



# Nearest Neighbors



SAILORS

July 12, 2016



# Today's Session

- › Today we will be learning about **Nearest Neighbors**, an important AI algorithm
- › You will be coding in Python
- › You will find out how movie recommenders work, and making your own!

# Activity

On a notecard, write down a few important facts about yourself, e.g.

- › Age
- › Height
- › Hobbies
- › Family
- › Hometown
- › Eye colour
- › Personality
- › Favorite food

# Activity

Imagine you are a friend-matching service.

Work in your group to match together compatible people.

# Discussion

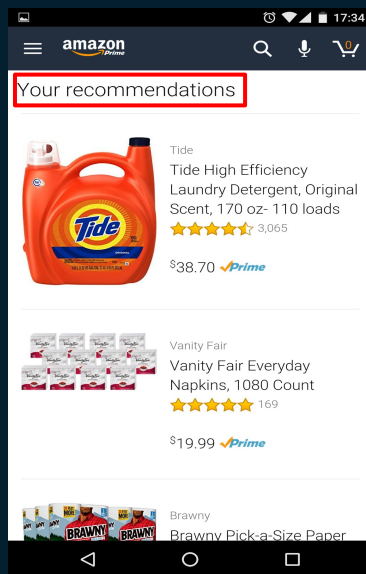
Which features were useful?

Which were not useful?

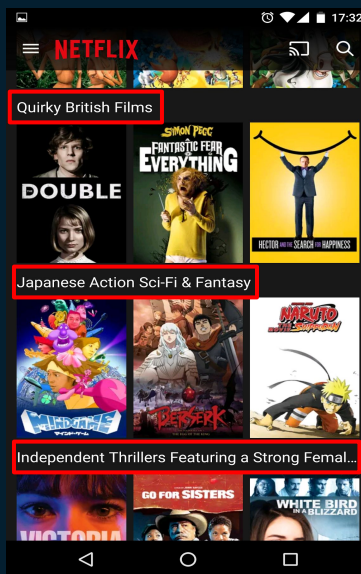
What if we were matching people for organ donation instead?

# Recommender Systems

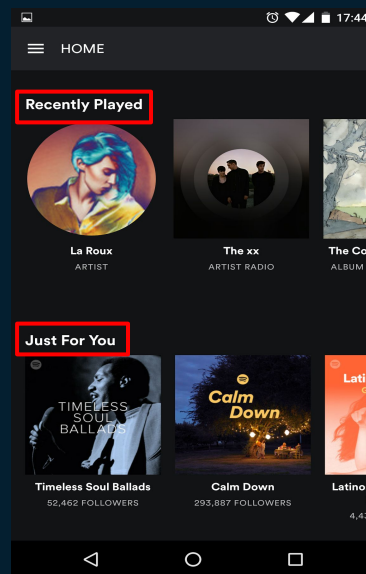
## Shopping



## Movies



## Music



# Recommender Systems

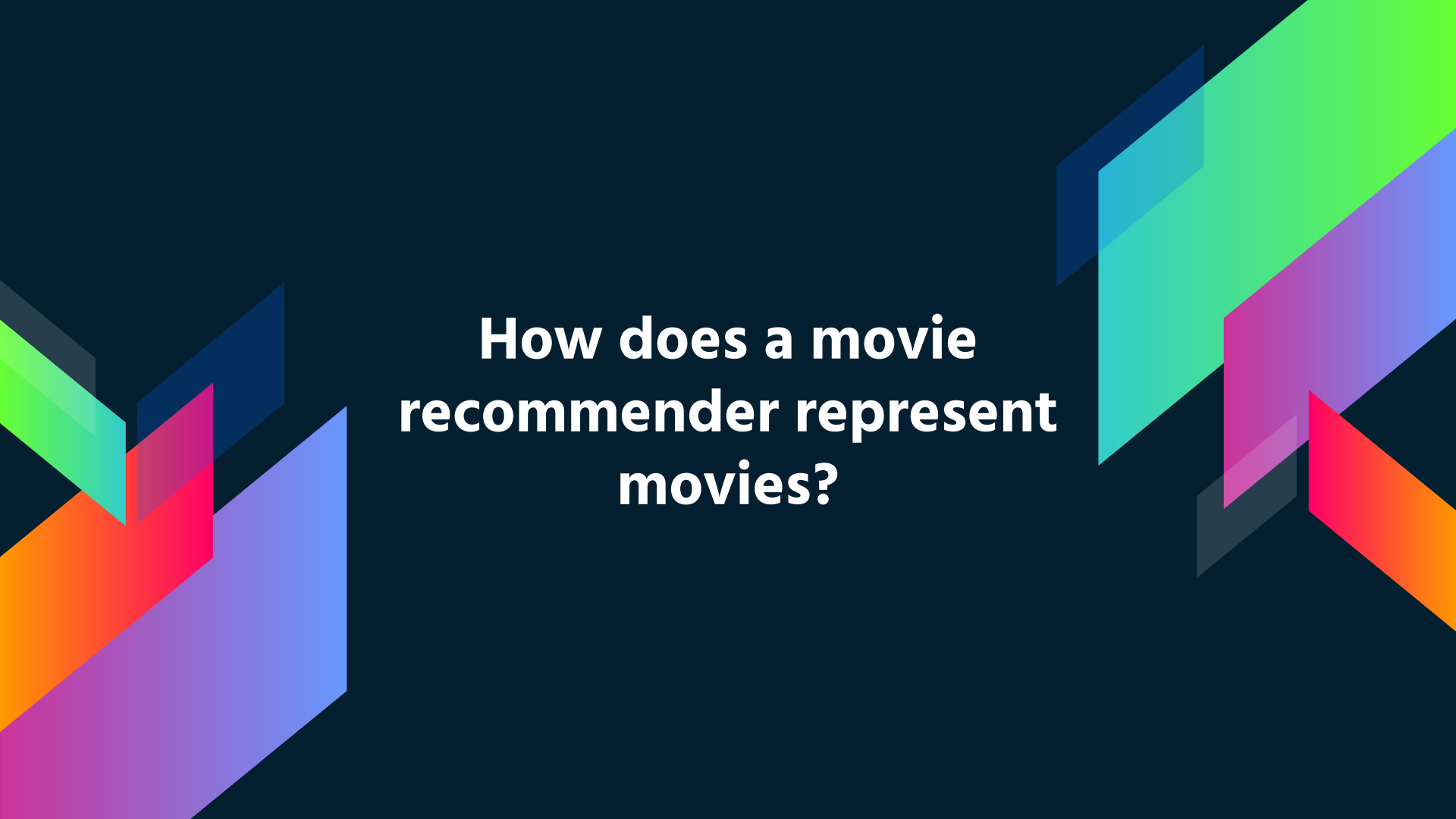
They're everywhere

- › Any kind of shopping
- › News articles
- › Social networks
- › Advertising
- › Dating services

# Recommender Systems

- › They're lucrative
  - › In 2009 Netflix offered \$1 million to the best recommendation system
  - › Google and Facebook make most of their money from advertising
- › They can be huge
  - › Amazon sells over 480 million products

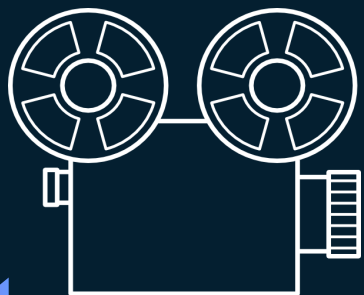


The background is a dark navy blue. It features several overlapping, semi-transparent geometric shapes in various colors: bright green, cyan, light blue, purple, pink, red, orange, and yellow. These shapes are arranged in a way that creates a sense of depth and movement, with some shapes appearing to be layered on top of others. The overall aesthetic is modern and tech-oriented.

**How does a movie  
recommender represent  
movies?**



Age  
Hobbies  
Hometown  
Family  
Height



Genre  
User rating  
Cast  
Runtime  
Director

# Representing Movies

- › Netflix represents each movie by its features.
- › For example: *year of release* and *runtime*

*Mean Girls* → (2004, 97)

*The Hunger Games* → (2012, 142)

# Measuring Similarity

*Mean Girls* → (2004, 97)

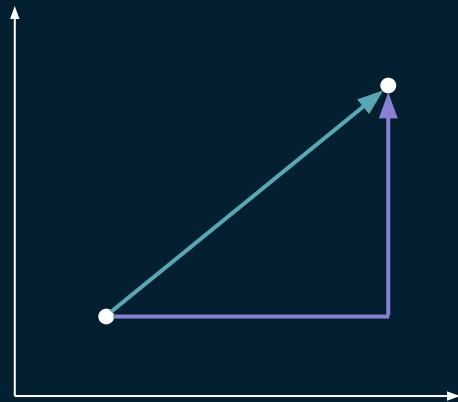
*The Hunger Games* → (2012, 142)

Q: How does a computer measure similarity between two movies?

A: Use the features as *co-ordinates*

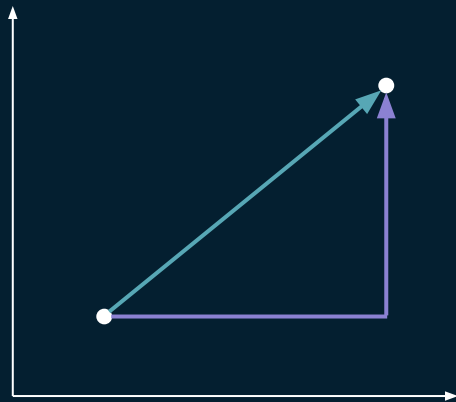
# Two types of distance

- › Euclidean
  - › "As the bird flies"
- › Manhattan
  - › "Horizontal then vertical"



# How to calculate?

- › Euclidean
  - › Use Pythagoras
- › Manhattan
  - › Add up the differences



# Nearest Neighbor Algorithm

- › A way to recommend a movie similar to your favorite
- › Measure the distance from your favorite movie to all other movies
- › Recommend the *closest one* (the *nearest neighbor*)



# Coding Activity

- › You'll be building a movie recommendation engine!
- › Given a movie that you liked, can the system recommend others to watch?

# Coding Activity

- › 162 movies with 12 features each
- › You choose which features to use
- › Discover which features are useful for recommendations, and which are not
- › Programming language: **Python**

# 1. Get the code

[github.com/abisee/movie-recommender](https://github.com/abisee/movie-recommender)

## 2. Exercise

- › The function **eucl\_dist** calculates the Euclidean distance between two multi-dimensional points.
- › Your job: write the function **manh\_distance** to calculate the Manhattan distance.
  - › Hint: copy **eucl\_dist** then modify.
  - › Hint: **abs(x,y)** gives the absolute difference  $|x-y|$

## 2. Solution

```
def manh_dist(vec1, vec2):  
    """Returns the Manhattan distance between vec1 and vec2.  
    Inputs:  
        vec1, vec2: Python lists of floats (numbers)  
    Returns:  
        a float (number)"""  
    dist = 0  
    num_dims = len(vec1)  
    for i in range(num_dims):  
        dist += abs(vec1[i] - vec2[i])  
    return dist
```

← Don't square this

← Don't take square root

### 3. Start recommending!

- › Currently we're just using *runtime* and *year* as features (marked by 1s).
- › Run the cell to see recommendations for *Harry Potter and the Goblet of Fire*.
- › Top recommendation: *Downfall*
- › Change the 1s and 0s to select a better set of features!



# Challenges

## Old Movies

Try "Casablanca". Do you get other old movies?

## Genres

Try: superhero films, space films, young adult films... do you get other similar?

## Use Manhattan distance

Try changing `eucl_dist` to `manh_dist` in the cell. Does it help?

## Foreign movies

Try "Spirited Away" (Japanese) or "Pan's Labyrinth" (Spanish). Can you get other Japanese/Spanish movies?

## Feature weights

Instead of 1 or 0 (on or off), you can give a feature *any weight*, like 0.5 (half as important), or 10 (ten times as important). This gives you more control over your recommendations.

## 4. Discussion

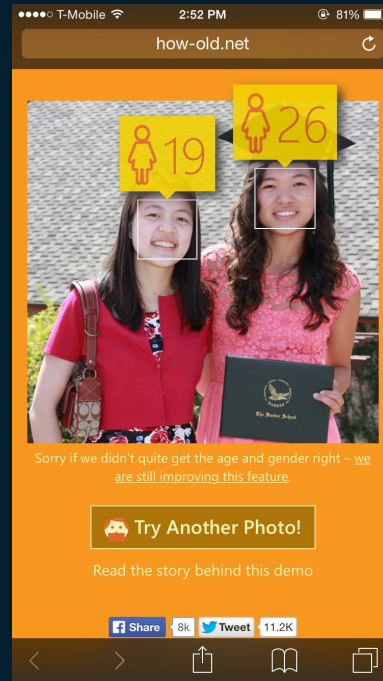
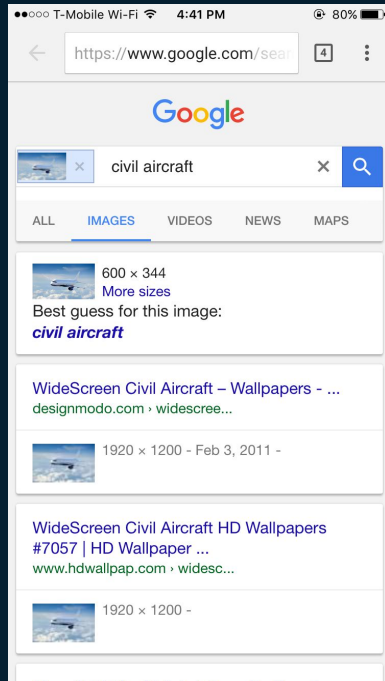
- › Which features were useful?
- › Which features were not useful?
- › How could this system be better?
- › Other observations?



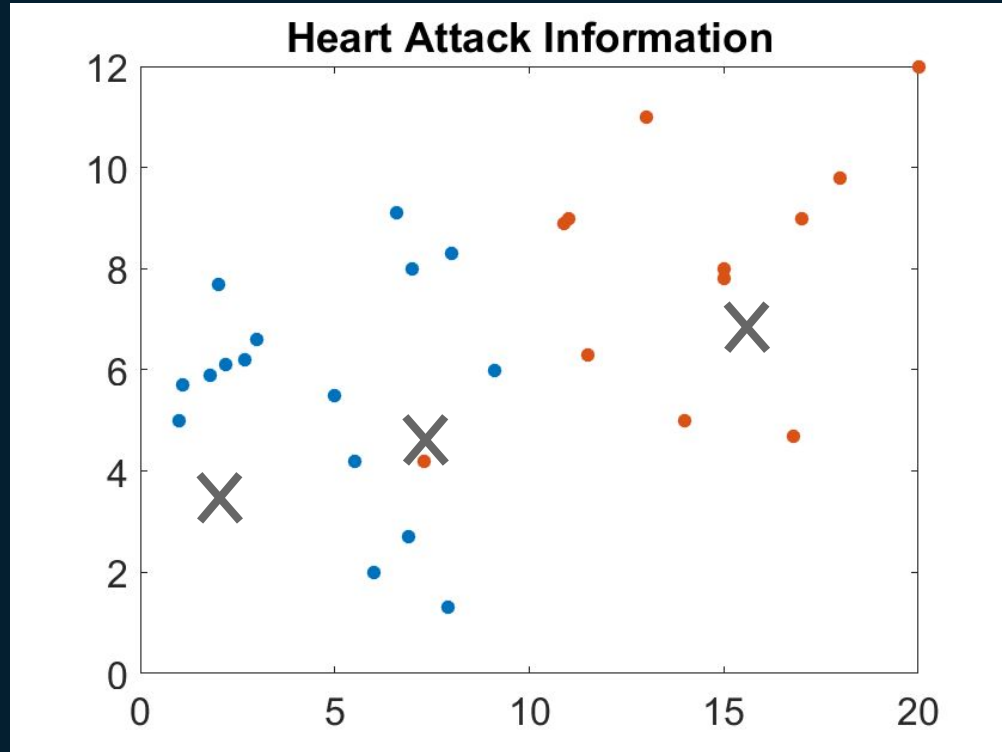
# Classification Problems

- › These are AI problems where we want to classify something into one of several classes. Examples:
  - › Given a patient's medical record, classify into "high risk" or "low risk" for a disease.
  - › Given a photo of an object, identify the object (e.g. Facebook face recognition).

# Classification Systems



# Example: Risk of Heart Attack



# k-Nearest Neighbors Algorithm

- › An extension of the nearest neighbors algorithm that can be used for classification problems (e.g. images).
- › Measure the distance from your image to all known images in your dataset.
- › Use plurality vote (with the  $k$  closest images) to classify your image.

# k-Nearest Neighbors Example

Want to Classify:



Plane: 2

Boat: 1

Car: 1

Helicopter: 1

The image features a dark navy blue background. In the top-left and bottom-left corners, there are overlapping geometric shapes in shades of light blue, purple, pink, and orange. Similarly, in the top-right and bottom-right corners, there are overlapping geometric shapes in shades of light blue, purple, pink, and orange. The central text "Questions?" is white and bold.

**Questions?**