CS 3430: SciComp with Py
Coding Exam 3

Vladimir Kulyukin
Department of Computer Science
Utah State University

May 2, 2017

Introduction

- This exam has four problems. The file `coding_exam_3.py` in the zip archive contains the stubs of all the functions you will have to define for this exam.
- Write your name and A-number at the beginning of your `coding_exam_3.py` before submitting it via Canvas.
- The zip archive also contains the subdirectory `images` with the 150 PNG images for Problem 1.
- The file `temps.txt` in the zip archive contains the data for Problem 2.
- The exam is open books and notes. If you use any online materials, please mention them briefly in the comments at the beginning of your source file.
- You may not collaborate with anyone on this exam orally, digitally, in writing, or in any sign language known to man.
- If you cannot code on your RPi, you can code up your solutions on your laptop or desktop so long as you have the necessary packages installed on it.
- Your solutions are due on Canvas by 11:30 a.m. on Tuesday, May 2, 2017. You are always better off submitting an incomplete solution for partial credit than a late complete solution for zero credit.
- The time constraints do not apply to students with learning disabilities who may need additional time on exams.
- Happy Hacking!

Problem 1 (5 pts)

The directory `images` has 150 50x50 PNG images of five Cyrillic character equivalents of the Latin characers `A`, `B`, `V`, `G`, and `D`. The characters have different fonts, sizes, and styles. Figures 1, 2, 3, 4, and 5 show five sample images. There are exactly 30 images of each Cyrillic character.

Figure 1: Cyrillic A

Figure 2: Cyrillic B

Write the function `eval_dtr_built_from_img_dir(img_dir)` that takes an image directory, builds a decision tree to classify the five Cyrillic characters from the character images in that directory, and prints a classification report and a confusion matrix for each of the following test sizes: 20%, 30%, 40%, 50%, 60%, 70%, and 80%. Encode `A` as target 0, `B` as target 1, `V` as target 2, `G` as target 3, and `D` as target 4. Below is a sample test run.

Figure 3: Cyrillic V

Figure 4: Cyrillic G

Figure 5: Cyrillic D

```
>>> eval_dtr_built_from_img_dir('images/')

Test size = 0.200000

Classification report:
             precision    recall  f1-score   support

          0       1.00      1.00      1.00         5
          1       1.00      1.00      1.00        10
          2       1.00      1.00      1.00         8
          3       1.00      1.00      1.00         2
          4       1.00      1.00      1.00         5

avg / total       1.00      1.00      1.00        30

Confusion matrix:
[[ 5  0  0  0  0]
 [ 0 10  0  0  0]
 [ 0  0  8  0  0]
 [ 0  0  0  2  0]
 [ 0  0  0  0  5]]
--------------------------------------------------
Test size = 0.300000

Classification report:
             precision    recall  f1-score   support

          0       1.00      1.00      1.00         8
          1       1.00      1.00      1.00        11
          2       1.00      1.00      1.00         8
          3       1.00      1.00      1.00         9
          4       1.00      1.00      1.00         9

avg / total       1.00      1.00      1.00        45

Confusion matrix:
[[ 8  0  0  0  0]
 [ 0 11  0  0  0]
 [ 0  0  8  0  0]
 [ 0  0  0  9  0]
 [ 0  0  0  0  9]]
--------------------------------------------------
Test size = 0.400000

Classification report:
```

```
             precision    recall  f1-score   support

          0       0.71      1.00      0.83        10
          1       1.00      1.00      1.00        16
          2       1.00      1.00      1.00        13
          3       1.00      0.64      0.78        11
          4       1.00      1.00      1.00        10

avg / total       0.95      0.93      0.93        60

Confusion matrix:
[[10  0  0  0  0]
 [ 0 16  0  0  0]
 [ 0  0 13  0  0]
 [ 4  0  0  7  0]
 [ 0  0  0  0 10]]
--------------------------------------------------
Test size = 0.500000

Classification report:
             precision    recall  f1-score   support

          0       1.00      0.67      0.80        15
          1       1.00      1.00      1.00        15
          2       1.00      1.00      1.00        18
          3       1.00      1.00      1.00        13
          4       0.74      1.00      0.85        14

avg / total       0.95      0.93      0.93        75

Confusion matrix:
[[10  0  0  0  5]
 [ 0 15  0  0  0]
 [ 0  0 18  0  0]
 [ 0  0  0 13  0]
 [ 0  0  0  0 14]]
--------------------------------------------------
Test size = 0.600000

Classification report:
             precision    recall  f1-score   support

          0       1.00      0.74      0.85        19
          1       1.00      0.72      0.84        18
          2       0.88      1.00      0.94        15
          3       0.78      0.90      0.84        20
          4       0.78      1.00      0.88        18

avg / total       0.89      0.87      0.86        90

Confusion matrix:
[[14  0  0  0  5]
 [ 0 13  0  5  0]
 [ 0  0 15  0  0]
 [ 0  0  2 18  0]
 [ 0  0  0  0 18]]
--------------------------------------------------
Test size = 0.700000

Classification report:
             precision    recall  f1-score   support

          0       0.77      1.00      0.87        17
          1       1.00      0.68      0.81        22
```

```
          2       0.91      0.91      0.91        22
          3       0.69      1.00      0.81        22
          4       1.00      0.64      0.78        22

avg / total       0.88      0.84      0.84       105

Confusion matrix:
[[17  0  0  0  0]
 [ 0 15  2  5  0]
 [ 0  0 20  2  0]
 [ 0  0  0 22  0]
 [ 5  0  0  3 14]]
--------------------------------------------------
Test size = 0.800000

Classification report:
             precision    recall  f1-score   support

          0       0.63      1.00      0.78        19
          1       0.83      1.00      0.91        25
          2       1.00      0.79      0.88        24
          3       0.69      1.00      0.81        24
          4       1.00      0.21      0.35        28

avg / total       0.84      0.78      0.73       120

Confusion matrix:
[[19  0  0  0  0]
 [ 0 25  0  0  0]
 [ 0  5 19  0  0]
 [ 0  0  0 24  0]
 [11  0  0 11  6]]
--------------------------------------------------
```

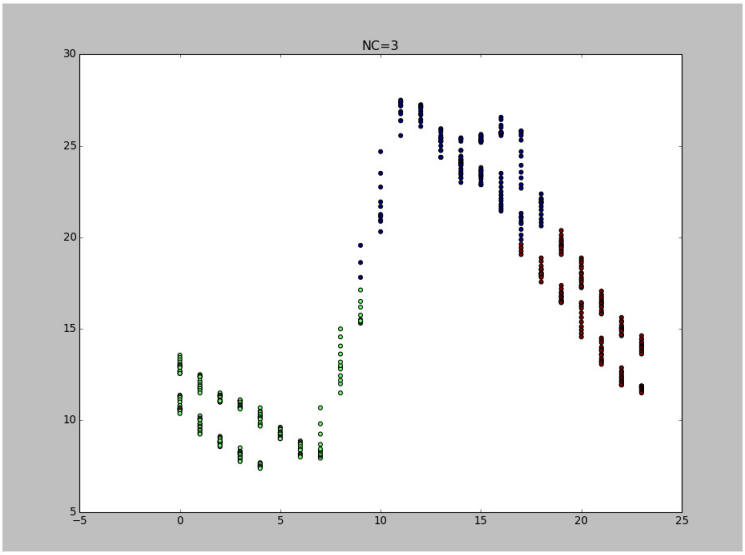

Figure 6: Clustering temperature readings with 3 K-Means clusters

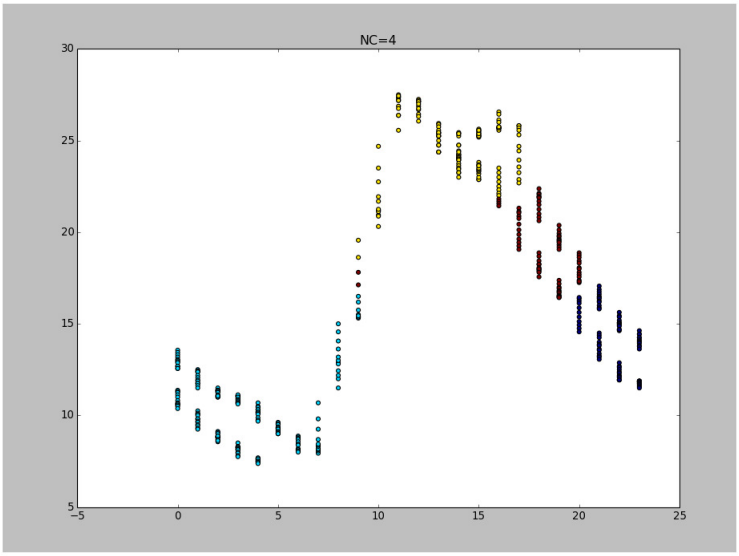

Figure 7: Clustering temperature readings with 4 K-Means clusters

Problem 2 (3 points)

The file `temps.txt` contains the timestamped readings of a Celsius temperature sensor. Below are the first seven lines of this file.

```
2017-04-22_13-35-44 25.25
2017-04-22_13-40-44 25.437
2017-04-22_13-45-44 25.437
2017-04-22_13-50-44 25.375
2017-04-22_13-55-44 25.25
2017-04-22_14-00-44 24.75
2017-04-22_14-05-44 24.437
```

Each line is a 2-tuple. The first element is a timestamp, e.g., `2017-04-22_13-35-44`, the second element is a Celsius reading, e.g., `24.75`. The timestamp consists of the date and time separated by the underscore character. For example, `2017-04-22_13-35-44` consists of the date `2017-04-22` and the time `13-35-44`. The date has the format `YYYY-MM-DD`. The time has the format `HH-MM-SS`. In the time element `HH` ranges from 00 to 23.

Write the function `plot_kmeans_clustered_data(temp_file_path, num_of_clusters)` that takes a file path to the temperature data file and a positive integer specifying the number of clusters for a K-Means classifier. The function creates a K-Means classifier with the specified number of clusters, fits the classifier to the data read from the file, and displays the plot where the cluster of each temperature reading is shown with a different color. Figures 6 and 7 show the temperature readings clustered with 3 and 4 K-Means clusters, respectively. The target of each temperature reading is the hour value of its timestamp. For example, in the tuple `2017-04-22_13-35-44 25.25`, the target of the temperature reading `25.25` is `13`; in the tuple `2017-04-22_14-05-44 24.437`, the target of the temperature reading `24.437` is `14`.

Problem 3 (4 points)

Table 1 gives the distribution of a sample of 2,000 individuals among the six age groups: 20, 30, 40, 50, 60, and 70. Table 2 gives the distribution of 905 recorded minor and major traffic accidents among the same sample of individuals.

Write the functions `displayDependentVars(probDiff=0.1)` and `displayIndependentVars(probDiff=0.1)` that display, for this distribution of traffic accidents, whether the variables `AgeGroup` (`AG`) and `TrafficAccident` (`TA`) are dependent and independent, respectively, at a given threshold. When the absolute value of the difference between the unconditional probability of the traffic accident and the conditional probability of the traffic accident given an age group does not exceed the threshold, the variables are independent. Otherwise, they are dependent. Below is a sample output. Note that this output does not represent the actual answer to this problem. It only exemplifies what your output should look like.

Age Group	Number of People
20	326
30	335
40	351
50	323
60	327
70	338

Table 1: Numbers of people in age groups

Age Group	Number of Traffic Accidents
20	169
30	94
40	136
50	63
60	214
70	229

Table 2: Numbers of traffic accidents in age groups

```
>>> displayIndependentVars()
TA and AG=20 are independent
TA and AG=30 are independent
TA and AG=60 are independent
>>> displayDependentVars()
TA and AG=40 are dependent
TA and AG=50 are dependent
TA and AG=70 are dependent
```

Problem 4 (3 points)

The Vidya Muttri numbers are defined as follows:

$$V(n) = \begin{cases} 1 & \text{if } n = 0 \\ \frac{1}{n+1}(4n-2)V(n-1) & \text{if } n > 0 \end{cases} \tag{1}$$

For example, $V(1) = 1$ and $V(7) = 429$. All Vidya Muttri numbers are positive integers. Write the generator `VG()` that generates the Vidya Muttri numbers. Use your generator in conjunction with list comprehension to implement the function `sum_of_first_n_odd_vms(n)` that computes the sum of the first n odd Vidya Muttri numbers. A couple of test runs:

```
>>> sum_of_first_n_odd_vms(2)
2
>>> sum_of_first_n_odd_vms(3)
7
```

What To Submit

Submit `coding_exam_3.py` via Canvas by 11:30 a.m. on Tuesday, May 2, 2017.