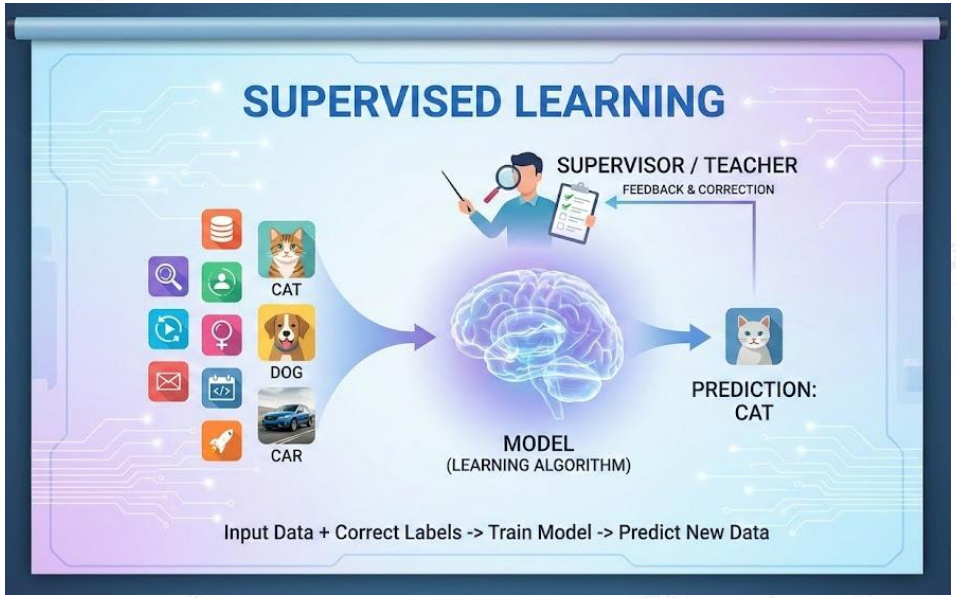
Supervised Learning



CLASSIFICATION, REGRESSION, ALGORITHMS & EVALUATION

NIELIT ROPAR DEEMED TO BE UNIVERSITY

Learning Agenda

- **Part 1:** Introduction to Supervised Learning
- **Part 2:** K-Nearest Neighbors (KNN)
- **Part 3:** Linear Regression
- **Part 4:** Logistic Regression
- **Part 5:** Support Vector Machines (SVM)
- **Part 6:** Evaluation (Regression)
- **Part 7:** Evaluation (Classification)
- **Part 8:** Summary & Q&A

A program learns from experience E, for a task T, with performance measure P, when its performance on T improves after exposure to E.

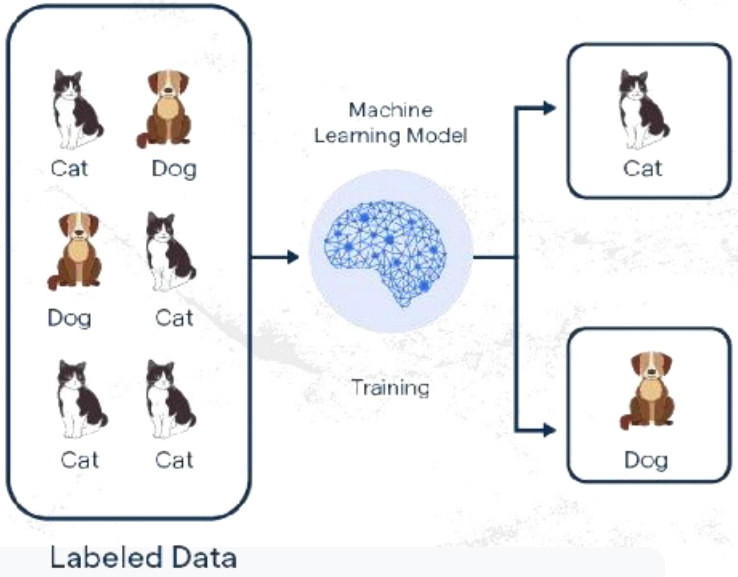
— Tom M. Mitchell

Part 1: Introduction

Introduction

What is Supervised Learning?

- A type of machine learning where the machine learns from **labeled data**.
- The model is provided with input data (X) and the correct output (Y).
- **Goal:** Learn a mapping function that predicts Y for new X.



ANALOGY: TEACHER & STUDENT

The algorithm is the student. The dataset is the textbook with answers key. The "Supervisor" is the answer key correcting the student during practice.

Introduction

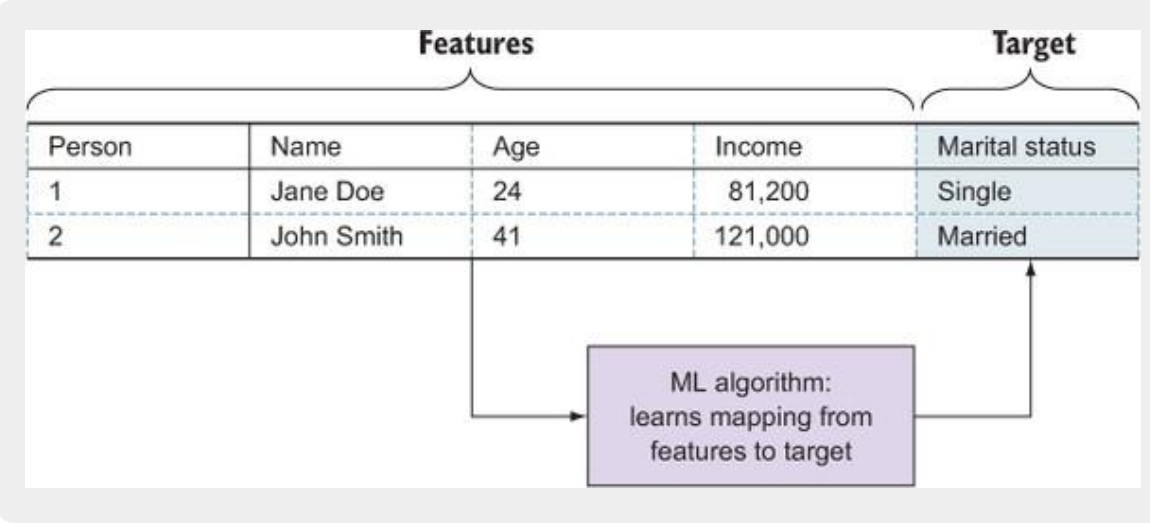
Key Components

1. Features (Input)

Variables used to make predictions (often denoted as X).

2. Target (Output)

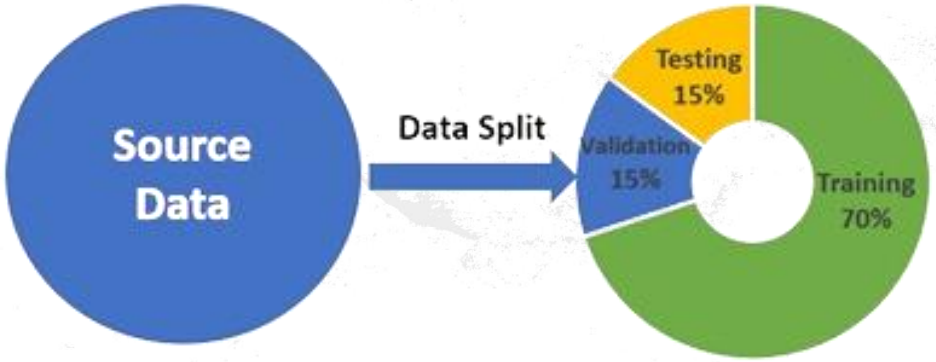
The variable we want to predict (often denoted as Y).



Introduction

The Learning Workflow

- **Data Collection:** Gather labeled historical data.
- **Splitting:** Divide into Training Set (80%) and Test Set (20%).
- **Training:** Model finds patterns in the Training Set.
- **Testing:** Model predicts on Test Set to verify accuracy.

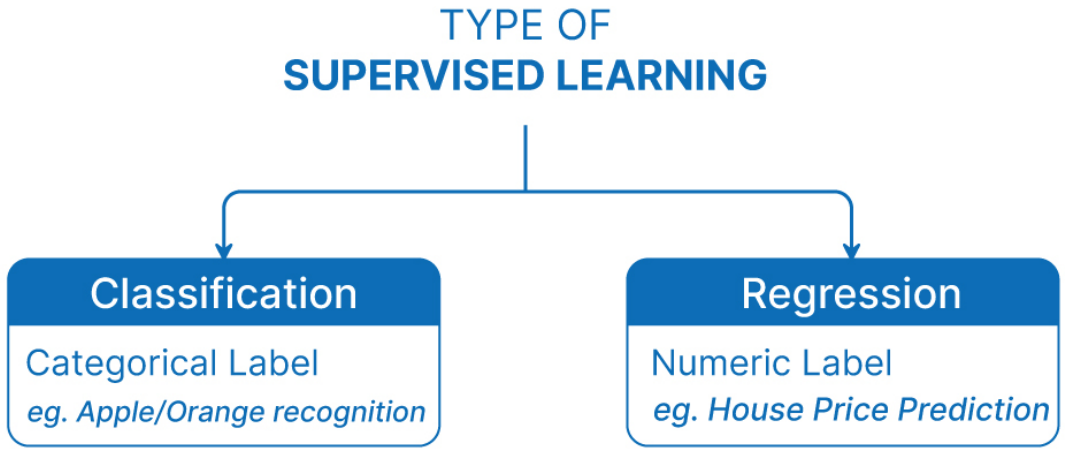


WHY SPLIT?

To ensure the model works on new data (generalization) and isn't just memorizing the answers (overfitting).

Introduction

Two Main Types



Classification

Predicts a **Category** or Class.

(Discrete Output)

Regression

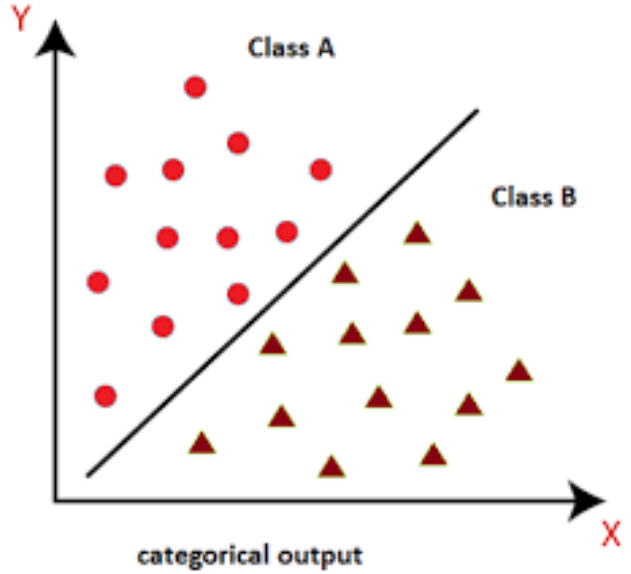
Predicts a **Numeric Value**.

(Continuous Output)

Introduction

1. Classification

The output variable is a category, such as "Red" or "Blue", "Spam" or "Not Spam".



- **Binary Classification:** Only two classes (0 or 1).
- **Multi-class Classification:** More than two classes.

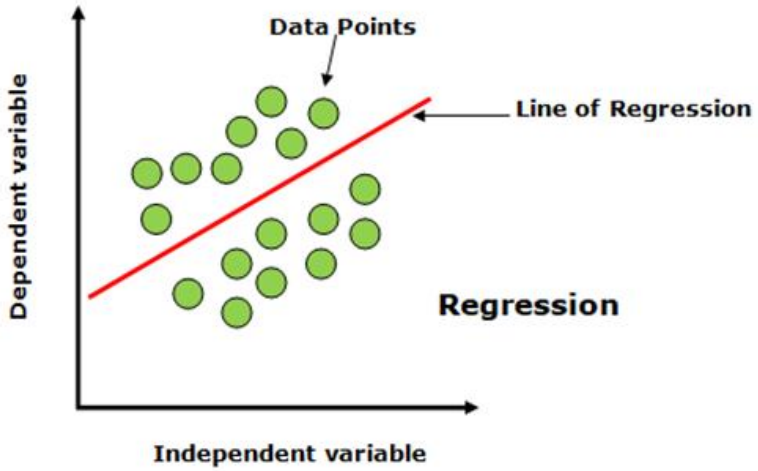
REAL-WORLD EXAMPLES

- Is this email Spam?
- Is the tumor malignant or benign?
- Is this transaction fraudulent?

Introduction

2. Regression

The output variable is a real/continuous value.



- Answers "How much?" or "How many?".
- Based on correlations between variables.

REAL-WORLD EXAMPLES

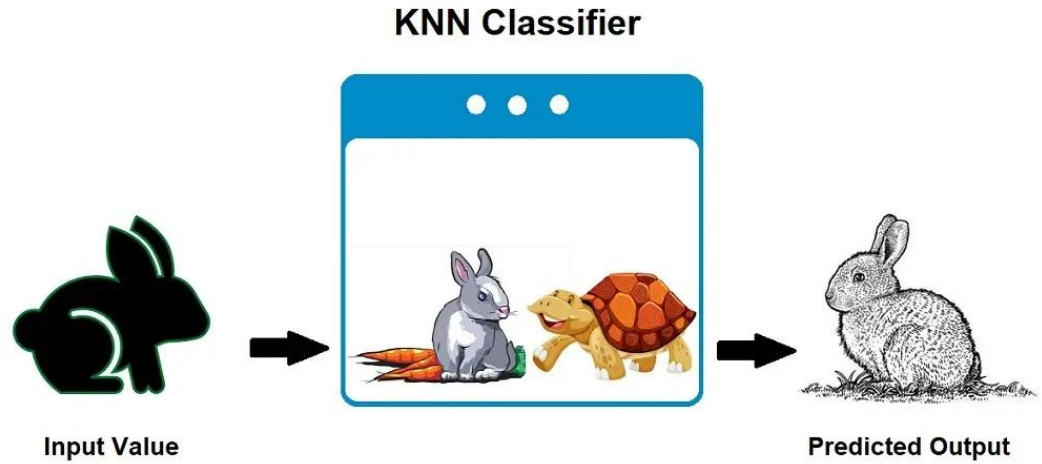
- Predicting house prices ($).
- Forecasting temperature (°C).
- Estimating stock market value.

Part 2:
K-Nearest Neighbors

KNN

K-Nearest Neighbors (KNN)

A simple, instance-based learning algorithm used for both classification and regression.



THE INTUITION

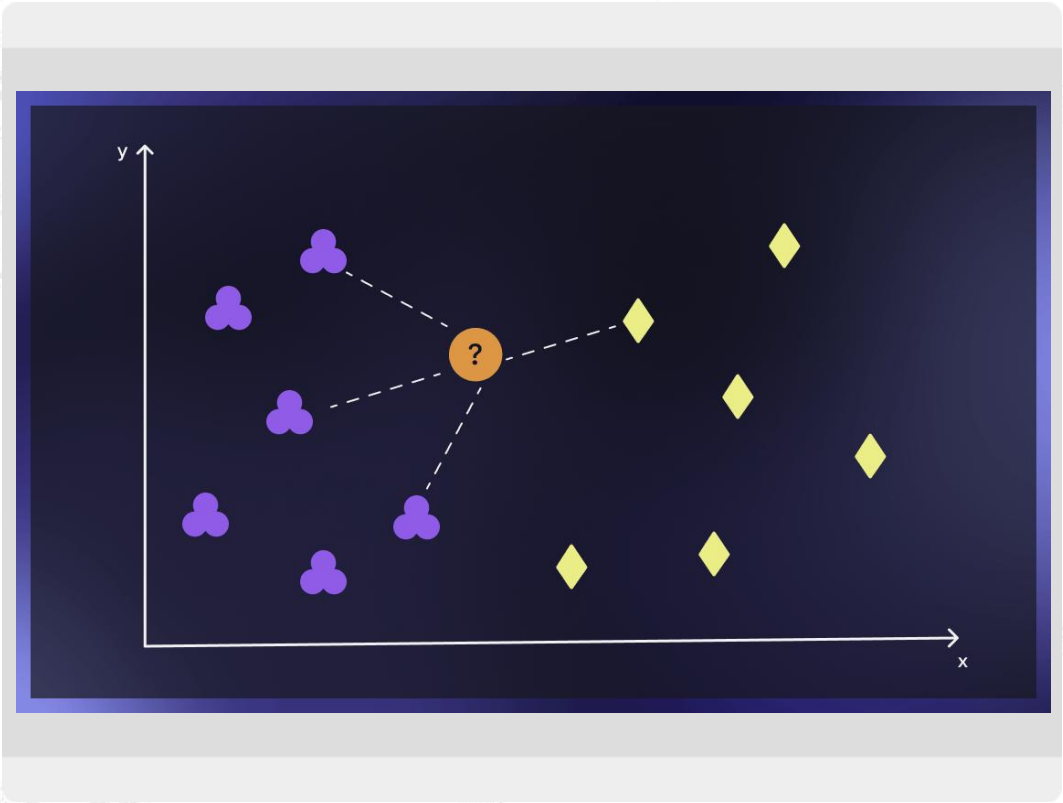
"Tell me who your friends are, and I will tell you who you are."

If you live in a neighborhood of rich people, you are likely rich. If most of your neighbors are red dots, you are likely a red dot.

KNN

Step-by-Step Algorithm

- Select the number **K** of neighbors.
- Calculate the **distance** between the new data point and all training points.
- Select the K nearest neighbors.
- **Vote:** Assign the most common class (Classification) or average value (Regression).



Calculating Distance: Euclidean

The most common metric. Straight-line distance between two points.

$$d = \sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2]}$$

- Used when data is continuous.
- Standard "Pythagorean" geometry.

Calculating Distance: Manhattan

Distance measured along axes at right angles (Taxicab geometry).

$$d = |x_2 - x_1| + |y_2 - y_1|$$

WHEN TO USE?

Better for high-dimensional data or grid-like structures (like city blocks).

Choosing the 'K' Value

The choice of K impacts model performance significantly.

Small K (e.g., K=1)

Sensitive to noise/outliers. Can lead to **Overfitting**.

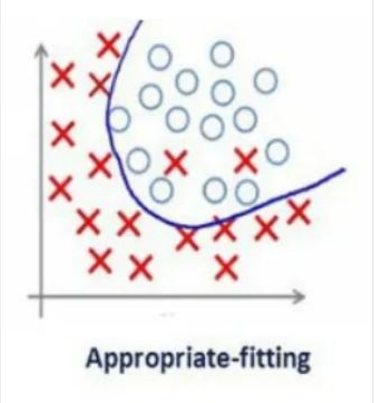
Large K (e.g., K=100)

Smoothes out boundaries too much. Can lead to **Underfitting**.

Rule of Thumb: K = √(N), where N is total samples.

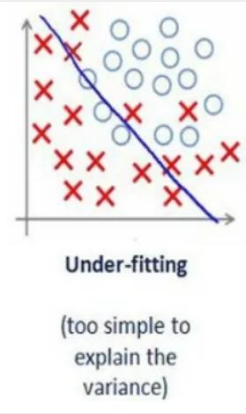
KNN

Overfitting vs Underfitting



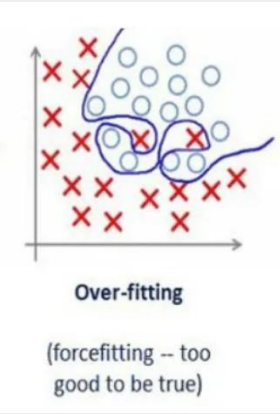
Good Fit

Captures pattern, ignores noise.



Underfitting

Underfitting happens when a model is too simple to capture the underlying patterns in the data. This results in poor performance on both training and validation data.



Overfitting

Overfitting occurs when a model learns not only the underlying patterns in the training data but also the noise and random fluctuations. This results in a model that performs well on training data but poorly on unseen data.

KNN: Summary

Pros

- Simple to understand and implement.
- No training period (Lazy learner).
- Works well for multi-class problems.

Cons

- Computationally expensive at prediction time.
- Sensitive to irrelevant features.
- Requires feature scaling (units matter).

Colab Notebook: https://colab.research.google.com/drive/1etg7f_aF4KuSiN6sCk4Vl92gmf2sgdmO?usp=sharing

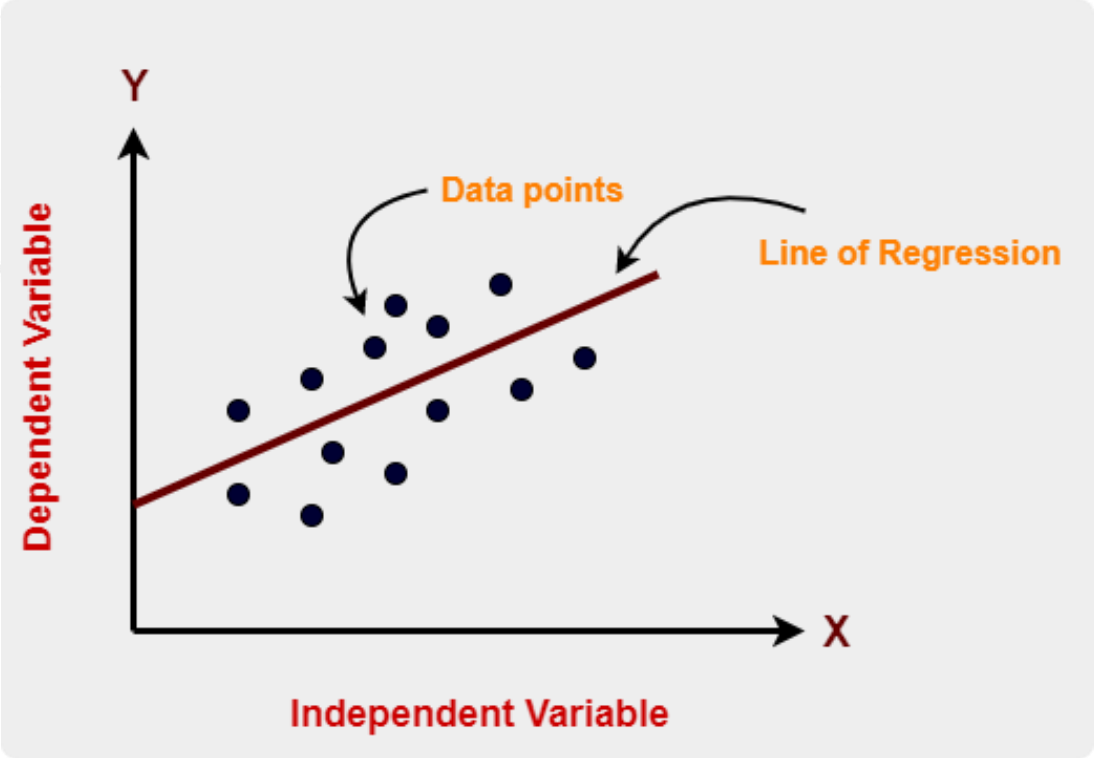
Part 3:
Linear Regression

Linear Regression

The Concept

A linear approach to modeling the relationship between a scalar response (y) and one or more explanatory variables (x).

- Finds the **"Line of Best Fit"**.
- Predicts a continuous value.



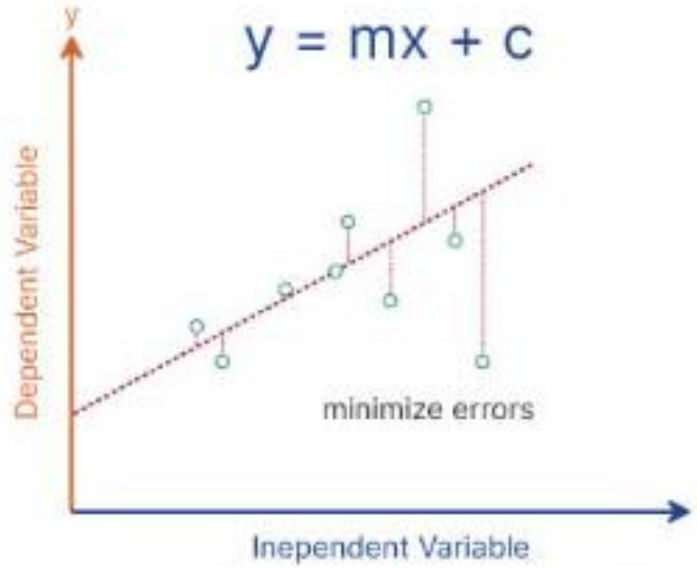
Linear Regression

The Equation

Mathematical representation of the line:

$$y = mx + c$$

- **y:** Target Variable (Prediction).
- **m:** Slope (Weight/Coefficient).
- **x:** Feature (Input).
- **c:** Intercept (Bias).



EXAMPLE

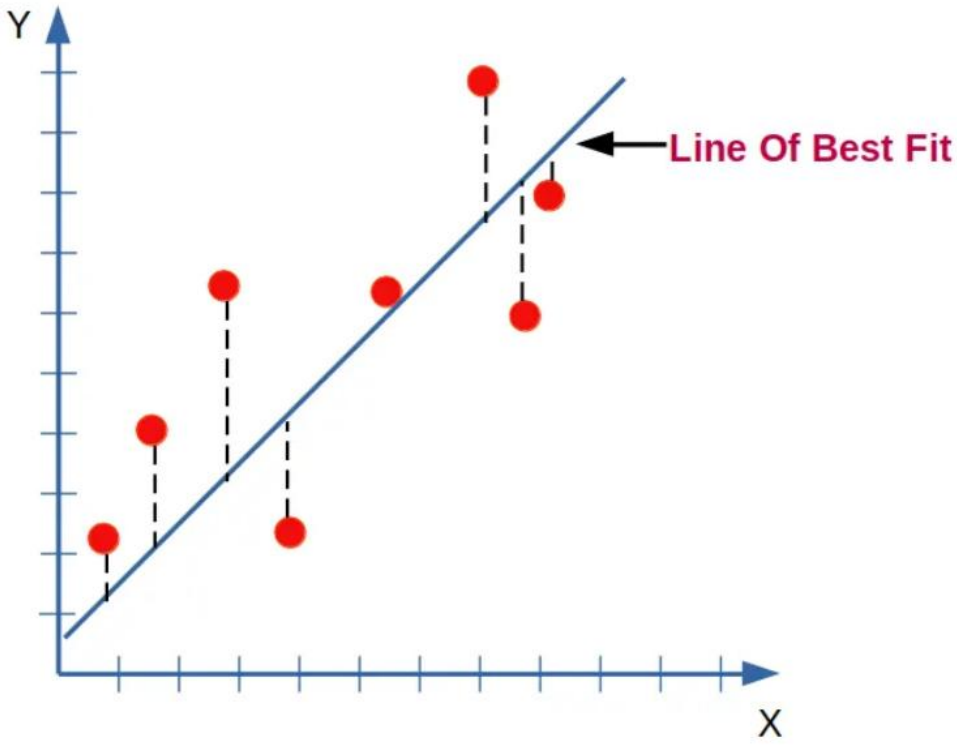
Price = (5000 * Size_sqft) + 20000

Linear Regression

What is a "Best Fit"?

The line that minimizes the error between predicted points and actual points.

- **Residual (Error):** The vertical distance between an actual data point and the line.
- Error = Actual_y - Predicted_y



> Our goal is to reduce the total sum of these errors.

Linear Regression

Cost Function (SSE)

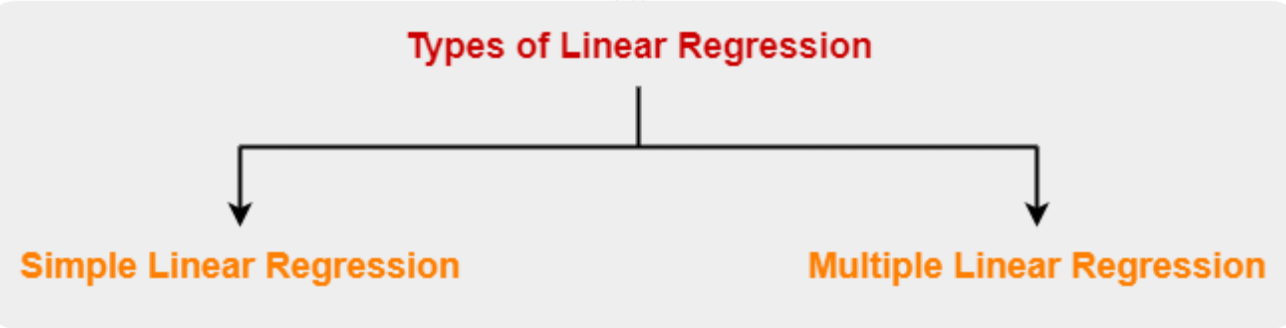
To find the best line, we calculate the **Sum of Squared Errors (SSE)**.

$$SSE = \Sigma (y\_actual - y\_predicted)^2$$

- We square the errors to remove negative signs and penalize larger errors.
- Algorithms like **Gradient Descent** are used to minimize this value.

Linear Regression

Types of Linear Regression



- In simple linear regression, the dependent variable depends only on a single independent variable.
- In multiple linear regression, the dependent variable depends on **more than one independent variables.**

SIMPLE LINEAR REGRESSION

One input variable (x) predicting one output (y).

Example: Height predicting Weight.

MULTIPLE LINEAR REGRESSION

Multiple input variables (x_1, x_2, x_3...) predicting one output (y).

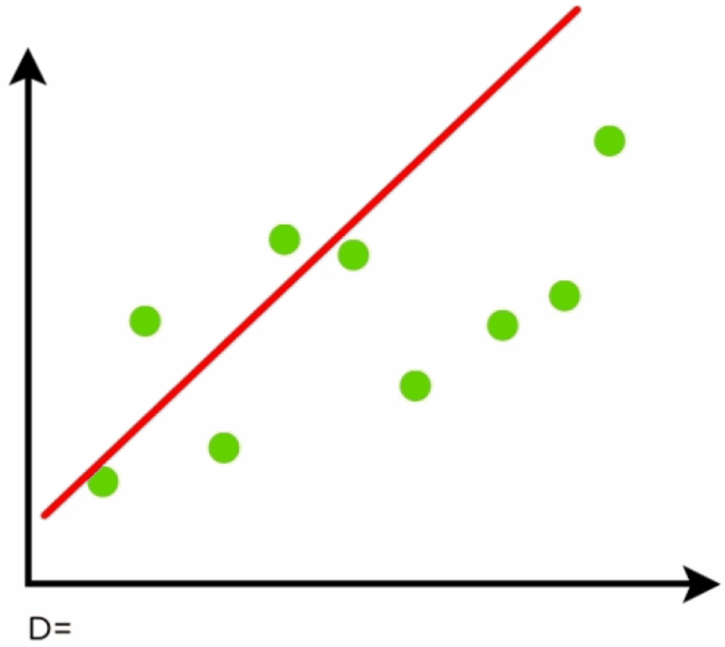
Example: House Price predicted by Size, Location, and Age.

Linear Regression

Assumptions

For Linear Regression to work well, data must meet certain criteria:

- **Linearity:** The relationship between X and Y is linear.
- **Independence:** Observations are independent.
- **Homoscedasticity:** Constant variance of errors.
- **Normality:** Errors are normally distributed.



Colab Notebook:
https://colab.research.google.com/drive/1ha_uM9PNeP8ZAPreSCP7gcb0nM8HpjL0?usp=sharing

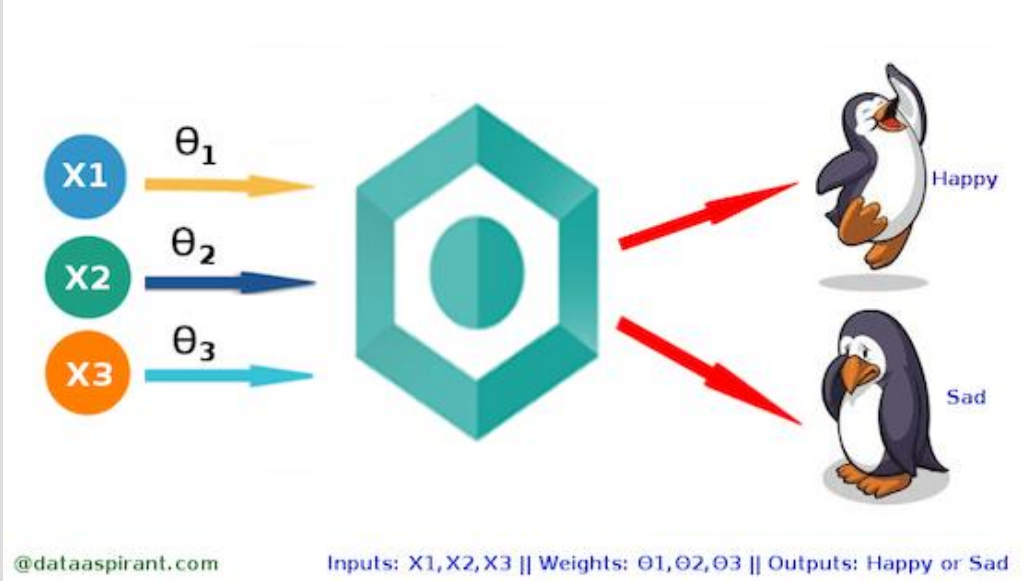
Part 4:
Logistic Regression

Logistic Regression

Why not Linear Regression?

We cannot use a straight line for binary classification (0 or 1).

- Linear lines can go towards infinity (> 1 or < 0).
- Probabilities must stay between 0 and 1.
- We need a function that squashes values into this range.



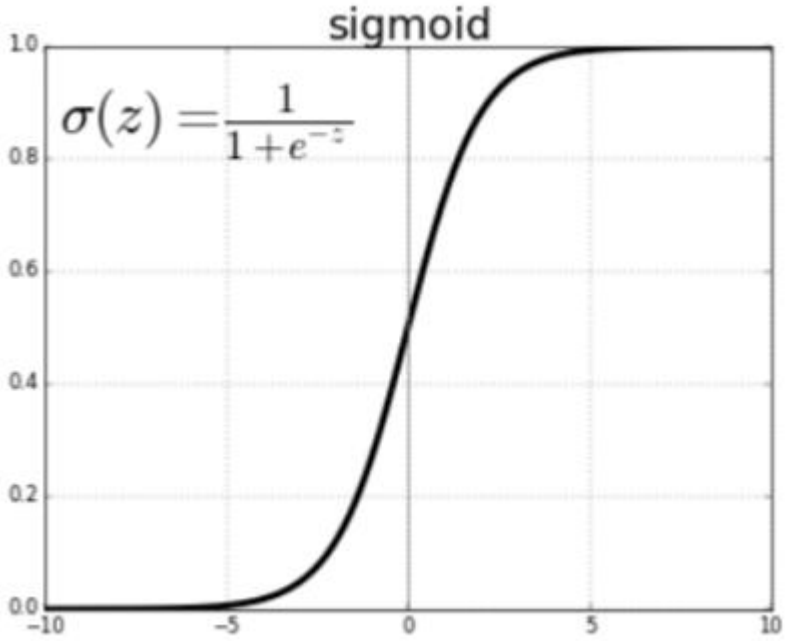
Logistic Regression

The Sigmoid Activation

We transform the linear output using the **Sigmoid** function.

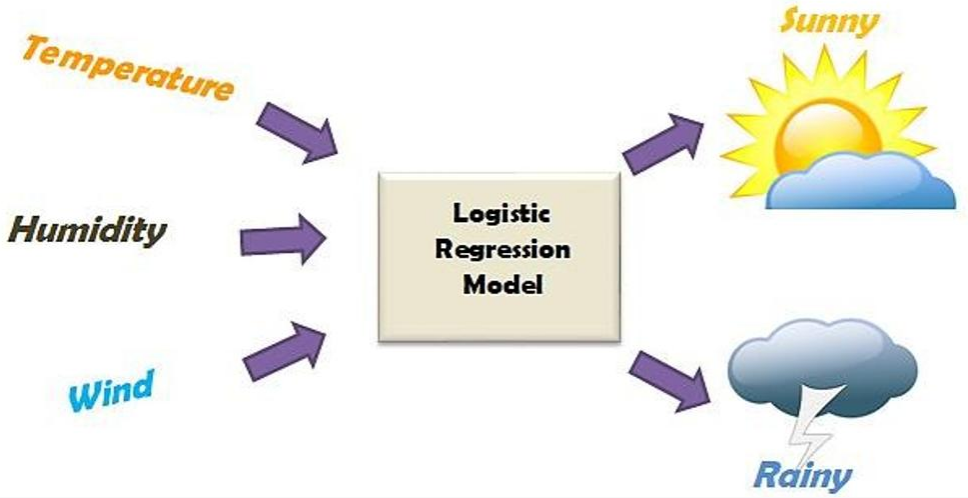
$$S(z) = 1 / (1 + e^{-z})$$

- Outputs a value between 0 and 1.
- Maps large positive numbers to ~1.
- Maps large negative numbers to ~0.



Logistic Regression

Interpreting Output



Logistic Regression outputs a **Probability**.

EXAMPLE

If Model Output = 0.8 for "Is Sunny?", it means there is an **80% chance** the weather is Sunny.

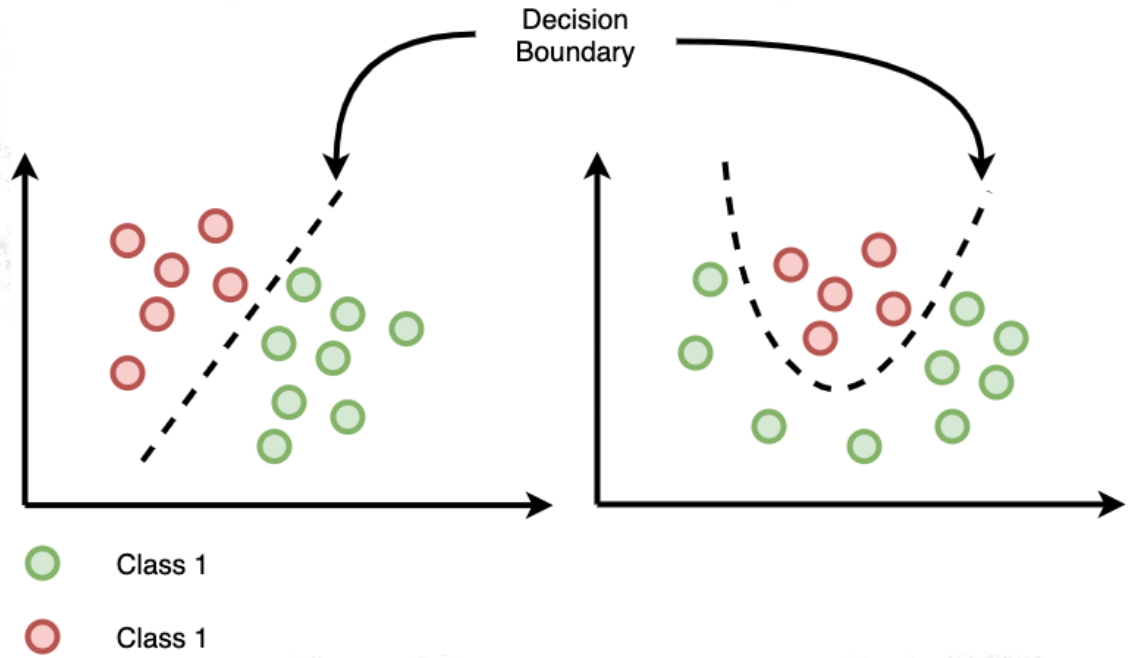
Logistic Regression

The Decision Boundary

To classify, we set a **Threshold** (usually 0.5).

- **If Probability ≥ 0.5:** Predict Class 1 (Positive)
- **If Probability < 0.5:** Predict Class 0 (Negative)

This creates a boundary separating the two classes.



Logistic Regression

Summary

Use Cases

- Spam Detection
- Credit Default Prediction
- Disease Diagnosis (Yes/No)

Colab Notebook:
https://colab.research.google.com/drive/14NwnhcVL5PlcBCA9--7th7vqPhzEDTyq?usp=sharing

Type

Ideally for **Binary Classification**, but can be extended to Multi-class (One-vs-Rest).

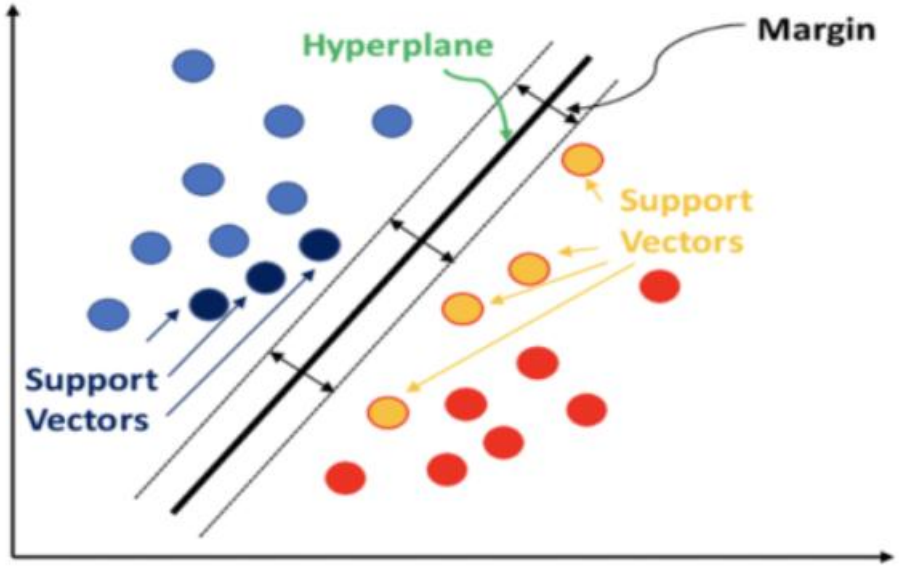
Part 5:
Support Vector Machines

SVM

The Intuition

Imagine trying to separate red and blue balls on a table with a stick.

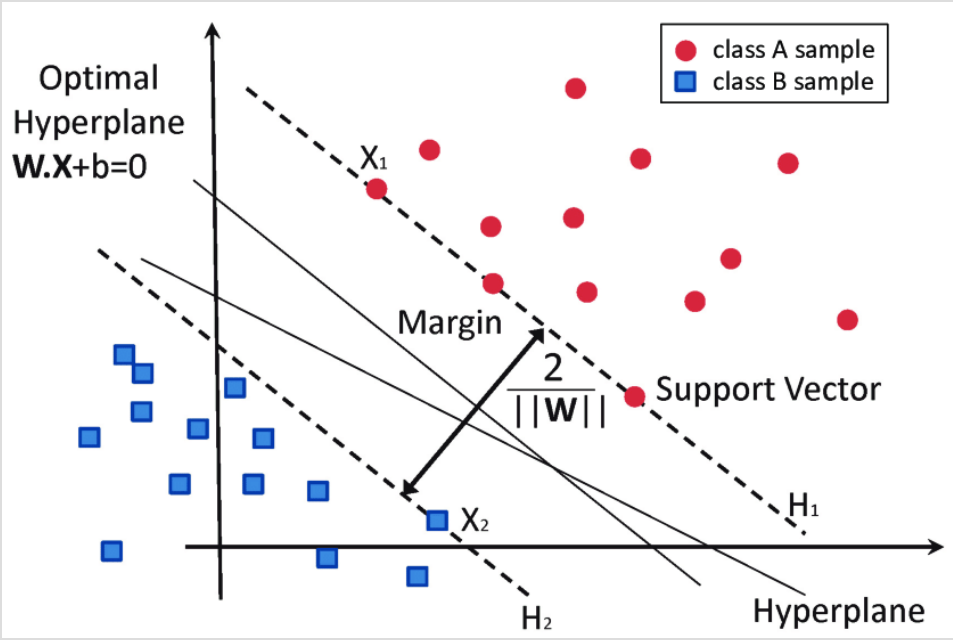
> SVM finds the "widest street" (Margin) that separates the two classes. It doesn't just separate them; it looks for the **optimal** separation.



SVM

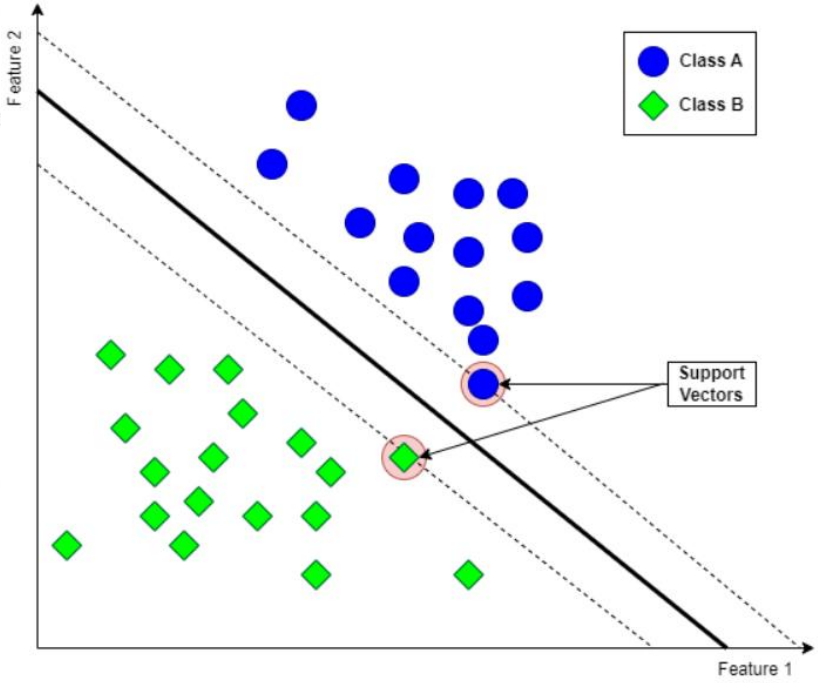
Key Terminology

- **Hyperplane:** The decision boundary line (in 2D) or plane (in 3D).
- **Margin:** The distance between the hyperplane and the nearest data points.



SVM

Support Vectors



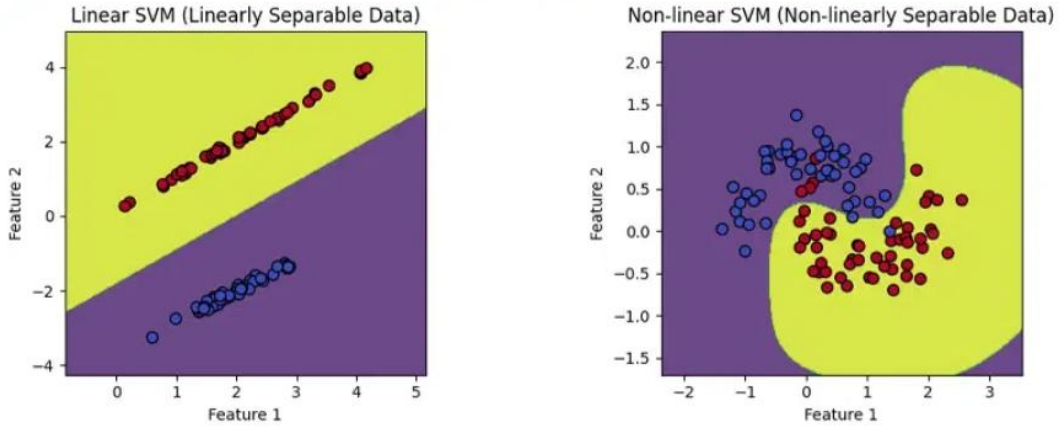
These are the data points **closest to the hyperplane**.

- They are the "hardest" points to classify.
- They dictate the position of the line.
- If you remove other points, the line doesn't change. If you remove support vectors, the line moves.

SVM

Non-Linear Data

What if data cannot be separated by a straight line?



The Kernel Trick

SVM projects data into a **higher dimension** (e.g., 2D to 3D) where it *becomes* linearly separable.

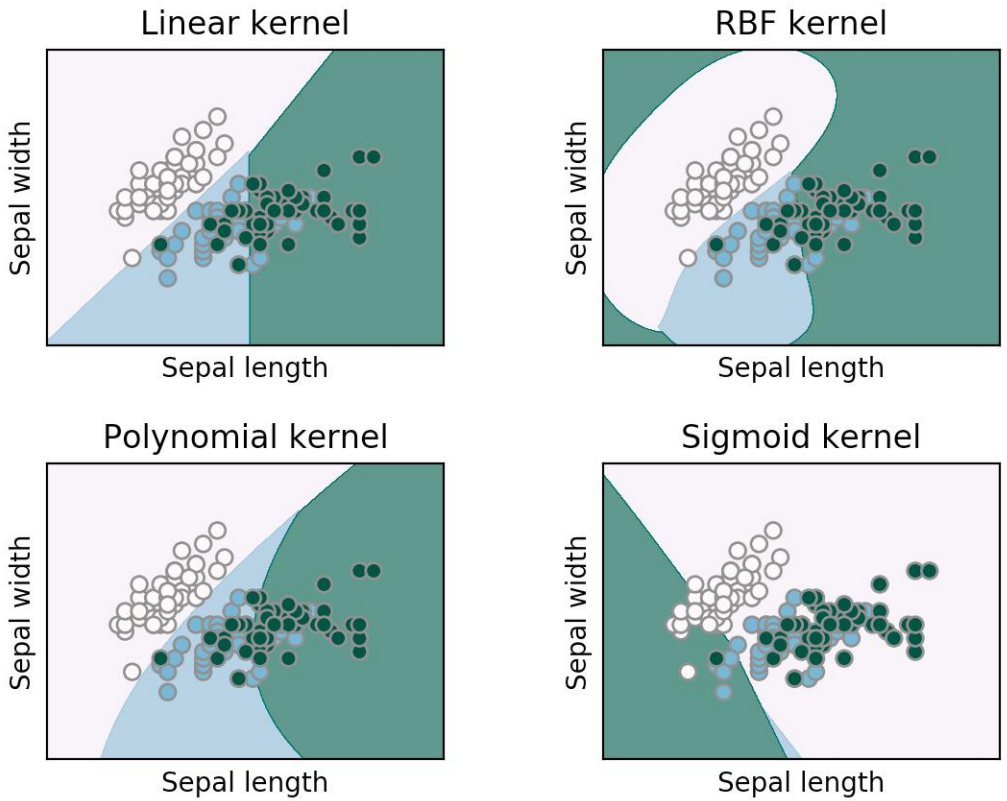
ANALOGY

If red and blue dots are mixed on a flat paper, lift the red dots up (3D) to slide a sheet of paper between them.

SVM

Common Kernels

- **Linear Kernel:** For simple, linear data.
- **Polynomial Kernel:** Curved boundaries.
- **RBF (Radial Basis Function):** Most popular, creates complex, circular boundaries (infinite dimensions).



Colab Notebook:
https://colab.research.google.com/drive/1df3uqnP1r8eLnMno38cgZNKLUonZrMW8?usp=sharing

Part 6:
Evaluation (Regression)

Evaluation (Regression)

Measuring Error

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

SSE (Sum Squared Error)

Total accumulated error. Hard to interpret if dataset size changes.

$$MSE = \frac{1}{n} \Sigma \underbrace{\left(y - \hat{y} \right)^2}_{\text{The square of the difference between actual and predicted}}$$

MSE (Mean Squared Error)

Average of squared errors.

$$MSE = (1/n) \Sigma (y - \hat{y})^2$$

Heavily penalizes large errors (outliers).

Evaluation (Regression)

RMSE (Root Mean Squared Error)

Square root of MSE. This is the standard metric.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

$$\text{RMSE} = \sqrt{\text{MSE}}$$

WHY USE RMSE?

It returns the error to the **original units** of the target variable (e.g., dollars, degrees), making it readable.

Evaluation (Regression)

R-Squared (R^2)

$$R^2 = 1 - \frac{\text{MSE(model)}}{\text{MSE(baseline)}}$$

Measures "Goodness of Fit".

- **Range:** 0 to 1.
- **Interpretation:** "How much variance in Y is explained by X?"

R^2 = 1.0: Perfect fit.

R^2 = 0.0: Model is no better than guessing the average.

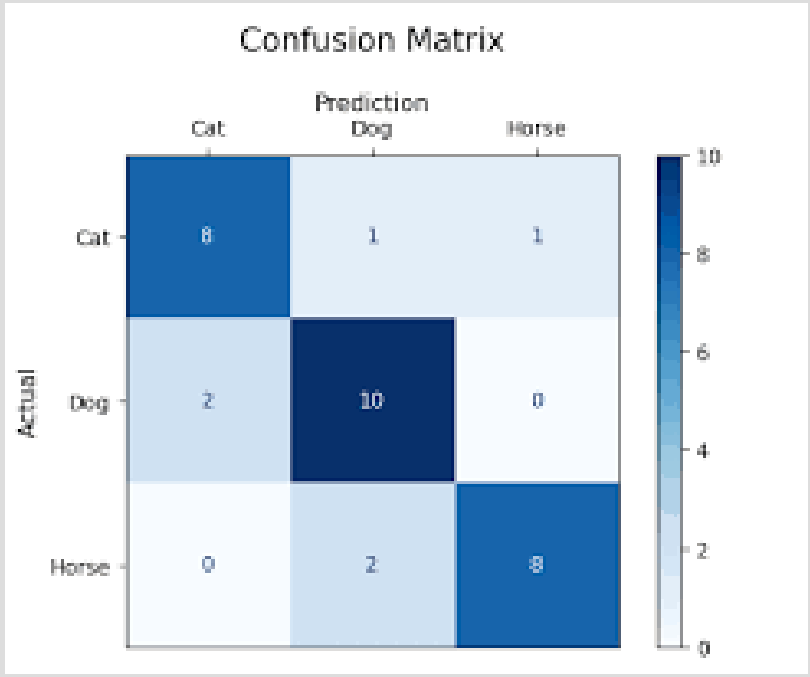
Colab Notebook:
https://colab.research.google.com/drive/1AuPZIE5sCJyoR7DApmFzmi75L5lZVZLI?usp=sharing

Part 7:
Evaluation (Classification)

Evaluation (Classification)

The Confusion Matrix

A table that describes the performance of a classification model.



Compares Predicted vs Actual values.

Evaluation (Classification)

Matrix Components

Correct

- **TP (True Positive):** Predicted Yes, Actually Yes.
- **TN (True Negative):** Predicted No, Actually No.

Incorrect

- **FP (False Positive):** Predicted Yes, Actually No (Type 1 Error).
- **FN (False Negative):** Predicted No, Actually Yes (Type 2 Error).

Evaluation (Classification)

Precision

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

"Out of all the times we predicted Positive, how often were we right?"

$$\text{Precision} = TP / (TP + FP)$$

USE CASE

Spam Detection: We don't want to label a good email as spam (Low False Positives).

Evaluation (Classification)

Recall (Sensitivity)

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

"Out of all the actual Positives, how many did we catch?"

$$\text{Recall} = TP / (TP + FN)$$

USE CASE

Cancer Detection: We cannot afford to miss a sick patient (Low False Negatives).

Evaluation (Classification)

F1-Score

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

The Harmonic Mean of Precision and Recall.

$$F1 = 2 * (Precision * Recall) / (Precision + Recall)$$

- Used when you need a balance between Precision and Recall.
- Useful for **imbalanced datasets**.

Evaluation (Classification)

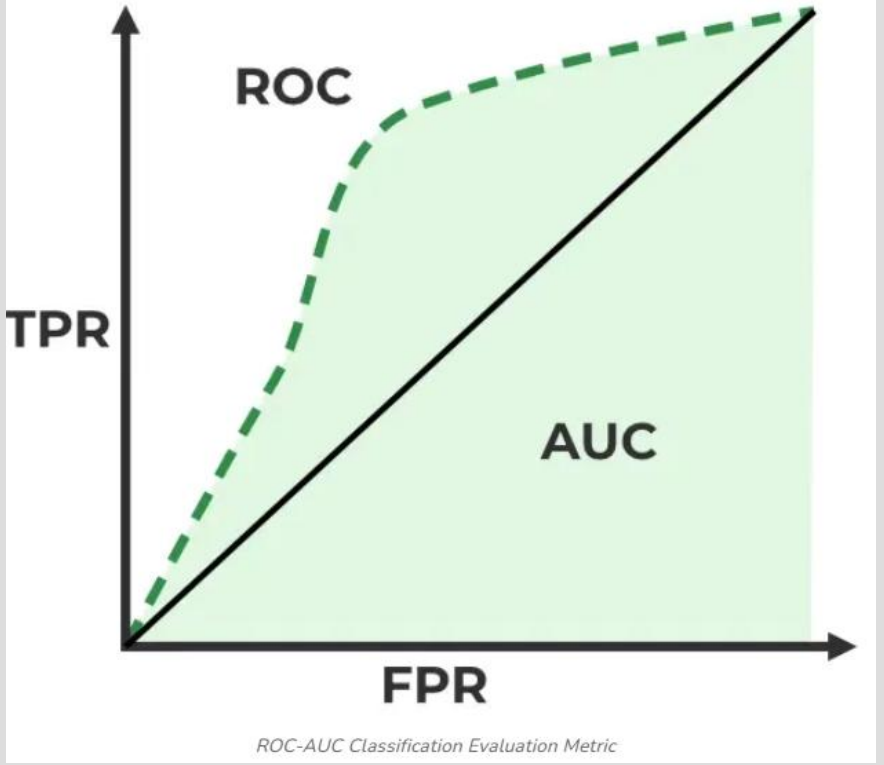
ROC Curve & AUC

ROC Curve

Plots **TPR** (Recall) vs **FPR** (False Positive Rate) at different thresholds.

AUC (Area Under Curve)

Score from 0 to 1. Higher is better.



ROC-AUC Classification Evaluation Metric

Colab Notebook:
https://colab.research.google.com/drive/1polmxTzUPQ9NRF6tegmwpYFjK6L9VJie?usp=sharing

Summary

Recap

- **Supervised Learning:** Learning with labeled data (Input -> Output).
- **Algorithms:** KNN (Distance), Linear Regression (Line), Logistic Regression (Sigmoid/Prob), SVM (Margin/Hyperplane).
- **Evaluation:** Use RMSE for Regression. Use Precision/Recall/F1 for Classification.

Summary

Quick Comparison

Algorithm	Type	Best For
Linear Regression	Regression	Predicting values (Price)
Logistic Regression	Classification	Binary Outcomes (Yes/No)
KNN	Both	Small datasets, simple Logic
SVM	Classification	Complex/High-dim data

Thank You!

Questions?