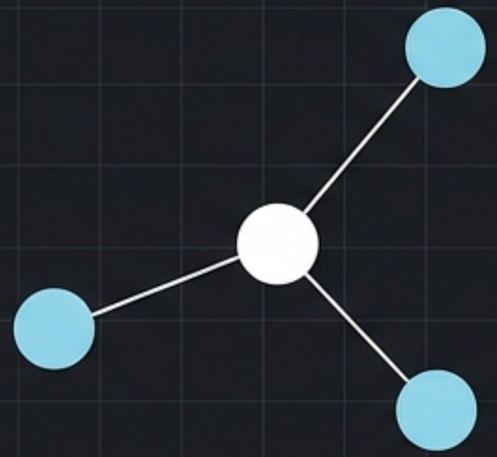
FROM INTUITION TO IMPLEMENTATION



K-Nearest Neighbors

Concept-to-Code Bridge

Concept → Mechanics → Optimization → Code

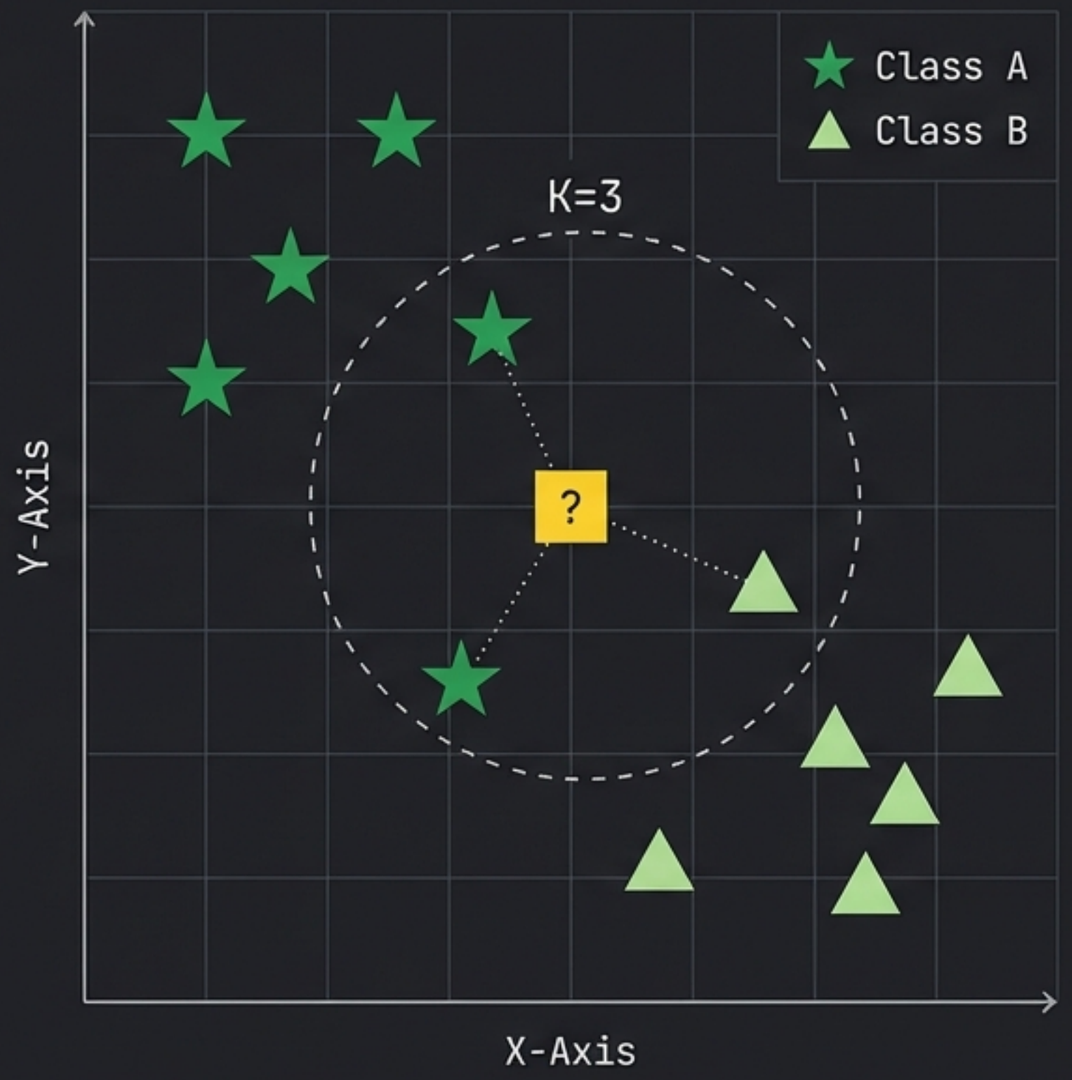
Mastering Supervised Learning

Proximity Equals Similarity

KNN operates on a simple social assumption: you are the average of your closest friends.

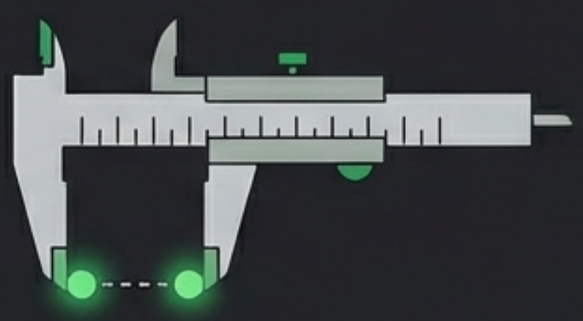
For any new data point, we determine its label by looking at its nearest neighbors.

- **Assumption:** Similar data points occupy close physical space.
- **Question:** Who is closest to the new point?
- **Answer:** The majority class (or average value) defines the new label.



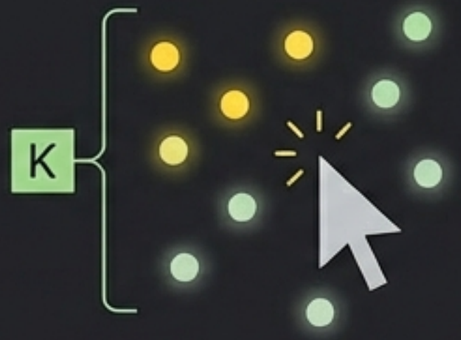
The Algorithm in Three Steps

MEASURE



Calculate the distance from the new point to every other point in the dataset.

SELECT



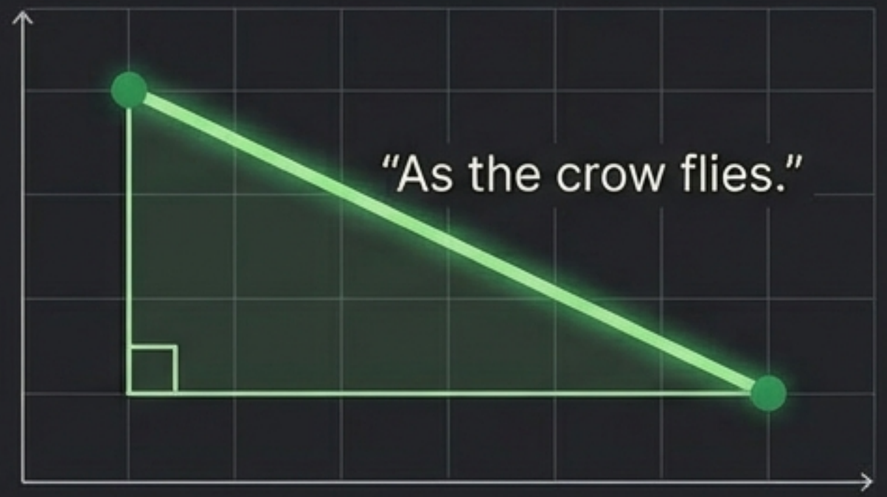
Identify and pick the top K points with the shortest distance values.

DECIDE

Vote on the class (Classification) or average the values (Regression).

The Mathematics of Distance

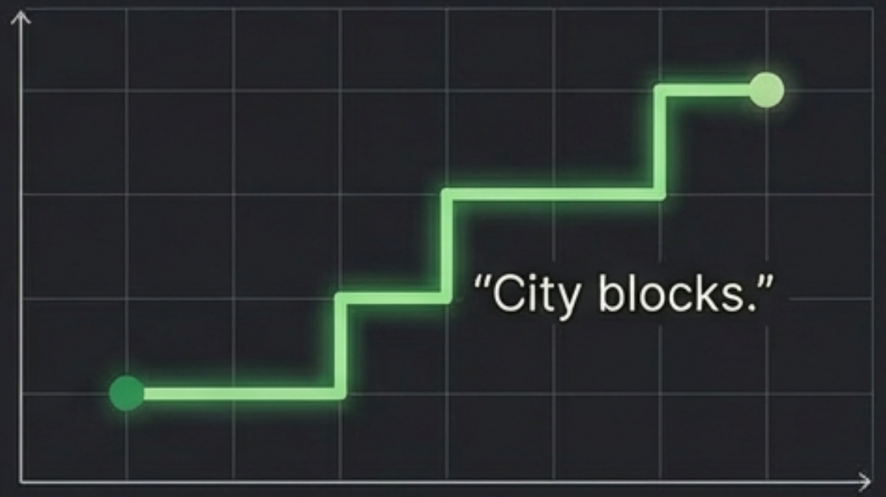
Euclidean Distance



$$d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

The straight-line distance. Standard for most continuous data.

Manhattan Distance



$$d = |x1 - x2| + |y1 - y2|$$

The grid-based distance. Useful for high-dimensional or grid-like structures.

One Algorithm, Two Decision Paths

Discrete Output. The most common class among neighbors becomes the prediction.

Continuous Output. The average value of neighbors becomes the prediction.

The Problem of Scale

The Brute Force Bottleneck: O(n)

By default, KNN calculates the distance to *every single point* in the dataset to find the closest neighbors.

If `N = 1,000,000` points, the algorithm must perform `1,000,000` calculations for every single prediction.

Consequence: As data grows, speed plummets. We need a smarter way to search.

Loading...

Structural Optimization: Skipping the Irrelevant

KD Tree (Boxes)

Splits data into boxes.
Creates a map of quadrants.
Good for low-dimensional data.

Ball Tree (Spheres)

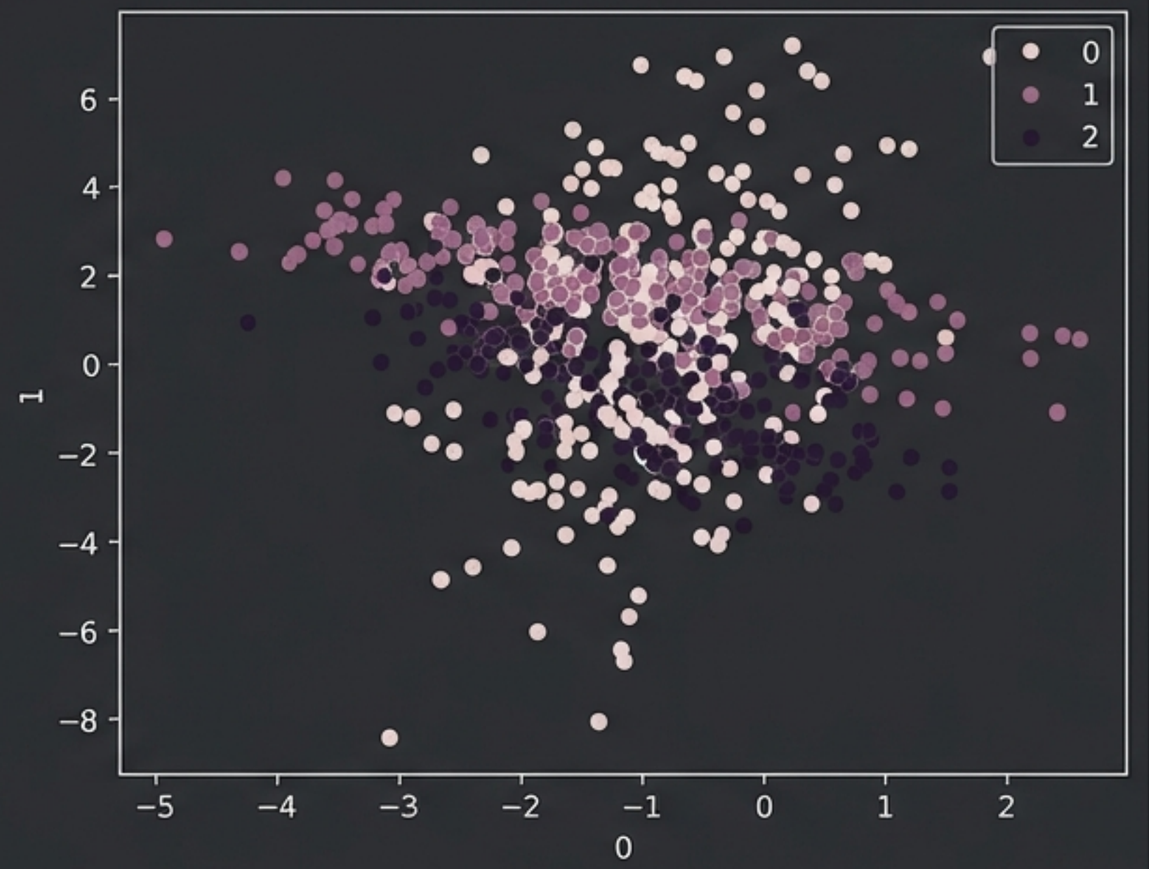
Organizes data into spheres (center + radius).
If a target is far from a sphere's edge, the algorithm skips all points inside.
Superior for high-dimensional data.

Lab Part I: Classification Setup

```
1.  X, y = make_classification(n_classes=3,
        n_features=10, n_samples=1000)
2.  # Split data: 70% Training, 30% Testing
3.  X_train, X_test, y_train, y_test =
        train_test_split(X, y, test_size=0.3)
```

Generating 3 distinct groups

Output



Training the Classifier

```python
from sklearn.neighbors import KNeighborsClassifier

# Define the model: Look for the 10 closest friends
knn_classify = KNeighborsClassifier(n_neighbors=10, algorithm='auto')

# Map the data
knn_classify.fit(X_train, y_train)

# Predict
y_pred = knn_classify.predict(X_test)
```

Hyperparameter K. The model considers the 10 nearest points.

Automatically selects Brute Force, KD Tree, or Ball Tree based on data structure.

Classification Performance

Accuracy Score

96%

0.956 on Test Data

Confusion Matrix Breakdown

Class 0	111 Correct	1 Wrong
Class 1	93 Correct	7 Wrong
Class 2	83 Correct	4 Wrong

Precision and Recall scores remain >0.92 across all classes.

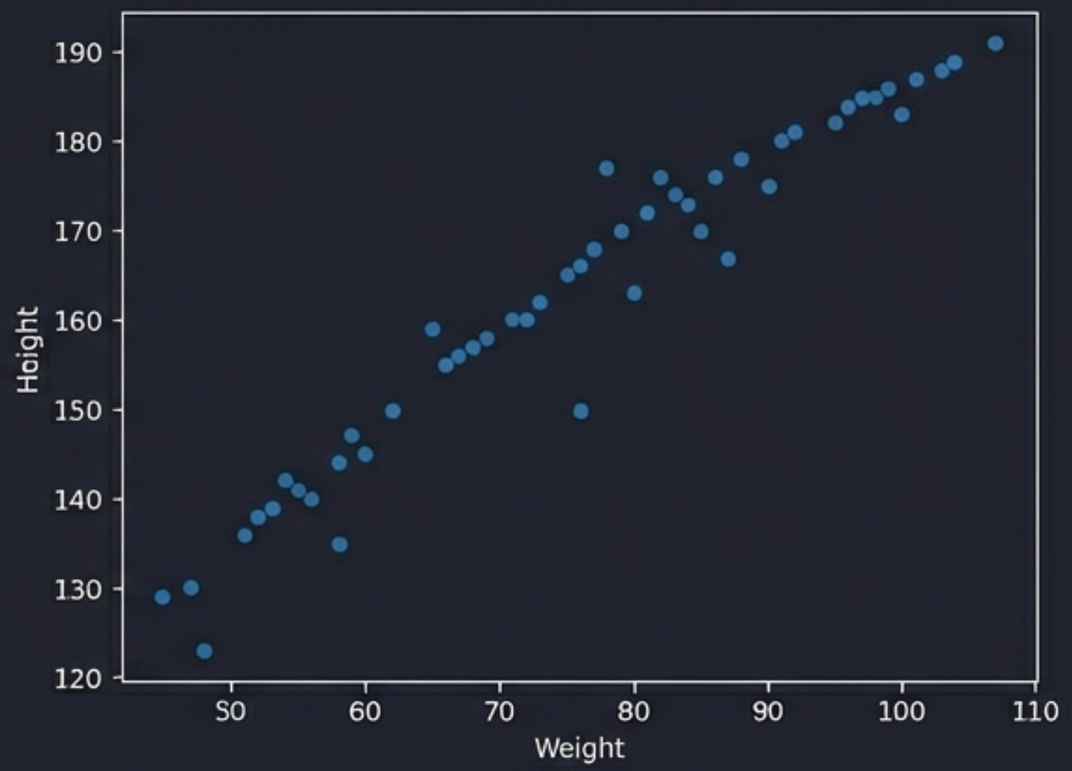
Lab Part II: Regression Setup

CODE EDITOR

```
1.  # Load dataset
2.  data = pd.read_csv('height-weight.csv')
3.  X, y = data['Weight'], data['Height']
4.
5.  # Reshape for Scikit-Learn
6.  X = np.array(X).reshape(len(X), 1)
```

Crucial Step: Converting 1D array to 2D matrix

OUTPUT TERMINAL/PLOT



Training the Regressor

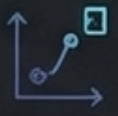
CODE EDITOR

```python
from sklearn.neighbors import KNeighborsRegressor

# Define the model: Look for 6 neighbors
knn_reg = KNeighborsRegressor(n_neighbors=6)

# Fit and Predict
knn_reg.fit(X_train, y_train)
y_pred = knn_reg.predict(X_test)
```

K=6. The output will be the average of these 6 points.

Regression Performance Metrics

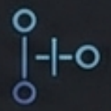
MSE (Mean Squared Error)



24.69

Average squared difference between estimated values and actual value.

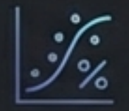
MAE (Mean Absolute Error)



3.54

On average, the height prediction deviates by 3.54 units.

R2 Score



0.94

Goodness of Fit. The model explairFit. The model explains 94% of the variance in the data.

Summary: The Power of Proximity

Intuitive

Based on the simple, explainable logic that similar data points cluster together.

Versatile

Handles both Classification (via Voting) and Regression (via Averaging) with a single algorithm.

Scalable

While brute force is slow, KD Trees and Ball Trees allow KNN to scale to massive, high-dimensional datasets.