CS57300: Assignment 2

Due date: Sunday September 29, 2019, 11:59pm (submit via Blackboard)

Naive Bayes algorithm on speed dating data

In this programming assignment, you are given a dataset of experimental speed dating events, and your task is to predict whether the participant of a date decides to give his or her partner a second date after the speed dating event (i.e., the "decision" column in the dataset). You will implement algorithms to learn and apply some naive Bayes classification (NBC) models to make such predictions.

More specifically, the dataset **dating-full.csv** is to be used for this assignment. This .csv file contains information for 6744 speed dating events in the comma-separated format. The file **field-meaning.pdf** contains the complete description for the meaning of each column of the dataset.

You are asked to implement your algorithms in Python. Note that although there are many data mining algorithms available online, for this assignment (as well as the next few programming assignments) you must design and implement your **own versions** of the algorithm. DO NOT use any publicly available code including libraries such as ***sklearn***. Your code will be checked against public implementations. In addition, we will not provide separate testing data to you. You are asked to design your own tests to ensure that your code runs correctly and meets the specifications below. **Note**: You may use the ***pandas, numpy, scipy*** libraries for data processing purposes. The only restriction is that you have to write your **own version** of data mining algorithms; you **can not** use any built-in functions for your algorithm. This is a general rule for this assignment and all the upcoming ones as well.

To make it easier to refer to a few sets of columns in the dataset, we will use the following terms (usages will be italicized):

1. *preference_scores_of_participant*: [attractive_important, sincere_important, intelligence_important, funny_important, ambition_important, shared_interests_important]
2. *preference_scores_of_partner*: [pref_o_attractive, pref_o_sincere, pref_o_intelligence, pref_o_funny pref_o_ambitious, pref_o_shared_interests]
3. *continuous_valued_columns*: All columns other than [gender, race, race_o, samerace, field, decision].
4. *rating_of_partner_from_participant*: [attractive_partner, sincere_partner, intelligence_partner, funny_partner, ambition_partner, shared_interests_partner]

In the following, we specify a number of steps you are asked to complete for this assignment. **Note that all results in sample outputs are fictitious and for representation only for this assignment and all upcoming assignments as well**.

1 Preprocessing (4 pts)

Write a Python script named **preprocess.py** which reads the file **dating-full.csv** as input and performs the following operations to output a new file **dating.csv**:

(i) The format of values in some columns of the dataset is not unified. Strip the surrounding quotes in the values for columns **race**, **race_o** and **field** (e.g., 'Asian/Pacific Islander/Asian-American' → Asian/Pacific Islander/Asian-American), count how many cells are changed after this pre-processing step, and output this number.

- Expected output line: **Quotes removed from [count-of-changed-cells] cells.**

(ii) Convert all the values in the column **field** to lowercase if they are not already in lowercases (e.g., Law → law). Count the number of cells that are changed after this pre-processing step, and output this number.

- Expected output line: **Standardized [count-of-changed-cells] cells to lower case.**

(iii) Use label encoding to convert the categorical values in columns **gender**, **race**, **race_o** and **field** to numeric values **start from 0**. The process of label encoding works by mapping each categorical value of an attribute to an integer number between 0 and $n_{values} - 1$ where n_{values} is the number of distinct values for that attribute. Sort the values of each categorical attribute **lexicographically/alphabetically** before you start the encoding process. You are then asked to output the mapped numeric values for 'male' in the **gender** column, for 'European/Caucasian-American' in the **race** column, for 'Latino/Hispanic American' in the **race_o** column, and for 'law' in the **field** column.

- Expected output lines:
  Value assigned for male in column gender: [value-for-male].
  Value assigned for European/Caucasian-American in column race: [value-for-European/Caucasian-American].
  Value assigned for Latino/Hispanic American in column race_o: [value-for-Latino/Hispanic American].
  Value assigned for law in column field: [value-for-law].

(iv) **Normalization:** As the speed dating experiments are conducted in several different batches, the instructions participants received across different batches vary slightly. For example, in some batches of experiments participants are asked to allocate a total of 100 points among the six attributes (i.e., attractiveness, sincerity, intelligence, fun, ambition, shared interests) to indicate how much they value each of these attributes in their romantic partner—that is, the values in *preference_scores_of_participant* columns of a row should sum up to 100 (similarly, values in *preference_scores_of_partner* columns of a row should also sum up to 100)—while in some other batches of experiments, participants are not explicitly instructed to do so.

To deal with this problem, let's conduct one more pre-process step for values in *preference_scores_of_participant* and *preference_scores_of_partner* columns. For each row, let's first sum up all the values in the six columns that belong to the set *preference_scores_of_participant* (denote the sum value as *total*), and then transform the value for each column in the set *preference_scores_of_participant* in that row as follows: *new_value=old_value/total.* We then conduct similar transformation for values in the set *preference_scores_of_partner*.

Finally, you are asked to output the **mean values for each column** in these two sets after the transformation.

- Expected output lines:
  Mean of attractive_important: [mean-rounded-to-2-digits].

...
Mean of shared_interests_important: [mean-rounded-to-2-digits].
Mean of pref_o_attractive: [mean-rounded-to-2-digits].
...
Mean of pref_o_shared_interests: [mean-rounded-to-2-digits].

In summary, below are the sample inputs and outputs we expect to see. We expect **18 lines** of outputs in total (the numbers are ficititious):

```
$python preprocess.py dating-full.csv dating.csv
Quotes removed from 123 cells.
Standardized 456 cells to lower case.
Value assigned for male in column gender:  0.
Value assigned for European/Caucasian-American in column race:  1.
Value assigned for Latino/Hispanic American in column race_o:  4.
Value assigned for law in column field:  2.
Mean of attractive_important:  0.12.
...
Mean of shared_interests_important:  0.34.
Mean of pref_o_attractive:  0.45.
...
Mean of pref_o_shared_interests:  0.56.
```

2 Visualizing interesting trends in data (6 pts)

(i) First, let's explore how males and females differ in terms of what are the attributes they value the most in their romantic partners. Please perform the following task on **dating.csv** and include your visualization code in a file named **2_1.py**.

   (a) Divide the dataset into two sub-datasets by the gender of participant

   (b) Within each sub-dataset, compute the mean values for each column in the set *preference_scores_of_participant*

   (c) Use a **single barplot** to contrast how females and males value the six attributes in their romantic partners differently. Please use color of the bars to indicate gender.

   What do you observe from this visualization? What characteristics do males favor in their romantic partners? How does this differ from what females prefer?

(ii) Next, let's explore how a participant's rating to their partner on each of the six attributes relate to how likely he/she will decide to give the partner a second date. Please perform the following task on **dating.csv** and include your visualization code in a file named **2_2.py**.

   (a) Given an attribute in the set *rating_of_partner_from_participant* (e.g., attractive_partner), determine the number of distinct values for this attribute.

   (b) Given a particular value for the chosen attribute (e.g., a value of 10 for attribute 'attractive_partner'), compute the fraction of participants who decide to give the partner a second date among all participants whose rating of the partner on the chosen attribute (e.g., attractive_partner) is the given value (e.g., 10). We refer to this probability as the *success rate* for the group of partners whose rating on the chosen attribute is the specified value.

(c) Repeat the above process for all distinct values on each of the six attributes in the set *rating_of_partner_from_participant*.

(d) For each of the six attributes in the set *rating_of_partner_from_participant*, draw a scatter plot using the information computed above. Specifically, for the scatter plot of a particular attribute (e.g., attractive_partner), use x-axis to represent different values on that attribute and y-axis to represent the success rate. We expect **6 scatter plots in total**.

What do you observe from these scatter plots?

3 Convert continuous attributes to categorical attributes (3 pts)

Write a Python script named **discretize.py** to discretize all columns in *continuous_valued_columns* by splitting them into **5** bins of equal-width in the range of values for that column (check **field-meaning.pdf** for the range of each column; for those columns that you've finished pre-processing in Question 1(iv), the range should be considered as [0, 1]). If you encounter any values that lie outside the specified range of a certain column, please treat that value as the max value specified for that column. The script reads **dating.csv** as input and produces **dating-binned.csv** as output. As an output of your scripts, please print the number of items in each of the 5 bins. **Bins should be sorted from small value ranges to large value ranges** for each column in *continuous_valued_columns*.

The sample inputs and outputs are as follows. We expect 47 lines of output, and the order of the attributes in the output should be the same as the order they occur in the dataset:

```
$python discretize.py dating.csv dating-binned.csv
age:  [3203 1188 1110 742 511]
age_o:  [2151 1292 1233 1383 685]
importance_same_race:  [1282 4306 1070 58 28]
...
like:  [119 473 2258 2804 1090]
```

4 Training-Test Split (2 pts)

Use the ***sample*** function from ***pandas*** with the parameters initialized as **random_state = 47, frac = 0.2** to take a random 20% sample from the entire dataset. This sample will serve as your test dataset, and the rest will be your training dataset. (**Note**: The use of the random_state will ensure all students have the same training and test datasets; incorrect or no initialization of this parameter will lead to non-reproducible results). Create a new script called **split.py** that takes **dating-binned.csv** as input and outputs **trainingSet.csv** and **testSet.csv**.

5 Implement a Naive Bayes Classifier (15 pts)

- Learn a NBC model using the data in the training dataset, and then apply the learned model to the test dataset.
- Evaluate the accuracy of your learned model and print out the model's accuracy on both the training dataset and the test dataset as specified below.

Code Specification:

Write a function named ***nbc(t_frac)*** to train your NBC which takes a parameter ***t_frac*** that represents the fraction of the training data to sample from the original training set. Use the ***sample*** function from ***pandas*** with the parameters initialized as **random_state = 47, frac =** ***t_frac*** to generate random samples of training data of different sizes.

1. Use all the attributes and all training examples in **trainingSet.csv** to train the NBC by calling your ***nbc(t_frac)*** function with ***t_frac*** = 1. After get the learned model, apply it on all examples in the training dataset (i.e., **trainingSet.csv**) and test dataset (i.e., **testSet.csv**) and compute the accuracy respectively. Please put your code for this question in a file called **5_1.py**.

   - Expected output lines:
     Training Accuracy: [training-accuracy-rounded-to-2-decimals]
     Testing Accuracy: [testing-accuracy-rounded-to-2-decimals]

   The sample inputs and outputs are as follows:
   ```
   $python 5_1.py
   Training Accuracy:  0.71
   Testing Accuracy:  0.68
   ```

2. Examine the effects of varying the number of bins for continuous attributes during the discretization step. Please put your code for this question parts(ii, iii, iv) in a file called **5_2.py**.

   (i) Given the number of bins $b \in B = \{2, 5, 10, 50, 100, 200\}$, perform discretization for all columns in set *continuous_valued_columns* by splitting the values in each column into b bins of equal width within its range. For this task, you can re-use your **discretize.py** code to perform the binning procedure, now taking the number of bins as a parameter and using **dating.csv** as input as earlier.)

   (ii) Repeat the train-test split as described in Question 4 for the obtained dataset after discretizing each continuous attribute into b bins.

   (iii) For each value of b, train the NBC on the corresponding new training dataset by calling your ***nbc(t_frac)*** function with ***t_frac*** = 1, and apply the learned model on the corresponding new test dataset.

   (iv) Draw a plot to show how the value of b affects the learned NBC model's performance on the training dataset and the test dataset, with x-axis representing the value of b and y-axis representing the model accuracy. Comment on what you observe in the plot.

   The sample inputs and outputs are as follows:
   ```
   $python 5_2.py
   Bin size:  2
   ```

```
Training Accuracy:  0.34
Testing Accuracy:  0.12
Bin size:  5
Training Accuracy:  0.78
Testing Accuracy:  0.56
.
.
Bin size:  200
Training Accuracy:  0.90
Testing Accuracy:  0.88
```

3. Plot the learning curve. Please put your code for this question in a file called **5_3.py**.

   (i) For each f in $F = \{0.01, 0.1, 0.2, 0.5, 0.6, 0.75, 0.9, 1\}$, randomly sample a fraction of the training data in **trainingSet.csv** with our fixed seed (i.e., **random_state=47**).

   (ii) Train a NBC model on the selected f fraction of the training dataset (You can call your ***nbc(t_frac)*** function with ***t_frac*** $= f$). Evaluate the performance of the learned model on all examples in the selected samples of training data as well as all examples in the test dataset (i.e., **testSet.csv**), and compute the accuracy respectively. Do so for all $f \in F$.

   (iii) Draw one plot of learning curves where the x-axis representing the values of f and the y-axis representing the corresponding model's accuracy on training/test dataset. Comment on what you observe in this plot.

Submission Instructions:

Submit through Blackboard

1. Make a directory named *yourFirstName_yourLastName_HW*2 and copy all of your files to this directory.

2. **DO NOT** put the datasets into your directory.

3. Make sure you compress your directory into a **zip folder** with the same name as described above, and then upload your zip folder to Blackboard.

   Keep in mind that old submissions are overwritten with new ones whenever you re-upload.

Your submission should include the following files:

1. The source code in python.

2. Your evaluation & analysis in .pdf format. Note that your analysis should include visualization plots as well as a discussion of results, as described in details in the questions above.

3. A README file containing your name, instructions to run your code and anything you would like us to know about your program (like errors, special conditions, etc).