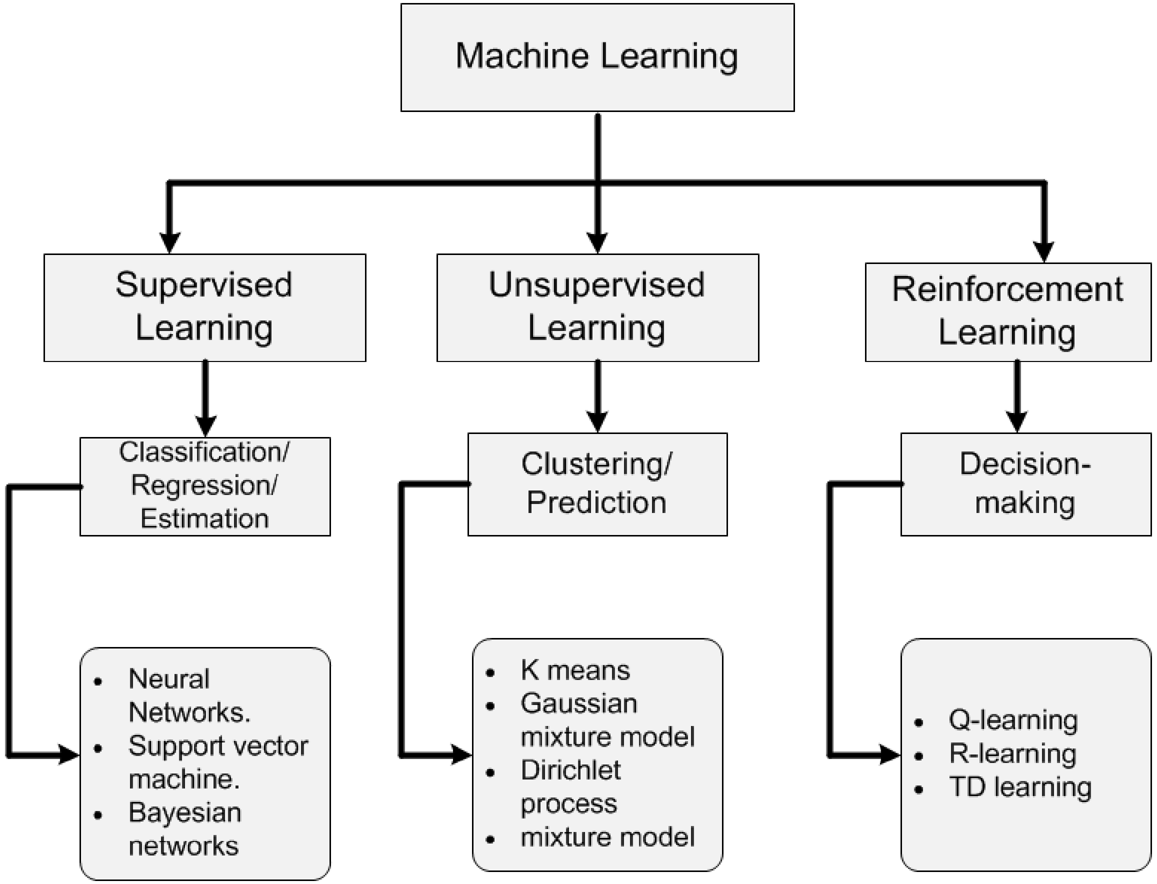


Week 5: Machine Learning Fundamentals

Recap of Previous Weeks

Week 1: Introduction to Jupyter Notebook, NumPy, and Pandas

Week 2: Working with Dataframes

Week 3: Data Manipulation and Preparation with Pandas

Week 4: Data Visualization with Matplotlib

Introduction to Machine Learning

What is machine learning?

"Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.“

How Machine Learning Works?

Learning from Data:

Algorithms learn patterns from historical data to make predictions on new, unseen data.

Iterative Process:

The model is trained on a dataset, evaluated, and refined through an iterative process.

Machine Learning Stages

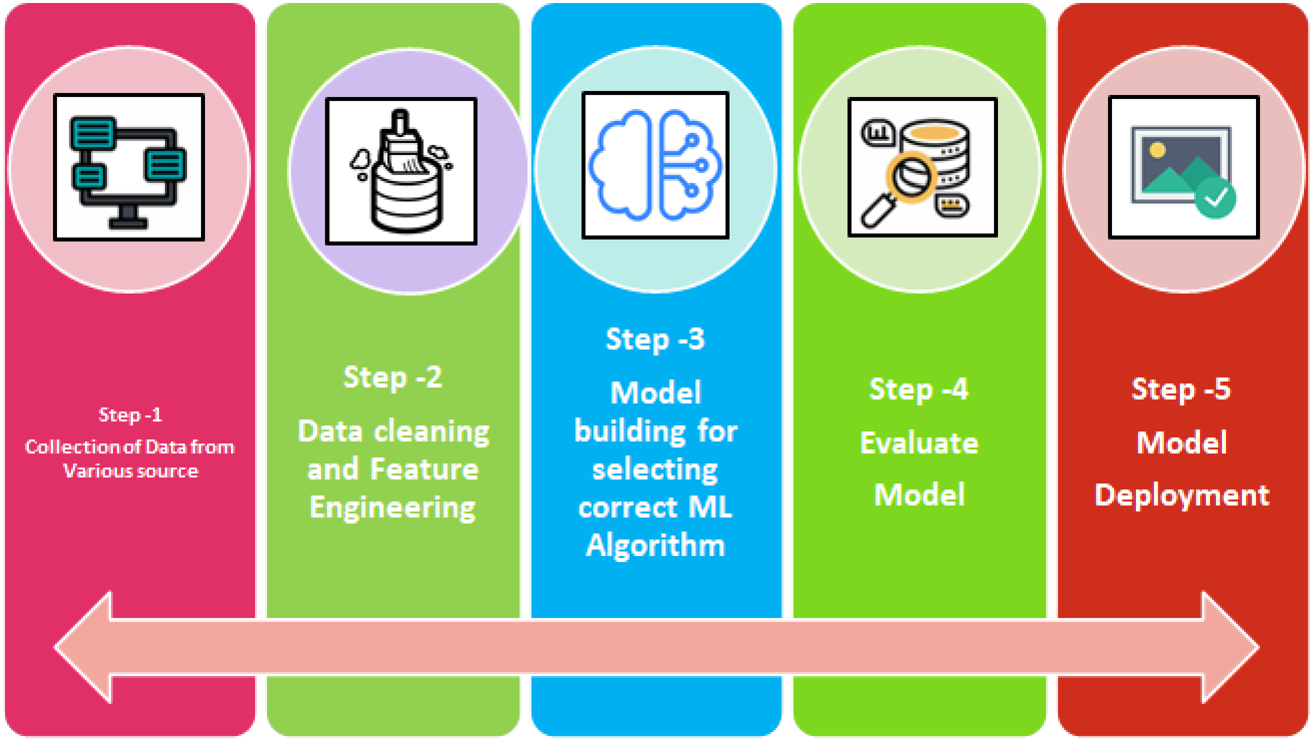
- We can split ML process stages into 5 as below mentioned in the flow diagram.

1) Collection of Data
2) Data Wrangling
3) Model Building
4) Model Evaluation
5) Model Deployment

So, we must be clear about the objective of the purpose of ML implementation.

Machine Learning Stages

➢ To find the solution for the given/identified problem we must collect the data and follow up the below stages appropriately.



Why Machine Learning?

Examples of real-world applications:

Recommendation Systems: Recommending movies, products, or songs based on user preferences.

Image Recognition: Identifying objects or people in images.

Predictive Modeling: Forecasting stock prices, weather conditions, or disease outbreaks.

Overview of common machine learning problems:

- Classification
- Regression
- Clustering

Types of Machine Learning

Supervised Learning: Learning from labeled data (input-output pairs).

Example: Predicting house prices based on features. Classifying emails as spam or not spam.

Unsupervised Learning: Finding patterns in unlabeled data.

Example: Grouping customers based on purchasing behavior without predefined categories.

Reinforcement Learning: Learning through trial and error.

Example: Training a computer to play and win games through repeated trials.

Types of Machine Learning Algorithms

Supervised Learning Algorithms:

Common algorithms: Linear Regression, Decision Trees, and Support Vector Machines.

Unsupervised Learning Algorithms:

Common algorithms: K-Means Clustering, and Principal Component Analysis (PCA).

Key Concepts in Machine Learning

Features:

Definition: These are the **input variables** (e.g., age, salary) or attributes used by the model to make predictions.

Example: In predicting house prices, features might include square meters, number of bedrooms, and location.

Labels:

Definition: **Output variable to predict** (e.g., spam/not spam). Also known as the target variable, this is what the model aims to predict.

Example: In the same house price prediction, the price is the label.

Key Concepts in Machine Learning

Training and Testing Data: Splitting data into two sets for model training and evaluation.

Training Data:

Definition: The portion of the dataset used to train the machine learning model.

Purpose: The model learns patterns and relationships from this data.

Example: 80% of the dataset used for training.

Testing Data:

Definition: The remaining portion of the dataset used to evaluate the model's performance.

Purpose: Assess how well the model generalizes to new, unseen data.

Example: 20% of the dataset used for testing.

Model Training and Prediction

Training the Model:

Process: The model is fed with the training data, learns patterns, and adjusts its parameters.

Objective: To make accurate predictions on new, unseen data.

Making Predictions:

Process: Once trained, the model can be used to predict outcomes for new data.

Example: After learning from housing data, the model predicts the price of a new house.

Evaluation Metrics

Accuracy:

Definition: The ratio of correctly predicted instances to the total instances.

Example: 90% accuracy means 9 out of 10 predictions are correct.

Precision:

Definition: The ratio of correctly predicted positive observations to the total predicted positives.

Example: Important in cases where false positives are costly.

Recall:

Definition: The ratio of correctly predicted positive observations to all actual positives.

Example: Important in cases where false negatives are costly.

Popular Machine Learning Libraries

- **Scikit-learn**:
  - Type: General-purpose machine learning library.
  - Use Case: Primarily used for classical machine learning tasks such as classification, regression, clustering, and dimensionality reduction.
- **TensorFlow**:
  - Type: Deep learning framework.
  - Use Case: Widely used for building and training deep neural networks for tasks like image recognition, natural language processing, and more.
- **Keras**:
  - Type: High-level neural networks API.
  - Use Case: Often used as a user-friendly interface for building neural networks on top of TensorFlow.

Introduction to Scikit-learn

What is Scikit-learn?

- A machine learning library for Python.
- Open-source and built on NumPy, SciPy, and Matplotlib.

Installation using pip :

```
pip install scikit-learn
```

Basic Import:

```
from sklearn import <algorithm>
```

Importing Specific Functions:

```
from sklearn.<algorithm> import <specific_function>
```

Why Scikit-learn?

1) User-friendly and efficient for small to medium-sized datasets.
2) Provides a wide range of machine learning algorithms.
   - Classification: SVM, decision trees, k-neighbors, etc.
   - Regression: Linear regression, Lasso, Ridge, etc.
   - Clustering: K-means, hierarchical, DBSCAN, etc.
3) Data Preprocessing: Standardization, normalization, handling missing values.
4) Model Evaluation Tools: Metrics, cross-validation, hyperparameter tuning.
5) Well-documented with examples.

Tips for Getting Started

Start Simple:

Begin with basic models and gradually explore more complex ones.

Understand Parameters:

Read documentation to understand algorithm parameters and their impact.

Explore Datasets:

Use built-in datasets for practice and experimentation.

Utilize Visualization:

Leverage Matplotlib and other visualization libraries to understand model behavior.

Community Support:

Scikit-learn has an active community; utilize forums for problem-solving.

Hands-on with Scikit-learn

- **Loading a Dataset**
  - **Built-in Datasets**: Scikit-learn provides several datasets for practice and experimentation. Example: Iris dataset, digits dataset.
  - **External Datasets**: Use Pandas or other libraries to load datasets.

- **Data Preprocessing**

Pandas, NumPy for data cleaning.

- **Choosing a Model**

Common algorithms: Decision Trees, Support Vector Machines, etc.

Loading Built-in Datasets

Using Built-in Datasets:

- Scikit-learn provides datasets for practice and experimentation.

Example: Iris dataset.

```python
from sklearn.datasets import load_iris

# Load the Iris dataset
iris = load_iris()

# Features and labels
X = iris.data
y = iris.target
```

Loading External Datasets

Using external dataset:

```
import pandas as pd
from sklearn.model_selection import train_test_split

# Load dataset using Pandas
data = pd.read_csv('your_dataset.csv')

# Split into features and labels
X = data.drop('target_column', axis=1)
y = data['target_column']

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Data Preprocessing: Standardization

Standardization:

Concept:

Standardization is a process of rescaling the features (or variables) in your dataset so that they have a mean of 0 and a standard deviation of 1.

Steps:

1. Calculate Mean and Standard Deviation:
   - Find the mean (μ) and standard deviation (σ) of each feature in your dataset.
2. Subtract Mean and Divide by Standard Deviation:
   - For each data point in a feature, subtract the mean and then divide by the standard deviation.

Data Preprocessing: Standardization

Formula:

$$\text{Standardized Value} = \frac{\text{Original Value} - \text{Mean}}{\text{Standard Deviation}}$$

Example:

If you have a set of exam scores, standardization would make the average score 0 and adjust other scores based on how many standard deviations away they are from the mean.

- Standardize features by removing the mean and scaling to unit variance.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

# Fit on training data and transform both training and testing data
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Data Preprocessing: Standardization

Normalization:

Concept:

Normalization is a process of scaling and transforming the values of your features so that they fall within a specific range, often between 0 and 1.

Steps:

1. Find Min and Max:
   - Determine the minimum (min) and maximum (max) values for each feature.
2. Rescale Values:
   - Rescale each data point in a feature within the range of 0 to 1 based on the minimum and maximum.

Data Preprocessing: Normalization

Formula:

$$\text{Normalized Value} = \frac{\text{Original Value} - \text{Min}}{\text{Max} - \text{Min}}$$

Example:

If you have a dataset of house prices, normalization would transform the prices so that the lowest price becomes 0, the highest becomes 1, and others fall in between proportionally.

- Scale features to a range between 0 and 1.

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

# Fit on training data and transform both training and testing data
X_train_normalized = scaler.fit_transform(X_train)
X_test_normalized = scaler.transform(X_test)
```

Key Differences

Standardization:

- Centers the data around 0 by adjusting the mean.
- Adjusts the scale of the features based on their standard deviation.
- Suitable for algorithms that assume a normal distribution of the features.

Normalization:

- Scales the data to a specific range (usually 0 to 1).
- Maintains the relative relationships between values.
- Useful for algorithms that rely on distances between data points.

Choosing Between Them

- Use standardization when your data follows a normal distribution.

- Use normalization when the distribution of your data is not known or when you want features to be on a similar scale.

Both standardization and normalization are common preprocessing techniques that help machine learning models perform better by ensuring that features are in a consistent and comparable format.

Model Selection

- Model selection involves choosing the type of machine learning model that best suits your problem. Different types of models are designed for different types of tasks.
- Based on the type of problem (classification, regression) and characteristics of the data.

```
from sklearn.tree import DecisionTreeClassifier

# Create a decision tree classifier
model = DecisionTreeClassifier()
```

Common Algorithms:

Decision Trees, Support Vector Machines, k-Nearest Neighbors, etc.

Model Selection Steps

1) Understand the Problem:
   - Determine whether your problem is a classification, regression, or clustering task.
2) Explore Model Types:
   - For classification, consider models like Decision Trees, Support Vector Machines, or Random Forests.
   - For regression, consider models like Linear Regression, Lasso, or Ridge Regression.
   - For clustering, consider models like K-Means or hierarchical clustering.
3) Consider Complexity:
   - Choose a model that balances complexity with the size and nature of your dataset. Simple models may be more interpretable but might not capture complex patterns.
4) Evaluate Options:
   - Try different models and evaluate their performance on your specific problem.

Model Training

- Model training is the process of teaching the selected machine learning model to recognize patterns and make predictions based on the input data.

Steps:

1) **Split the Data**:
   - Divide your dataset into two parts: a **training set** and a **testing set**.
   - The training set is used to train the model, while the testing set is used to evaluate its performance.
2) **Feed Data to the Model**:
   - Present the training set to the model along with the corresponding labels (correct answers).

Model Training

3) **Adjust Model Parameters**:
   - The model adjusts its internal parameters to learn patterns in the data.
   - This process is known as "**fitting**" the model to the training data.

4) **Repeat Until Convergence**:
   - The training process repeats until the model converges, meaning it has learned the patterns in the training data to a satisfactory level.

Model Training

Training a Model:

```
from sklearn.<algorithm> import <Model>

# Create an instance of the model
model = <Model>()

# Train the model
model.fit(X_train, y_train)
```

Making Predictions:

```
# Make predictions
predictions = model.predict(X_test)
```

Evaluating the Model:

Metrics: Accuracy, precision, recall.

Model Evaluation

➢ Model evaluation assesses how well the trained model performs on new, unseen data.

Steps:

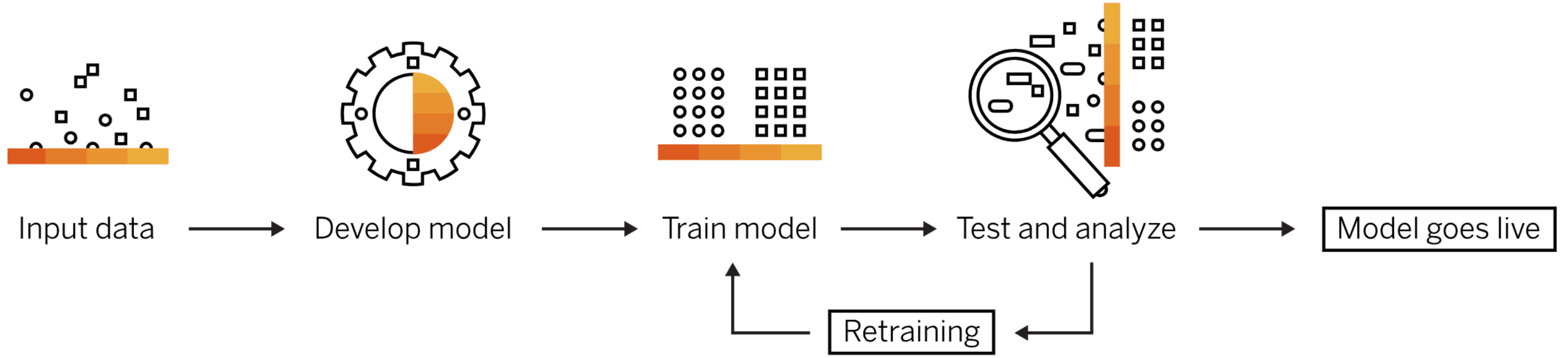
1) **Use the Testing Set**:
   - Present the testing set to the trained model and observe its predictions.

2) **Evaluate Performance**:
   - Use metrics such as accuracy, precision, recall, or F1 score to measure how well the model is performing.
   - Compare the model's predictions to the actual labels in the testing set.

Model Evaluation

3) **Adjust if Necessary:**
   - If the model is not performing well, consider adjusting its parameters, trying a different model, or exploring feature engineering.

4) **Avoid Overfitting:**
   - Ensure that the model generalizes well to new data and does not memorize the training set (overfitting).



Model Persistence

- In Python, the **joblib** library is commonly used to save and load machine learning models.

Saving a Model:

- Save a trained model for future use.
- The **joblib.dump()** function is used to save the trained model (model) to a file named 'knn_model.joblib'. (You can replace 'knn_model.joblib' with any desired filename.)

```
import joblib

# Save the model
joblib.dump(model, 'your_model.pkl')
```

Model Persistence

Loading a Model:

- To load a saved model, use the joblib.load() function and provide the filename ('knn_model.joblib' in this case).
- The loaded_model object now contains the model you saved earlier, and you can use it to make predictions on new data.

```
# Load the model
loaded_model = joblib.load('your_model.pkl')

# Make predictions
new_predictions = loaded_model.predict(new_data)
```

Scikit-learn Resources

- **Documentation**: Scikit-learn Documentation

- **Tutorials**: Scikit-learn Tutorials

Exercises

- Iris Dataset - Classification

Dataset: Download the 'Iris.csv' dataset from the Datset folder in Microsoft Teams.

Objective: Build a classification model to predict the species of iris flowers based on their features.

Steps:

- Load the Iris dataset.
- Split the dataset into features (X) and labels (y).
- Split the data into training and testing sets.
- Choose a classification algorithm (e.g., Decision Trees, k-Nearest Neighbors).
- Train the model on the training set.
- Evaluate the model on the testing set.

Exercises

Breast Cancer Diagnosis - Binary Classification

Dataset: Download the 'breast_cancer.csv’ dataset from the Datset folder in Microsoft Teams.

Objective: Build a binary classification model to predict whether a breast tumor is malignant (cancerous) or benign (non-cancerous) based on various features.

Steps:

- Load the Breast Cancer dataset.
- Explore the dataset to understand its structure.
- Split the dataset into features (X) and labels (y).
- Split the data into training and testing sets.
- Choose a classification algorithm (e.g., Support Vector Machines, Logistic Regression).
- Train the model on the training set.
- Evaluate the model on the testing set.