COM1011: Fundamentals of Machine Learning

Lecturer: Dr Chico Camargo [f.camargo@exeter.ac.uk]

Practical Session: *Hierarchical Clustering*

In this practical we'll cluster data describing cancer tumours. The table includes a `diagnosis` column, containing `M` for malign tumours and `B` for benign tumours, followed by columns with numerical values.

1. Read the `cancer.csv` file into a `pandas` dataframe.

2. What is the shape of this dataset? What are the first few lines like?
Hint: use pandas functions `shape` and `head()`.

3. Choose a pair of columns (but not `diagnosis`), and make a scatterplot with one column in each axis.
   a. **Option 1:** Create a new column in the dataframe, containing a zero where `diagnosis == 'B'` and a 1 where `diagnosis=='M'`. Then make a scatterplot, colour the points according to this new column.
   b. **Option 2:** Try using a tool from the `seaborn` library, such as `sns.pairplot`, as below:

```
sns.pairplot(df[['radius_mean','texture_mean', 'diagnosis']], hue="diagnosis", height=5)
```

Hint: Try adding `'area_mean'` to the list of columns in the line above!

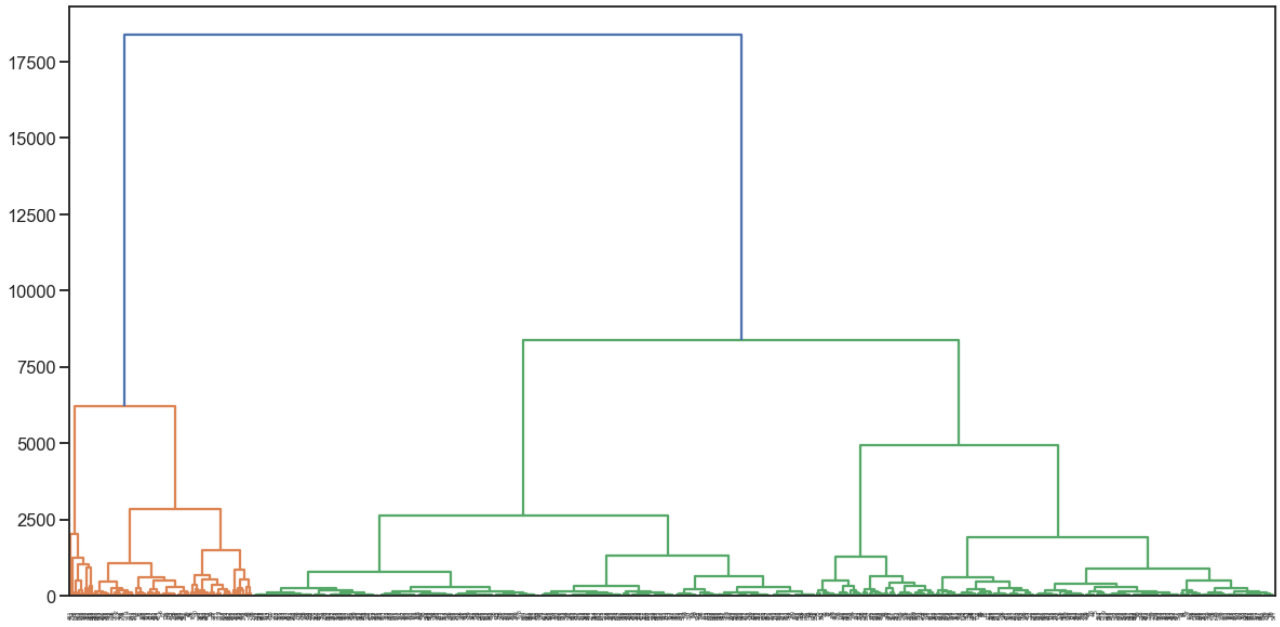
4. Using the internet, find the documentation of the `linkage()` function in Python.
This function builds a dendrogram from your data. It takes a matrix (or 2D array) of values as input, and produces a matrix with N rows and 4 columns. Read its documentation, see if you can understand what those columns represent.
   a. In the documentation, see if you can identify how to specify the method for cluster distance. If the user doesn't specify any method, which one is used by default? Simple linkage? Complete linkage? Centroid? Ward's method? Something else?
   b. In the documentation, see if you can identify how to specify the metric for the distance between points. If the user doesn't specify any metric, which one is used by default? L1 norm? L2 norm? Something else?

1. To apply the `linkage()` function to the data coming from the table, create a variable `X`, and assign it the matrix containing the values from all columns except for `diagnosis`.
Hint: for a dataframe `df`, the command `df.columns` returns a list containing the names of all columns in `df`, from the 0th to the last one. Can you guess what `df.columns[1:]` returns?

2. Import the `linkage` function from the `scipy.cluster.hierarchy` library, and run it on `X`, using *Euclidean distance* and *Ward's method*. Save the output to a variable `dend`.

3. Import the `dendrogram` function from the `scipy.cluster.hierarchy` library, and run it on the variable `dend`.

Note: this part took 60 seconds on my computer. Don't worry if it takes a little while!
The output should be something like this:



4. Now we're going to do the same using `sklearn`. Import the `AgglomerativeClustering()` function from `sklearn.cluster`, and define it with the arguments below:
```
agg_clust = AgglomerativeClustering(n_clusters = 2, affinity = "euclidean", linkage = "ward")
```

5. Run the `AgglomerativeClustering` object on the data and save the outputs to a new column on the dataframe:
```
cluster_labels = agg_clust.fit_predict(X)
df["cluster"] = cluster_labels
```

6. Make the same scatterplot as above, but using the cluster labels (in the column `cluster`) as colours.

7. Repeat the hierarchical clustering task using `AgglomerativeClustering` by increasing the number of cluster from 2 to 10. Scatterplot the corresponding results for every `n_clusters`.

8. Import `silhouette_score` from `sklearn.metrics` and repeat question **11**, but instead of plotting a scatterplot, calculate the silhouette score for every `n_clusters` by using:

```
agg_clust.fit_predict(X)
sil_score = silhouette_score(X, agg_clust.labels_, metric='euclidean')
```

For every `n_clusters`, print `n_clusters` and the corresponding `sil_score`.
Which `n_clusters` produces the best silhouette score?