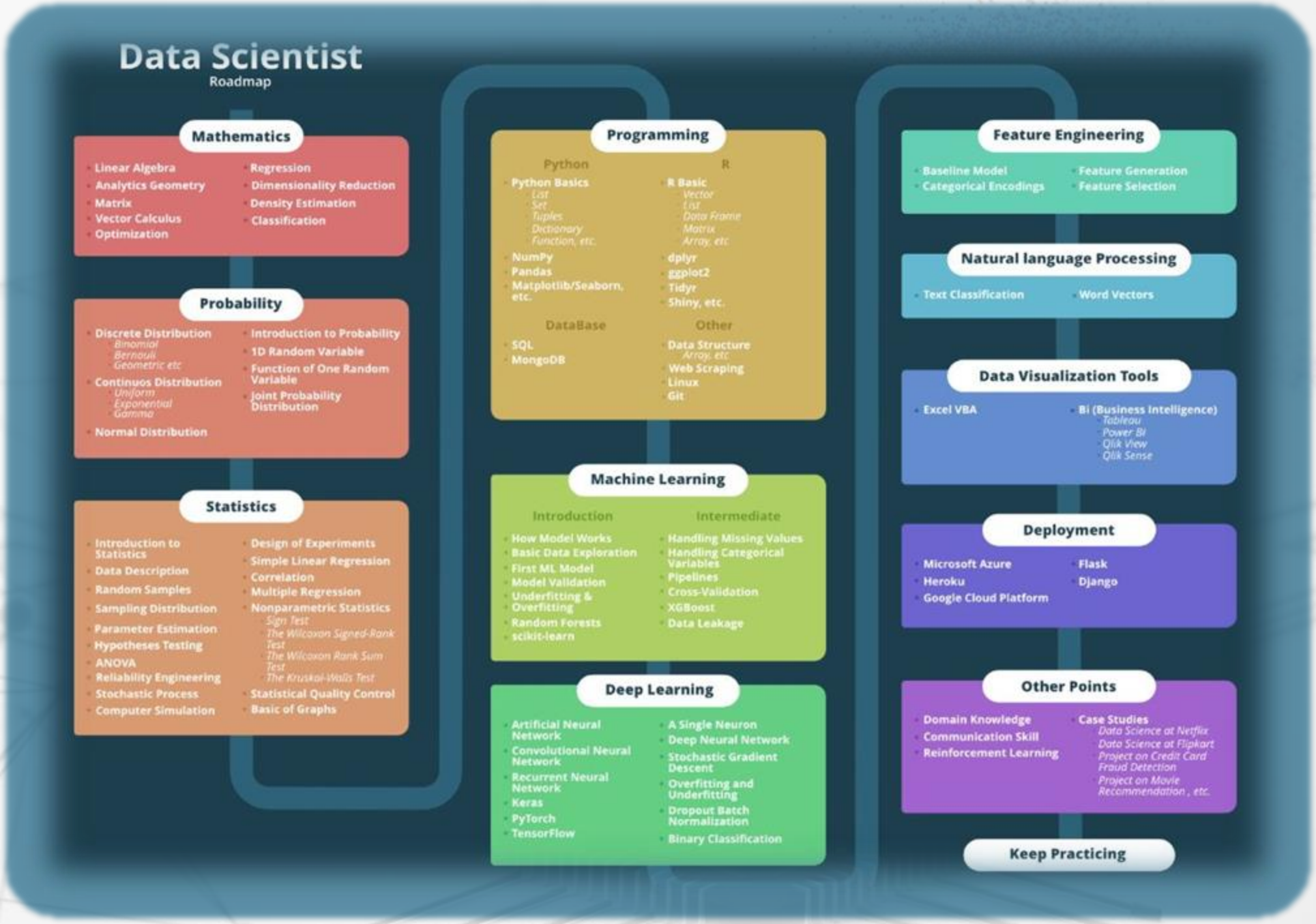
Chapter 1:Introduction to Machine Learning

Ph.D. Nguyen Thi Khanh Tien

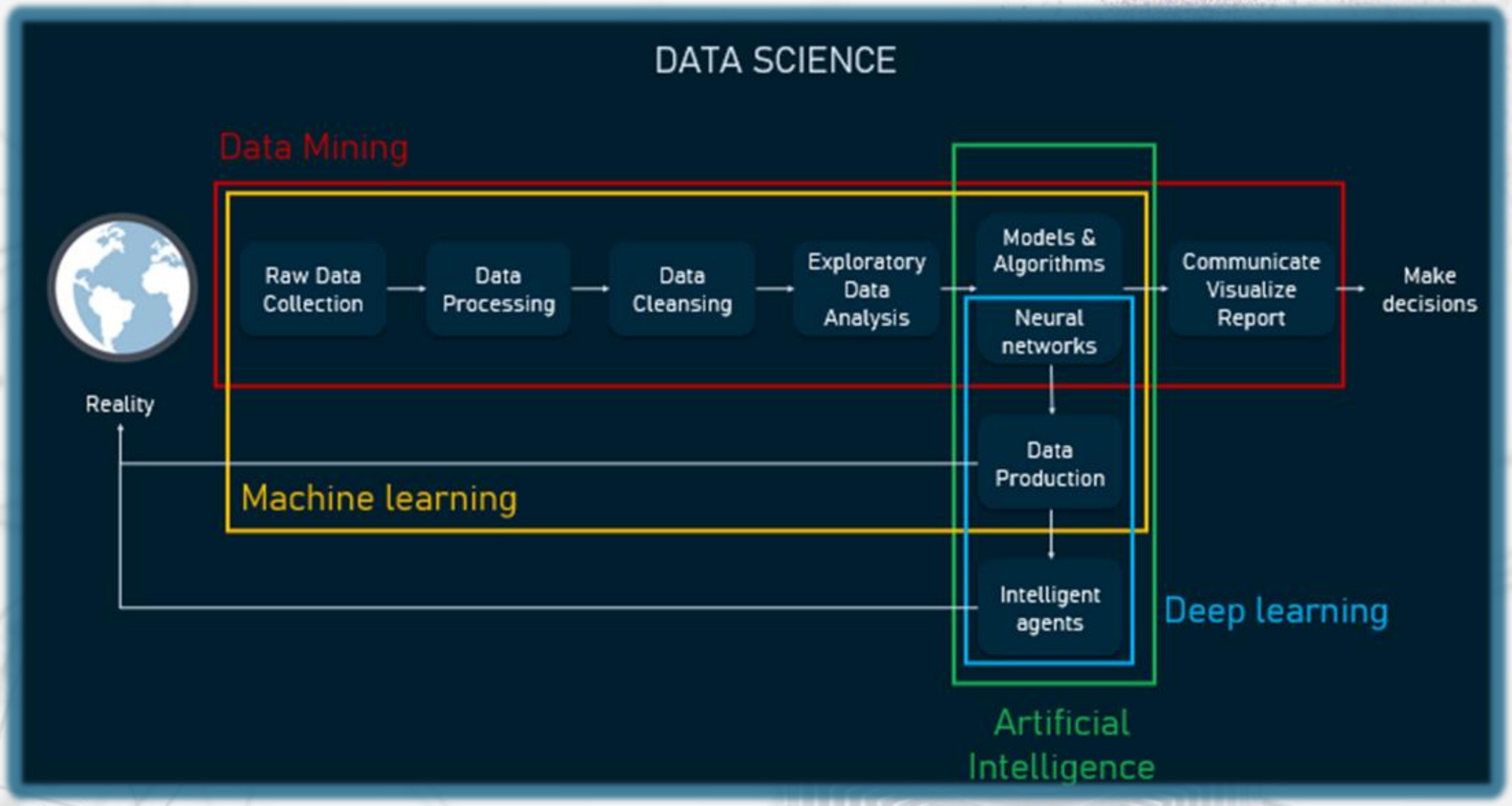
tiennntk@ut.edu.vn

DS Roadmap (Recap)



DS Roadmap (Recap)

Data Science Pipeline



AI, ML, DL, GenAI

Artificial Intelligence

AI involves techniques that equip computers to emulate human behavior, enabling them to learn, make decisions, recognize patterns, and solve complex problems in a manner akin to human intelligence.

Machine Learning

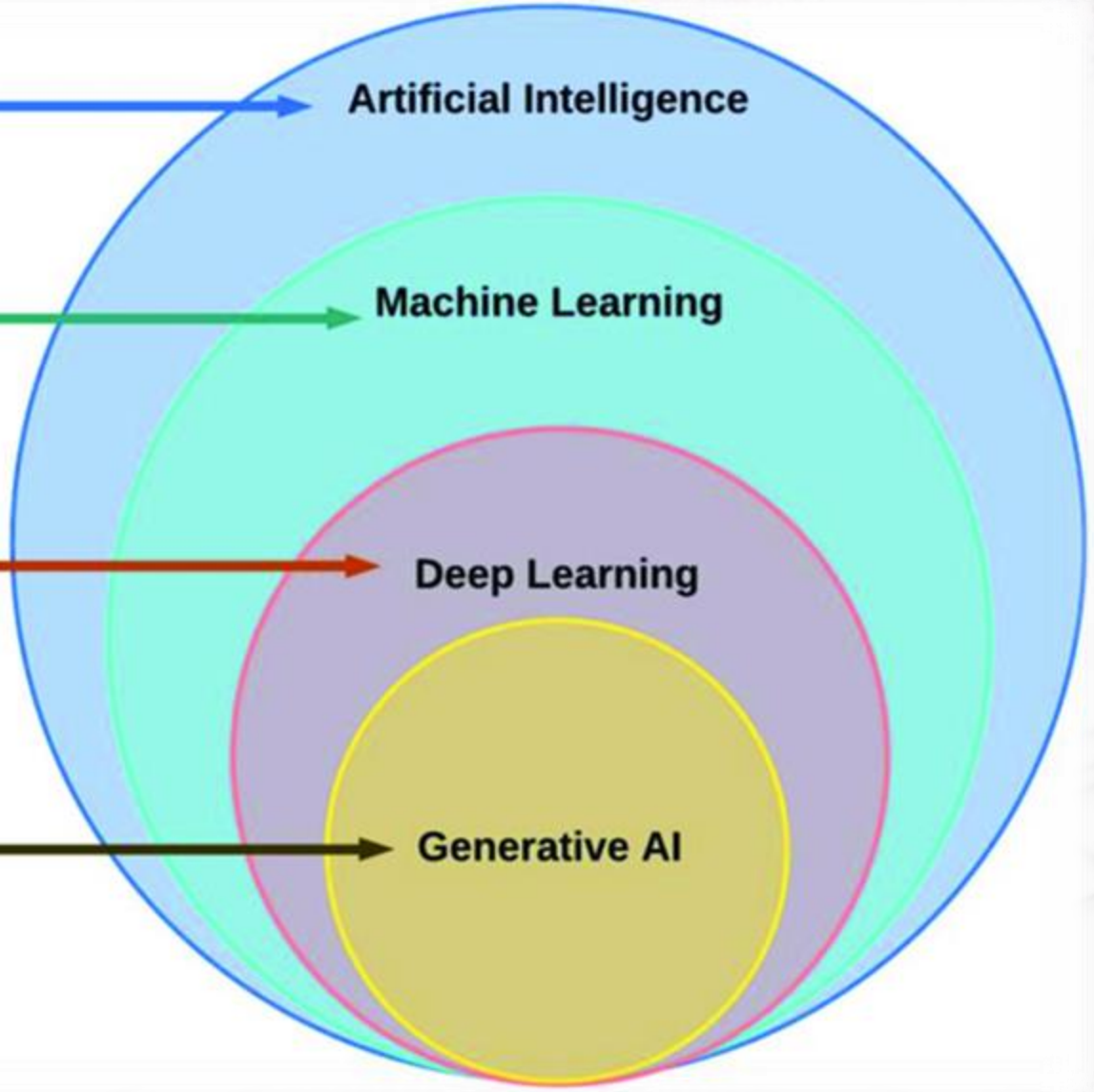
ML is a subset of AI, uses advanced algorithms to detect patterns in large data sets, allowing machines to learn and adapt. ML algorithms use supervised or unsupervised learning methods.

Deep Learning

DL is a subset of ML which uses neural networks for in-depth data processing and analytical tasks. DL leverages multiple layers of artificial neural networks to extract high-level features from raw input data, simulating the way human brains perceive and understand the world.

Generative AI

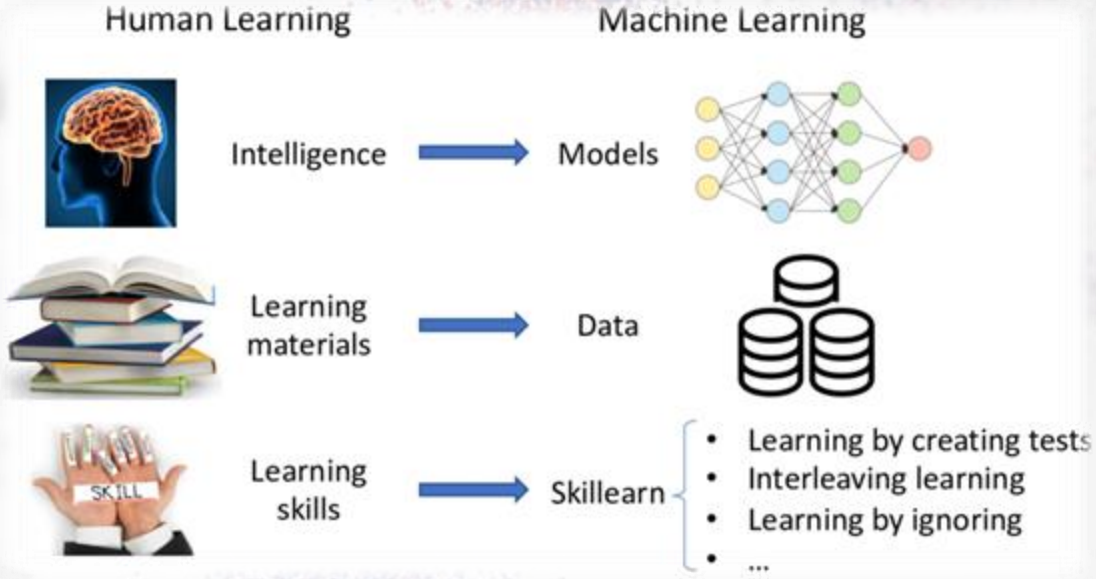
Generative AI is a subset of DL models that generates content like text, images, or code based on provided input. Trained on vast data sets, these models detect patterns and create outputs without explicit instruction, using a mix of supervised and unsupervised learning.

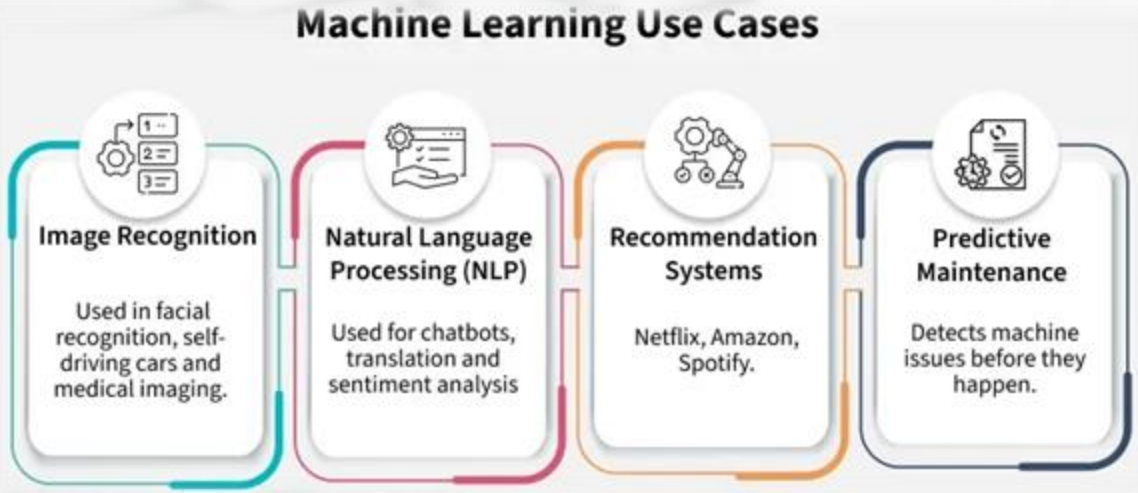


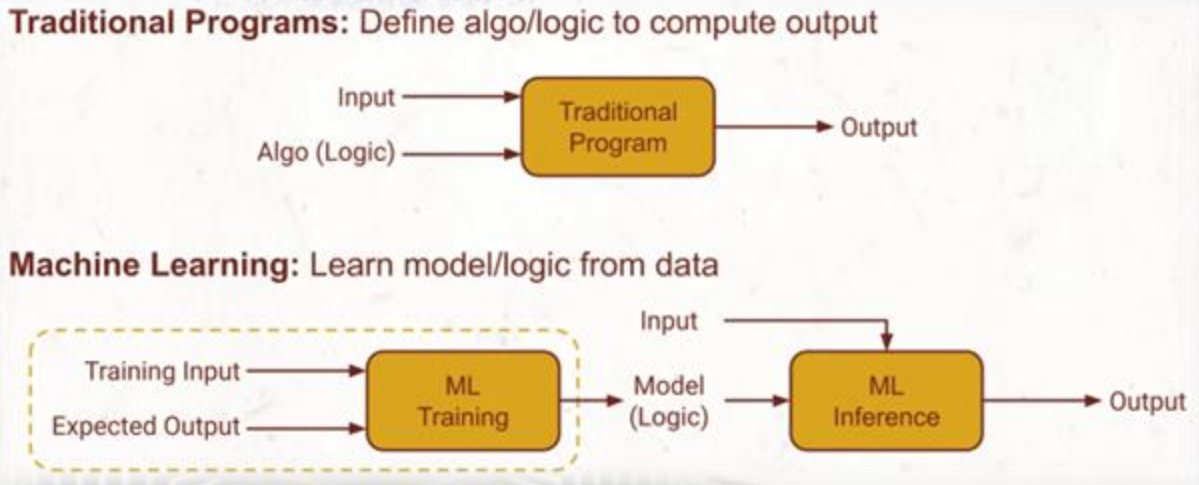
Understanding Machine Learning

Machine learning is a subset of artificial intelligence that enables computers to learn from data and improve their performance on a specific task without being explicitly programmed.

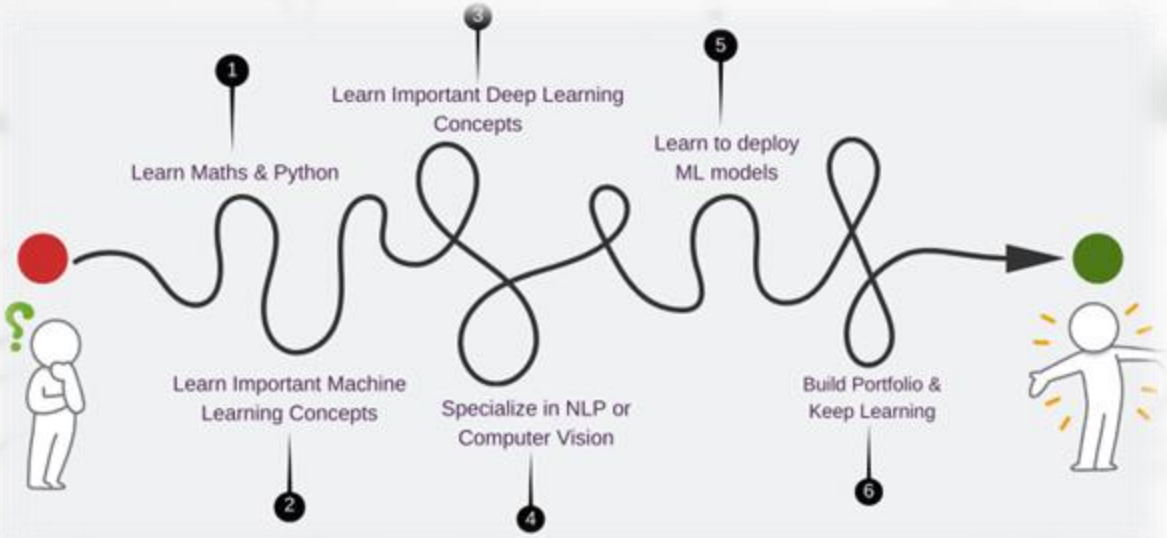
Python has become the de facto language for machine learning due to its simplicity, readability, and extensive libraries.

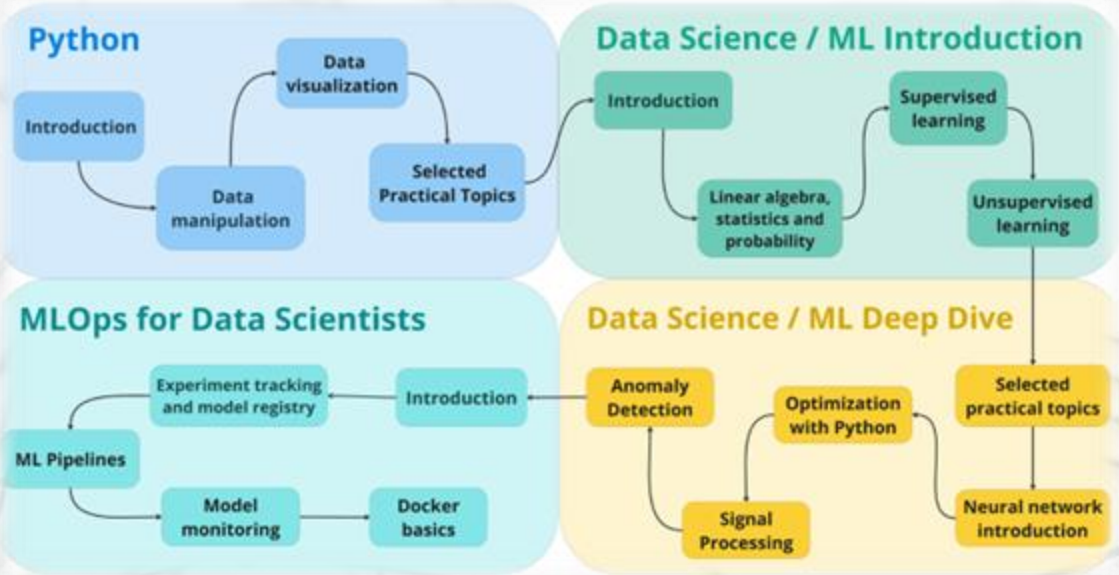


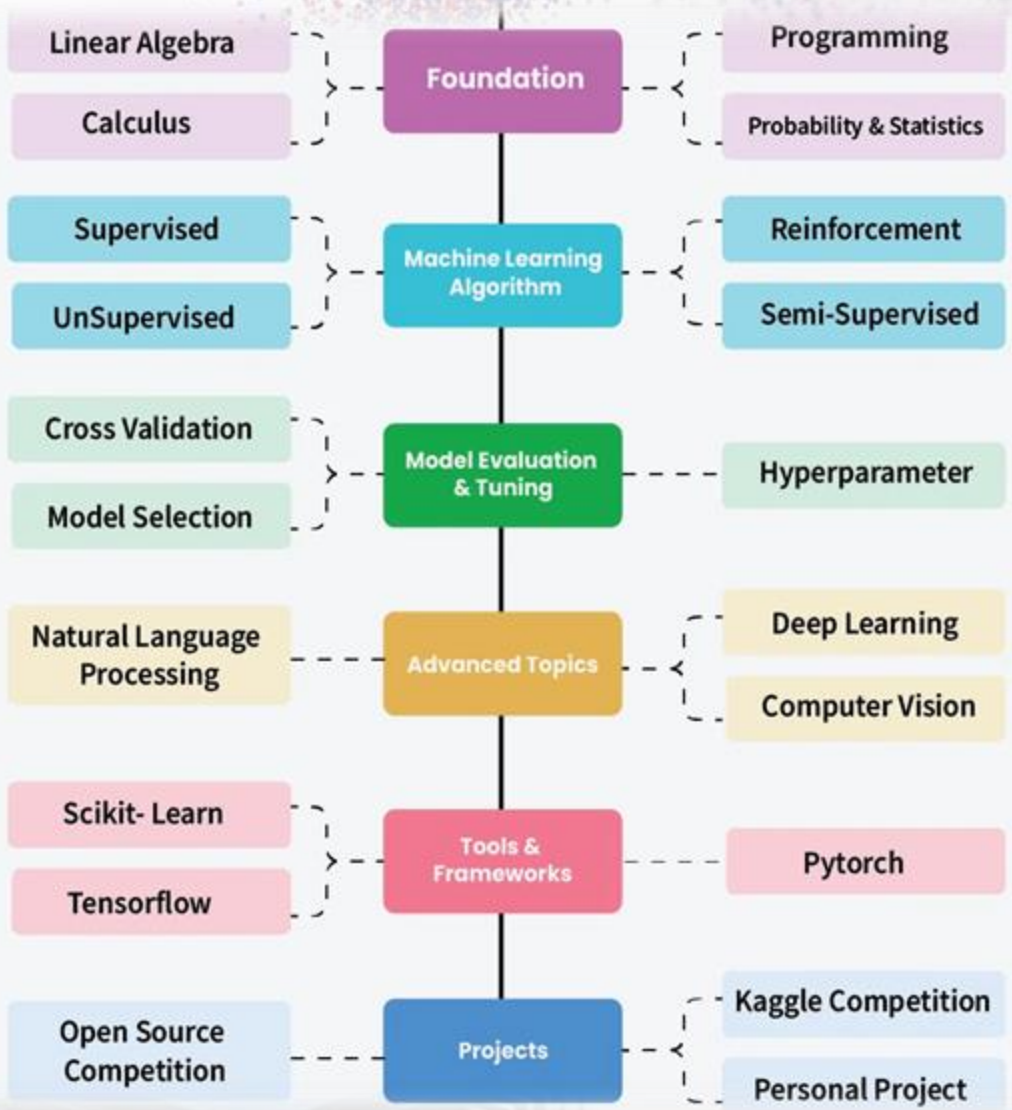




Roadmap to learn ML



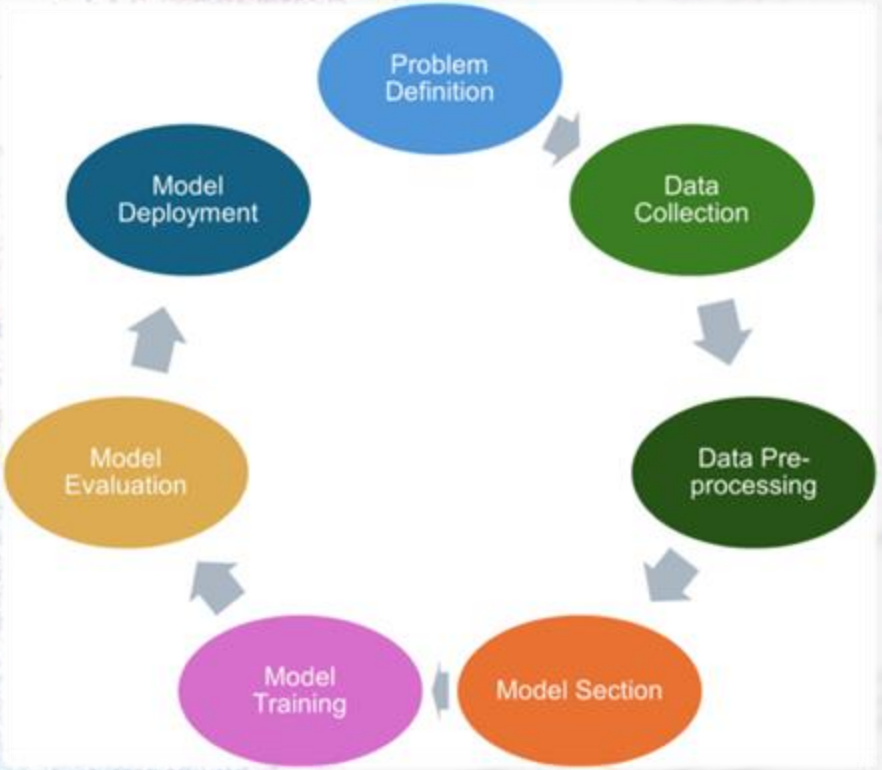




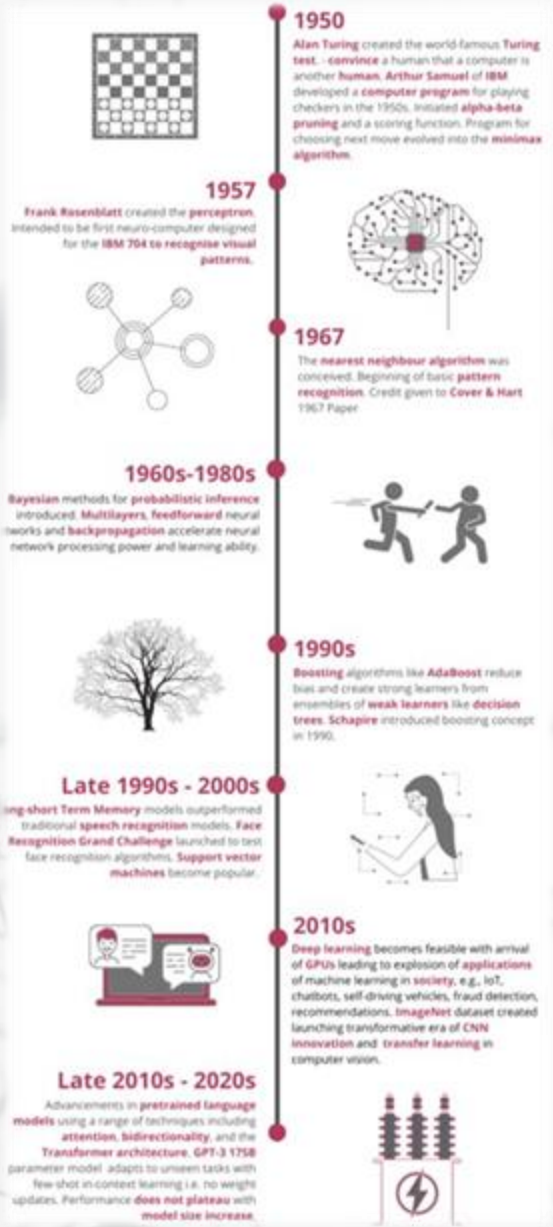
Roadmap to learn ML

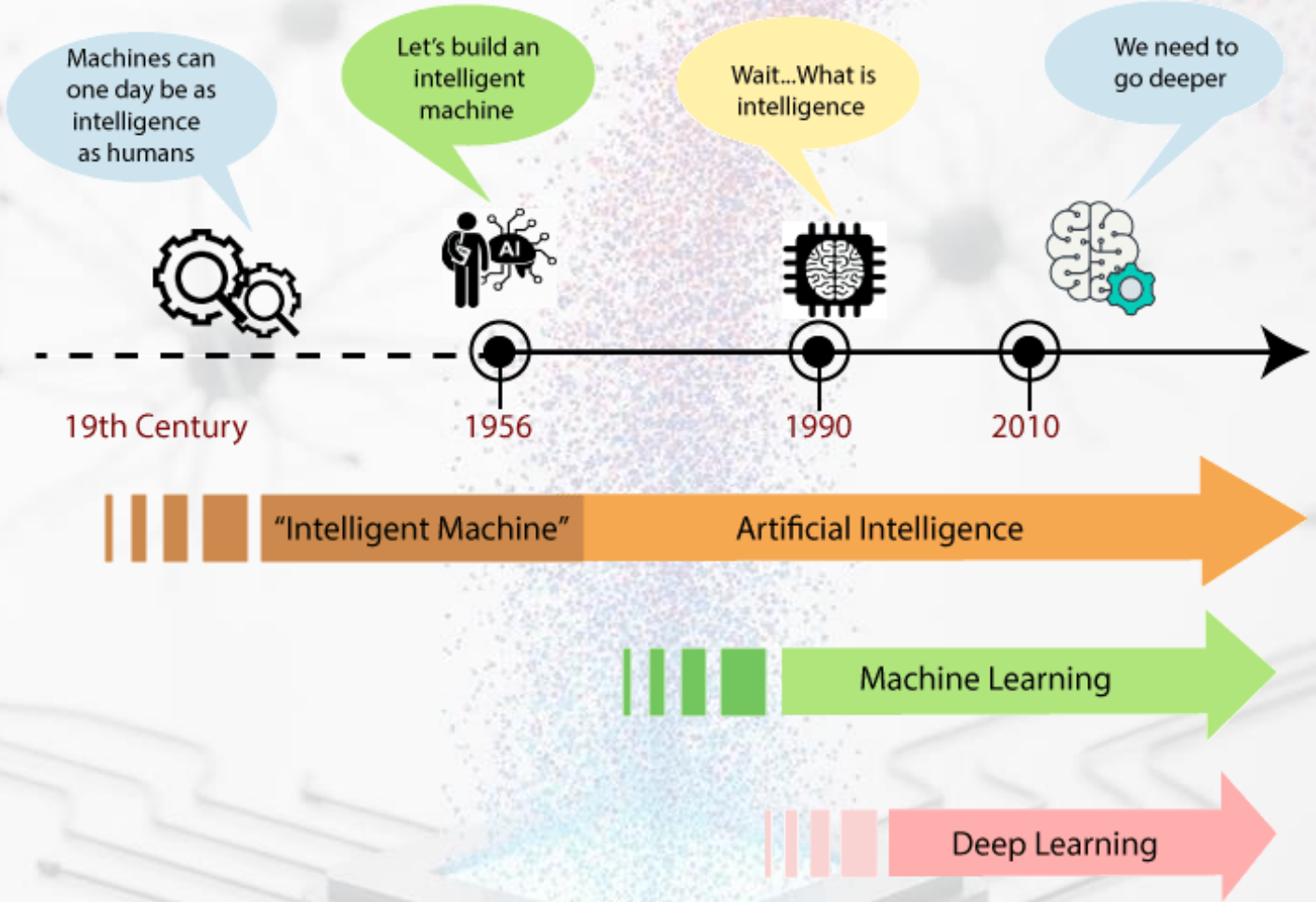
Key Concepts in Machine Learning

- **Data:** The raw material that machines learn from. It can be structured (e.g., CSV files) or unstructured (e.g., images, text).
- **Algorithms:** The methods used to process data and extract patterns. Examples include linear regression, decision trees, random forests, and neural networks.
- **Model:** A representation of the learned patterns. It can be used to make predictions on new, unseen data.
- **Training:** The process of feeding data to an algorithm to teach it to recognize patterns.
- **Testing:** Evaluating the model's performance on a separate dataset to assess its accuracy.

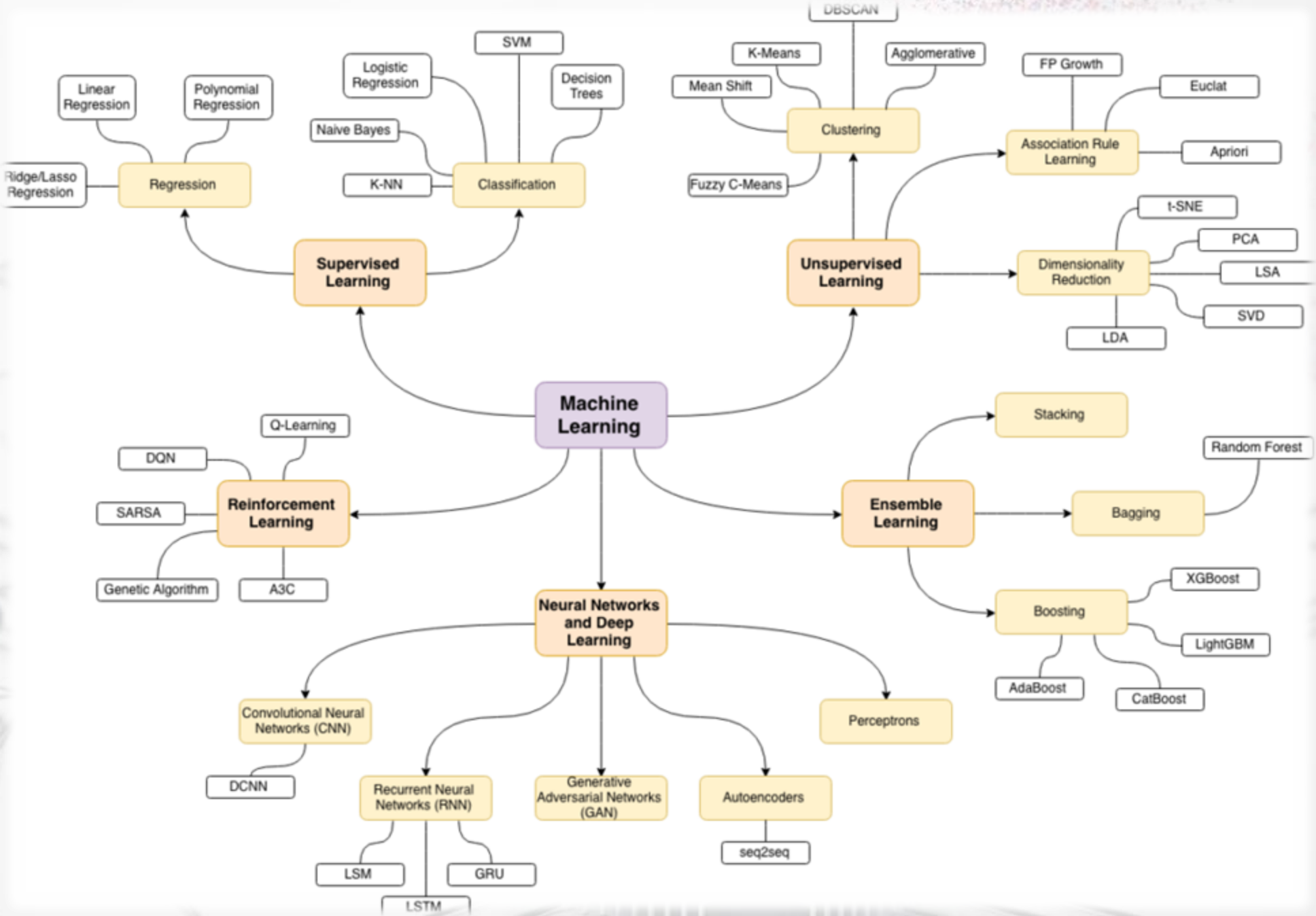


A Brief History of ML

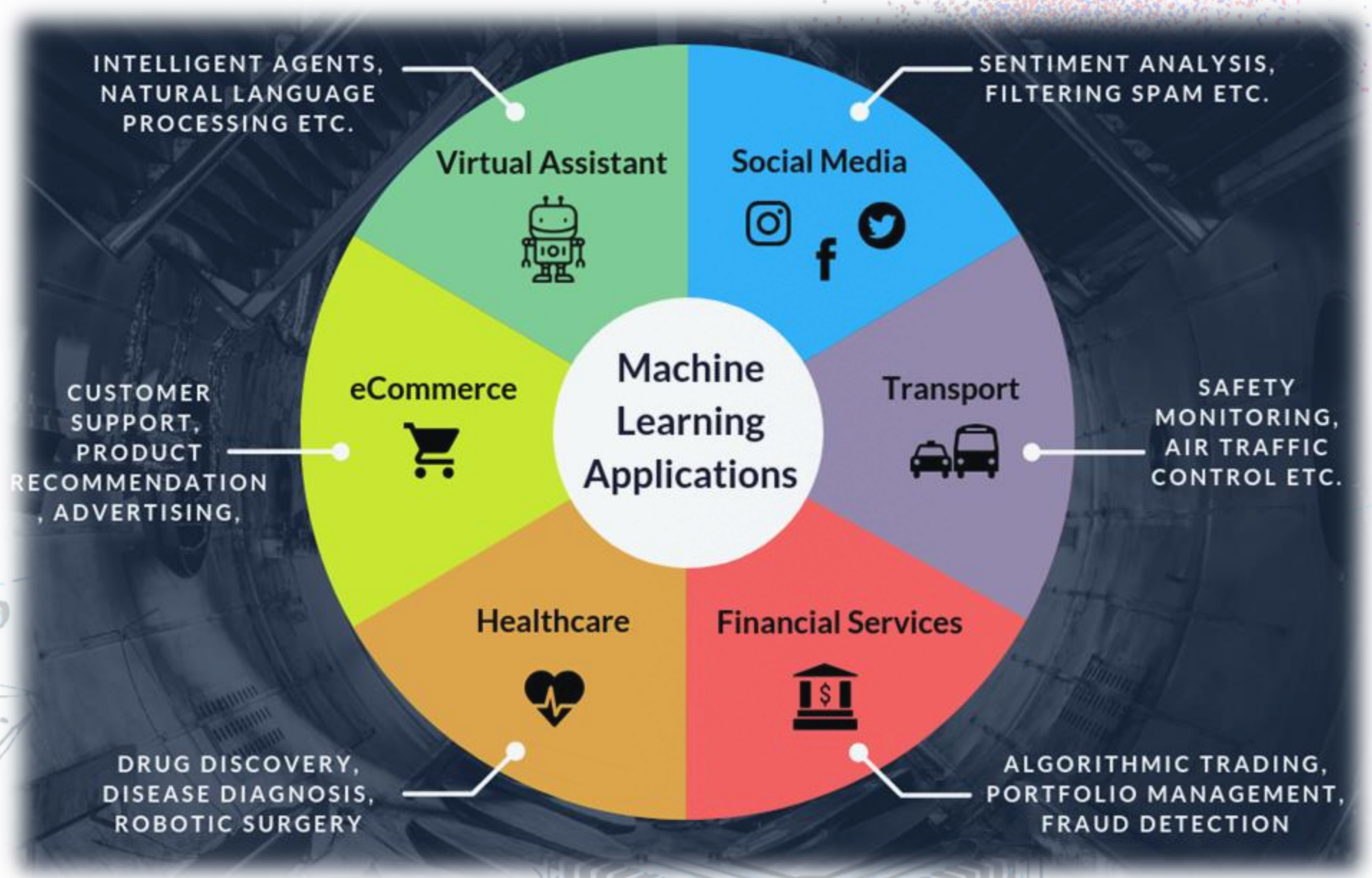




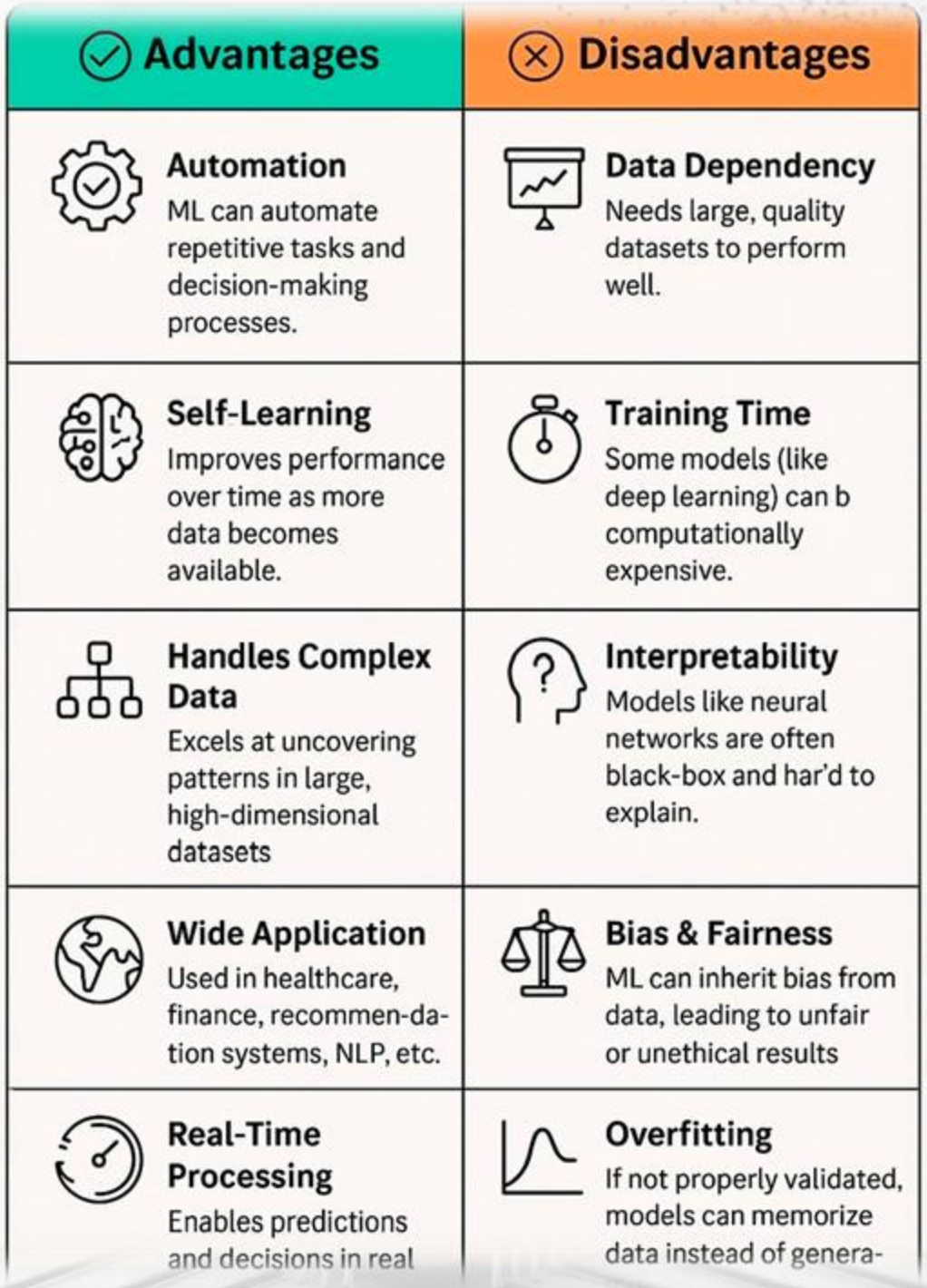
The map of ML


Machine Learning
Supervised Learning
Regression
Linear Regression
Polynomial Regression
Ridge/Lasso Regression
Classification
Logistic Regression
SVM
Decision Trees
Naive Bayes
K-NN
Unsupervised Learning
Clustering
K-Means
Agglomerative
Mean Shift
Fuzzy C-Means
DBSCAN
Association Rule Learning
FP Growth
Euclat
Apriori
Dimensionality Reduction
t-SNE
PCA
LSA
SVD
LDA
Reinforcement Learning
Q-Learning
DQN
SARSA
Genetic Algorithm
A3C
Ensemble Learning
Stacking
Bagging
Random Forest
Boosting
XGBoost
LightGBM
AdaBoost
CatBoost
Neural Networks and Deep Learning
Convolutional Neural Networks (CNN)
DCNN
Recurrent Neural Networks (RNN)
LSM
GRU
LSTM
Generative Adversarial Networks (GAN)
Autoencoders
seq2seq
Perceptrons

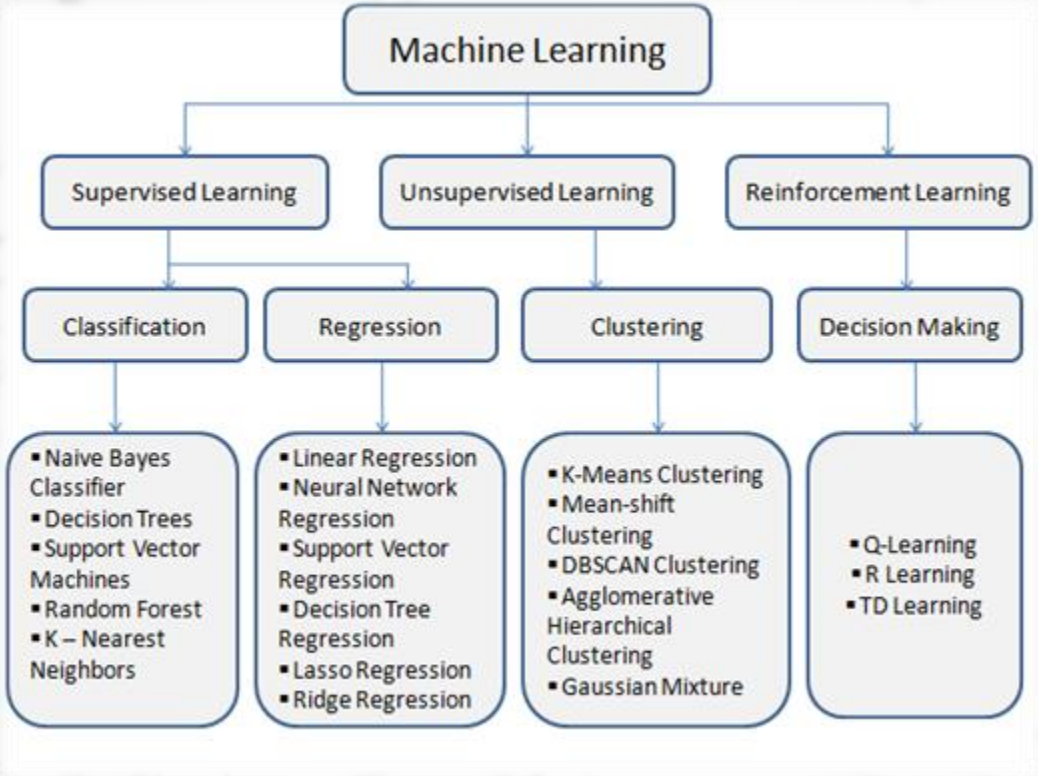
Application of ML

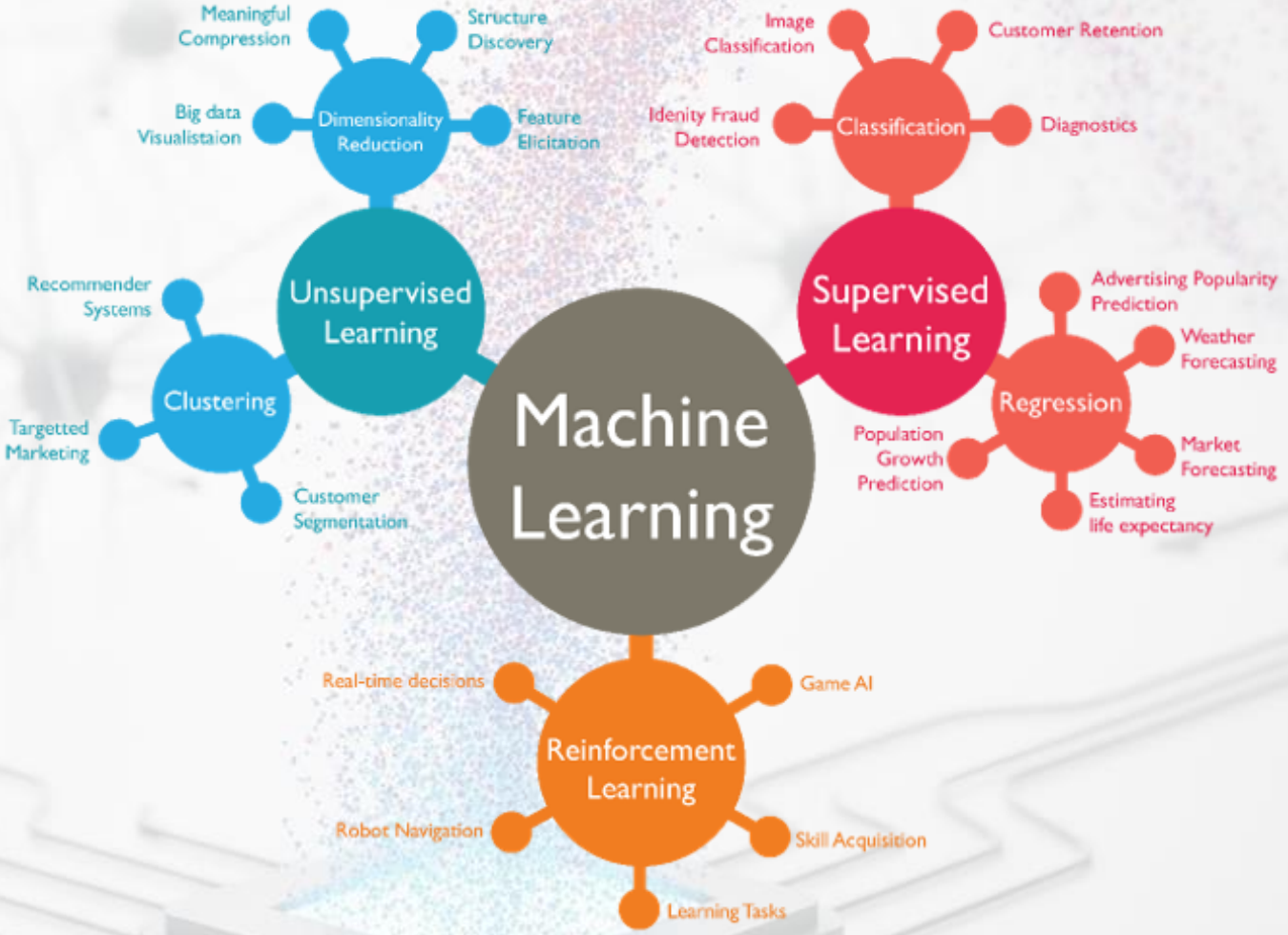


Advantage & Disadvantage of ML

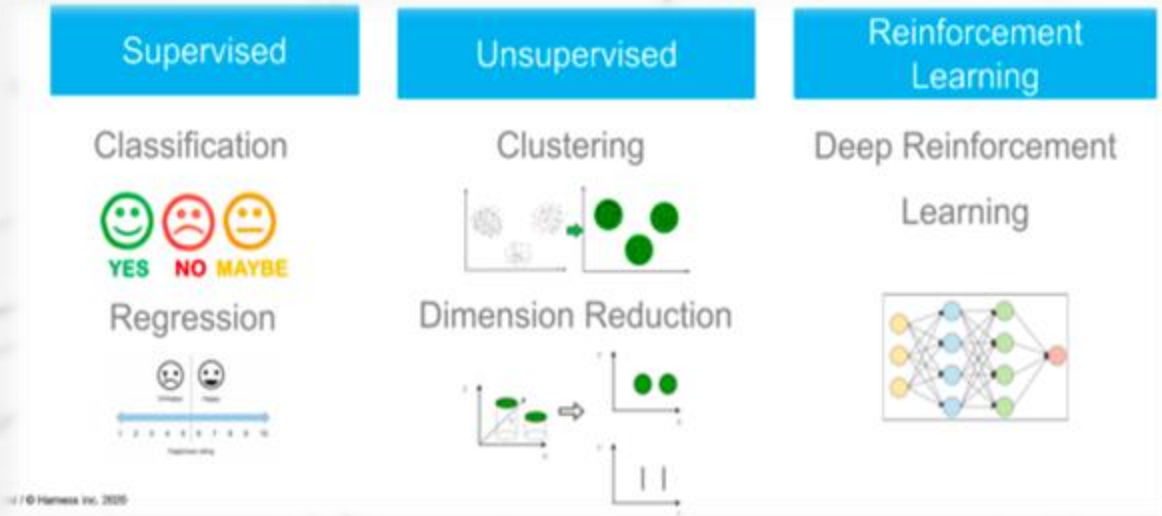


Type of ML


Machine Learning
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Classification
Regression
Clustering
Decision Making
▪ Naive Bayes Classifier
▪ Decision Trees
▪ Support Vector Machines
▪ Random Forest
▪ K – Nearest Neighbors
▪ Linear Regression
▪ Neural Network Regression
▪ Support Vector Regression
▪ Decision Tree Regression
▪ Lasso Regression
▪ Ridge Regression
▪ K-Means Clustering
▪ Mean-shift Clustering
▪ DBSCAN Clustering
▪ Agglomerative Hierarchical Clustering
▪ Gaussian Mixture
▪ Q-Learning
▪ R Learning
▪ TD Learning


Machine Learning
Unsupervised Learning
Supervised Learning
Reinforcement Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Population Growth Prediction
Market Forecasting
Estimating life expectancy
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

Type of ML

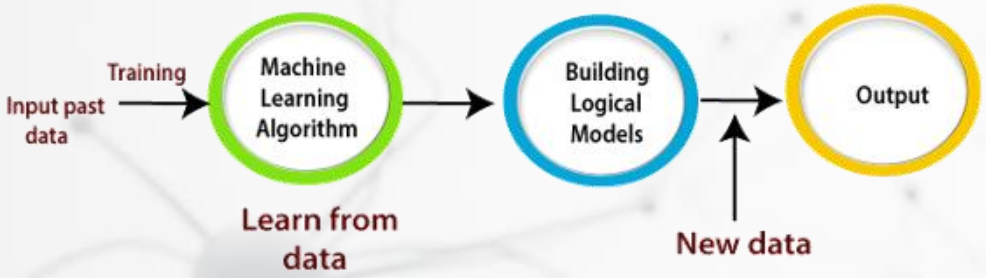


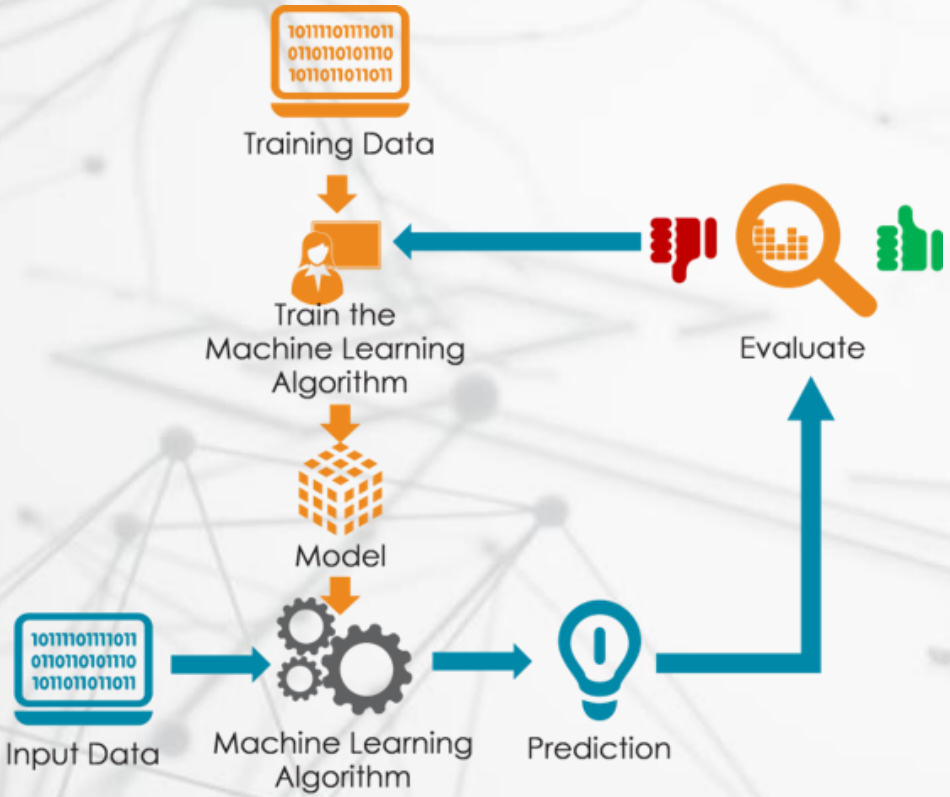
Which to Use When?

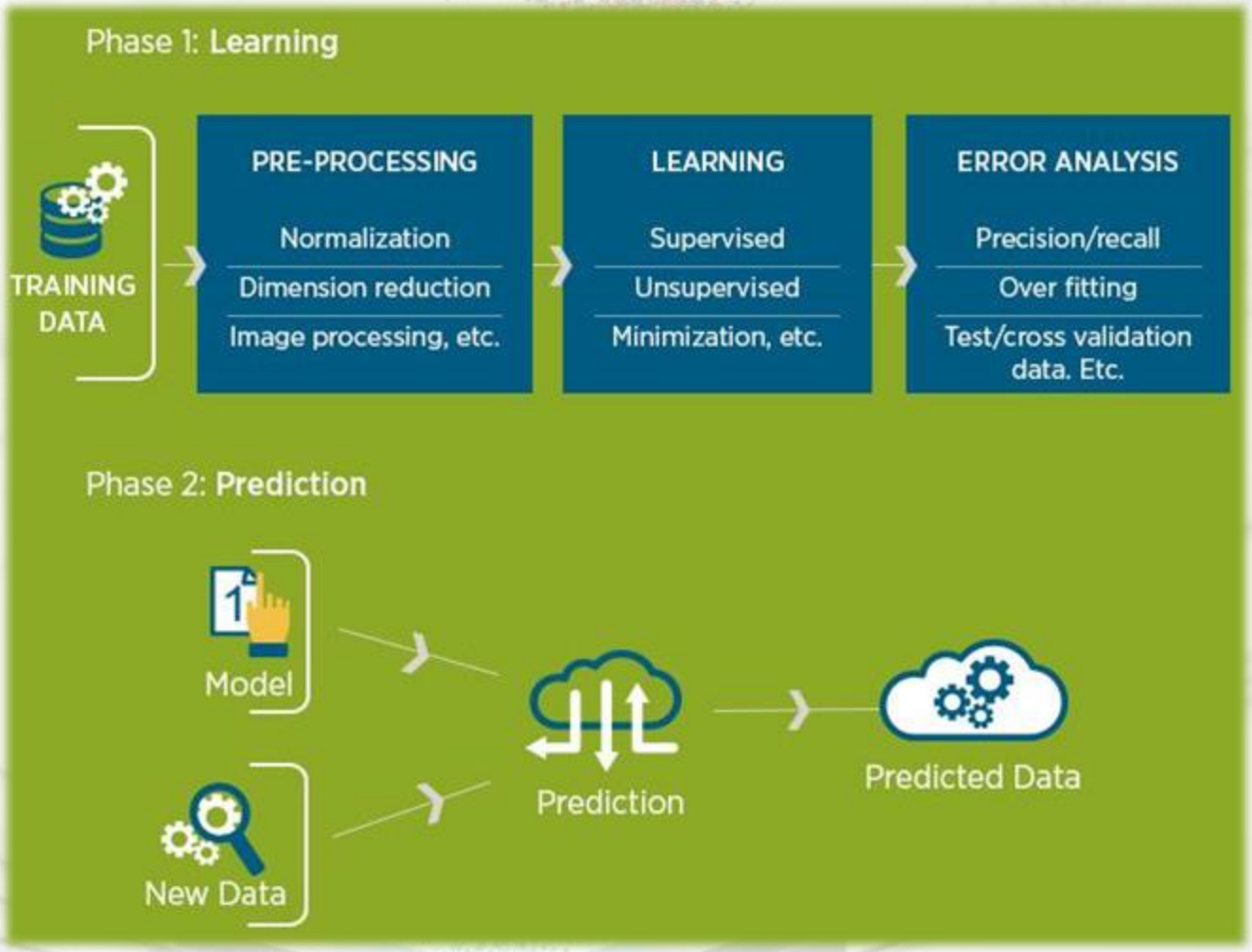
- **Supervised Learning:** When you have labeled data and want to make predictions or classifications.
- **Unsupervised Learning:** When you have unlabeled data and want to discover hidden patterns or group similar data points.
- **Reinforcement Learning:** When you want to train an agent to make decisions in an environment to maximize rewards.

Feature	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Data	Labeled data	Unlabeled data	No predefined data, learns from interactions with the environment
Goal	Predict outputs based on inputs	Discover patterns and structures	Learn a policy to maximize rewards
Supervision	Requires human supervision	No human supervision	No explicit supervision, learns through trial and error

ML process

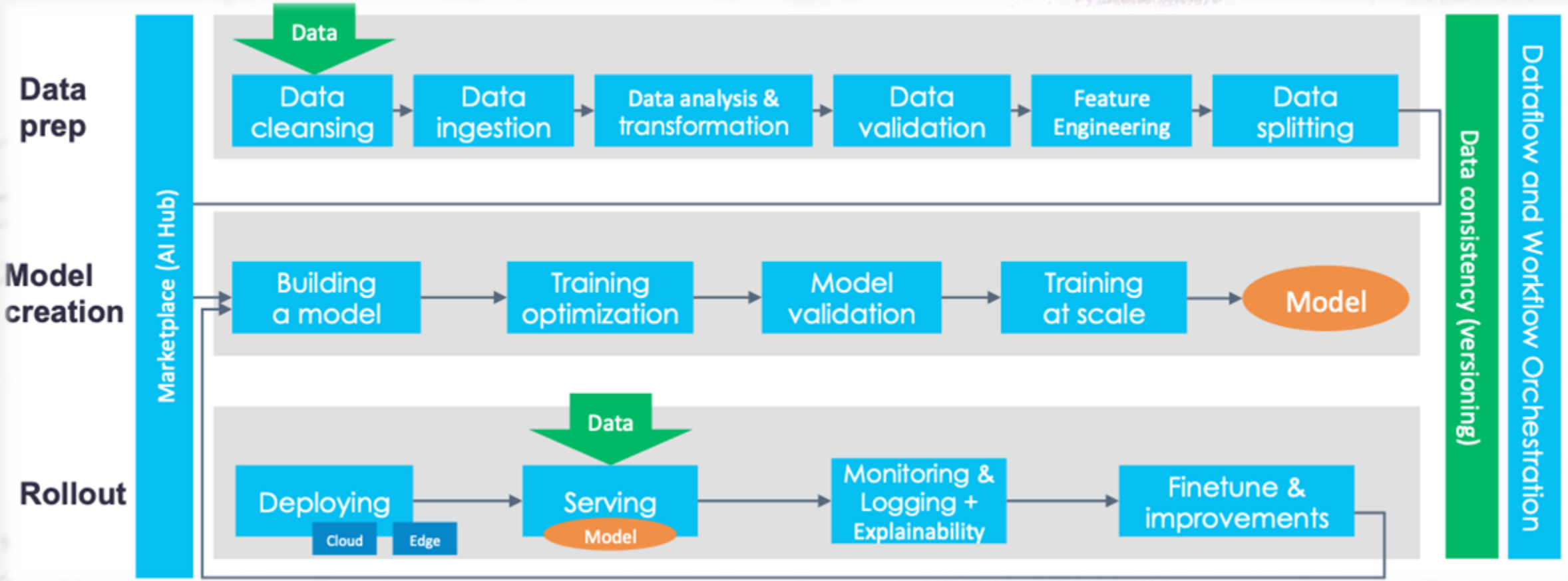

Training
Input past data
Machine Learning Algorithm
Building Logical Models
Output
Learn from data
New data


Training Data
Train the Machine Learning Algorithm
Evaluate
Model
Input Data
Machine Learning Algorithm
Prediction


Phase 1: Learning
TRAINING DATA
PRE-PROCESSING
Normalization
Dimension reduction
Image processing, etc.
LEARNING
Supervised
Unsupervised
Minimization, etc.
ERROR ANALYSIS
Precision/recall
Over fitting
Test/cross validation data. Etc.
Phase 2: Prediction
Model
New Data
Prediction
Predicted Data

Workflow of a Machine Learning Project

Workflow of a Machine Learning Project



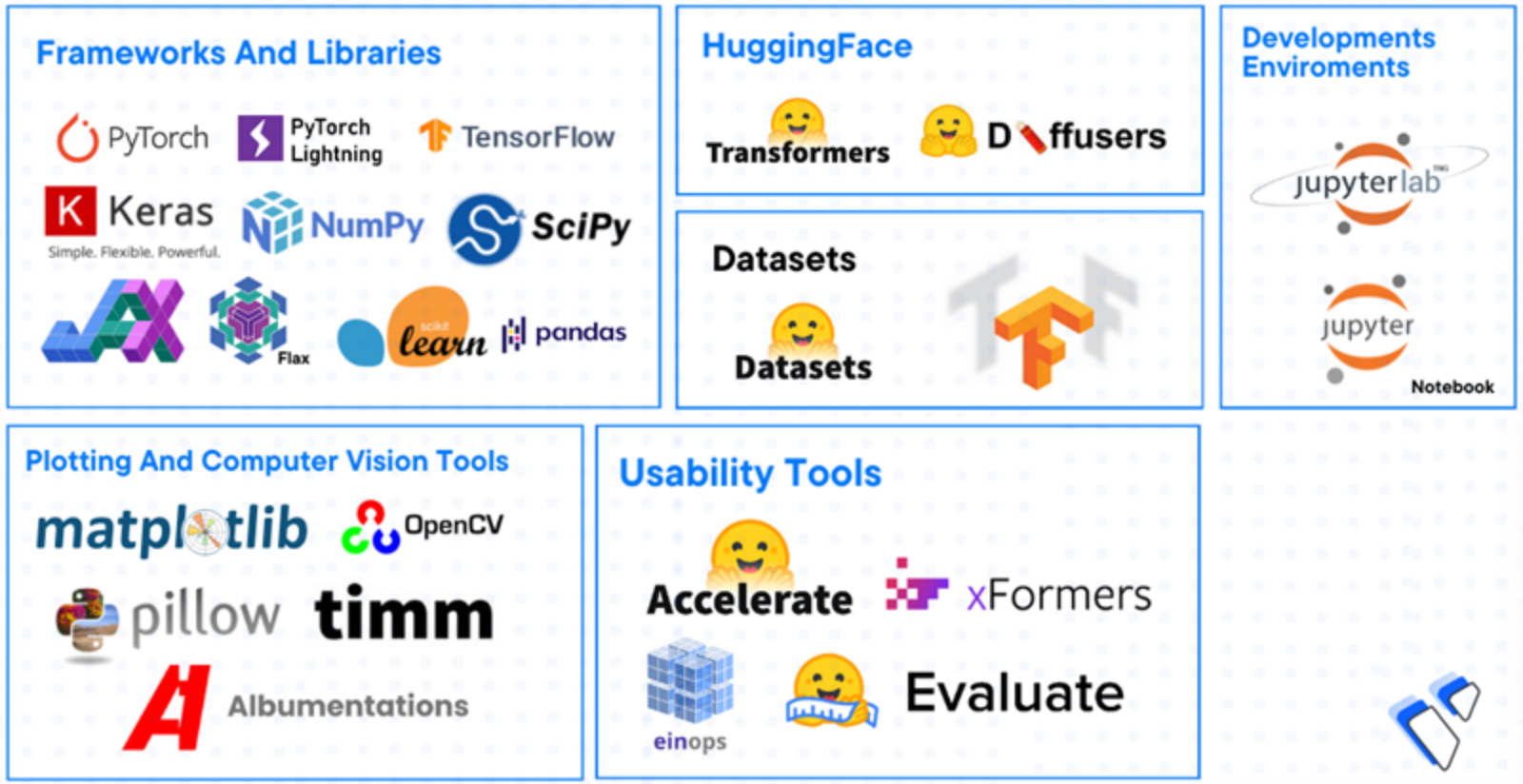
Basic Workflow of a Machine Learning Project

1. **Data Collection:** Collect relevant and high-quality data aligned with the problem you're solving.
2. **Data Preparation:** Clean, preprocess, and transform the raw data into a usable format (handle missing values, encoding, normalization, etc.).
3. **Feature Engineering:** Create or select key features that improve the model's ability to learn patterns.
4. **Model Selection:** Choose the most suitable algorithm based on your task (classification, regression, etc.) and data characteristics.
5. **Model Training:** Fit the selected model to your training dataset to learn underlying patterns.
6. **Model Evaluation:** Test model performance using appropriate metrics (e.g., accuracy, F1-score, MSE) on validation/test data.
7. **Model Deployment:** Deploy the trained model into a real-world application or system for live predictions.

Example: Predicting House Prices

1. **Data Collection:** Obtain a dataset with house prices and related attributes (e.g., area, location, bedrooms).
2. **Data Preparation:** Handle missing data, encode categorical variables, normalize numerical values.
3. **Feature Engineering:** Add derived features (e.g., price per square meter, age of the house).
4. **Model Selection:** Use regression models such as Linear Regression or Random Forest.
5. **Model Training:** Train on historical house data.
6. **Model Evaluation:** Use MSE or R^2 to evaluate prediction accuracy.
7. **Model Deployment:** Integrate into a web app or real estate tool for pricing estimation.

Important AI / ML Python Packages



Data Handling & Preprocessing
- NumPy
- Pandas
- Scikit-learn

Data Visualization
- Matplotlib
- Seaborn
- Plotly

Machine Learning
- Scikit-learn
- XGBoost
- LightGBM
- CatBoost

Deep Learning
- TensorFlow
- Keras
- PyTorch
- Transformers

Model Deployment & Utilities
- Joblib
- Flask
- FastAPI
- ONNX

Experiment Tracking & Tuning
- Optuna
- MLf/ow
- Weights & Biases

→ https://docs.vultr.com/deep-dive-into-important-ai-ml-python-packages#usability-tools

Key Libraries of Python used for ML

Category	Library	Primary Purpose	Key Features & Use Cases
Foundational / Data Ops	**NumPy**	Numerical computing with multi-dimensional arrays	Efficient array operations; mathematical functions; foundation for other ML libraries; data representation (vectors, matrices, tensors).
	Pandas	Data manipulation and analysis (tabular data)	DataFrames for structured data; data loading, cleaning, transformation; handling missing values; merging/joining datasets; essential for data preprocessing.
	Matplotlib	Static, animated, and interactive data visualization	Highly customizable plots (line, bar, scatter, etc.); essential for EDA and presenting results.
	Seaborn	High-level statistical data visualization (built on Matplotlib)	Simplifies complex statistical plots (heatmaps, violin plots, pair plots); enhances Matplotlib's aesthetics for statistical graphics.
	SciPy	Scientific and technical computing (advanced math functions)	Optimization, linear algebra, integration, signal processing; often used by other ML libraries under the hood or for specific pre-processing.
Core Machine Learning	**Scikit-learn**	Traditional Machine Learning algorithms and tools	Comprehensive range of supervised (classification, regression) and unsupervised (clustering, dimensionality reduction) algorithms; consistent API; includes tools for data preprocessing, model selection, and evaluation (e.g., cross-validation, metrics).
Deep Learning Frameworks	**TensorFlow**	End-to-end open-source platform for deep learning	High scalability; powerful for building and deploying large-scale neural networks (CNNs, RNNs, Transformers); strong for production environments; developed by Google.
	PyTorch	Open-source deep learning framework with dynamic computation graphs	Flexible and "Pythonic" API; favored in research for rapid prototyping and experimentation; easier debugging; strong community support; developed by Facebook AI Research.
	Keras	High-level Neural Networks API (runs on TensorFlow, PyTorch, JAX)	Simplifies deep learning model building and training; designed for fast experimentation; very beginner-friendly; provides a simpler interface to complex deep learning frameworks.
Specialized / Boosting	**XGBoost**	Optimized Gradient Boosting framework	High performance, speed, and accuracy on structured data; regularization to prevent overfitting; parallel processing; widely used in Kaggle competitions.
	LightGBM	Fast, distributed, high-performance Gradient Boosting framework	Focus on efficiency and scalability, especially for large datasets; uses histogram-based algorithms for faster computation and less memory; developed by Microsoft.
	CatBoost	Gradient Boosting library with native categorical feature handling	Automatically handles categorical features without extensive preprocessing; reduces overfitting through ordered boosting; developed by Yandex.
Specialized NLP	**Hugging Face Transformers**	Provides pre-trained models for Natural Language Processing (NLP)	Access to thousands of state-of-the-art Transformer models (BERT, GPT, T5); simplifies advanced NLP tasks like text classification, sentiment analysis, and generation.
Computer Vision	**OpenCV**	Library of programming functions mainly aimed at real-time computer vision	Image and video processing; feature detection; object detection; often integrated with deep learning models for complex vision tasks.

Python Implementation and practical example: Predicting House Prices

1. Import Libraries

- *pandas:* For data handling.
- *sklearn.model_selection.train_test_split*: Split data into training/testing sets.
- *sklearn.linear_model.LinearRegression*: The prediction model.
- *sklearn.metrics.mean_squared_error*: Evaluate model accuracy.

2. Load Dataset

- *data = pd.read_csv("housing_data.csv")*: Load house data from a CSV file.

3. Feature Engineering

- *data['rooms_per_person'] = ...*: Create a new feature from existing ones (e.g., rooms_per_person).

4. Split Data

- *X = ..., y = ...:* Define features (input) and target (output).
- *X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2):* Divide data into 80% for training and 20% for testing.

5. Train Model

- *model = LinearRegression():* Initialize the linear regression model.
- *model.fit(X_train, y_train)* : Train the model using the training data.

6. Make Predictions

- *y_pred = model.predict(X_test):* Use the trained model to predict values on the unseen test data.

7. Evaluate Model

- *mse = mean_squared_error(y_test, y_pred):* Calculate how well the model performed using Mean Squared Error.
- *print(...)* : Display the evaluation result.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Load the dataset
data = pd.read_csv("housing_data.csv")

# Feature engineering (e.g., creating new features)
data['rooms_per_person'] = data['total_rooms'] / data['total_bedrooms']

# Split into training and testing sets
X = data[['total_rooms', 'total_bedrooms', 'rooms_per_person']]
y = data['median_house_value']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Create and train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

References

[1] Tom Mitchell, Machine Learning. McGraw-Hill, 1997.

[2] R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification, 2nd edition, John Wiley & Sons, 2000.

[3] C. M. Bishop, “Pattern Recognition and Machine Learning”, Springer, 2006.