CS37300 Homework 2

Due date: Thursday October 10, 2019 11:59 pm

In this programming assignment you will implement a naive Bayes classification algorithm and evaluate it on modified *Yelp* dataset. Instructions below detail how to turn in your code and assignment on data.cs.purdue.edu.

Note: You need to implement this assignment from scratch in Python. Do not use any models already implemented in machine learning libraries (e.g., scikit-learn). Also, programs that print more than what is required will be penalized. Your code needs to run in the server: `data.cs.purdue.edu`. Check that it works there by executing your script from the terminal.

For the coding implementations, submit your code as a Python program (Python 3.6). For all other parts write your answers in a single PDF. Name this yourusername_hw2.pdf.

As usual, label the plots with the question number. Your homework must contain your name and Purdue ID at the start of the file. If you omit your name or the question numbers for the plots, you will be penalized.

To submit your assignment, log into data.cs.purdue.edu (physically go to the lab or use ssh remotely) and follow these steps:

1. Make a directory named yourusername_hw2 (all letters in lower case) and copy your PDF file and Python file inside it. To do it remotely, use:
   `scp ./path/to/your-file.pdf your-id@data.cs.purdue.edu:./remote/path/from-home-dir/`
2. Go to the directory containing yourusername_hw2 (e.g., if the files are in `/homes/dan/danhw2`, go to `/homes/dan`), and execute the following command:
   `turnin -c cs373 -p hw2 yourusername_hw2`
   (e.g. Dan would use: `turnin -c cs373 -p hw2 dan_hw2` to submit his work)
3. To overwrite an old submission, simply execute this command again.
4. To verify the contents of your submission, execute the following command:
   `turnin -v -c cs373 -p hw2`
   Do not forget the `-v` option, else your submission will be overwritten with an empty submission.

Dataset Details

Download the datafile `yelp_data.csv` from Piazza. This data set is a pre-processed version of the Yelp dataset you used in HW1. The datafile has 24,813 instances in rows and 20 attributes. Here, 4 attributes are multi-valued: ({`ambience`, `parking`, `dietaryRestrictions`, `recommendedFor`}).

Algorithm Details

You will implement algorithms to learn (i.e., estimate) and apply (i.e., predict) the NBC that we discussed in class.

Pre-processing: The following steps have to be followed:

- Binary features (columns) should be created for every value corresponding to multi-valued attributes ({`ambience`, `parking`, `dietaryRestrictions`, `recommendedFor`}). For example: The attribute "recommendedFor' can take the following multiple values for a data instance: ('breakfast','lunch','brunch','dinner','latenight','dessert'). The column corresponding to "recommendedFor" should be replaced by 6 new columns having binary indicator (True/False) illustrating whether that value is true for that data instance.
- Missing value treatment in attributes should be done by adding "None".
- If any further pre-processing of the data is required or used, it should be clearly mentioned in the report.

Features: All the attributes except the last in the pre-processed data will be used as features for NBC .i.e. X.

Class label: Consider the attribute `outdoorSeating` as the class label, i.e., $Y = \{1, 0\}$.

Smoothing: Implement Laplace smoothing in the parameter estimation. For an attribute X_i with k values, Laplace correction adds 1 to the numerator and k to the denominator of the maximum likelihood estimate for $P(X_i = x_i \mid Y)$.

Evaluation: When you apply the learned NBC to predict the class labels of the examples in the test set, use (i) zero-one loss, and (ii) squared loss to evaluate the predictions.

Let $y(i)$ be the true class label for example i and let $\hat{y}(i)$ be the prediction for i. Let p_i refer to the probability that the NBC assigns to the true class of example i (i.e., $p_i := p(\hat{y}(i) = y(i))$. If $p_i \geq 0.5$, the prediction for example i will be correct (i.e., $\hat{y}(i) = y(i)$), but otherwise if $p_i < 0.5$, the prediction will be incorrect (i.e., $\hat{y}(i) \neq y(i)$).

Then the zero-one loss for a test dataset T of n instances is:

$$Loss_{0/1}(T) = \frac{1}{n} \sum_{i \in n} \left\{ \begin{array}{ll} 0 & \text{if } y(i) = \hat{y}(i) \\ 1 & \text{otherwise} \end{array} \right\} \tag{1}$$

The squared loss for a test dataset T of n instances is:

$$Loss_{sq}(T) = \frac{1}{n} \sum_{i \in n} (1 - p_i)^2 \tag{2}$$

Code specification

Your python script should take two arguments as input.

1. *trainingDataFilename*: corresponds to a subset of the Yelp data (in the same format as `yelp_data.csv`) that should be used as the *training set* in your algorithm.
2. *testDataFilename*: corresponds to another subset of the Yelp data (in the same format as `yelp_data.csv`) that should be used as the *test set* in your algorithm.

Your code should read in the training/test sets from the csv files, learn a NBC model from the training set, apply the learned model to the test set, and evaluate the predictions with zero-one loss and squared loss.

Name your file nbc.py. The input and output should look like this:

```
$ python nbc.py train-set.csv test-set.csv
...
ZERO-ONE LOSS=0.2305
SQUARED LOSS=0.0711
```

Note: This is how we will run your code to grade correctness in Q2 below.

Assignment

After preprocessing the Yelp dataset obtained, D with discrete attributes $\mathbf{X}$ and class Y as described above,

1. NBC details (20 pts)
   (a) Write down the mathematical expression for $P(Y \mid X)$ given by the NBC. (2 pts)
   (b) Suppose your data has binary class labels , i.e $y \in \{0, 1\}$, write the expression for predicting the class for a given input row. You will use this expression in your implementation. (1 pts)
   (c) State the naive assumption that lets us simplify the expression $P(X \mid Y)P(Y)$. What rule(s) of probability are used to simplify the expression? Is this assumption true for the given Yelp data? Explain why or why not? (3 pts)
   (d) What part of the expression corresponds to the class prior? Considering the entire Yelp data as the training dataset, calculate the maximum likelihood estimate for the class prior with and without smoothing. What is the effect of smoothing on the final probabilities? (4 pts)
   (e) Specify the full set of parameters that need to be estimated for the NBC model of the Yelp data. How many parameters are there? (2 pts)
   (f) Write an expression for an arbitrary conditional probability distribution (CPD) of a discrete attribute X_i with k distinct values (conditioned on a binary class Y). Include a mathematical expression for the maximum likelihood estimates of the parameters of this distribution (with smoothing), which correspond to counts of attribute value combinations in a data set D. (2 pts)

(g) For the Yelp data, explicitly state the mathematical expression for the maximum likelihood estimates (with smoothing) of the CPD parameters for the attribute `priceRange` conditioned on the the class label `outdoorSeating`. (2 pts)

(h) Consider the entire Yelp data as the training dataset and `outdoorSeating` as the class label. Estimate the conditional probability distributions of the following attributes with and without smoothing:

(i) `delivery`
(ii) `alcohol`
(iii) `noiseLevel`
(iv) `attire`

What is the effect of smoothing (e.g., any difference compared to Q1d)? Which attribute shows the most association with the class? (4 pts)

2. Implement a naive Bayes classification algorithm in python. (20 pts)
We will run several tests on your code to assess the accuracy for different samples.

3. Evaluate the NBC using cross validation and learning curves. (10 pts)

Cross-validation is a powerful tool to assess how well your learned model generalizes: a subset of the data is kept aside (called the *validation set*) before training begins. After a model is learned, the validation set can be used to test the performance of the learned model.

In k-fold cross-validation, data is divided into k subsets. In each iteration, one such subset is used as the test set and the remaining $k-1$ subsets are collectively used as the training set. This process is repeated k times. We then take the average error across all k trials. This gives a more accurate measure of model quality than if you were to hold out some fixed subset of the training data as validation set.

A **learning curve** is a way to plot how the scoring function (e.g., 0/1-loss, squared loss) evolves as size of the training set changes.

(a) For each k in $[1, 10, 50]$:

```
For $i$ in [0...9]:
    • Randomly sample k % of the data to use as the training dataset.
    • Use the remaining (100-k)% of the data as the test dataset.
    • Learn a model from the training data and apply it to the test data.
    • Measure the performance on the test data using zero-one loss.
```

Record the mean zero-one loss observed across the ten trials for each training set size (i.e., sample %). Record the mean squared loss across the ten trials for each training set size. (8 pts)

(b) Plot a learning curve of training set size vs. zero-one-loss (report the mean performance measured above). Compare to the baseline *default* error that would be achieved if you just predicted the most frequent class label in the overall data. Discuss the results (e.g., how is zero-one loss impacted by training set size). (6 pts)

(c) Plot a learning curve of training set size vs. square-loss. Discuss how zero-one loss performance compares to square-loss.(6 pts)