AI5213/EC4213: Machine Learning and Deep Learning **Student ID: 20242084**
Homework 4 (SVM, NNs, Survival Analysis, Clustering) **Name: Nguyen Van Thanh**

Due: Sunday, Dec 7, 11:59PM

This homework comprises four coding exercises from the ISLP book. Please start early!

> Guideline for those new to data analysis using Python:
> Review the Lab section within each chapter prior to tackling the programming tasks.
> (e.g., p. 489 of Chapter 11 or https://islp.readthedocs.io/en/latest/labs/Ch11-surv-lab.html)
> Find datasets and Jupyter notebooks at https://github.com/intro-stat-learning/ISLP_labs/.

Deliverables: Submit a PDF of your homework, with an appendix listing all your code, to the Gradescope assignment entitled "HW4". You may typeset your homework in LaTeX. Make sure each solution is on a new page, and graphs are included in the correct sections. We need each solution to be self-contained on pages of its own.

Guideline:

1. On the first page of your write-up, please sign next to the integrity statement. We want to make extra clear the consequences of cheating.
2. On the first page of your write-up, please list students who helped you or whom you helped on the homework. (Note that sending each other code is not allowed.)
3. Please write your answers in English. Korean is not allowed. Non-Korean staff members may not be able to grade responses in Korean.
4. Please don't forget to select all the pages that are related to each question during the Gradescope submission! (Submissions that do not clearly reference the exact pages containing the solution may not be graded.)

For staff use only

Honor Code	Q1	Q2	Q3	Q4	Total
/ 4	/ 24	/ 24	/ 24	/ 24	/ 100

Honor Code [4 pts]

Declare and sign the following statement:

"I certify that all solutions in this document are entirely my own and that I have not looked at anyone else's solution. I have given credit to all external sources I consulted."

Signature: Nguyen Van Thanh

We welcome group discussions, but the work you submit should be entirely your own. If you use any information or pictures not from our lectures or readings, make sure to say where they came from. Please note that breaking academic rules can lead to severe penalties.

(a) Did you receive any help whatsoever from anyone in solving this assignment? If your answer is 'yes', give full details (e.g. "Junho explained to me what is asked in Q2-a")

Solution: NO

(b) Did you give any help whatsoever to anyone in solving this assignment? If your answer is 'yes', give full details (e.g. "I pointed Josh to Ch. 2.3 since he didn't know how to proceed with Q2")

Solution: NO

(c) Did you find or come across code that implements any part of this assignment? If your answer is 'yes', give full details (book & page, URL & location within the page, etc.).

Solution: NO

Q1. Support Vector Machine ⌨ [24 pts]

At the end of ISLP Chapter 9.6.1, it is claimed that in the case of data that is just barely linearly separable, a support vector classifier with a small value of C that misclassifies a couple of training observations may perform better on test data than one with a huge value of C that does not misclassify any training observations. You will now investigate this claim.

(a) Generate two-class data with $p = 2$ in such a way that the classes are just barely linearly separable. [8 pts]

Solution:

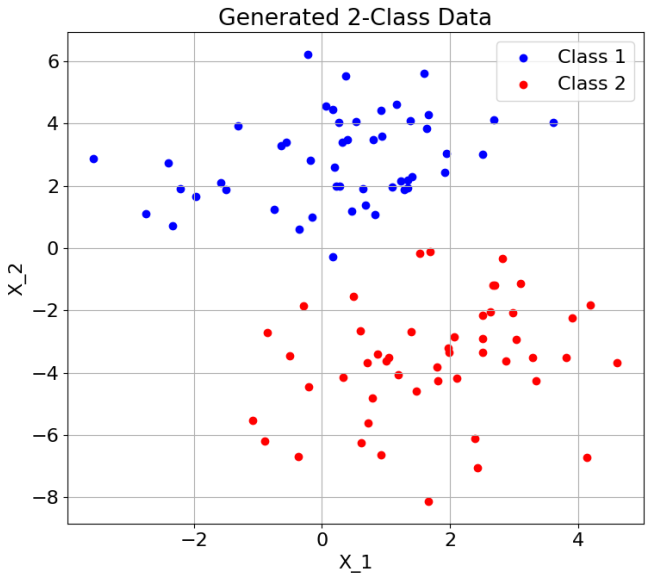

Figure 1: A 2-class generated data

In this solution, 2 classes are generated from 2D normal distributions:

$$\begin{aligned} Pr(X|C_1) &= \mathcal{N}(\mu_1, \sigma_1) \\ Pr(X|C_2) &= \mathcal{N}(\mu_2, \sigma_2) \end{aligned} \tag{1}$$

where μ_1 and μ_2 are mean vectors and σ_1 and σ_2 are covariance matrices.
In order to handle the separability level of the 2 classes, we control the distance between two mean vector μ_1 and μ_2, i.e $||\mu_1 - \mu_2||$. If $||\mu_1 - \mu_2||$ large, then 2 classes become more separable and vice versa. Main properties of the data set includings:

- Total observations/samples: 100 (50 for each class)
- Mean vector: $\mu_1 = [0.2, 2.8], \mu_2 = [2, -3]$
- Covariance matrices:

$$\sigma_1 = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}, \qquad \sigma_2 = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}.$$

(b) Compute the cross-validation error rates for support vector classifiers with a range of C values. How many training observations are misclassified for each value of C considered, and how does this relate to the cross-validation errors obtained? [8 pts]

Solution:
We consider C, which is required to be strictly possivtive, in the following set:

$$C \in [0.001,\ 0.01,\ 0.1,\ 1,\ 10,\ 100,\ 1000]$$

- In part (a), we generate 100 observations, we apply a $K - fold$ cross-validation with $K = 5$ on the data set. For each value of C, the error rate is taken as the mean value of 5-fold cross-validation. The calculated result is given in Table 1 below:

Table 1: Cross-Validation Error for Different Values of C

C	cross-validation error rate
0.001	0.020
0.01	0.010
0.1	0.010
1	0.010
10	0.010
100	0.010
1000	0.010

- The number of missclassified observations on the generated data set as C varies is given in the Table 2 below

Table 2: Number of Misclassified Observations for Different Values of C

C	No. of Misclassified Observations
0.001	2
0.01	2
0.1	0
1	0
10	0
100	0
1000	0

- We see that as C increases, the error rate decreases and the number of misclassified observations decreases also. We illustrate the decision boundary for 2 representatives $C = 0.001$ and $C = 100$ as in Fig 2 below.

Solution:

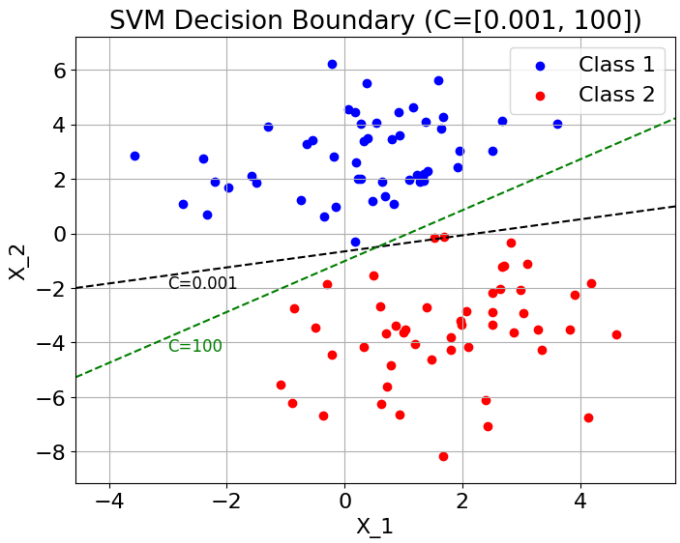

Figure 2: Decision boundaries for $C = 0.001$ and $C = 100$

(c) Generate an appropriate test data set, and compute the test errors corresponding to each of the values of C considered. Which value of C leads to the fewest test errors, and how does this compare to the values of C that yield the fewest training errors and the fewest cross-validation errors? [8 pts]

Solution:

- We generate a new test set with the same parameters as described in part (a), just change the random control parameter(np.random.seed()). The test is visualized as below:

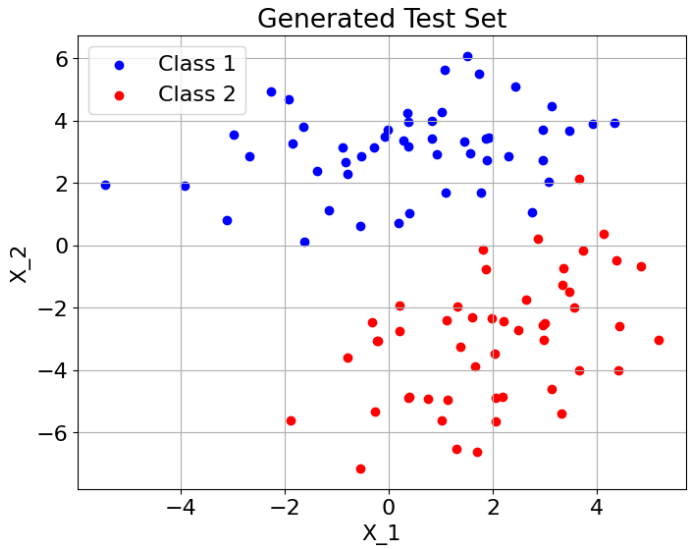

Figure 3: The generated test set.

- We validate the classifiers trained in part (b) for

$$C \in [0.001,\ 0.01,\ 0.1,\ 1,\ 10,\ 100,\ 1000]$$

with the generated test set. The error rate is shown in Table 3 below.

Table 3: Number of Misclassified Observations for Different Values of C

C	Test error rate
0.001	0.020
0.01	0.010
0.1	0.010
1	0.020
10	0.010
100	0.010
1000	0.010

We see that for $C = 0.1$, the test error rate is smaller than the case $C = 1$. The decision boundary for those 2 cases are shown in Fig 4 below. We see that for $C = 0.1$, there is one missclassified observation whereas there are two for $C = 1$

Solution:

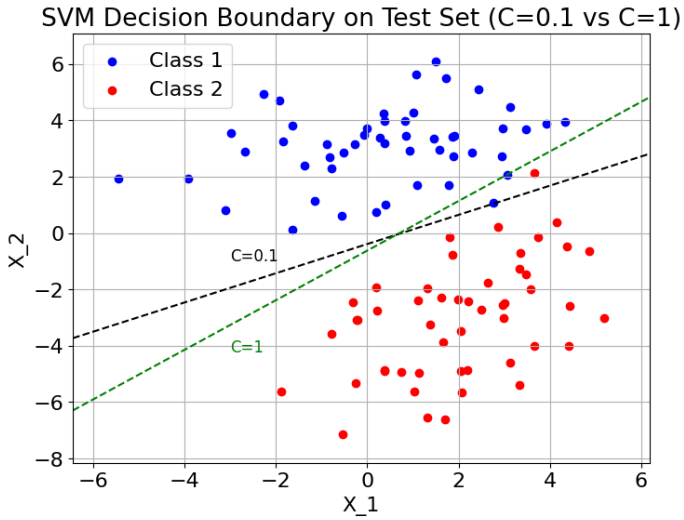

Figure 4: Decision boundary for $C = 0.1$ and $C = 1$ on the test set

Q2. RNN ⌨ [24 pts]

In ISLP Chapter 10.9.6 (page 458), we showed how to fit a linear AR model to the NYSE data using the LinearRegression() function. However, we also mentioned that we can "flatten" the short sequences produced for the RNN model in order to fit a linear AR model. Use this latter approach to fit a linear AR model to the NYSE data.

(a) Compare the test R^2 of this linear AR model to that of the linear AR model that we fit in the lab (ISLP Chapter 10.9.6). What are the advantages/disadvantages of each approach? [8 pts]

Solution: The procedure performed in this task is as follow:

- First we create a lagged data (5-day lagged)
- Fit a linear AR model with the lagged data. This is the flatten data used for the RNN model
- Reshape the lagged data and feed to an RNN model

The results for the test R^2 of the linear AR model and RNN model is given below

Table 4: Comparison of R^2 Scores for Linear AR and RNN Models

Model	**R^2 Score**
Linear AR	0.412891
RNN	0.416047

We see that the RNN model has a higher R^2 score which is better in terms of matching between model's prediction and actual data.
Advantages and disadvantages of **a linear AR model**:

- Advantages:
  - Has a fewer parameters. Therefore, training process is much faster
  - Easier to interpret the relation between the response and features
  - Low variance
- Disadvantages
  - Can not capture nonlinear relation between the response and features
  - High bias

Advantages and disadvantages of an **RNN** model:

- Advantages:
  - Able to capture nonlinear relation between response and features (has a better performance in terms of R^2 and test loss compared to linear AR model)
  - Low bias

Solution:

- Disadvantages
  - Interpreting the response and feature relation is not straightforward
  - High variance
  - Have more parameters, training time is longger.

(b) Repeat the previous exercise, but now fit a nonlinear AR model by "flattening" the short sequences produced for the RNN model. [8 pts]

Solution: We create a feedforward neural network with following specifications:

- 1 hidden layer with 32 units
- use ReLU activation function
- Dropout 10%

The class for this nonlinear AR model is given as in the code below:

```
class NonLinearARModel(nn.Module):
    def init(self, inputsize):
        super(NonLinearARModel, self).init()
        self.flatten = nn.Flatten()
        self.sequential = nn.Sequential(
            nn.Linear(inputsize, 32), # input to hidden, 32 units
            nn.ReLU(), # activation function, inject non-linearity
            nn.Dropout(0.1), # 10% dropout
            nn.Linear(32, 1)) # hidden to output
    def forward(self, x):
        x = self.flatten(x)
        return torch.flatten(self.sequential(x))
```

Solution: For this model we have following parameters:

- Input to hidden: 15x32 = 480 weights + 15 bias = 512 parameters
- Hidden to output: 32 weight + 1 bias = 33 params

We have 512 + 33 = 545 parameters
We train the model with the flatten data set as used in part (a). We have following comparison results:

Table 5: Comparison of R^2 Scores for Linear AR, Nonlinear AR and RNN Models

Model	**R^2 Score**
Linear AR	0.412891
RNN	0.416047
Nonlinear AR	0.419263

We see that the nonlinear AR model has a better performance in terms of R^2 score index. Advantages of nonlinear AR model

- Able to capture the nonlinearity relation in the data
- Low bias compare to linear AR model

Disavantages:

- Can not capture the sequence dependent property like RNN
- Has much more parameter compare to linear AR model

(c) Modify the code with a 12-level factor representing the month, and the variable day_of_week. Do these factors improve the performance of the model? Compute the test R^2. [8 pts]

Solution:

- For an RNN model, if we include day_of_week and month feature, the feature dimension now become 20 (3 original ones + 5 day_of_week + 12 months). For Linear and Nonlinear AR model, it will be 100 as we flatten the data (5x20).
- The data processing is performed to include day_of_week and moth features.
- For then RNN and nonlinear AR models, as the number of features is increased, the number of parameters increases significantly as a result. However, we keep setting the same number of hidden layer's units as in part (a) and (b). Table below summarizes the RNN and nonlinear AR models' specifications.

Table 6: RNN and nonlinear AR models' specification

Model	**# of hidden layer**	**# of hidden layer's units**	**# of parameters**
RNN	1	12	421
Nonlinear AR	1	32	3265

The table below show the R^2 performance results of the three models as adding day_of_week and month features.

Table 7: Comparison of R² Scores for Linear AR, Nonlinear AR and RNN Models

Model	**R² Score**
Linear AR	0.465049
RNN	0.445714
Nonlinear AR	0.422710

As we compare to part (b), adding day_of_week and month features does improve the three models' performance.

Q3. Survival Analysis ⌨ [24 pts]

This exercise makes use of the data in the below table.

Observation (Y)	Censoring Indicator (δ)	Covariate (X)
26.5	1	0.1
37.2	1	11
57.3	1	-0.3
90.8	0	2.8
20.2	0	1.8
89.8	0	0.4

(a) Create two groups of observations. In Group 1, $X < 2$, whereas in Group 2, $X \geq 2$. Plot the Kaplan-Meier survival curves corresponding to the two groups. Be sure to label the curves so that it is clear which curve corresponds to which group. By eye, does there appear to be difference between the two groups' survival curves? [8 pts]

Solution:

It is easy to divide the data into two groups as required below:

Table 8: Group 1 ($X \geq 2$)

observation (Y)	**censoring indicator (δ)**	**covariate (X)**
37.2	1	11.0
90.8	0	2.8

Table 9: Group 2 ($X < 2$)

observation (Y)	**censoring indicator (δ)**	**covariate (X)**
26.5	1	0.1
57.3	1	-0.3
20.2	0	1.8
89.8	0	0.4

The Kaplan-Meier survival curves corresponding to the two groups is given as the figure below

Solution:

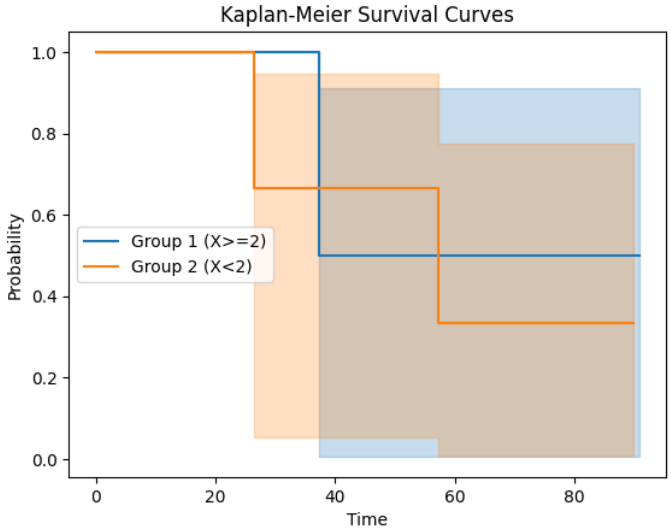

Figure 5: Kaplan Meier survival curve of the two groups

From the figure above, it seems like there is a slightly different between the two groups' survival curves.

(b) Fit Cox's proportional hazards model, using the group indicator as a covariate. What is the estimated coefficient? Write a sentence providing the interpretation of this coefficient, in terms of the hazard or the instantaneous probability of the event. Is there evidence that the true coefficient value is non-zero? [8 pts]

Solution: For this problem, we first encode a feature called **group** that takes the value 1 if $X \geq 2$ else 0. We have the following data frame to fit a Cox's proportional hazards model.

Table 10: Data frame with covariate **group** to fit a Cox's proportional hazards model

observation_Y	censoring	group
26.5	1	0
37.2	1	1
57.3	1	0
90.8	0	1
20.2	0	0

The fitting result is given as below

Table 11: Coefficient Estimates

Variable	coef	se(coef)	p
group	-0.340143	1.235876	0.783144

The estimated coefficient is $\beta = -0.340143$ show that being in group 1 ($X \geq 2$) is associated with a decrease in the hazard by a factor of:

$$\exp(-0.340143) = 0.71$$

This means individuals in group 1 have roughly 29% lower instantaneous risk of the event.
There is no an evidence that the true coefficient value is non-zero because:

$$p - value = 0.783144 >> 0.05$$

which means the estimate is not statistically significant (sample size is small).

(c) Recall that in the case of a single binary covariate, the log-rank test statistic should be identical to the score statistic for the Cox model. Conduct a log-rank test to determine whether there is a difference between the survival curves for the two groups. How does the p-value for the log-rank test statistic compare to the p-value for the score statistic for the Cox model from (b)? [8 pts]

Solution: The result for the log-rank test for the group binary covariate is given in the table below:

Table 12: Log-rank Test Result

t_0	-1
null distribution	chi squared
degrees of freedom	1
test name	logrank_test
test statistic	0.08
p-value	0.78
$-\log_2(p)$	0.35

From the log-rank test, we conclude that:

- There is **no statistical evidence** of a difference in the survival curves between the two groups because the $p - value = 0.78$ is large ($>> 0.05$)

Compare to part (b), we see that the p-value between two cases are closed: 0.783144 (Cox model) vs 0.78 (log-rank)

Q4. Hierarchical Clustering ⌨ [24 pts]

On the book website, there is a gene expression data set (https://www.statlearning.com/s/Ch12Ex13.csv) that consists of 40 tissue samples with measurements on 1000 genes. The first 20 samples are from healthy patients, while the second 20 are from a diseased group.

(a) Load in the data using pd.read_csv() with header = None. Then, apply hierarchical clustering to the samples using correlation-based distance, and plot the dendrogram. Do the genes separate the samples into the two groups? Do your results depend on the type of linkage used? [12 pts]

Solution:
In order to use the correlation-based distance as a measure for the dissimilarity, we set the attribute **metric='precomputed'** in the class **AgglomerativeClustering**. In this case, we need to feed a **correlation distance matrix** for the method **fit()**. The captured code below shows the main steps of the solution. In this code, we use **complete** linkage

```
HClust = AgglomerativeClustering # for a shorter name
# Compute correlation-based distance
# Correlation-based distance = 1 - correlation
correlationmatrix = genedataset.T.corr()
correlationdistance = 1 - correlationmatrix # correlation distance

# Perform hierarchical clustering with complete linkage
hccomp = HClust(distancethreshold=0,
                nclusters=None,
                linkage='complete',
                metric='precomputed')  # Use precomputed distance matrix
hccomp.fit(correlationdistance)
```

After clustering the data, we compute the linkage matrix to plot the dendrogram for the fitted model.

```
# Compute linkage for dendrogram plotting
linkagematrix = computelinkage(hccomp)
# Plot the dendrogram
from matplotlib.pyplot import subplots
fig, ax = subplots(figsize=(12, 6))
dendrogram(linkagematrix, ax=ax)
ax.setxlabel('Sample')
ax.setylabel('Correlation-based Distance')
ax.settitle('Dendrogram (Complete Linkage)')
fig.tightlayout()
```

The resulted dendrogram is shown as below:

Solution:

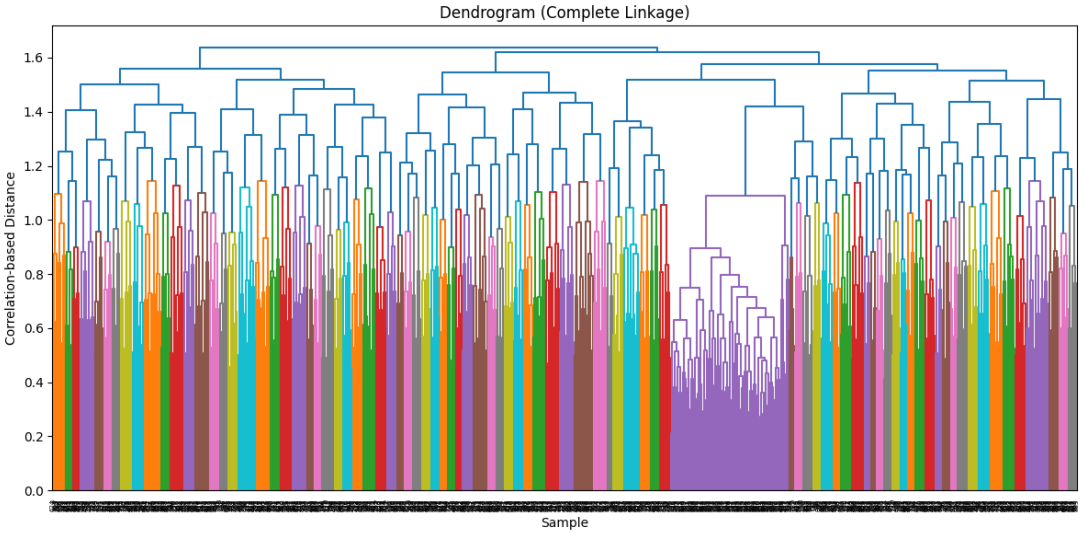

Figure 6: Dendrogram of the clustering model using **complete** linkage

- Do the genes separate the samples into the two groups?
  Yes. From the dendrogram, we see that there is a group of samples that seems likely to have similarity compare to others.
- Do the results depend on the type of linkage used
  Yes, they do depend on the type of linkage used. The figure below show the denndrogram as we use **"average"** linkage. Compare to "complete" linkage, the separated group can be shifted.

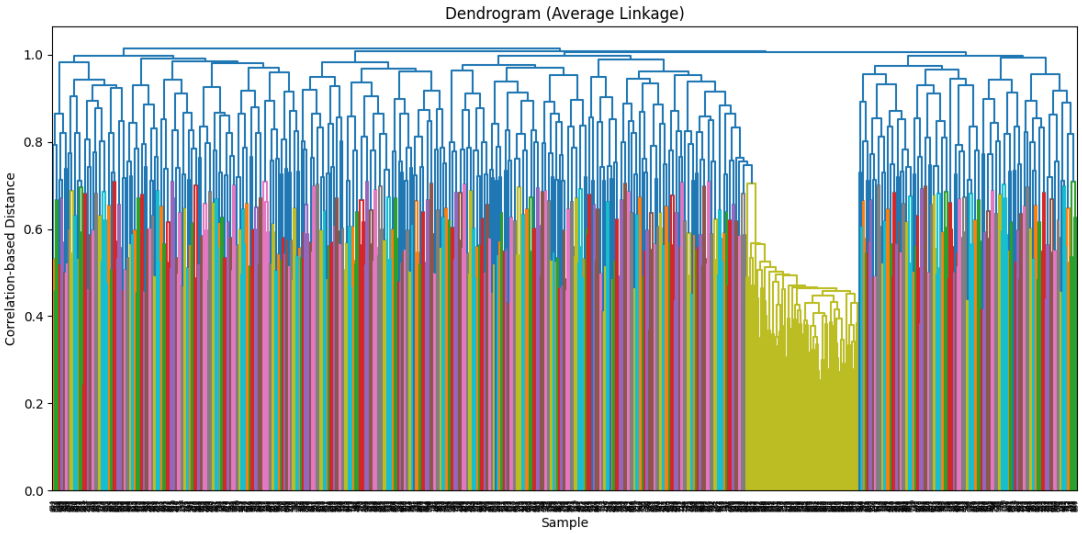

Figure 7: Dendrogram of the clustering model using **average** linkage

(b) Your collaborator wants to know which genes differ the most across the two groups. Suggest a way to answer this question, and apply it here. [12 pts]

Solution: To solve this, for each of the 1000 genes, we compute the **t-statistic** for the two groups, 20 healthy people and 20 patients. If a gene has a large —**t**— value, then it is different among the two groups. We use $|t|$ to rank the genes' differences across the two groups. We use **ttest_ind** from **scipy** to compute the t-statistics values:

```
from scipy.stats import ttestind
```

Process the data to divide the set into healthy and patient groups:

```
genedataset = genedataset.T # transpose to have genes in columns
healthygroup = genedataset.iloc[:20,:] # healthy group in the data set
patientgroup = genedataset.iloc[20:,:]  # patient group in the data set
tstats = [] # to store t-statistics for each gene
```

Next we compute the absolute **t-statistics** value for each gene among the two groups:

```
for gene in range(genedataset.shape[1]):
    stat, pvalue = ttestind(healthygroup.iloc[:,gene],
                            patientgroup.iloc[:,gene],
                            equalvar=False) # Welch's t-test
    tstats.append(np.absolute(stat)) # store absolute t-statistic value

# Create a DataFrame to store t-statistics values
tstatistictestresult = pd.DataFrame(-
    "gene": np.arange(genedataset.shape[1]),
    "tstat": tstats,
)
```

Finally we rank the genes by their t-statistics value. The more higher t-statistics value, the more different that gene is between the two groups.

```
# sort genes by absolute t-statistic values in descending order
tstatistictestresult = tstatistictestresult.sortvalues(by='tstat',""
ascending=False)

print(tstatistictestresult.head(5)) # print top 5 genes
```

The top 5 genes is given in the table below:

Gene	t-statistic
501	10.316705
588	10.024607
599	9.623413
589	8.976247
564	8.628749

Table 13: Top genes ranked by t-statistic

Appendix

The source code for this Homework is given as below:
https://github.com/ThanhNV-Robotics/MLDL-HW4