The UK's European university

University of Kent

COMP8270 / PROGRAMMING FOR ARTIFICIAL INTELLIGENCE

Fernando Otero
febo@kent.ac.uk
cs.kent.ac.uk/people/staff/febo

overview:

1. Cluster analysis
2. Scikit-learn
   - Machine Learning in Python
   - K-Means
3. Pre-processing
   - Missing values
   - Categorical data

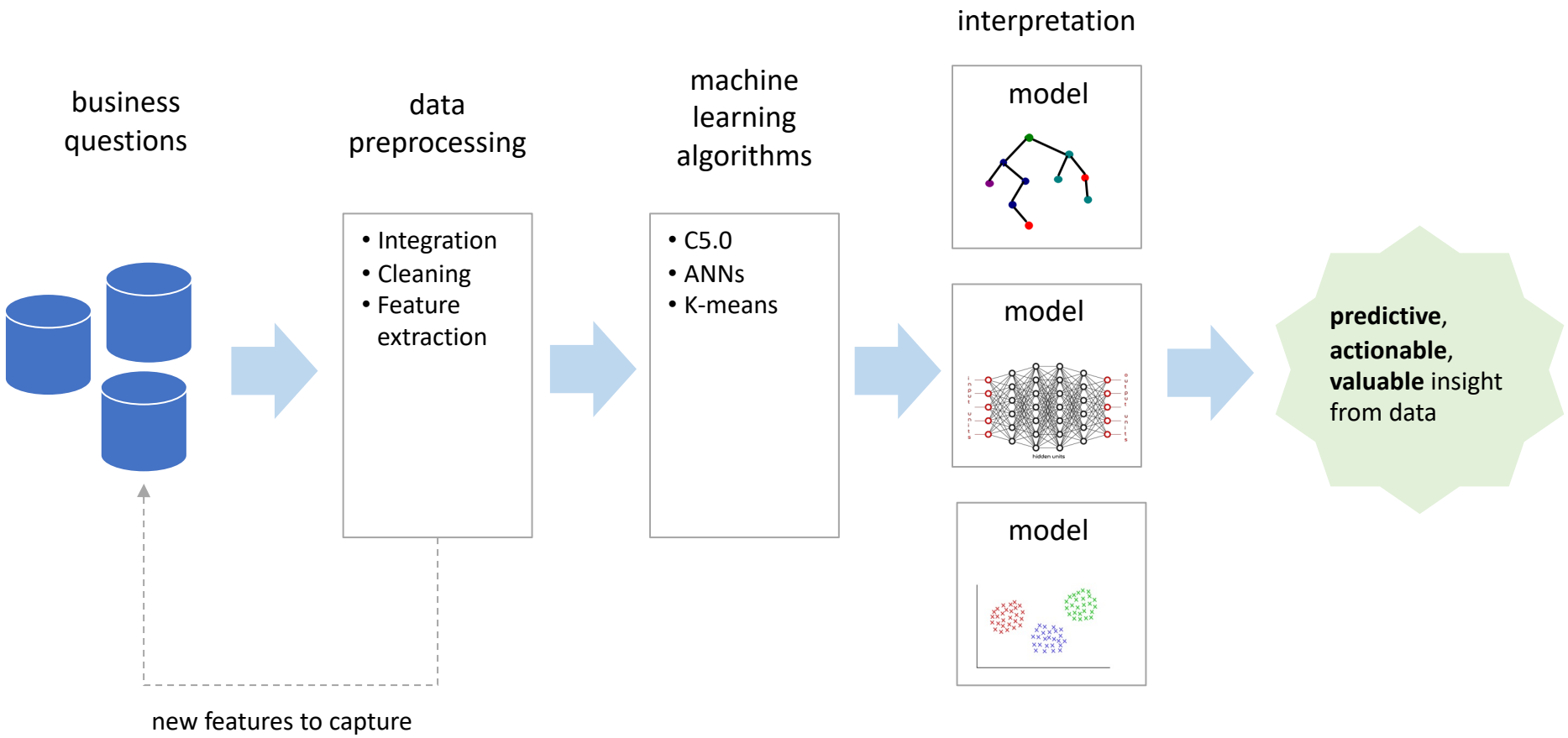
… but:

- Let's talk a bit a about artificial intelligence and machine learning:

> *How can one construct computer systems that automatically improve through experience?*

- **machine learning**:
  - adapt to new circumstances
  - detect and extrapolate patterns.

learning 'pipeline':



overview:

1. Cluster analysis
2. Scikit-learn
   - Machine Learning in Python
   - K-Means
3. Pre-processing
   - Missing values
   - Categorical data

supervised vs. unsupervised:

- Supervised learning
  - An attribute is specified as the target class before mining
  - e.g. classification

- Unsupervised learning
  - No predefined classes
  - e.g. **cluster analysis**

cluster analysis (1):

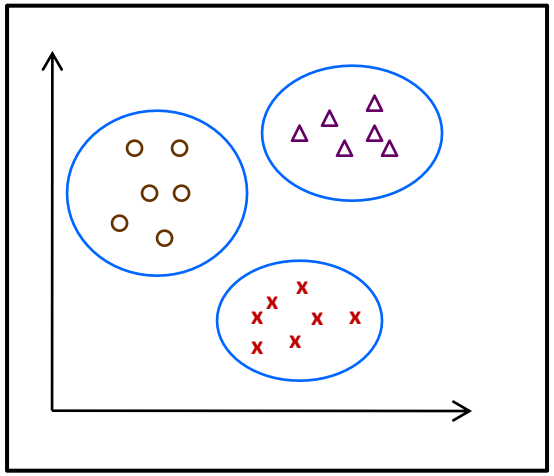
- A cluster is a collection of data objects:
  - Similar to one another within the same cluster (**intra-class**)
  - Dissimilar to the objects in other clusters (**inter-class**)

- Applications:
  - Identify similarities among data – e.g., customer segmentation
  - Group similar data objects into clusters – e.g., grouping experiment outcomes

cluster analysis (2):

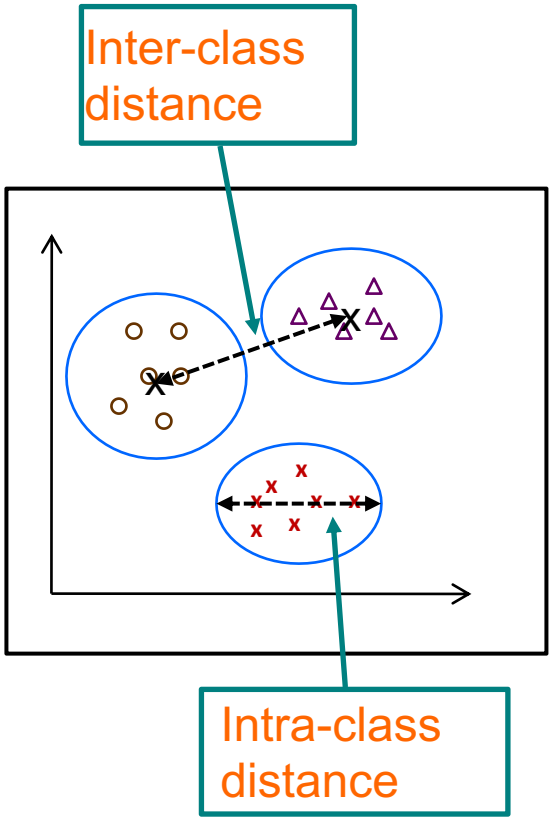
- Clustering is an **unsupervised** learning task

- The goal is to:
  - maximize **intra-**class similarity
  - minimize **inter-**class similarity



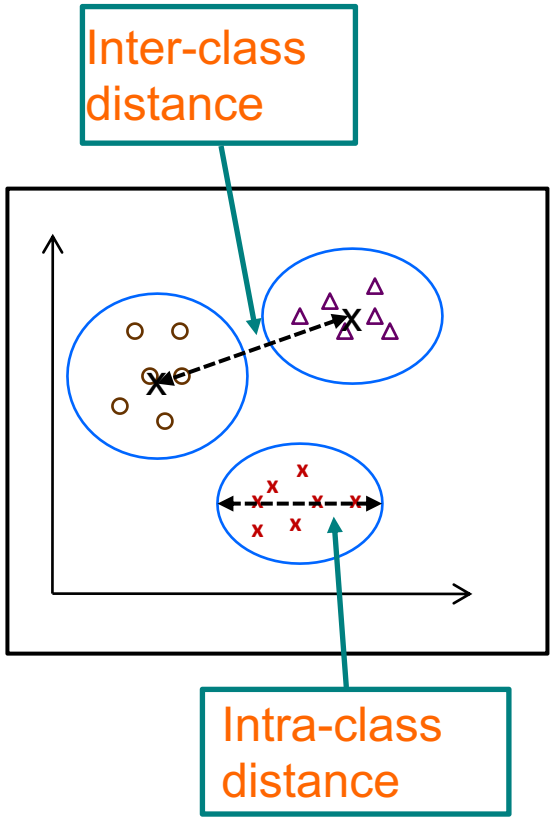
similarity measure (1):

- A distance function is used to measure similarity between two instances
  - intra-class distance
  - inter-class distance



similarity measure (2):

- A distance function is used to measure similarity between two instances
  - intra-class distance
  - inter-class distance
- Measure is dependent of the type of data
  - Euclidean distance for numeric data types



K-Means:

- Given a number ***K***, the K-Means algorithm consists of the following steps:

1. Randomly select K points as the initial centroids
2. Loop
    1. Assign (reassign) each point to the cluster with the nearest centroid
    2. Compute the centroid of each cluster
    3. Stop if there is no more new assignment, i.e. all centroids do not change any more

overview:

1. Cluster analysis
2. Scikit-learn
   - Machine Learning in Python
   - K-Means
3. Pre-processing
   - Missing values
   - Categorical data

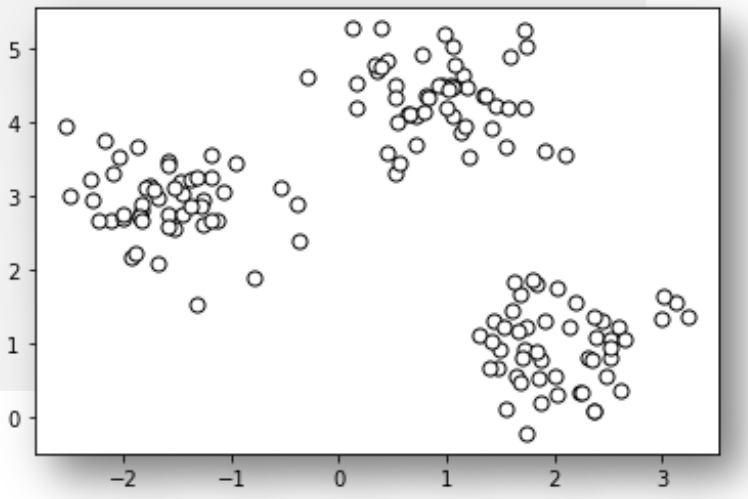
scikit-learn:

- Open source machine learning library that supports supervised and unsupervised learning
- Contain many algorithms implementations
  - "standard" library for machine learning
- Today we will focus on clustering, more specifically on **K-Means algorithm**

example (1):

- let's create some data:

```
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt

# create dataset
X, y = make_blobs(
    n_samples=150, n_features=2,
    centers=3, cluster_std=0.5,
    shuffle=True, random_state=0
)

# plot
plt.scatter(
    X[:, 0], X[:, 1],
    c='white', marker='o',
    edgecolor='black', s=50
)
plt.show()
```



example (2):

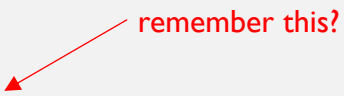
- K-Means clustering

```
from sklearn.cluster import KMeans  # required import

km = KMeans(
    n_clusters=3,    # number of clusters
    init='random',   # centroid initialisation
    n_init=10,       # number of executions
    max_iter=300,    # number of iterations
    random_state=0   # random seed
)
y_km = km.fit_predict(X)
```

example (3):

- Plotting the clusters:

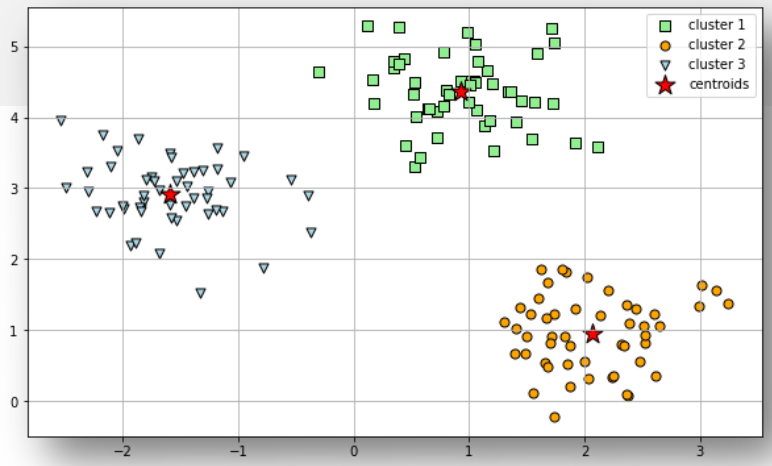
```
# cluster 1
plt.scatter(
    X[y_km == 0, 0], X[y_km == 0, 1],
    s=50, c='lightgreen',
    marker='s', edgecolor='black',
    label='cluster 1'
)

# cluster 2
plt.scatter(
    X[y_km == 1, 0], X[y_km == 1, 1],
    s=50, c='orange',
    marker='o', edgecolor='black',
    label='cluster 2'
)

# cluster 3
plt.scatter(
    X[y_km == 2, 0], X[y_km == 2, 1],
    s=50, c='lightblue',
    marker='v', edgecolor='black',
    label='cluster 3'
)
```



example (4):

- Plotting the centroids:

```
# plot the centroids
plt.scatter(
    km.cluster_centers_[:, 0], km.cluster_centers_[:, 1],
    s=250, marker='*',
    c='red', edgecolor='black',
    label='centroids'
)

plt.legend(scatterpoints=1)
plt.grid()
plt.show()
```



considerations:

- In the previous example, all values (dimensions) were present and had a numeric value
- How to deal with missing values?
- How to deal with categorical data?

overview:

1. Cluster analysis
2. Scikit-learn
   - Machine Learning in Python
   - K-Means
3. Pre-processing
   - Missing values
   - Categorical data

missing values (1):

- Checking if the data has missing values:

```
import pandas as pd

# titanic dataset
data_url = "http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv"
df = pd.read_csv(data_url)

# prints the number of NaN values per column
df.isna().sum()
```

- There are a couple of ways to handle missing values:
  - Impute missing values
  - Remove rows with missing values

missing values (2):

- Replacing missing values with a specific value:

```
import pandas as pd

# titanic dataset
data_url = "http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv"
df = pd.read_csv(data_url)

# prints the number of NaN values per column
df.isna().sum()
```

- There are a couple of ways to handle missing values:
  - Impute missing values
  - Remove rows with missing values

impute missing values:

```
# fill values with the mean
# mean() only works for numeric columns
df['Age'].fillna(df['Age'].mean(), inplace=True)
```

remove missing:

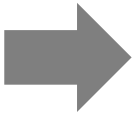
- Remove rows or columns with missing data:

```
# if there are too many missing values for a column
# consider dropping the entire column

df.drop(columns=['Cabin'])


# rows with missing values
df[df['Embarked'].isna()]

# remove rows with missing values
df.drop(df[df['Embarked'].isna()].index)
```

categorical data:

- Use **OneHotEncoder** to create vector representation in which all of the elements in a vector are 0, except for one, which has 1 as its value

Label	ID
Strawberry	1
Apple	2
Watermelon	3
Lemon	4
Peach	5
Orange	6



Strawberry	Apple	Watermelon	Lemon	Peach	Orange	ID
1	0	0	0	0	0	1
0	1	0	0	0	0	2
0	0	1	0	0	0	3
0	0	0	1	0	0	4
0	0	0	0	1	0	5
0	0	0	0	0	1	6

OneHotEncoding:

- Remove rows or columns with missing data:

```
from sklearn.preprocessing import OneHotEncoder

# creates the encoder object
enc = OneHotEncoder(handle_unknown='ignore', sparse_output=False)

# encodes and replaces the original values on the dataframe
values = enc.fit_transform(df.loc[:, ['Sex']])

# checks the categories name ['female', 'male']
enc.categories_

# adds the columns to the dataframe
df['Female'] = values[:, 0]
df['Male'] = values[:, 1]

# removes the categorical column
df.drop(columns=['Sex'])
```

Next lecture:

- Classification