DATA, MSML, BIOI 602 Principles of Data Science

Introduction to Machine Learning

[Slides from Fardina Fathmiul Alam]

The Problem:

A lady walks into your bank and requests a loan.

- If you give her a loan and she pays it back, you will **make money**.
- If you give her a loan and she doesn't pay it back, you will **lose money.**

Ques: How do you know whether to give this woman a loan?

Let’s say: You Have Historical Data:

Name	**Income**	**Wealth**	**Job**	**Paid Back Loan**
Alice	100,000	300,000	Finance	Yes
Bob	150,000	10,000	Gambler	No
Eve	500,000	1,500,000	Hitman	Yes
Fardina	100,000	-100,000	Professor	No

Historical Data:

Name	Income	Wealth	Job	Paid Back Loan
Alice	100,000	300,000	Finance	Yes
Bob	150,000	10,000	Gambler	No
Eve	500,000	1,500,000	Hitman	Yes
Fardina	100,000	-100,000	Professor	No

New Case:

Joan	100,000	350,000	Finance	?

Obvious Solution:

Name	Income	Wealth	Job	Paid Back Loan
Alice	100,000	300,000	Finance	Yes
Bob	150,000	10,000	Gambler	No
Eve	500,000	1,500,000	Hitman	Yes
Max	100,000	-100,000	Professor	No

Joan	100,000	350,000	Finance	?

This is a **Classification Problem**

A classification problem has:

- A **target variable** you want to predict
- A **set of historical data** where that target **label** is **known**
- **New data** where that target variable is **unknown**
- A **model** is a mathematical object that takes data *where the label is **unknown** and assigns it a label*

What is Classification

A type of **supervised machine learning**.

Goal: Assign categories or labels to data points based on patterns in the data.

- In our case, categorizing loan applicants (Paid Back Loan) into "Yes" or "No" labels.

What is a Model?

A model is a mathematical object that

- **takes** your training data,
- learns **a function mapping data** -> labels,
- and then is **able to take** in unlabeled data and **assign them** a label

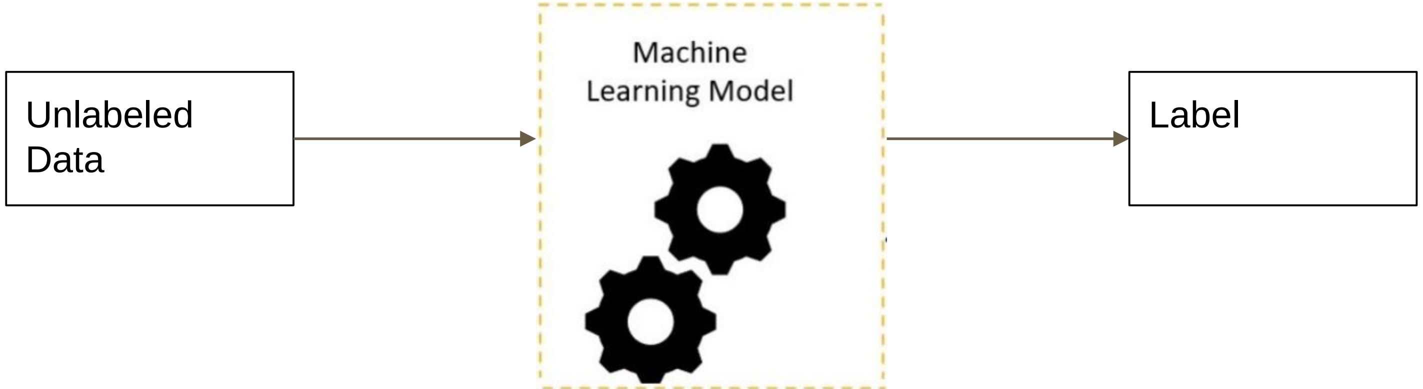
f(some_new_data) => new_label

Here, f(some_new_data) represents a function that takes new data as input.

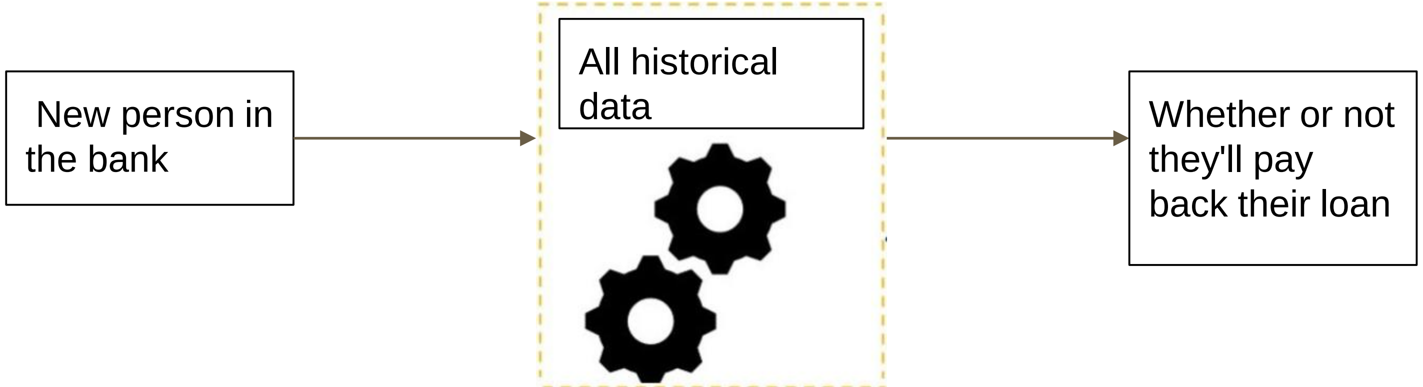
f(x) = x_wealth > 100,000

f(x) is a model!

Our First Model!



Our First Model!



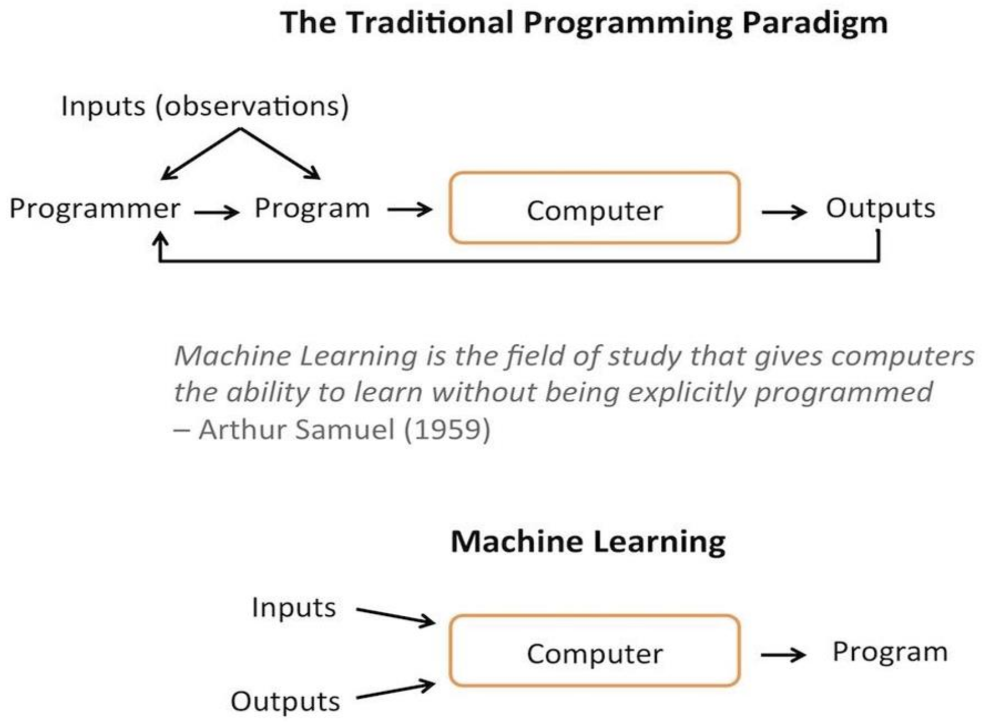
Let’s talk about “ML” briefly

What is Machine Learning (ML)

Machine learning is a subfield of artificial intelligence (AI) that focuses on the development of algorithms and statistical models that enable computer systems to learn and improve their performance on a specific task or problem without being explicitly programmed.

In other words, machine learning allows computers to make predictions or decisions based on patterns and insights derived from data.

Traditional programs are explicitly programmed with rules, while machine learning programs learn from data to make predictions or decisions autonomously.

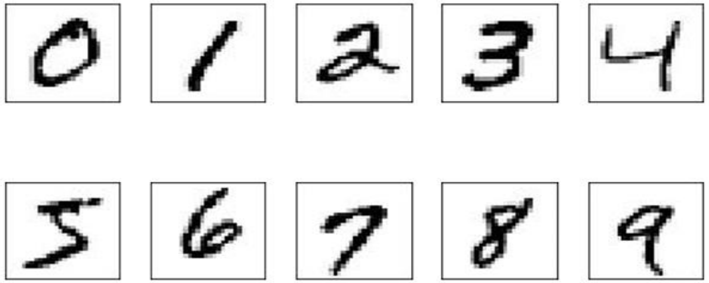


Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed
– Arthur Samuel (1959)

"A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

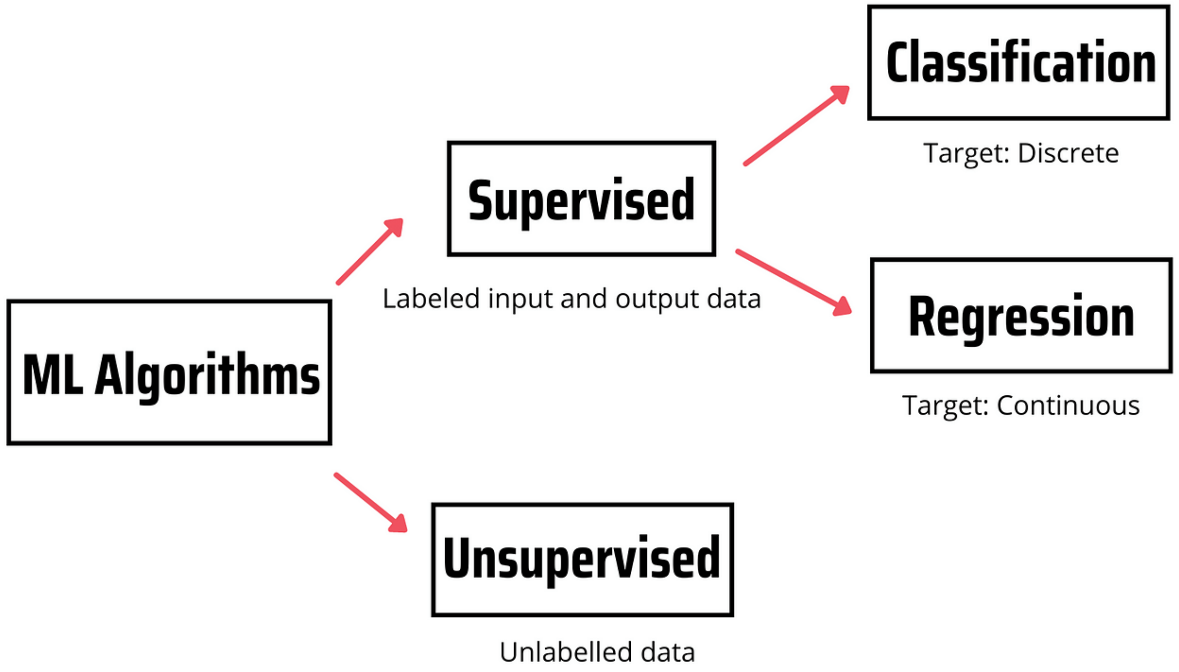
— Tom Mitchell, Professor at Carnegie Mellon University

Handwriting Recognition Example:



- Task T: Recognize Handwriting
- Performance measure P : How accurately the program can recognize the handwritten characters or words.
- Training experience E: Data the program learns from

Types of Learning



Types of Learning

- ❑ **Supervised Learning:** T**he algorithm learns from labeled data,** where each example in the training set is associated with a known target or output. It is used for tasks like classification and regression.

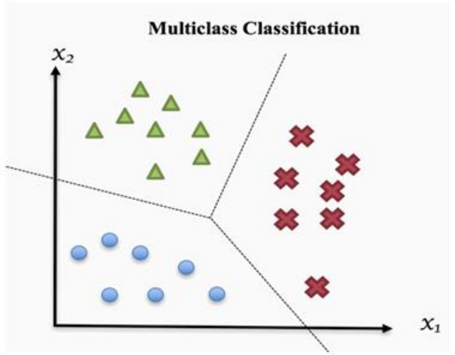
- ❑ **Unsupervised Learning: The algorithm learns from unlabeled data and tries to identify patterns or structures within the data**, such as clustering similar data points or reducing data dimensionality.

- ❑ **Reinforcement Learning:** The algorithm learns to make a sequence of decisions or actions in an environment to maximize a reward signal (uses rewards and punishments to learn a task). It's often used in robotics, game playing, and autonomous systems.

Types of Learning: Examples

- ❑ **Supervised Learning:** learn to recognize whether an email is spam or not based on a dataset of labeled emails.
- ❑ **Unsupervised Learning:** Group similar customer profiles together in a retail dataset without knowing in advance what those groups represent.
- ❑ **Reinforcement Learning:** The computer learns by taking actions in an environment and receiving rewards. Think of a self-driving car learning to navigate roads safely by getting positive feedback (rewards) for following traffic rules and avoiding accidents.

Types of Learning



Supervised Learning.

- Learn a **function** from examples of its inputs and outputs.
- E.g., An agent is presented with many camera images and is told to learn which ones contain busses.
- Agent learns to map from images to Boolean output 0/1 of bus not present/present.
- Learning decision trees is a form of supervised learning.

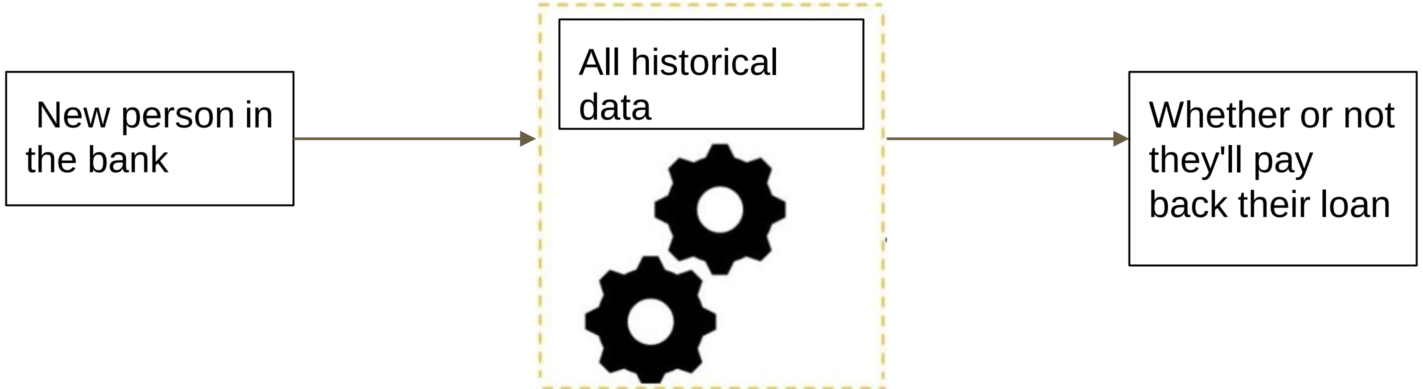
Unsupervised Learning.

- Learn patterns in the input with no output values supplied.
- E.g,: Identify communities on the Internet.

Reinforcement Learning.

- Learn from reinforcement (occasional rewards).
- E.g., An agent learns how to play backgammon or go or chess against itself.

Back to Our First Model!

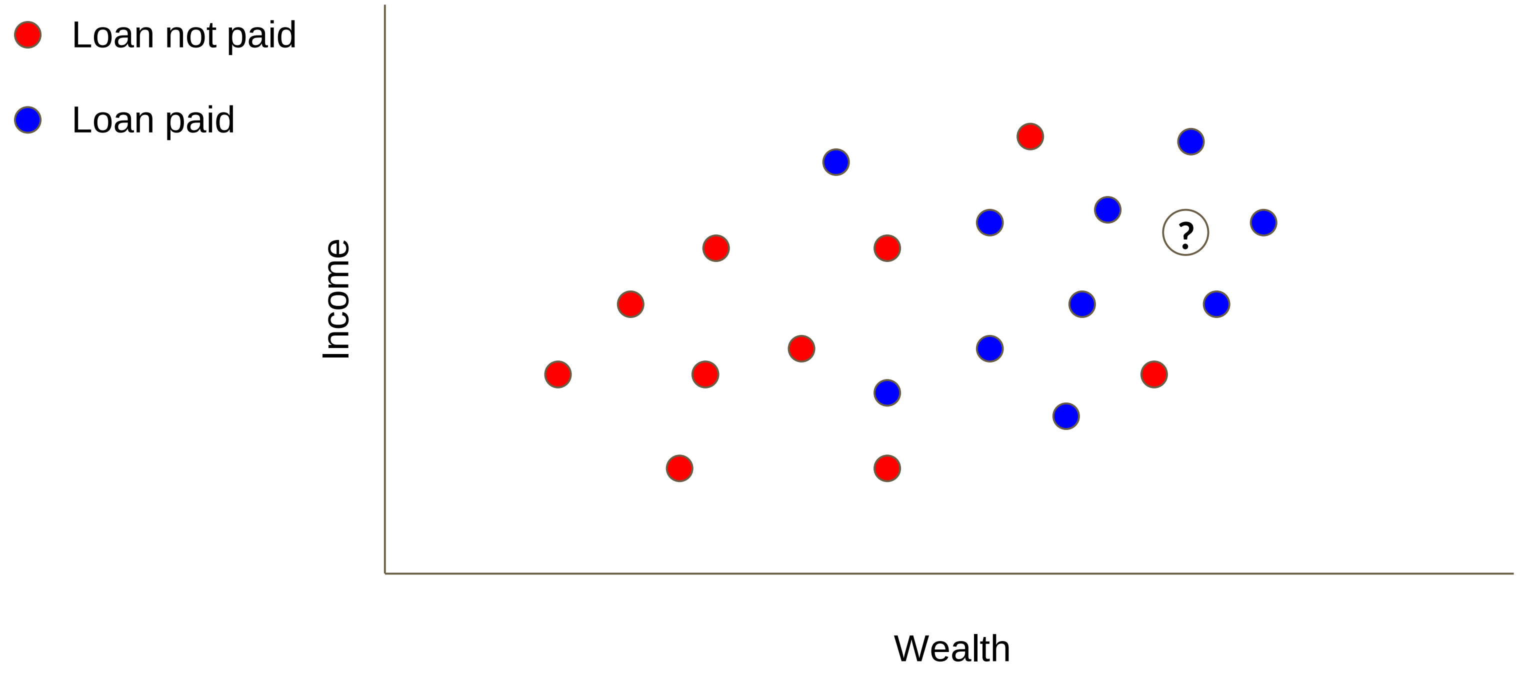


Q: What type of ML model it is?

Our First Algorithm

Let's Dumb It Down Even Further

Plot the datapoint:

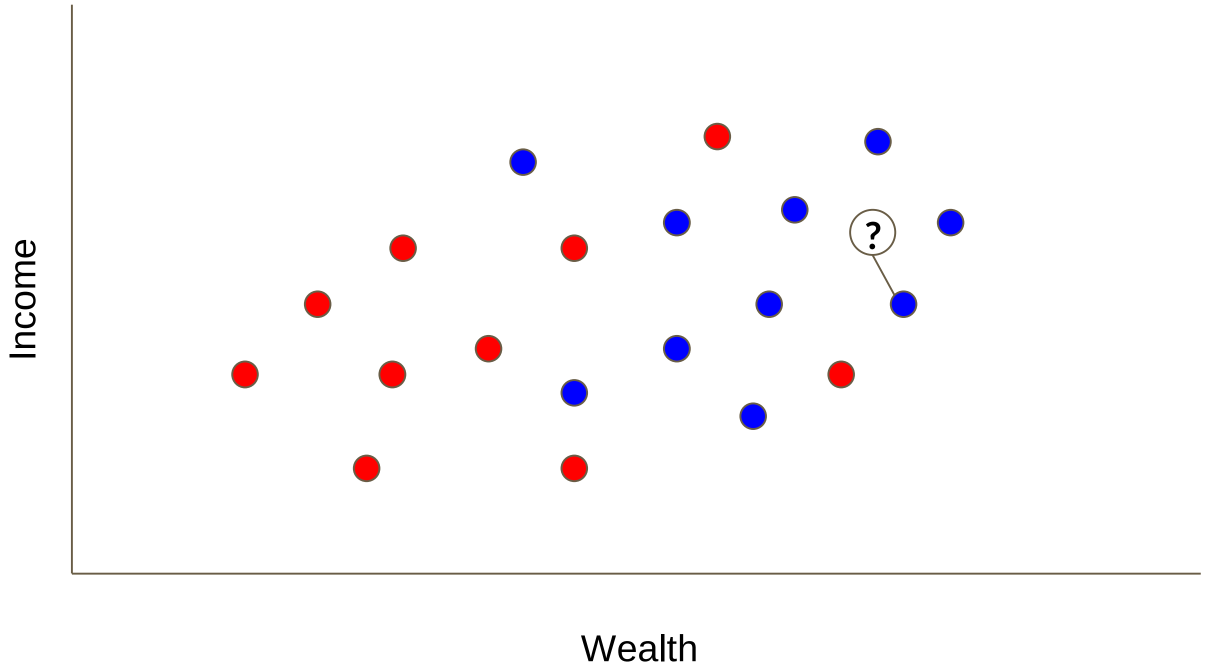


Let's Dumb It Down Even Further

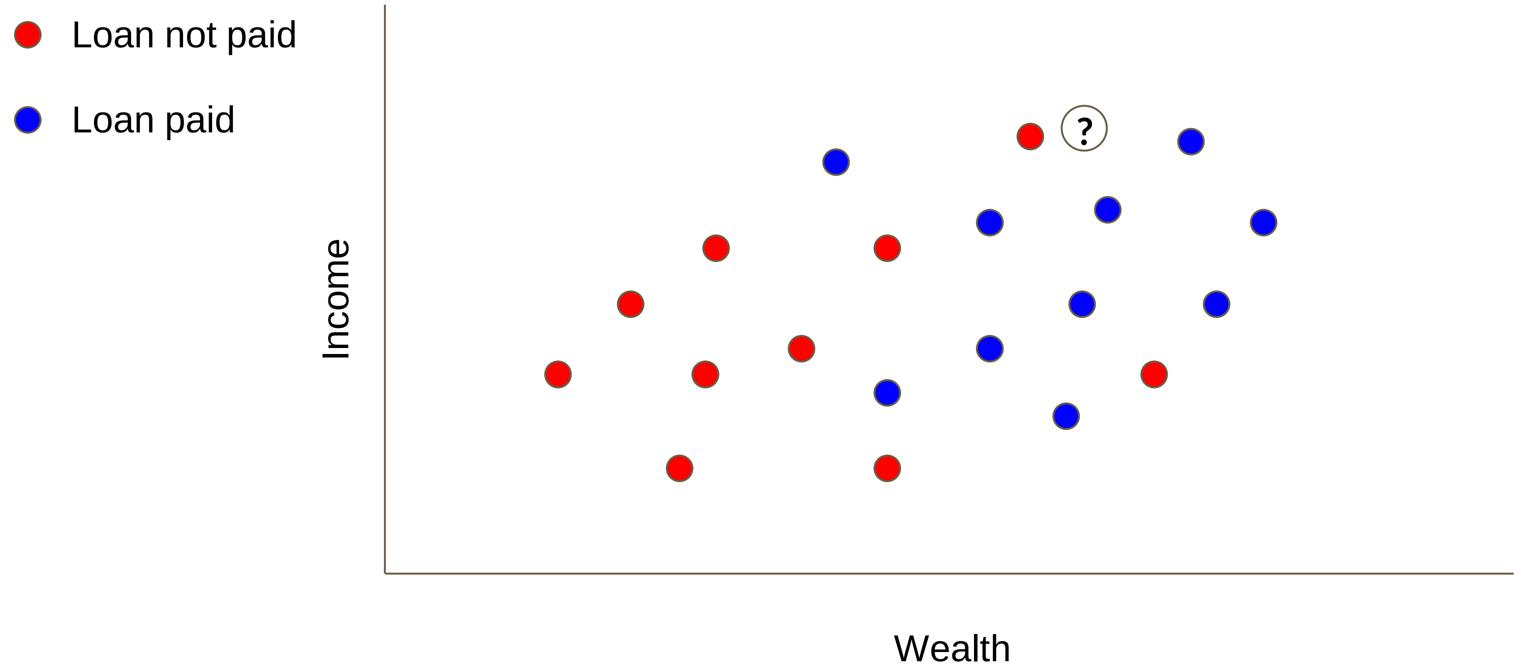
● Loan not paid

● Loan paid

Predict the nearest neighbor for the new datapoint and take the vote if it is a classification problem !



Let's Dumb It Down Even Further



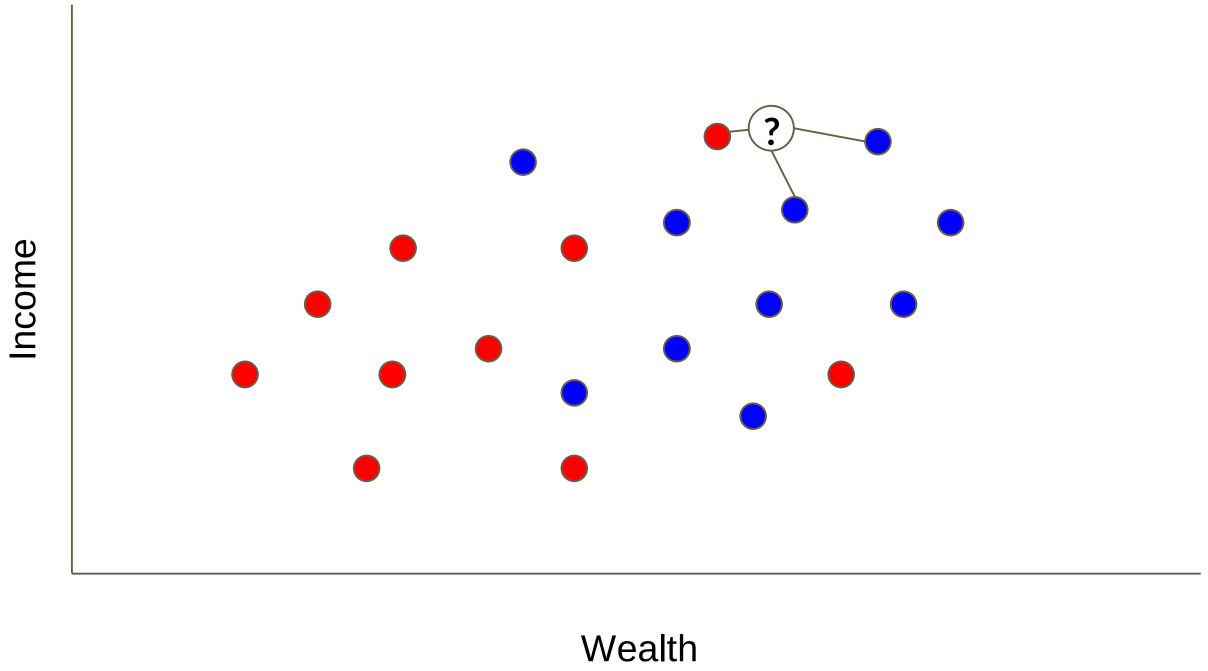
How to improve accuracy? Let’s look at more than one neighbours

Let's Dumb It Down Even Further

● Loan not paid

● Loan paid

For Regression:
Average the 3
nearest neighbors

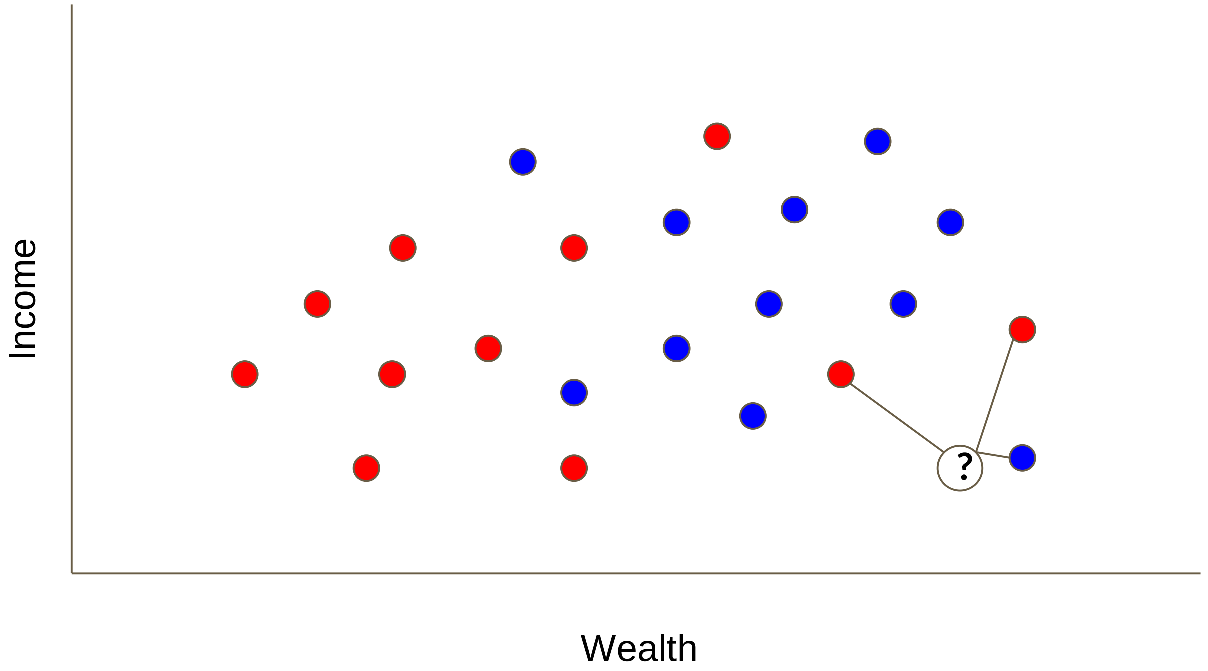


Let's Dumb It Down Even Further

🔴 Loan not paid

🔵 Loan paid

What about this? Do you want to classify as "blue" or "red"?



The k-nearest neighbors (KNN)

We just talk about “the k-nearest neighbors (KNN)” algorithm

A simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.

Intuition for Nearest Neighbor Classification

Simple idea

- Store all training examples
- Classify new examples based on most similar training examples

Remember, KNN predicts outcomes based on the **similarity or proximity between data points**.

Components of a k-NN Classifier

- **Distance metric**
    - How do we measure distance between data points/ instances?

- **The k hyperparameter**
    - How large a neighborhood should we consider?
    - **The "K" in KNN:** a user-defined parameter that specifies the number of neighboring data points to consider when making predictions.
        - Choosing "K": K is typically an odd number to prevent ties in decision-making.

In general, in KNN

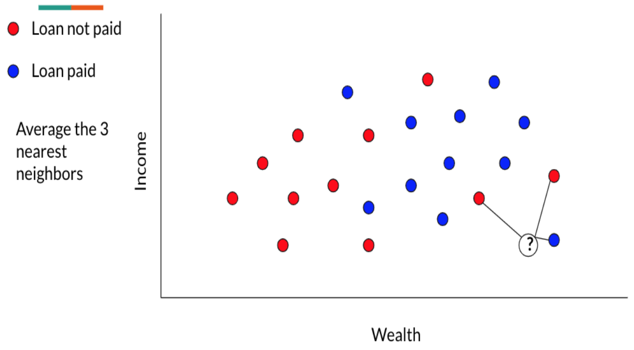
Given the set K of the K nearest neighbors and point p

- KNN Algorithm: KNN identifies the K most similar individuals (neighbors) to the person for whom we want to make a prediction (point p).

Let's represent loan repayment behavior or result using labels:

a. **Label -1:** Represents someone who did not pay back the loan.

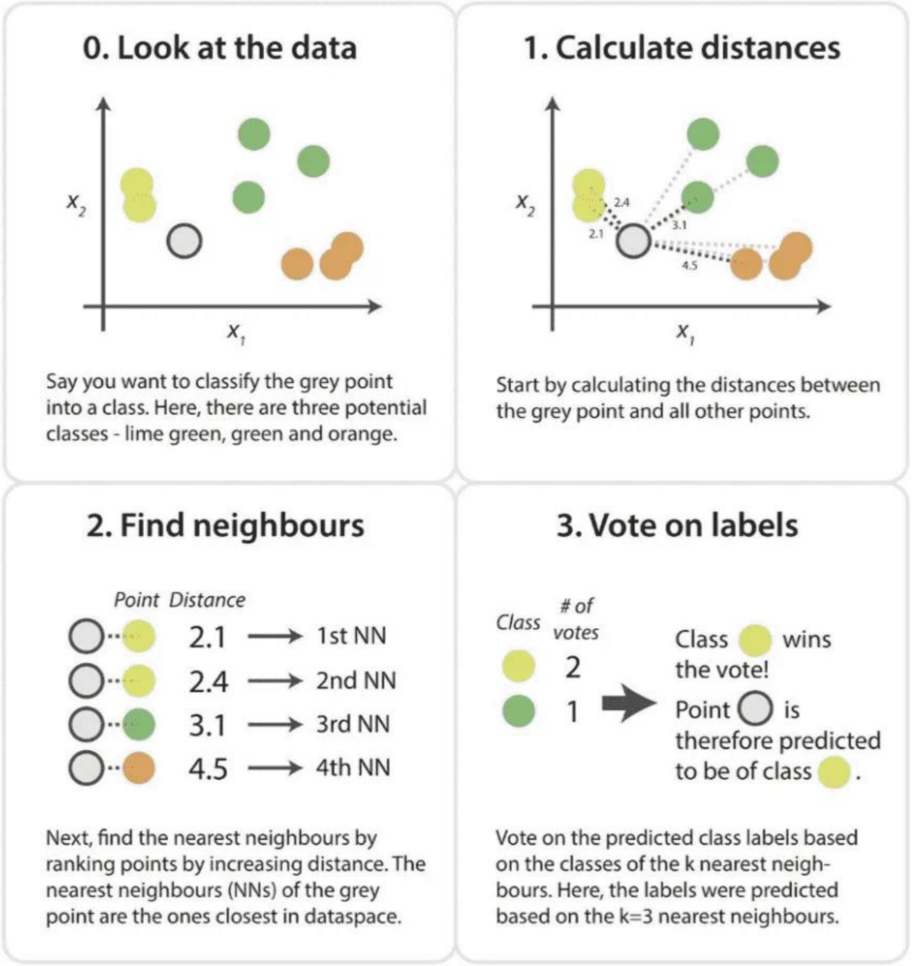
b. **Label 1:** Represents someone who did pay back the loan.



KNN Algorithm

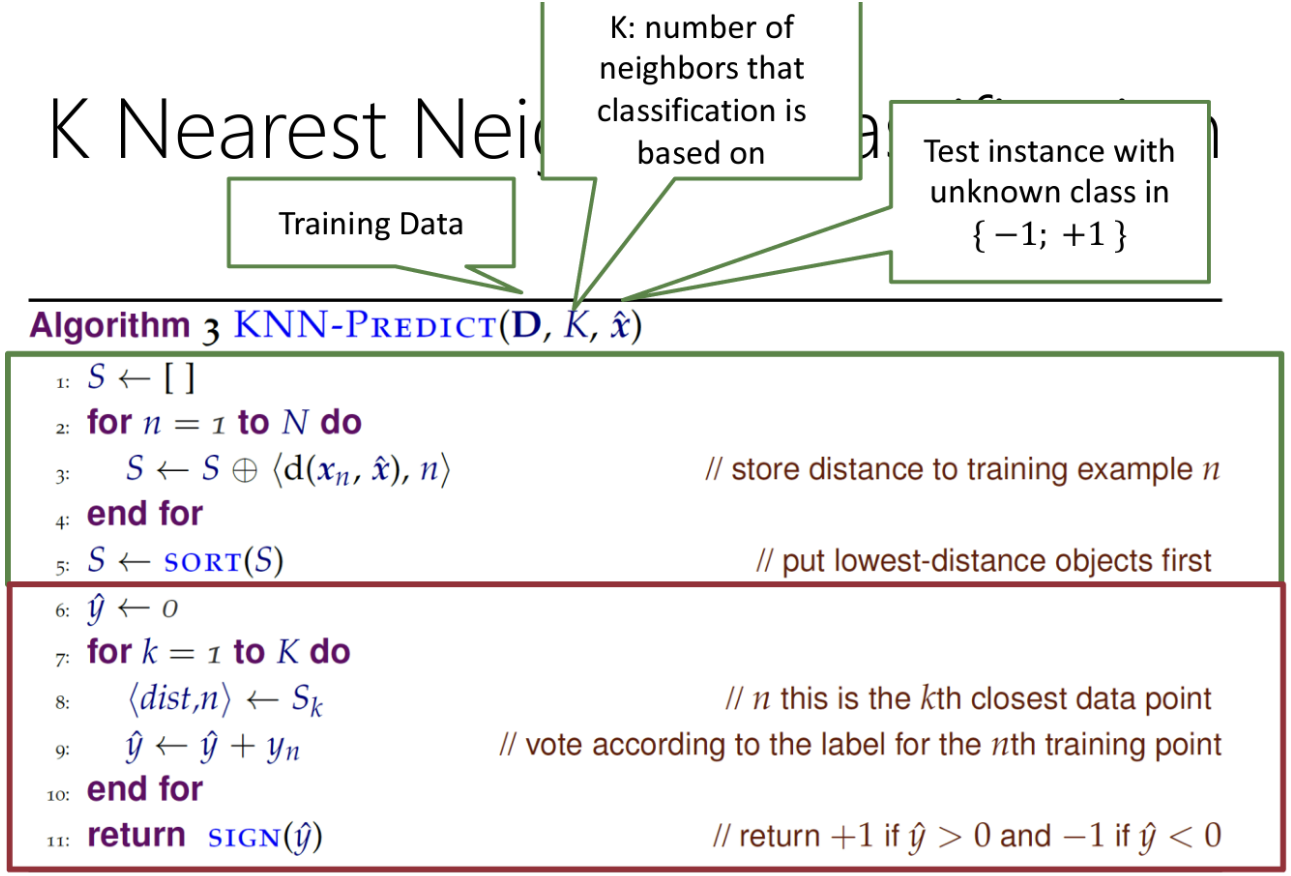
Step-1: Select the number K of the neighbors
Step-2: Calculate the Euclidean distance of K number of neighbors
Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.
Step-4: Among these k neighbors, count the number of the data points in each category.
Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Our KNN model is ready.



0. Look at the data
x_2
x_1
Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.
1. Calculate distances
2.4
3.1
2.1
4.5
Start by calculating the distances between the grey point and all other points.
2. Find neighbours
Point Distance
2.1 → 1st NN
2.4 → 2nd NN
3.1 → 3rd NN
4.5 → 4th NN
Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.
3. Vote on labels
Class
# of votes
2
1
Class wins the vote!
Point is therefore predicted to be of class .
Vote on the predicted class labels based on the classes of the k nearest neighbours. Here, the labels were predicted based on the k=3 nearest neighbours.

KNN Algorithm Pseudocode


K Nearest Nei
K: number of neighbors that classification is based on
Training Data
Test instance with unknown class in $\{-1; +1\}$

Algorithm 3 KNN-PREDICT($\mathbf{D}$, K, $\hat{x}$)

```
1:  $S \leftarrow [\,]$
2:  for $n = 1$ to $N$ do
3:      $S \leftarrow S \oplus \langle \mathrm{d}(x_n, \hat{x}), n \rangle$      // store distance to training example $n$
4:  end for
5:  $S \leftarrow$ SORT($S$)      // put lowest-distance objects first
6:  $\hat{y} \leftarrow 0$
7:  for $k = 1$ to $K$ do
8:      $\langle dist, n \rangle \leftarrow S_k$      // $n$ this is the $k$th closest data point
9:      $\hat{y} \leftarrow \hat{y} + y_n$      // vote according to the label for the $n$th training point
10: end for
11: return SIGN($\hat{y}$)      // return $+1$ if $\hat{y} > 0$ and $-1$ if $\hat{y} < 0$
```

Variations on k-NN: Weighted voting

Weighted voting Default

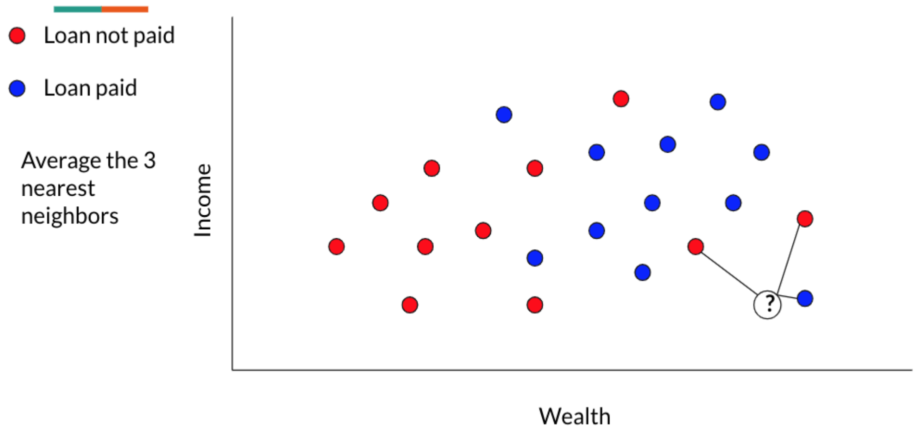
- All neighbors have equal weight
- Extension: weight neighbors by (inverse) distance
  - NEXT TOPIC: Weighted KNN

A Modified Version: Weighted KNN

In a Standard KNN, prediction is based on the majority class among the K nearest data points **and Equal weight is given to all K nearest neighbors.**

Ex: If K=5 and the nearest neighbors are {1,1,1,−1,−1}, the prediction would be 1 (majority class).

Weighted KNN is Similar to KNN, but the nearest k points are assigned a weight based on their distance.

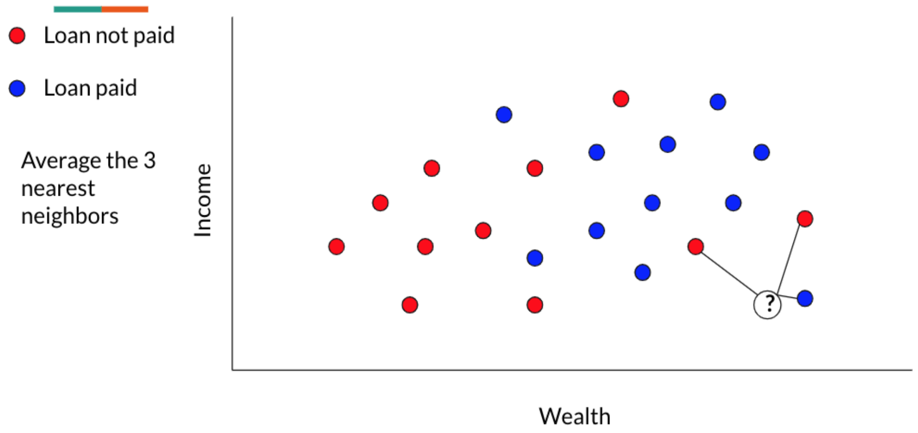


A Modified Version: Weighted KNN

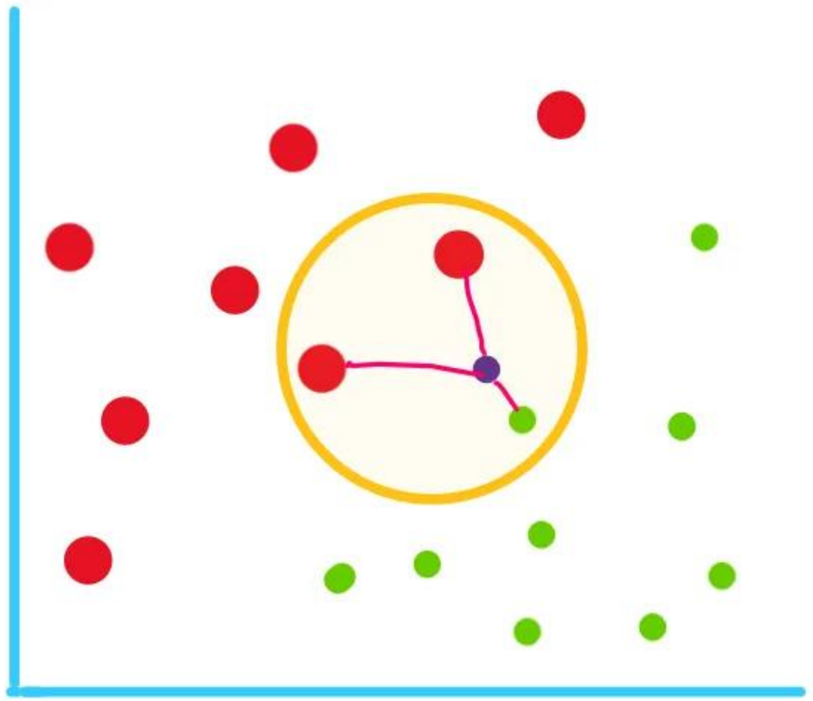
In a Standard KNN, prediction is based on the majority class among the K nearest data points **and Equal weight is given to all K nearest neighbors.**

Ex: If K=5 and the nearest neighbors are {1,1,1,−1,−1}, the prediction would be 1 (majority class).

The intuition behind weighted KNN is to give more weight to the points which are nearby and less weight to the points which are farther away.



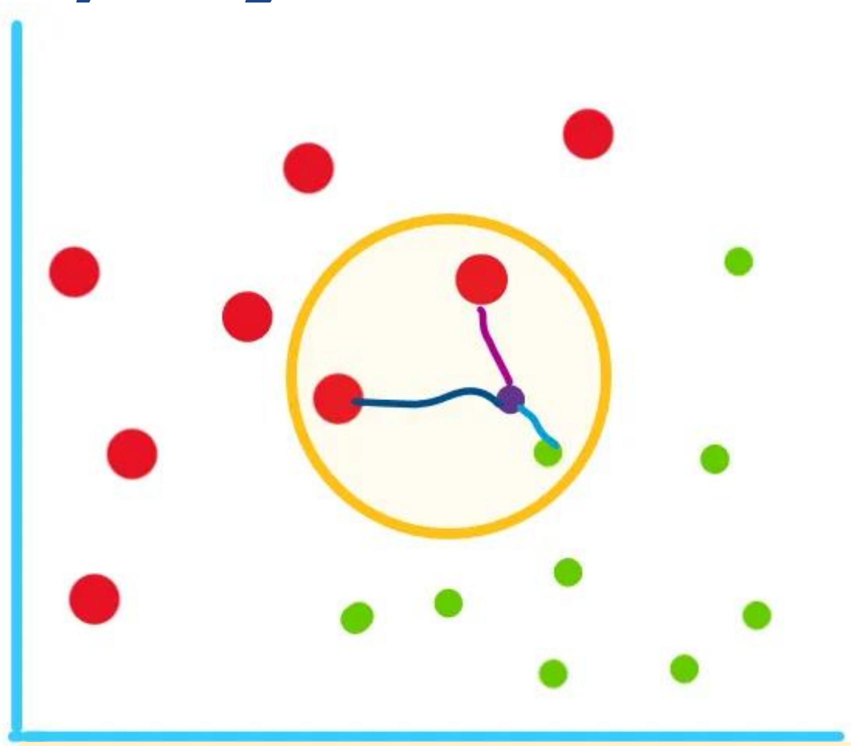
Why Weighted KNN instead of a standard KNN ?



When using basic k-NN algorithm, the new point is predicted to be ‘red’ class whereas in real, the point is closer to the ‘green’ class

This basic k-NN algorithm works quite well in many cases, but it **treats all neighbors equally**.

Why Weighted KNN instead of a standard KNN ?



Weighted k-NN introduces the concept of **assigning different weights to neighbors based on their proximity** to the query point, which can lead to improved performance.

In weighted k-NN, as we consider the weights of the points the predicted value will be ‘green’ as the point has higher weight compared to the other points.

Summary: KNN vs Weighted KNN

KNN	**Weighted KNN**
It treats all neighbors equally	Instead of treating all k-nearest neighbors equally, assign different weights to them based on their distance from the query point.
For classification tasks: count the occurrences of each class among the k-nearest neighbors and choose the class with the **highest count(mode)** as the predicted class.	**For classification tasks:** when determining the predicted class, the contributions of neighbors are scaled by their weights. Closer neighbors have a stronger influence on the prediction.
For regression tasks: take the **average (mean)** of the values associated with the k-nearest neighbors as the predicted value.	**For regression tasks:** the values associated with neighbors are multiplied by their weights before averaging, giving more importance to closer neighbors in the prediction.

Weighted KNN Equation

Weighted KNN not only considers the nearest neighbors but also how close or far away they and assigns different weights to the K nearest neighbors based on their distances to point p.

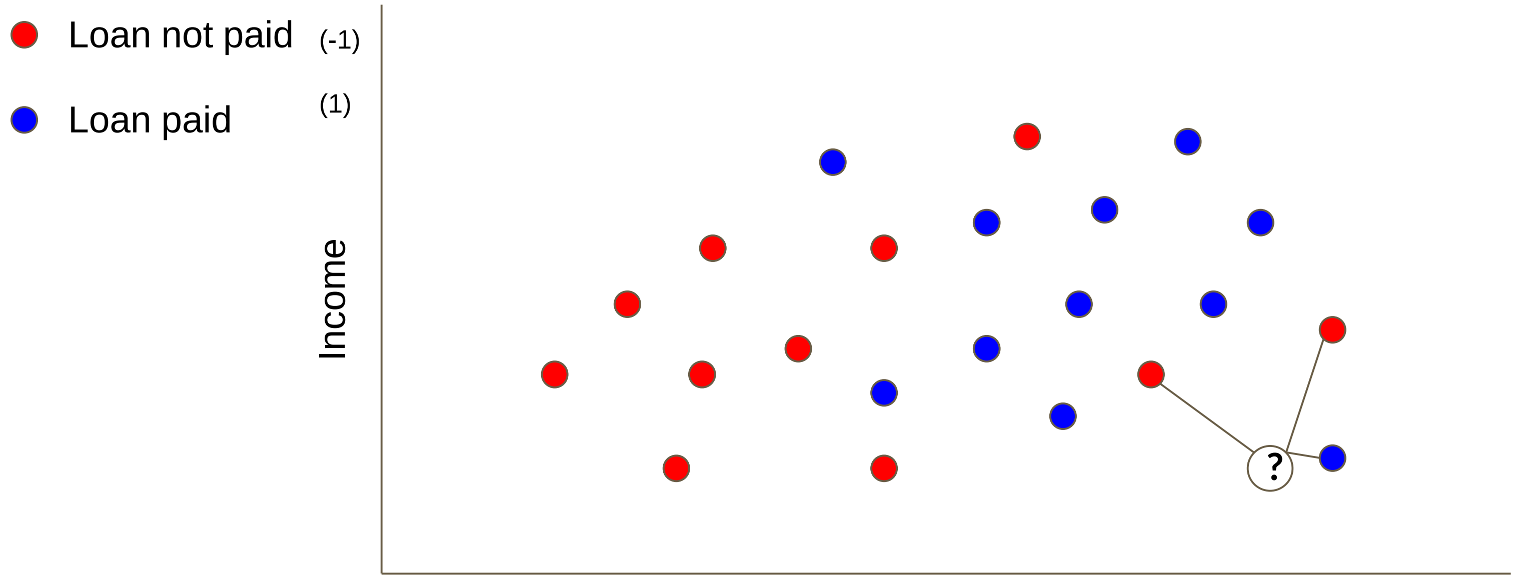
- How Assign? the weights are inversely proportional to the distances
- Closer neighbors have a stronger influence on the prediction.
    - Why? closer neighbors get higher weights, while farther neighbors get lower weights

label (or value) of the k-th nearest neighbor.

$$y = \sum_{k \in K} \frac{1}{dist(k,p)} * k_{label}$$

predicted label or value for a new data point p.

Let's Dumb It Down Even Further

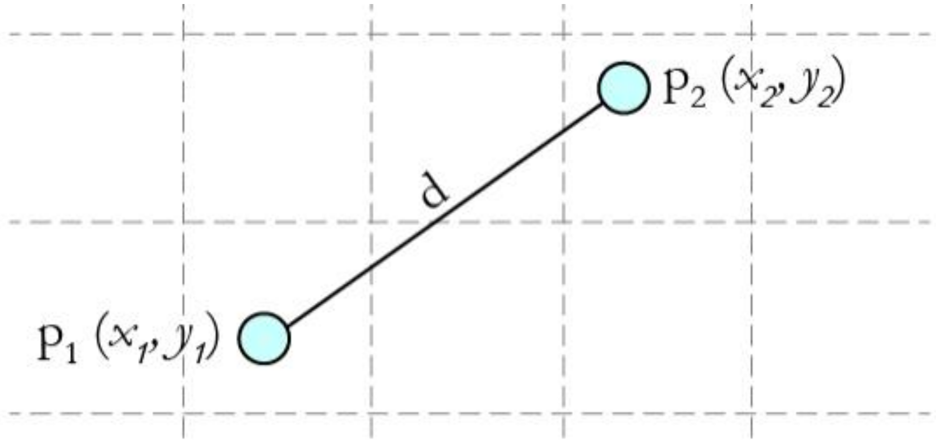


$$y = (\frac{1}{bigDistance} * -1) + (\frac{1}{bigDistance} * -1) + (\frac{1}{smallDistance} * 1)$$

Hold on--What do we even mean by distance?

Distance is a crucial concept in both KNN and Weighted KNN.

- It measures the similarity or proximity between data points.
  - the numerical measure of how far apart two data points are in the feature space.



Why Distance Matters?

Many of the Supervised and Unsupervised machine learning models such as K-Nearest Neighbor and K-Means depend upon the distance between two data points to predict the output.

Example:

In Weighted KNN, the distance between a data point (e.g., point p) and its K nearest neighbors determines the weighting.

- **Closer** neighbors have higher influence, while **distant neighbor**s have less impact.

Different kind of distance functions

The choice of distance metric depends on the nature of the data and the problem at hand.

Different kind of distance functions:

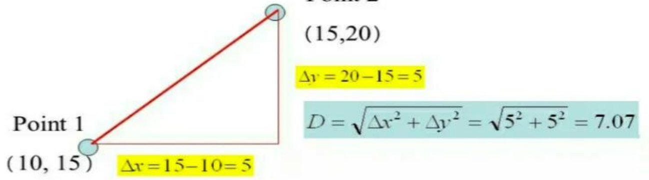
- Manhattan Distance
- Euclidean Distance
- Hamming distance
- Cosine Similarity

Euclidean Distance (L2 Distance)

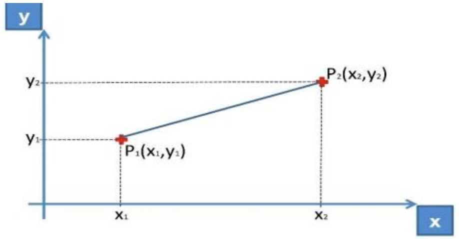
Euclidean Distance represents the shortest distance between two points.

- Most frequently used distance metric in machine learning.
- Measures the geometric (straight-line) distance between points, considering differences along each dimension.
- Best for continuous numeric data as it considers the magnitude or size of differences.
    - larger differences have a greater impact on the calculated distance.
- Sensitive to Outliers

$$\text{Euclidean distance} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$



$$d(x,\ y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$



Euclidean Distance between P_1 and $P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

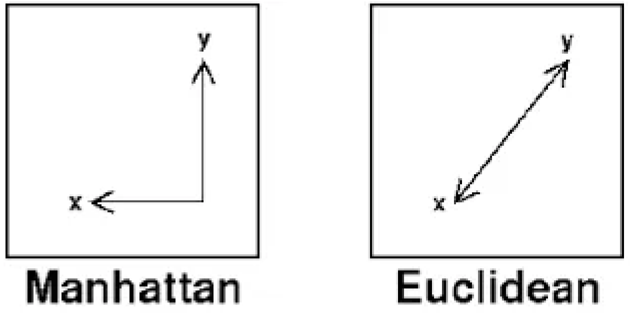
Manhattan Distance (L1 distance or city block or taxicab distance)

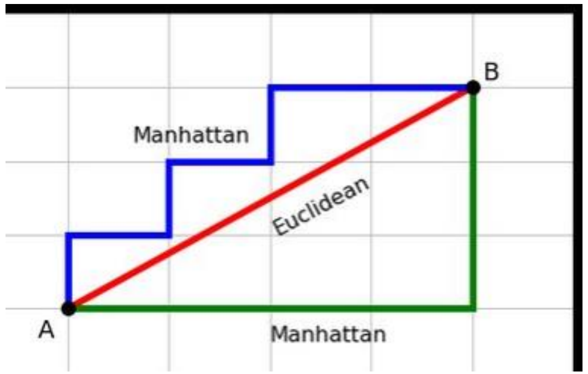
Manhattan Distance measure the distance between two points on a grid, considering only vertical and horizontal movements (mimicking a taxicab navigating city streets).

- Imagine walking through a city with square blocks and only being able to move along the streets, not diagonally. The Manhattan distance would be the total number of blocks you'd have to walk to get from one point to another.

The distance is calculated by summing the **absolute values of the changes** in coordinates between the two points.

$$d = \sum_{i=1}^{n} |x_i - y_i|$$





Manhattan Distance (L1 distance or city block or taxicab distance)

- **Preferred in very high-dimensional situations where dimensions are not equally important.**
  - Reduces influence of irrelevant dimensions: Focuses on absolute differences, giving less weight to the magnitude of values (In high-dim. spaces, many dimensions may not be equally relevant).

    .
- **Suitable for data with categorical features**
  - Ignores the magnitude of differences, focusing on directional changes.
    - Calculates distance based on the number of moves or changes needed

Compared to the Euclidean distance, which calculates the straight-line distance between two points, the Manhattan distance provides a more realistic measure of distance in situations where movement is restricted to specific directions (like navigating along the grid of a city, moving only horizontally and vertically.)

Hamming Distance

Given two binary strings, count the number of different bits.

- Used for strings (comparing binary strings or categorical data.)!
- Commonly applied in text classification and DNA sequence analysis.

Cosine Similarity

A metric used to measure how similar two vectors are, especially in high-dimensional spaces. It calculates the cosine of the angle between the two vectors in a multi-dimensional space.

- **Purpose:** Used to check if vectors are pointing in the same direction.
- **Applications:** Commonly used for comparing the similarity of document vectors, text data, and high-dimensional data.
- **Range:** Values range from -1 (completely dissimilar or pointing in opposite directions) to 1 (identical or similar in direction).

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Summary: The choice of metric

The choice of distance metric depends on the problem, data type, and dimensionality.

- Manhattan Distance for high-dimensional feature vectors with categorical data.
  - Treats each category as a separate dimension and simply sums the absolute differences between categories, making it suitable for categorical data.
- Euclidean Distance for continuous numerical data.
- Hamming Distance for binary data like document comparisons.
- Cosine Similarity for text mining, recommendation systems, and clustering.

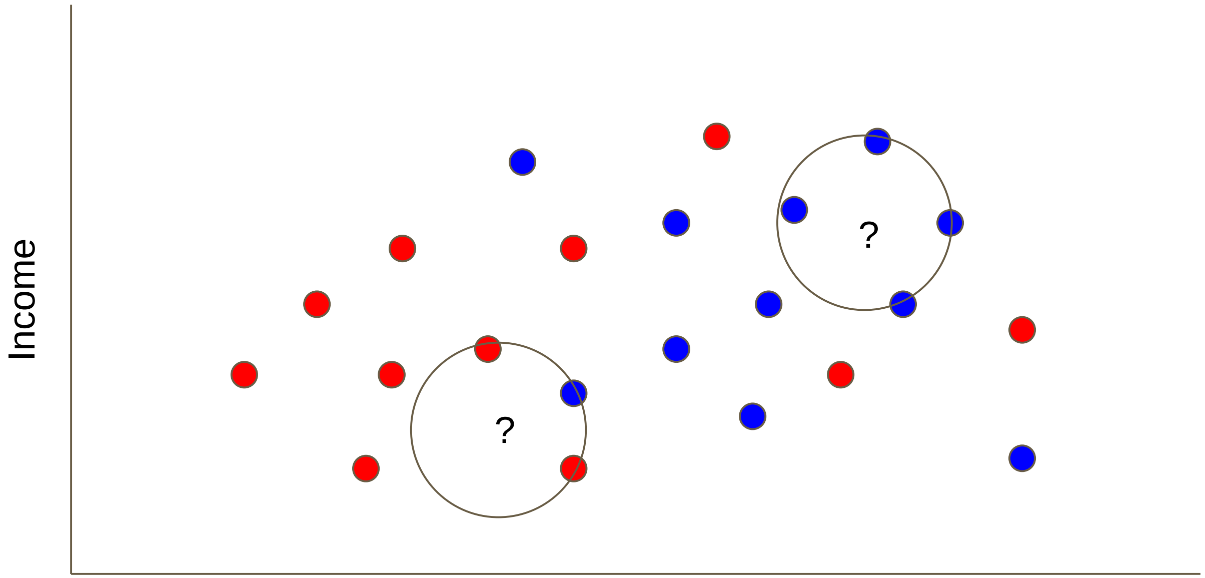
Back to KNN in more Details

Variations on k-NN: Spherical K-Nearest Neighbors

Euclidean distance doesn't accurately reflect "nearest neighbors" on a sphere because it doesn't account for the sphere's curvature. On a globe, the shortest path between two points isn't a straight line but an arc along the surface.

Spherical K-Nearest Neighbors (Spherical KNN) is a variation of the traditional K-Nearest Neighbors (KNN) algorithm.

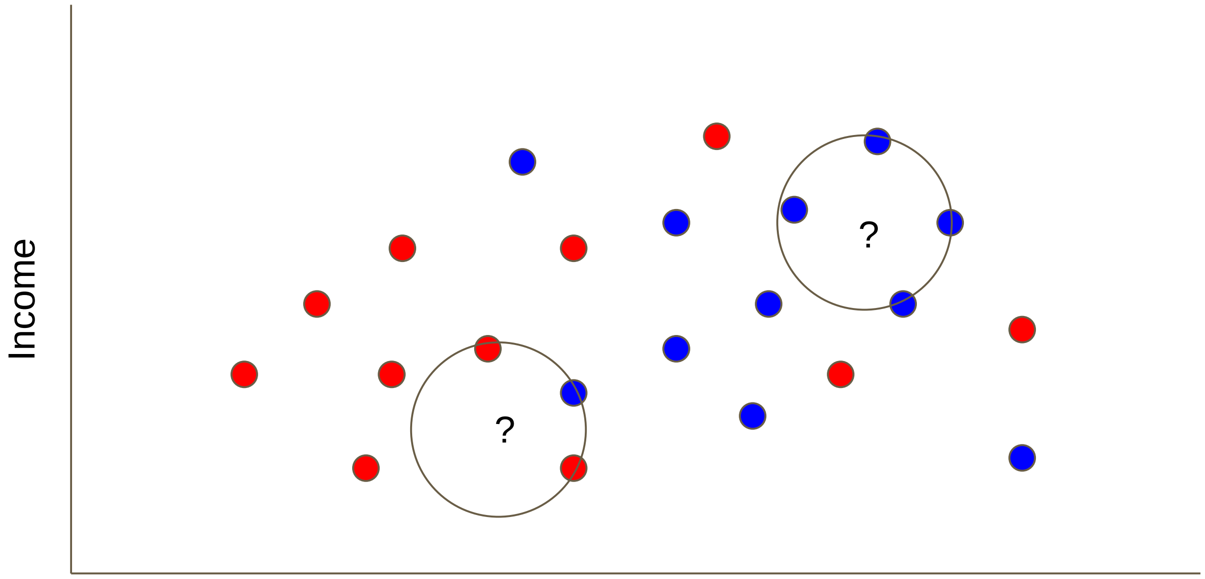
- Specifically designed for datasets where the underlying data distribution is spherical or roughly spherical in shape.



Spherical K-Nearest Neighbors

You can also predefine sphere sizes, and then let everything within the sphere vote!

- instead of finding neighbors for each point individually, you predefine spherical regions in your data space
- Data points within these spheres vote to determine the class of a new point.

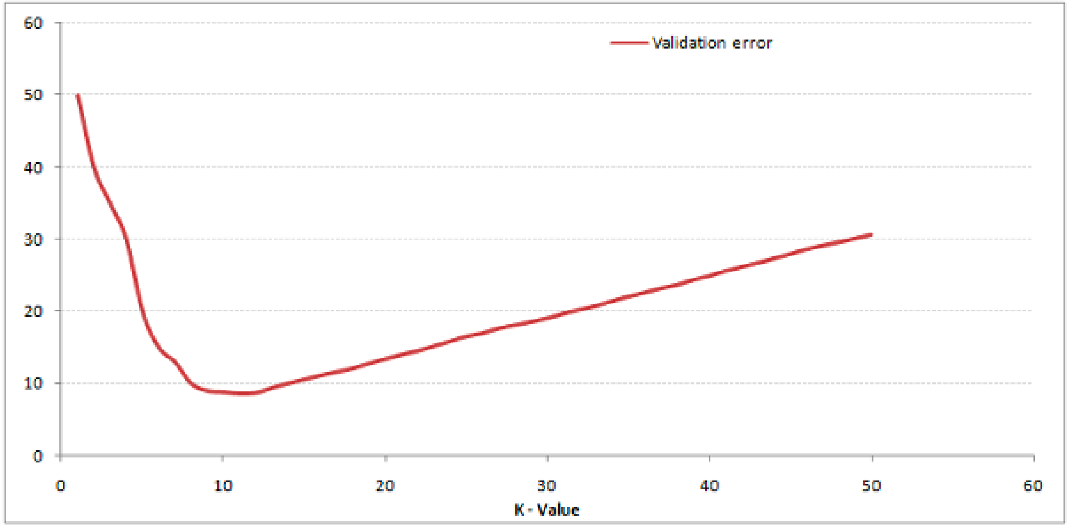


This saves you the trouble of searching all the points.

How to Choose K in KNN

Try multiple Ks and graph the error rate, pick the lowest!

- This approach is known as the **"elbow method"** and can be an effective way to identify an appropriate K value for your specific problem.



Decision Boundary of a Classifier

Is the line that separates positive and negative regions in the feature space

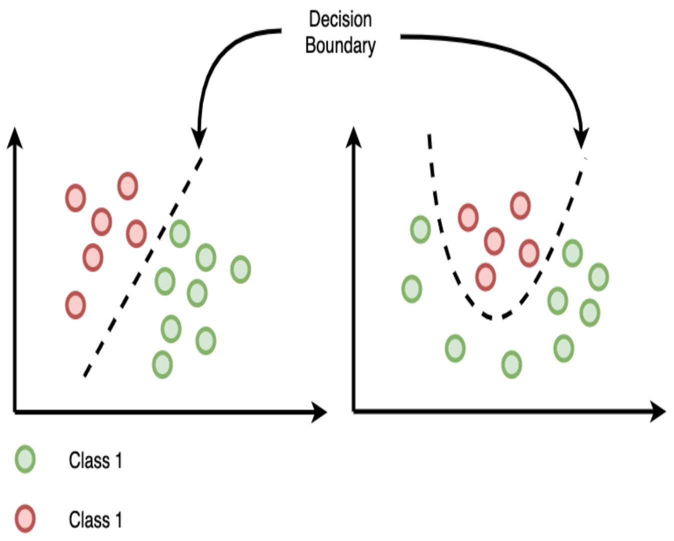
Purpose: It determines how new data points are classified based on their features.

Types:

- **Linear Boundary:** Straight line that separates classes in a linearly separable problem.
- **Non-linear Boundary:** Complex shapes that separate classes in non-linearly separable problems.

Why is it useful?

- it helps us visualize how examples will be classified for the entire feature space
- it helps us visualize the complexity of the learned model



Boundary Conditions

What if K=1?

Boundary Conditions

What if K=1?

- Our KNN model considers only the nearest neighbor when making predictions.
  - **Decision Boundary:** The decision boundary can be irregular and jagged because the model is sensitive to individual data points.
- Our model will basically be almost memorizing things--it will just say whatever is closest. Our model will be able to lack the ability to **generalize.**
  - **doesn't consider any broader patterns or relationships in the data.**

What if K=N?

Boundary Conditions

What if K=N?

- Our KNN model considers all data points as neighbors, effectively taking into account the entire dataset.
    - It treats all data points as relevant neighbors, regardless of their actual similarity to the query point.
    - It ignores the local structure of the data.
- Soon, more and more irrelevant points will contribute, and we will just predict the mode of the data for any new data point.

Pros and Cons of K-Nearest Neighbor

Pros and Cons of K-Nearest Neighbor

Pros

- Extremely accurate--used in large scale production in industry
- Easy to implement: Given the algorithm’s simplicity and accuracy, it is one of the first classifiers that a new data scientist will learn.
- - Few hyperparameters: KNN only requires a k value and a distance metric, which is low when compared to other machine learning algorithms.

Pros and Cons of K-Nearest Neighbor

KNN is a lazy algorithm: doesn't have a training step because it stores the entire dataset in memory and uses it directly to make predictions.

To make a prediction, KNN looks at the stored data and finds the nearest neighbors to the query point to make a decision. There's no separate training phase where the model learns from the data; it's a "lazy learner" that uses the data on the spot to make predictions.

Cons

- Requires more memory and storage compared to other classifiers.
  - Needs to store the entire training set, which can be very large (on the order of petabytes or exabytes).
- K must be tuned correctly: The choice of K impacts the performance of the KNN model and needs to be selected carefully
- Searching for the nearest neighbors is very time consuming and computationally expensive for larger dataset. (This can be helped by the spherical approach).

Summary: Vocabulary

- This is a **supervised learning** model, which means the training data **has labels**.
  - For this example "Whether or not they paid back their loans."
- This example is called **classification** where we are deciding between categories
  - We're sorting things into categories. In our case, we're deciding if someone belongs to one of two groups (Pay back or not).
- We are doing **binary classification** where there are only two categories.
- KNN is an **instance based model** meaning it requires all the training data be stored to work.
  - This type of model makes decisions based on similar examples from the training data.

Vocabulary

- A **hyperparameter** is an argument to the model that determines how the model behaves
  - For example, K
  - It's like a setting for the model.
  - It decides how the model behaves. For KNN, it tells us how many similar examples to consider when making a decision.

Models

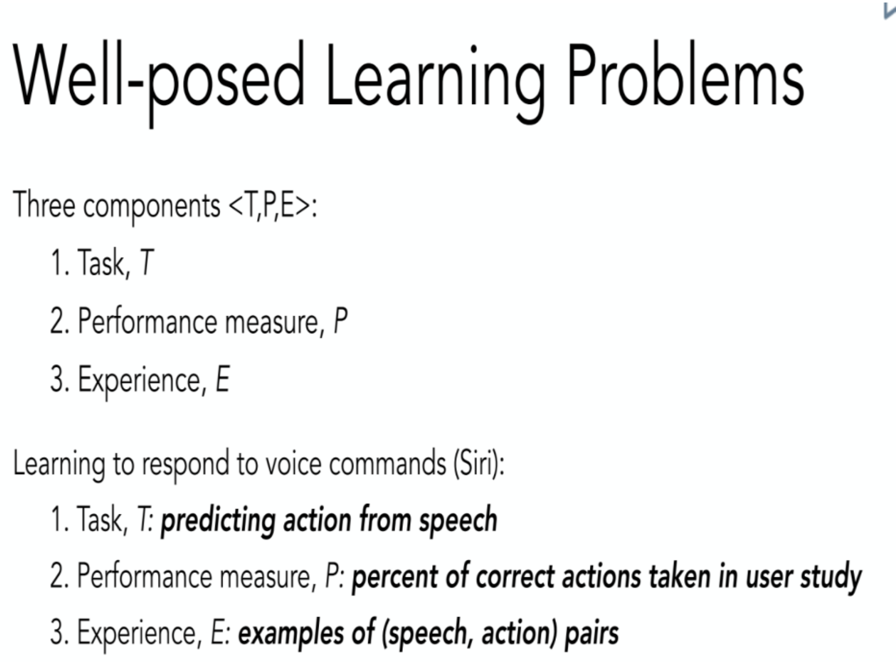
We will be learning about the following models:

- Decision trees (in detail)
- Random forests (in detail)
- Neural Networks (in detail)
- Naive Bayes
- Support Vector Machines

Additional Slides

What is Machine Learning (ML)

“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, **if its performance at tasks in T, as measured by P, improves with experience E.**



Well-posed Learning Problems

Three components <T,P,E>:

1. Task, *T*
2. Performance measure, *P*
3. Experience, *E*

Learning to respond to voice commands (Siri):

1. Task, *T*: ***predicting action from speech***
2. Performance measure, *P*: ***percent of correct actions taken in user study***
3. Experience, *E*: ***examples of (speech, action) pairs***