Forward School

Program Code: J620-002-4:2020

Program Name: FRONT-END SOFTWARE DEVELOPMENT

Title : Exe26 - Clusters of Grain

Name: Chong Mun Chen

IC Number: 960327-07-5097

Date : 31/7/2023

Introduction : Practising Kmeans clustering on a dataset of the measurements of samples of grain.

Conclusion : Succeeded in using the best number of clusters to plot the scatter graph to visualize the different clusters of grain varieties.

How many clusters of grain?

This exercise is taken and modified from https://github.com/benjaminwilson/python-clustering-exercises (https://github.com/benjaminwilson/python-clustering-exercises)

This is a class to choose a good number of clusters for a dataset using the k-means inertia graph. You are given a dataset of the measurements of samples of grain. What's a good number of clusters in this case?

This dataset was obtained from the UCI (https://archive.ics.uci.edu/ml/datasets/seeds).

Step 1: Load the dataset *(written for you)*.

In [2]:
```
import pandas as pd

seeds_df = pd.read_csv('../data_samples2/seeds.csv')
# forget about the grain variety for the moment - we'll use this later
del seeds_df['grain_variety']
```

Step 2: Display the DataFrame to inspect the data. Notice that there are 7 columns - so each grain sample (row) is a point in 7D space! Scatter plots can't help us here.

In [3]:
```
seeds_df
```

Out[3]:

	area	perimeter	compactness	length	width	asymmetry_coefficient	groove_length
0	15.26	14.84	0.8710	5.763	3.312	2.221	5.220
1	14.88	14.57	0.8811	5.554	3.333	1.018	4.956
2	14.29	14.09	0.9050	5.291	3.337	2.699	4.825
3	13.84	13.94	0.8955	5.324	3.379	2.259	4.805
4	16.14	14.99	0.9034	5.658	3.562	1.355	5.175
...	...	...	...	...	...	...	...
205	12.19	13.20	0.8783	5.137	2.981	3.631	4.870
206	11.23	12.88	0.8511	5.140	2.795	4.325	5.003
207	13.20	13.66	0.8883	5.236	3.232	8.315	5.056
208	11.84	13.21	0.8521	5.175	2.836	3.598	5.044
209	12.30	13.34	0.8684	5.243	2.974	5.637	5.063

210 rows × 7 columns

Step 3: Extract the measurements from the DataFrame using its `.values` attribute:

In [4]:
```
samples = seeds_df.values
samples
```

Out[4]:
```
array([[15.26  , 14.84  ,  0.871 , ...,  3.312 ,  2.221 ,  5.22  ],
       [14.88  , 14.57  ,  0.8811, ...,  3.333 ,  1.018 ,  4.956 ],
       [14.29  , 14.09  ,  0.905 , ...,  3.337 ,  2.699 ,  4.825 ],
       ...,
       [13.2   , 13.66  ,  0.8883, ...,  3.232 ,  8.315 ,  5.056 ],
       [11.84  , 13.21  ,  0.8521, ...,  2.836 ,  3.598 ,  5.044 ],
       [12.3   , 13.34  ,  0.8684, ...,  2.974 ,  5.637 ,  5.063 ]])
```

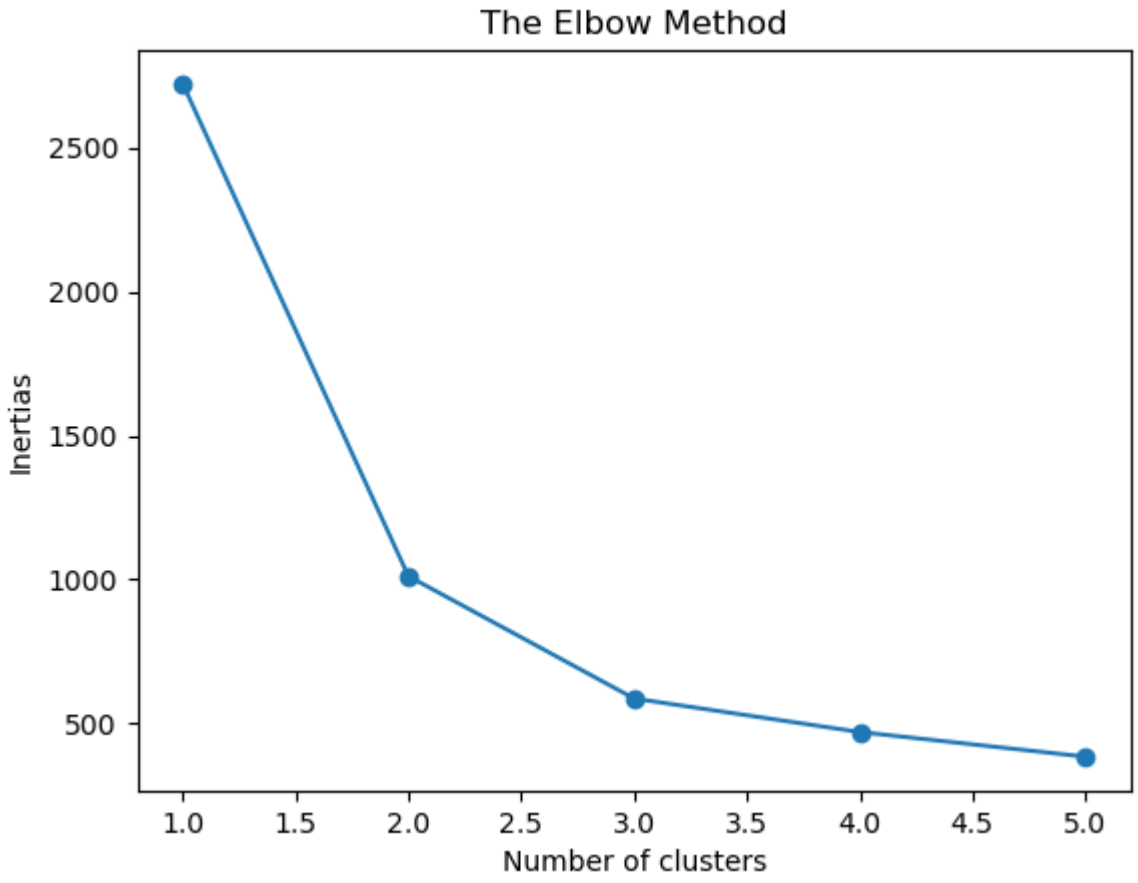
Step 4: *(Written for you)*. Measure the quality of clusterings with different numbers of clusters using the inertia. For each of the given values of `k`, perform the following steps:

- Create a `KMeans` instance called `model` with `k` clusters.
- Fit the model to the grain data `samples`.
- Append the value of the `inertia_` attribute of `model` to the list `inertias`.

In [5]:
```
from sklearn.cluster import KMeans
import warnings

warnings.filterwarnings('ignore')

inertias = []
for k in range(1, 6):
    model = KMeans(n_clusters = k, init = 'k-means++', max_iter = 300, n_init =
    model.fit(samples)
    inertias.append(model.inertia_)
```

Step 5: Plot the inertia to see which number of clusters is best. Remember: lower numbers are better!

In [6]:
```
import matplotlib.pyplot as plt

plt.plot(range(1, 6), inertias, marker='o')
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertias')
plt.show()
```



Excellent work! You can see from the graph the "best" number of clusters. Use this value for the next steps.

Step 6: Create a `KMeans` model called `model` with the best value from the above steps.

In [7]:
```
model = KMeans(n_clusters = 3)
```

Step 7: Use the .fit_predict() method of model to fit it to samples and derive the cluster labels.

Calling .fit_predict() is the same as calling .fit() and then calling .predict().

In [8]:
```
labels = model.fit_predict(samples)
labels
```

Out[8]:
```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2,
       2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 1, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 2, 2,
       2, 2, 2, 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, 2, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 0, 0, 0, 0, 0,
       2, 2, 2, 2, 0, 2, 2, 2, 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, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1])
```

Step 8: Create a DataFrame `df` with two columns named `'labels'` and `'varieties'`, using `labels` and `varieties`, respectively, for the column values.

In [9]:
```
new_df = pd.read_csv('../data_samples2/seeds.csv')

varieties = new_df['grain_variety']
```

In [10]:
```
df = pd.DataFrame({'labels': labels, 'varieties': varieties})
df
```

Out[10]:

	labels	varieties
0	2	Kama wheat
1	2	Kama wheat
2	2	Kama wheat
3	2	Kama wheat
4	2	Kama wheat
...	...	...
205	1	Canadian wheat
206	1	Canadian wheat
207	1	Canadian wheat
208	1	Canadian wheat
209	1	Canadian wheat

210 rows × 2 columns

Step 9: Use the `pd.crosstab()` function on `df['labels']` and `df['varieties']` to count the number of times each grain variety coincides with each cluster label. Assign the result to `ct` .

In [11]:
```
ct = pd.crosstab(df['labels'], df['varieties'])
```

Step 10: Display `ct` by evaluating it - and inspect your cross-tabulation! You'll see that your clustering is pretty good.

In [12]:
```
ct
```

Out[12]:

varieties labels	Canadian wheat	Kama wheat	Rosa wheat
0	0	1	60
1	68	9	0
2	2	60	10

Now you are done. If you wish, you can also try to plot the clusters to visualize it.

In [18]:
```
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE

tsne = TSNE(learning_rate = 200)

tsne_features = tsne.fit_transform(samples)

plt.scatter(tsne_features[:, 0], tsne_features[:, 1], c=labels, s=20, cmap='vi
```

Out[18]:
```
<matplotlib.collections.PathCollection at 0x2197f8954b0>
```