Homework 2

May 1, 2019

1 Homework 2

Data Mining Ross Lewis

Question 1

```
In [1]: %matplotlib inline
        import matplotlib.pyplot as plt
        import matplotlib.image as mpimg
        import numpy as np

        numImages = 16
        fig = plt.figure(figsize=(7,7))
        imgData = np.zeros(shape=(numImages,36963))

        for i in range(1,numImages+1):
            filename = 'pics/Picture'+str(i)+'.jpg'
            img = mpimg.imread(filename)
            ax = fig.add_subplot(4,4,i)
            plt.imshow(img)
            plt.axis('off')
            ax.set_title(str(i))
            imgData[i-1] = np.array(img.flatten()).reshape(1,img.shape[0]*img.shape[1]*img.sha
```

In [2]:
```
import pandas as pd
from sklearn.decomposition import PCA

numComponents = 2

pca = PCA(numComponents)
pca.fit(imgData)
projected = pca.transform(imgData)

projected = pd.DataFrame(projected,columns=['pc1','pc2'],index=range(1,numImages+1))
projected['food'] = ['burger', 'burger','burger','burger','drink','drink','drink','dri
                     'pasta', 'pasta', 'pasta', 'pasta', 'chicken', 'chicken', 'chick
projected
```

```
Out[2]:              pc1          pc2     food
        1    -1576.667016  6641.462674   burger
        2     -493.825788  6396.184274   burger
        3      990.057233  7236.840769   burger
        4     2189.898458  9051.034582   burger
        5    -7843.064019 -1061.583683    drink
        6    -8498.435161 -5438.442879    drink
        7   -11181.787910 -5319.885107    drink
        8    -6851.941725  1124.542067    drink
        9     7635.140046 -5043.628374    pasta
        10    -708.061138  -528.765053    pasta
        11    7236.249736 -5301.766423    pasta
        12    4417.335795 -4659.524969    pasta
        13   11864.507522  1472.188629  chicken
        14      76.468838  1366.295780  chicken
        15   -7505.666465 -1163.706654  chicken
        16   10249.791595 -4771.245634  chicken
```

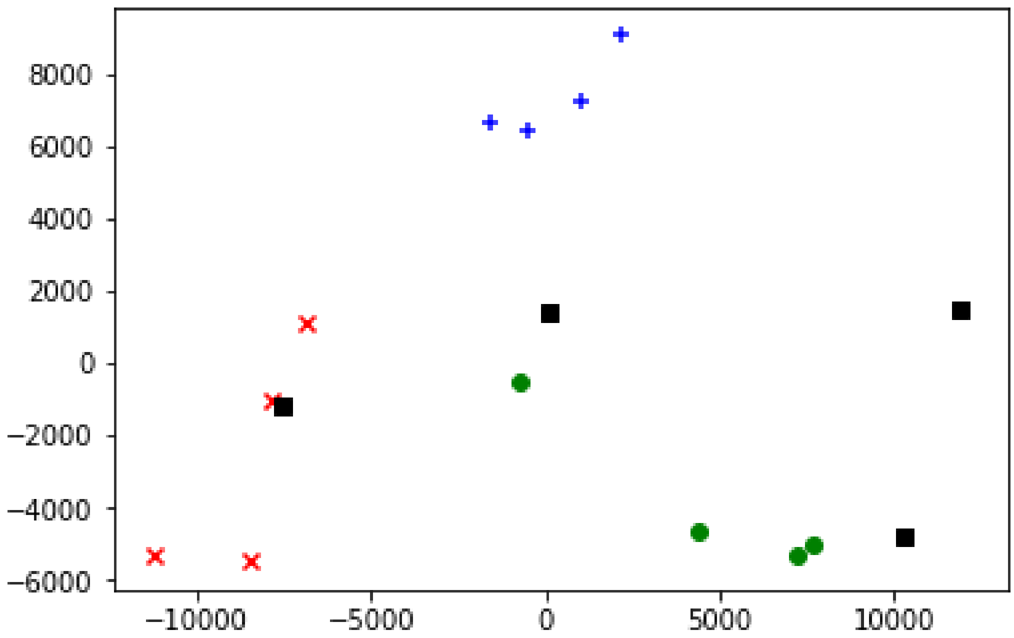
Finally, we draw a scatter plot to display the projected values. Observe that the images of burgers, drinks, and pastas are all projected to the same region. However, the images for fried chicken (shown as black squares in the diagram) are harder to discriminate.

Code:

```
In [23]: import matplotlib.pyplot as plt

         colors = {'burger':'b', 'drink':'r', 'pasta':'g', 'chicken':'k'}
         markerTypes = {'burger':'+', 'drink':'x', 'pasta':'o', 'chicken':'s'}

         for foodType in markerTypes:
             d = projected[projected['food']==foodType]
             plt.scatter(d['pc1'],d['pc2'],marker=markerTypes[foodType],c=colors[foodType])
```



1.0.1 Extra Credit

```
In [107]: import pandas as pd

          prof = pd.read_csv('country_profile_variables.csv')
          #prof.select_dtypes(['number']).head()
          #prof = pd.concat([profDat['Region'],prof.select_dtypes(['number'])],axis=1)
          prof.head()
```

```
Out[107]:           country          Region  Surface area (km2)  \
          0     Afghanistan    SouthernAsia              652864
          1         Albania  SouthernEurope               28748
          2         Algeria  NorthernAfrica             2381741
          3  American Samoa       Polynesia                 199
          4         Andorra  SouthernEurope                 468

             Population in thousands (2017)  Population density (per km2, 2017)  \
          0                           35530                                54.4
          1                            2930                               106.9
          2                           41318                                17.3
          3                              56                               278.2
          4                              77                               163.8

             Sex ratio (m per 100 f, 2017)  \
          0                          106.3
          1                          101.9
```

```
2                          102.0
3                          103.6
4                          102.3

   GDP: Gross domestic product (million current US$)  \
0                                              20270
1                                              11541
2                                             164779
3                                                -99
4                                               2812

  GDP growth rate (annual %, const. 2005 prices)  \
0                                           -2.4
1                                            2.6
2                                            3.8
3                                            -99
4                                            0.8

   GDP per capita (current US$) Economy: Agriculture (% of GVA)  ...  \
0                         623.2                            23.3  ...
1                        3984.2                            22.4  ...
2                        4154.1                            12.2  ...
3                         -99.0                             -99  ...
4                       39896.4                             0.5  ...

   Mobile-cellular subscriptions (per 100 inhabitants).1  \
0                                                8.3
1                                               63.3
2                                               38.2
3                                                -99
4                                               96.9

   Individuals using the Internet (per 100 inhabitants)  \
0                                                 42
1                                                130
2                                                135
3                                                 92
4                                                 13

  Threatened species (number) Forested area (% of land area)  \
0                         2.1                        9.8/0.3
1                        28.2                        5.7/2.0
2                         0.8                      145.4/3.7
3                        87.9                            -99
4                        34.0                        0.5/6.4

  CO2 emission estimates (million tons/tons per capita)  \
0                                                 63
```

```
1                                                    84
2                                                  5900
3                                                   -99
4                                                     1

  Energy production, primary (Petajoules)  \
0                                       5
1                                      36
2                                      55
3                                     -99
4                                     119

  Energy supply per capita (Gigajoules)  \
0                             78.2/47.0
1                             94.9/95.2
2                             84.3/81.8
3                           100.0/100.0
4                           100.0/100.0

   Pop. using improved drinking water (urban/rural, %)  \
0                                          45.1/27.0
1                                          95.5/90.2
2                                          89.8/82.2
3                                          62.5/62.5
4                                        100.0/100.0

   Pop. using improved sanitation facilities (urban/rural, %)  \
0                                              21.43
1                                               2.96
2                                               0.05
3                                                -99
4                                                -99

  Net Official Development Assist. received (% of GNI)
0                                                -99
1                                                -99
2                                                -99
3                                                -99
4                                                -99

[5 rows x 50 columns]
```

In [108]:
```
from sklearn.preprocessing import normalize

numComponents = 2

pca = PCA(numComponents)
#data = normalize(prof.select_dtypes(['number']), axis=0, norm='max')
```

```
pca.fit(normalize(prof.select_dtypes(['number']), axis=0, norm='max'))
projected = pca.transform(normalize(prof.select_dtypes(['number']), axis=0, norm='ma

projected = pd.DataFrame(projected,columns=['pc1','pc2'],index=range(0,len(prof)))
projected['region'] = prof['Region']
projected.head()
```

Out[108]:

	pc1	pc2	region
0	-1.225279	0.101471	SouthernAsia
1	-1.080330	0.048536	SouthernEurope
2	-1.221848	-0.069134	NorthernAfrica
3	5.664929	0.996393	Polynesia
4	-1.125041	0.017737	SouthernEurope

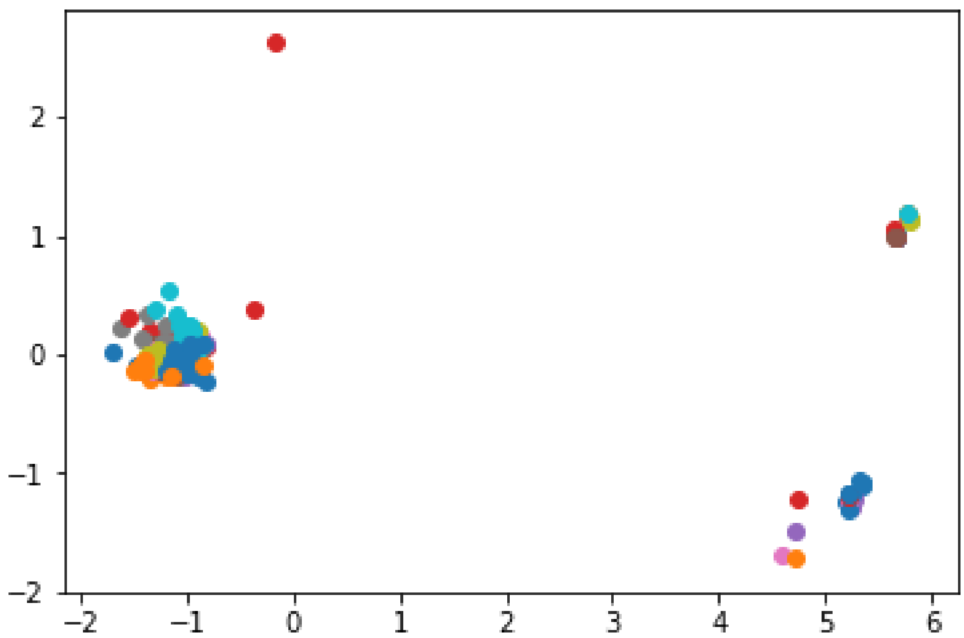
In [115]:

```
print(len(np.unique(projected['region'])))

for reg in np.unique(projected['region']):
    d = projected[projected['region']==reg]

    plt.scatter(d['pc1'],d['pc2'])
```

22



Above is the normalized data projected onto two dimensions with different regions labaled by color.

Question 2

1.1 Decision Tree

In [24]:
```
# Load the dataset

import pandas as pd

data = pd.read_csv('vertebrate.csv',header='infer')
data
```

Out[24]:
```
             Name  Warm-blooded  Gives Birth  Aquatic Creature  \
0           human             1            1                 0
1          python             0            0                 0
2          salmon             0            0                 1
3           whale             1            1                 1
4            frog             0            0                 1
5          komodo             0            0                 0
6             bat             1            1                 0
7          pigeon             1            0                 0
8             cat             1            1                 0
9   leopard shark             0            1                 1
10         turtle             0            0                 1
11        penguin             1            0                 1
12      porcupine             1            1                 0
13            eel             0            0                 1
14     salamander             0            0                 1

    Aerial Creature  Has Legs  Hibernates       Class
0                 0         1           0     mammals
1                 0         0           1    reptiles
2                 0         0           0      fishes
3                 0         0           0     mammals
4                 0         1           1  amphibians
5                 0         1           0    reptiles
6                 1         1           1     mammals
7                 1         1           0       birds
8                 0         1           0     mammals
9                 0         0           0      fishes
10                0         1           0    reptiles
11                0         1           0       birds
12                0         1           1     mammals
13                0         0           0      fishes
14                0         1           1  amphibians
```

In [28]:
```
# Pre-processing data

# convert the classes to be binary: replacing fishes, birds, amphibians and retiles a
```

```
data['Class'] = np.where(data['Class'] == 'mammals','mammals','non-mammals')
data
```

Out[28]:

```
             Name  Warm-blooded  Gives Birth  Aquatic Creature  \
0           human             1            1                 0
1          python             0            0                 0
2          salmon             0            0                 1
3           whale             1            1                 1
4            frog             0            0                 1
5          komodo             0            0                 0
6             bat             1            1                 0
7          pigeon             1            0                 0
8             cat             1            1                 0
9   leopard shark             0            1                 1
10         turtle             0            0                 1
11        penguin             1            0                 1
12      porcupine             1            1                 0
13            eel             0            0                 1
14     salamander             0            0                 1

    Aerial Creature  Has Legs  Hibernates        Class
0                 0         1           0      mammals
1                 0         0           1  non-mammals
2                 0         0           0  non-mammals
3                 0         0           0      mammals
4                 0         1           1  non-mammals
5                 0         1           0  non-mammals
6                 1         1           1      mammals
7                 1         1           0  non-mammals
8                 0         1           0      mammals
9                 0         0           0  non-mammals
10                0         1           0  non-mammals
11                0         1           0  non-mammals
12                0         1           1      mammals
13                0         0           0  non-mammals
14                0         1           1  non-mammals
```

In [29]:

```
# Pandas provides a function cross-tabulation that can examine the relationship betwe
pd.crosstab([data['Warm-blooded'],data['Gives Birth']],data['Class'])
```

Out[29]:

```
Class                     mammals  non-mammals
Warm-blooded Gives Birth
0            0                  0            7
             1                  0            1
1            0                  0            2
             1                  5            0
```

The results above show that it is possible to distinguish mammals from non-mammals using these two attributes alone since each combination of their attribute values would yield only instances that belong to the same class. For example, mammals can be identified as warm-blooded vertebrates that give birth to their young. Such a relationship can also be derived using a decision tree classifier, as shown by the example given in the next subsection.

```
In [58]: # apply a decision tree classifier to the vertebrate dataset described in the previou

         from sklearn import tree

         Y = data['Class']
         X = data.drop(['Name','Class'],axis=1)

         clf = tree.DecisionTreeClassifier(criterion='gini',splitter='random',max_depth=3,min_
         clf = clf.fit(X, Y)
```

The preceding commands will extract the predictor (X) and target class (Y) attributes from the vertebrate dataset and create a decision tree classifier object using entropy as its impurity measure for splitting criterion. The decision tree class in Python sklearn library also supports using 'gini' as impurity measure. The classifier above is also constrained to generate trees with a maximum depth equals to 3. Next, the classifier is trained on the labeled data using the fit() function.

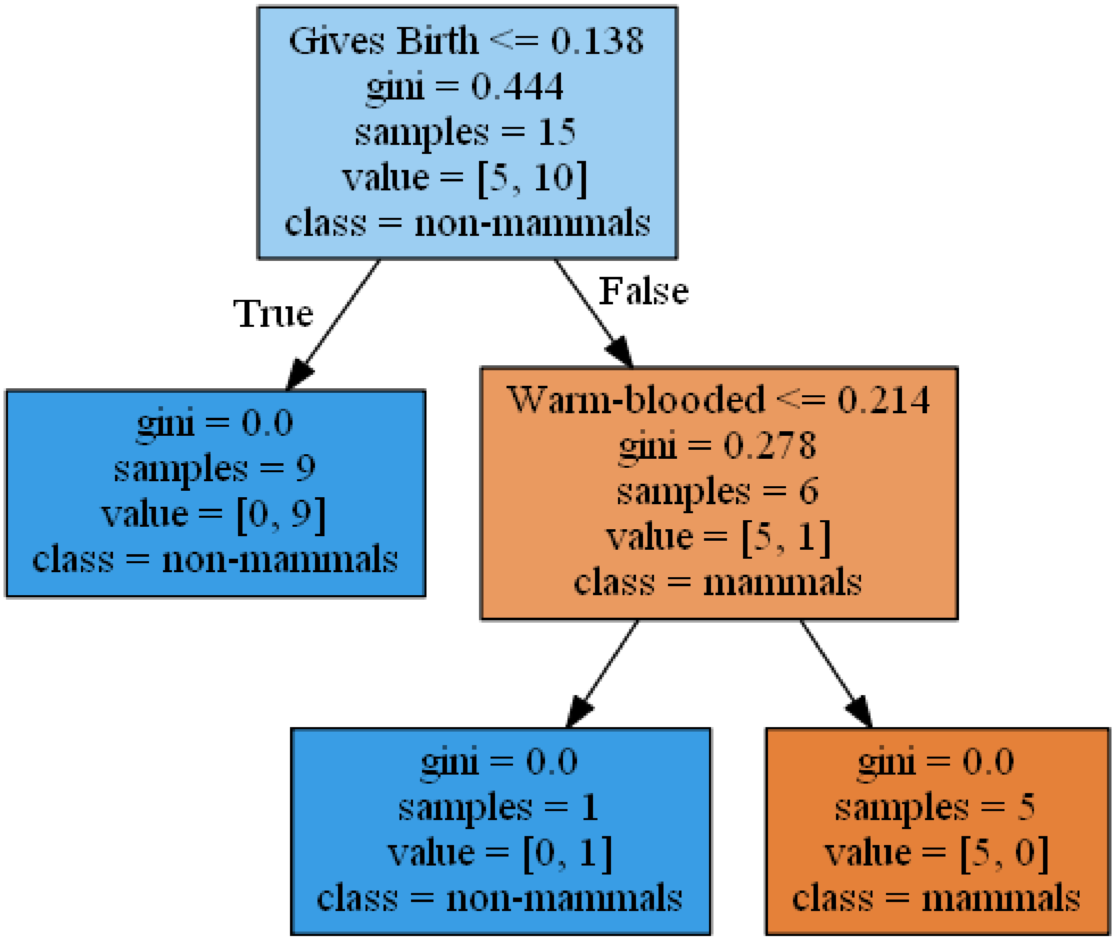
We can plot the resulting decision tree obtained after training the classifier. To do this, you must first install both graphviz (http://www.graphviz.org) and its Python interface called pydotplus (http://pydotplus.readthedocs.io/).

```
In [59]: # visualizing the tree

         import pydotplus
         from IPython.display import Image

         dot_data = tree.export_graphviz(clf, feature_names=X.columns, class_names=['mammals',
                                         out_file=None)
         graph = pydotplus.graph_from_dot_data(dot_data)
         Image(graph.create_png())
```

Out[59]:



Next, suppose we apply the decision tree to classify the following test examples.

```
In [60]: testData = [['gila monster',0,0,0,0,1,1,'non-mammals'],
                    ['platypus',1,0,0,0,1,1,'mammals'],
                    ['owl',1,0,0,1,1,0,'non-mammals'],
                    ['dolphin',1,1,1,0,0,0,'mammals']]
         testData = pd.DataFrame(testData, columns=data.columns)
         testData
```

```
Out[60]:            Name  Warm-blooded  Gives Birth  Aquatic Creature  Aerial Creature  \
         0  gila monster             0            0                 0                0
         1      platypus             1            0                 0                0
         2           owl             1            0                 0                1
         3       dolphin             1            1                 1                0

            Has Legs  Hibernates        Class
         0         1           1  non-mammals
         1         1           1      mammals
         2         1           0  non-mammals
         3         0           0      mammals
```

We first extract the predictor and target class attributes from the test data and then apply the decision tree classifier to predict their classes.

```
In [62]: testY = testData['Class']
         testX = testData.drop(['Name','Class'],axis=1)

         predY = clf.predict(testX)
         predictions = pd.concat([testData['Name'],pd.Series(predY,name='Predicted Class')], a
         predictions
```

```
Out[62]:            Name Predicted Class
         0  gila monster     non-mammals
         1      platypus     non-mammals
         2           owl     non-mammals
         3       dolphin         mammals
```

Except for platypus, which is an egg-laying mammal, the classifier correctly predicts the class label of the test examples. We can calculate the accuracy of the classifier on the test data as shown by the example given below.

```
In [63]: from sklearn.metrics import accuracy_score

         print('Accuracy on test data is %.2f' % (accuracy_score(testY, predY)))

Accuracy on test data is 0.75
```