Md Rifat Ahmmad Rashid, Associate Professor, Dept. of CSE
East West University, Bangladesh

Lecture 12: Machine Learning

Overview of Machine Learning

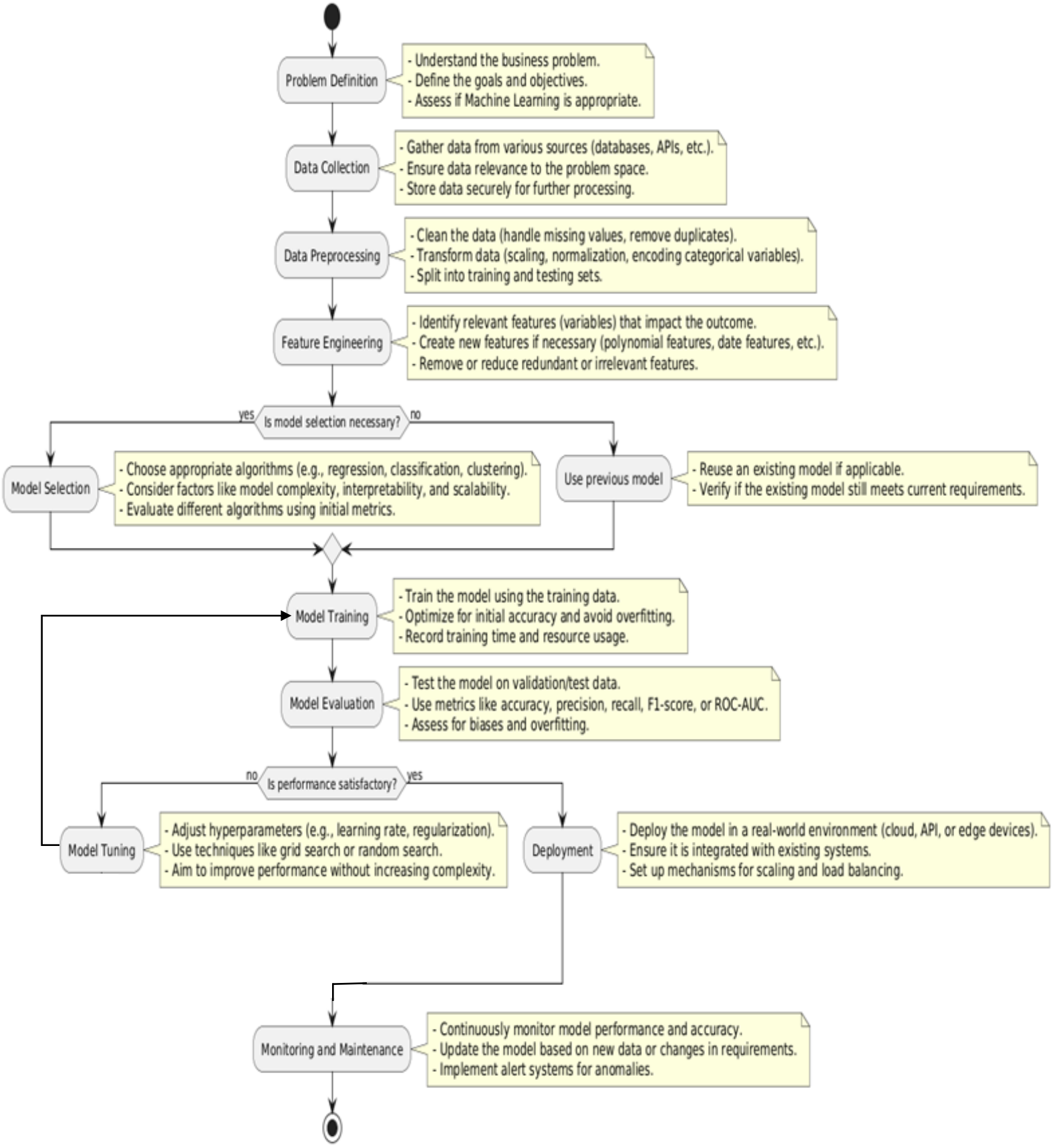
Machine learning (ML) is a subset of artificial intelligence (AI) that enables systems to learn from data, identify patterns, and make decisions with minimal human intervention. It is grounded in the idea that systems can automatically improve their performance on a task with experience, typically through training on a large amount of data. Machine learning algorithms can be broadly categorized into three types: supervised learning, unsupervised learning, and reinforcement learning.

- **Supervised Learning:** The algorithm learns from labeled data, meaning the input data is associated with known outcomes or labels.
- **Unsupervised Learning:** The algorithm learns from unlabeled data and tries to uncover hidden patterns without any explicit output labels.
- **Reinforcement Learning:** The algorithm learns through trial and error by interacting with an environment and receiving feedback through rewards or penalties.

Machine Learning Workflow

The machine learning workflow involves several key stages:

1. **Problem Definition:** Understanding the problem at hand and deciding whether machine Learning is the right approach.
2. **Data Collection:** Gathering relevant data that represents the problem space.
3. **Data Preprocessing:** Cleaning, transforming, and normalizing data to ensure it is suitable for the machine learning model.
4. **Feature Engineering:** Selecting or extracting relevant features (variables) that will be used by the model.
5. **Model Selection:** Choosing the appropriate machine learning algorithm(s) for the task.
6. **Model Training:** Feeding the model with training data and allowing it to learn patterns.
7. **Model Evaluation:** Assessing the model's performance using test data and appropriate metrics.
8. **Model Tuning:** Optimizing model parameters (hyperparameters) to improve performance.
9. **Deployment:** Implementing the model in a real-world environment to make predictions.
10. **Monitoring and Maintenance:** Continuously monitoring model performance and updating it as needed based on new data.


Problem Definition
- Understand the business problem.
- Define the goals and objectives.
- Assess if Machine Learning is appropriate.
Data Collection
- Gather data from various sources (databases, APIs, etc.).
- Ensure data relevance to the problem space.
- Store data securely for further processing.
Data Preprocessing
- Clean the data (handle missing values, remove duplicates).
- Transform data (scaling, normalization, encoding categorical variables).
- Split into training and testing sets.
Feature Engineering
- Identify relevant features (variables) that impact the outcome.
- Create new features if necessary (polynomial features, date features, etc.).
- Remove or reduce redundant or irrelevant features.
yes
Is model selection necessary?
no
Model Selection
- Choose appropriate algorithms (e.g., regression, classification, clustering).
- Consider factors like model complexity, interpretability, and scalability.
- Evaluate different algorithms using initial metrics.
Use previous model
- Reuse an existing model if applicable.
- Verify if the existing model still meets current requirements.
Model Training
- Train the model using the training data.
- Optimize for initial accuracy and avoid overfitting.
- Record training time and resource usage.
Model Evaluation
- Test the model on validation/test data.
- Use metrics like accuracy, precision, recall, F1-score, or ROC-AUC.
- Assess for biases and overfitting.
no
Is performance satisfactory?
yes
Model Tuning
- Adjust hyperparameters (e.g., learning rate, regularization).
- Use techniques like grid search or random search.
- Aim to improve performance without increasing complexity.
Deployment
- Deploy the model in a real-world environment (cloud, API, or edge devices).
- Ensure it is integrated with existing systems.
- Set up mechanisms for scaling and load balancing.
Monitoring and Maintenance
- Continuously monitor model performance and accuracy.
- Update the model based on new data or changes in requirements.
- Implement alert systems for anomalies.

Fine-Tuning a Model

Fine-tuning is a process of improving model performance by adjusting model parameters, architecture, or training processes. Here's how students can fine-tune their models:

1. **Adjust Hyperparameters**:
   - **Learning Rate:** Decrease the learning rate if the model's loss is fluctuating wildly. Increase it slightly if the model is learning too slowly.
   - **Batch Size:** A larger batch size can stabilize gradient updates but requires more memory. A smaller batch size can lead to noisier updates but often helps in generalization.

2. **Regularization:**
   - **L2 Regularization (Ridge):** Helps to reduce overfitting by adding a penalty to large weights.
   - **Dropout (for deep learning):** Randomly "drops" neurons during training to prevent the network from becoming too reliant on any one node, thus reducing overfitting.

3. **Early Stopping:**

   - Stop training when the validation loss stops improving. This can prevent overfitting, as the model will stop when it starts performing poorly on the validation set, even if the training loss is decreasing.

4. **Optimizer Adjustment:**
   - For deep learning models, using optimizers like Adam or RMSprop often improves performance over traditional SGD (Stochastic Gradient Descent). Try different optimizers or adjust the learning rate schedule (e.g., step decay, exponential decay).

Bias-Variance Tradeoff

The bias-variance tradeoff is a fundamental concept in machine learning that describes the balance between two sources of errors:

Bias: Refers to the error introduced by approximating the real-world problem too simply. High bias occurs when the model is too simple and underfits the data.

Example: A linear model trying to capture a non-linear relationship.

Variance: Refers to the error introduced by the model's sensitivity to small fluctuations in the training data. High variance occurs when the model is too complex and overfits the data.

Example: A deep neural network with many layers memorizing training data.

Goal: The goal is to find the "sweet spot" where the model complexity is just right, minimizing both bias and variance.

- **Underfitting**: High bias and low variance — model too simple.
- **Overfitting**: Low bias and high variance — model too complex.

Strategies to Manage Bias-Variance Tradeoff:

For High Bias (Underfitting):
- Use a more complex model (e.g., from linear regression to polynomial regression).
- Increase the number of features or add more layers to neural networks.

For High Variance (Overfitting):
- Use regularization techniques (L1, L2 regularization).
- Increase the size of the training dataset.
- Reduce the model complexity (e.g., reduce the depth of a decision tree).

Epochs and Iterations

In deep learning, understanding the relationship between epochs and iterations is critical for setting up proper training.

Epoch: One epoch means that the model has seen the entire training data once.
Iteration: One iteration means that the model has updated its weights once after processing a batch of data.

Choosing the Ideal Number of Epochs:
- The number of epochs depends on the dataset's size and complexity.
- Too few epochs may lead to underfitting, while too many may lead to overfitting.
- Use early stopping to automatically halt training when the validation error stops improving.

Typical Rule of Thumb:
- For small datasets: 50–100 epochs.
- For large datasets (e.g., ImageNet): Several hundred epochs may be necessary.

Monitoring Training:
- Track training and validation loss across epochs.
- Stop training when the validation loss starts to increase, indicating overfitting.

Data Splitting and Cross-Validation

Data Splitting: Overview and Ideal Ratios

Data splitting involves partitioning the dataset into different subsets to train, validate, and test the model. The goal is to evaluate the model's performance reliably while maximizing the use of available data.

Ideal Data Splits for ML Problems

1. **Training Set**:
    - The largest portion (typically 60-80%) of the data is used for training the model.
    - The model learns patterns, adjusts weights, and minimizes error on this subset.
2. **Validation Set**:
    - 10-20% of the data is reserved for validation.
    - It is used to fine-tune the model's hyperparameters, check for overfitting, and evaluate the model's performance during training.
3. **Test Set**:
    - The remaining 10-20% of the data is set aside as a test set, which is used to assess the final performance of the model.
    - This set is kept entirely separate from the training and validation processes to provide an unbiased evaluation of the model's accuracy.

Data Splits for Specific Types of Data

- **Image Data**:
    - When dealing with image datasets, it's common to split data into training (70-80%), validation (10-15%), and test (10-15%) sets. For smaller image datasets, additional techniques like data augmentation are often applied to increase the diversity of the training set.
    - For object detection and segmentation tasks, maintaining a balanced distribution of classes and object scales across the splits is crucial.
- **NLP Data**:
    - NLP datasets are often split in the same ratios as typical machine learning problems (e.g., 70% train, 15% validation, 15% test). However, special attention is needed to maintain a balanced representation of different classes, languages, or document types (e.g., sentiment classes or entity types) across all splits.
    - Additionally, for sequence data, ensuring that similar sequences are not overrepresented in the training set compared to validation and test sets is essential.

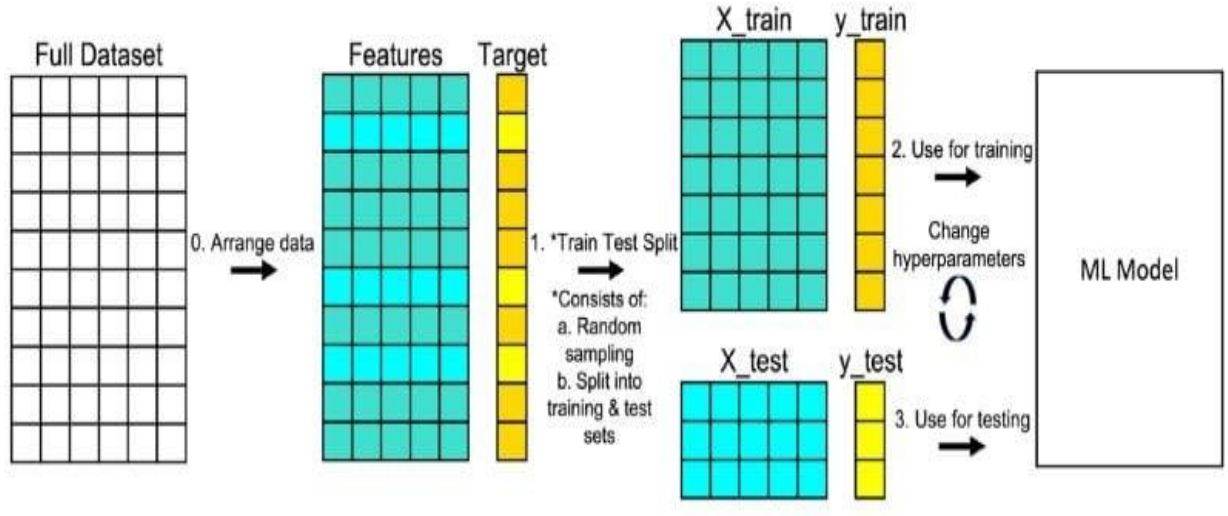


Image source: https://www.linkedin.com/pulse/data-splitting-strategies-machine-learning-swapnil-sharma/

Cross-Validation

Cross-validation is a technique used to assess how well a model will generalize to an independent dataset. By training and validating a model multiple times with different data partitions, cross-validation provides a more accurate and reliable measure of model performance.

K-Fold Cross-Validation:

1. **Concept**: The dataset is divided into K subsets (or folds). The model is trained on K-1 folds and validated on the remaining fold. This process is repeated K times, with each fold serving as the validation set once. The performance metric is averaged across all K iterations.
    - **When to Use**:
        - Ideal for small to medium-sized datasets where it's crucial to maximize the use of all available data.
        - Suitable for most types of machine learning problems (e.g., tabular data, image data, and NLP).

- **Benefit**: Reduces bias and variance by ensuring that every data point is used for both training and validation. It provides a more reliable measure of model performance, particularly for small datasets where every observation is valuable.

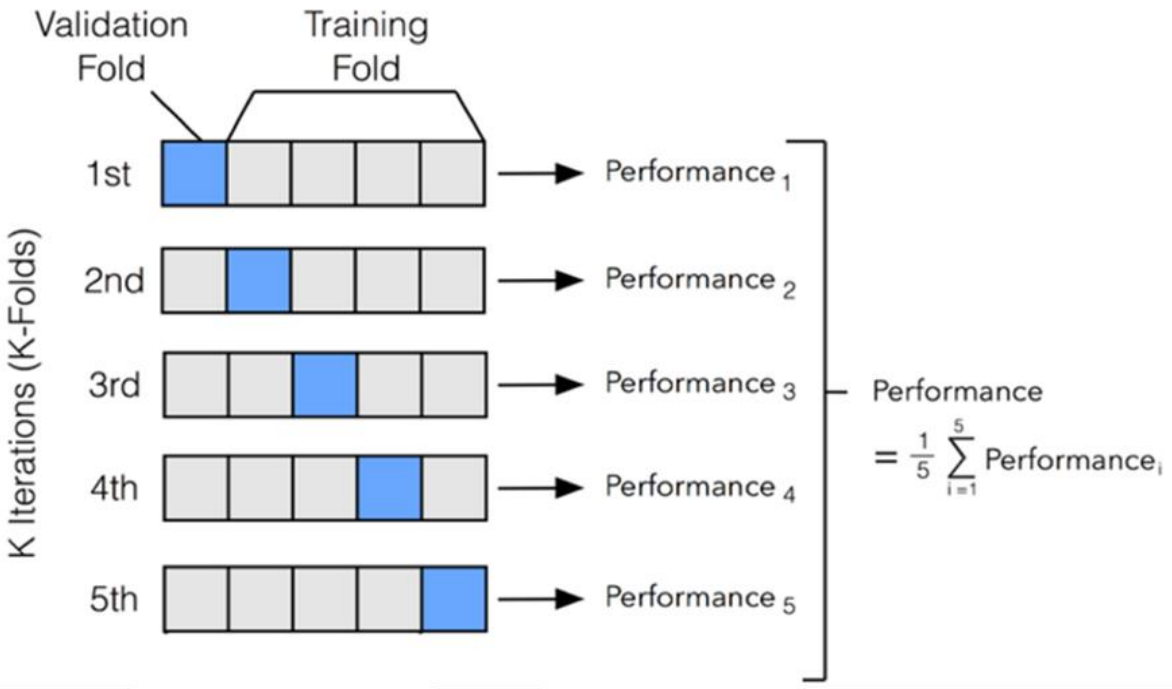


Image Source: https://medium.com/@ompramod9921/cross-validation-623620ff84c2

When to Use Cross-Validation

- **Small to Medium-Sized Datasets**: When you have limited data, cross-validation maximizes the use of your dataset by ensuring every point is used for both training and validation.
- **Model Selection and Hyperparameter Tuning**: Cross-validation helps compare different models or hyperparameter settings reliably, as it provides consistent performance estimates across multiple data splits.
- **Algorithm Evaluation**: For models like decision trees, logistic regression, or SVMs, cross-validation is a suitable method to validate their performance, especially when the dataset is not large.

When Not to Use Cross Validation

- **Large Datasets**: When working with large datasets (e.g., millions of observations), a simple train-validation-test split is often sufficient, as the dataset is already large enough to provide reliable estimates.
- **Computationally Expensive Models**: For deep learning models or other complex algorithms that require significant training time, cross-validation may not be practical due

to the need to train multiple models. In such cases, using a separate validation set or employing alternative validation strategies like early stopping is more efficient.

Monitoring and Evaluation

Once the model is trained, it's critical to evaluate its performance:

- **Use the test set:** Ensure that the test data has never been used during training or validation.

Plot learning curves: Monitor both the training and validation error throughout the

---

1. Evaluation Metrics for Regression

Regression tasks involve predicting continuous numerical values. Evaluation metrics for regression measure the difference between the predicted and actual values.

1. **Mean Absolute Error (MAE)**:
    - **Definition**: The average of the absolute differences between predicted and actual values.
    - **Formula**:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

    - **Interpretation**: Lower values indicate better performance.
2. **Mean Squared Error (MSE)**:
    - **Definition**: The average of the squared differences between predicted and actual values.
    - **Formula**:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

    - **Interpretation**: Penalizes larger errors more heavily than MAE. Lower values are better.
3. **Root Mean Squared Error (RMSE)**:
    - **Definition**: The square root of the MSE. It represents the standard deviation of the residuals (prediction errors).

- **Formula**:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- **Interpretation**: Lower RMSE indicates a better fit, making it more interpretable in the original unit of the target variable.

4. **R-Squared (Coefficient of Determination)**:
   - **Definition**: Measures the proportion of the variance in the dependent variable that is predictable from the independent variables.

   Formula: $R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$, where SS_{res} is the sum of squares of residuals, and SS_{tot} is the total sum of squares.

   - **Interpretation**: Values range from 0 to 1, with higher values indicating better model performance.

5. **Adjusted R-Squared**:
   - **Definition**: A modified version of R-Squared that accounts for the number of predictors in the model. It penalizes for adding irrelevant variables.

   Formula: $\text{Adjusted } R^2 = 1 - \frac{(1-R^2)(n-1)}{n-p-1}$, where n is the number of observations, and p is the number of predictors.

   - **Interpretation**: Useful when comparing models with different numbers of predictors.

6. **Mean Absolute Percentage Error (MAPE)**:
   - **Definition**: The average of the absolute percentage errors between predicted and actual values.
   - **Formula**:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

   - **Interpretation**: Expressed as a percentage; lower values indicate better performance.

7. **Explained Variance Score**:
   - **Definition**: Measures the proportion of variance explained by the model, similar to R-squared but can be negative.

- **Range**: [-∞, 1], where 1 indicates perfect prediction and lower values indicate worse performance.

2. Evaluation Metrics for Classification

Classification tasks involve predicting categorical outcomes. Metrics for classification are designed to evaluate the accuracy and robustness of such predictions.

1. **Accuracy**:
   - **Definition**: The ratio of correctly predicted instances to the total number of instances.
   - **Formula**:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

   - **Interpretation**: Suitable when the dataset is balanced, but less informative for imbalanced datasets.
2. **Precision**:
   - **Definition**: The ratio of true positive predictions to the total positive predictions.
   - **Formula**:

$$\text{Precision} = \frac{TP}{TP+FP}$$

   - **Interpretation**: Indicates how many positive predictions are actually correct; useful in cases where false positives are costly.
3. **Recall (Sensitivity)**:
   - **Definition**: The ratio of true positive predictions to the total actual positives.
   - **Formula**:

$$\text{Recall} = \frac{TP}{TP+FN}$$

   - **Interpretation**: Indicates how many actual positives the model captures; crucial in scenarios where false negatives are costly.
4. **F1 Score**:
   - **Definition**: The harmonic mean of precision and recall, providing a balance between the two.
   - **Formula**:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Interpretation**: Useful for imbalanced datasets where accuracy alone is not sufficient.

5. **Area Under the ROC Curve (AUC-ROC)**:
   - **Definition**: Measures the model's ability to distinguish between classes by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR).
   - **Range**: [0, 1], where 1 indicates perfect separation and 0.5 indicates no separation.
   - **Interpretation**: Higher values indicate better model performance.

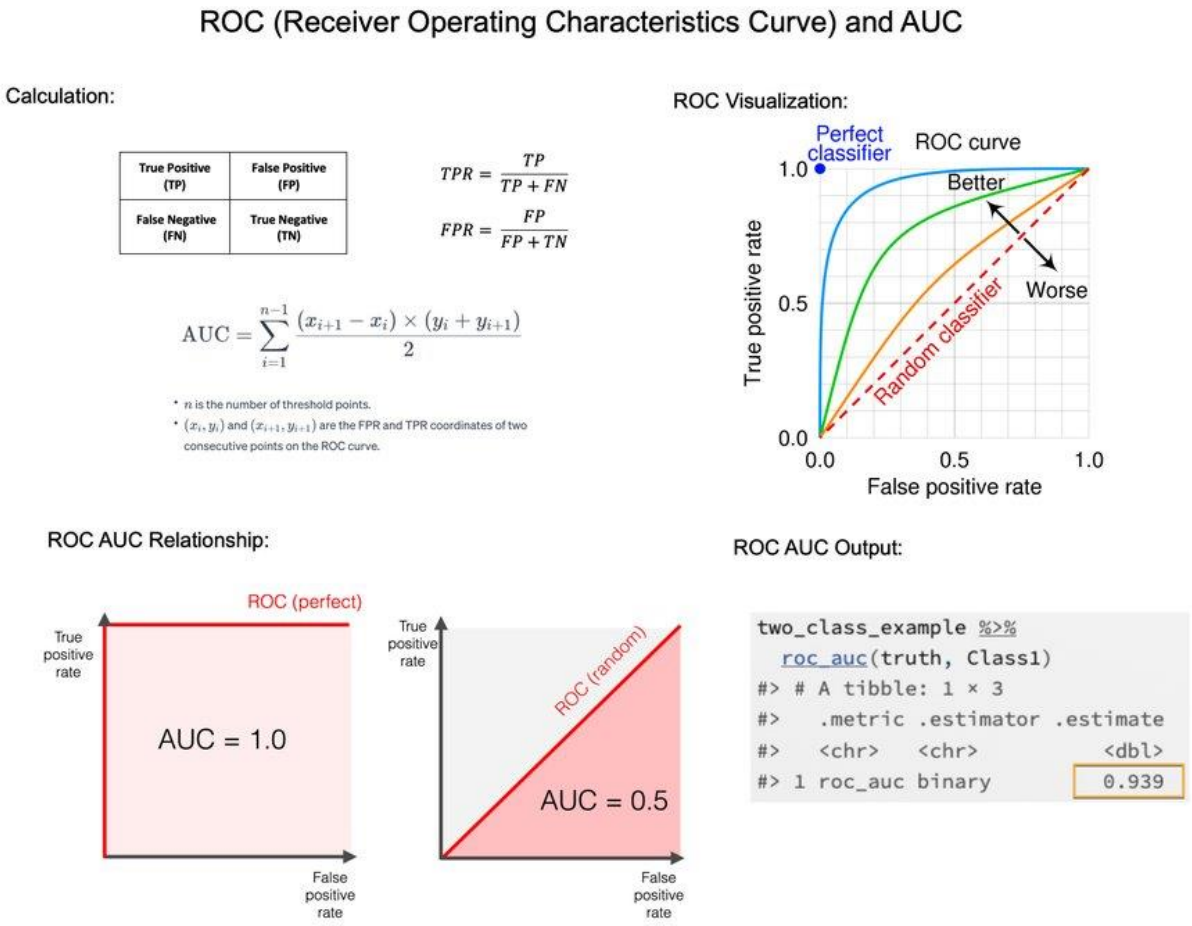


Image source: https://x.com/mdancho84/status/1752019456664973641

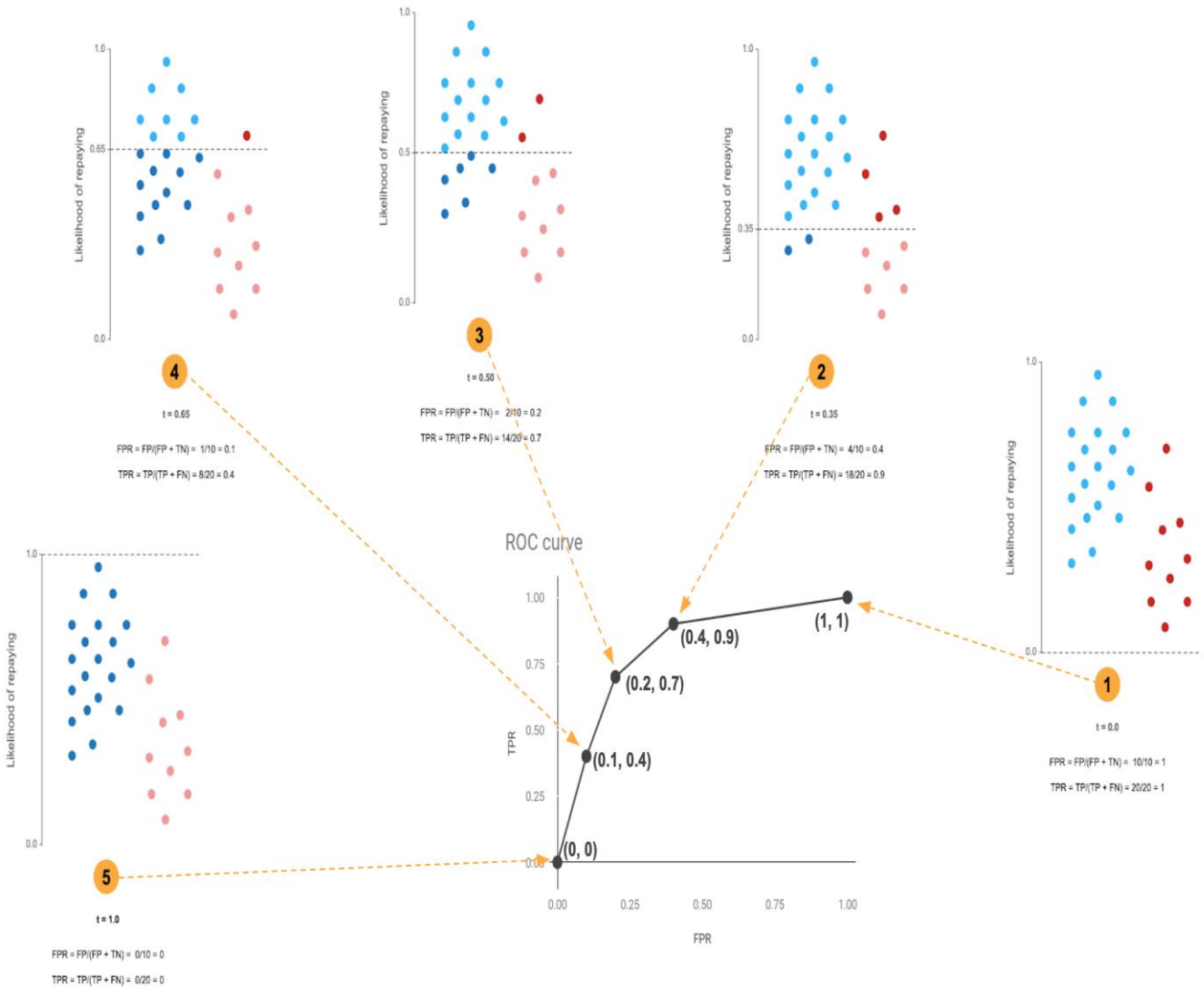


Image source: https://towardsdatascience.com/understanding-the-roc-curve-in-three-visual-steps-795b1399481c

6. **Area Under the Precision-Recall Curve (AUC-PR)**:
    - **Definition**: A plot that shows the trade-off between precision and recall for different thresholds.
    - **Interpretation**: Especially informative for imbalanced datasets, where the AUC-ROC may not provide enough insight.
7. **Logarithmic Loss (Log Loss)**:
    - **Definition**: Measures the uncertainty of the predictions. It penalizes incorrect predictions with higher penalties for confident but incorrect predictions.
    - **Formula**:

$$\text{Log Loss} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

- **Interpretation**: Lower values indicate better performance.

What is Supervised Learning?

In supervised learning, the dataset consists of inputs (features) and their corresponding outputs (labels or target values). The model is trained using this labeled dataset, and the objective is to learn a mapping function from the input to the output. The learning process involves minimizing the error between the model's predictions and the actual outputs.

- **Input (Features)**: Variables or attributes of the data that are used as input for the model (e.g., age, height, income).
- **Output (Label/Target)**: The variable that the model aims to predict (e.g., house price, classification category).

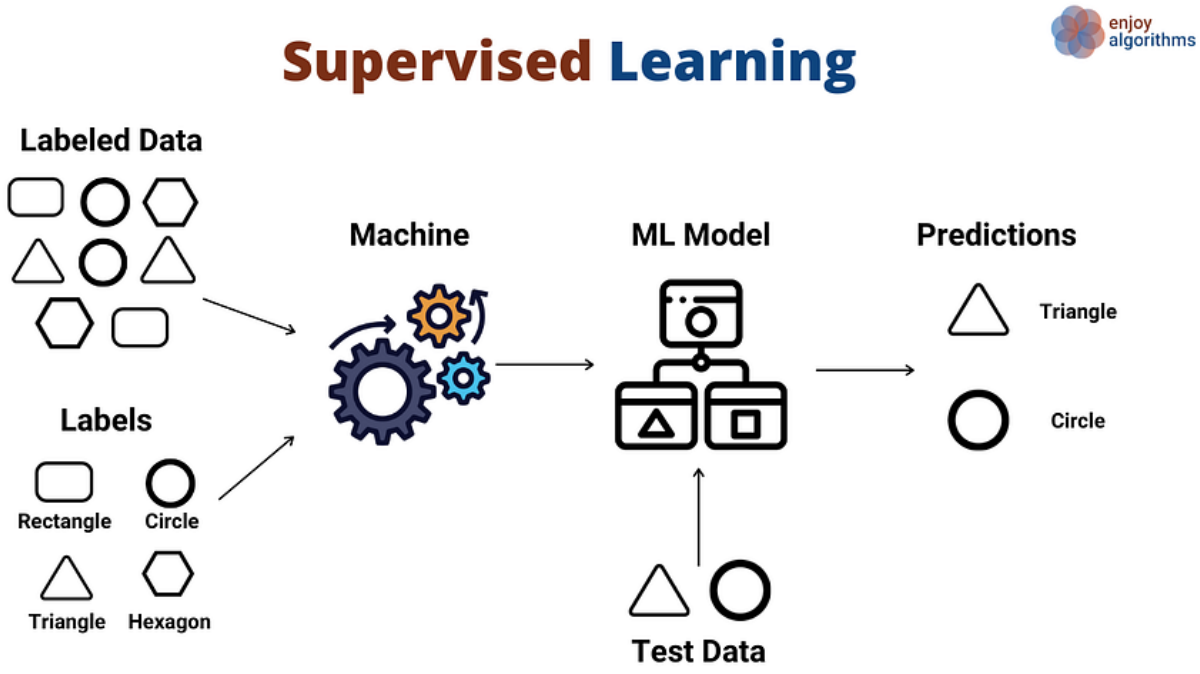


Image source: https://medium.com/@metehankozan/supervised-and-unsupervised-learning-an-intuitive-approach-cd8f8f64b644

Types of Supervised Learning Problems

Supervised learning problems can be broadly categorized into two types: regression and classification.

1. **Regression**:
    - **Definition**: Regression involves predicting continuous numerical values. The output variable is quantitative and can take any real value.
    - **Examples**:
        - Predicting house prices based on features like square footage, location, and number of bedrooms.
        - Forecasting temperature changes over time based on historical weather data.
    - **Common Algorithms**:
        - Linear Regression
        - Decision Trees for Regression
        - Random Forest Regressor
        - Support Vector Regressor (SVR)
        - Neural Networks (for continuous outputs)

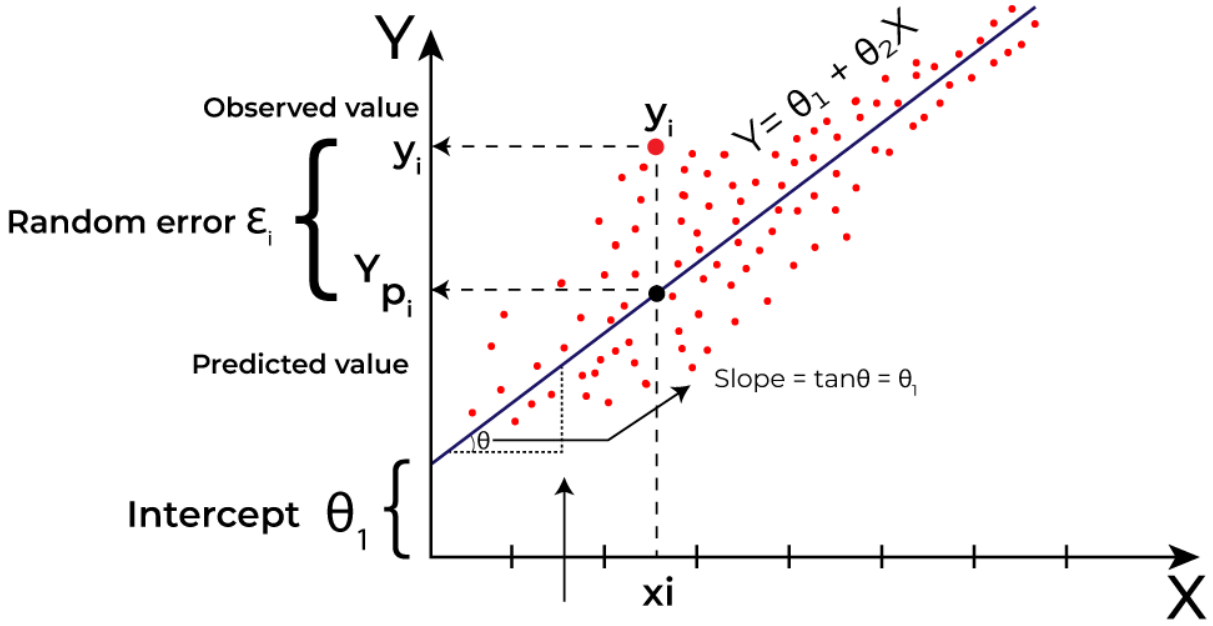


Image source: https://www.geeksforgeeks.org/ml-linear-regression/

2. **Classification**:
    - **Definition**: Classification involves predicting discrete class labels or categories. The output variable is categorical and takes one of a limited set of values (e.g., binary or multi-class).
    - **Examples**:
        - Classifying email as spam or not spam.
        - Predicting whether a tumor is benign or malignant based on medical images.
    - **Common Algorithms**:
        - Logistic Regression
        - Decision Trees
        - Random Forest Classifier
        - Support Vector Machine (SVM)
        - k-Nearest Neighbors (k-NN)
        - Naive Bayes
        - Neural Networks (for categorical outputs)

The Supervised Learning Process

The process of supervised learning generally involves the following steps:

1. **Data Collection**:
    - Gather a dataset that includes both inputs (features) and corresponding outputs (labels). The dataset must be labeled accurately for the model to learn effectively.
2. **Data Preprocessing**:
    - **Cleaning**: Remove or handle missing values, outliers, and inconsistencies.
    - **Normalization/Standardization**: Scale features to a uniform range (e.g., 0 to 1) or a standard distribution (mean = 0, variance = 1).
    - **Encoding**: Convert categorical features into numerical format using techniques like one-hot encoding or label encoding.
    - **Splitting Data**: Divide the dataset into training, validation, and test sets to evaluate model performance accurately.
3. **Model Selection**:

- Choose a suitable algorithm based on the type of problem (regression or classification), the size of the dataset, and the complexity of the relationships within the data.

4. **Model Training**:
   - The model is trained on the training set using a chosen algorithm. The model iteratively updates its parameters to minimize the error (loss function) between the predicted and actual outputs.
5. **Model Evaluation**:
   - Evaluate the model's performance using the validation set. Performance metrics vary based on the type of task:
     - **For regression**: Mean Absolute Error (MAE), Mean Squared Error (MSE), R-squared.
     - **For classification**: Accuracy, Precision, Recall, F1 score, Area Under the ROC Curve (AUC-ROC).
6. **Hyperparameter Tuning**:
   - Adjust the model's hyperparameters (e.g., learning rate, depth of a decision tree) to optimize performance. This is often done using techniques like grid search or random search with cross-validation.
7. **Model Testing**:
   - The final model is tested on the test set, which is unseen during training and validation, to assess its generalization ability.

Key Concepts in Supervised Learning

1. **Training and Test Sets**:
   - **Training Set**: Used to train the model. It represents the majority of the data.
   - **Test Set**: Used to evaluate the final model's performance. It provides an unbiased estimate of how the model will perform on new data.
2. **Loss Function**:
   - A function that measures the error between the predicted output and the actual output. The objective of training is to minimize this error.
   - **Common Loss Functions**:
     - **For regression**: Mean Squared Error (MSE), Mean Absolute Error (MAE).
     - **For classification**: Cross-entropy loss, Hinge loss.
3. **Optimization Algorithms**:

- Methods used to minimize the loss function and update the model's parameters. These algorithms determine how the model learns from the data.
- **Examples**:
  - Gradient Descent
  - Stochastic Gradient Descent (SGD)
  - Adam Optimizer (Adaptive Moment Estimation)

4. **Bias-Variance Tradeoff**:
   - **Bias**: The error introduced by approximating a real-world problem with a simplified model. High bias leads to underfitting.
   - **Variance**: The model's sensitivity to small fluctuations in the training data. High variance leads to overfitting.
   - The goal is to find a balance between bias and variance to achieve optimal model performance.
5. **Cross-Validation**:
   - A technique used to evaluate model performance and reduce overfitting by splitting the dataset into multiple folds. K-fold cross-validation is a common method where the dataset is divided into K subsets, and the model is trained and validated K times.
   - **When to Use**: Suitable for small to medium-sized datasets to maximize data utilization.

Common Algorithms in Supervised Learning

1. **Linear Regression**:
   - **Use Case**: Predicting continuous outcomes based on linear relationships between features.
   - **Example**: Estimating house prices.
2. **Logistic Regression**:
   - **Use Case**: Binary classification tasks, such as spam detection.
   - **Example**: Predicting whether an email is spam or not.
3. **Decision Trees**:
   - **Use Case**: Both regression and classification tasks.
   - **Example**: Classifying customers based on their likelihood of purchasing a product.
4. **Support Vector Machines (SVM)**:

- **Use Case**: Classification tasks, particularly with high-dimensional data.
- **Example**: Text classification in NLP.

5. **k-Nearest Neighbors (k-NN)**:
   - **Use Case**: Simple and intuitive classification and regression tasks based on similarity measures.
   - **Example**: Recommending products based on similar user preferences.
6. **Naive Bayes**:
   - **Use Case**: Classification tasks, especially in text classification where independence assumptions can hold.
   - **Example**: Sentiment analysis in social media posts.
7. **Ensemble Methods**:
   - **Random Forest**: Combines multiple decision trees for improved accuracy and stability.
   - **Gradient Boosting (e.g., XGBoost)**: Sequentially trains models to correct the errors of previous ones.
   - **Use Case**: Effective in both regression and classification, particularly for complex datasets.

Advantages and Disadvantages of Supervised Learning

Advantages:

1. **Accuracy**: Supervised learning models can achieve high accuracy if there is sufficient labeled data and the model is well-tuned.
2. **Interpretability**: Some models (e.g., linear regression, decision trees) are easy to interpret and understand.
3. **Versatility**: It can be applied to a wide range of problems, including regression, binary classification, and multi-class classification.

Disadvantages:

1. **Dependence on Labeled Data**: Supervised learning requires a significant amount of labeled data, which can be expensive and time-consuming to obtain.
2. **Overfitting**: Models may perform well on the training data but poorly on new data if they overfit. Regularization techniques and cross-validation are used to mitigate this risk.
3. **Scalability**: For large datasets or complex tasks (e.g., deep learning models), training can be computationally intensive and time-consuming.