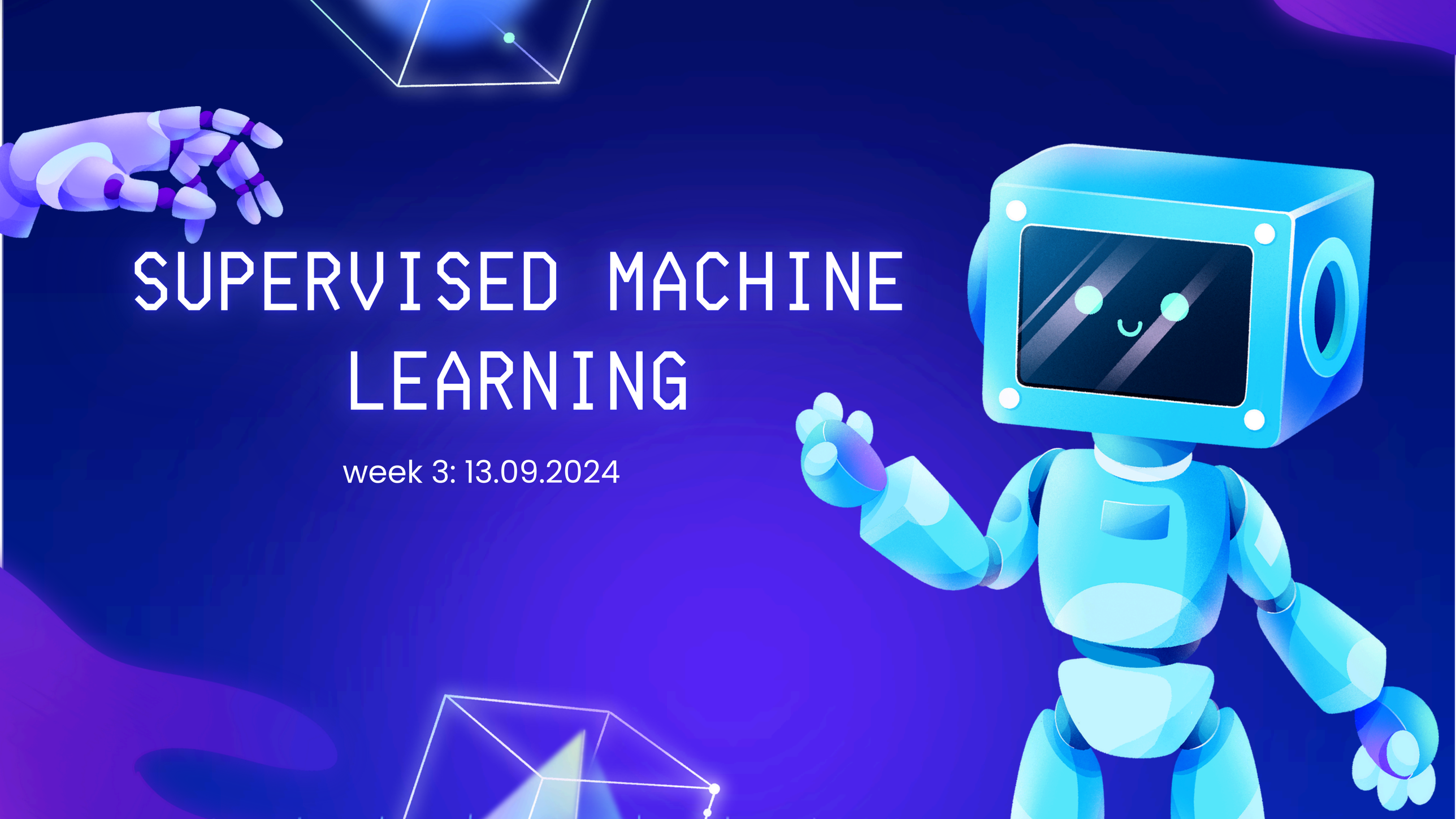
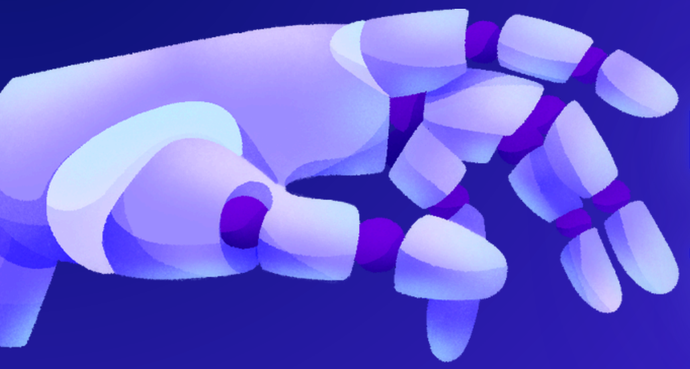
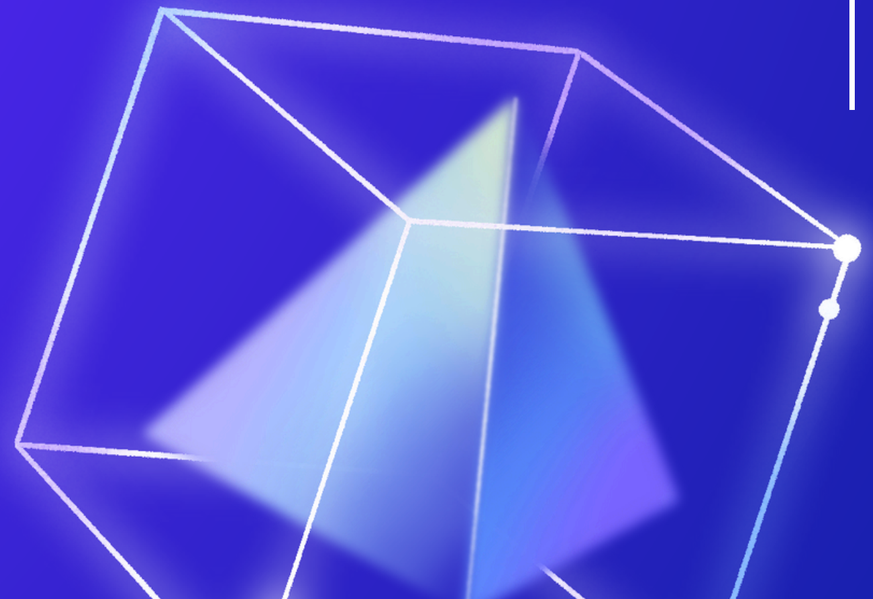

SUPERVISED MACHINE LEARNING
week 3: 13.09.2024



OUTLINE FOR THE SESSION

- Repetition of Supervised machine learning
- Confusion matrix
- kNN
- Scaling and normalisation
- Linear classification
- Perceptron
- Quiz !




WHAT IS SUPERVISED LEARNING?

WHAT SUPERVISED LEARNING IS ALL ABOUT

Supervised learning is a fundamental approach in machine learning where algorithms learn by example, just like a student learns from a teacher. It involves feeding the algorithm **labeled data sets**, where **each data point has both an input (features) and a desired output (label)**. This allows the algorithm to learn the relationship between the inputs and outputs, enabling it to make predictions on new, unseen data.

WHAT ARE THE MAIN COMPONENTS OF SUPERVISED MACHINE LEARNING?

MAIN COMPONENTS OF SUPERVISED MACHINE LEARNING

1 . Labeled Data

2. Features:
Characteristics or attributes of the data used by the algorithm to learn and make predictions.

3. Learning Algorithm:

Classification:
Categorizes data points (spam/not spam, cat/dog). Examples: Decision trees, support vector machines (SVMs), k-nearest neighbors (kNN).

Regression: Predicts continuous values (housing prices, weather forecasts). Examples: Linear regression, polynomial regression, random forests.

4. Model Training(Train):
iteratively adjusting the algorithm's internal parameters based on the labeled data to minimize the prediction error.

5. Model Evaluation(Dev):
Assessing the trained model's performance on unseen data to measure its ability to generalize to new situations.

6. Prediction(Test):
Using the trained model to make predictions on new, unseen data points based on their input features.

WHAT IS TRAIN, DEV, TEST AND WHY DO WE NEED THEM?

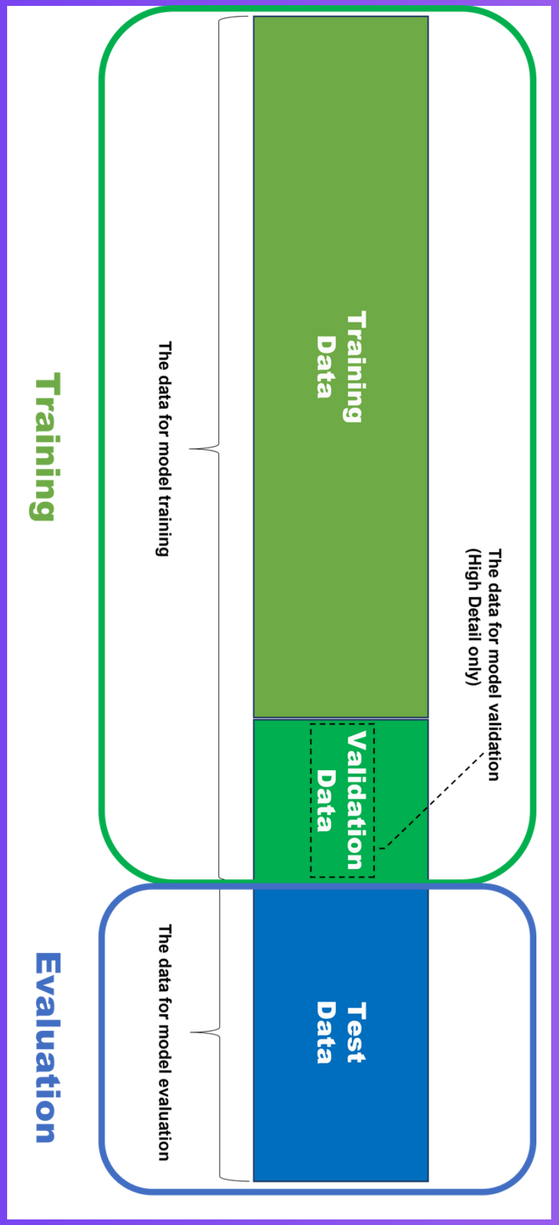
TRAIN DEV TEST

1. Train Set:
Largest portion of data (usually 70-80%), used to train the model and learn its parameters.

2. Development/Validation Set (Dev Set):
A smaller subset (10-20%) of data used to fine-tune the model hyperparameters (e.g., learning rate, number of layers). Used it to identify potential overfitting or underfitting issues

3. Test Set:
The smallest subset (10-20%) of your data held out completely unseen by the model during training and development. Performance on unseen data, simulating real-world scenarios.

Why are they necessary?
Train-Dev split: Prevents overfitting, where the model memorizes the training data and performs poorly on unseen data.
Test Set: Provides an unbiased estimate of the model's true performance on completely new data.



WHAT IS THE DIFFERENCE BETWEEN CLASSIFICATION AND REGRESSION?

REGRESSION VS CLASSIFICATION

Feature	Regression	Classification
Output type	Continuous value	Discrete categories
Examples of tasks	Predicting prices, temperatures, population growth	Classifying emails, handwritten digits, images
Model goal	Predict a specific value	Assign a data point to a class

HOW TO MEASURE ERRORS/PERFORMANCE FOR CLASSIFICATION?

CONFUSION MATRIX AND IT'S FORMULAS

Predicted Class

Actual Class	**Positive**	**Negative**	
Positive	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP + FN)}$
Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN + FP)}$
	Precision $\frac{TP}{(TP + FP)}$	**Negative Predictive Value** $\frac{TN}{(TN + FN)}$	**Accuracy** $\frac{TP + TN}{(TP + TN + FP + FN)}$

Accuracy = (True Positives + True Negatives) / (Total Instances)	**Accuracy: Ratio of correctly classified instances to the total number of instances.**
Precision = TP / (TP + FP)	Precision: Ratio of correctly predicted positive instances to all predicted positive instances.
Recall = TP / (TP + FN)	Recall (Sensitivity): Ratio of correctly predicted positive instances to all actual positive instances.
Specificity = TN / (TN + FP)	Specificity: Ratio of correctly predicted negative instances to all actual negative instances.
F1 = 2 * (Precision * Recall) / (Precision + Recall)	F1-score: Harmonic mean of precision and recall, combining both metrics into a single score.

MACRO VS MICRO AVERAGE

Macro-averaging calculates each class's performance metric (e.g., precision, recall) and then takes the arithmetic mean across all classes. So, the macro-average gives equal weight to each class, regardless of the number of instances.

$$\text{Macro Average (Metric)} = (1 / N) * \Sigma (\text{Metric}_i)$$

Micro-averaging, on the other hand, aggregates the counts of true positives, false positives, and false negatives across all classes and then calculates the performance metric based on the total counts. So, the micro-average gives equal weight to each instance, regardless of the class label and the number of cases in the class.

$$\text{Micro Average (Metric)} = (\Sigma TP_i + \Sigma TN_i) / (\Sigma TP_i + \Sigma TN_i + \Sigma FP_i + \Sigma FN_i)$$

WHAT IS KNN?

KNN ALGORITHM

Step 1: Selecting the optimal value of K

- K represents the number of nearest neighbors that needs to be considered while making prediction.

Step 2: Calculating distance

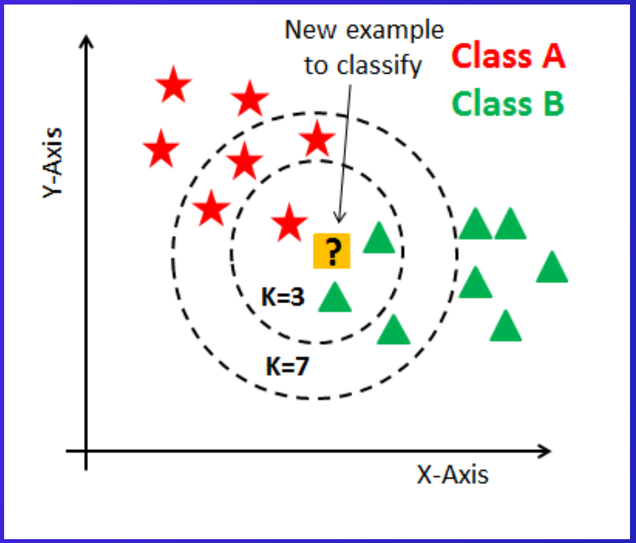
- To measure the similarity between target and training data points, Euclidean distance is used. Distance is calculated between each of the data points in the dataset and target point.

Step 3: Finding Nearest Neighbors

- The k data points with the smallest distances to the target point are the nearest neighbors.

Step 4: Voting for Classification or Taking Average for Regression

- The class with the most occurrences among the neighbors becomes the predicted class for the target data point.
- In the regression problem, the class label is calculated by taking average of the target values of K nearest neighbors.



PROS AND CONS OF KNN

Benefits of KNN:

Simple and easy to understand, Versatile: Can be used for both classification and regression tasks.

No explicit model training: No need to define a complex model, making it efficient for small datasets.

Drawbacks of KNN:

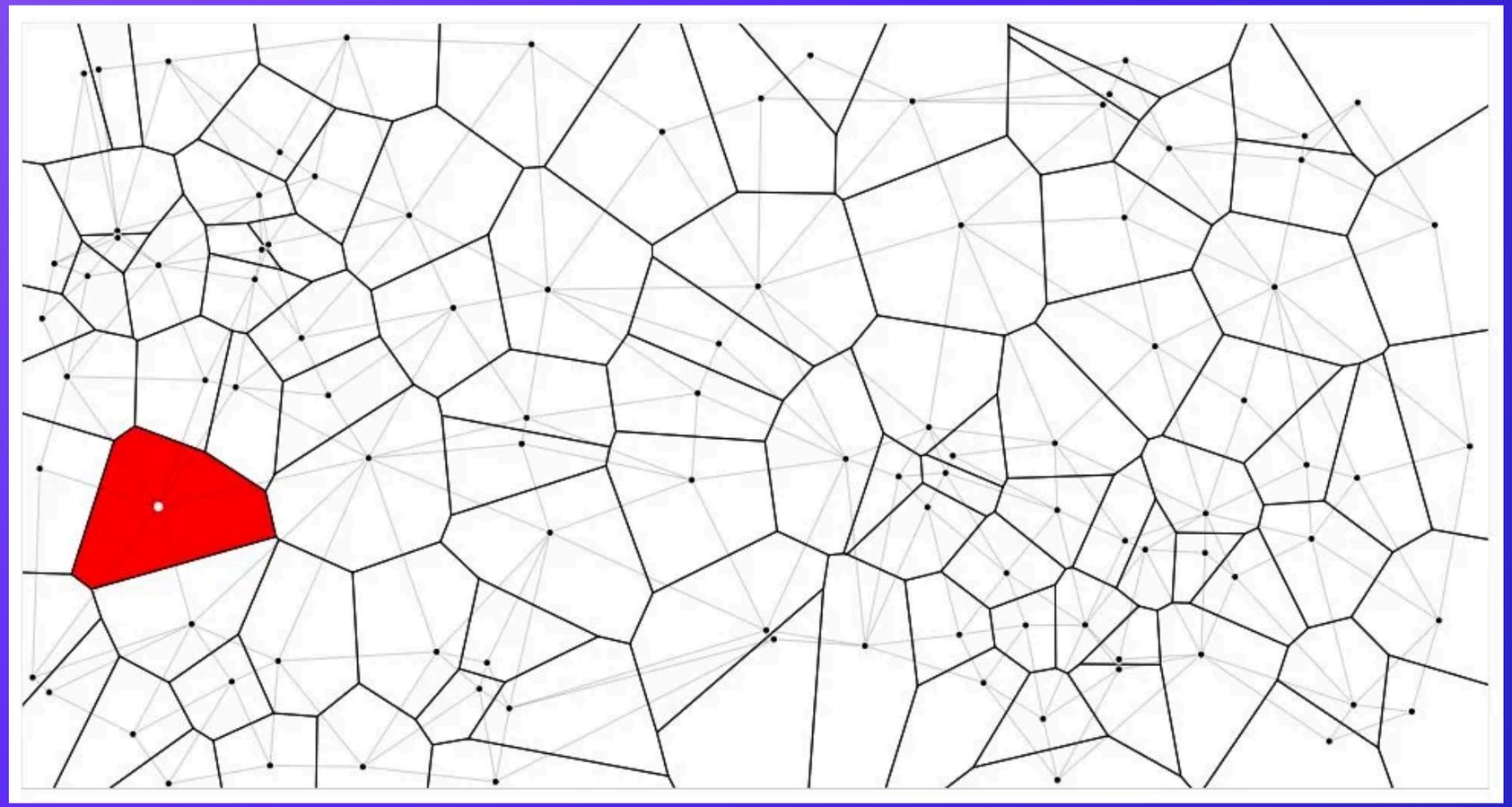
Sensitive to distance metric, High computational cost, Curse of dimensionality

Real-world applications of KNN:

Recommending products based on past purchases (similar users bought these).

VORONOI TESSALATION

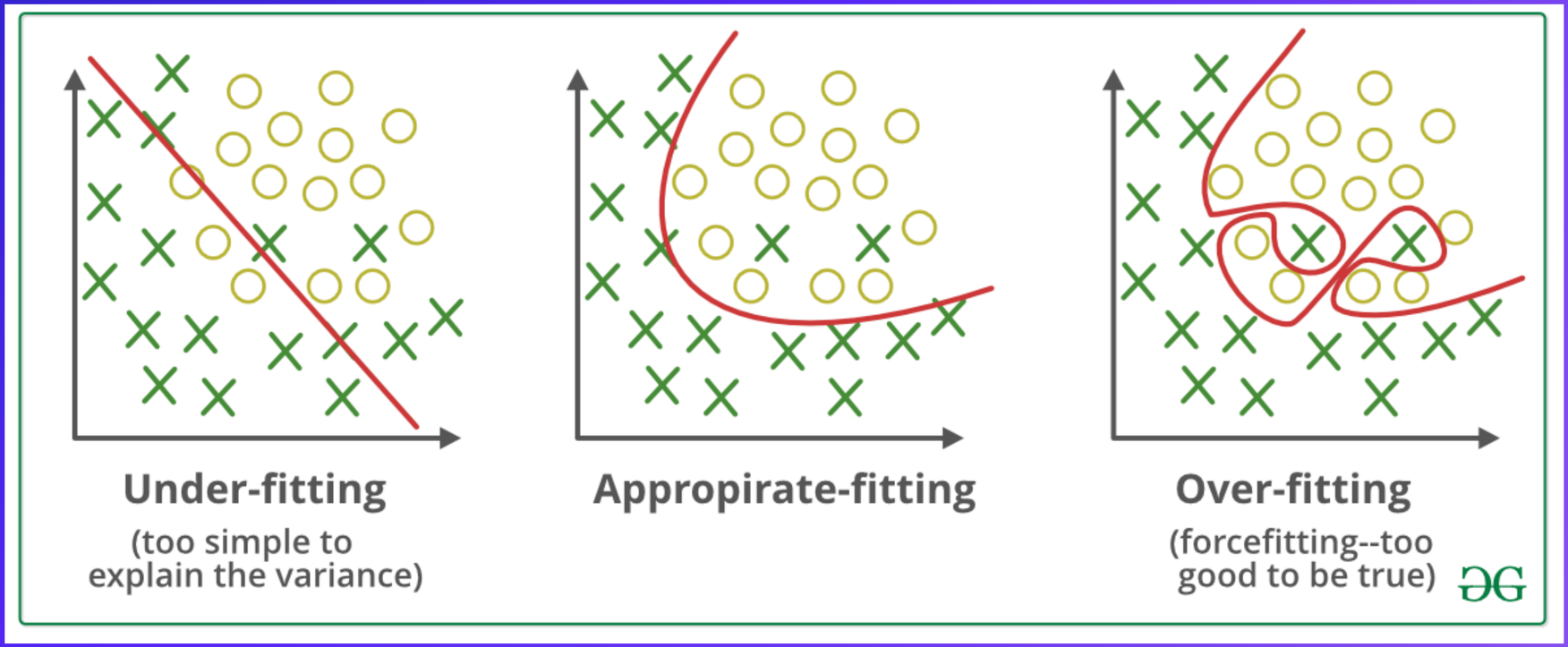
A CASE OF K-NN MACHINE LEARNING ALGORITHM WHEN K=1



https://medium.com/@malhotra.amit.236/love-thy-k-nearest-neighbors-58932fa66a08

OVERFITTING/UNDERFITTING OF KNN

The value of k in the KNN algorithm is related to the error rate of the model. A small value of k could lead to overfitting as well as a big value of k can lead to underfitting. Overfitting imply that the model is well on the training data but has poor performance when new data is coming.



FEATURES AS VECTORS

In order to make data features readable for the computer we turn them into vectors. Feature vectors can be binary, numerical .... other types

Example of feature vectors:

Image Recognition:

Consider classifying images of cats and dogs. Features could be:

- Pixel intensities: Flattened vector of pixel values (e.g., 28x28 grayscale image becomes a 784-dimensional vector)
- Edge detection: Vector representing detected edges in the image
- Texture analysis: Vector summarizing image texture properties

A dog image vector might have higher values in edge detection features for legs, ears, and tail.

1. Grayscale Image:

If the image is grayscale, each pixel can be represented by a single intensity value ranging from 0 (black) to 255 (white). In this case, your feature vector would be a 9-dimensional vector, where each element corresponds to the intensity value of a single pixel. For example:

[100, 150, 200, 220, 255, 230, 180, 120, 50]
This vector represents an image where the top left pixel is darkest, the center pixel is brightest, and there's a gradual decrease in intensity towards the bottom right corner.

WHAT IS FEATURE NORMALISATION AND SCALING? WHAT DO WE NEED IT FOR?

FEATURE SCALING AND NORMALISATION OF VECTORS

Feature Scaling

Min-Max scaling: Scales features between a specified minimum and maximum value (usually 0 and 1).
Standard scaling (Z-score normalization): Subtracts the mean of each feature and then divides by the standard deviation.

Other scaling methods: Logarithmic scaling, power scaling, etc., depending on the data distribution.

Let

- $min_i = \min(\{x_{j,i} \mid j = 1,2,\dots,N\})$
- $max_i = \max(\{x_{j,i} \mid j = 1,2,\dots,N\})$

Define $scale_i(x_{j,i}) = \frac{x_{j,i} - min_i}{max_i - min_i}$

Benefits:

Improves convergence and stability of optimization algorithms.
Prevents features with large magnitudes from dominating the learning process.
Improves interpretability of some machine learning models.

Normalization:

L1 normalization (Manhattan distance): Sums the absolute values of each feature and divides each feature by the sum.
L2 normalization (Euclidean distance): Calculates the square root of the sum of squared values for each feature and divides each feature by the square root.
Benefits:

Improves performance in distance-based algorithms like k-Nearest Neighbors.
Can be useful for data with sparse features.

Find the

- mean $\mu_i = \text{mean}(\{x_{j,i} \mid j = 1,2,\dots,N\})$, and
- standard deviation $\sigma_i = \text{std}(\{x_{j,i} \mid j = 1,2,\dots,N\})$
- of your training set.

Define $scale_i(x_{j,i}) = \frac{x_{j,i} - \mu_i}{\sigma_i}$ for each feature $i = 1, \dots m$

WHAT IS LINEAR CLASSIFICATION?

LINEAR CLASSIFICATION

The core of linear classification is a linear equation that defines the boundary between categories. This equation takes the feature values of a data point and combines them with weights and a bias term. If the equation's result is greater than a certain threshold, it's classified as one category; otherwise, it's classified as the other.

$$f(\boldsymbol{x}) = f((x_1, x_2, \dots, x_m)) = w_0 + w_1 x_1 + w_2 x_2 + \dots w_m x_m$$

Strengths of Linear Classification:

Interpretable: The decision boundary is a straight line, making it easy to understand why a data point is classified a certain way.
Efficient: Relatively fast and computationally inexpensive, especially for large datasets.
Works well with high-dimensional data: Can handle many features, provided they are linearly separable.

Weaknesses:

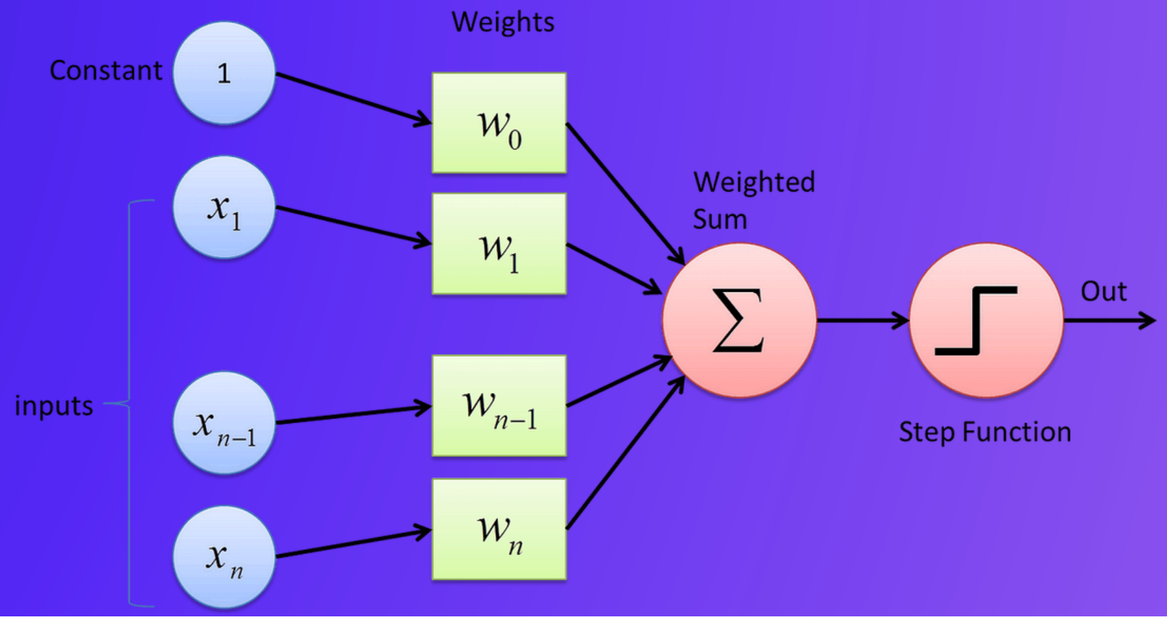
Limited to linearly separable data: Cannot perfectly classify data that cannot be separated by a straight line.
May not capture complex relationships: Assumes a linear relationship between features and the target variable, which may not always hold true.

Real-world applications: Spam filtering

WHAT IS A PERCEPTRON?

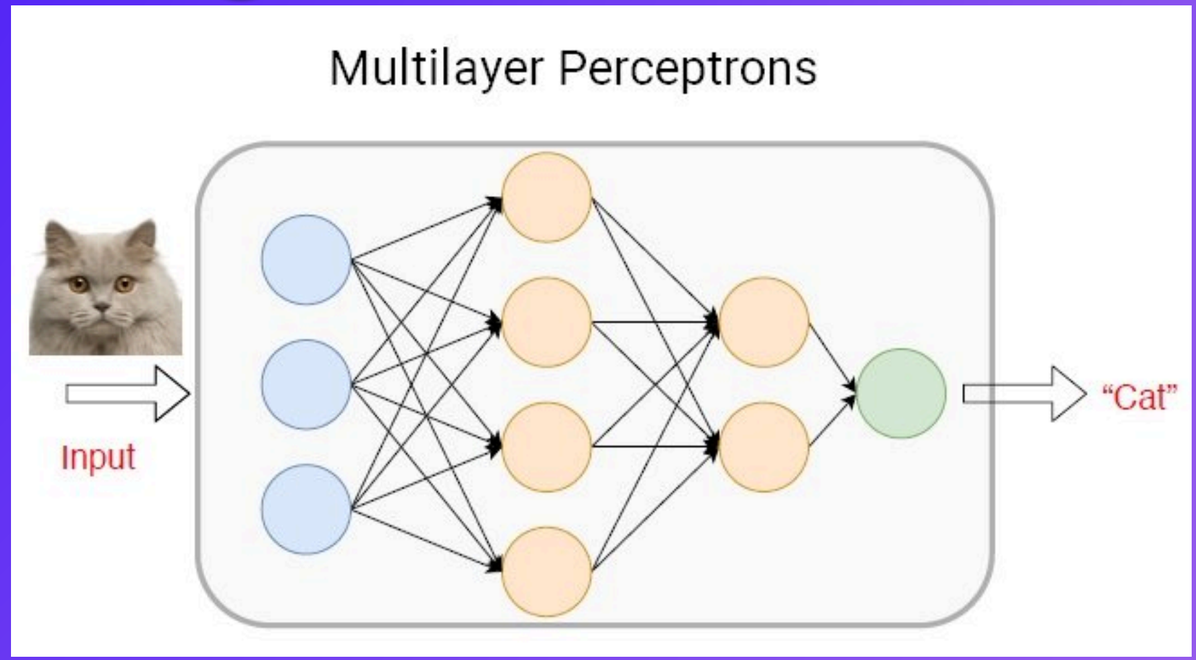
PERCEPTRON

A perceptron is a single-layer neural network that forms the building block of more complex neural networks. It acts as **a basic binary classifie**r, meaning it can only distinguish between two classes.



Structure:

Inputs: Represented by a vector of numerical values.
Weights: Each input has a corresponding weight, determining its influence on the output.
Bias: A constant value added to the weighted sum of inputs.
Activation function: A mathematical function that applies a threshold to the weighted sum and bias, producing the final output (0 or 1).



Functioning:

Each input is multiplied by its corresponding weight.
The weighted sums are added together and the bias is added.
The activation function is applied to the resulting sum.
If the sum is greater than or equal to a threshold (usually 0), the output is 1 (positive class). Otherwise, the output is 0 (negative class).

Learning:

A perceptron can learn by adjusting its weights through a process called the Perceptron Learning Rule. This rule iterates through the training data, adjusting weights to minimize the error between the predicted and desired outputs.

PERCEPTRON UPDATE RULE

- Adjust the threshold
  - Adjust bias term (w_0)

  - add a new feature $x_0 = -1$ for all items
  - Replace $(x_1, x_2, \ldots, x_m)$ with $(-1, x_1, x_2, \ldots, x_m)$
  - Replace $\sum_{i=1}^{m} w_i x_i > \theta$ with $h = \sum_{i=0}^{m} w_i x_i > 0$

- Adjust weights

$$w_i = w_i + \eta(t - y)x_i = w_i - \eta(y - t)x_i$$

(covers all cases)

PERCEPTRON FORWARD VS UPDATE STEP

The forward step calculates the predicted output based on current parameters, while the update step uses the error to adjust those parameters for better future predictions.

The forward step doesn't involve any updating, only calculating a prediction.

WHAT IS AN APPROPRIATE LOSS FUNCTION FOR LINEAR REGRESSION?

MSE

MSE is the type of loss function used for the linear regression/ classification.

We want to find the weights that minimise the mean square error

$$\frac{1}{N}\sum_{j=1}^{N}\left(t_j - \sum_{i=0}^{m} w_i x_{j,i}\right)^2$$

$$= \frac{1}{N}\sum_{j=1}^{N}\left(t_j - y_j\right)^2$$