

Distance & Similarity

Distance function if and only if:

- $d(i, j) = 0$ if and only if $i = j$
- $d(i, j) = d(j, i)$
- $d(i, j) \leq d(i, k) + d(k, j)$ triangle inequality

Minkowski Distance

For x, y points in d dimensional real space

Given that $p \geq 1$:

$$L_p(x, y) = \left(\sum_{i=1}^d |x_i - y_i|^p \right)^{\frac{1}{p}}$$

$p = 1$: Manhattan Distance

$p = 2$: Euclidean Distance

Jaccard Similarity

$$JDist(x, y) = 1 - \frac{|x \cap y|}{|x \cup y|}$$

$x \cap y$ is the count of positions where both vectors have a 1 (intersection). How many

$x \cup y$ is the count of positions where at least one vector has a 1 (union).

Cosine Similarity

$s(x, y) = \cos(\theta)$

Where θ is the angle between x and y

Two proportional vector has $s()$ of 1 and two orthogonal is 0

Clustering

K-means

Given $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ our dataset, \mathbf{d} the euclidean distance, and \mathbf{k}

Find \mathbf{k} centers $\{\mu_1, \dots, \mu_k\}$ that minimize the **cost function**:

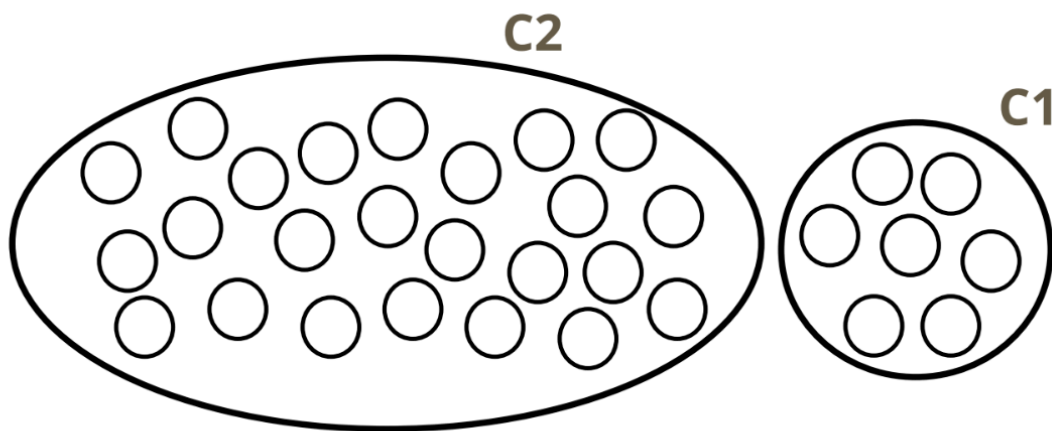
$$\sum_i^k \sum_{x \in C_i} d(x, \mu_i)^2$$

When $\mathbf{k}=1$ and $\mathbf{k}=n$ this is easy. Why?

When \mathbf{x}_i lives in more than 2 dimensions, this is a very difficult (**NP-hard**) problem

K Means - Lloyd's Algorithm

1. Randomly pick k centers
2. Assign each point to its closest center
3. Compute the new centers as the means of each cluster
4. Repeat 2 and 3 until convergence



No this cannot happen, since edges of the C2 cluster would be closer to centroid of C1

Kmeans always converge but not always optimal

K-means ++

Start with one random center from the data points.

For every other point x , compute $D(x)$, the distance between x and the **nearest** center already chosen.

Choose the next center from the remaining points. But **not randomly**—instead, pick a point with a **probability proportional to $D(x)^2$** .

- This means points **farther away** from existing centers are **more likely** to be chosen as new centers.
- The squared distance $D(x)^2$ exaggerates this effect, encouraging new centers to be spread out.

Repeat steps 2–3 until k centers are chosen.

Silhouette Scores

For each data point i :

- a_i = average distance from i to all other points **in the same cluster**
→ Measures **intra-cluster cohesion**
- b_i = lowest average distance from i to all points in **any other cluster** (i.e., the next nearest cluster)
→ Measures **inter-cluster separation**

Then the **Silhouette Score s_i** is:

$$s_i = (b_i - a_i) / (a_i, b_i)$$

If s_i is less than 0, possible misclassified

DBScan

ϵ and min_pts given:

1. Find the ϵ -neighborhood of each point
2. Label the point as core if it contains at least min_pts
3. For each core point, assign to the same cluster all core points in its neighborhood (crux of the algorithm)
4. Label points in its neighborhood that are not core as border
5. Label points as noise if they are neither core nor border
6. Assign border points to nearby clusters

Hierarchical Clustering

Single Link Distance

- Is the minimum of all pairwise distance between a point from one cluster and a point from the other cluster

Complete Link Distance

- Is the maximum of all pairwise distances between a point from one cluster and a point from the other cluster.

Average-Link Distance

- Is the average of all pairwise distances between a point from one cluster and a point from the other cluster.

Centroid Distance

- The distance between the centroids of clusters.

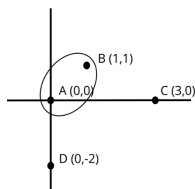
Ward's Distance

- Is the difference between the spread / variance of points in the merged cluster and the unmerged clusters.

Agglomerative Clustering Algorithm

- Let each point in the dataset be in its own cluster
- Compute the distance between all pairs of clusters
- Merge the two closest clusters
- Repeat 3 & 4 until all points are in the same cluster

Ex.



Distance Matrix

	A	B	C	D
A	0	$\sqrt{2}$	3	2
B	$\sqrt{2}$	0	$\sqrt{5}$	$\sqrt{10}$
C	3	$\sqrt{5}$	0	$\sqrt{13}$
D	2	$\sqrt{10}$	$\sqrt{13}$	0

Distance Matrix

	A & B	C	D
A & B	0	$\sqrt{5}$	2
C	$\sqrt{5}$	0	$\sqrt{13}$
D	2	$\sqrt{13}$	0

Soft Clustering

Goal: find the $P(S_j | X_i)$ distribution. Where S_j is species and X_i is the i th weight in the dataset

How to compute $P(S_j | X_i)$?

$$P(S_j | X_i) = \frac{P(X_i | S_j)P(S_j)}{P(X_i)}$$

$P(X | S)$ is the probability of a certain weight given that is species j

$P(S)$ is the prior probability of seeing a species S_j

How about $P(X_i)$?

- We combine the weight distribution of all species

$$P(X_i) = \sum_j P(S_j)P(X_i | S_j)$$

Therefore its called a Gaussian Mixture Model, where X comes from k mixture components

M Step:

Calculate $P(S_j | X_i)$ for all X_i by using mean, covariance and $P(S_j)$

E Step:

Compute and Update mean, covariance and $P(S_j)$ from the q -function derived from

$$\prod_i P(X_i) = \prod_i \sum_j P(S_j)P(X_i | S_j)$$

Not guaranteed to converge to a global maximum, not guarantee to converge in the finite number of steps

Need to calculate $3k$ parameters

Clustering Aggregation

Disagreement Distance

	P	C
\mathbf{x}_1	1	1
\mathbf{x}_2	1	2
\mathbf{x}_3	2	1
\mathbf{x}_4	3	3
\mathbf{x}_5	3	4

Step-by-step pair comparisons:

1. (x_1, x_2)

- P: same (1,1)
- C: different (1,2) → **disagreement**

○ P: different (1,3)

- C: different (1,3) → **no disagreement**

.....

4. (x_4, x_5)

2. (x_1, x_3)

- P: different (1,2)
- C: same (1,1) → **disagreement**

○ P: same (3,3)

- C: different (3,4) → **disagreement**

3. (x_1, x_4)

Aggregate Clustering

Goal: from a set of clusterings, generate a clustering that minimizes:

$$\sum_{i=1}^m D(C^*, C_i)$$

Singular Value Decomposition

Linear Algebra Review:

A set of n vectors $\{v_1, v_2, \dots, v_n\}$ in an **n -dimensional space** is **linearly independent if and only if**:

$$\det(A) \neq 0$$

Where:

- $A = [v_1 \ v_2 \ \dots \ v_n]$ is an $n \times n$ matrix with each vector as a column.

Definition: Rank

The **rank** of a matrix A is:

- The **dimension of the vector space** spanned by its **columns** (i.e., the **column space** of A).
- Equivalently, it is the **maximum number of linearly independent columns or rows** of A .

Definition: Full Rank

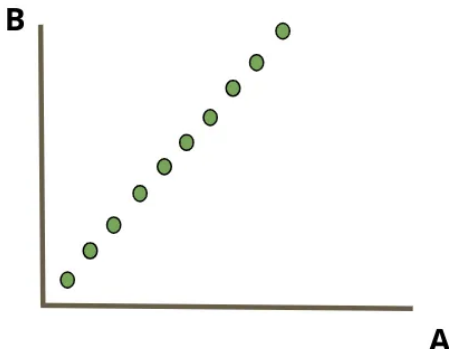
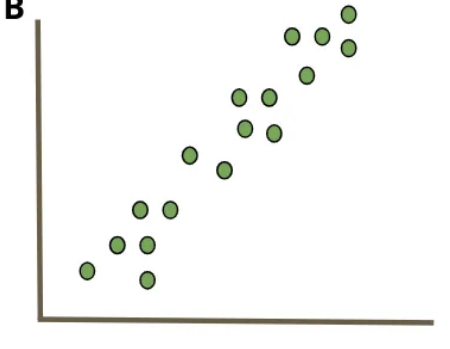

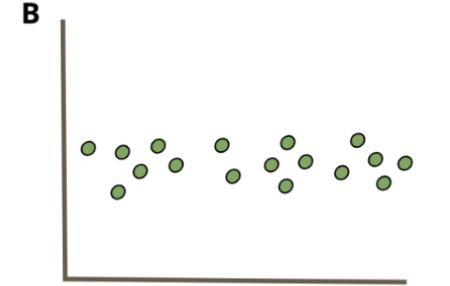
A matrix $A \in \mathbb{R}^{m \times n}$ is **full-rank** if:

$$\text{rank}(A) = \min(m, n)$$

- For a **square matrix** ($m=n$): full rank means it is **invertible**.
- For a **rectangular matrix**: full rank means its columns (if $m > n$) or rows (if $n > m$) are linearly independent.

Frobenius Distance

$$d_F(A, B) = \|A - B\|_F = \sqrt{\sum_{i,j} (a_{ij} - b_{ij})^2}$$

	
Linear Related, data is 2 dimension rank 1	Linear Related, data is full rank / 2D
	
Rank 1 dimension 1	Rank 2 dimension 2

if not a line then rank 2, most the time its dimension 2 except for when B is fixed

[Singular Value Decomposition - by Lance Galletti](#)

The Singular Value Decomposition of a rank-r matrix A has the form:

$$A = U\Sigma V^T$$

U is $n \times r$

The columns of U are orthogonal & unit length ($U^T U = I$)

ith singular vector in E represents the direction of the ith most variance

V is $m \times r$

The columns of V are orthogonal & unit length ($V^T V = I$)

Use case:

- Low rank approximation
 - Same dimension different rank
- Dimension reduction
- Anomaly Detection

Classification and KNN

What makes a good predictor?

- We want features that are related to the target but not to each other.
- Correlation tests
 - Pearson coefficient
 - Spearman coefficient

How do we know we've done well at classification

- Testing without cheating. Learning not memorizing.
 - Split up our data into a training set and a separate testing set
 - Use the training set to find patterns and create a model
 - Use the testing set to evaluate the model on data it has not seen before

KNN

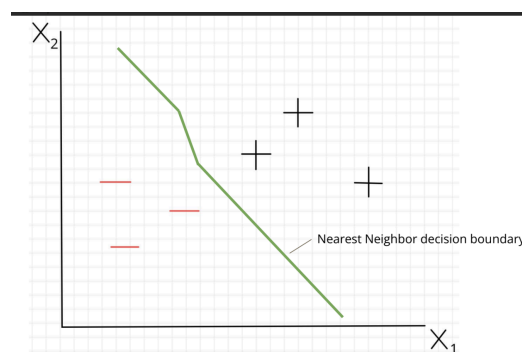
- Compute distances of unseen record to all training record
- Identify the K nearest neighbors
- Aggregate the labels of these k neighbors to predict the unseen record class (ex. Majority rule)

If k is too small

- Sensitive to noise points, not generalize well

If k is too big

- Might include points from other classes



The decision boundary for K=1 always in the middle

Decision Trees

Hunt's Algorithm

Recursive Algorithm

- Repeatedly split the dataset based on attributes
- Find the attribute that best splits the data

Base cases:

- IF Split and all data points in the same class
 - Great! Predict that class
- IF Split and no data points
 - No problem! Predict a reasonable default

GINI Index

$$GINI(t) = 1 - \sum_j p(j|t)^2$$

NO	1
YES	7

$$p(\text{NO} | t) = \frac{1}{8}$$

$$p(\text{YES} | t) = \frac{7}{8}$$

$$GINI(t) = 1 - \frac{1}{64} - \frac{49}{64} = \frac{14}{64}$$

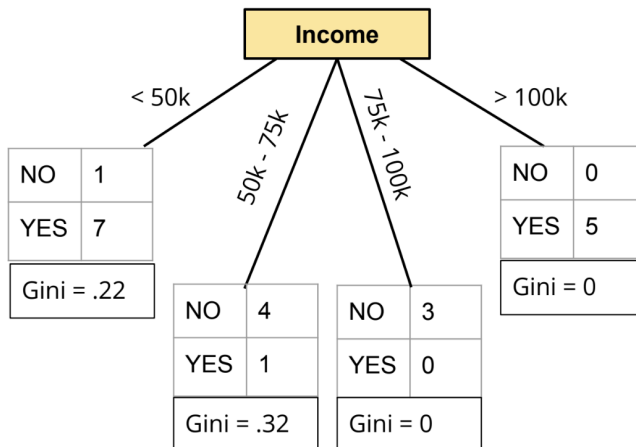
NO	4
YES	3

$$p(\text{NO} | t) = \frac{4}{7}$$

$$p(\text{YES} | t) = \frac{3}{7}$$

$$GINI(t) = 1 - \frac{16}{49} - \frac{9}{49} = \frac{24}{49}$$

For multi class



$$GINI_{split} = \sum_{t=1}^k \frac{n_t}{n} GINI(t)$$

$$n = 21$$

$$GINI_{split} = .22 * \frac{8}{21} + .32 * \frac{5}{21} + 0 * \frac{3}{21} + 0 * \frac{5}{21} = .16$$

Model Evaluation

Confusion Matrix

	Predicted Class		
		Class = Yes	Class = No
	Class = Yes	a (TP)	b (FN)
	Class = No	c (FP)	d (TN)

Accuracy = $(a + d) / (a + b + c + d)$

Precision = $a / (a + c)$

Recall = $a / (a + b)$

F-measure = $2RP / (R + P)$

Cross Validation

- Hyperparameter tuning
- Split the training set into k folds; iteratively train on $k - 1$ folds, validate on the held-out fold, average the scores, pick the hyper-parameter setting with the best average.

Ensemble Methods

We poll different classifiers and take the class that the majority agrees on

Bagging

Boosting

1. **“Focus on what’s still wrong” loop**
 - o **Round 1:** Treat every training example equally, train base learner #1.
 - o **Check mistakes:** See which examples it got wrong.
Re-weight data: Give *more* weight (higher sampling probability) to those hard examples and *less* weight to the ones already handled.
 - o **Round 2:** Draw a new training set using these updated weights, train base learner #2.
 - o **Repeat** for a fixed number of rounds (or until accuracy stops improving)

2. Final model = weighted vote

Each learner gets a say that is proportional to how well it performed when it was trained. The ensemble prediction is the sign/arg-max of this weighted vote.

Why the classifiers are (almost) independent

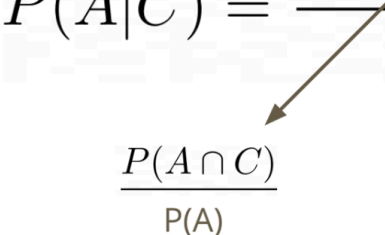
Every round sees a **different effective dataset** because the sampling distribution keeps changing.

- Early learners train on a near-uniform view of the data.
- Later learners are forced to stare at the stubborn cases the earlier ones still miss.
- This change in perspective reduces correlation between their errors, which makes the final vote stronger than any single learner.

Summary

Bagging samples randomly with replacement while **Boosting** samples randomly but the weights of data points varies depending on how successfully these points were predicted. **Bagging** averages the predictions of all models while **Boosting** takes a weighted average.

Naive Bayes

$$P(A|C) = \frac{P(C|A)P(A)}{P(C)}$$

$$\frac{P(A \cap C)}{P(A)}$$

Given a_1, a_2 to a_m data how do we know which class it is from, we find:

$$P(C|A_1 \cap A_2 \cap \dots \cap A_n) = \frac{P(A_1 \cap A_2 \cap \dots \cap A_n|C)P(C)}{P(A_1 \cap A_2 \cap \dots \cap A_n)}$$

We assume a_1 to a_m attributes are independent:

$$P(A_1 A_2 \dots A_n | C) = P(A_1 | C) P(A_2 | C) \dots P(A_n | C)$$

Refund	Marital Status	Income	Class
Yes	Single	125k	No
No	Married	100k	No
No	Single	70k	No
Yes	Married	120k	No
No	Divorced	90k	Yes
No	Married	60k	No
Yes	Divorced	220k	No
No	Single	85k	Yes
No	Married	75k	No
No	Single	90k	Yes

Ex.

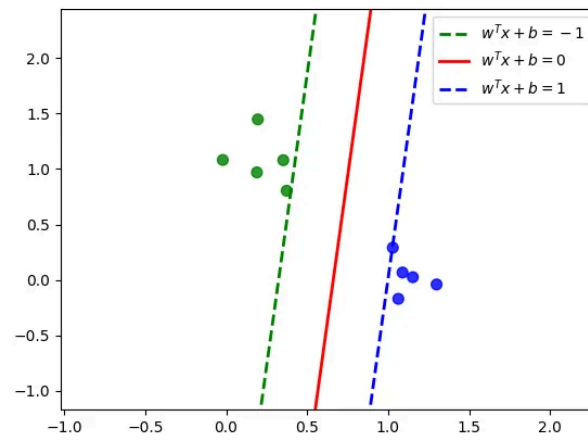
Test Record:

X = (Refund = No, Married, Income = 120k)

- $P(X | \text{No}) = P(\text{Refund} = \text{No} | \text{No})$
 $P(\text{Married} | \text{No}) P(\text{Income} = 120k | \text{No}) =$
 $4/7 * 4/7 * .0072 = .0024$
- $P(X | \text{Yes}) = P(\text{Refund} = \text{No} | \text{Yes})$
 $P(\text{Married} | \text{Yes}) P(\text{Income} = 120k | \text{Yes}) =$
 $1 * 0 * 1.2 * 10^{-9} = 0$

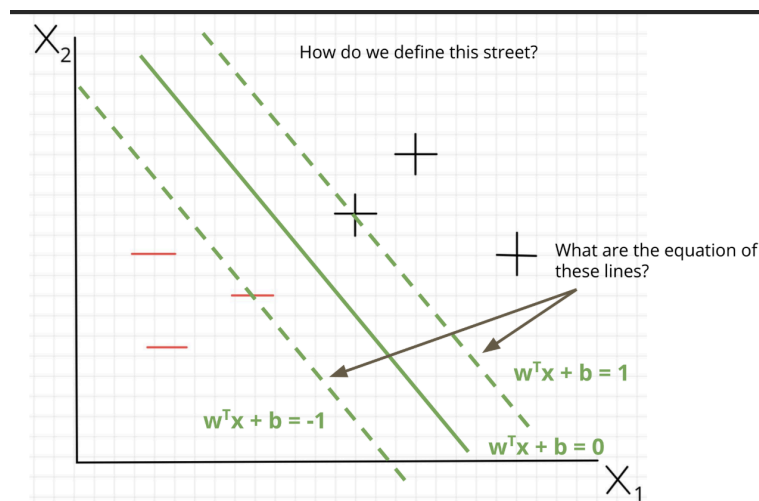
Since $P(X | \text{No})P(\text{No}) > P(X | \text{Yes})P(\text{Yes})$
=> predict No

Support Vector Machines



In this example we have two classes (blue = +1 and green = -1). The line in red is the decision boundary — to classify an unknown point u using the above SVM means:

- $w^T u + b \geq 0$ THEN blue
- $w^T u + b < 0$ THEN green



$$\text{width} = \frac{2}{\|w\|}$$

So to find the widest street, it means we have to find the smallest w

Notice that multiplying w and b by the same constant c doesn't change the decision boundary but does change the width of the street. If:

- $0 < c < 1$ the width will expand

- $c > 1$ the width will retract

Perceptron Algorithm

$$w_{new} = w_{old} + y_i * lr * x_i$$

$$b_{new} = b_{old} + y_i * lr$$

[Support Vector Machines From Scratch | by Lance Galletti | Medium](#)

RBF Kernel ??

Linear Regression

Assumptions

- Our data was generated by a linear function plus some noise:

$$\vec{y} = h_X(\beta) + \vec{\epsilon}$$

Where \mathbf{h} is linear in a parameter β .

Where ϵ_i are independent $\mathbf{N}(\mathbf{0}, \sigma^2)$ distribution.

Noise needs to be normally distributed

Variance of the noise is constant

The relationship has to be linear

Example linear function:

Which functions below are linear in β ?

$$h(\beta) = \beta_1 x \quad \checkmark$$

$$h(\beta) = \beta_0 + \beta_1 x \quad \checkmark$$

$$h(\beta) = \beta_0 + \beta_1 x + \beta_2 x^2 \quad \checkmark$$

$$h(\beta) = \beta_1 \log(x) + \beta_2 x^2 \quad \checkmark$$

$$h(\beta) = \beta_0 + \beta_1 x + \beta_1^2 x \quad \times$$

Goal:

- Minimize the cost function
 - Least squares

$$\begin{aligned}
\beta_{LS} &= \arg \min_{\beta} \sum_i d(h_{\beta}(x_i), y_i) \\
&= \arg \min_{\beta} \|\vec{y} - h_{\beta}(X)\|_2^2 \\
&= \arg \min_{\beta} \|y - X\beta\|_2^2
\end{aligned}$$

$$\beta_{LS} = (X^T X)^{-1} X^T y$$

pseudo inverse of X

Maximum Likelihood

Another way to define this problem is in terms of probability.

Define $P(Y | h)$ as the probability of observing Y given that it was sampled from h .

Goal: Find h that maximizes the probability of having observed our data.

Maximize $L(h) = P(Y | h)$

Since $\varepsilon \sim N(0, \sigma^2)$ and $Y = X\beta + \varepsilon$ then $Y \sim N(X\beta, \sigma^2)$.

NLL – Negative Log-Likelihood (Gaussian case here)

- **What it measures:** How *unlikely* the observed data are under a probabilistic model of the form

$$y_i \sim \mathcal{N}(\hat{y}_i, \sigma^2)$$

(with σ^2 the noise variance). The function you pasted uses

$$\text{NLL}(w, b) = \frac{1}{2} \left[\frac{(y - \hat{y})^2}{\sigma^2} + \ln(2\pi\sigma^2) \right]$$

Evaluation

y_i is the “true” value from our data set (i.e. $\mathbf{x}_i \beta + \epsilon_i$)

\hat{y}_i is the estimate of y_i from our model (i.e. $\mathbf{x}_i \beta_{LS}$)

\bar{y} is the sample mean all y_i

$y_i - \hat{y}_i$ are the estimates of ϵ_i and are referred to as residuals

QQ plot ??

$$TSS = \sum_i^n (y_i - \bar{y})^2$$

← This is a measure of the spread of y_i around the mean of y

$$ESS = \sum_i^n (\hat{y}_i - \bar{y})^2$$

← This is a measure of the spread of our model's estimates of y_i around the mean of y

$R^2 = ESS / TSS$, measures the fraction of variance that is explained by \hat{y}

$$TSS = \sum_i^n (y_i - \bar{y})^2 \quad R^2 = \frac{ESS}{TSS} = 1 - \frac{RSS}{TSS}$$

$$RSS = \sum_i^n (y_i - \hat{y}_i)^2$$

$$ESS = \sum_i^n (\hat{y}_i - \bar{y})^2$$

↑ This equality only holds for a linear model that minimizes the RSS

Variance formula:

- Cannot be negative
- Can be greater than 1
- Only use when \hat{y} is a linear model (minimizing RSS) and is centered around \bar{Y} (i.e. not omitting an intercept for example)

Residual-based R^2 formula

- Can be negative
- Cannot be greater than 1
- Adapts to any model. A linear model that passes through all points will have an R^2 of 1

Hypothesis Testing

Goal: answer how likely are we to observe this? Is there enough evidence that we can reject assumption


T Test

Ex. We test whether each coefficient β_i is **significantly different from 0**.

 Null Hypothesis (H_0):

$$H_0: \beta_i = 0$$

→ There is **no** relationship between x_i and the dependent variable y .

 Alternative Hypothesis (H_1):


$$H_1: \beta_i \neq 0$$

→ There **is** a relationship between x_i and y .

 t-Statistic:

$$t = \beta^i / \text{std err}(\beta^i)$$

This measures **how many standard errors away** the estimated coefficient is from zero.

 p-Value:

This is the probability of observing a t-statistic **at least as extreme** as the one computed, **if the null hypothesis were true**.

 How to Interpret (with typical 5% significance level):

- If $p < 0.05$, reject H_0 : the coefficient is **statistically significant**.
- If $p \geq 0.05$, fail to reject H_0 : **not enough evidence** to say the coefficient is significant.

Confidence Intervals

Goal: for a given confidence level (let's say 90%), construct an interval around an estimate such that, if the estimation process were repeated indefinitely, the interval would contain the true value (that the estimate is estimating) 90% of the time.

Let's say if you want to be 90% confident, construct a \pm margin of error around your estimate so you can be right 90% of the time. Ex. $\hat{\beta} = 6.9 \pm 0.2$

A (frequentist) "p% confidence interval" is *any* rule that, in the long run, contains the true parameter with probability $p/100$

Quick-Reference Notes on Confidence Intervals		
Topic	Key points	
What a CI is	A $p\%$ confidence interval (CI) is any rule that, when repeatedly applied to fresh random samples from the same population, contains the true parameter with probability $p/100$.	
Frequentist interpretation	The confidence level refers to the procedure's long-run success rate , not to the probability that a <i>single realised interval</i> contains the parameter (for that single interval the probability is 0 or 1).	
Basic formula for a mean (known σ)	$CI_p : \bar{y} \pm z_{1-\alpha/2} \frac{\sigma}{\sqrt{n}}$ where $z_{1-\alpha/2}$ is the critical value from the standard normal (e.g. 1.96 for 95 %).	
Standard error (SE) vs s.d.	SE measures the sampling variability of the estimator (e.g. σ / \sqrt{n}); <i>standard deviation</i> describes spread of raw data. Narrower CI \Leftrightarrow smaller SE \Leftrightarrow more precision , not "more confidence."	
Effect of confidence level	Higher confidence (e.g. 99 % versus 95 %) \Rightarrow larger critical value \Rightarrow wider interval. So the statement "intervals shrink as confidence level rises" is false.	
Coverage in repeated sampling	In the long run, ~99 of 100 independently constructed 99 % CIs will cover the true parameter.	

Z-values

- These are the number of standard deviation away from the mean of a standard normal distribution

We use z values to find the std from the standard normal distribution to contain a specific % of values

To find the .95 z-value (the value z such that 95% of the observations lie within z standard deviations of the mean ($\mu \pm z * \sigma$)) you need to solve:

Ex.
$$\int_{-z}^z \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx = .95$$

Example calculating the confidence interval

Goal:

Estimate the population mean μ using the sample mean \bar{y} , and give a range where μ is likely to fall — with 95% confidence.

Given:

- $Y_i \sim \mathcal{N}(5, 25)$: Each observation has mean 5, variance 25 (i.e., std dev = 5).
- $n = 100$: Number of observations
- $\mu_{LS} = \bar{y}$: Our best estimate of μ is the sample mean.
- $\epsilon \sim \mathcal{N}(0, 25)$: The noise is normally distributed.

Step-by-Step Calculation:

1. Standard Error (SE):

$$SE(\mu_{LS}) = \frac{\sigma_{\epsilon}}{\sqrt{n}} = \frac{5}{\sqrt{100}} = 0.5$$

This measures the variability of the sample mean \bar{y} as an estimator of μ .

2. Z-value for 95% CI:

- From the standard normal table:

$$Z_{0.975} = 1.96$$

- This captures the middle 95% of a normal distribution.

3. Confidence Interval Formula:

$$\begin{aligned} CI_{.95} &= [\bar{y} - 1.96 \cdot SE, \bar{y} + 1.96 \cdot SE] \\ &= [\bar{y} - 1.96 \cdot 0.5, \bar{y} + 1.96 \cdot 0.5] = [\bar{y} - 0.98, \bar{y} + 0.98] \end{aligned}$$

Logistic Regression