CS39000-DM0 Homework 3

Due date: Thursday March 9, 11:59pm

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

Note: You are welcome to use any methods from the standard python libraries available (e.g., numpy) but you shouldn't use methods from machine learning libraries (e.g., scikit-learn).

Dataset Details

Download the datafile `yelp2.csv` from Blackboard. The datafile has 21,091 rows and 15 discrete attributes. The first column, `goodForGroups`, is used as a class label in this assignment.

Algorithm Details

You will implement algorithms to learn (i.e., estimate) and apply (i.e., predict) the NBC that we discussed in class. **For this assignment, consider a missing value as a value, "BLANK", instead of ignoring the examples with missing values. For instance, the column, priceRange, should have five possible values, 1, 2, 3, 4, and BLANK.**

Features: Consider the 14 discrete attributes excluding the first column, `goodForGroups`, in `yelp2.csv` for $\mathbf{X}$. Since the NBC can handle multi-valued discrete attributes, there is no need to create binary features.

Class label: Consider the attribute `goodForGroups` 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., $\hat{y}(i) = y(i)$). Otherwise, 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:

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

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 `yelp2.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 `yelp2.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. You can (and should) use wrapper methods to run your own analysis for Q3.

Assignment

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

1. NBC details (20 pts)

   Consider the entire Yelp data as the training dataset for questions (a)-(g).

   (a) Write down the mathematical expression for $P(Y \mid X)$ given by the NBC.

   (b) 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?

   (c) What part of the expression corresponds to the class prior? Calculate the maximum likelihood estimate for the class prior with and without smoothing. What is the effect of smoothing on the final probabilities?

   (d) Specify the full set of parameters that need to be estimated for the NBC model of the Yelp data. How many parameters are there? Note that each conditional probability is considered one parameter.

   (e) 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.

(f) 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 `goodForGroups`. Don't forget missing values.

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

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

What is the effect of smoothing (e.g., any difference compared to Q1c)? Which attribute shows the most association with the class?

2. Implement a naive Bayes classification algorithm in python. (20 pts)
Your code should read in the training/test sets from the csv files as it is explained in the code specification. Your NBC calculates the CPDs based only on the training data, and makes a prediction for each example in the test data. If the example in the test data contains a value for column X_i that never occurs in the training data, you should use the following conditional probability:

$$P(X_i = \text{unseen value}|Y = y_j) = \frac{1}{(\#\text{examples of } Y = y_i \text{ in the training data}) + k}$$

where k denotes the number of values for column X_i that are seen in the training data.

We will run several tests on your code to assess the accuracy for different samples, so your code has to strictly follow the aforementioned code specification.

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

(a) For each $\rho \in \{0.1, 1, 10, 50\}$:

For i-th trial ($i = [0, 9]$):

- Randomly sample ρ % of the data to use as the training dataset.
- Use the remaining $(100 - \rho)$ % 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 ρ %). Record the mean squared loss across the ten trials for each training set size.

(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).

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

Submission Instructions

You should upload your work to Blackboard, as a ZIP file. Your submission should include the following files:

1. The source code in python.
2. Solutions to Q1 and Q3 in .pdf format. Note that your analysis for Q3 should include two plots and discussion of the results.
3. A README file contaiting your name, instructions to run your code and anything you would like us to know about your program (like errors, special conditions etc.).

Extra Credit

This part is optional
You can get extra credit in three ways-

1. Improve the performance of NB (5 pts). Feel free to experiment with different smoothing techniques, selecting a subset of the features, or adding new features (e.g., conjunctions of the basic features). **Your report should include a full description, and you should compare your results to the original feature set.**
2. Compare NB to the Perceptron algorithm (5 pts). You will need to implement the Perceptron algorithm, and add a hyperparameter T, which determines the number of iterations over the training data (to avoid an infinite loop). **Your report should include a full description of the algorithm, and you should also compare the results to your NB implementation.**