Econ 426

**Homework Assignment 2**

Due: 11:59pm, 5/10/2019

**Submission:**

* Submit both your writeup and source codes before the due date

**Part I: Principal Component Analysis (PCA)**

PCA is a classical method for projecting data from its high-dimensional space into a lower dimensional space. Each component (or axis) of the space created by PCA has a few properties: (I) being orthogonal to each other, and (II) capturing the maximum amount of variations in the data. PCA is widely used in visualization, denoising, data analysis and others.

This mini-project illustrates how PCA can be used to an image dataset. There are 16 image files (RGB Format), each of which has a size of 111 by 111 pixels. The provided script will read each image file and convert the RGB image into a 111 by 111 by 3 = 36963 feature values. This leads to a data matrix of size 16 by 36963. We will use PCA to reduce the feature dimension of each image to be 2, so we can visualize these images in a two dimension space.



You will need to complete the provided script, main\_ha2\_pca.ipynb, which includes three major steps:

**Step 1**: Load images from files to form the feature matrix. The script uses the function matplotlib.image.imread() to load image files, and use the function .reshape() to form the feature matrix.

You don’t need to change this step.

**Step 2: Run PCA on the feature matrix**

There is a placeholder in this step. You might need to import the PCA function:

from sklearn.decomposition import PCA

and run PCA on a feature matrix

pca = PCA(n\_components)

pca.fit(A)

then, you can project a matrix into the lower-dimensional space.

projected = pca.transform(A)

The number of components in this step should be 2. Please complete the placeholder in the script using the above functions.

**Step 3: Plot the data in the lower-dimensional space.**

This step will draw a scatter plot to display the projected values. There is a placeholder in this step. You might use the function matplotlib.pyplot.scatter() to create the following scatter. Study this reference link: <https://matplotlib.org/api/_as_gen/matplotlib.pyplot.scatter.html>.

Pay attention to the provided figure. You need to properly set up colors and marker types for data points.



Observe that the images of burgers, drinks, and pastas are all projected to the same region. However, the images for fried chicken are harder to discriminate (black squares).

**Extra Credits**

Extra credits will be available for those who apply PCA algorithm to visualize the UNData, stored in the ‘country\_profile\_variables.csv’ file. This dataset includes the indicators of 229 countries present in UNData, e.g., Population in thousands (2017), sex ratio, GDP, GDP growth rate, GDP per capita, Economy: Agriculture (% of GVA), etc. A full description about this dataset can be found in this link: <https://www.kaggle.com/sudalairajkumar/undata-country-profiles#country_profile_variables.csv>

You might consider all the 48 indicators (except ‘country’, ‘region’) or a subset of these indicators.

Run PCA on the 229 by 48 matrix, and keep the top two components. Project each data point on these two components. In your plots, you might use different colors for different regions (i.e. the second column of the csv file).

**Part II: Decision Tree**

In this mini-project, you will apply the decision tree method to classify vertebrates. The file ‘vertebrate.csv’ includes 15 samples from five categories: mammals, reptiles, birds, fishes, and amphibians. Each sample is described by a set of explanatory attributes. Except for "name", the rest of the attributes (e.g., Warm-blood, Gives Birth, Aerial Creature) have been converted into a binary representation (i.e. 0 or 1).

The script main\_ha2\_tree.ipynb includes multiple steps for loading data, preprocessing data, creating the optimal decision tree, and testing the tree on unseen data. Please read the following step-wise instructions, and complete the place-holders in the scripts.

**Step 1: load data**

We employ the function pandas.read\_csv() to load the .csv file as a pandas.DataFrame object. No need to change this step.

Link: <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html>

This step will show the following figure.

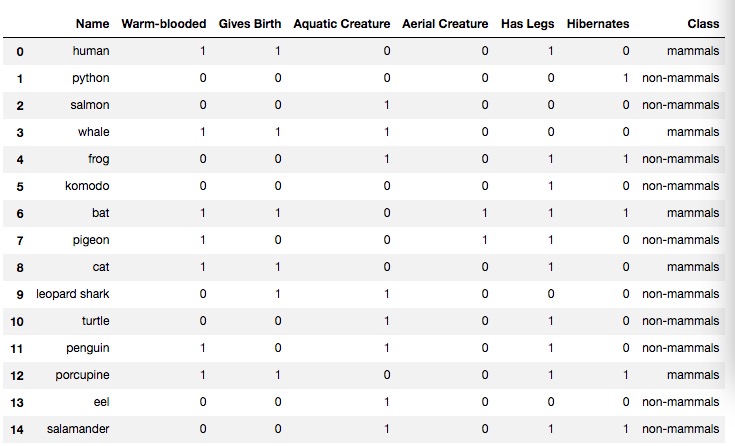


**Step 2: preprocessing data**

Since there are only 15 samples in this small dataset, we convert the classification problem to be a binary classification task, i.e. (mammals versus non-mammals). To do so, we need to replace the class labels of the instances to non-mammals except for those that belong to the mammals class.

Please write your codes to complete this function. You might need to use the function pandas.DataFame.replace(). Study this reference link: <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.replace.html>

Once replaced, the new data table is as follows:



We also use the pandas.crosstab() function to examine the relationships between the two attributes: warm-blooded and Gives Birth, with respect to the class.

**Step 3: create a decision tree**

We will employ the sklearn.tree. DecisionTreeClassifier (), .fit() to create the best decision tree. The function DecisionTreeClassifier() includes the following options: criterion (default=’gini’), splitter (default=’best’), min\_samples\_split (default=2), min\_samples\_leaf (default=1), max\_leaf\_nodes (default=None).

Please study this reference link: <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>

This step includes a placeholder, but you only need to change the option values. Please select and evaluate at least three options from the above list. For each selected option, e.g., criterion, revise its value (e.g., from ‘gini’ to ‘entropy’) and observe how this change (with other options fixed) affects the testing results (see step 6). Write down your observations in your writeup.

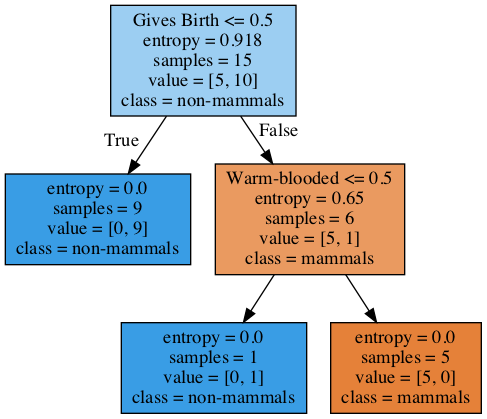
Step 4: visualize the tree

We can plot the resulting decision tree using the GraphVIZ tool. You will need to install both graphviz ([http://www.graphviz.org](http://www.graphviz.org/)) and its Python interface called pydotplus (<http://pydotplus.readthedocs.io/)>. For example,

conda install pydotplus

condata install graphviz

The following figure shows the plotted tree.



**Step 5: Test**

We test the trained decision tree using a few unseen samples, e.g., platypus, owl, dolphin etc. No need to change this step, but please pay attention to how to create a new DataFrame object and use it to call the .predict() function.

**Step 6: Calculate accuracy**

Calculate the accuracy: number of correct predictions over number of testing samples. To do so, we use the sklearn.metrics.accuracy\_score function.

EC: get Kaggle dataset and run classifier