Data Science

Machine Learning

- **Model Training**:
  - The model is trained to learn patterns from data.
  - Once trained, the model is used to predict outcomes for new data.
- **General Equation**:
  - Every ML algorithm solves: **y = f(x)**
    - **y**: The target/output to predict.
    - **x**: The input features used to make predictions.
- **Examples**:
  - **Salary Prediction**:
    - Salary (y) = f(exp, college, degree_type, previous salary, etc.)
  - **Survival Prediction** (Titanic dataset):
    - Survival (y) = f(age, Pclass, fare, gender, etc.)

Type of ML

- **Supervised Learning:** The model is trained on labeled data (input-output pairs).
- **Goal:** Predict the output (**y**) for new inputs (**x**).
- **Examples**:
  - Predicting house prices.
  - Classifying emails as spam or not spam.

- **Unsupervised Learning:** The model is trained on unlabeled data (only inputs, no outputs).
- **Goal**: Find hidden patterns or structure in the data.
- **Examples**:
  - Customer segmentation.
  - Market basket analysis.

Supervised Learning: Classification vs Regression

- **Classification:** Predicts discrete categories or classes.
- **Output**: Categorical (e.g., Yes/No, 0/1, Spam/Not Spam).
- **Examples**:
  - Predicting if an email is spam.
  - Classifying images as cats or dogs.

- **Regression:** Predicts continuous numerical values.
- **Output:** Continuous (e.g., price, temperature, salary).
- **Examples**:
  - Predicting house prices.
  - Estimating sales revenue.

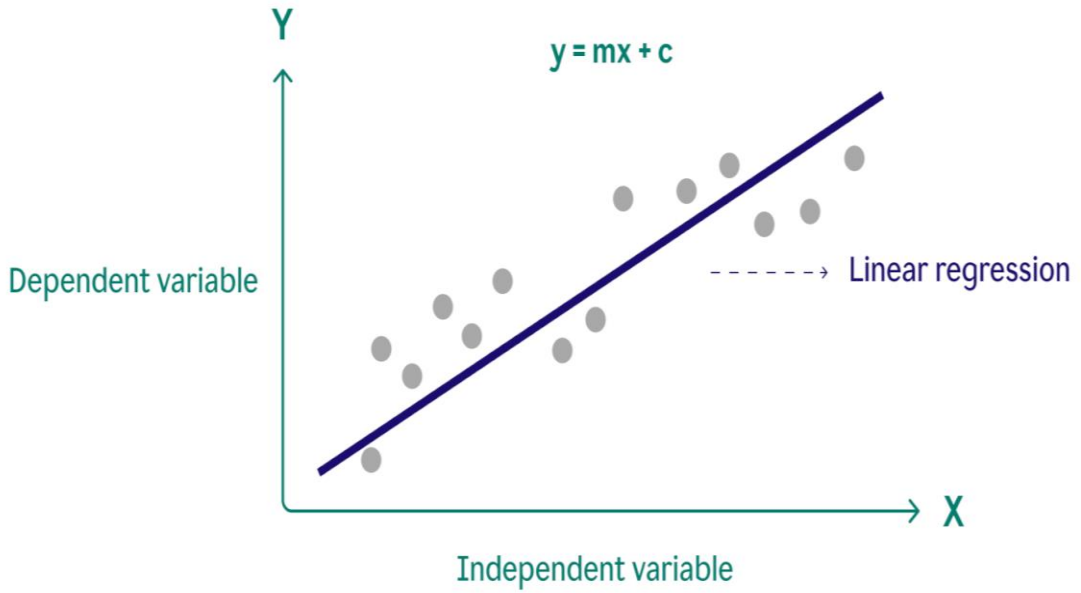
ML Algorithms

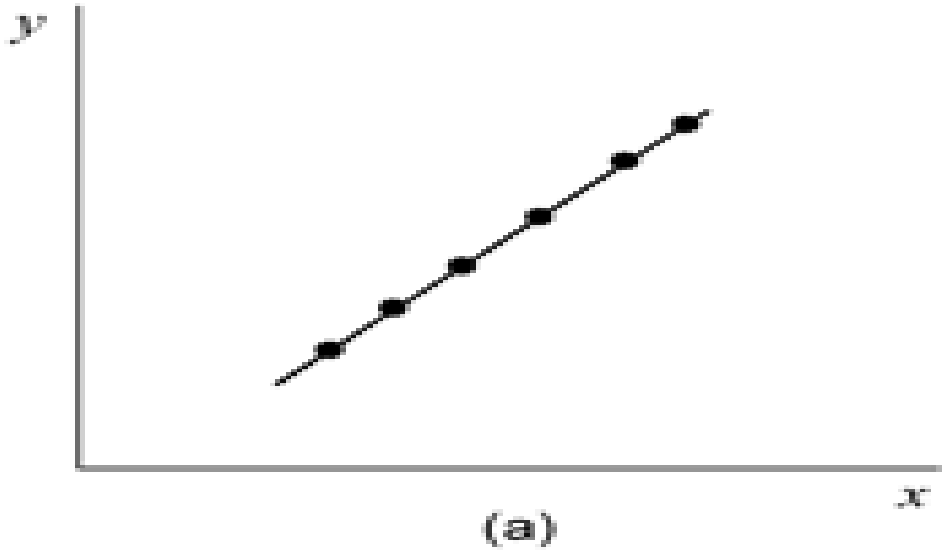
- Classification: k-nearest neighbors (KNN)
- Regression: Linear Regression, k-nearest neighbors (KNN)

- Other algos:
- Classification: Logistic Regression, SVM, Decision tree, random forest, xgboost, catboost, etc
- Regression: SVR, Decision tree, random forest, xgboost, catboost, etc

Linear Regression

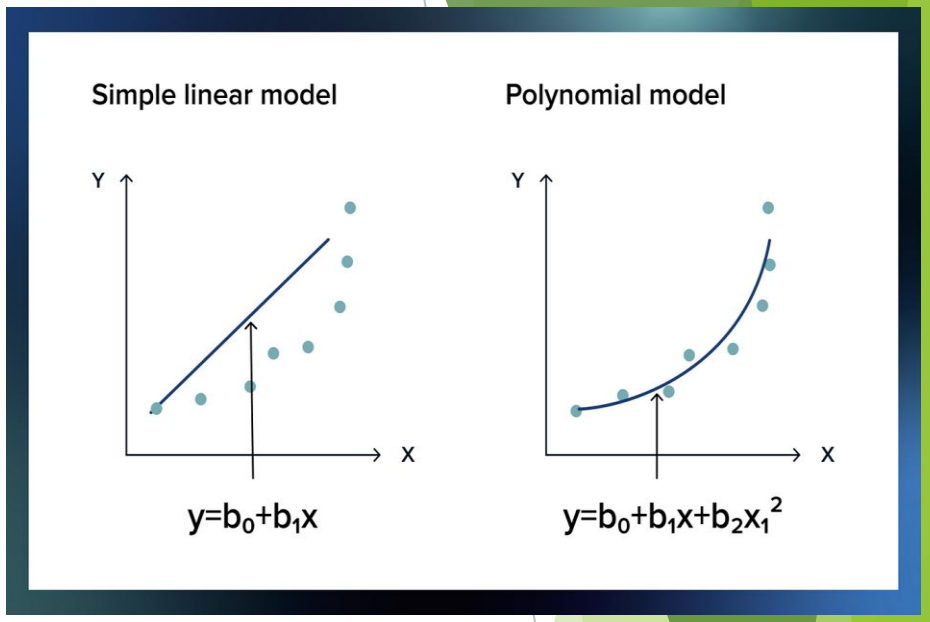
- It tries to fit best possible line (2d) or plane (3d) that fits perfectly on data.
- Equation of hyperplane (nd): y = w0 + w1*x1 + w2*x2 + …. + wnxn
- It tries to find the slope(w1,w2,….,wn) and y intercept (w0).
- Eg. House price prediction, sales prediction

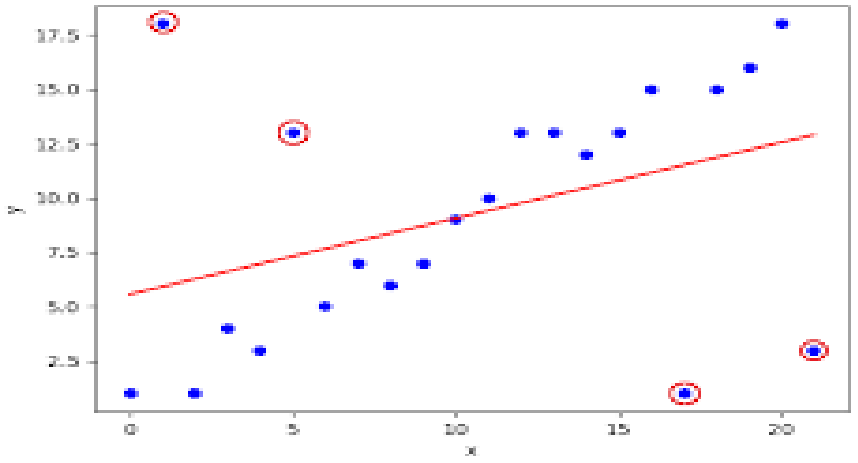




Linear Regression Example

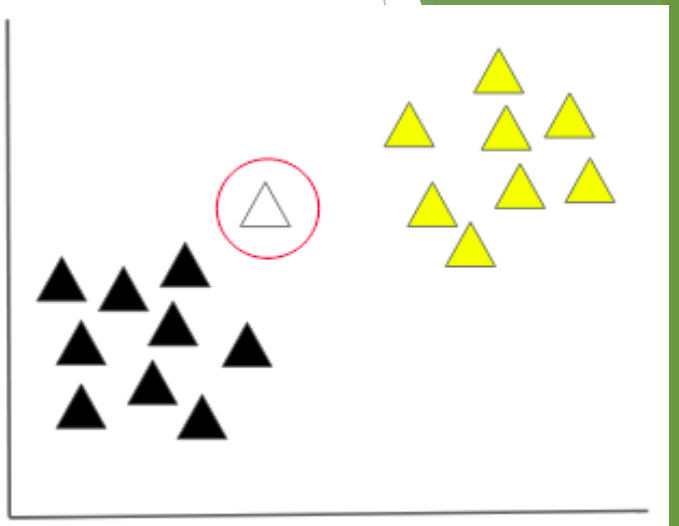
- House price prediction: **y(price) = f(n_rooms, age, zip)**
- So equation will be: **price = w0 + w1*n_rooms + w2*age + w3*zip**
- So objective of linear regression is to find intercept (w0) and weights/slope (w1,w2,w3). Once we have this values we can find price for any n_rooms, age, zip.
- Note: since w1,w2,w3 represent slope it tells us how important that column is in predicting price and sign tells you what is correlation of it with price. Also, w0 tells you what will be price if n_rooms, age and zip is zero.
- Linear regression fails in case of non linear data and outlier.

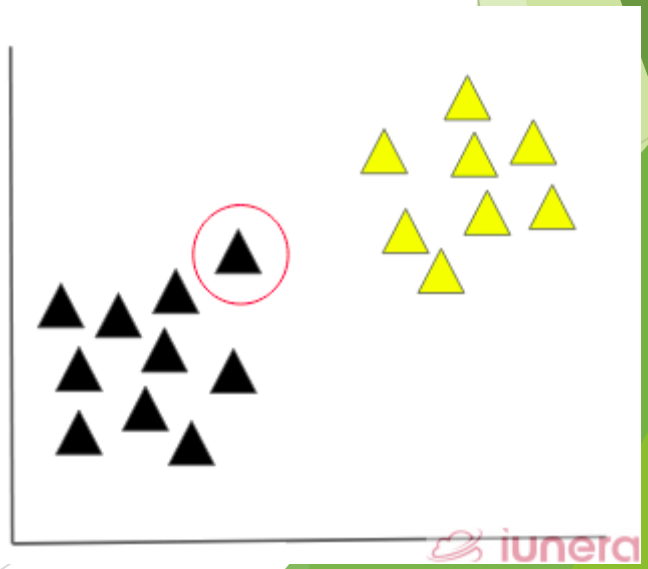




k-nearest neighbors (KNN)

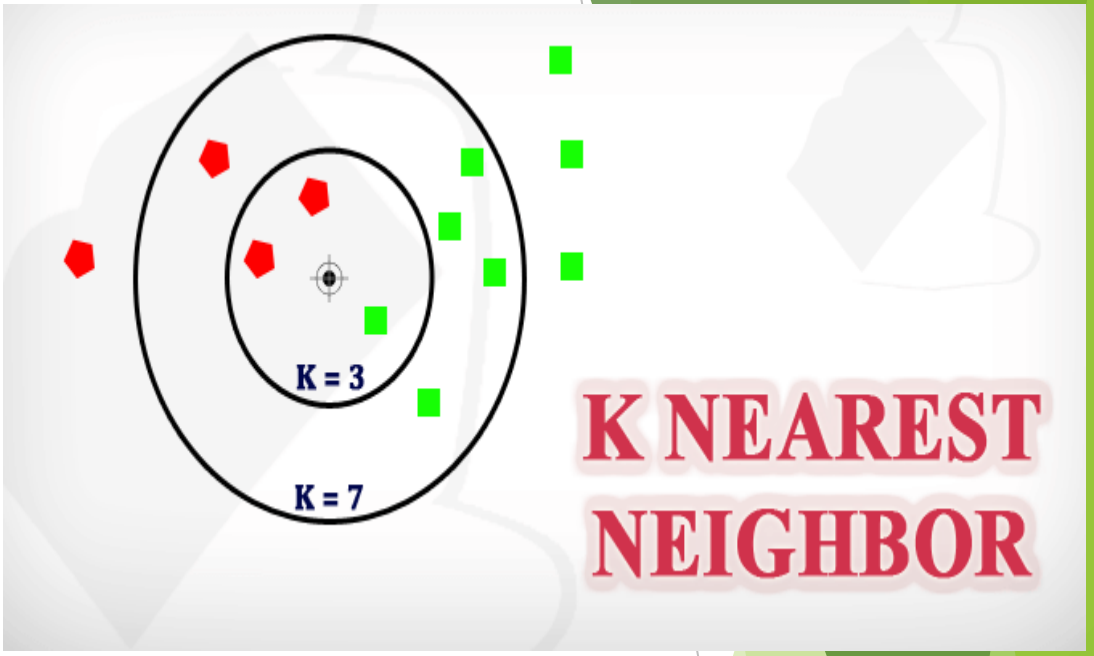
- It follows rule: If two points are close to each other then they share similar properties.
- For eg. If a new patient comes and if properties of this patient matches more with other cancer patients then algorithm also says that this patient also has cancer else vice versa.
- Its simple and easily interpretable.
- It can be both for regression and classification.

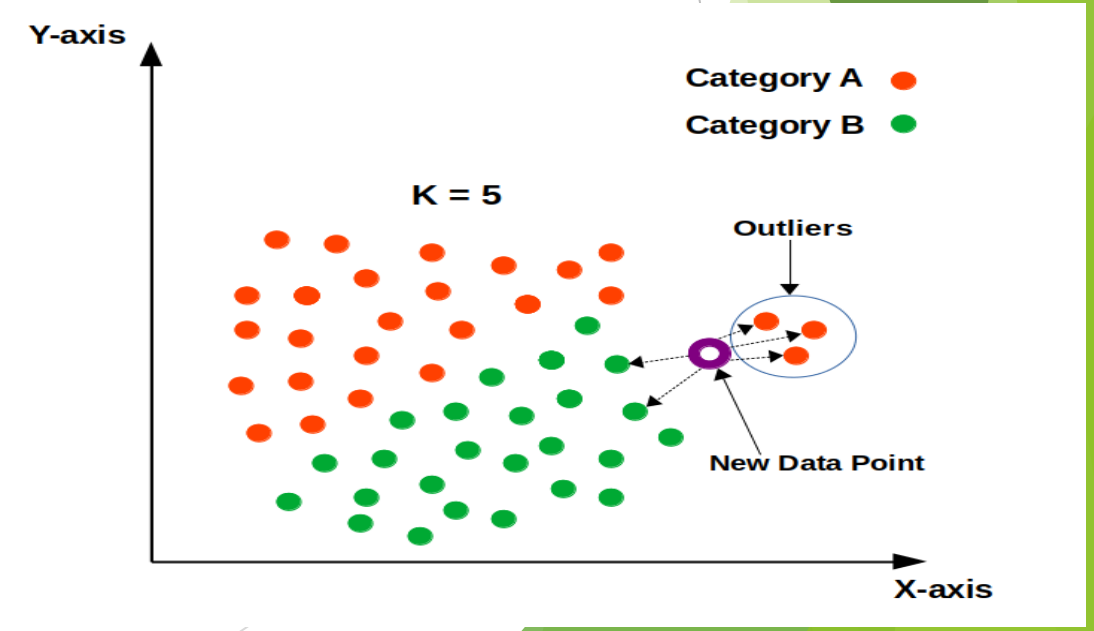




KNN continued

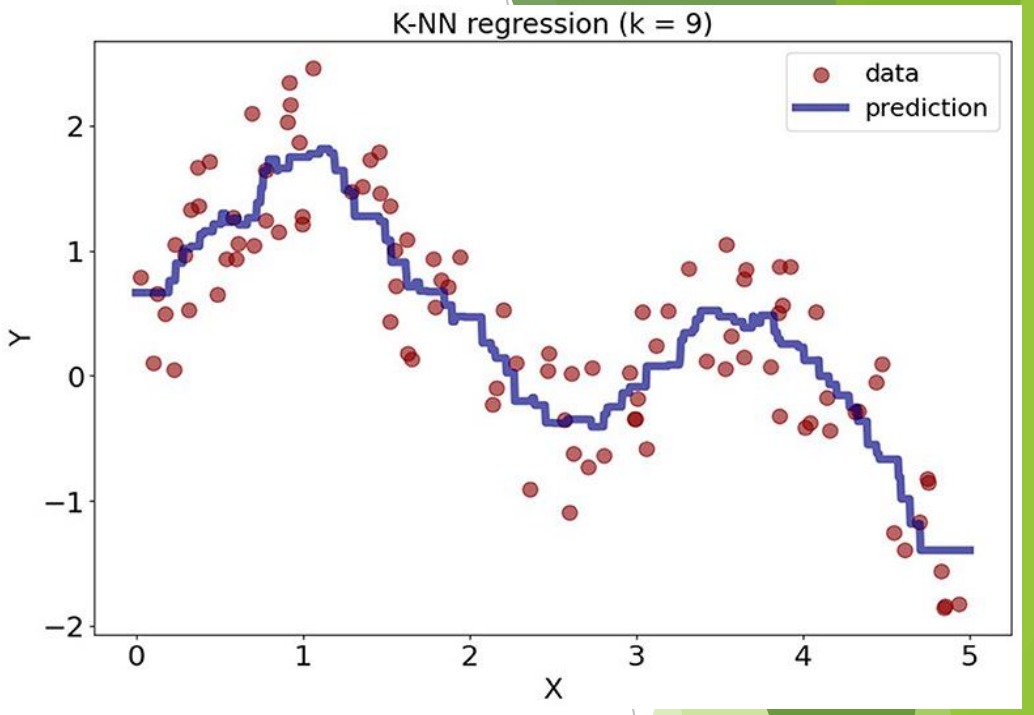
- Hyperparameter k: Here it is defined as how many close points to check to find the class of query point
- Its good practice to have value of k as odd else it can result in tie.
- Typical k values: 3,5,7,9
- Never use k=1 (sensitive to outlier). As value of k increases outlier impact decreases.

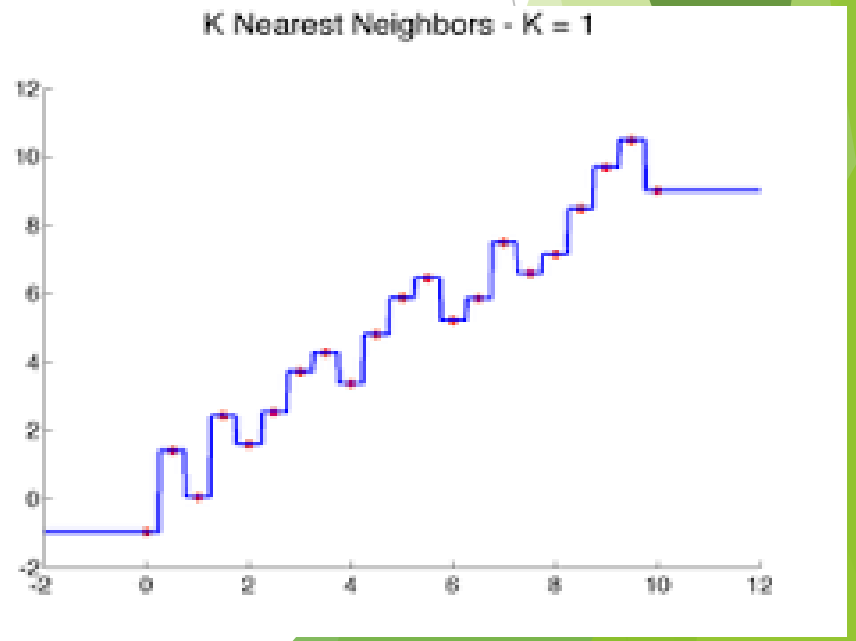




KNN regression

- Everything remains same except now it takes average value of k nearest data.
- For eg for house price it will find the nearest match and find avg price of its nearest matches and that price will be assign to the query point.
- Its also sensitive to outliers. Here also, never use k=1. As k increases impact of outlier decreases.
- Note that its non linear in nature and highly interpretable.





KNN Algorithm

- For each query point for which you want to know the class you have to find its distance (Euclidean distance n-dim) from all other points in data.
- Sort that based on distance in ascending order. So close point comes at start and far points comes at last.
- Pick a k value and filter first k points.
- For classification find majority class of nearest k points and assign that class to query point.
- For regression find mean value of nearest k points and assign that to query point.
- Algorithm is similar to its name k-nearest neighbors.
- Drawback: Very sensitive to k, lazy learner (expensive to run)