IRIS Clustering with K-means & Hierarchical

Python notebook using data from Iris Species · 2,836 views · 2y ago

In [1]:

```
'''
We are analyzing IRIS dataset with k-means and hierachical clustering methods

'''
```

Out[1]:

```
'\nWe are analyzing IRIS dataset with k-means and hierachical clustering methods\n\n'
```

In [2]:

```
from subprocess import check_output
print(check_output(["ls", "../input"]).decode("utf8"))
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="white", color_codes=True)

%matplotlib inline
from pandas import Series, DataFrame
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.cluster import KMeans
from pylab import rcParams
rcParams['figure.figsize'] = 9, 8  # set plot size
```

```
Iris.csv
database.sqlite
```

In [3]:

```
iris = pd.read_csv("../input/Iris.csv")
iris.head()
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	I
0	1	5.1	3.5	
1	2	4.9	3.0	
2	3	4.7	3.2	
3	4	4.6	3.1	
4	5	5.0	3.6	

In [4]:

```
```

In [4]:

```
iris_SP = iris[['SepalLengthCm','SepalWi
dthCm','PetalLengthCm','PetalWidthCm']]
iris_SP.head()
```

Out[4]:

	SepalLengthCm	SepalWidthCm	PetalL
0	5.1	3.5	1.4
1	4.9	3.0	1.4
2	4.7	3.2	1.3
3	4.6	3.1	1.5
4	5.0	3.6	1.4

In [5]:

```
iris_SP.describe()
```

Out[5]:

	SepalLengthCm	SepalWidthCm	P
count	150.000000	150.000000	1
mean	5.843333	3.054000	3
std	0.828066	0.433594	1
min	4.300000	2.000000	1

25%	5.100000	2.800000	1
50%	5.800000	3.000000	4
75%	6.400000	3.300000	5
max	7.900000	4.400000	6

In [6]:

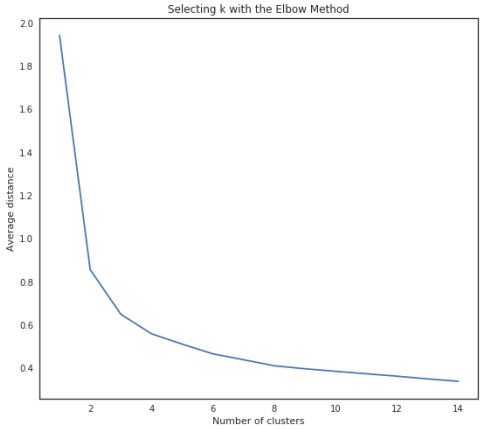
```
# k-means cluster analysis for 1-15 clusters
from scipy.spatial.distance import cdist
clusters=range(1,15)
meandist=[]

# loop through each cluster and fit the model to the train set
# generate the predicted cluster assingment and append the mean
# distance my taking the sum divided by the shape
for k in clusters:
    model=KMeans(n_clusters=k)
    model.fit(iris_SP)
    clusassign=model.predict(iris_SP)
    meandist.append(sum(np.min(cdist(iris_SP, model.cluster_centers_, 'euclidean'), axis=1))
    / iris_SP.shape[0])

"""
Plot average distance from observations from the cluster centroid
to use the Elbow Method to identify number of clusters to choose
"""
plt.plot(clusters, meandist)
plt.xlabel('Number of clusters')
plt.ylabel('Average distance')
plt.title('Selecting k with the Elbow Method')
# pick the fewest number of clusters that reduces the average distance
# If you observe after 3 we can see graph is almost linear
```

Out[6]:

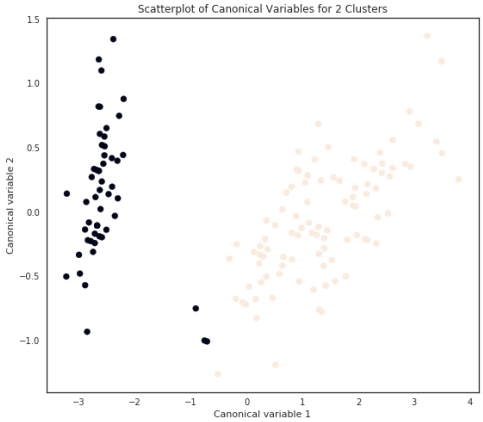
```
<matplotlib.text.Text at 0x7fd09a0c8a90>
```



In [7]:

```
# Here we are just analyzing if we consid
er 2 cluster instead of 3 by using PCA
model3=KMeans(n_clusters=2)
model3.fit(iris_SP) # has cluster assing
ments based on using 2 clusters
clusassign=model3.predict(iris_SP)
# plot clusters
''' Canonical Discriminant Analysis for v
ariable reduction:
1. creates a smaller number of variables
2. linear combination of clustering varia
bles
3. Canonical variables are ordered by pro
portion of variance accounted for
4. most of the variance will be accounted
for in the first few canonical variables
'''
from sklearn.decomposition import PCA #
 CA from PCA function
pca_2 = PCA(2) # return 2 first canonica
l variables
plot_columns = pca_2.fit_transform(iris_
SP) # fit CA to the train dataset
plt.scatter(x=plot_columns[:,0], y=plot_
columns[:,1], c=model3.labels_,)
# plot 1st canonical variable on x axis,
 2nd on y-axis
plt.xlabel('Canonical variable 1')
plt.ylabel('Canonical variable 2')
plt.title('Scatterplot of Canonical Vari
ables for 2 Clusters')
plt.show()
# close or overlapping clusters idicate c
orrelated variables with low in-class var
```

```
iance
# but not good separation. 2 cluster migh
t be better.
```



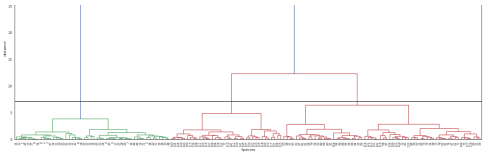
In [8]:

```
# calculate full dendrogram
from scipy.cluster.hierarchy import dend
rogram, linkage

# generate the linkage matrix
Z = linkage(iris_SP, 'ward')

# set cut-off to 150
max_d = 7.08                # max_d as i
n max_distance

plt.figure(figsize=(25, 10))
plt.title('Iris Hierarchical Clustering
 Dendrogram')
plt.xlabel('Species')
plt.ylabel('distance')
dendrogram(
    Z,
    truncate_mode='lastp',  # show only
 the last p merged clusters
    p=150,                  # Try changi
ng values of p
    leaf_rotation=90.,      # rotates th
e x axis labels
    leaf_font_size=8.,      # font size
 for the x axis labels
)
plt.axhline(y=max_d, c='k')
plt.show()
```



In [9]:

```
# calculate full dendrogram for 50
from scipy.cluster.hierarchy import dend
rogram, linkage

# generate the linkage matrix
Z = linkage(iris_SP, 'ward')

# set cut-off to 50
max_d = 7.08                  # max_d as i
n max_distance

plt.figure(figsize=(25, 10))
plt.title('Iris Hierarchical Clustering
 Dendrogram')
plt.xlabel('Species')
plt.ylabel('distance')
dendrogram(
    Z,
```


Did you find this Kernel useful?
Show your appreciation with an upvote

1

Data

Data Sources

- Iris Sp...
  - 150 x 6
- data...
  - 150 x

Iris Species

Classify iris plants into three species in this classic dataset

Last Updated: 3 years ago (Version 2)

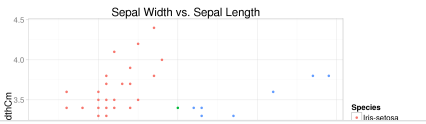
About this Dataset

The Iris dataset was used in R.A. Fisher's classic 1936 paper, The Use of Multiple Measurements in Taxonomic Problems, and can also be found on the UCI Machine Learning Repository.

It includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

The columns in this dataset are:

- Id
- SepalLengthCm
- SepalWidthCm
- PetalLengthCm
- PetalWidthCm
- Species



Run Info

Succeeded	True	Time	500.7 seconds
Exit Code	0	Queue Time	0 seconds
Docker Image Name	kaggle/python (Dockerfile)	Output Size	0
Timeout Exceeded	False	Used All Space	False
Failure Message			

Log

Download Log

Time	Line #	Log Message
	1	[{
	2	"data": "[NbConvertApp] Converting notebook __temp_notebook_source__.ipynb to html\n",
	3	"stream_name": "stderr",
	4	"time": 1.8479714569984935
	5	},{
	6	"data": "[NbConvertApp] Support files will be in __results___files/\n[NbConvertApp] Making directory __results___files\n",
	7	"stream_name": "stderr",
	8	"time": 1.999072188977152

```
},{
  "data": "[NbConvertApp] Making directory __results___files\n[NbConvertApp] Making directory __results___files\n[NbConvertApp] Making directory __results___files\n[NbConvertApp] Writing 274700 bytes to __results__.html\n",
  "stream_name": "stderr",
  "time": 2.004945817985572
}{
  "data": "[NbConvertApp] Converting notebook __temp_notebook_source__.ipynb to notebook\n",
  "stream_name": "stderr",
  "time": 1.9487506449804641
},{
  "data": "[NbConvertApp] Executing notebook with kernel: python3\n",
  "stream_name": "stderr",
  "time": 1.956019080011174
},{
  "data": "Fontconfig warning: ignoring C.UTF-8: not a valid language tag\n",
  "stream_name": "stderr",
  "time": 3.605599074973725
},{
  "data": "[NbConvertApp] Writing 135758 bytes to __notebook__.ipynb\n",
  "stream_name": "stderr",
  "time": 9.70323542895494
}{
  "data": "[NbConvertApp] Converting notebook __notebook__.ipynb to html\n",
  "stream_name": "stderr",
  "time": 1.8791770079988055
},{
  "data": "[NbConvertApp] Support files will be in __results___files/\n[NbConvertApp] Making directory __results___files\n[NbConvertApp] Making directory __results___files\n",
  "stream_name": "stderr",
  "time": 2.0350738629931584
},{
  "data": "[NbConvertApp] Making directory __results___files\n[NbConvertApp] Making directory __results___files\n[NbConvertApp] Writing 275002 bytes to __results__.html\n",
  "stream_name": "stderr",
  "time": 2.03949808498146
}

Complete. Exited with code 0.
```

Comments (0)

Click here to comment...