied with its performance, we can now use this model to predict new, future data. It is important to note that the parameters for the previously mentioned procedures—such as feature scaling and dimensionality reduction—are solely obtained from the training dataset, and the same parameters are later re-applied to transform the test dataset, as well as any new data samples—the performance measured on the test data may be overoptimistic otherwise.