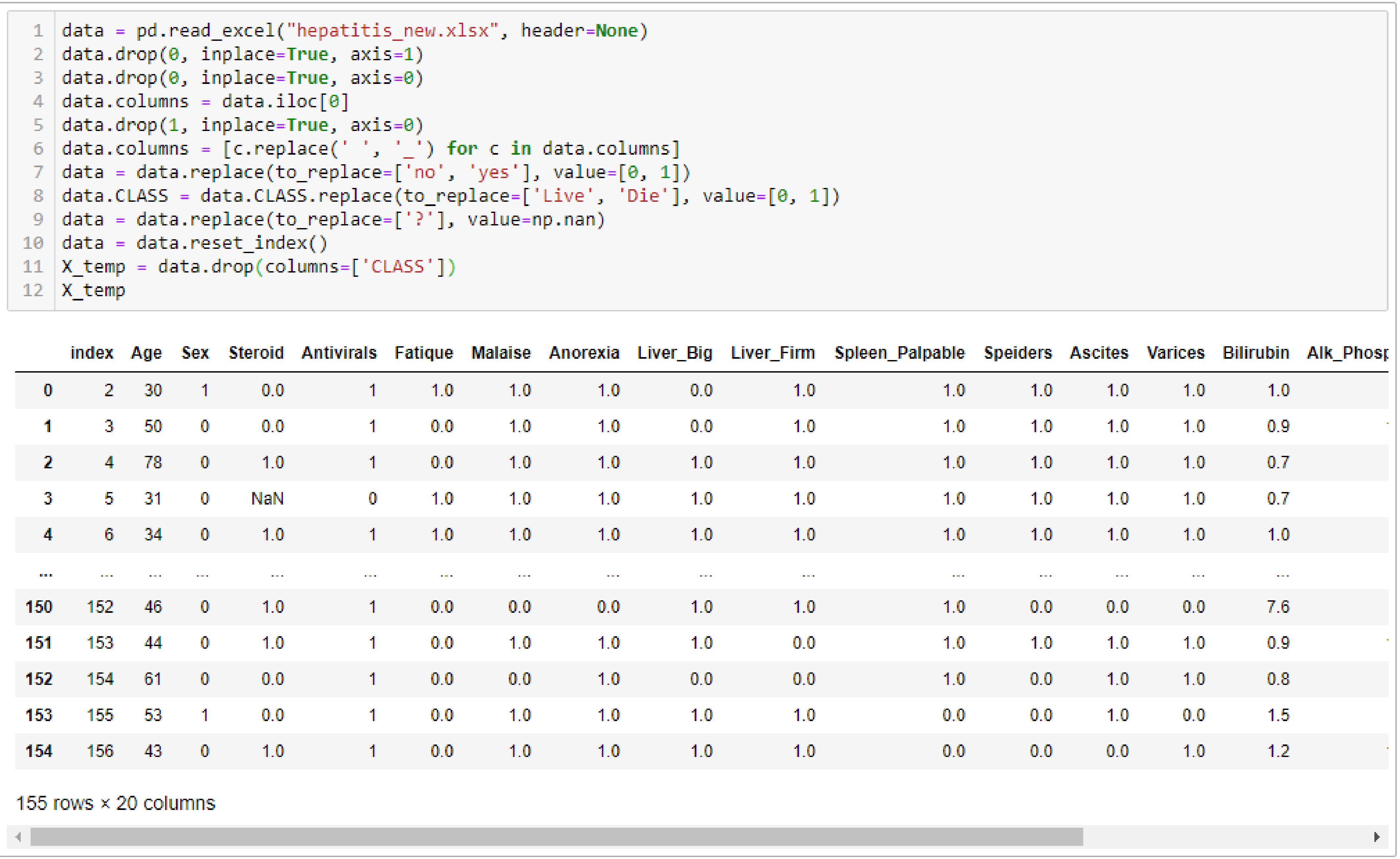
Clustering & Cluster Analysis

Knowledge Discovery

Hanif Izzudin Rahman

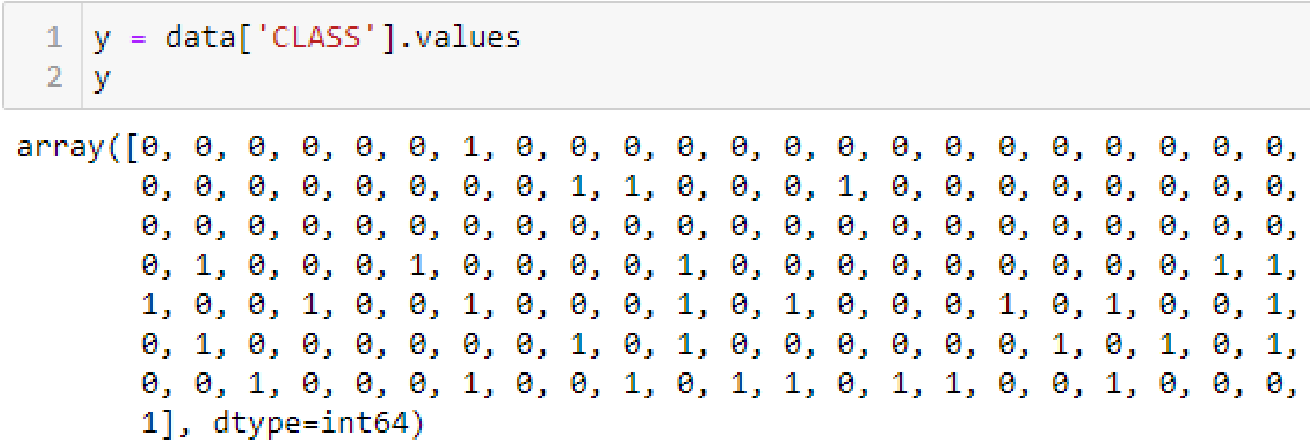
1. Convert that data into the numerical values



```python
data = pd.read_excel("hepatitis_new.xlsx", header=None)
data.drop(0, inplace=True, axis=1)
data.drop(0, inplace=True, axis=0)
data.columns = data.iloc[0]
data.drop(1, inplace=True, axis=0)
data.columns = [c.replace(' ', '_') for c in data.columns]
data = data.replace(to_replace=['no', 'yes'], value=[0, 1])
data.CLASS = data.CLASS.replace(to_replace=['Live', 'Die'], value=[0, 1])
data = data.replace(to_replace=['?'], value=np.nan)
data = data.reset_index()
X_temp = data.drop(columns=['CLASS'])
X_temp
```

	index	Age	Sex	Steroid	Antivirals	Fatique	Malaise	Anorexia	Liver_Big	Liver_Firm	Spleen_Palpable	Speiders	Ascites	Varices	Bilirubin	Alk_Phosp
0	2	30	1	0.0	1	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	
1	3	50	0	0.0	1	0.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	0.9	
2	4	78	0	1.0	1	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7	
3	5	31	0	NaN	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7	
4	6	34	0	1.0	1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
150	152	46	0	1.0	1	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	7.6	
151	153	44	0	1.0	1	0.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.9	
152	154	61	0	0.0	1	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	1.0	0.8	
153	155	53	1	0.0	1	0.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	1.5	
154	156	43	0	1.0	1	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	1.2	

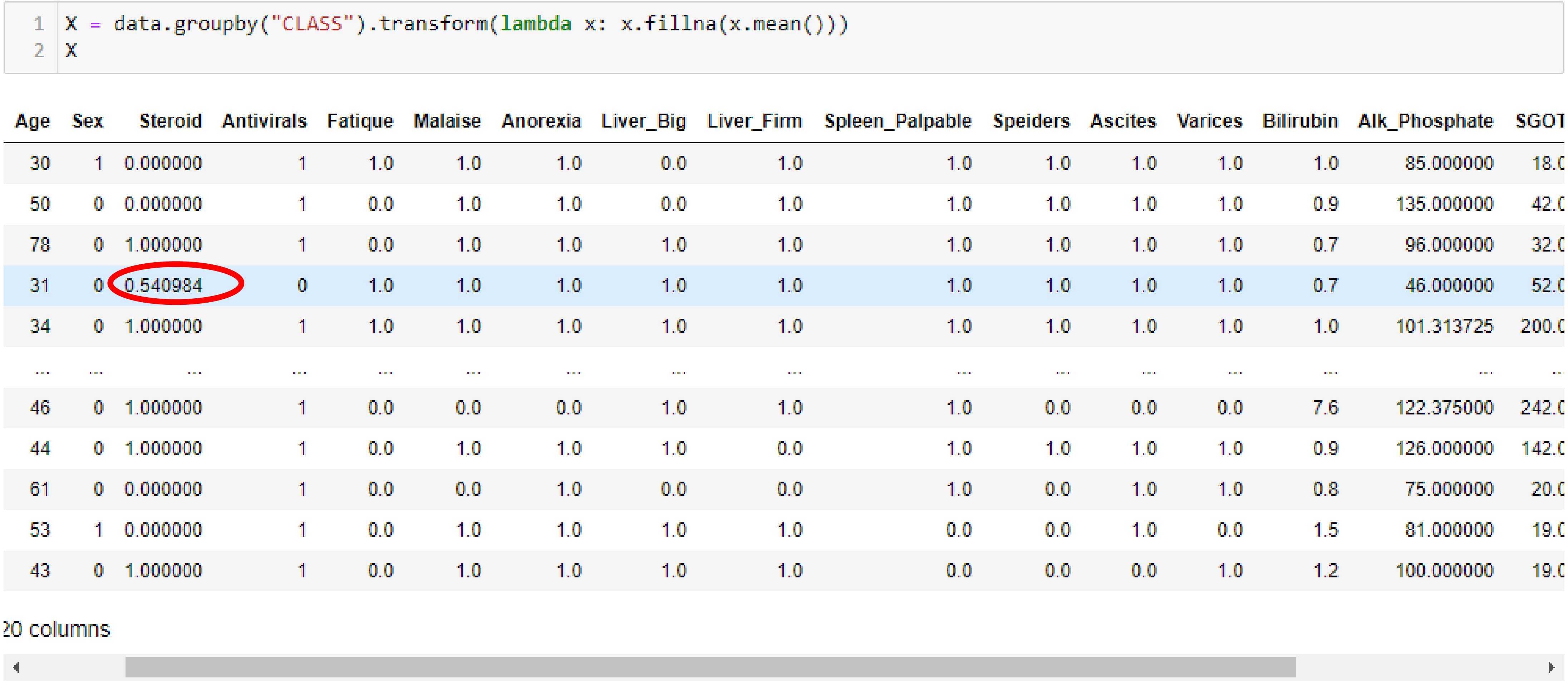
155 rows × 20 columns



```
y = data['CLASS'].values
y
```

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

2. Impute the missing data with the mean values of same attribute in the same class



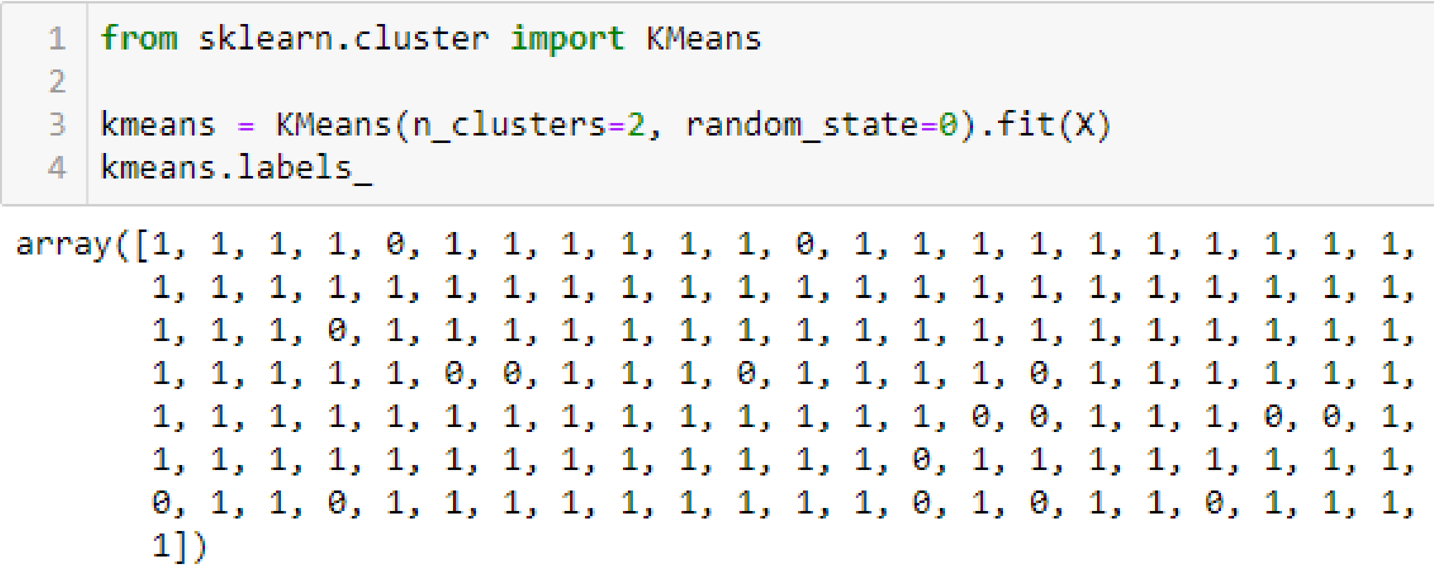
```
X = data.groupby("CLASS").transform(lambda x: x.fillna(x.mean()))
X
```

Age	Sex	Steroid	Antivirals	Fatique	Malaise	Anorexia	Liver_Big	Liver_Firm	Spleen_Palpable	Speiders	Ascites	Varices	Bilirubin	Alk_Phosphate	SGOT
30	1	0.000000	1	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	85.000000	18.0
50	0	0.000000	1	0.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	0.9	135.000000	42.0
78	0	1.000000	1	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7	96.000000	32.0
31	0	0.540984	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7	46.000000	52.0
34	0	1.000000	1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	101.313725	200.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
46	0	1.000000	1	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	7.6	122.375000	242.0
44	0	1.000000	1	0.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.9	126.000000	142.0
61	0	0.000000	1	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	1.0	0.8	75.000000	20.0
53	1	0.000000	1	0.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	1.5	81.000000	19.0
43	0	1.000000	1	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	1.2	100.000000	19.0

20 columns

3. Hide the class label of the supervised data

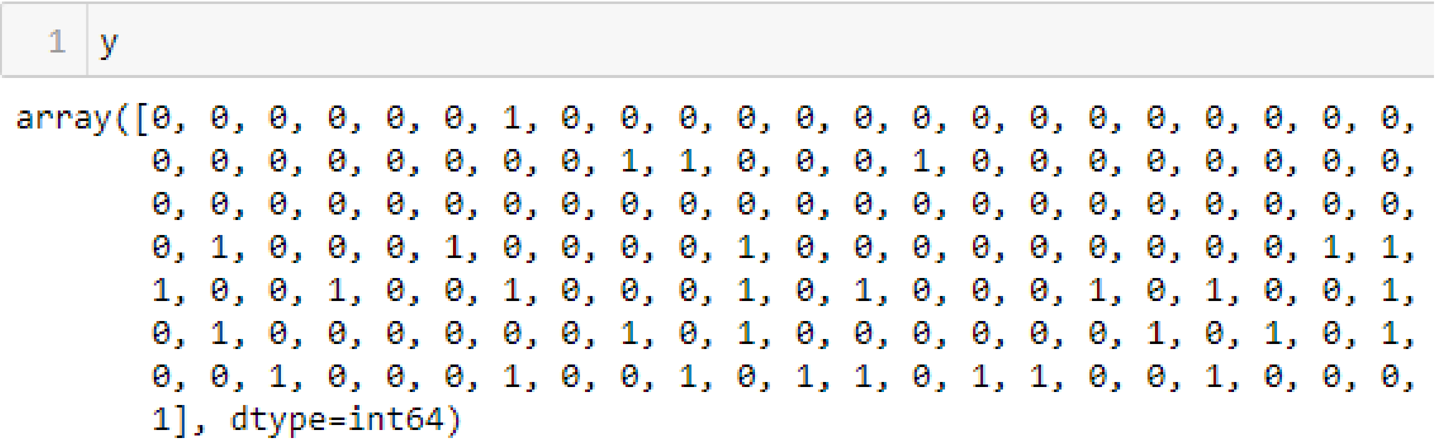
4. Cluster the data using K-Means or Hierarchical Clustering into 2 groups



```python
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
kmeans.labels_
```

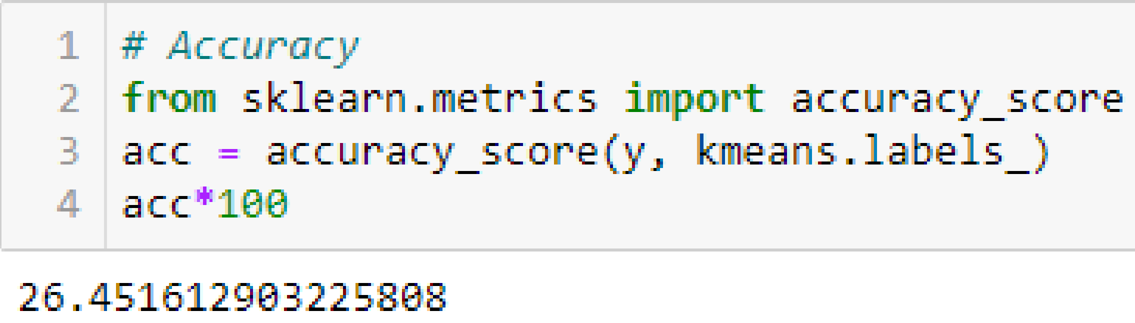
```
array([1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1,
       1])
```



```python
y
```

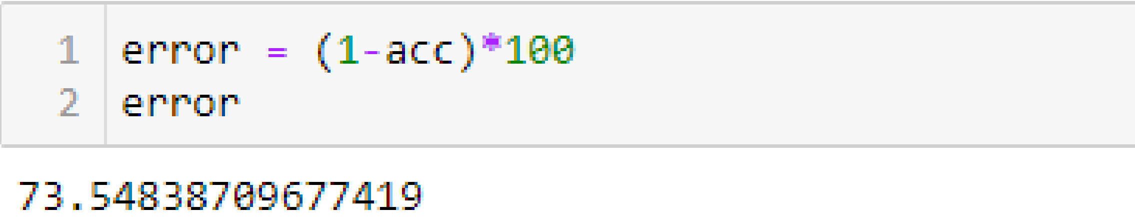
```
array([0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
    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

5. Do cluster analysis



```python
# Accuracy
from sklearn.metrics import accuracy_score
acc = accuracy_score(y, kmeans.labels_)
acc*100
```

```
26.451612903225808
```



```python
error = (1-acc)*100
error
```

```
73.54838709677419
```

Analysis

Error yang dihasilkan sangat tinggi, yaitu 73.54 %. Menurut saya, ini dikarnakan saat proses clustering, algoritma kmeans tidak dapat mendefinisikan cluster mana yang “Live” atau “Die”. Jika saat mendefinisikannya tepat, maka error akan kecil, begitu juga sebaliknya, jika saat mendefinisikan terbalik, maka error akan besar.