DSC-530: Predictive Modeling

Week 5 Assignment: Predictive Modeling Using Clustering

Author: Jonathan Ibifubara Pollyn

College of Science, Engineering and Technology, Grand Canyon University

In [1]:
```
#Importing the required packages
import pandas as pd
from scipy import stats
from sklearn.cluster import KMeans
```

In [2]:
```
#Importing the data
fram_train = pd.read_csv('C:/School/DSC-530/DataSets/Framingham_Training')
fram_test = pd.read_csv('C:/School/DSC-530/DataSets/Framingham_Test')
```

In [3]:
```
#Viewing the imported data
fram_train
```

Out[3]:

	Index	Sex	Age	Educ	Death
0	1	1	39	4	0
1	2	1	52	4	0
2	3	2	46	2	0
3	5	2	58	2	0
4	7	1	54	1	0
...	...	...	...	...	...
7948	11327	1	40	3	0
7949	11329	1	52	3	0
7950	11330	2	39	3	0
7951	11331	2	46	3	0
7952	11332	2	50	3	0

7953 rows × 5 columns

1. Run k-means clustering on the Framingham_training data set, requesting k = 2 clusters

In [4]:
```
#Isolating the Age variable
X = fram_train[['Age']]
```

In [5]:
```
#Standardizing the variable
Xz = pd.DataFrame(stats.zscore(X), columns=['Age'])
```

In [9]:
```
#Running the KMeans clustering
kmeans01 = KMeans(n_clusters=2).fit(Xz)
```

1. Construct a table of statistics summarizing your clusters

In [10]:
```
#Investigating the cluster
cluster = kmeans01.labels_
```

In [11]:
```
#Add a label to the for the predicted cluster
Xz['Predicted_Cluster'] = cluster
```

In [12]:
```
Xz
```

Out[12]:

	Age	Predicted_Cluster
0	-1.654186	0
1	-0.283752	0
2	-0.916260	0
3	0.348755	1
4	-0.072916	0
...	...	...
7948	-1.548768	0
7949	-0.283752	0
7950	-1.654186	0
7951	-0.916260	0
7952	-0.494588	0

7953 rows × 2 columns

In [30]:
```
#Separate the clusters
cluster01 = Xz.loc[cluster == 0]
cluster02 = Xz.loc[cluster == 1]
```

In [31]:
```
cluster01.describe()
```

Out[31]:

	Age
count	4336.000000
mean	-0.756066
std	0.552180
min	-2.392111
25%	-1.127096
50%	-0.705424
75%	-0.283752
max	0.032502

In [33]:
```
cluster02.describe()
```

Out[33]:

	Age
count	3617.000000
mean	0.906360
std	0.571567
min	0.137919
25%	0.454173
50%	0.770427
75%	1.297517
max	2.773368

In [53]:
```
#Frequency table
pd.value_counts(cluster)
```

Out[53]:
```
0    4336
1    3617
dtype: int64
```

1. Perform k-means clustering on the Framingham_test data set

In [34]:
```
#Viewing the imported test data
fram_test
```

Out[34]:

	Index	Sex	Age	Educ	Death
0	4	2	52	2	0
1	6	1	48	1	0
2	10	2	46	3	0
3	14	2	49	2	0
4	16	2	63	1	0
...	...	...	...	...	...
2252	11307	2	56	1	0
2253	11313	1	68	1	1
2254	11323	2	50	1	1
2255	11326	2	58	2	0
2256	11328	1	46	3	0

2257 rows × 5 columns

In [36]:
```
#Isolating the Age variable
X_test = fram_test[['Age']]
```

In [37]:
```
#Standardizing the variable
Xz_test = pd.DataFrame(stats.zscore(X_test), columns=['Age'])
```

In [42]:
```
#Running the KMeans clustering
kmeans01_test = KMeans(n_clusters=2).fit(Xz_test)
```

In [47]:
```
#Investigating the cluster
cluster_test = kmeans01_test.labels_
```

In [48]:
```
#Separate the clusters
cluster01_test = Xz_test.loc[cluster_test == 0]
cluster02_test = Xz_test.loc[cluster_test == 1]
```

In [49]:
```
cluster01_test.describe()
```

Out[49]:

	Age
count	1245.000000
mean	-0.751650
std	0.537307
min	-2.232349
25%	-1.098477
50%	-0.686160
75%	-0.273843
max	0.035395

In [51]:
```
cluster02_test.describe()
```

Out[51]:

	Age
count	1012.000000
mean	0.924708
std	0.570556
min	0.138474
25%	0.447712
50%	0.860029
75%	1.272346
max	2.715456

In [54]:
```
pd.value_counts(cluster_test)
```

Out[54]:
```
0    1245
1    1012
dtype: int64
```

5 Again run k-means clustering on the Framingham_training data set - k=3

In [55]:
```
#Running the KMeans clustering
kmeans02 = KMeans(n_clusters=3).fit(Xz)
```

1. Construct a table of statistics summarizing your clusters.

In [59]:
```
#Investigating the cluster
cluster_02 = kmeans02.labels_
```

In [63]:
```
#Separate the clusters
cluster01 = Xz.loc[cluster_02 == 0]
cluster02 = Xz.loc[cluster_02 == 1]
cluster03 = Xz.loc[cluster_02 == 2]
```

In [61]:
```
cluster01.describe()
```

Out[61]:

	Age
count	2262.000000
mean	1.240520
std	0.459489
min	0.665009
25%	0.875845
50%	1.086681
75%	1.508353
max	2.773368

In [62]:
```
cluster02.describe()
```

Out[62]:

	Age
count	2521.000000
mean	-1.133912
std	0.401279
min	-2.392111
25%	-1.443350
50%	-1.021678
75%	-0.810842
max	-0.600006

In [64]:
```
cluster03.describe()
```

Out[64]:

	Age
count	3170.000000
mean	0.016572
std	0.330724
min	-0.494588
25%	-0.283752
50%	0.032502
75%	0.348755
max	0.559591

In [66]:
```
pd.value_counts(cluster_02)
```

Out[66]:
```
2    3170
1    2521
0    2262
dtype: int64
```

1. Perform k-means clustering on the Framingham_test data set, specifying k = 3 clusters.

In [67]:
```
#Running the KMeans clustering
kmeans02_test = KMeans(n_clusters=3).fit(Xz_test)
```

In [68]:
```
#Investigating the cluster
cluster_test01 = kmeans02_test.labels_
```

In [91]:
```
#Separate the clusters
cluster01_test01 = Xz_test.loc[cluster_test01 == 0]
cluster02_test01 = Xz_test.loc[cluster_test01 == 1]
cluster03_test01 = Xz_test.loc[cluster_test01 == 2]
```

In [92]:
```
pd.value_counts(cluster_test01)
```

Out[92]:
```
0    942
1    660
2    655
dtype: int64
```

In [71]:
```
cluster01_test01.describe()
```

Out[71]:

	Age
count	942.000000
mean	-0.049848
std	0.350032
min	-0.583081
25%	-0.376922
50%	-0.067684
75%	0.241553
max	0.550791

In [72]:
```
cluster02_test01.describe()
```

Out[72]:

	Age
count	660.000000
mean	1.237049
std	0.455188
min	0.653870
25%	0.860029
50%	1.169267
75%	1.478505
max	2.715456

In [75]:
```
cluster03_test01.describe()
```

Out[75]:

	Age
count	660.000000
mean	1.237049
std	0.455188
min	0.653870
25%	0.860029
50%	1.169267
75%	1.478505
max	2.715456

1. Run k-means clustering on the Framingham_training data set. Specify k = 4 clusters.

In [76]:
```
#Running the KMeans clustering
kmeans03 = KMeans(n_clusters=4).fit(Xz)
```

1. Construct a table of statistics summarizing your four clusters. Clearly describe your four clusters.

In [77]:
```
#Investigating the cluster
cluster_03 = kmeans03.labels_
```

In [78]:
```
pd.value_counts(cluster_03)
```

Out[78]:
```
3    2607
2    2284
0    1729
1    1333
dtype: int64
```

In [81]:
```
#Separate the clusters
cluster10 = Xz.loc[cluster_03 == 0]
cluster11 = Xz.loc[cluster_03 == 1]
cluster12 = Xz.loc[cluster_03 == 2]
cluster13 = Xz.loc[cluster_03 == 3]
```

In [80]:
```
cluster10.describe()
```

Out[80]:

	Age
count	1729.000000
mean	-1.330615
std	0.329184
min	-2.392111
25%	-1.548768
50%	-1.232514
75%	-1.021678
max	-0.916260

In [82]:
```
cluster11.describe()
```

Out[82]:

	Age
count	1333.000000
mean	1.533818
std	0.373427
min	1.086681
25%	1.192099
50%	1.508353
75%	1.719189
max	2.773368

In [83]:
```
cluster12.describe()
```

Out[83]:

	Age
count	2284.000000
mean	0.540160
std	0.268401
min	0.137919
25%	0.348755
50%	0.559591
75%	0.770427
max	0.981263

In [84]:
```
cluster13.describe()
```

Out[84]:

	Age
count	2607.000000
mean	-0.375017
std	0.266598
min	-0.810842
25%	-0.600006
50%	-0.389170
75%	-0.178334
max	0.032502

11 Perform k-means clustering on the Framingham_test data set, requesting k = 4 clusters.

In [87]:
```
#Running the KMeans clustering
kmeans04_test = KMeans(n_clusters=4).fit(Xz_test)
```

In [88]:
```
#Investigating the cluster
cluster_test04 = kmeans04_test.labels_
```

In [90]:
```
pd.value_counts(cluster_test04)
```

Out[90]:
```
0    748
3    586
2    572
1    351
dtype: int64
```

In [100…
```
#Separate the clusters
cluster01_test10 = Xz_test.loc[cluster_test04 == 0]
cluster02_test11 = Xz_test.loc[cluster_test04 == 1]
cluster03_test12 = Xz_test.loc[cluster_test04 == 2]
cluster04_test13 = Xz_test.loc[cluster_test04 == 3]
```

In [95]:
```
cluster01_test10.describe()
```

Out[95]:

	Age
count	748.000000
mean	-0.284592
std	0.257784
min	-0.686160
25%	-0.480001
50%	-0.273843
75%	-0.067684
max	0.138474

In [97]:
```
cluster02_test11.describe()
```

Out[97]:

	Age
count	351.000000
mean	1.577179
std	0.353074
min	1.169267
25%	1.272346
50%	1.478505
75%	1.787742
max	2.715456

In [98]:
```
cluster03_test12.describe()
```

Out[98]:

	Age
count	572.000000
mean	-1.245707
std	0.340322
min	-2.232349
25%	-1.510794
50%	-1.201556
75%	-0.995398
max	-0.789239

In [102…
```
cluster04_test13.describe()
```

Out[102…

	Age
count	586.000000
mean	0.634521
std	0.262624
min	0.241553
25%	0.447712
50%	0.653870
75%	0.860029
max	1.066187

In [ ]: