ZENITH STUDY HUB

Artificial Intelligence for Robotics

UNIT IV | MACHINE LEARNING

Supervised and unsupervised learning, regression, classification, K-Means, SVM, neural networks and model evaluation

Syllabus-first explanations, clear figures and numericals

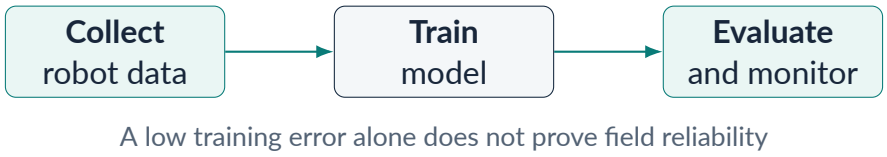

Figure 1. Original robotics machine-learning lifecycle. Assessment on separate data is an essential part of learning.

Exact official Unit IV syllabus trace
Introduction to Machine Learning, supervised and unsupervised learning; classification and regression concepts; K-Means clustering (concept), Support Vector Machines and introductory Neural Networks; model evaluation, overfitting; object classification, fault detection and predictive maintenance applications.

Course: PCC301-ROA | SPPU TE Robotics & Automation | 2024 Pattern

Prepared for: Om Deshmukh | 21 November 2026 | Final-review candidate

Learning map and a responsible workflow

Syllabus topic	Learning task
Basics and terminology	State features, labels, training and inference.
Supervised learning	Distinguish labelled classification from regression.
Unsupervised learning	Explain K-Means assignments and centroid updates.
Support Vector Machine	Draw a margin and classify with a linear decision function.
Neural networks	Calculate a single artificial neuron's weighted sum and activation.
Evaluation / overfitting	Compute confusion metrics, regression errors and understand leakage.
Robot applications	Design quality inspection, fault detection and maintenance pipelines.

In simple language
Machine learning (ML) finds statistical patterns in examples. A traditional hand-written rule might say "if vibration > 4 then alarm"; an ML model instead learns how vibration, temperature and other features are associated with observed machine conditions. It still requires careful engineering.

Read this before studying the graphs. All numerical datasets and drawn points in this PDF are original teaching examples, not production machine logs. Models fitted to these tiny datasets are demonstrations of calculations, not approved robotic controllers or reliable diagnostic systems.

1 Introduction to Machine Learning

1.1 AI, ML and robotics

Artificial intelligence is the broad area of machines performing tasks that require intelligent behaviour, such as planning and learning. **Machine learning** is a collection of methods that learn useful patterns from data. **Robotics** involves sensing, actuation, motion and interaction with the physical world. A robot may use ML for recognizing parts but still rely on conventional feedback control and protective hardware.

1.2 Five key words every student should know

1. **Example or sample:** one observation or record, such as a robot-camera image of a component.
2. **Feature:** an input variable, such as edge strength, motor current, vibration RMS or temperature.
3. **Label or target:** a reference answer, such as “defect” or measured remaining bearing life.
4. **Model:** a function $\hat{y} = f_\theta(x)$ with fitted parameters θ.
5. **Training versus inference:** training adjusts parameters from data; inference applies the fitted model to new inputs.

1.3 A real-world robotics learning workflow

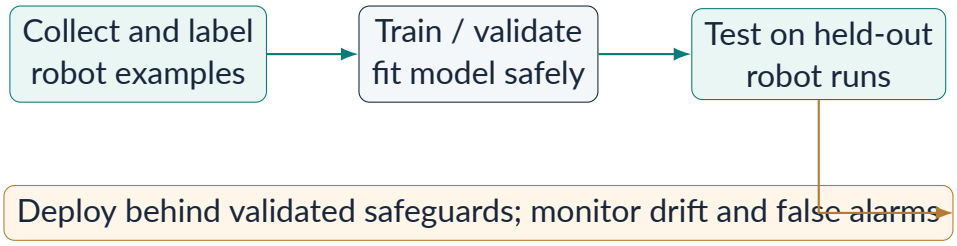

Figure 2. Original data-to-deployment process: held-out testing and live monitoring are separate from training.

1.4 Supervised versus unsupervised learning

In **supervised learning**, training data contain input–target pairs (x_i, y_i). The model learns a relationship that predicts labels or numerical targets for new inputs. In **unsupervised learning**, the data have no target labels; algorithms find structure such as groups or lower-dimensional representations. Unsupervised clusters are not automatically verified fault categories.

	Supervised learning	Unsupervised learning
Input	Features plus target labels	Features without target labels
Output	Predicted class or number	Clusters or representations
Example	Good/defective classification	Group motor vibration patterns
Limitation	Labels can be wrong or biased	Groups may not match true machine faults

2 Classification: Predict a Class

In simple language
Classification returns a category such as “pass/fail”, “bolt present/missing”, or “machine normal/faulty”. It does not directly tell you how much longer the machine will operate.

2.1 Binary versus multi-class classification

Binary: class 0 or 1, for example normal versus anomaly. Multi-class: one of several known classes such as gear, bearing or shaft. Multi-label: more than one label may apply to the same inspection image. Class labels are defined by the project and must be checked for accuracy.

2.2 A confusion matrix you can calculate

Suppose a robot quality checker evaluates 30 parts. It flags 8 truly defective parts and misses 2 defective parts. It also falsely flags 3 good parts and correctly accepts 17 good parts.

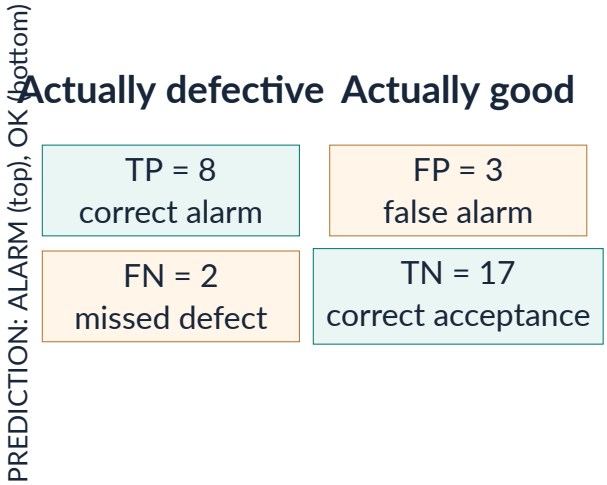

Figure 3. Original confusion matrix with counts $(TP, FP, FN, TN) = (8, 3, 2, 17)$. The positive class is "defective".

2.3 Evaluation metrics from that matrix

$$\begin{aligned}
\text{Accuracy} &= \frac{TP + TN}{TP + TN + FP + FN} = \frac{25}{30} = .8333, \\
\text{Precision} &= \frac{TP}{TP + FP} = \frac{8}{11} = .7273, \\
\text{Recall} &= \frac{TP}{TP + FN} = \frac{8}{10} = .8000, \\
\text{F1 score} &= \frac{2TP}{2TP + FP + FN} = \frac{16}{21} = .7619.
\end{aligned}$$

Accuracy is the overall fraction correct. Precision answers "of the alarmed parts, how many were truly defective?" Recall answers "of the truly defective parts, how many were found?" For safety-critical or rare faults, reporting accuracy alone can hide missed defects.

Missing denominator. If the system predicts no positives, precision $TP/(TP + FP)$ is undefined. Evaluation software must specify how to handle that case; do not silently divide by zero.

2.4 Class imbalance and error costs

If only 1 in 1000 parts is defective, a model that always reports "good" achieves 99.9% accuracy but detects no defects. Such a model is useless as a defect

detector. Include recall, false-negative rate, precision, class-specific costs and realistic operating conditions when choosing the model.

3 Regression: Predict a Number

In simple language
Regression predicts a continuous quantity such as motor temperature, tool-wear amount or predicted vibration amplitude. Its errors are measured in the target's numerical units.

3.1 Linear regression concept

A linear model is $\hat{y} = wx + b$, where w is a fitted slope and b is an intercept. With several sensor features, $\hat{y} = w^T x + b$. Training chooses parameters to reduce a loss on labelled examples; using a linear formula does not guarantee the physics itself is linear.

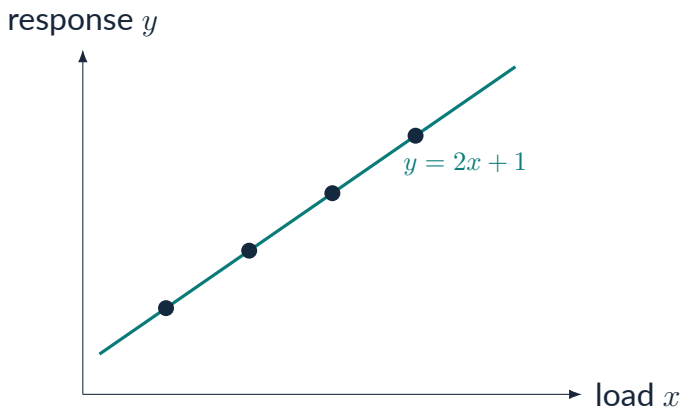

Figure 4. Original linear regression illustration using idealised points on $y = 2x + 1$; not a calibrated mechanical response curve.

3.2 MAE, MSE and RMSE

For true values y_i and predictions $\hat{y}_i$,

$$\text{MAE} = \frac{1}{n}\sum_i |y_i - \hat{y}_i|, \quad \text{MSE} = \frac{1}{n}\sum_i (y_i - \hat{y}_i)^2, \quad \text{RMSE} = \sqrt{\text{MSE}}.$$

MAE and RMSE have target units, while MSE has squared target units. MSE/RMSE penalize large errors more heavily than MAE.

WORKED EXAMPLE: Three robot temperature predictions

Suppose reference values are $(5, 7, 9)$ degrees Celsius above baseline and predictions are $(4, 7, 10)$. Errors are $(-1, 0, 1)$. MAE $= (1 + 0 + 1)/3 = 2/3 \approx .6667$ degree Celsius; MSE $= (1 + 0 + 1)/3 = 2/3 \approx .6667$ (degrees Celsius)^2; RMSE $= \sqrt{2/3} \approx .8165$ degrees Celsius. Numerical equality of MAE and MSE here is coincidental because every nonzero absolute error equals one.

3.3 Regression versus classification in one sentence

Prediction "motor temperature will be 86°C" is regression. Prediction "motor is overheating" is classification. The latter might be derived from a threshold on a temperature estimate, but the evaluated target and risk of error are different.

4 K-Means Clustering

In simple language
K-Means divides feature points into K groups by repeatedly assigning each point to its nearest current cluster centre and recalculating centres. It does not know the true machine-fault class without labels.

4.1 K-Means objective and steps

The commonly optimized criterion (inertia) is the within-cluster sum of squared Euclidean distances:

$$J = \sum_{i=1}^{n} \min_{k=1,\ldots,K} \|x_i - \mu_k\|^2.$$

Choose K and initial centroids μ_k. Assign each point to the closest centroid; update each centroid to the mean of its assigned points; repeat until convergence or the iteration budget is reached. Different starting centres can lead to different local solutions.

4.2 Worked one-dimensional K-Means numerical

Take four vibration-feature values $x = (1, 2, 8, 9)$ and $K = 2$. Choose initial centres $\mu_1 = 2$, $\mu_2 = 8$.

1. **Assign:** 1 and 2 are nearer centre 2; 8 and 9 are nearer centre 8. (Here the words “centre 2” and “centre 8” name numerical locations, not cluster IDs.)
2. **Update:** $\mu_1 = (1+2)/2 = 1.5$; $\mu_2 = (8+9)/2 = 8.5$.
3. **Compute inertia:** $(1-1.5)^2 + (2-1.5)^2 + (8-8.5)^2 + (9-8.5)^2 = 1.0$.
4. **Reassign:** assignments remain unchanged; this one-dimensional example has converged.

Initial inertia was $(1-2)^2 + (2-2)^2 + (8-8)^2 + (9-8)^2 = 2.0$; updated inertia decreased to 1.0.

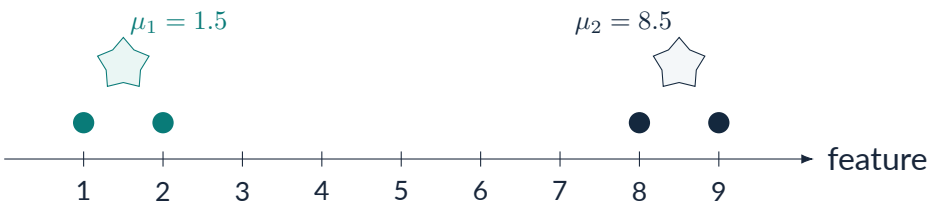

Figure 5. Original 1D K-Means example: filled circles are data, stars represent recalculated centroids after one assignment.

4.3 Limitations and correct preprocessing

K-Means uses Euclidean distances and is sensitive to feature scale. A temperature feature in degrees and a high-amplitude vibration channel can dominate a distance calculation; fit any scaling transformation on training data only, then apply it to held-out data. Cluster boundaries may be unsuitable for non-spherical shapes, imbalanced sizes, categorical features or clusters with many outliers.

5 Support Vector Machines (SVM): Concept

In simple language
A linear SVM finds a separating boundary between labelled classes and seeks a wide margin around it. Points near the margin influence the solution strongly and are called support vectors.

5.1 Decision function, labels and margin

For class labels $y \in \{-1, +1\}$, a linear decision function is $f(x) = w^T x + b$. Class prediction often uses $\text{sign}(f(x))$. The decision boundary is $w^T x + b = 0$ and the margin boundaries for the canonical hard-margin scaling are $w^T x + b = +1$ and -1. The geometric distance between those two margin planes is $2/\|w\|$.

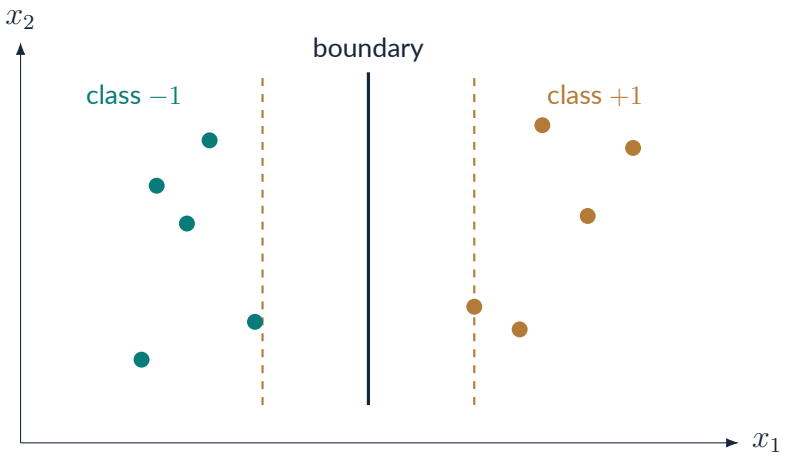

Figure 6. Original schematic linear SVM showing a separating hyperplane and two margin lines; data positions are illustrative, not a fitted optimizer output.

5.2 Hand-solved classification

Choose an illustrative linear model $w = [1, 0]^T$ and $b = -3$, so $f(x) = x_1 - 3$. Its separating line is $x_1 = 3$; canonical margin lines are $x_1 = 2$ and $x_1 = 4$ because w has norm one. Point $x_1 = 5$ predicts class $+1$ because $f = 2$, while point $x_1 = 2$ has score -1 and predicts class -1.

5.3 Soft margins and kernels: introduction only

Many real datasets overlap or are not separable with a straight line. A soft-margin SVM permits violations penalized by parameter C, trading margin size against training errors. Kernel methods can represent nonlinear decision boundaries through suitable similarity functions. Neither a huge C nor a complex kernel guarantees better generalization; evaluate on unseen data.

SVM is not a sensor safety rule. A classifier output or decision score is not automatically a calibrated event probability, and an SVM trained in laboratory light may fail in dusty or reflective operating conditions.

6 Neural Networks: Introductory Concepts

In simple language
An artificial neuron combines input values with weights, adds a bias and passes the result through an activation function. A neural network arranges many such units into layers and adjusts their weights during training.

6.1 A single neuron: equation and worked calculation

$$a = \sigma\left(\sum_j w_j x_j + b\right).$$

For two sensor inputs $x = (0.5, 1.0)$, weights $w = (2, -1)$, bias $b = 0.5$, the pre-activation score is $z = 2(0.5) - 1(1.0) + 0.5 = 0.5$. If σ is ReLU, $a = \max(0, 0.5) = 0.5$. If σ is sigmoid, $a = 1/(1 + e^{-0.5}) \approx 0.62246$. These are activation outputs of a demonstration neuron, not a calibrated defect probability.

6.2 Inputs, hidden layers, output and training

A simple feed-forward network maps measured features through hidden transformations to an output. For a classifier, a final sigmoid or softmax may be used with appropriate losses; for regression a linear output may be suitable. *Back-propagation* computes gradients so an optimizer can adjust weights to reduce training loss. More layers do not guarantee reliability on new machines.

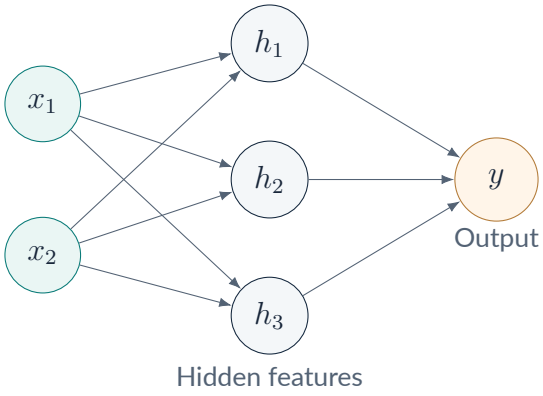

Figure 7. Original schematic two-input, three-hidden-unit, one-output network; weights and activation functions would be learned/configured.

6.3 Neural-network failure modes

A model can memorize machine IDs, operator handwriting or background texture instead of learning actual defects. It may react unpredictably to lighting changes, rare conditions and novel sensor faults. Use representative data, separation by machine/date, robustness checks and human oversight where needed.

7 Model Evaluation and Overfitting

In simple language
A model must generalize: it should behave well on genuinely new robots, components or operating runs, not merely on examples it has already seen.

7.1 Training, validation and test sets

Train: fit model weights and preprocessing statistics. **Validation:** compare hyperparameters, thresholds and model candidates. **Test:** estimate performance after final choices have been fixed. If images from the same physical part appear in both train and test, the reported result may be overly optimistic even if the files have different names.

7.2 Data leakage in robotic data

A sensor sequence often contains many nearly identical adjacent frames. Randomly splitting frames can place near-duplicates from one failure event in both training and test sets. Group by robot or machine, production batch, part identity or calendar time as appropriate to the deployment question. Fit normalization, imputation and feature selection only on training folds.

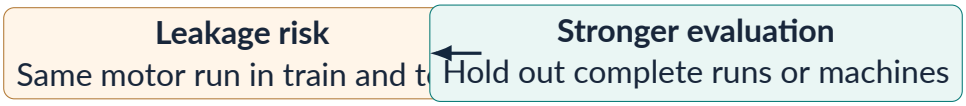

Figure 8. Original example of group-aware validation for correlated robot sensor streams.

7.3 Underfitting versus overfitting

Underfitting: both training and validation error remain high because the model cannot capture useful structure. Overfitting: training error becomes very low while validation error increases. A good fit depends on the data, noise, objective and deployment distribution; one curve is not a universal rule.

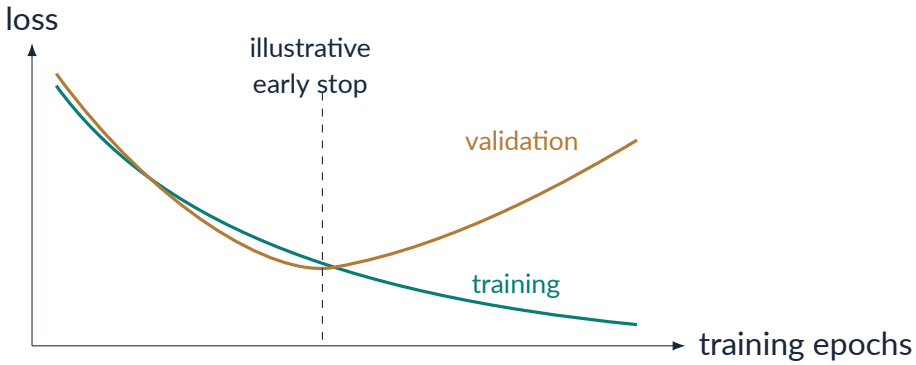

Figure 9. Original conceptual learning curves. Validation loss can rise after excessive training; this is not generated from a real dataset.

7.4 Cross-validation and honest metrics

K-fold cross-validation splits training data into K folds, fits on $K-1$, evaluates on the held-out fold, and rotates. It helps assess model variability and select hyperparameters. **Do not** tune repeatedly against the final untouched test set; that leaks test knowledge into development. Use an evaluation unit matching deployment: per part, per machine, per shift or per failure episode.

Metric policy. In equipment fault detection, compute recall for real faults, false alarms per operating hour, latency and performance under changing loads. High image accuracy on a curated dataset does not prove operational uptime or a safety performance level.

8 Robotics and Manufacturing Applications

8.1 Object classification and inspection

A robot camera captures an image. Calibration and lighting control make pixel values more consistent; segmentation or features isolate the relevant part. A

classifier predicts a category and confidence score, after which a separate decision layer may accept, reject or send the part for human review. Evaluate per-part false rejects and missed defects, not only frame-wise accuracy.

8.2 Fault detection from industrial sensors

Vibration RMS, motor current, encoder deviation and temperature can be features for a supervised fault classifier. The dataset should contain labelled examples of each fault at representative operating loads. K-Means can also group unfamiliar patterns, but an unlabeled cluster does not identify a specific physical failure without diagnosis.

8.3 Predictive maintenance

A predictive-maintenance workflow uses historical sensor trajectories and maintenance records to anticipate the likelihood or timing of future failure. A regression model may estimate remaining useful life; a classification model may flag risk within the next operating period. Censored failures, rare breakdowns, changing maintenance policy and nonstationary wear can make the real prediction task much harder than a classroom example.

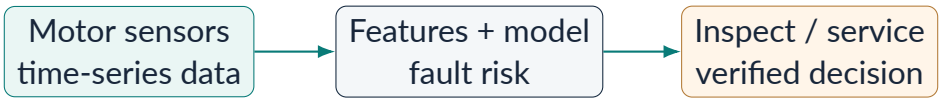

Figure 10. Original predictive-maintenance pipeline; the model recommendation is checked by a separate maintenance decision process.

8.4 Robotics deployment checklist

1. Identify what prediction must improve and define its cost of errors.
2. Get representative data and correctly justified labels; document consent and access permissions.
3. Split by the actual generalization target, such as new robot runs, parts or machines.
4. Build a transparent baseline; compare ML against the simple rule it is intended to replace.

5. Measure precision/recall, regression error, runtime and false alarms; check subgroups and shifts.
6. Specify model updates, drift monitors, uncertainty handling and fail-safe operating modes.
7. Keep protective interlocks and safety-related control functions independently validated.

9 Exam Practice and Worked Answer Bank

9.1 Twelve original syllabus-aligned questions

1. Define machine learning and give one robotics example.
2. Distinguish supervised learning, unsupervised learning and rule-based automation.
3. Classify tasks: defect category, tool-wear amount and unlabeled vibration clusters.
4. Compute accuracy, precision, recall and F1 for $(TP, FP, FN, TN) = (8, 3, 2, 17)$.
5. A process generates 999 good parts and 1 defective. Explain why 99.9% accuracy may be misleading.
6. Calculate MAE, MSE and RMSE for reference $(5, 7, 9)$ and predictions $(4, 7, 10)$.
7. Complete one K-Means iteration for $x = (1, 2, 8, 9)$ and centres $(2, 8)$.
8. Given SVM score $f(x) = x_1 - 3$, classify $x_1 = 2$ and $x_1 = 5$.
9. Compute a neuron's pre-activation with inputs $(.5, 1)$, weights $(2, -1)$ and bias $.5$.
10. Explain overfitting using train and validation loss curves.
11. Explain two leakage paths in a robot-camera dataset.
12. Design a valid model evaluation plan for bearing-fault detection across different machines.

9.2 Concise answer key

Q.	Correct result / principle
1	ML fits statistical relations from data; part classification from images is one robotics example.
2	Supervised uses labels; unsupervised has no targets; hand-written rules are explicit logic.
3	Defect category = classification; tool wear amount = regression; vibration groups = clustering.
4	Accuracy $25/30 = .8333$, precision $8/11 = .7273$, recall $.8$, F1 $16/21 = .7619$.
5	Always "good" misses the only defect: fault recall 0% despite high accuracy.
6	MAE $2/3$, MSE $2/3$, RMSE $\sqrt{2/3} \approx .8165$ (with appropriate units).

Q.	Correct result / principle
7	Clusters $\{1, 2\}$ and $\{8, 9\}$, new centres $(1.5, 8.5)$, inertia 1.0.
8	$f(2) = -1$ class -1; $f(5) = 2$ class $+1$.
9	$z = 2(0.5) - 1(1) + 0.5 = 0.5$; ReLU 0.5; sigmoid $\approx .62246$.
10	Training loss low but held-out validation loss starts rising; generalization is worsening.
11	Near-duplicate frames or same part/machine in both train and test; leak from scaling before split.
12	Label faults; hold out complete machines/runs; evaluate recall, false-alarm rate and robustness.

9.3 Frequent mistakes

1. Confusing classification of a fault type with numerical regression of remaining life.
2. Treating K-Means cluster IDs as confirmed physical failure labels.
3. Saying all SVMs are linear or that an SVM score is automatically probability.
4. Assuming that deeper neural networks always generalize better.

5. Calling accuracy a sufficient metric on heavily imbalanced fault data.
6. Scaling inputs on the full dataset before separating test data.
7. Using adjacent frames from the same robot run as independent test cases.
8. Ignoring the operating cost of false positives and false negatives.
9. Replacing a protective interlock with an unverified ML classification.

One-minute summary

Supervised learning predicts labels or numerical targets; unsupervised learning groups unlabeled data. Classification metrics use a confusion matrix; regression needs error in correct units. K-Means alternates assignments and centroid means. SVM uses a boundary and margin. Neural networks apply weighted transformations. Always test on genuinely new robotic data, prevent leakage, and evaluate drift and failure costs.

Final review workshop: avoiding misleading robot-ML results

The same camera scene must not leak across train and test

Imagine 100 images captured from the *same* manufactured part in the same second. Randomly allocating 80 to training and 20 to test measures recognition of nearly identical views, not performance on an *unseen part*. Instead, assign entire physical parts or production batches to the test set. If the intended deployment is a different robot, hold out complete robots. Document the test population before fitting the model.

Worked example: class imbalance and the apparent 99% model
Out of 100 inspected parts, 99 are good and 1 is defective. A system that predicts *good* for every part obtains $99/100 = 99\%$ accuracy, but it finds $0/1 = 0\%$ of defects. For the defective class, $TP = 0$, $FN = 1$, $FP = 0$, $TN = 99$.
Interpretation: The accuracy sounds excellent; the defect recall is zero. For an inspection station, report defect recall, false rejection rate, confusion counts per batch and the consequences of errors. Precision is undefined when there are no predicted positives; a software package may represent it as zero by convention, but this is *not* the same as an observed precision of zero.

How preprocessing can invalidate a fair experiment

Suppose two sensor channels have ranges $[0, 2]$ and $[0, 2000]$. Fitting a scaler using all samples, including the test set, allows held-out statistics to influence model preparation. Instead: split robot runs first; fit the scaler and any feature-selection logic using training data only; transform validation/test data using those fitted parameters. In cross-validation, fit the scaler separately within each training fold.

Engineering decision gate
Record the held-out parts, shifts and machines; preserve the exact preprocessing and model pipeline; measure uncertainty, false-negative cost and

runtime; and keep safety-rated interlocks independent of the unverified ML prediction chain.

Sources, figure provenance and verification scope

Syllabus: SPPU, TE Robotics & Automation, 2024 Pattern, PCC301-ROA, Unit IV *Machine Learning for Robotics*, user-provided *TE Syllabus.pdf*, printed and physical PDF p. 15. This guide tracks its named topics.

Technical sources:

1. Google for Developers, *Machine Learning Crash Course*, regression/classification, generalization and overfitting. https://developers.google.com/machine-learning/crash-course/
2. scikit-learn, *K-Means*, official API and inertia definition. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
3. scikit-learn, *Support Vector Machines*, introductory user guide. https://scikit-learn.org/stable/modules/svm.html
4. scikit-learn, *Cross-validation: evaluating estimator performance*. https://scikit-learn.org/stable/modules/cross_validation.html
5. scikit-learn, *Common pitfalls and recommended practices*, data leakage. https://scikit-learn.org/stable/common_pitfalls.html

Figure provenance: All figures in this PDF are new TikZ vector drawings. No external graph, copyrighted photo or third-party figure was embedded. Numerical data and example operating scenarios are constructed for teaching and are not claimed to be measured industrial performance.

Limits: A passing PDF or arithmetic test does not demonstrate predictive generalization, industrial safety certification or successful deployment on a physical robot.

AIFR UNIT IV | VERIFIED EXAM SUPPLEMENT

Zenith Study Hub | SPPU TE Robotics and Automation | PCC301-ROA | 2024 Pattern

Review addition to the original 18-page Machine Learning guide. This appendix works through the exact Unit IV question-bank numerical data. It is a review candidate, not an instructor-approved release.

Audit summary

Scope	Status
Official syllabus: supervised/unsupervised, regression/classification, K-Means, SVM, ANN, evaluation, overfitting, robotics applications	Topics present in original 18-page guide
Source guide arithmetic: confusion metrics, MAE/MSE/RMSE, 1D K-Means, illustrative SVM, sigmoid	Spot calculations independently confirmed
Question bank Q2, Q4, Q6, Q8, Q10, Q12	Worked below with numerical assumptions stated
Question bank Q8 weight mapping	Underspecified in source; conditional demonstration supplied
Question bank Q12 bias terms	Unspecified; example assumes zero biases

Q2 | K-Means on seven robotics samples: two full iterations

Initial centres C1=(1,1), C2=(5,7). Use Euclidean distance; assign to the nearest centre, then recompute each centre as its assigned point mean. At the initial centres, R3 is equidistant from both; choose cluster 1 for this worked trace. An alternative tie-break could change the subsequent iterations.

Point	Coordinates	Iter 1	Iter 2
R1	(1,1)	C1	C1
R2	(1.5,2)	C1	C1
R3	(3,4)	C1	C2
R4	(5,7)	C2	C2
R5	(3.5,5)	C2	C2
R6	(4.5,5)	C2	C2
R7	(3.5,4.5)	C2	C2

After iteration 1: C1=((1+1.5+3)/3,(1+2+4)/3)=(1.8333,2.3333). C2=((5+3.5+4.5+3.5)/4,(7+5+5+4.5)/4)=(4.1250,5.3750).

After iteration 2: C1=((1+1.5)/2,(1+2)/2)=(1.25,1.50). C2=((3+5+3.5+4.5+3.5)/5,(4+7+5+5+4.5)/5)=(3.90,5.10).

Answer: After the second iteration: C1={R1,R2}, centroid (1.25,1.50); C2={R3,R4,R5,R6,R7}, centroid (3.90,5.10). The next reassignment should be checked if convergence is required; the question requests only two iterations.

Q4 | Five-point K-Means: first iteration

Initial centres C1=(4,4), C2=(1,3). Distances from P3=(3,2) to both initial centres are sqrt(5), giving a genuine tie. The question does not define a tie-break; below, assign a tied point to C1.

Point	Distance to C1	Distance to C2	Assigned
P1 (4,4)	0	sqrt(10)	C1
P2 (1,3)	sqrt(10)	0	C2
P3 (3,2)	sqrt(5)	sqrt(5)	C1 (tie)
P4 (2,4)	2	sqrt(2)	C2
P5 (3,5)	sqrt(2)	sqrt(8)	C1

With P3 assigned C1: C1=((4+3+3)/3, (4+2+5)/3)=(10/3,11/3); C2=((1+2)/2,(3+4)/2)=(1.5,3.5).

With P3 assigned C2 instead: C1=((4+3)/2,(4+5)/2)=(3.5,4.5); C2=((1+3+2)/3,(3+2+4)/3)=(2,3). Both answers are valid unless the instructor specifies a tie rule.

Q6 | Pizza diameter linear regression

Data: x=[8,10,12] inches; y=[10,13,16] dollars. Mean x=10, mean y=13. Least-squares slope m=sum((x-xbar)(y-ybar))/sum((x-xbar)^2)=12/8=1.5. Intercept b=13-1.5(10)=-2.

Best-fit line: y=1.5x-2. Predicted price at 20 inches: **$28**. This is a numerical exercise from the question bank, not a robot-specific dataset. A 20-inch input lies outside the observed 8-12-inch range; extrapolation is not validated.

Q8 | Single hidden neuron, ambiguous weight mapping

Question states i1=0.4, i2=0.2, four weights w1=0.20, w2=0.10, w3=0.12, w4=0.26 and bias=0.25, but does **not** specify which two weights connect to neuron h1. A unique numerical answer is impossible without that assignment.

Conditional example (if h1 receives w1 and w2): z_h1 = 0.4(0.20)+0.2(0.10)+0.25=0.35; sigmoid(z)=1/(1+exp(-0.35))=**0.586618**.

Alternative common diagram convention (if h1 receives w1 and w3): z_h1=0.4(0.20)+0.2(0.12)+0.25=0.354; sigmoid=**0.587587**. State the mapping with the answer; no source diagram was provided.

Q10 | Least-squares regression for five data pairs

x=[3,2,1,4,6], y=[5,4,4,7,6]. n=5; sum x=16; sum y=26; sum x^2=66; sum xy=91. Slope m=[5(91)-16(26)]/[5(66)-16^2]=39/74=0.527027.

Check the product sum explicitly: sum(xy)=3*5+2*4+1*4+4*7+6*6=91. Hence the slope is **0.527027**.

Intercept b=(26-m*16)/5=130/37=**3.513514**. Final answer: **y = 0.527027 x + 3.513514**.

Q12 | Two-hidden-neuron feed-forward network

Inputs x1=.35, x2=.9. Connections: w13=.1, w14=.4, w23=.8, w24=.6, w35=.3, w45=.9. The question gives no biases; assume all biases are zero. Sigmoid(t)=1/(1+exp(-t)).

z3=.35(.1)+.9(.8)=.755; h3=sigmoid(.755)=**0.680267**. z4=.35(.4)+.9(.6)=.68; h4=sigmoid(.68)=**0.663739**.

z5=.3(h3)+.9(h4)=**0.801445**; output o5=sigmoid(z5)=**0.690283**. With nonzero biases, answer changes.

Editorial and publication QA notes

Verified strengths: The core guide contains the official Unit IV topic families, learner explanations, 10 captioned teaching figures, classification metrics, regression-error arithmetic, K-Means example, an illustrative SVM decision function, and sigmoid activation. Source diagrams must be judged by rendered appearance as well as extracted text.

Specific improvement: Include exact question-bank calculations rather than substituting unrelated practice datasets. Q4 must disclose the exact tie policy. Q8 is underdetermined without a weight-to-neuron map. Q12 must disclose the zero-bias assumption.

Independent numeric checks: Q2 centroid iteration and assignments reproduced with NumPy; Q4 tied Euclidean distances computed; Q6 and Q10 OLS slopes recomputed; Q8 and Q12 sigmoid values recalculated.

Source authority: User-supplied TE Syllabus.pdf, PCC301-ROA Unit IV, printed page 15. Question data: user-supplied AIFR_Unit_Wise_Question_Bank.pdf, Unit IV, Q2/Q4/Q6/Q8/Q10/Q12.

Release boundary: This document does not claim a production-trained robotics model, field safety certification, or faculty approval. The original edition remains unchanged. This supplement is appended to an existing 18-page review candidate.