SNS COLLEGE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION)

COIMBATORE – 35

DEPARTMENT OF COMPUTER SIENCE AND ENGINEERING

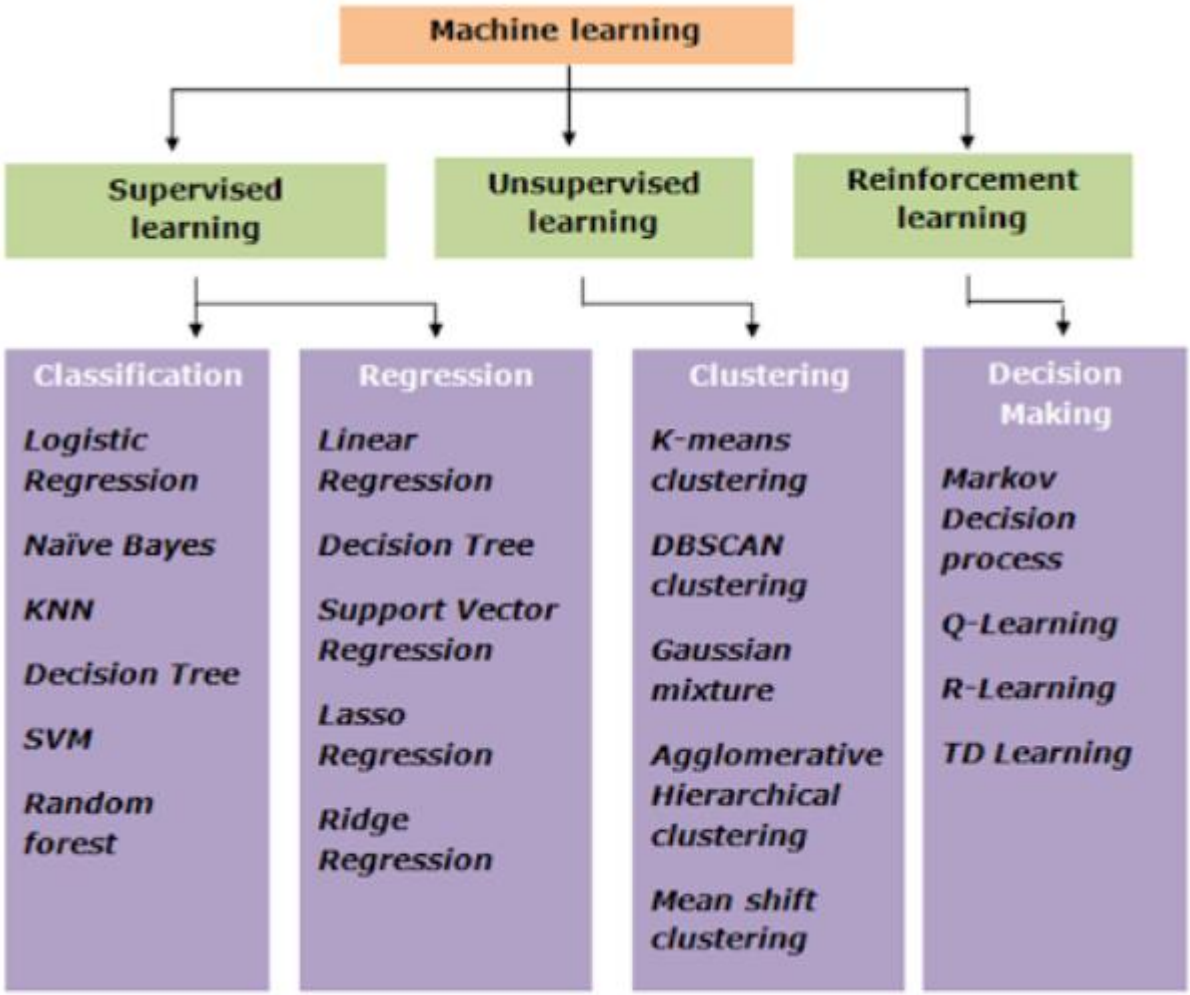
UNIT 1
INTRODUCTION

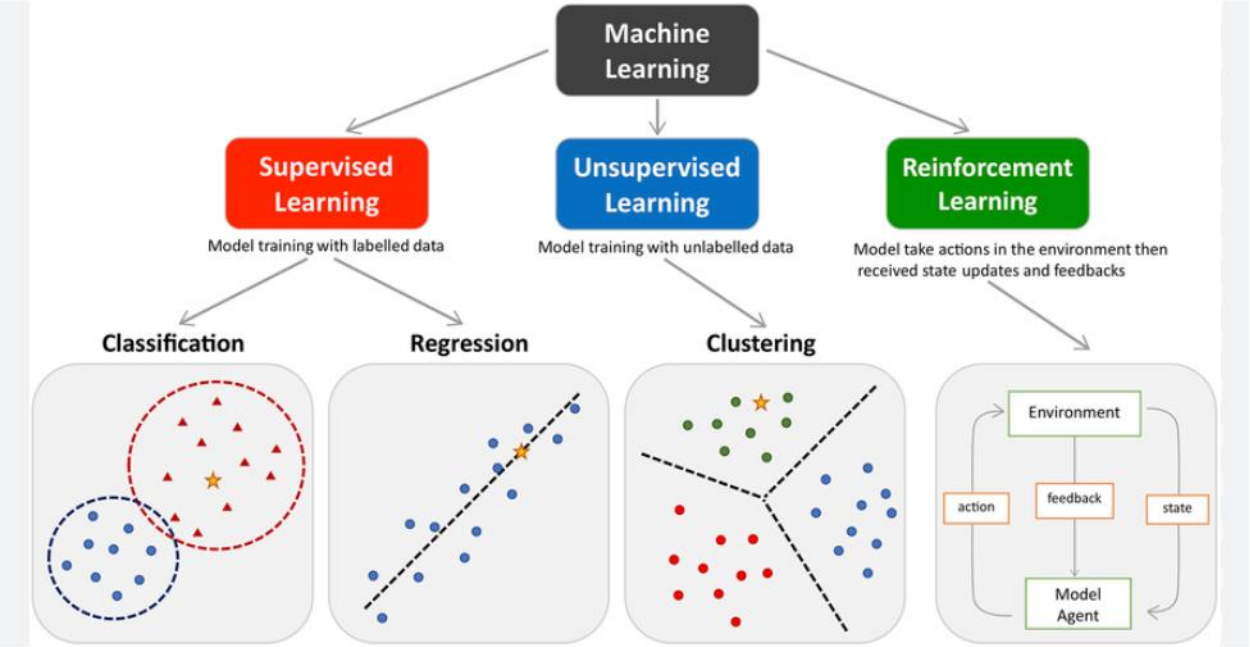
Machine Learning – perspective -Issues
Examples of Machine Learning Applications
Types of Machine Learning –Machine learning process- preliminaries, testing
Machine Learning algorithms
Turning data into Probabilities, and Statistics for Machine Learning
Probability theory -Bayesian Decision Theory.

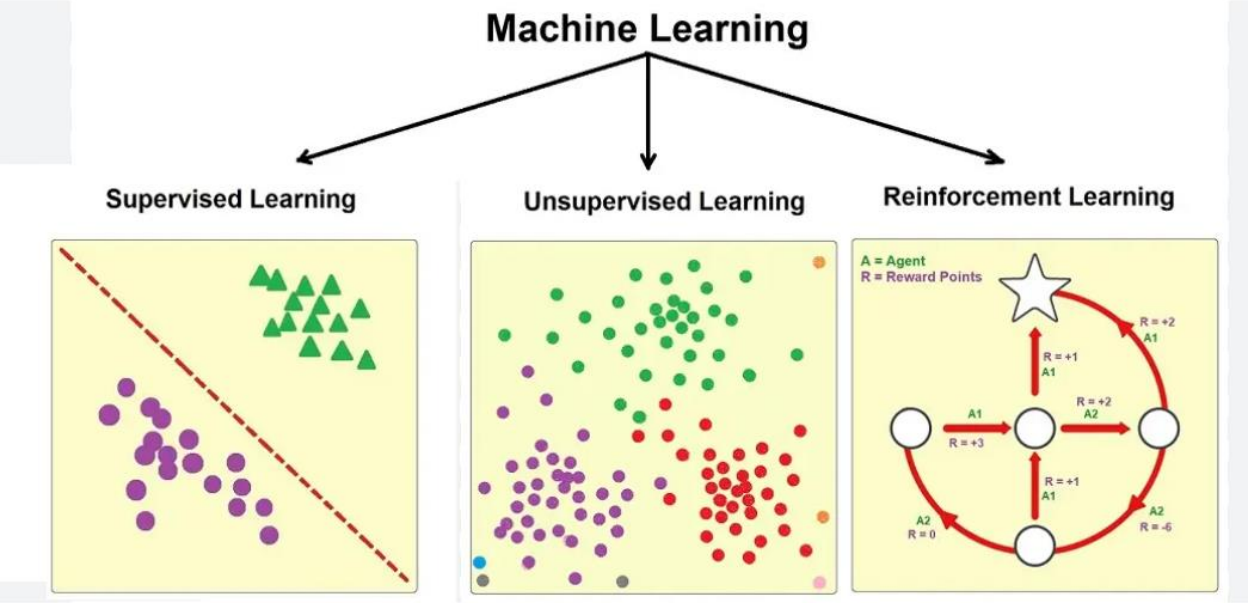
Types of machine learning algorithms

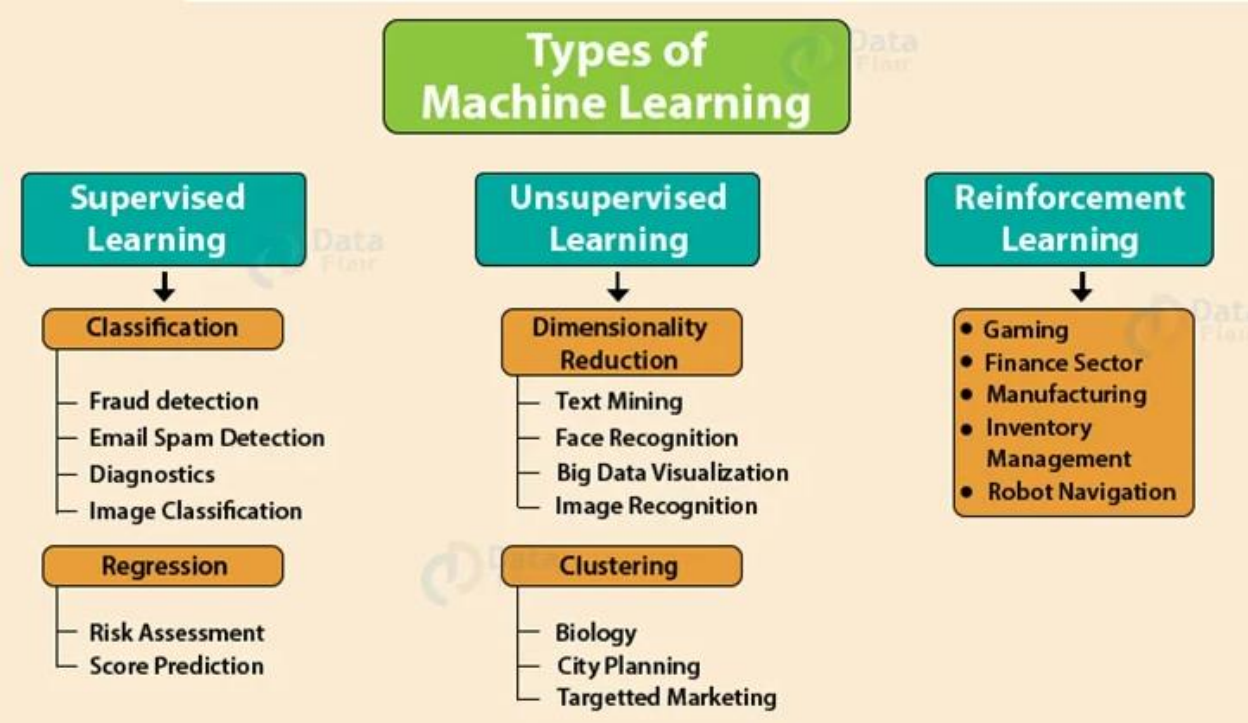
There are several types of machines learning algorithms, including

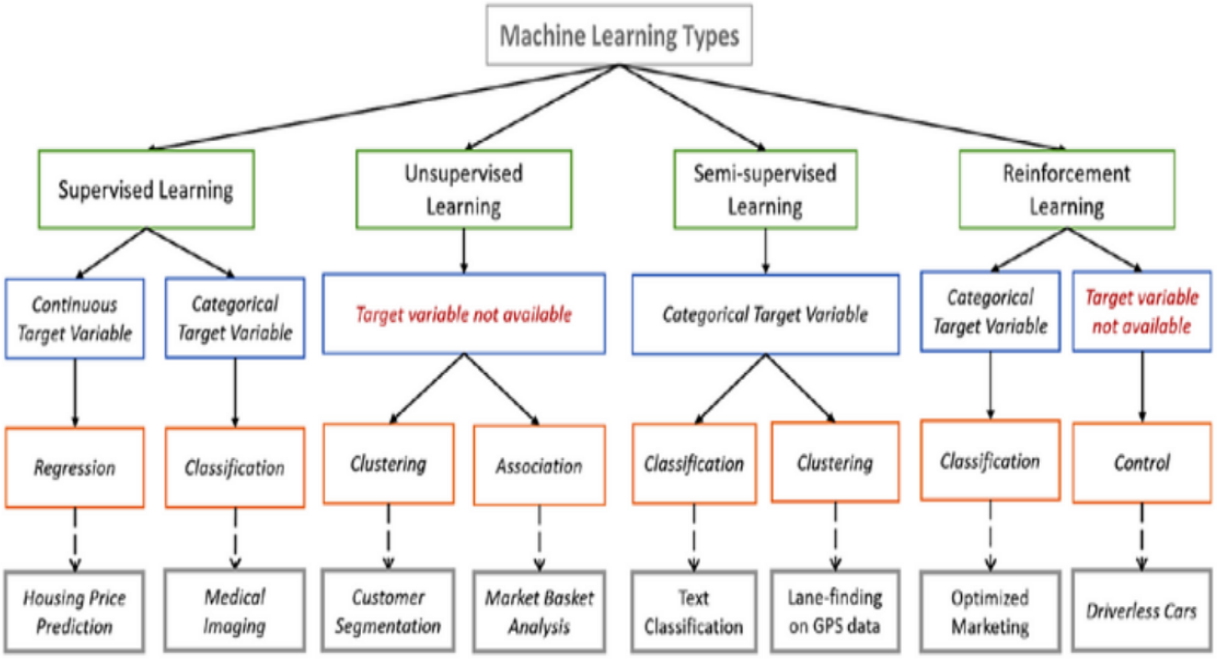
- supervised learning (where the algorithm learns from labeled data),
- unsupervised learning (where the algorithm finds patterns in unlabeled data), and
- reinforcement learning (where the algorithm learns to make decisions by interacting with an environment and receiving feedback).


Machine learning
Supervised learning
Unsupervised learning
Reinforcement learning
Classification
Logistic Regression
Naïve Bayes
KNN
Decision Tree
SVM
Random forest
Regression
Linear Regression
Decision Tree
Support Vector Regression
Lasso Regression
Ridge Regression
Clustering
K-means clustering
DBSCAN clustering
Gaussian mixture
Agglomerative Hierarchical clustering
Mean shift clustering
Decision Making
Markov Decision process
Q-Learning
R-Learning
TD Learning


Machine Learning
Supervised Learning
Model training with labelled data
Unsupervised Learning
Model training with unlabelled data
Reinforcement Learning
Model take actions in the environment then received state updates and feedbacks
Classification
Regression
Clustering
Environment
action
feedback
state
Model Agent


Machine Learning
Supervised Learning
Unsupervised Learning
Reinforcement Learning
A = Agent
R = Reward Points
R = +2
A1
R = +1
A1
R = +2
A2
A1
R = +3
R = +1
A1
A2
R = 0
A2
R = -6

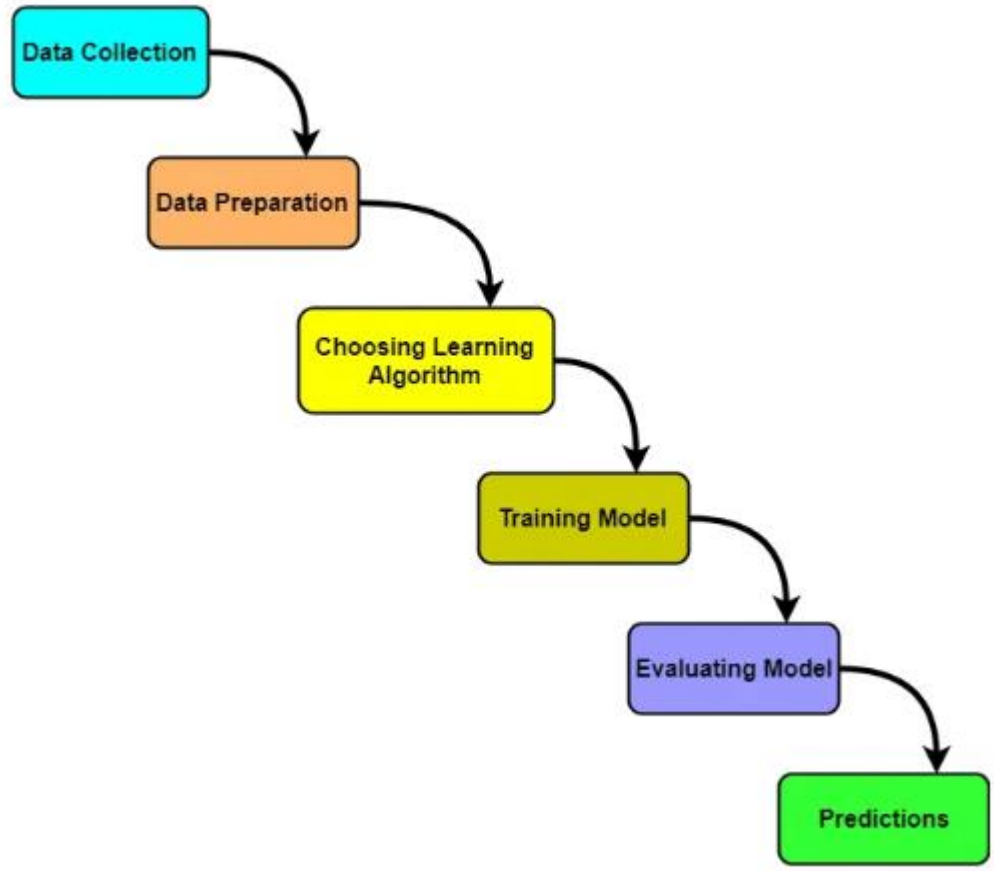

Types of Machine Learning
Supervised Learning
Classification
Fraud detection
Email Spam Detection
Diagnostics
Image Classification
Regression
Risk Assessment
Score Prediction
Unsupervised Learning
Dimensionality Reduction
Text Mining
Face Recognition
Big Data Visualization
Image Recognition
Clustering
Biology
City Planning
Targetted Marketing
Reinforcement Learning
Gaming
Finance Sector
Manufacturing
Inventory Management
Robot Navigation


Machine Learning Types
Supervised Learning
Unsupervised Learning
Semi-supervised Learning
Reinforcement Learning
Continuous Target Variable
Categorical Target Variable
Target variable not available
Categorical Target Variable
Categorical Target Variable
Target variable not available
Regression
Classification
Clustering
Association
Classification
Clustering
Classification
Control
Housing Price Prediction
Medical Imaging
Customer Segmentation
Market Basket Analysis
Text Classification
Lane-finding on GPS data
Optimized Marketing
Driverless Cars

Machine learning process - preliminaries

Machine learning workflow refers to the series of stages or steps involved in the process of building a successful machine learning system.

The various stages involved in the machine learning workflow are-



- Data Collection
- Data Preparation
- Choosing Learning Algorithm
- Training Model
- Evaluating Model
- Predictions

1. Data Collection-

In this stage,

- Data is collected from different sources.
- The type of data collected depends upon the type of desired project.
- Data may be collected from various sources such as files, databases etc.

- The quality and quantity of gathered data directly affects the accuracy of the desired system.

2. Data Preparation-

In this stage,
- Data preparation is done to clean the raw data.
- Data collected from the real world is transformed to a clean dataset.
- Raw data may contain missing values, inconsistent values, duplicate instances etc.
- So, raw data cannot be directly used for building a model.

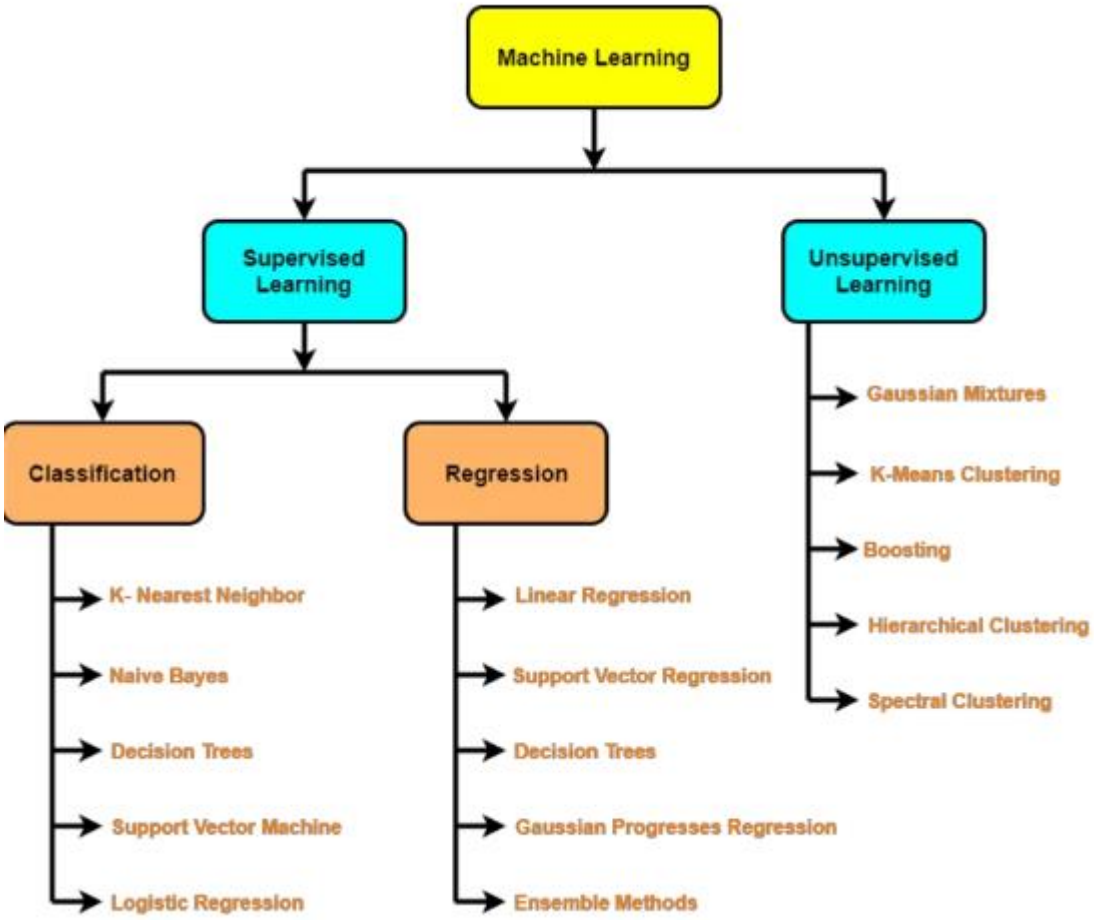
Different methods of cleaning the dataset are-
- Ignoring the missing values
- Removing instances having missing values from the dataset.
- Estimating the missing values of instances using mean, median or mode.
- Removing duplicate instances from the dataset.
- Normalizing the data in the dataset.

This is the most time consuming stage in machine learning workflow.

3. Choosing Learning Algorithm-

In this stage,
- The best performing learning algorithm is researched.
- It depends upon the type of problem that needs to solved and the type of data we have.
- If the problem is to classify and the data is labeled, classification algorithms are used.
- If the problem is to perform a regression task and the data is labeled, regression algorithms are used.
- If the problem is to create clusters and the data is unlabeled, clustering algorithms are used.

The following chart provides the overview of learning algorithms-



4. Training Model-

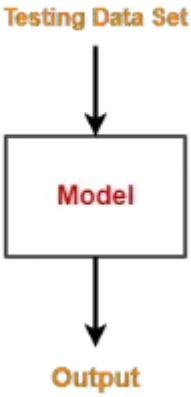
In this stage,

- The model is trained to improve its ability.
- The dataset is divided into training dataset and testing dataset.
- The training and testing split is order of 80/20 or 70/30.
- It also depends upon the size of the dataset.
- Training dataset is used for training purpose.
- Testing dataset is used for the testing purpose.
- Training dataset is fed to the learning algorithm.
- The learning algorithm finds a mapping between the input and the output and generates the model.

5. Evaluating Model-

- The model is evaluated to test if the model is any good.

- The model is evaluated using the kept-aside testing dataset.
- It allows to test the model against data that has never been used before for training.
- Metrics such as accuracy, precision, recall etc are used to test the performance.
- If the model does not perform well, the model is re-built using different hyper parameters.
- The accuracy may be further improved by tuning the hyper parameters.



6. Predictions-

In this stage,

- The built system is finally used to do something useful in the real world.
- Here, the true value of machine learning is realized.

Additional reference for students:

The machine learning process typically involves several key stages, from preliminaries to testing and deployment. Here's a detailed breakdown:

1. Preliminary Steps

a. Problem Definition

- **Objective**: Clearly define the problem you want to solve with machine learning.
- **Scope**: Determine the scope and constraints of the problem.

b. Data Collection

- **Sources**: Identify and gather data from various sources (databases, online repositories, APIs, etc.).
- **Formats**: Ensure data is collected in a suitable format for analysis (e.g., CSV, JSON).

c. Data Preprocessing

- **Cleaning**: Handle missing values, remove duplicates, and correct inconsistencies.
- **Transformation**: Normalize or standardize data, encode categorical variables, and transform features as needed.
- **Splitting**: Divide the data into training, validation, and test sets.

d. Exploratory Data Analysis (EDA)

- **Visualization**: Use plots (e.g., histograms, scatter plots) to understand the distribution and relationships within the data.
- **Statistics**: Calculate summary statistics (mean, median, standard deviation) to grasp data characteristics.
- **Feature Selection**: Identify the most relevant features for the model.

2. Model Development

a. Model Selection

- **Algorithm Choice**: Choose an appropriate machine learning algorithm based on the problem type (classification, regression, clustering).
- **Baseline Model**: Develop a simple model to serve as a performance benchmark.

b. Model Training

- **Training**: Use the training data to fit the model.
- **Hyperparameter Tuning**: Optimize hyperparameters using techniques like grid search, random search, or Bayesian optimization.

c. Model Validation

- **Cross-Validation**: Perform k-fold cross-validation to assess model performance and ensure it generalizes well to new data.
- **Metrics**: Evaluate using appropriate metrics (accuracy, precision, recall, F1 score for classification; MSE, RMSE, MAE for regression).

3. Model Testing

a. Testing on Unseen Data

- **Test Set**: Evaluate the final model on the test set, which was not used during training or validation.
- **Performance Metrics**: Compute the performance metrics on the test set to get an unbiased estimate of model performance.

b. Error Analysis

- **Misclassifications**: Analyze where the model makes errors to understand its weaknesses.
- **Bias-Variance Tradeoff**: Assess if the model is overfitting (high variance) or underfitting (high bias).

4. Model Deployment

a. Model Integration

- **API Development**: Develop an API to serve the model predictions.
- **Infrastructure**: Set up the necessary infrastructure (servers, cloud services) to deploy the model.

b. Monitoring and Maintenance

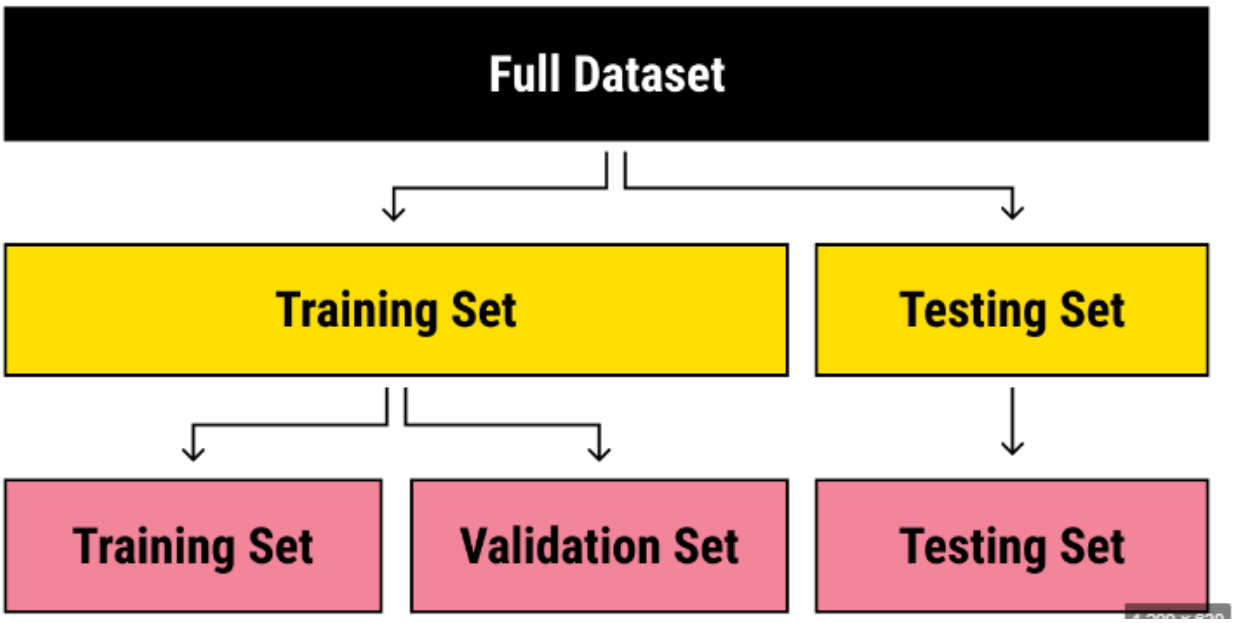
- **Performance Monitoring**: Continuously monitor model performance in the real world.
- **Retraining**: Periodically retrain the model with new data to maintain performance.
- **Feedback Loop**: Incorporate user feedback to improve the model.

5. Documentation and Reporting

- **Documentation**: Document the entire process, from data collection to model deployment, including any assumptions and decisions made.
- **Reporting**: Present the findings, model performance, and potential improvements to stakeholders.

This structured approach ensures a comprehensive and effective machine learning workflow, from initial problem definition to deploying and maintaining the model in a real-world environment.

Testing in ML:



Extra ref link : https://labelyourdata.com/articles/machine-learning-and-training-data

Example video:
Weka tool
https://youtu.be/U63ExiTJMic?si=LyoJq5LxDBZZxA25

Weka training and testing
https://youtu.be/uiDFa7iY9yo?si=mc8H07y_KZYQNKar

Testing in machine learning is a critical step in the model development process that ensures the model's generalization and performance on unseen data. Here's a detailed look at the testing phase:

1. Data Splitting

Training Set: The subset of data used to train the model.
Validation Set: The subset used to tune hyperparameters and select the best model.
Test Set: The subset used to evaluate the final model's performance.

2. Evaluation Metrics

Different tasks require different metrics to evaluate model performance:

- **Classification**:
  - **Accuracy**: The proportion of correctly classified instances.
  - **Precision**: The proportion of true positives among predicted positives.

  - **Recall**: The proportion of true positives among actual positives.
  - **F1 Score**: The harmonic mean of precision and recall.
  - **Confusion Matrix**: A table showing true positives, true negatives, false positives, and false negatives.
  - **ROC-AUC**: The area under the receiver operating characteristic curve.
- **Regression**:
  - **Mean Absolute Error (MAE)**: The average of absolute errors.
  - **Mean Squared Error (MSE)**: The average of squared errors.
  - **Root Mean Squared Error (RMSE)**: The square root of MSE.
  - **R-squared (R^2)**: The proportion of variance in the dependent variable predictable from the independent variables.

3. Cross-Validation

k-Fold Cross-Validation: The dataset is divided into k subsets, and the model is trained and validated k times, each time using a different subset as the validation set and the remaining as the training set. This provides a more robust evaluation.

Ref link: https://youtu.be/PF2wLKv2lsI?si=l_cfl5nLu7FMxTNE

Types of Cross-Validation

1. **k-Fold Cross-Validation**
2. **Stratified k-Fold Cross-Validation**
3. **Leave-One-Out Cross-Validation (LOOCV)**
4. **Leave-P-Out Cross-Validation**
5. **Nested Cross-Validation**
6. **Time Series Cross-Validation**
7. **Shuffle Split Cross-Validation**
8. **Group k-Fold Cross-Validation**

Brief Descriptions

1. **k-Fold Cross-Validation**: The dataset is divided into k equal-sized folds. The model is trained k times, each time using a different fold as the test set and the remaining folds as the training set.
2. **Stratified k-Fold Cross-Validation**: Similar to k-fold, but ensures each fold has approximately the same distribution of class labels as the entire dataset. Useful for imbalanced datasets.
3. **Leave-One-Out Cross-Validation (LOOCV)**: Each data point is used once as a test set while the remaining points form the training set. This is repeated for each data point.
4. **Leave-P-Out Cross-Validation**: Similar to LOOCV, but instead of leaving out one sample, p samples are left out as the test set and the remaining samples form the training set. This is repeated for all combinations.
5. **Nested Cross-Validation**: Used for hyperparameter tuning. The data is split into two loops: an outer loop for model evaluation and an inner loop for hyperparameter tuning.

6. **Time Series Cross-Validation**: Specifically for time series data, it respects the temporal order by splitting the data into training and test sets without shuffling, typically using techniques like forward chaining.
7. **Shuffle Split Cross-Validation**: Randomly splits the data into training and test sets multiple times. Each split is independent of the others.
8. **Group k-Fold Cross-Validation**: Ensures that the same group is not represented in both the training and test sets. Useful when data points are grouped by an identifier (e.g., patient ID).

These cross-validation techniques help in robust model evaluation and selection, catering to different types of data and specific needs of the machine learning problem at hand.

4. Model Testing Process

1. **Train the Model**: Use the training set to train the model.

python
Copy code
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
```

2. **Validate the Model**: Use the validation set to tune hyperparameters and choose the best model.

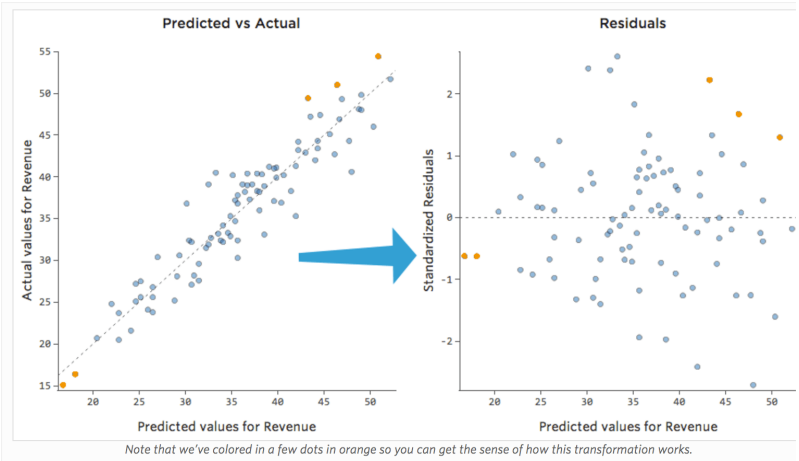
python
Copy code
```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, X_train, y_train, cv=5,
scoring='neg_mean_squared_error')
print("Validation MSE:", -scores.mean())
```

3. **Test the Model**: Evaluate the final model on the test set.

python
Copy code
```
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Test MSE:", mse)
```

5. Error Analysis

- **Residual Analysis**: Plot residuals to check for patterns that suggest model misfit.



- **Confusion Matrix (for Classification)**: Analyze where the model makes errors.

Actual Class \ Predicted Class	Positive	Negative	
Positive	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
	Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

python
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

- **Visual Inspection**: Plot predicted vs actual values.

python
plt.scatter(y_test, y_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted')
plt.show()

6. Performance Monitoring

- **Deployment Testing**: Test the model in a real-world environment.
- **A/B Testing**: Compare the model with a baseline or previous model in production.
- **Monitoring**: Continuously monitor the model's performance using metrics like precision, recall, and latency.

Example: Testing a Machine Learning Model

Here's a simple example using the Boston housing dataset with a Linear

Regression model:

```
python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

# Load the dataset
from sklearn.datasets import load_boston
boston = load_boston()
X = pd.DataFrame(boston.data, columns=boston.feature_names)
y = pd.Series(boston.target)

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)

# Train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Validate the model using cross-validation
scores = cross_val_score(model, X_train, y_train, cv=5,
scoring='neg_mean_squared_error')
print("Validation MSE:", -scores.mean())

# Test the model
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Test MSE:", mse)
```

```
# Plot predicted vs actual values
plt.scatter(y_test, y_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted')
plt.show()
```

This script covers data splitting, model training, validation, testing, and basic error analysis. Adjust this example for your specific use case and dataset.