CAPAIAN PEMBELAJARAN PRAKTIKUM

1. Menjelaskan konsep dari pembelajaran mesin kemudian di terapkan pada permasalahan – (C2)
2. Mengimplementasikan algoritma dan/atau metode di pembelajaran mesin untuk mendapatkan pemecahan maslaah dan model yang sesuai – (C3)
3. Menghasilkan suatu model terbaik dari permasalahan dan terus melakukan jika terdapat perkembangan di bidang rekayasa cerdas – (C6)

MINGGU 10

Algoritma Hierarki

DESKRIPSI TEMA

Mahasiswa mempelajari pembelajaran tidak terbimbing dengan algoritma hierarki yaitu algoritma Agglomerative Single Link, Avarage Link dan Complete Link

CAPAIAN PEMBELAJARAN MINGGUAN (SUB CPMK)

Mahasiswa dapat menerapkan algoritma hirarki sehingga bisa menghasilkan model terbaik

PERALATAN YANG DIGUNAKAN

Anaconda Python 3
Laptop atau Personal Computer

LANGKAH-LANGKAH PRAKTIKUM

In machine learning, the types of learning broadly be classified into 3 types. So this is one of three types. It called Unsupervised Learning. Algorithms belonging to the family of unsupervised learning have no variable to predict tied to the data. Here some of the algorithm.

Hierarchical Clustering : Agglomerative Clustering

Also known as Agglomerative Clustering, does not require the user to specify the number of clusters. Initially, each point is considered as a separated cluster, then it recursively clusters the points together depending upon the distance between them. The point are clustered in such a way that the distance between points within a cluster is minimum and distance between the cluster is maximum. Unlike K Means clustering, it is bottom – up approach, because starting from the leaves and combining clusters up to the trunk.

The approach in small words :

- Start with each point in its own cluster.
- Identify the closest two clusters and merge them.
- Repeat.

- Ends when all points are in a single cluster

1. Import the important libraries and read the dataset (‘German_Credit_Card.csv’)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv('german_credit_data.csv')
```

2. Take a quick look the dataset :

```
df.shape
```

```
(1000, 10)
```

```
df.head()
```

	Unnamed: 0	Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose
0	0	67	male	2	own	NaN	little	1169	6	radio/TV
1	1	22	female	2	own	little	moderate	5951	48	radio/TV
2	2	49	male	1	own	little	NaN	2096	12	education
3	3	45	male	2	free	little	little	7882	42	furniture/equipment
4	4	53	male	2	free	little	little	4870	24	car

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
Unnamed: 0          1000 non-null int64
Age                 1000 non-null int64
Sex                 1000 non-null object
Job                 1000 non-null int64
Housing             1000 non-null object
Saving accounts     817 non-null object
Checking account    606 non-null object
Credit amount       1000 non-null int64
Duration            1000 non-null int64
Purpose             1000 non-null object
dtypes: int64(5), object(5)
memory usage: 78.2+ KB
```

```
1 df.describe()
```

	Unnamed: 0	Age	Job	Credit amount	Duration
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	499.500000	35.546000	1.904000	3271.258000	20.903000
std	288.819436	11.375469	0.653614	2822.736876	12.058814
min	0.000000	19.000000	0.000000	250.000000	4.000000
25%	249.750000	27.000000	2.000000	1365.500000	12.000000
50%	499.500000	33.000000	2.000000	2319.500000	18.000000
75%	749.250000	42.000000	2.000000	3972.250000	24.000000
max	999.000000	75.000000	3.000000	18424.000000	72.000000

3. Missing values identification handling

```
1 df.isnull().sum()
```

```
Unnamed: 0            0
Age                   0
Sex                   0
Job                   0
Housing               0
Saving accounts     183
Checking account    394
Credit amount         0
Duration              0
Purpose               0
dtype: int64
```

```
numerical = ['Credit amount','Age','Duration']
categorical = ['Sex','Job','Housing','Saving accounts','Checking account','Purpose']
unused = ['Unnamed: 0']
```

```
df = df.drop(columns = unused)
df.shape
```

```
(1000, 9)
```

```
for cat in categorical:
    df[cat] = df[cat].fillna(df[cat].mode().values[0])
```

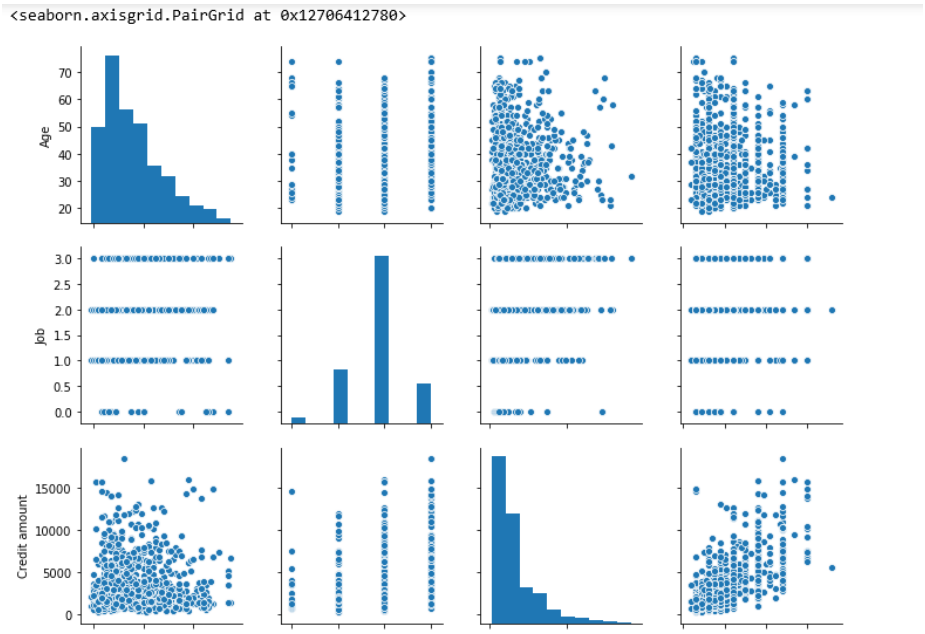
```
df.isnull().sum()
```

```
Age                 0
Sex                 0
Job                 0
Housing             0
Saving accounts     0
Checking account    0
Credit amount       0
Duration            0
Purpose             0
dtype: int64
```

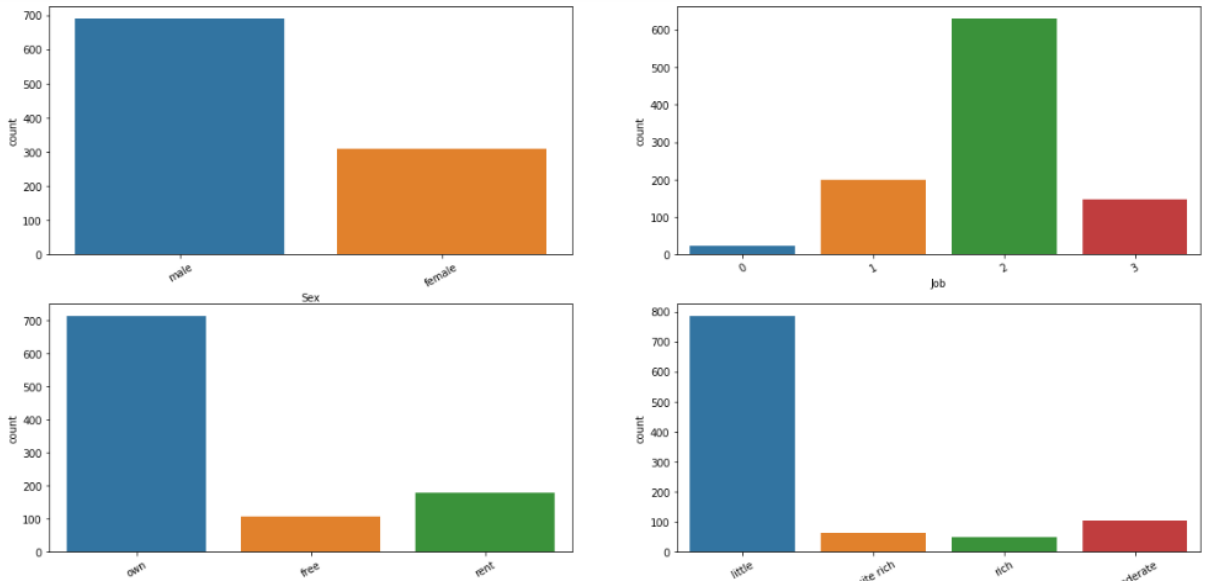
4. Let's visualize the dataset

```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x12706412780>
```



```
fig = plt.figure(figsize = (20,15))
axes = 320
for cat in categorical:
    axes += 1
    fig.add_subplot(axes)
    sns.countplot(data = df, x = cat)
    plt.xticks(rotation=30)
plt.show()
```



Above are the bar plot of all categorical feature. From the bar plot above, we can get some insight. That are :

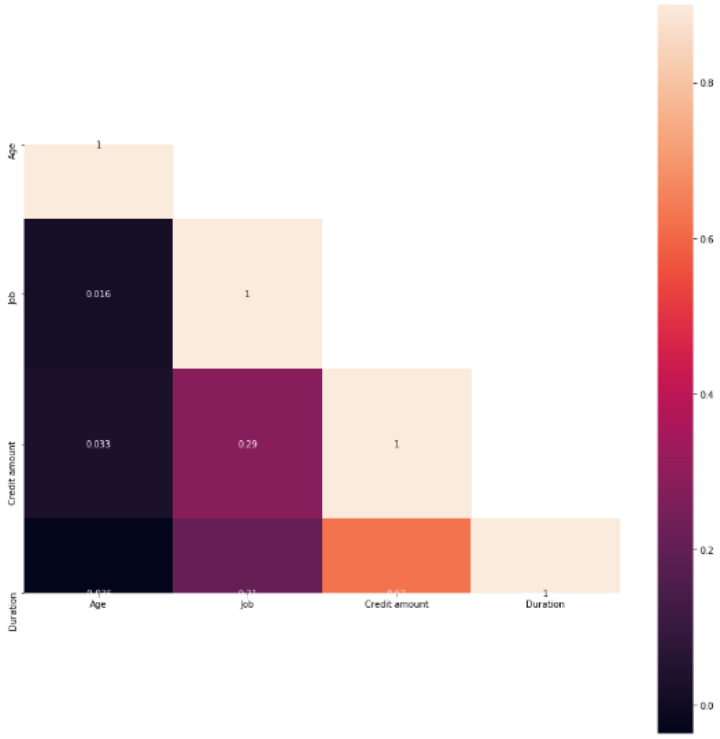
1. The amount of men are greater than women
2. Most of the customer are skilled on their job
3. Most of the customer have their own house
4. Most of the customer have little saving account
5. Most of the customer have little checking account
6. Most of the customer use credit for car

```python
#create correlation
corr = df.corr(method = 'pearson')

#convert correlation to numpy array
mask = np.array(corr)

#to mask the repetitive value for each pair
mask[np.tril_indices_from(mask)] = False
fig, ax = plt.subplots(figsize = (15,12))
fig.set_size_inches(15,15)
sns.heatmap(corr, mask = mask, vmax = 0.9, square = True, annot = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x12707391f98>
```

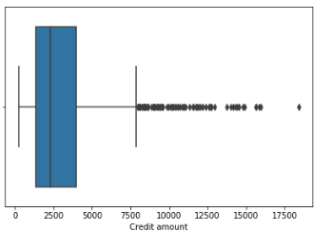


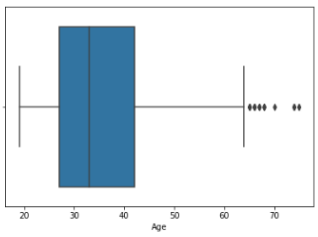
5. From the heatmap above we can see that the best correlation is between credit amount and duration. But so far we will still use all the numeric features for the clustering.

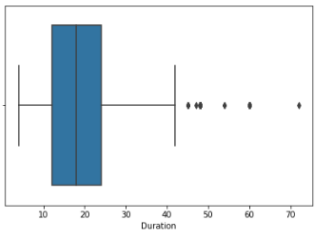
```
df_cluster = pd.DataFrame()
df_cluster['Credit amount'] = df['Credit amount']
df_cluster['Age'] = df['Age']
df_cluster['Duration'] = df['Duration']
df_cluster['Job'] = df['Job']
df_cluster.head()
```

	Credit amount	Age	Duration	Job
0	1169	67	6	2
1	5951	22	48	2
2	2096	49	12	1
3	7882	45	42	2
4	4870	53	24	2

```
fig = plt.figure(figsize = (15,10))
axes = 220
for num in numerical:
    axes += 1
    fig.add_subplot(axes)
    sns.boxplot(data = df, x = num)
plt.show()
```

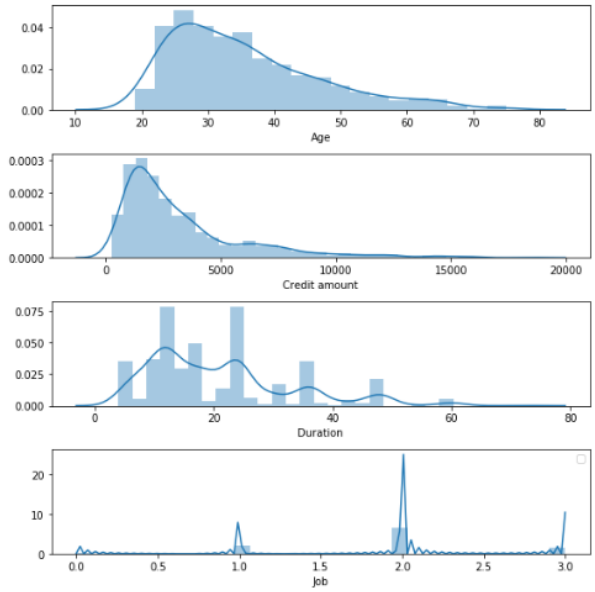







```
fig, (ax1, ax2, ax3, ax4) = plt.subplots(4,1, figsize=(8,8))
sns.distplot(df["Age"], ax=ax1)
sns.distplot(df["Credit amount"], ax=ax2)
sns.distplot(df["Duration"], ax=ax3)
sns.distplot(df["Job"], ax=ax4)
plt.tight_layout()
plt.legend()
```

```
<matplotlib.legend.Legend at 0x127075db6a0>
```



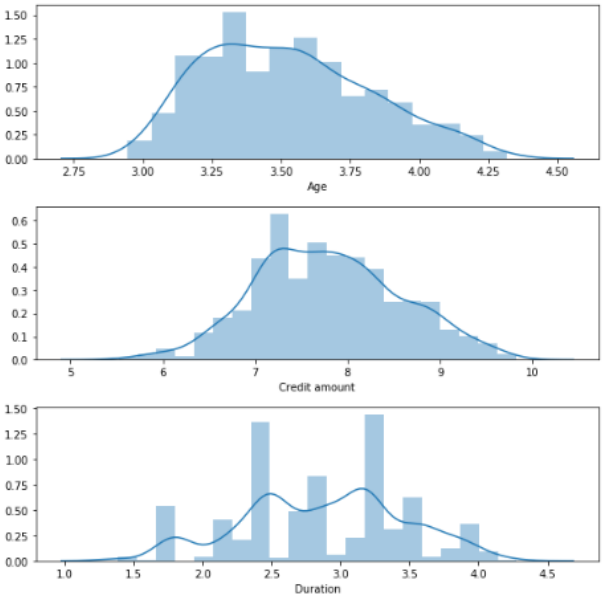
From the figure above, we can see that distributions are right-skewed.

Feature Engineering

6. We can use logarithmic transformation to reduce the outliers and distribution skewness.

```
df_cluster_log = np.log(df_cluster[['Age', 'Credit amount','Duration']])

fig, (ax1, ax2, ax3) = plt.subplots(3,1, figsize=(8,8))
sns.distplot(df_cluster_log["Age"], ax=ax1)
sns.distplot(df_cluster_log["Credit amount"], ax=ax2)
sns.distplot(df_cluster_log["Duration"], ax=ax3)
plt.tight_layout()
```



7. We can see that the skewness of the distributions is eliminated. Beside log transformation, we can use Fit and Transform.

```
1  df_cluster_log.head()
```

	Age	Credit amount	Duration
0	4.204693	7.063904	1.791759
1	3.091042	8.691315	3.871201
2	3.891820	7.647786	2.484907
3	3.806662	8.972337	3.737670
4	3.970292	8.490849	3.178054

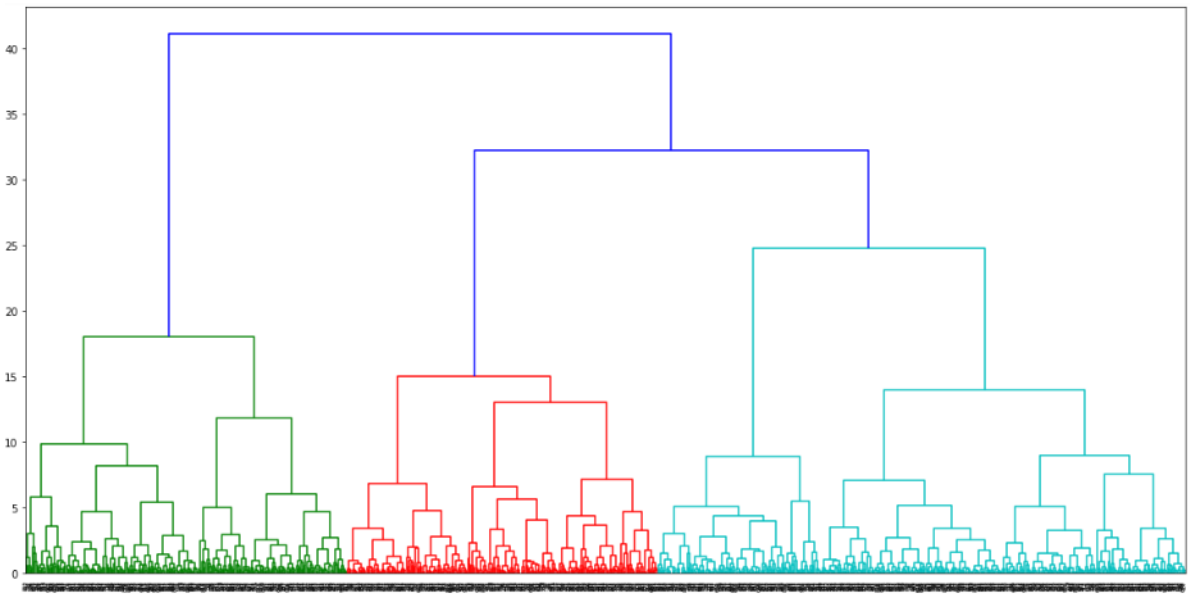
8. Preprocessing the dataset using StandardScaler()

```
1  from sklearn.preprocessing import StandardScaler
2
3  scaler = StandardScaler()
4  cluster_scaled = scaler.fit_transform(df_cluster_log)
```

9. Models the dataset using Hierarchical Agglomerative Clustering.

```python
import scipy.cluster.hierarchy as sch

plt.figure(figsize=(20,10))
dendrogram = sch.dendrogram(sch.linkage(cluster_scaled, method='ward'))
```



10. From the dendrogram above, we can see that the most optimal n_clusters is 4

Make the 3D projection :

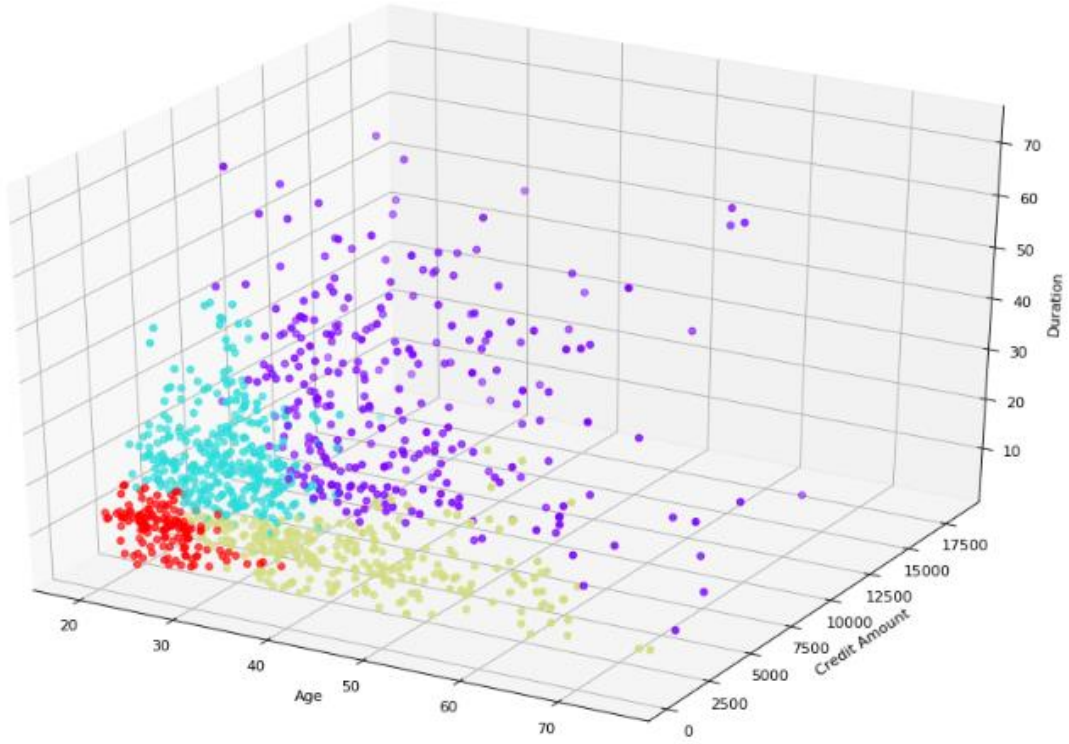
```python
from sklearn.cluster import AgglomerativeClustering
from mpl_toolkits.mplot3d import Axes3D
model = AgglomerativeClustering(n_clusters=4)
model.fit(cluster_scaled)
hac_labels = model.labels_

fig = plt.figure(num=None, figsize=(15, 10), dpi=80, facecolor='w', edgecolor='k')
ax = plt.axes(projection="3d")

ax.scatter3D(df_cluster['Age'],df_cluster['Credit amount'],df_cluster['Duration'],c=hac_labels, cmap='rainbow')

xLabel = ax.set_xlabel('Age', linespacing=3.2)
yLabel = ax.set_ylabel('Credit Amount', linespacing=3.1)
zLabel = ax.set_zlabel('Duration', linespacing=3.4)
print("Hierarchical Agglomerative Clustering")
```

```
Hierarchical Agglomerative Clustering
```




```
df_clustered_hac = df_cluster.assign(Cluster=hac_labels)
grouped_hac = df_clustered_hac.groupby(['Cluster']).mean().round(1)
grouped_hac
```

Cluster	Credit amount	Age	Duration	Job
0	6477.0	40.4	32.0	2.2
1	2748.0	28.8	23.6	1.9
2	1644.1	44.0	11.7	1.7
3	1231.5	25.1	10.6	1.7

The table above shows that centroid of each clusters that could determine the clusters rule. These are :

Cluster 0 : Higher credit amount, old, long duration customers

Cluster 1 : Lower credit amount, young, long duration customers

Cluster 2 : Lower credit amount, old, short duration customers

Cluster 3 : Lower credit amount, young, short duration customers.

The Dendrogram

Dendrogram is so powerful in the hierarchical clustering because dendrogram showing relationship between similar sets of data. They are frequently represent any type of group data. Dendrogram is a module from scipy learn (scipy.cluster.hierarchy.dendrogram). With this diagram, we can illustrates how each cluster is composed by a drawing U-Shaped link between a non-singleton cluster and its children. The top of the U-Link indicates a cluster merge. The two legs of the U-link indicate which clusters were merged. The length of the two legs of the U-link represents the distance between the child clusters. It is also the cophenetic distance between original observations in the two children clusters.

Let's more understand about dendrogram using example below :

11. Import the required libraries :

```
#The Dendrogram using WholesaleCustomer dataset
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

12. Look the data and look at the first few rows

```
data = pd.read_csv('Wholesale_customers_data.csv')
data.head()
```

Out[2]:

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776
2	2	3	6353	8808	7684	2405	3516	7844
3	1	3	13265	1196	4221	6404	507	1788
4	2	3	22615	5410	7198	3915	1777	5185

These are multiple product categories – Fresh, Milk, Grocery, etc. The values represent the number of units purchases by each client for each product. **Our aim, is to make clusters from this data that can segment similar clients together**.

Before applying Hierarchical Clustering, we have to normalize the data so that the scale of each variable is the same. Why is this important? Well, if the scale of the variables is not the same, the model might become biases towards the variables with a higher magnitude like Fresh or Milk (refer to the above table).

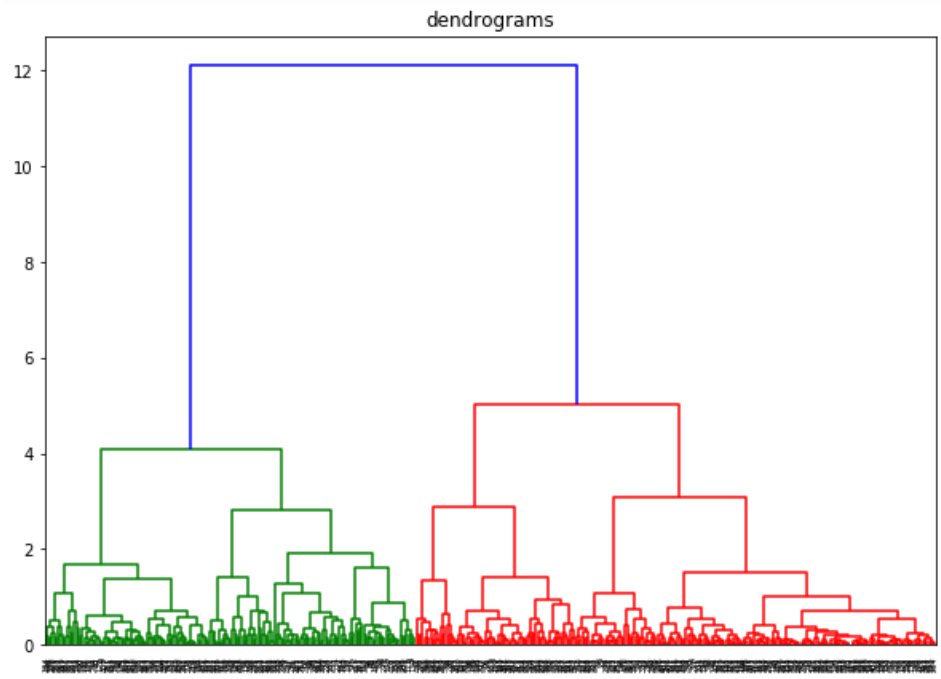
13. Let's normalize the data and bring all the variables to the same scale:

```
from sklearn.preprocessing import normalize
data_scaled=normalize(data)
data_scaled=pd.DataFrame(data_scaled,columns=data.columns)
data_scaled.head()
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	0.000112	0.000168	0.708333	0.539874	0.422741	0.011965	0.149505	0.074809
1	0.000125	0.000188	0.442198	0.614704	0.599540	0.110409	0.206342	0.111286
2	0.000125	0.000187	0.396552	0.549792	0.479632	0.150119	0.219467	0.489619
3	0.000065	0.000194	0.856837	0.077254	0.272650	0.413659	0.032749	0.115494
4	0.000079	0.000119	0.895416	0.214203	0.284997	0.155010	0.070358	0.205294

14. Here we can see that the scale of all the variables is almost similar. Now, we are good to go. Let's first draw the dendrogram to help us decide the number of clusters for this particular problem :

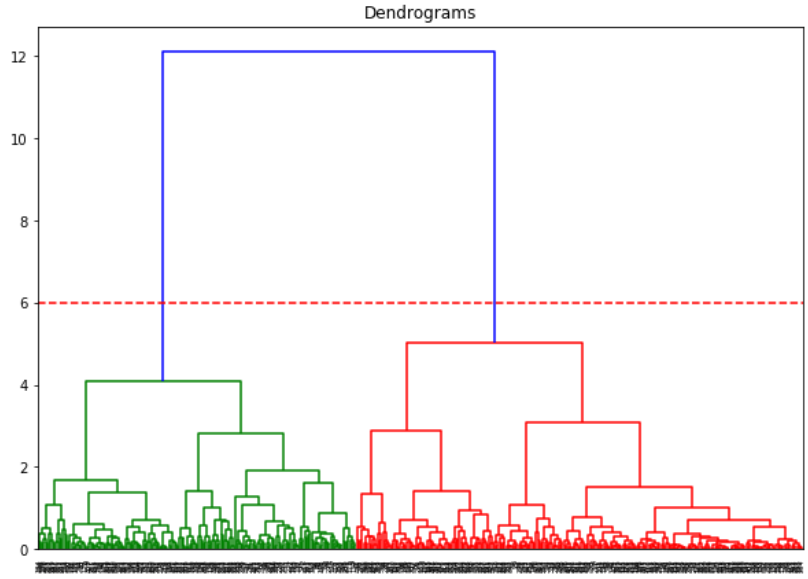
```
import scipy.cluster.hierarchy as shc
plt.figure(figsize=(10,7))
plt.title("dendrograms")
dend=shc.dendrogram(shc.linkage(data_scaled,method='ward'))
```



15. The x-axis contains the samples and y-axis represents the distance between these samples. The vertical line with maximum distance is the blue line and hence we can decide a threshold of 6 and cut the dendrogram :

```
plt.figure(figsize=(10,7))
plt.title("Dendrograms")
dend=shc.dendrogram(shc.linkage(data_scaled,method='ward'))
plt.axhline(y=6,color='r',linestyle='--')
```

```
<matplotlib.lines.Line2D at 0x146b345ab38>
```



16. We have 2 clusters as this line cuts the dendrogram at two points. Let's now apply hierarchical clustering for 2 clusters :

```
from sklearn.cluster import AgglomerativeClustering
cluster=AgglomerativeClustering(n_clusters=2,affinity='euclidean',
                                linkage='ward')
cluster.fit_predict(data_scaled)
```

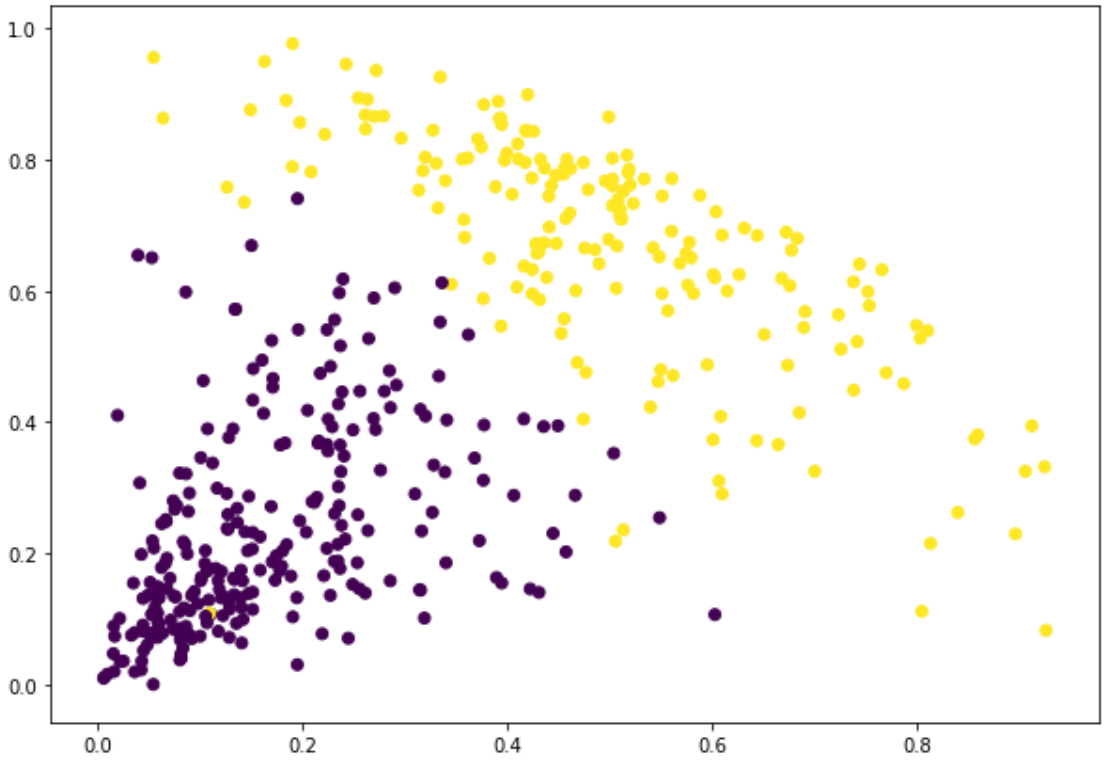
```
array([1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1],
      dtype=int64)
```

We can see the values of 0s and 1s in the output since we defined 2 clusters. 0 represents the points that belong to the first cluster and 1 represents points in the second cluster.

17. Lets now visualize the two clusters :

```
plt.figure(figsize=(10,7))
plt.scatter(data_scaled['Milk'],data_scaled['Grocery'],c=cluster.labels_)
```

```
<matplotlib.collections.PathCollection at 0x146b31c8978>
```



Referensi :

Raschka S, Mirjailili. 2017. Phython Machine Leraning 2nd edition. Packt Publishing