Training Supervision

Training Supervision

ML systems can be classified according to the amount and type of supervision they get during training. There are many categories, but we'll discuss the main ones:

- supervised learning
- unsupervised learning
- self-supervised learning
- reinforcement learning

Prompt:

Role--------------------
I am a student of Machine Learning at the college level.
You are an experienced Machine Learning Engineer and Data Scientist.
Context----------------
This term, I am learning Foundations of Machine Learning as an entry-level student.
Expectations-----------
I want to understand the distinction between supervised, unsupervised, self-supervised, and reinforcement learning.
I want you to make clear the distinction in the context of training models.
Output Format---------
Produce a table with the ML technique, the supervision type, and the training type
Sources-----------------
Look for this information in ARXIV articles and in trustworthy blogs like Medium

Output

Training Signals and Objectives in Major Machine Learning Paradigms

ML Technique	Supervision Signal	Typical Training Procedure / Objective
Supervised learning	Human-provided labels for every training example (x → y pairs)	Minimise a loss (e.g., cross-entropy, MSE) between the model's prediction and the true label to learn a mapping for classification or regression; classic methods include linear/logistic regression, decision trees, random forests and deep nets.
Unsupervised learning	**No** external labels – the algorithm relies on structure inherent in the data	Optimise objectives that expose latent structure: cluster similar points (k-means, DBSCAN), learn low-dimensional manifolds (PCA, auto-encoders) or fit a density model; goal is pattern discovery, anomaly detection or compression.
Self-supervised learning	Pseudo-labels created from the data itself (e.g., predict masked tokens, solve contrastive pairs)	Train on a pretext task to learn rich representations without manual labels; the resulting encoder is fine-tuned or probed on downstream tasks with little or no extra supervision.
Reinforcement learning	Scalar reward signal returned by the environment after each action (delayed, possibly sparse)	Through trial-and-error interaction, iteratively update a policy or value function to maximise expected cumulative reward (e.g., Q-learning, policy gradients, PPO).

Supervised Learning

- A type of machine learning where a model is trained on a labeled dataset.
- Each example in the training dataset is paired with a corresponding output value or class label.
- The model learns to map the input features to the output during the training process.
- The ultimate goal of supervised learning is to make accurate predictions for new, unseen data based on the relationships the model learned from the training data.
- Common examples of supervised learning include:
    - regression tasks, where the output is a continuous value
    - classification tasks, where the output is a categorical value.

Supervised Learning, Categorical Target: Spam or Ham?

Training Data

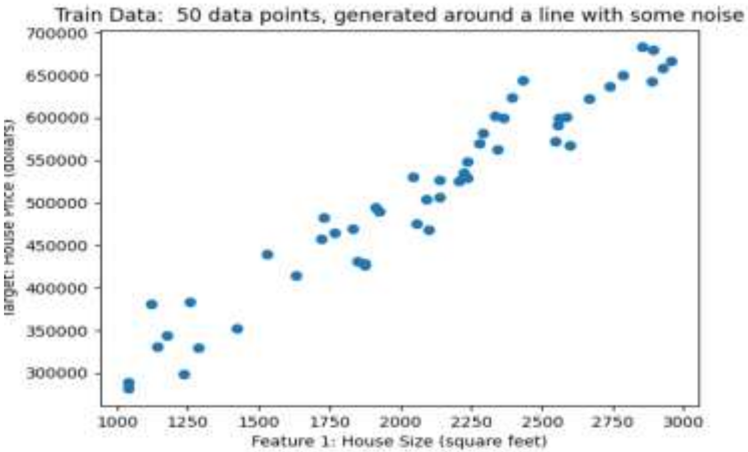
Label

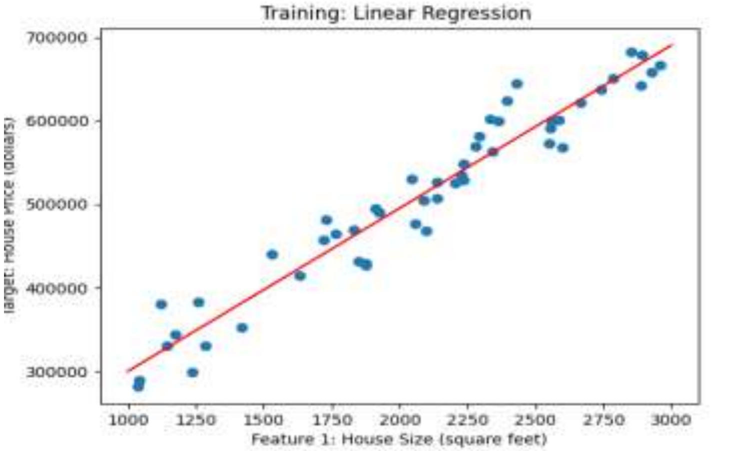
Data Instance	Subject Line Text	Email Body Text	Includes Images	Number of links	**Is Spam?**
Email1	Hello	Hi Friend..	Yes	1	**Yes**
Email2	Click Me	Dear John..	No	1	**No**
Email3	Payroll	Hi, Please..	No	0	**No**

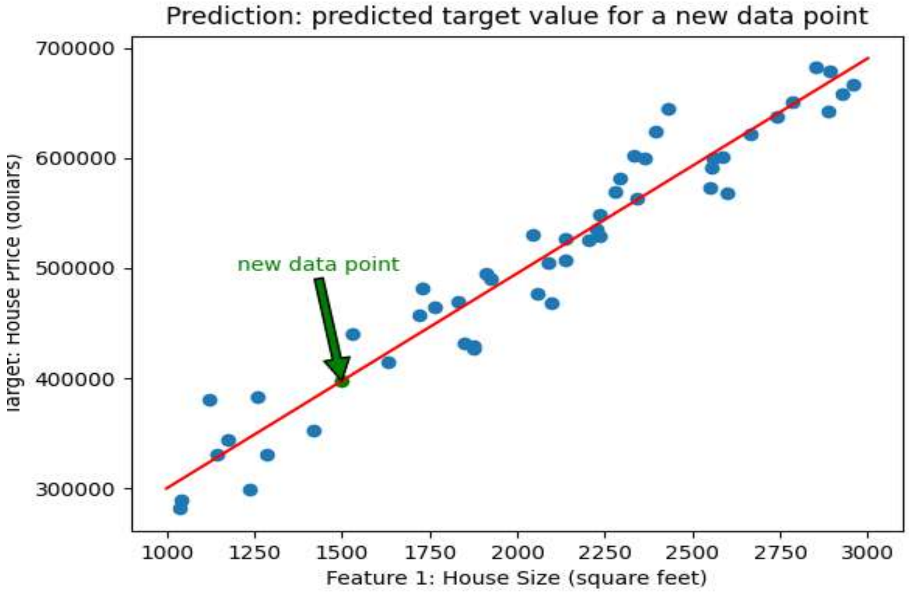
New Data

Data Instance	Subject Line Text	Email Body Text	Includes Images	Number of links	**Is Spam?**
Email4	Catch-up	Hi Emily..	No	0	**?**
Email5	Free Movies	Join now..	Yes	1	**?**

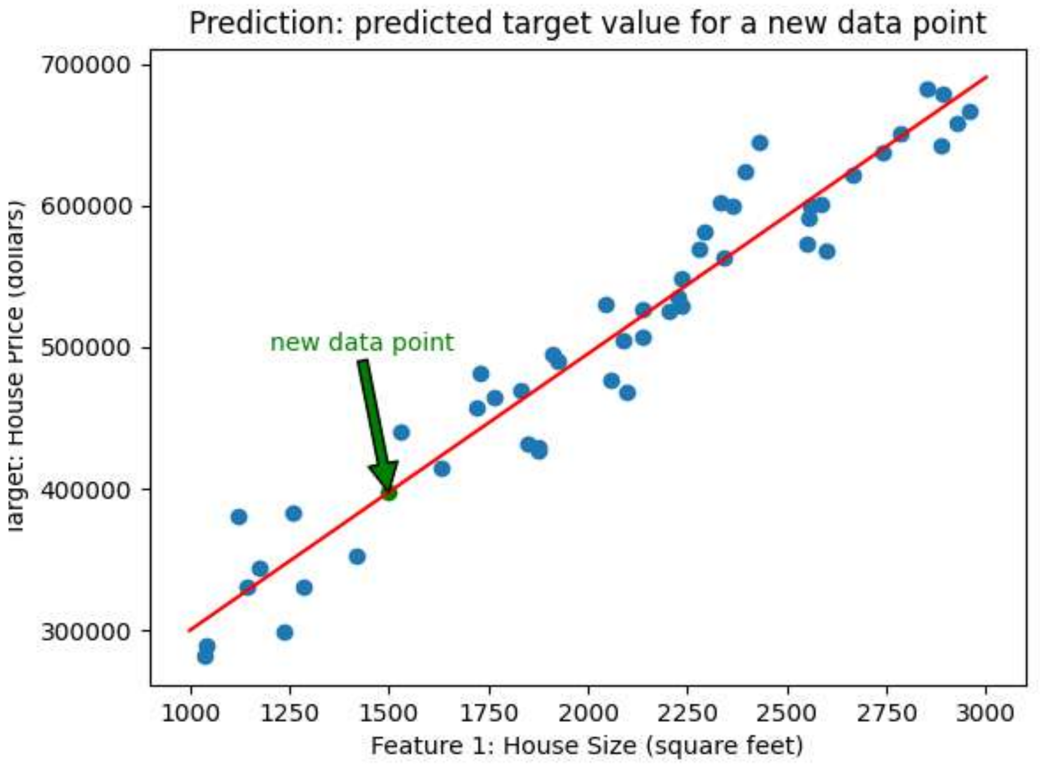
Supervised Learning, continuous target:







Key concepts



→ Upload a snapshot of the chart

Prompt:

Role--------------------
I am a student of Machine Learning at the college level.
You are an experienced Machine Learning Engineer and Data Scientist.
Context----------------
I am an entry-level student, so I have never heard of loss functions or cross-entropy, or MSE.
Expectations-----------
I want to understand the three concepts in the context of training models, so I need you to explain each one.
Output Format---------
Produce a table with the concept, an explanation, its mathematical form, and how it is applied to the use case in the uploaded image.
The mathematical symbols must be easy to copy and paste into MS Word or PowerPoint.

Output

Three Core Error-Measurement Ideas in Machine-Learning Training

Concept	Plain-language role	Mathematical form	How it appears in the plot
Loss Function (general idea)	A single numerical “badness score” for the whole model; training means choosing parameters that **minimise** this score.	$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{nl} \ell\big(y_i, \hat{y}_i(\theta)\big)$$ where θ are the model’s parameters, y_i the true labels and $\hat{y}_i$ the predictions.	The red line is the parameter set (w, b) that makes the chosen loss the smallest possible for the blue dots. (arXiv)
Mean-Squared Error (MSE)	**Standard loss for regression.** Squares each vertical gap, then averages, so large mistakes hurt a lot.	$$\text{MSE}(w,b) = \frac{1}{n}\sum_{i=1}^{n}\big(y_i - (w\,x_i + b)\big)^2$$	For a 1500 sq ft ho, the model predicts that $\hat{y}$≈$400,000. If it later sells for y=$430,000, that adds $(30{,}000)^2 = 9\times10^8$ to the numerator. Minimizing those squared gaps produced the red best-fit line. (arXiv)
Cross-Entropy	**Standard loss for classification.** Compares the model’s predicted probability for each class with the true class (one-hot vector). Penalizes confident wrong answers heavily.	For binary classes (e.g. “expensive” = 1, “affordable” = 0): $$\text{CE}(p,y) = -[y\log p + (1-y)\log(1-p)]$$	Suppose we turn the same data into a yes/no task: Is the price > $500,000? The model outputs p=0.85 for a given house. If the true label is **1** (it is expensive), the loss is −ln0.85≈0.16-\ln0.85\approx0.16; if the true label is **0**, the loss rockets to −ln(1−0.85)≈1.90-\ln(1-0.85)\approx1.90. This “log-penalty” shapes the model to shift probability mass toward the correct side of the red decision boundary. (Medium)

Output

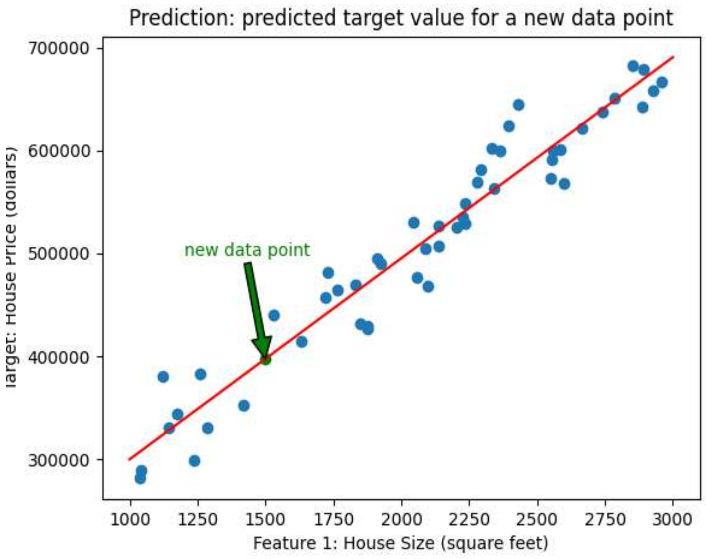
Three Core Error-Measurement Ideas in Machine-Learning Training

Concept	Plain-language role	Mathematical form	How it appears in the plot
Loss Function (general idea)	A single numerical “badness score” for the whole model; training means choosing parameters that **minimize** this score.	$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{nl} \ell\big(y_i, \hat{y}_i(\theta)\big)$$ where θ are the model’s parameters, y_i the true labels and $\hat{y}_i$ the predictions.	The red line is the parameter set (w, b) that makes the chosen loss the smallest possible for the blue dots. (arXiv)

→ Upload a snapshot of the scatter plot
→ Upload a snapshot of the table above

Prompt:

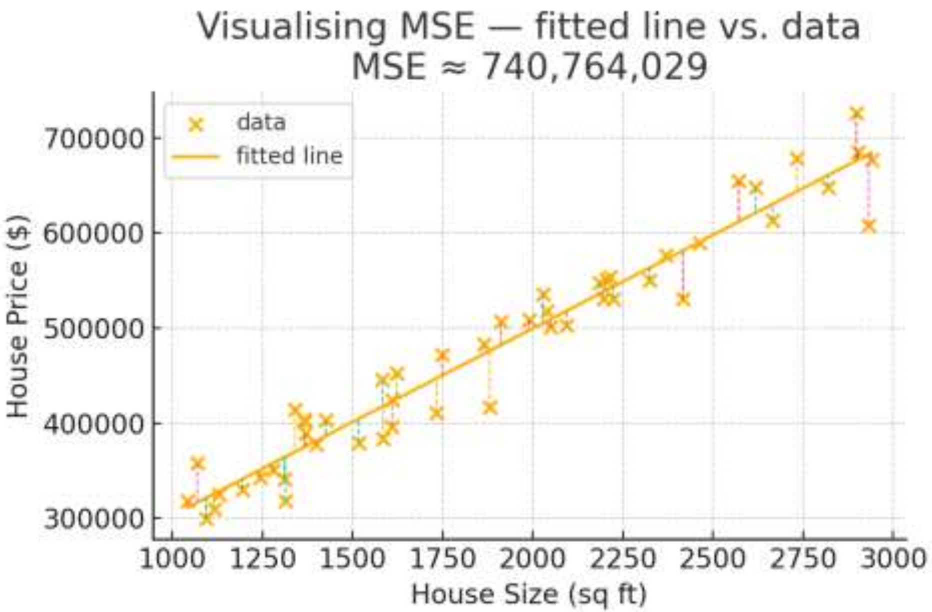
Expectations-----------
Produce another scatter plot or chart to explain the loss function according to the mathematical form and applied to the sample data in the scatter plot.



Visualizing Loss and Mean Squared Error

Concept	Plain-language role	Mathematical form
Loss Function (general idea)	A single numerical “badness score” for the whole model; training means choosing parameters that **minimise** this score.	$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{nl} \ell\big(y_i, \hat{y}_i(\theta)\big)$$ where θ are the model’s parameters, y_i the true labels and $\hat{y}_i$ the predictions.
Mean-Squared Error (MSE)	**Standard loss for regression.** Squares each vertical gap, then averages, so large mistakes hurt a lot.	$$\text{MSE}(w,b) = \frac{1}{n}\sum_{i=1}^{n}\big(y_i - (w\,x_i + b)\big)^2$$

Output



Concept	Plain-language role	Mathematical form
Loss Function (general idea)	A single numerical “badness score” for the whole model; training means choosing parameters that **minimise** this score.	$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell\big(y_i, \hat{y}_i(\theta)\big)$$ where θ are the model's parameters, y_i the true labels and $\hat{y}_i$ the predictions.

Loss still not clear?

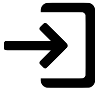
→ Upload a snapshot of the Loss equation (highlighted)

Prompt:

Expectations-----------
Please explain the loss in terms of this equation.

Loss still not clear?

Concept	Plain-language role	Mathematical form
Loss Function (general idea)	A single numerical "badness score" for the whole model; training means choosing parameters that **minimise** this score.	$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{nl} \ell\big(y_i, \hat{y}_i(\theta)\big)$$ where θ are the model's parameters, y_i the true labels and $\hat{y}_i$ the predictions.


Output

Symbol	What it is	Role during training
n	Number of training examples (blue dots in the scatter plot).	Sets how many individual errors we average.
I	Index of a single data point (runs from 1 to n).	Let us loop through all records.
y_i	**True** target value for point i. In the house-price plot, it is the actual sale price of house i.	The ground-truth we want to approximate.
$\hat{y}_i(\theta)$	Model's **prediction** for the same point, produced with parameters θ. For linear regression: $\hat{y} = wx_i + b$.	Moves when we tweak θ. Good settings make y_i close to y.
$\ell(\cdot)$	**Point-wise loss function** – measures the error for one example. Common choices: • Squared error: $(y - \hat{y}_i)^2$ • Cross-entropy: $-[y \log p + (1-y)\log(1-p)]$	Converts each "vertical gap" into a non-negative penalty.
$\mathcal{L}(\theta)$	**Overall (average) loss** – the single number we try to minimize.	Training is the search for θ*=arg $\min_{\theta} \mathcal{L}(\theta)$

Generative AI

Model Training

Prompt:

Expectations-----------
Now, explain the training process for linear regression, step by step.
Conditions-------------
Use the scatter plot, equations, and use case you produced & used before.
I don't have a data source, so generate the housing data from the scatter plots.
You can assume I have Visual Studio with Python and Jupyter Notebook extensions already installed.
Format-----------------
I expect Python code that is ready to run in a Jupyter Notebook.
Add all library installation (pip) commands before generating the code and starting the training.
Add Markdown before each step of the training so I can understand the code cell underneath.
Give me a link to download the IPYNB.

1. Imports & random seed

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from IPython.display import display, Markdown
np.random.seed(42)
```

[2] ✓ 20.1s Python

2. Generate synthetic housing data

```python
n = 50
X = np.random.uniform(1_000, 3_000, n)
true_w, true_b = 200, 100_000
noise = np.random.normal(0, 30_000, n)
y = true_w * X + true_b + noise
df = pd.DataFrame({'size_sqft': X, 'price_$': y})
df.head()
```

[3] ✓ 0.0s Open 'df' in Data Wrangler Python

	# size_sqft	# price_$
0	1749.080237694725	471970.0449388073
1	2901.4286128198323	685426.7709996655
2	2463.98788362281	589328.1282529148
3	2197.3169683940732	530430.282811136
4	1312.037280884873	318051.79646595183

5 rows x 2 cols 10 per page Page 1 of 1

Gradient-descent training

The goal is to minimize the average squared error, defined as the loss function:
$L(w,b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

Where $\hat{y}_i = wx_i + b$ is the predicted value.
Substituting $\hat{y}_i$ into the loss function, we get:
$L(w,b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (wx_i + b))^2$

Now, let's calculate the partial derivatives with respect to w and b.

Gradient-descent training

Partial Derivative with respect to w ($\partial L/\partial w$)

We apply the chain rule. Let $u = y_i - (wx_i + b)$. Then $L = \frac{1}{n}\sum u^2$.

1. **Differentiate the outer function:** $\frac{\partial}{\partial u}(u^2) = 2u$
   So, $\frac{\partial}{\partial(y_i - (wx_i + b))}(y_i - (wx_i + b))^2 = 2(y_i - (wx_i + b))$
2. **Differentiate the inner function with respect to w:** $\frac{\partial}{\partial w}(y_i - (wx_i + b))$
   $= \frac{\partial}{\partial w}(y_i - wx_i - b)$
   $= -x_i$ (since y_i and b are constants with respect to w)
3. **Multiply them together and apply the summation:**
   $\frac{\partial L}{\partial w} = \frac{1}{n}\sum_{i=1}^{n}[2(y_i - (wx_i + b)) \cdot (-x_i)]$
   $\frac{\partial L}{\partial w} = \frac{1}{n}\sum_{i=1}^{n}[-2(y_i - \hat{y}_i)x_i]$
   $\frac{\partial L}{\partial w} = -\frac{2}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)x_i$

Gradient-descent training

Partial Derivative with respect to b ($\partial L/\partial b$)

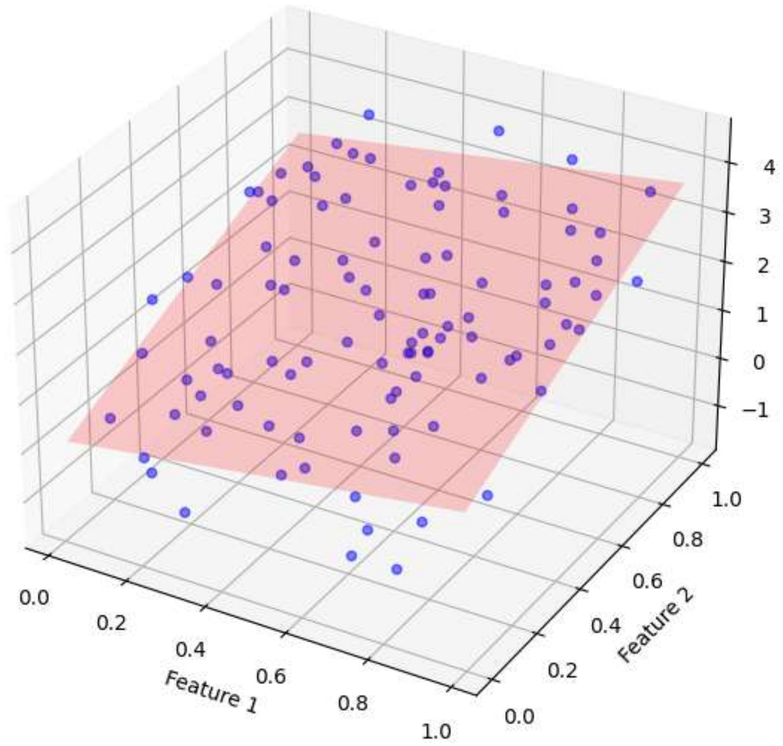
Again, we apply the chain rule. Let $u = y_i - (wx_i + b)$.

1. **Differentiate the outer function:** As before, $2(y_i - (wx_i + b))$.
2. **Differentiate the inner function with respect to b:** $\frac{\partial}{\partial b}(y_i - (wx_i + b))$
   $= \frac{\partial}{\partial b}(y_i - wx_i - b)$
   $= -1$ (since y_i and wx_i are constants with respect to b)
3. **Multiply them together and apply the summation:**
   $\frac{\partial L}{\partial b} = \frac{1}{n}\sum_{i=1}^{n}[2(y_i - (wx_i + b)) \cdot (-1)]$
   $\frac{\partial L}{\partial b} = \frac{1}{n}\sum_{i=1}^{n}[-2(y_i - \hat{y}_i)]$
   $\frac{\partial L}{\partial b} = -\frac{2}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)$

This matches the formula given in your screenshot: $\partial L/\partial b = -\frac{2}{n}\sum(y - \hat{y})$.

Gradient-descent training

These derivatives are essential for gradient descent, as they tell us the direction of the steepest ascent of the loss function. By moving in the opposite direction (subtracting the learning rate times the gradient), we iteratively adjust w and b to minimize the loss.



Gradient-descent training

Goal: Minimise the average squared error $\mathcal{L}(w,b) = \frac{1}{n}\sum_i (y_i - \hat{y}_i)^2$

1. **Initialise** parameters: *start with* `w, b` ≈ 0.
2. **Predict:** $\hat{y}_i = wx_i + b$.
3. **Compute gradients:**
   - $\partial\mathcal{L}/\partial w = \frac{-2}{n}\sum(y-\hat{y})x$
   - $\partial\mathcal{L}/\partial b = \frac{-2}{n}\sum(y-\hat{y})$
4. **Update:** $w \leftarrow w - \eta\,\partial\mathcal{L}/\partial w, \quad b \leftarrow b - \eta\,\partial\mathcal{L}/\partial b$
5. **Loop** steps 2-4 until the loss no longer drops (convergence).

Outcome: final w, b define the red best-fit line that minimises the mean-squared error.

Prompt:
Expectations-----------
I want to dive deeper into the training algorithm and understand the variables there, as well as how they are later used to make predictions on housing costs based on square footage.
Conditions-------------
Add to the last output you created an explanation of the 'lr' variable in the step 6 code cell, its role in the training process, how it affects the generation of the 'Normal Eq. solution' and the 'Gradient Descent' lines, and of the ‘w_hat’, ‘b_hat’, ‘w_gd’, and ‘b_gd’ values.
Format-----------------
Format the output as Jupyter Notebook Markdown

Additional help with Model Training

Learning Rate

What is it?	**Why it matters**

A small positive scalar that scales the gradient update.
`w ← w − lr · ∂L/∂w`
`b ← b − lr · ∂L/∂b`
	Controls the step size while descending the loss landscape.
	
	▪ **Too small** → many epochs, slow convergence.
	▪ **Too large** → can overshoot, diverge, or oscillate.

Variables

Variable	**How it is obtained**	**Numerical mear**
`w_hat`	Closed-form slope from the **Normal Equation:** `(Xᵀ X)⁻¹ Xᵀ y`	Increase in predic foot.
`b_hat`	Closed-form intercept (same solve)	Baseline price wh extrapolation anc
`w_gd`	Slope found after 2 000 **gradient-descent** updates with `lr`.	Approximates `w_` `lr`, epochs.
`b_gd`	Intercept from gradient-descent training	Approximates `b_`

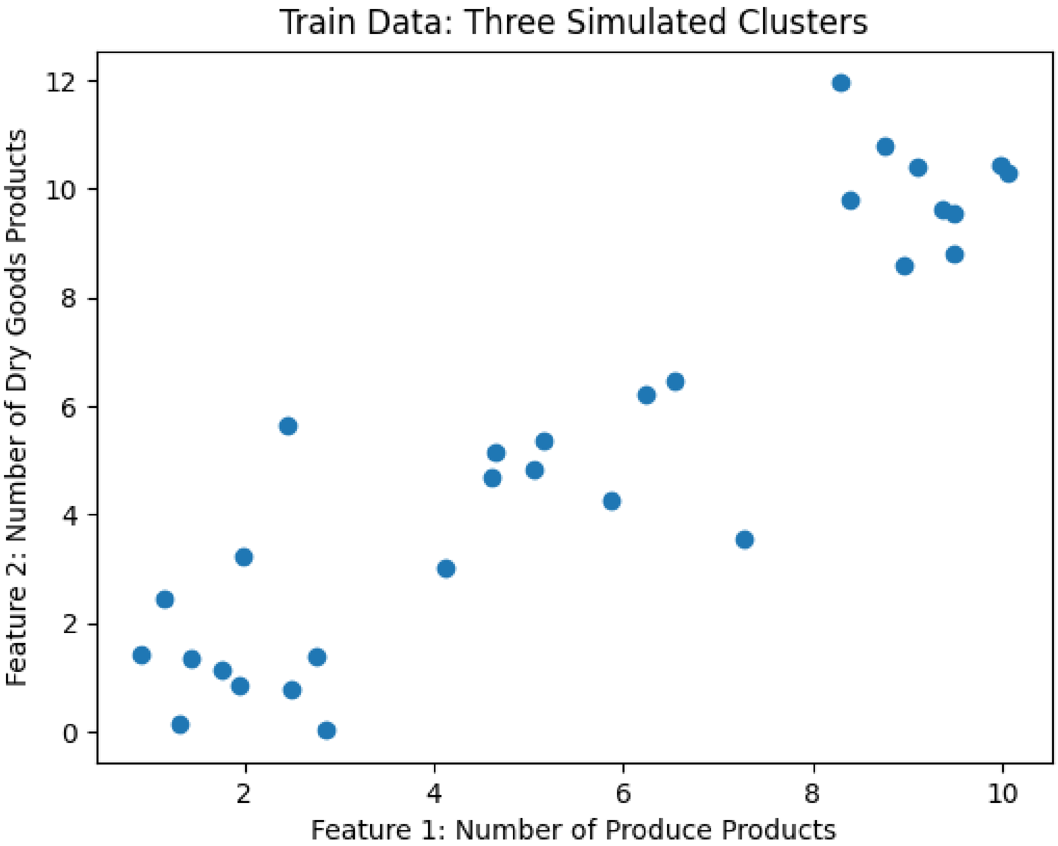
Ways to gauge the accuracy of a linear regression model

Category	Method	What it tells you
Hold-out evaluation	**Train / validation / test split**	Performance on data that the model never saw during training.
Cross-validation	**k-Fold CV** (e.g., k = 5 or 10)	Average error over k different train/test partitions; variance of the scores.
Point-error metrics (same units as target)	**MAE** (Mean Absolute Error)	Typical absolute dollar error; less sensitive to outliers.
	MSE / RMSE (Mean / Root-Mean-Squared Error)	Penalises large errors more than small ones. RMSE is in the same units as the target.
	MAPE (Mean Absolute Percentage Error)	Average relative error (%)
Goodness-of-fit	**R^2 (coefficient of determination)**	Fraction of variance in yy explained by the model (1 = perfect).
	Adjusted R^2	R^2 corrected for the number of predictors.
Statistical inference	**F-statistic & overall p-value**	Whether the regression explains significantly more variance than a constant model.
	t-tests for coefficients	Whether each slope differs significantly from 0 given the sample size and noise.
Residual diagnostics	**Residual plot**	Reveals non-linearity, heteroscedasticity, or outliers.
	Q–Q plot / Shapiro–Wilk test	Check normality of residuals (a linear-regression assumption for inference).
	Breusch–Pagan / White test	Detect heteroscedasticity (variance of residuals changes with xx).
Model-selection criteria	**AIC / BIC**	Trade-off between fit quality and model complexity.
Learning-curve diagnostics	**Train vs. validation error vs. sample size**	Detect high bias (both errors high) vs. high variance (train ≪ val).

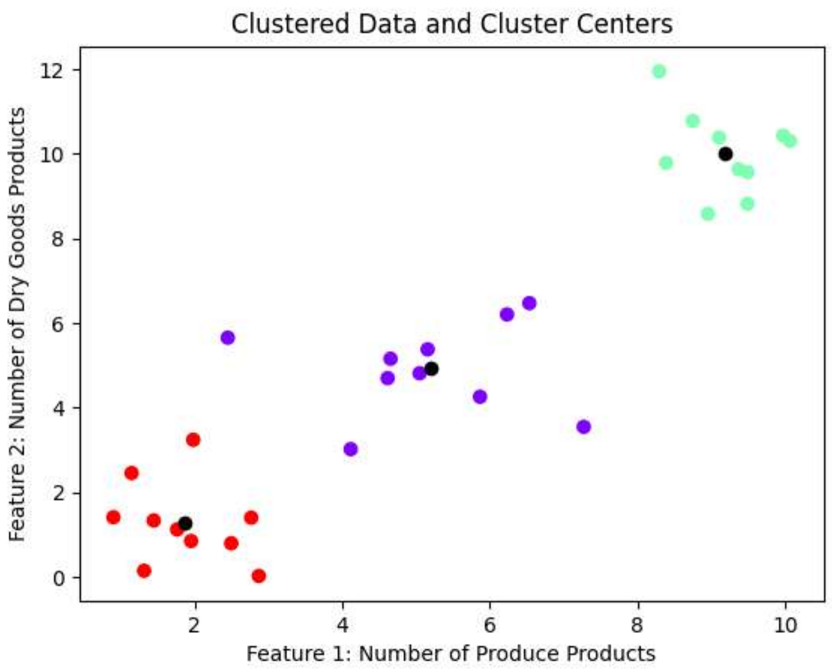
Unsupervised Learning

- A type of machine learning where a model is trained on an **unlabeled** dataset.
- The training data consists of a set of data instances without any corresponding target values.
- The goal in unsupervised learning is to discover the underlying structure in the data. The model attempts to **find patterns** or intrinsic relationships within the input data itself.
- Common examples of unsupervised learning include
    - clustering tasks, where the objective is to group similar instances together
    - anomaly detection tasks, where the objective is to detect unusual instances based on their feature characteristics.

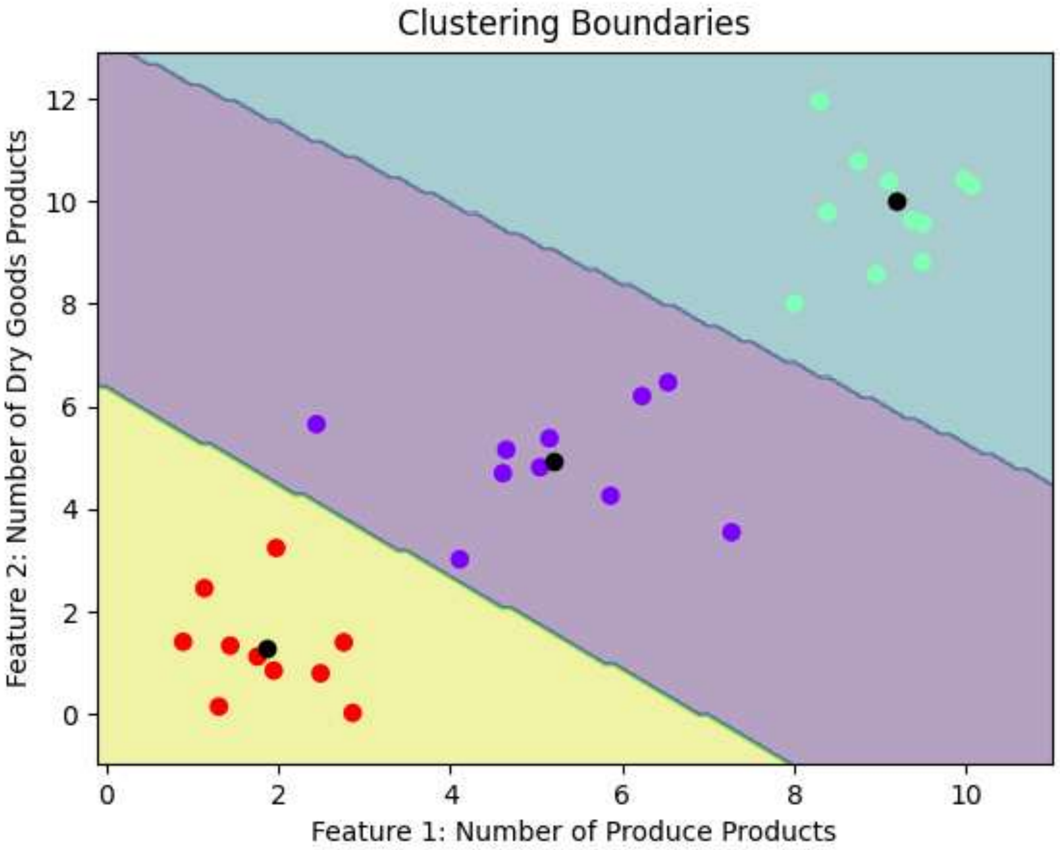
Unsupervised Learning: Clustering



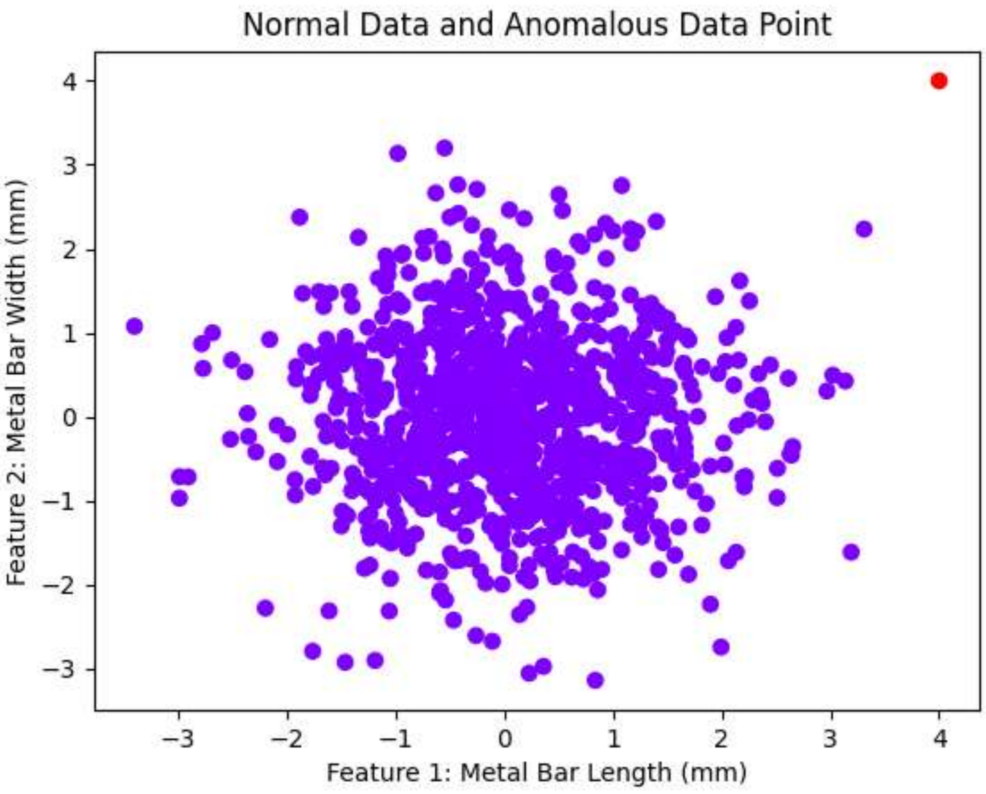
Unsupervised Learning: Clustering



Unsupervised Learning: Clustering



Anomaly detection



Self-Supervised Learning

- A type learning where t**he data provides the supervision**: the model learns to predict part of the data from other parts of the data.
- This is the method used for training the base models of large language models such as BERT and GPT 1-4.
- Method:
  - Get data
  - Block a part of it
  - Ask the model to guess the missing data
  - Training by comparing the prediction to the actual data

Self-Supervised Learning: textual domain

Training Text	Ground Truth
The child didn’t want to go to ▮	The child didn’t want to go to bed

Self-Supervised Learning: visual domain

Training Image

Ground Truth

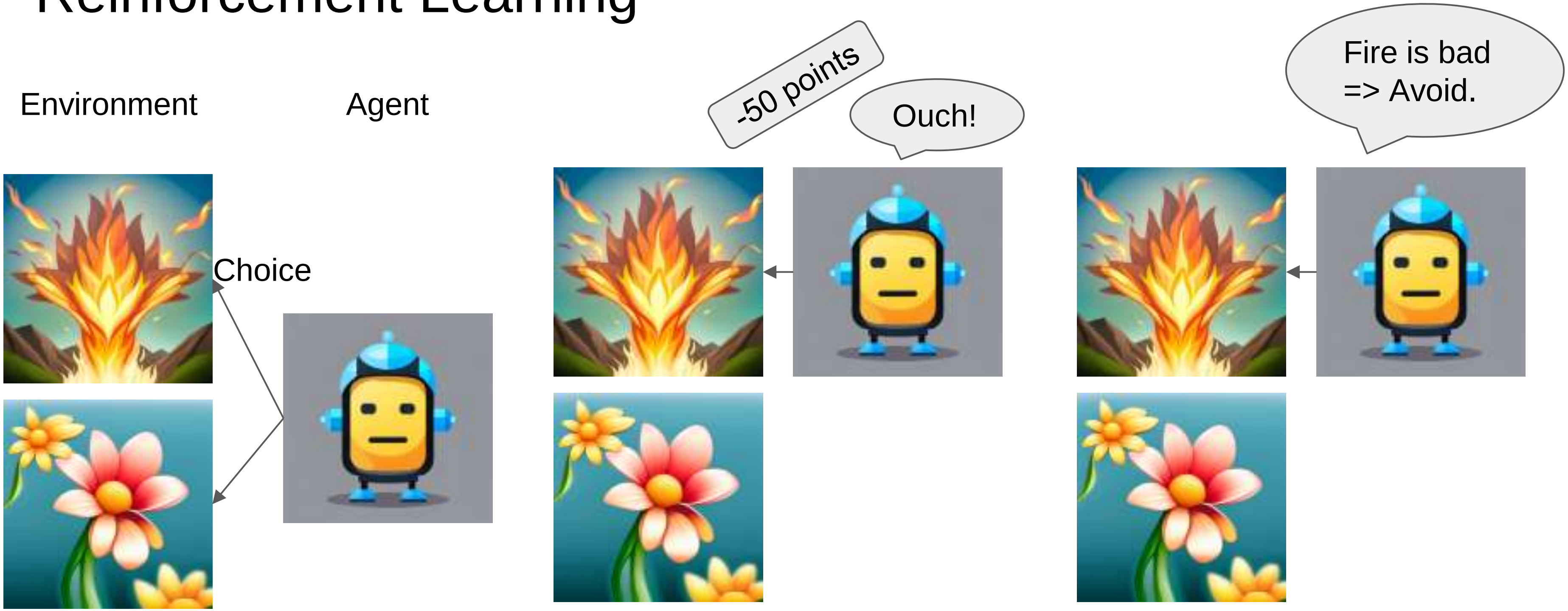
Image generated with DreamStudio

Reinforcement Learning

- A type of machine learning where an agent learns to make decisions by interacting with its environment and receiving feedback in the form of **rewards or penalties**
  - At each time step, the agent receives the current state of the environment, selects an action based on a policy, and performs the chosen action.
  - The environment then transitions to a new state and the agent receives a reward (positive or negative).
  - The goal of the agent is to learn a policy that maximizes the sum of rewards over time.

Reinforcement learning is particularly well-suited to problems where there is a clear reward signal and sequence of decisions, such as game playing and robotics.

Reinforcement Learning



1. Observe
2. Select Action Using Policy

3. Action
4. Get Reward or Penalty

5. Update Policy (learning step)
6. Iterate until an optimal policy is found

Images generated with [DreamStudio]