The Engineering World #DataScience 24 & 25

May 31, 2018

AKKAL BAHADUR BIST
DATA SCIENTIST AT
KATHMANDU INSTITUTE OF APPLIED SCIENCES (KIAS)
Center for Conservation Biology (CCB)

1 K-MEANS METHOD FOR CLUSTERING

In [5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pylab import rcParams

import sklearn
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import scale

import sklearn.metrics as sm
from sklearn.metrics import confusion_matrix, classification_report
```

In [6]:
```
%matplotlib inline
rcParams['figure.figsize'] = 7, 4
```

In [7]:
```
iris = datasets.load_iris()
X = scale(iris.data)
Y = pd.DataFrame(iris.target)
variable_names = iris.feature_names
X[0:10,]
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-7-c0b168f8d8bb> in <module>()
----> 1 iris = datasets.load_iris()
      2 X = scale(iris.data)
```

```
      3 Y = pd.DataFrame(iris.target)
      4 variable_names = iris.feature_names
      5 X[0:10,]


    NameError: name 'datasets' is not defined
```

1.0.1 Building and Running your model

```
In [ ]: clustering = KMeans(n_clusters = 3, random_state = 5)
        clustering.fit(X)
```

1.0.2 Plotting your model output

```
In [ ]: iris_df = pd.DataFrame(iris.data)
        iris_df.columns = ['Sepal_Length', 'Sepal_Width', 'Petal_Length', 'Petal_Width']
        Y.columns = ['Targets']
```

```
In [ ]: color_theme = np.array(['darkgray', 'lightsalmon', 'powderblue'])

        plt.subplot(1,2,1)
        plt.scatter(x = iris_df.Petal_Length, y = iris_df.Petal_Width, c = color_theme[iris.targ
        plt.title('Ground Truth Classification')

        plt.subplot(1,2,2)
        plt.scatter(x = iris_df.Petal_Length, y = iris_df.Petal_Width, c = color_theme[clusterin
        plt.title('K-Means Classification')
```

```
In [ ]: relabel = np.choose(clustering.labels_, [2,0,1]).astype(np.int64)
        plt.subplot(1,2,1)
        plt.scatter(x = iris_df.Petal_Length, y = iris_df.Petal_Width, c = color_theme[iris.targ
        plt.title('Ground Truth Classification')

        plt.subplot(1,2,2)
        plt.scatter(x = iris_df.Petal_Length, y = iris_df.Petal_Width, c = color_theme[clusterin
        plt.title('K-Means Classification')
```

1.0.3 Evaluate your clustering result

```
In [ ]: print (classification_report(Y,relabel))
```

2 HIERARCHICAL CLUSTERING

```
In [ ]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from pylab import rcParams
```

```
import scipy
from scipy.cluster.hierarchy import dendrogram, linkage
from scipy.cluster.hierarchy import fcluster
from scipy.cluster.hierarchy import cophenet
from scipy.spatial.distance import pdist

import seaborn as sb

import sklearn
from sklearn.cluster import AgglomerativeClustering
import sklearn.metrics as sm
```

In [ ]:
```
np.set_printoptions(precision=4, suppress=True)
plt.Figure(figsize=(10,3))
%matplotlib inline
plt.style.use('seaborn-whitegrid')
```

In [ ]:
```
address = 'mtcars.csv'
cars = pd.read_csv(address)
cars.columns = ['car_names', 'mpg', 'cyl', 'disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am
X = cars.ix[:,(1,3,4,6)].values
Y = cars.ix[:,(9)].values
```

2.0.1 Using scipy to generate dendrogram

In [ ]:
```
Z = linkage(X, 'ward')
```

In [ ]:
```
dendrogram(Z, truncate_mode='lastp', p = 12, leaf_rotation=45, leaf_font_size=15, show_c
plt.title('Truncate Hierarchical Clustering Dendrogram')
plt.xlabel("cluster Size")
plt.ylabel('Distance')
plt.axhline(y = 500)
plt.axhline(y = 150)
plt.show()
```

2.0.2 Generate Hierarchical Clusters

In [ ]:
```
k = 2
Hclustering = AgglomerativeClustering(n_clusters=k, affinity='euclidean', linkage= 'ward
Hclustering.fit(X)
sm.accuracy_score(Y, Hclustering.labels_)
```

In [ ]:
```
Hclustering = AgglomerativeClustering(n_clusters=k, affinity='euclidean', linkage= 'comp
Hclustering.fit(X)
sm.accuracy_score(Y, Hclustering.labels_)
```

In [ ]:
```
Hclustering = AgglomerativeClustering(n_clusters=k, affinity='euclidean', linkage= 'aver
Hclustering.fit(X)
sm.accuracy_score(Y, Hclustering.labels_)
```

In [ ]:
```
Hclustering = AgglomerativeClustering(n_clusters=k, affinity='manhattan', linkage= 'aver
Hclustering.fit(X)
sm.accuracy_score(Y, Hclustering.labels_)
```