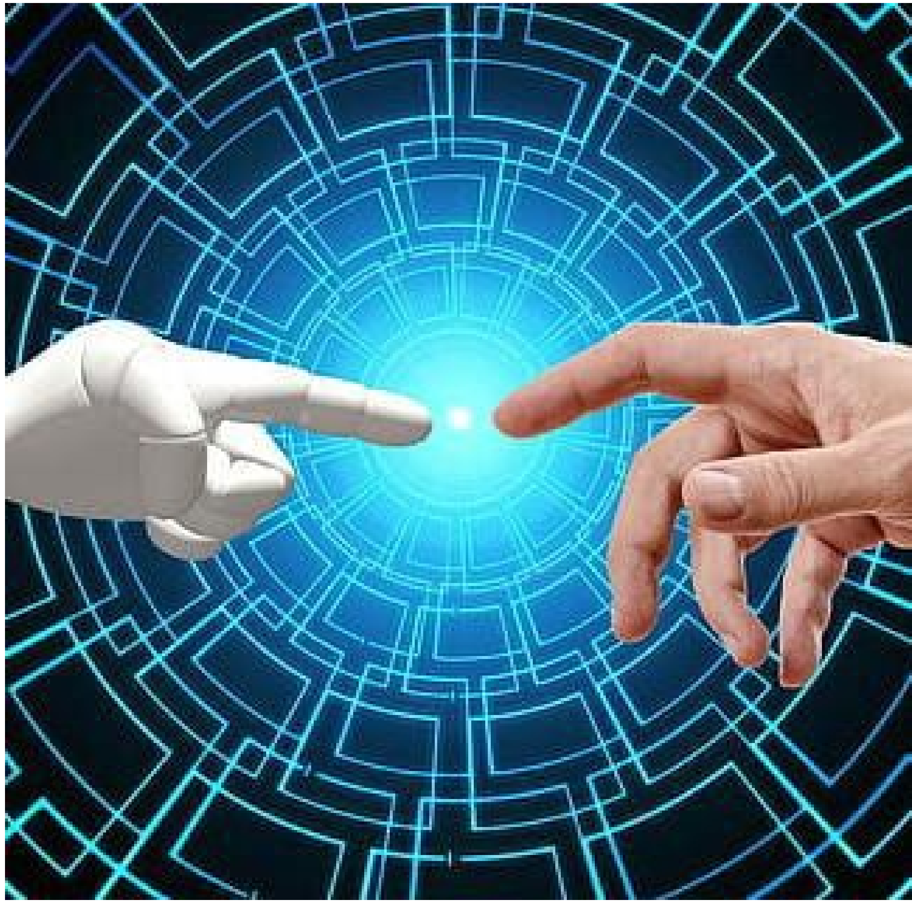
# Artificial Intelligence (AI) for Engineering

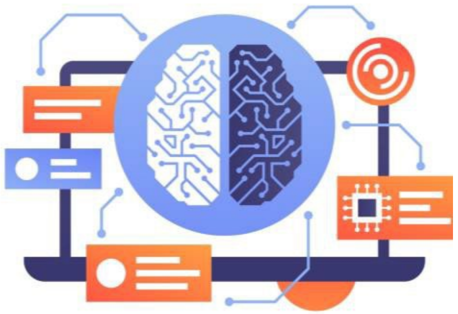
## COS40007

Dr. Afzal Azeem Chowdhary

Lecturer, SoCET, Swinburne University of Technology

Seminar 9: 29th April 2025

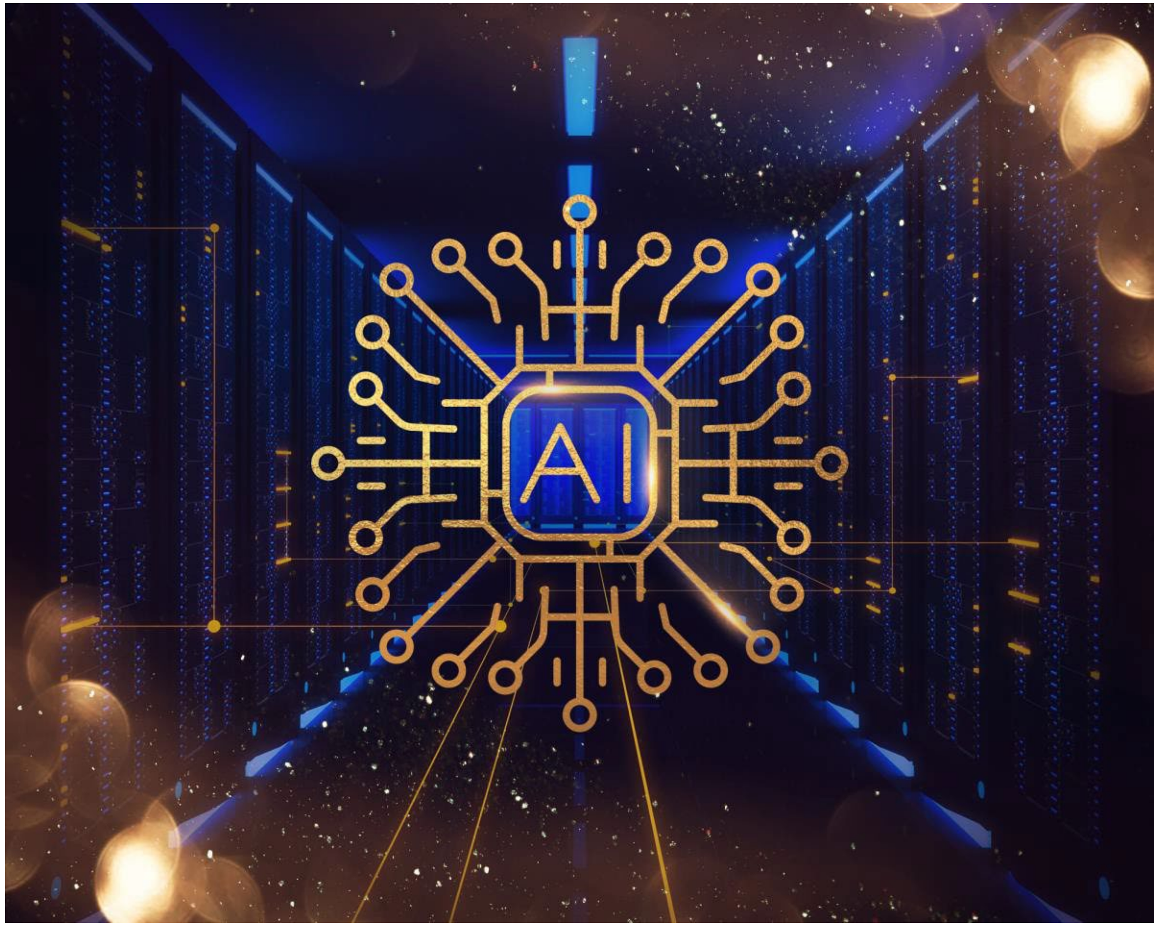




SWINBURNE UNIVERSITY OF TECHNOLOGY

# Overview

- Basics of Recommendation Engines
- Types of Recommender
- Collaborative Filtering
- *K*-Nearest Neighbour
- Content-based Filtering
- Predictive Analytics as Recommender
- Prescriptive Analytics as Recommender



# At the end of this you should be able to

- Understand about the recommendation engine
- Understand about the data for the recommendation engine
- Understand different recommendation system
- Understand applications of recommendation system

# Recommender Systems

# Recommender/Recommendation Engine

- A recommendation engine, a recommender, is an AI system that suggests items to a user. Recommendation systems rely on data-driven analytics and ML algorithms to find patterns in user behaviour data and recommend relevant items based on those patterns.
- Recommendation engines help users discover content, products or services they might not have found on their own
- The suggestions created by recommendation systems also play a vital role in personalising user experiences.
- **Examples:** video to watch, a similar song to listen to, relevant search results or a product that complements a particular order.

# Recommendation Engine examples

- **Facebook**–"People You May Know"
- **Netflix**–"Other Movies You May Enjoy"
- **LinkedIn**–"Jobs You May Be Interested In"
- **Amazon**–"Customer who bought this item also bought …"
- **YouTube**–"Recommended Videos"
- **Google**–"Search results adjusted"
- **Pinterest**–"Recommended Images"

# How Recommendation Engine works

**Collect Data:** User activities, products, demographics, psychographics (interests or lifestyle)

**Data Storage:**

- A data warehouse can aggregate data from different sources to support data analysis and machine learning.
- A data lakehouse combines the best aspects of data warehouses and data lakes into a single data management solution

**Analysis:** ML algorithms trained on large datasets detect patterns, identify correlations and weigh the strength of those patterns and correlations.

**Filtering:**

- Showing the most relevant items
- Apply specific mathematical rules and formulas to the data depending on the type of recommendation engine used.

**Refining:** Regularly assess the outputs of a recommendation system and further optimise the model

SWINBURNE UNIVERSITY OF TECHNOLOGY

# Types of Recommendation Engines

## *Collaborative Filtering*

- Filter suggestions are based on a particular user's likeness to others.
- Assume that users with comparable preferences will likely be interested in the same items
- **Example:** Amazon product recommendations, Spotify recommendations
- **Limitation:** cold start problem, which happens when the system has limited historical data to draw from, especially for new users.

## *Content-based Filtering*

- Filters recommendations based on an item's features.
- Assume that if a user likes a particular item, they will also like another similar item
- **Example:** Recommending a movie with similar genres, actors, or directors to movies a user has previously watched and enjoyed

## *Hybrid Recommendation System*

- Merges collaborative filtering and content-based filtering.
- A hybrid approach can greatly enhance a recommendation engine's performance but requires advanced architectures and intensive computational power.
- **Example:** Netflix

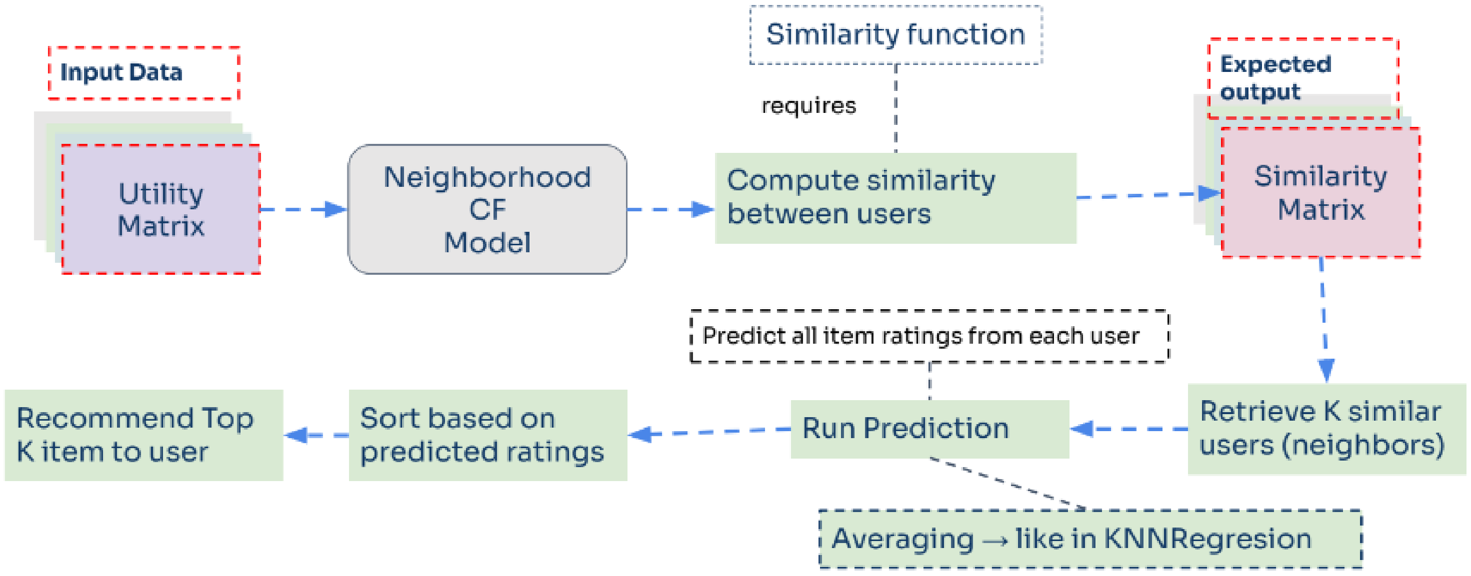
SWINBURNE UNIVERSITY OF TECHNOLOGY

# Collaborative Filtering

*Memory-based*

Represent users and items as a matrix. They are an extension of the *k*-nearest neighbours (kNN) algorithm because they aim to find their "nearest neighbours," which can be similar users or items. There are two types:

- ✓ User-based filtering computes similarities between a particular user and all other users in the matrix
- ✓ Item-based filtering computes item similarity through user behaviour (how users interact with items, not item features). No **ML models** are used.



# Collaborative Filtering

*Model-based*

- Create a predictive machine learning model of the data.
- The user-item matrix serves as the training data set for the model, which then yields predictions for missing values, that is, items that a user has not yet found and will, therefore, be recommended.
- Use matrix factorisation
- A more advanced implementation of matrix factorisation harnesses deep learning neural networks. Other model-based systems employ machine learning algorithms such as Bayes classifiers, clustering and decision trees.

# Collaborative Filtering

**Item** - What the system recommends to the user (CD, news, books, movies...)

**User preferences** - ratings for products

**User actions** - user browsing history

The task of a CF algorithm is to find the item likeness of two forms :

1. Prediction – a numerical value expressing the predicted likelihood of an item for the active user
2. Recommendation – a list of *N* items that the active user will like the most

# *k*- Nearest Neighbour Algorithm

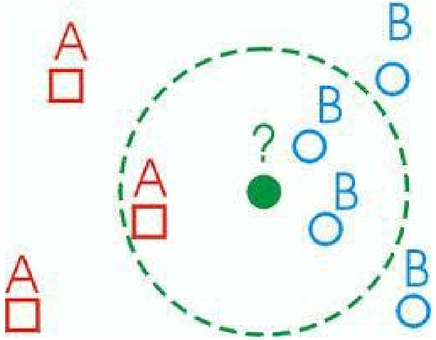
Tell me about your friends(*who your neighbours are),* and *I will tell you who you are*.

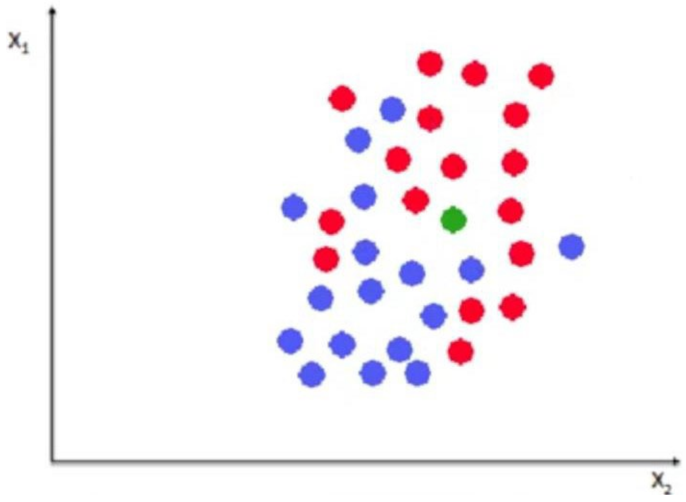
- A distance measure is needed to determine the "closeness" of instances
- Classify an instance by finding its nearest neighbours and picking the most popular class among the neighbours

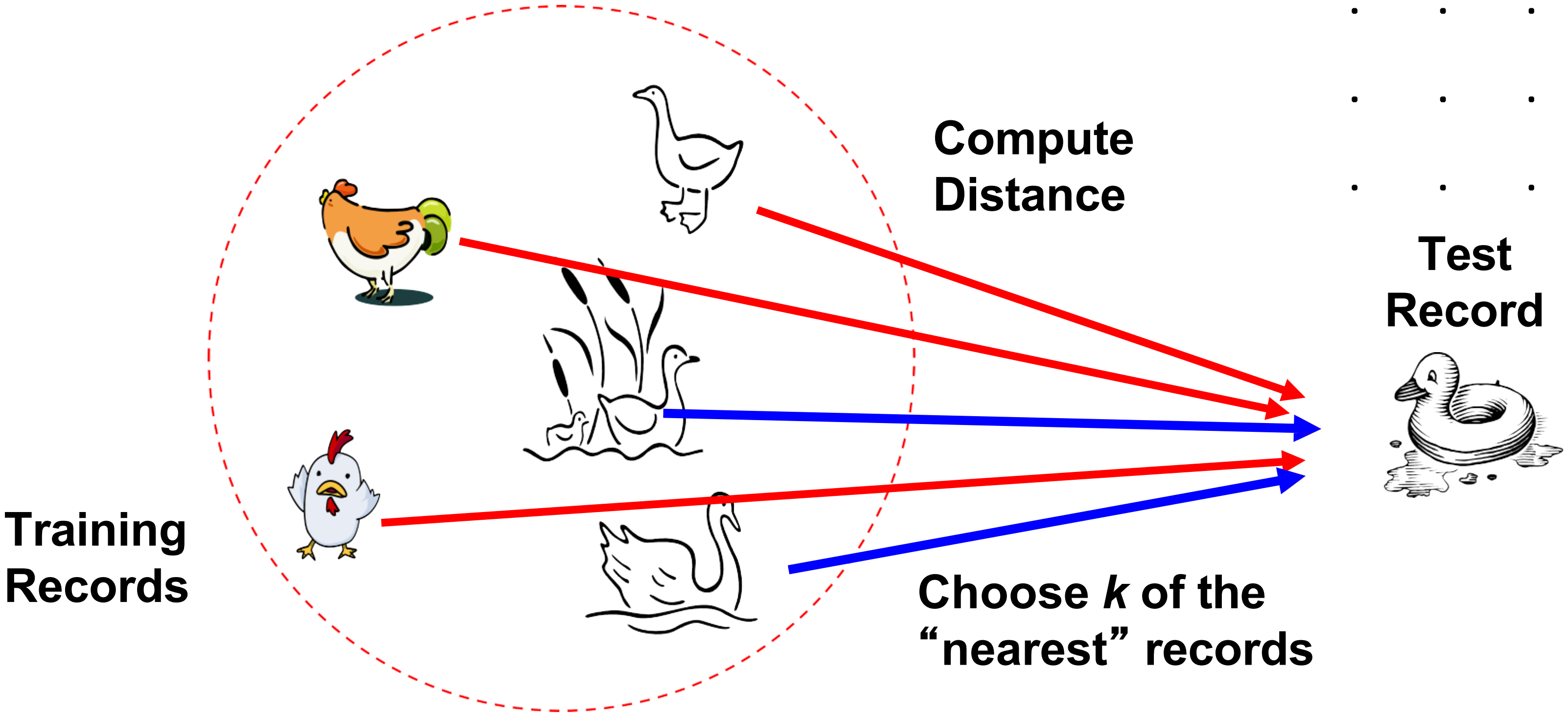
- ➢ Simple to implement and use
- ➢ Comprehensible – easy to explain the prediction
- ➢ Robust to noisy data by averaging k-nearest neighbours

# KNN approach

- An object (a new instance) is classified by a majority vote for its neighbour classes.
- The object is assigned to the most common class amongst its *K* nearest neighbours (*measured by a distance function*)



# Distance between neighbours

- Calculate the distance between the new example ($E$) and all examples in the training set.
- *Euclidean* distance between two examples,
  - $X = [x_1, x_2, x_3, .., x_n]$
  - $Y = [y_1, y_2, y_3, \ldots, y_n]$
  - The Euclidean distance between $X$ and $Y$ is defined as:

$$D(X, Y) = \sqrt{\sum_{k=1}^{n} (x_i - y_i)^2}$$

SWINBURNE UNIVERSITY OF TECHNOLOGY

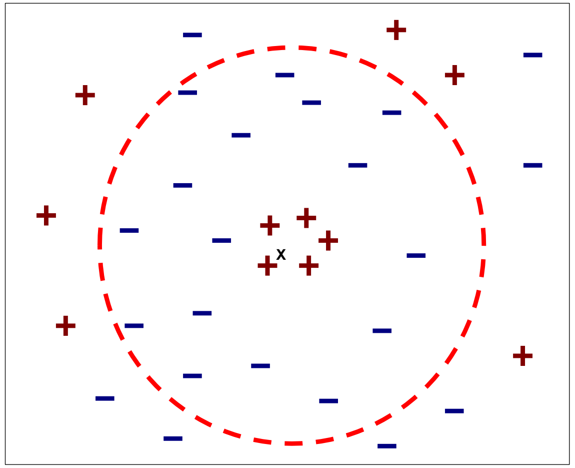
# *K*- Nearest Neighbour Algorithm

- All the instances correspond to points in an *n*-dimensional feature space.
- Each instance is represented with a set of numerical attributes.
- Each training data consists of vectors and a class label associated with each vector.
- Classification is done by comparing feature vectors of different *K* nearest points.
- Select the *K*-nearest examples to *E* in the training set.
- Assign E to the most common class among its *K*-nearest neighbours.

# How to choose *K*

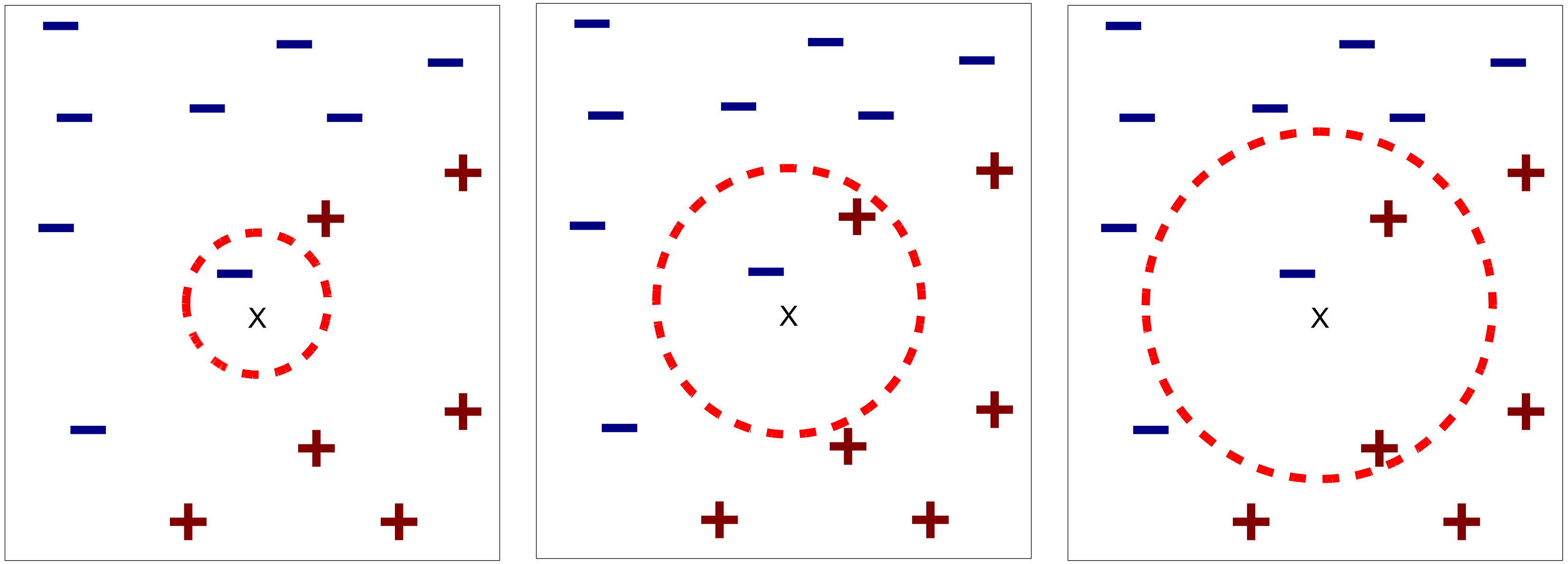
If *K* is too small, it is sensitive to noise points.

Larger *K* works well. But too large *K* may include majority points from other classes.




The rule of thumb is $K < \mathit{sqrt}(n)$, and $n$ is a number of examples.

# How to choose K




(a) 1-nearest neighbour (b) 2-nearest neighbour (c) 3-nearest neighbour

*K*-nearest neighbours of a record *x* are data points that have the *k* smallest distances to *x*

# KNN Feature Weighting

- Scale each feature by its importance for classification,

$$D(a,b) = \sqrt{\sum_k w_k (a_k - b_k)^2}$$

- Can use our prior knowledge about which features are more important
- Can learn the weights $\mathbf{w_k}$ using **cross-validation** (to be covered later)

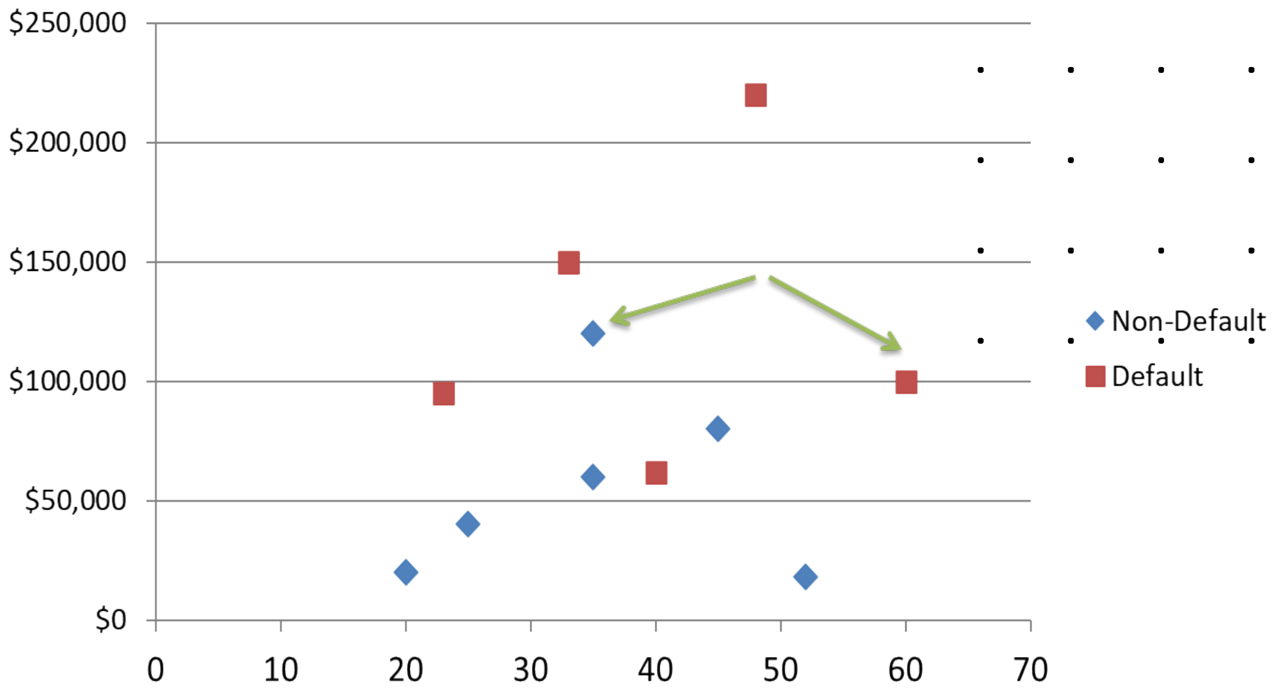
# KNN Feature Normalisation

- The distance between neighbours could be dominated by some attributes with relatively large numbers, e.g., the income of customers in our previous example.
- Arises when two features are on different scales.

- It is important to normalise those features.
  – Mapping values to numbers between 0 and 1.

# KNN Classification

| Age | Loan | Default | Distance |
|---|---|---|---|
| 0.125 | 0.11 | N | 0.7652 |
| 0.375 | 0.21 | N | 0.5200 |
| 0.625 | 0.31 | N | 0.3160 |
| 0 | 0.01 | N | 0.9245 |
| 0.375 | 0.50 | N | 0.3428 |
| 0.8 | 0.00 | N | 0.6220 |
| 0.075 | 0.38 | Y | 0.6669 |
| 0.5 | 0.22 | Y | 0.4437 |
| 1 | 0.41 | Y | 0.3650 |
| 0.7 | 1.00 | Y | 0.3861 |
| 0.325 | 0.65 | Y | 0.3771 |
| | | | |
| **0.7** | **0.61** | **?** | |




$$X_s = \frac{X - Min}{Max - Min}$$

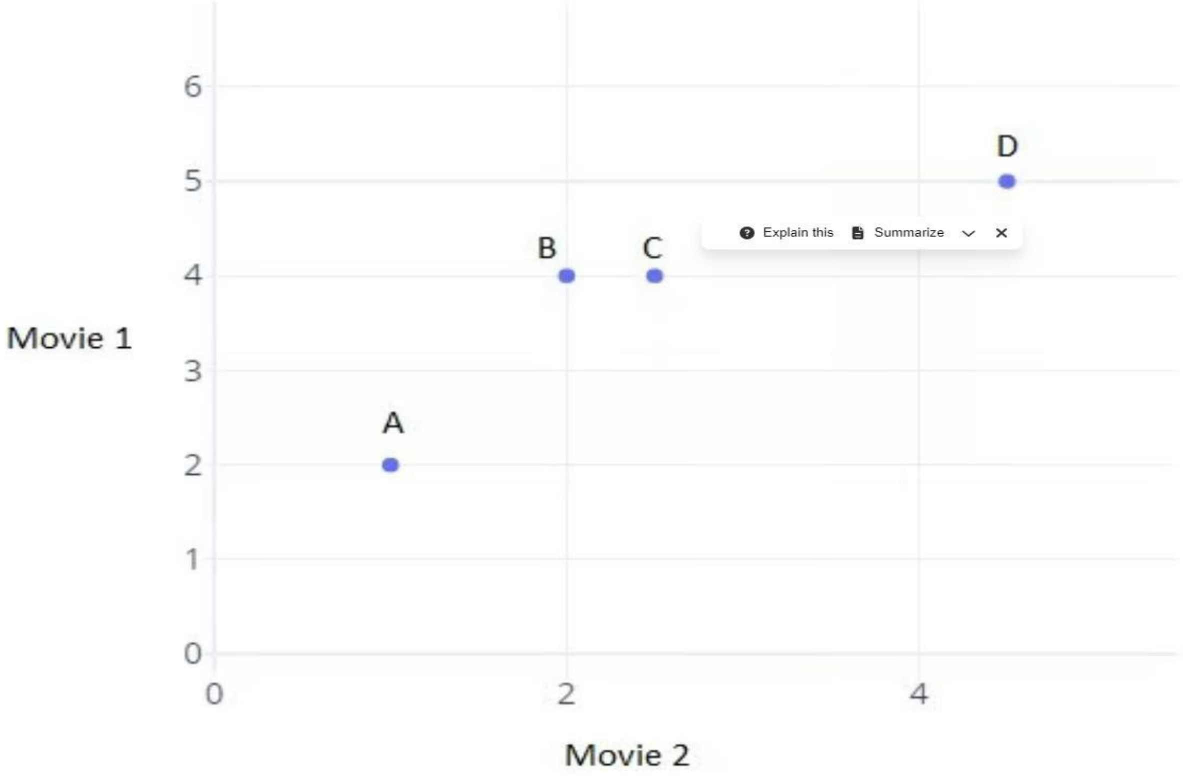
# Collaborative Filtering Steps

- How do you determine which users or items are similar to one another?
- Given that you know which users are similar, how do you determine the rating that a user would give to an item based on the ratings of similar users?
- How do you measure the accuracy of the ratings you calculate?

# Find Similar Users on the Basis of Ratings

- Users A, B, C, and D, who have rated two movies

Ratings by A are [1.0, 2.0].
Ratings by B are [2.0, 4.0].
Ratings by C are [2.5, 4.0].
Ratings by D are [4.5, 5.0].



# Compute Similarity

```
from scipy import spatial

a = [1, 2]
b = [2, 4]
c = [2.5, 4]
d = [4.5, 5]

spatial.distance.euclidean(c, a)
2.5

spatial.distance.euclidean(c, b)
0.5

spatial.distance.euclidean(c, d)
2.23606797749979
```

# Compute Rating

After determining a list of users similar to user $U$, you need to calculate the rating $R$ that $U$ would give to a certain item $I$,

$$R_U = \frac{\sum_{u=1}^{n} R_u}{n}$$

SWINBURNE UNIVERSITY OF TECHNOLOGY

# Collaborative Filtering

- **User-based:** For a user *U*, with a set of similar users determined based on rating vectors consisting of given item ratings, the rating for an item *I*, which hasn't been rated, is found by picking out *N* users from the similarity list who have rated the item *I* and calculating the rating based on these *N* ratings.
- **Item-based:** For an item *I*, with a set of similar items determined based on rating vectors consisting of received user ratings, the rating by a user *U*, who hasn't rated it, is found by picking out *N* items from the similarity list that *U* has rated and calculating the rating based on these *N* ratings.

# Python Program for Collaborative Filtering

```
# load_data.py
import pandas as pd
from surprise import Dataset
from surprise import Reader

# This is the same data that was plotted for similarity earlier
# with one new user "E" who has rated only movie 1
ratings_dict = {
"item": [1, 2, 1, 2, 1, 2, 1, 2, 1],
"user": ['A', 'A', 'B', 'B', 'C', 'C', 'D', 'D', 'E'],
"rating": [1, 2, 2, 4, 2.5, 4, 4.5, 5, 3],
}

df = pd.DataFrame(ratings_dict)
reader = Reader(rating_scale=(1, 5))

# Loads Pandas dataframe
data = Dataset.load_from_df(df[["user", "item", "rating"]], reader)

#Loads the built-in Movielens-100k data
movielens = Dataset.load_builtin('ml-100k')
```

# Using KNN

```
# recommender.py
from surprise import KNNWithMeans

# To use item-based cosine similarity
sim_options = {
"name": "cosine",
"user_based": False, # Compute similarities between items
}

algo = KNNWithMeans(sim_options=sim_options)

from load_data import data
from recommender import algo
trainingSet = data.build_full_trainset()
algo.fit(trainingSet)

# Computing the cosine similarity matrix...
# Done computing similarity matrix.
# <surprise.prediction_algorithms.knns.KNNWithMeans object at 0x7f04fec56898>
prediction = algo.predict('E', 2)
>>> prediction.est 4.15
```

# Content-Based Filtering

**Benefits:**

- Independent of other user data
- Tailored to user preferences - aligning recommendations with the user's interests and preferences.
- Transparency in recommendations is directly tied to the user's actions.
- Simplicity in creation and data science- focus primarily on classifying items based on attributes, leveraging techniques such as vector space models and term frequency analysis.
- Overcoming the "cold start" problem -only initial user inputs deliver quality recommendations.

**Challenges:**

- Limited novelty and diversity may suggest overly familiar options and limit users' diversity of possibilities.
- Scalability and attributes—Adding a new item, product, service, or content piece necessitates defining and tagging its attributes.
- Accuracy and attribute assignment - the precision and uniformity in assigning attributes significantly affect its success.
- Over-reliance on content quality and availability- highly dependent on item metadata or descriptions' quality, accuracy, and availability.

# Difference Between Content-Based Filtering and Collaborative Filtering

| **Aspect** | **Content-Based Filtering** | **Collaborative Filtering** |
|---|---|---|
| *Focus* | Item attributes | User Behaviour |
| *Recommendation* | Items similar to what the user likes | Items liked by similar users to the user |
| *Data required* | Information about the item | User behavior data, such as ratings or purchases |
| *Advantage* | Doesn’t require user data | Can recommend niche or new items |
| *Disadvantage* | May miss out on new interests | Needs sufficient user data to be effective |

SWINBURNE UNIVERSITY OF TECHNOLOGY

# Prescriptive Analytics-based Recommendation

- **Descriptive analytics** - Answers what has already happened
- **Diagnostic analytics** - Answers why did this happen
- **Predictive analytics** - Answers what could/might happen
- **Prescriptive analytics** - Answers what should happen/what should we do next

Prescriptive analytics is analysing data to identify patterns, which can be used to make predictions and determine optimal courses of action.

- Its focus is predicting future outcomes and recommending actions or decisions to achieve desired outcomes or prevent undesirable ones.
- Prescriptive analytics adds a recommendation layer on top of predictive analytics.
- **Example:** An e-commerce platform can use prescriptive analytics to analyse customers' actions, such as browsing and buying habits, to give the most spot-on product recommendations.
- **Example:** Using prescriptive analytics to dive into transaction patterns and customer behaviour, they can spot fraud and leverage insights to implement more effective protection measures.

# How Prescriptive Analytics-based Recommendation Works

How it works:

**1. Data Collection and Analysis:**

Prescriptive analytics uses various data sources to analyse patterns, trends, and relationships.

**2. Model Building:**

Mathematical models and algorithms are used to simulate different scenarios and evaluate potential outcomes.

**3. Scenario Simulation:**

The models explore various possibilities and consider constraints and objectives to determine the most effective course of action.

**4. Recommendation Generation:**

Based on the analysis and simulation, prescriptive analytics provides specific recommendations for decision-makers.

# Learn, Practice and Enjoy the AI journey