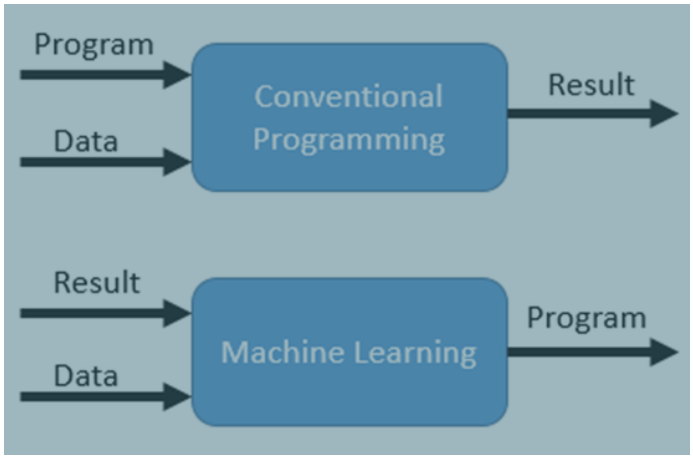
01

Machine Learning and Scikit Learn Introduction

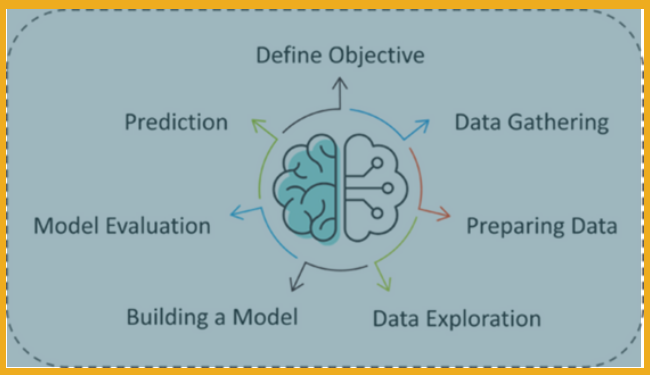
Machine Learning

Machine learning (ML) is a category of an algorithm that allows software applications to become more accurate in predicting outcomes without being explicitly programmed.



Machine Learning Process

The craft of creating machine learning (ML) processes is comprised of a number of steps:



Machine Learning Process

1. Decide on the Question

   Most ML processes start by asking a question that cannot be answered by a simple conditional program or rules-based engine. These questions often revolve around predictions based on a collection of data. At this step we must understand what exactly needs to be predicted.

2. Collecting Data

   To be able to answer your question, you need data. The quality and, sometimes, quantity of your data will determine how well you can answer your initial question. Data collection can be done manually or by web scraping. However, if you're a beginner and you're just looking to learn Machine Learning you don't have to worry about getting the data. There are lots of data resources on the web, you can just download the data set and get going.

3. Data Preparation

   The data you collected is almost never in the right format. You will encounter a lot of inconsistencies in the data set such as missing values, redundant variables, duplicate values, etc. Removing such inconsistencies is very essential because they might lead to wrongful computations and predictions. Therefore, at this stage, you scan the data set for any inconsistencies and you fix them then and there

4. Exploratory Data Analysis

   This is the brainstorming stage of Machine Learning. Data Exploration involves understanding the patterns and trends in the data. At this stage, all the useful insights are drawn and correlations between the variables are understood. Visualizing data is an important aspect of this phase.

5. Building ML model

   This stage always begins by splitting the data set into two parts, training and testing data. Choosing the right algorithm depends on the type of problem you're trying to solve, the data set and the complexity of the problem. The model is built from training process where we pass data and the machine will find patterns to make predictions. Over time, with training, the model gets better at predicting.

6. Evaluate and optimize the modell

   Evaluate and optimize the model. After training, we have to check the model to see how it's performing. This is done by testing the performance of the model on previously unseen data (testing data)

7. Parameter tuning

   Based on the performance of your model, you can redo the process using different parameters, or variables, that control the behavior of the algorithms used to train the model.

8. Paredict

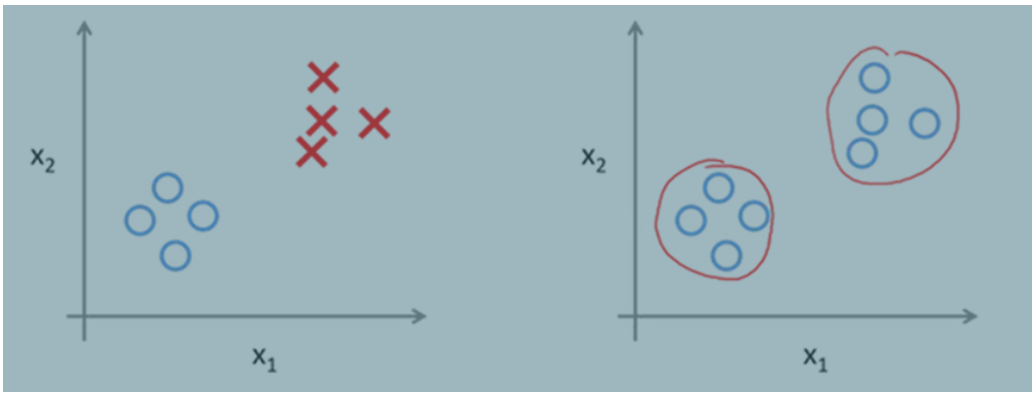
   Use new inputs to test the accuracy of your model.

Machine Learning Types

A machine can learn to solve a problem by following any one of the following three approaches. These are the ways in which a machine can learn:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Supervised vs Unsupervised



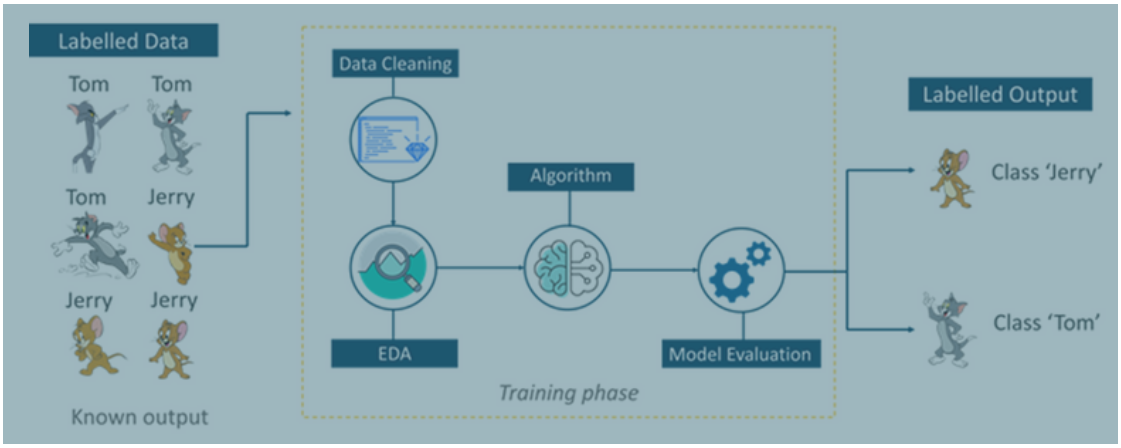
Supervised	Unsupervised
In supervised learning algorithms, the output for the given input is known.	In unsupervised learning algorithms, the output for the given input is unknown.
The algorithms learn from labeled set of data. This data helps in evaluating the accuracy on training data.	The algorithm is provided with unlabeled data where it tries to find patterns and associations in between the data items.
It is a Predictive Modeling technique which predicts the future outcomes accurately.	It is a Descriptive Modeling technique which explains the real relationship between the elements and history of the elements.
It includes classification and regression algorithms.	It includes clustering and association rules learning algorithms.
Some algorithms of supervised learning are Linear Regression, Naïve Bayes, and Neural Networks.	Some algorithms for unsupervised learning are k- means clustering, Apriori, etc.
This type of learning is relatively complex as it requires labelled data.	It is less complex as there is no need to understand and label data.
It is more accurate than unsupervised learning as input data and corresponding output is well known, and the machine only needs to give predictions.	It has less accuracy as the input data is unlabeled. Thus the machine has to first understand and label the data and then give predictions.
It is an online process of data analysis and does	This is a real time analysis of data.

Supervised Learning

Supervised learning is a technique in which we teach or train the machine using data which is well labeled. This approach is similar to human learning under the supervision of a teacher. The teacher provides good examples for the student to memorize, and the student then derives general rules from these specific examples.

2 types of supervised learning problems:

- Regression problems: the target is a numeric value. Example: determines the average prices of houses in the certain area
- Classification problems: the target is a qualitative variable, such as a class or a tag. Example: distinguishes between kinds of iris flowers based on their sepal and petal measures.

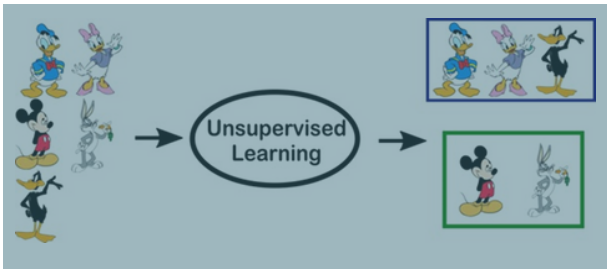


Unsupervised Learning

Unsupervised learning involves training by using unlabeled data and allowing the model to act on that information without guidance.

Some recommendation systems that you find on the web in the form of marketing automation are based on this type of learning.

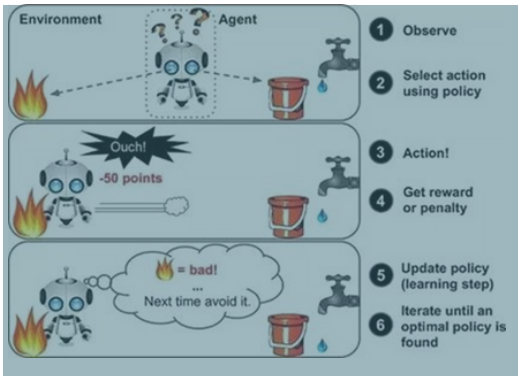
The marketing automation algorithm derives its suggestions from what you've bought in the past. The recommendations are based on an estimation of what group of customers you resemble the most and then inferring your likely preferences based on that group.



Reinforce-ment Learning

Reinforcement Learning is a part of Machine learning where an agent is put in an environment and he learns to behave in this environment by performing certain actions and observing the rewards which it gets from those actions.

Example: when computers learn to play video games by themselves.



Various Classical Machine Learning Algorithms

- Linear Regression
- Logistic Regression
- K-nearest neighbors
- Support Vector machine
- K means clustering
- Decision Trees
- Random Forest
- Naive Bayes

Libraries used in Machine Learning

Libraries save developers from writing redundant code over and over. Also, there are all sorts of libraries to deal with different things. Example: Numpy, Pandas Matplotlib and Seaborn already covered

Scikit Learn Introduction

Scikit-learn (Sklearn) is an open-source library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistent interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Scikit Learn Features

Some of the most popular groups of models provided are:

- Supervised Learning algorithms - Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are part of scikit-learn.
- Unsupervised Learning algorithms - It also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.
- Cross Validation - It is used to check the accuracy of supervised models on unseen data.
- Dimensionality Reduction - It is used for reducing the number of attributes in data which can be further used for summarization, visualization and feature selection.

- Ensemble methods - As the name suggests, it is used for combining the predictions of multiple supervised models.
- Feature extraction - It is used to extract the features from data to define the attributes in image and text data.
- Feature selection - It is used to identify useful attributes to create supervised models.

A collection of data is called a dataset. It has following components:

- Features – The variables of data are called its features. They are also known as predictors, inputs or attributes.
- Feature matrix – It is the collection of features, in case there are more than one.
- Feature Names – It is the list of all the names of the features.
- Response – It is the output variable that basically depends upon the feature variables. They are also known as target, label or output.
- Response Vector – It is used to represent the response column.
- Target Names – It represents the possible values taken by a response vector.

Scikit Learn Data Preprocessing – Binarization

This preprocessing technique is used when we need to convert our numerical values into Boolean values.

```
import numpy as np
from sklearn import preprocessing
input_data = np.array(
   [[2.1, -1.9, 5.5],
   [-1.5, 2.4, 3.5],
   [0.5, -7.9, 5.6],
   [5.9, 2.3, -5.8]]
)
data_binarized = preprocessing.Binarizer(threshold=0.5).transform(input_data)
print("Binarized data:\n", data_binarized)
```

```
inarized data:
[[1. 0. 1.]
[0. 1. 1.]
[0. 0. 1.]
[1. 1. 0.]]
```

Scikit Learn Dataset Loading

Scikit-learn have few built-in datasets examples like iris and digits for classification and the diabetes for regression.

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target
feature_names = iris.feature_names
target_names = iris.target_names
print("Feature names:", feature_names)
print("Target names:", target_names)
print("First 10 rows of X:\n", X[:10])
```

```
Feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Target names: ['setosa' 'versicolor' 'virginica']
First 10 rows of X:
 [[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]]
```

Scikit Learn Data Preprocessing- Feature Scaling

Feature Scaling is the process of changing the scale of certain features to a common one.

- Normalization is the process of scaling data into a range of [xmin, xmax]. It's more useful and common for regression tasks.

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- Standardization is the process of scaling data so that they have a mean value of 0 and a standard deviation of 1. It's more useful and common for classification tasks.

$$x' = \frac{x - \mu}{\sigma}$$

Normalization

- MinMaxScaler rescales the data points into range of 0 to 1.
- MaxAbsScaler is similar to minmax scaler but it scales values within the range of [-1,1]

```
scaler1 = preprocessing.MinMaxScaler()
scaled_data1 = scaler1.fit_transform(input_data)
print ("Min Max scaled data:\n", scaled_data1,'\n')

scaler2 = preprocessing.MaxAbsScaler()
scaled_data2 = scaler2.fit_transform(input_data)
print ("Max Abs scaled data:\n", scaled_data2,'\n')
```

```
Min Max scaled data:
 [[0.48648649 0.58252427 0.99122807]
 [0.         1.         0.81578947]
 [0.27027027 0.         1.        ]
 [1.         0.99029126 0.        ]]

Max Abs scaled data:
 [[ 0.3559322  -0.24050633  0.94827586]
 [-0.25423729  0.30379747  0.60344828]
 [ 0.08474576 -1.          0.96551724]
 [ 1.          0.29113924 -1.        ]]
```

or


```
scaler1 = preprocessing.MinMaxScaler()
scaled_data1 = scaler1.fit_transform(input_data)
print ("Min Max scaled data:\n", scaled_data1,'\n')

scaler2 = preprocessing.MaxAbsScaler()
scaled_data2 = scaler2.fit_transform(input_data)
print ("Max Abs scaled data:\n", scaled_data2,'\n')
```

```
Min Max scaled data:
 [[0.48648649 0.58252427 0.99122807]
 [0.         1.         0.81578947]
 [0.27027027 0.         1.        ]
 [1.         0.99029126 0.        ]]

Max Abs scaled data:
 [[ 0.3559322  -0.24050633  0.94827586]
 [-0.25423729  0.30379747  0.60344828]
 [ 0.08474576 -1.          0.96551724]
 [ 1.          0.29113924 -1.        ]]
```


Standardization

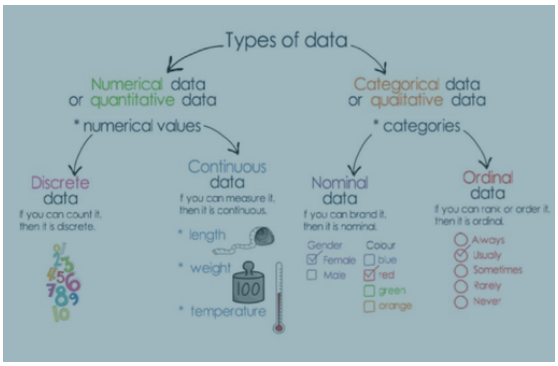
Standardization (Z-score normalization) is the process where the features are rescaled so that they'll have the properties of a standard normal distribution with mean 0 and standard deviation 1.

```
Raw data:
 [[ 2.1 -1.9  5.5]
 [-1.5  2.4  3.5]
 [ 0.5 -7.9  5.6]
 [ 5.9  2.3 -5.8]]

std_scaler = preprocessing.StandardScaler()
normalized_data = std_scaler.fit_transform(input_data)
print ("Normalized data:\n", normalized_data)

Normalized data:
 [[ 0.12894603 -0.14880162  0.70300338]
 [-1.19735598  0.8749535   0.27694073]
 [-0.46052153 -1.57729713  0.72430651]
 [ 1.52893149  0.85114524 -1.70425062]]
```

Some tips for selecting the appropriate method to apply.

- Normalization is good to use when you know that the distribution of your data does not follow a Gaussian distribution.
- Standardization, on the other hand, can be helpful in cases where the data follows a Gaussian distribution. However, this does not have to be necessarily true.

Scikit Learn Data Preprocessing – Encoding



Label Encoding

Suitable for ordinal data. In this technique, each label is assigned a unique integer based on alphabetical ordering.

```
import pandas as pd

df = pd.DataFrame({
    'name': ['J.A.R.V.I.S', 'WALL-E', 'Baymax', 'BB-8', 'Cortana'],
    'color': ['blue', 'brown', 'white', 'yellow', 'blue']
    })
df
```

	name	color
0	J.A.R.V.I.S	blue
1	WALL-E	brown
2	Baymax	white
3	BB-8	yellow
4	Cortana	blue

```
df['encoded_color'] = preprocessing.LabelEncoder().fit_transform(df.color)
df
```

	name	color	encoded_color
0	J.A.R.V.I.S	blue	0
1	WALL-E	brown	1
2	Baymax	white	2
3	BB-8	yellow	3
4	Cortana	blue	0

One-Hot Encoding

Suitable for nominal data. Though label encoding is straight-forward but it has the disadvantage that the numeric values can be misinterpreted by algorithms as having some sort of hierarchy/order in them.

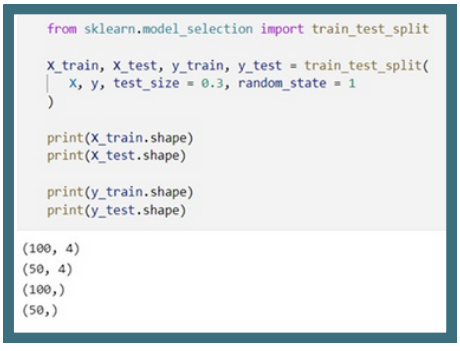
```
# creating additional columns for one hot encoder
color_columns = []
for col in df['color'].unique():
    color_columns.append(col)

onehot_encoded_color = preprocessing.OneHotEncoder().fit_transform(df[['color']])
df2 = pd.DataFrame(onehot_encoded_color.toarray(), columns=color_columns)
pd.concat([df, df2], axis=1)
```

	name	color	encoded_color	blue	brown	white	yellow
0	J.A.R.V.I.S	blue	0	1.0	0.0	0.0	0.0
1	WALL-E	brown	1	0.0	1.0	0.0	0.0
2	Baymax	white	2	0.0	0.0	1.0	0.0
3	BB-8	yellow	3	0.0	0.0	0.0	1.0
4	Cortana	blue	0	1.0	0.0	0.0	0.0

Scikit Learn Modelling Preprocess – Splitting the Dataset

To check the accuracy of our model, we can split the dataset into two pieces- a training set and a testing set.

- X, y – Here, X is the feature matrix and y is the response vector, which need to be split.
- test_size – This represents the ratio of test data to the total given data.
- random_size – It is used to guarantee that the split will always be the same. This is useful in the situations where you want reproducible results.



```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.3, random_state = 1
)

print(X_train.shape)
print(X_test.shape)

print(y_train.shape)
print(y_test.shape)
```

```
(100, 4)
(50, 4)
(100,)
(50,)
```

Scikit Learn Modelling Preprocess – Train the Model

Scikit-learn has a wide range of Machine Learning (ML) algorithms which have a consistent interface for fitting, predicting accuracy, recall etc..

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.3, random_state = 1
)

print(X_train.shape)
print(X_test.shape)

print(y_train.shape)
print(y_test.shape)
```

```
(100, 4)
(50, 4)
(100,)
(50,)
```

Scikit Learn Modelling Preprocess – Train the Model

Once you train the model, it is desirable that the model should be persist for future use so that we do not need to retrain it again and again.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
classifier_knn = KNeighborsClassifier(n_neighbors = 3)
classifier_knn.fit(X_train, y_train)
y_pred = classifier_knn.predict(X_test)

# Finding accuracy by comparing actual response values(y_test)with predicted response value(y_pred)
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))

# Providing sample data and the model will make prediction out of that data
sample = [[5, 5, 3, 2], [2, 4, 3, 5]]
preds = classifier_knn.predict(sample)
pred_species = [iris.target_names[p] for p in preds]
print("Predictions:", pred_species)
```

```
Accuracy: 0.9833333333333333
Predictions: ['versicolor', 'virginica']
```

Scikit Learn Modelling Preprocess – Model Persistence

Once you train the model, it is desirable that the model should be persist for future use so that we do not need to retrain it again and again.

```
import joblib
joblib.dump(classifier_knn, 'iris_classifier_knn.joblib')
```

```
['iris_classifier_knn.joblib']
```

```
joblib.load('iris_classifier_knn.joblib')
```

```
KNeighborsClassifier(n_neighbors=3)
```

The above code will save the model into file named iris_classifier_k n.joblib. Now, the object can be reloaded from the file with the help of following code –

Picture Source

- https://rstefanus16.medium.com/conventional-programming-vs-machine-learning-a3b7b3425531
- https://www.edureka.co/blog/introduction-to-machine-learning/
- https://towardsdatascience.com/introduction-to-machine-learning-for-beginners-eed6024fdb08
- pexels.com
- Youtube Channel LegaC-Mathematics Resources