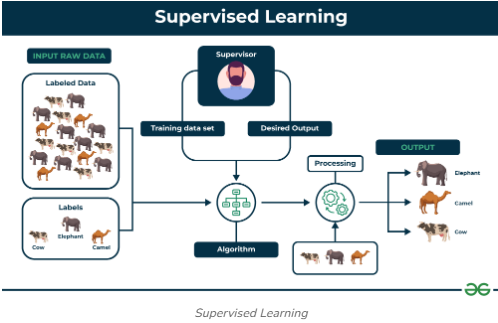
YouTube URLs:

1. 100 Days of Machine Learning by CampusX: <https://www.youtube.com/playlist?list=PLKnIA16_Rmvbr7zKYQuBfsVkjoLcJgxHH>
2. Machine Learning Tutorial using Python by Codebasics: <https://www.youtube.com/playlist?list=PLeo1K3hjS3uvCeTYTeyfe0-rN5r8zn9rw>
3. Document URL: https://www.geeksforgeeks.org/types-of-machine-learning/

|  |  |  |
| --- | --- | --- |
| Types of ML | | |
| **Supervised** | Definition | Example |
|  | [Supervised learning](https://www.geeksforgeeks.org/supervised-machine-learning/) is defined as when a model gets trained on a “**Labelled Dataset”**. Labelled datasets have both input and output parameters. In **Supervised Learning** algorithms learn to map points between inputs and correct outputs. It has both training and validation datasets labelled. | **Example:**Consider a scenario where you have to build an image classifier to differentiate between cats and dogs. If you feed the datasets of dogs and cats labelled images to the algorithm, the machine will learn to classify between a dog or a cat from these labelled images. When we input new dog or cat images that it has never seen before, it will use the learned algorithms and predict whether it is a dog or a cat. This is how **supervised learning** works, and this is particularly an image classification.  **Categories:**   * [Classification](https://www.geeksforgeeks.org/getting-started-with-classification/) where output y is categorical data * [Regression](https://www.geeksforgeeks.org/types-of-regression-techniques/) where output y is numeric data |
| **Unsupervised** | [Unsupervised Learning](https://www.geeksforgeeks.org/unsupervised-machine-learning-the-future-of-cybersecurity/) Unsupervised learning is a type of machine learning technique in which an algorithm discovers patterns and relationships using unlabelled data. Unlike supervised learning, unsupervised learning doesn’t involve providing the algorithm with labelled target outputs. The primary goal of Unsupervised learning is often to discover hidden patterns, similarities, or clusters within the data, which can then be used for various purposes, such as data exploration, visualization, dimensionality reduction, and more. | **Example:**Consider that you have a dataset that contains information about the purchases you made from the shop. Through clustering, the algorithm can group the same purchasing behaviour among you and other customers, which reveals potential customers without predefined labels. This type of information can help businesses get target customers as well as identify outliers.  **Categories:**   * [Clustering](https://www.geeksforgeeks.org/clustering-in-machine-learning/) * Dimensionality Reduction * Anomaly Detection * [Association](https://www.geeksforgeeks.org/association-rule/) |
| **Semi Supervised** | [Semi-Supervised learning](https://www.geeksforgeeks.org/ml-semi-supervised-learning/)is a machine learning algorithm that works between the [supervised and unsupervised](https://www.geeksforgeeks.org/supervised-unsupervised-learning/) learning so it uses both **labelled and unlabelled** data. It’s particularly useful when obtaining labelled data is costly, time-consuming, or resource intensive. This approach is useful when the dataset is expensive and time-consuming. Semi-supervised learning is chosen when labelled data requires skills and relevant resources in order to train or learn from it.  We use these techniques when we are dealing with data that is a little bit labelled and the rest large portion of it is unlabelled. We can use the unsupervised techniques to predict labels and then feed these labels to supervised techniques. This technique is mostly applicable in the case of image data sets where usually all images are not labelled. | **Example**: Consider that we are building a language translation model, having labelled translations for every sentence pair can be resources intensive. It allows the models to learn from labelled and unlabelled sentence pairs, making them more accurate. This technique has led to significant improvements in the quality of machine translation services. |
| **Reinforcement** | [Reinforcement machine learning](https://www.geeksforgeeks.org/what-is-reinforcement-learning/)algorithm is a learning method that interacts with the environment by producing actions and discovering errors. **Trial, error, and delay** are the most relevant characteristics of reinforcement learning. In this technique, the model keeps on increasing its performance using Reward Feedback to learn the behaviour or pattern. These algorithms are specific to a particular problem e.g. Google Self Driving car, AlphaGo where a bot competes with humans and even itself to get better and better performers in Go Game. Each time we feed in data, they learn and add the data to their knowledge which is training data. So, the more it learns the better it gets trained and hence experienced. | **Example:**Consider that you are training an [AI](https://www.geeksforgeeks.org/artificial-intelligence-an-introduction/) agent to play a game like chess. The agent explores different moves and receives positive or negative feedback based on the outcome. Reinforcement Learning also finds applications in which they learn to perform tasks by interacting with their surroundings.  **Types of Reinforcement Machine Learning**  There are two main types of reinforcement learning:   * **Positive reinforcement** * **Negative reinforcement** |

1. **Supervised**

****

1. [**Classification**](https://www.geeksforgeeks.org/getting-started-with-classification/):

[**Classification**](https://www.geeksforgeeks.org/getting-started-with-classification/)deals with predicting **categorical** target variables, which represent discrete classes or labels. For instance, classifying emails as spam or not spam, or predicting whether a patient has a high risk of heart disease. Classification algorithms learn to map the input features to one of the predefined classes.

**Algorithms**:

* [**Logistic Regression**](https://www.geeksforgeeks.org/understanding-logistic-regression/)
* [**Support Vector Machine**](https://www.geeksforgeeks.org/support-vector-machine-algorithm/)
* [**Random Forest**](https://www.geeksforgeeks.org/random-forest-regression-in-python/)
* [**Decision Tree**](https://www.geeksforgeeks.org/decision-tree/)
* [**K-Nearest Neighbors (KNN)**](https://www.geeksforgeeks.org/k-nearest-neighbours/)
* [**Naive Bayes**](https://www.geeksforgeeks.org/naive-bayes-classifiers/)

1. **Regression**:

[**Regression**](https://www.geeksforgeeks.org/regression-classification-supervised-machine-learning/), on the other hand, deals with predicting **continuous** target variables, which represent numerical values. For example, predicting the price of a house based on its size, location, and amenities, or forecasting the sales of a product. Regression algorithms learn to map the input features to a continuous numerical value.

**Algorithms**:

* [**Linear Regression**](https://www.geeksforgeeks.org/ml-linear-regression/)
* [**Polynomial Regression**](https://www.geeksforgeeks.org/videos/polynomial-regression-algorithm-machine-learning/)
* [**Ridge Regression**](https://www.geeksforgeeks.org/videos/lasso-ridge-regression-algorithm-machine-learning/)
* [**Lasso Regression**](https://www.geeksforgeeks.org/videos/lasso-ridge-regression-algorithm-machine-learning/)
* [**Decision tree**](https://www.geeksforgeeks.org/decision-tree-introduction-example/)
* [**Random Forest**](https://www.geeksforgeeks.org/random-forest-regression-in-python/)

**Advantages of Supervised Machine Learning**

* **Supervised Learning** models can have high accuracy as they are trained on **labelled data**.
* The process of decision-making in supervised learning models is often interpretable.
* It can often be used in pre-trained models which saves time and resources when developing new models from scratch.

**Disadvantages of Supervised Machine Learning**

* It has limitations in knowing patterns and may struggle with unseen or unexpected patterns that are not present in the training data.
* It can be time-consuming and costly as it relies on**labelled**data only.
* It may lead to poor generalizations based on new data.

**Applications of Supervised Learning**

Supervised learning is used in a wide variety of applications, including:

* **Image classification**: Identify objects, faces, and other features in images.
* **Natural language processing:** Extract information from text, such as sentiment, entities, and relationships.
* **Speech recognition**: Convert spoken language into text.
* **Recommendation systems**: Make personalized recommendations to users.
* **Predictive analytics**: Predict outcomes, such as sales, customer churn, and stock prices.
* **Medical diagnosis**: Detect diseases and other medical conditions.
* **Fraud detection**: Identify fraudulent transactions.
* **Autonomous vehicles**: Recognize and respond to objects in the environment.
* **Email spam detection**: Classify emails as spam or not spam.
* **Quality control in manufacturing**: Inspect products for defects.
* **Credit scoring**: Assess the risk of a borrower defaulting on a loan.
* **Gaming**: Recognize characters, analyse player behaviour, and create NPCs.
* **Customer support**: Automate customer support tasks.
* **Weather forecasting**: Make predictions for temperature, precipitation, and other meteorological parameters.
* **Sports analytics**: Analyse player performance, make game predictions, and optimize strategies.

1. **Unsupervised**
2. **Clustering**

[Clustering](https://www.geeksforgeeks.org/clustering-in-machine-learning/) is the process of grouping data points into clusters based on their similarity. This technique is useful for identifying patterns and relationships in data without the need for labelled examples.

**Algorithms:**

* [**K-Means Clustering algorithm**](https://www.geeksforgeeks.org/k-means-clustering-introduction/)
* [**Mean-shift algorithm**](https://www.geeksforgeeks.org/ml-mean-shift-clustering/)
* [**DBSCAN Algorithm**](https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/)
* [**Principal Component Analysis**](https://www.geeksforgeeks.org/principal-component-analysis-pca/)
* [**Independent Component Analysis**](https://www.geeksforgeeks.org/ml-independent-component-analysis/)

1. **Dimensionality Reduction**

**URL:** <https://www.analyticsvidhya.com/blog/2018/08/dimensionality-reduction-techniques-python/>

<https://www.analyticsvidhya.com/blog/2015/07/dimension-reduction-methods/>

Dimensionality reduction is a technique used in machine [learning](https://www.analyticsvidhya.com/blog/2018/08/dimensionality-reduction-techniques-python/) and data analysis to reduce the number of features or variables under consideration. The aim is to simplify the dataset while retaining as much relevant information as possible. This is particularly useful when dealing with high-dimensional data, where the number of features is large compared to the number of samples.

There are various methods for dimensionality reduction, including:

1. **Feature selection**: Selecting a subset of the original features based on specific criteria such as relevance, importance, or correlation.
2. **Feature extraction**: Transforming the original features into a lower-dimensional space using techniques like principal component analysis (PCA), linear discriminant analysis (LDA), or t-distributed stochastic neighbour embedding (t-SNE). These methods aim to preserve the most important information while reducing the dimensionality.

**Why are Dimensionality Reduction Techniques Required?**

Here are some of the benefits of applying dimensionality reduction to a dataset:

* The space required to store the data is reduced as the number of dimensions comes down
* Fewer dimensions lead to less computation/training time
* Some algorithms do not perform well when we have large dimensions. So, reducing these dimensions needs to happen for the algorithm to be useful
* It takes care of multicollinearity by removing redundant features. For example, you have two variables – ‘time spent on treadmill in minutes’ and ‘calories burnt’. These variables are highly correlated, as the more time you spend running on a treadmill, the more calories you burn. Hence, there is no point in storing both, as just one of them does what you require
* It helps in visualizing data. As discussed earlier, it is very difficult to visualize data in higher dimensions, so reducing our space to 2D or 3D may allow us to plot and observe patterns more clearly

**Common Dimensionality Reduction Techniques**

Dimensionality reduction can be done in two different ways:

* By only keeping the most relevant variables from the original dataset (this technique is called feature selection)
* By finding a smaller set of new variables, each being a combination of the input variables, containing the same information as the input variables (this technique is called dimensionality reduction)

1. **Missing Value Ratio**

* Suppose you’re given a [dataset.](https://www.analyticsvidhya.com/blog/category/maths/page/14/) What would be your first step? You would naturally want to explore the data first before building a model. While exploring the data, your dataset has some missing values. Now what? You will try to find out why these are missing values and then impute them or drop the variables entirely that have missing values (using appropriate methods).
* What if we have too many missing values (more than 50%)? Should we impute the missing values or drop the variable? I would prefer to drop the variable since it will not have much information. However, this isn’t set in stone. We can set a threshold value; if the percentage of missing values in any variable exceeds that threshold, we will drop the variable.

**Let’s implement this approach in Python.**  
**Python Code:**

<span data-mce-type="bookmark" style="display: inline-block; width: 0px; overflow: hidden; line-height: 0;" class="mce\_SELRES\_start">﻿</span>

**First, let’s load the data:**

**Python Code:**

# import required libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# read the data

train=pd.read\_csv("Train\_UWu5bXk.csv")

*Note: The path of the file should be added while reading the data.*

Now, we will check the percentage of missing values in each variable. We can use. **isnull().sum()** to calculate this.

# checking the percentage of missing values in each variable

train.isnull().sum()/len(train)\*100

A screenshot of a computer screen

Description automatically generated

As you can see in the above table, there aren’t too many missing values (just 2 variables have them, actually). We can impute the values using appropriate methods or set a threshold of, say, 20% and remove the variable having more than 20% missing values. Let’s look at how this can be done in Python:

# saving missing values in a variable

a = train.isnull().sum()/len(train)\*100

# saving column names in a variable

variables = train.columns

variable = [ ]

for i in range(0,12):

    if a[i]<=20:   #setting the threshold as 20%

    variable.append(variables[i])

So, the variables to be used are stored in “variable,” which contains only those features with missing values of less than 20%

1. **Low Variance Filter**

Consider a variable in our dataset where all the observations have the same value, say 1. If we use this variable, can it improve the model we will build? The answer is no because this variable will have zero variance.

So, we need to calculate the variance of each variable we are given. Then, we drop the variables with low variance compared to other variables in our dataset. As I mentioned above, this is because variables with a low variance will not affect the target variable.

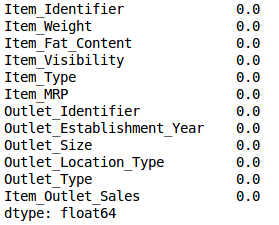
**Let’s first impute the missing values in the *Item\_Weight* column using the median value of the known *Item\_Weight*observations. For the *Outlet\_Size* column, we will use the mode of the known *Outlet\_Size* values to impute the missing values:**

train['Item\_Weight'].fillna(train['Item\_Weight'].median(), inplace=True)

train['Outlet\_Size'].fillna(train['Outlet\_Size'].mode()[0], inplace=True)

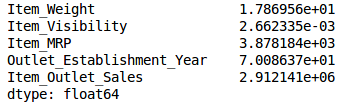
Let’s check whether all the missing values have been filled:

train.isnull().sum()/len(train)\*100



Voila! We are all set. Now let’s calculate the variance of all the numerical variables.

train.var()



As the above output shows, the variance of *Item\_Visibility* is very low compared to the other variables. We can safely drop this column. This is how we apply a low-variance filter. Let’s implement this in Python:

numeric = train[['Item\_Weight', 'Item\_Visibility', 'Item\_MRP', 'Outlet\_Establishment\_Year']]

var = numeric.var()

numeric = numeric.columns

variable = [ ]

for i in range(0,len(var)):

    if var[i]>=10:   #setting the threshold as 10%

       variable.append(numeric[i+1])

The above code gives us a list of variables with a variance greater than 10.