Statistical Learning Theory, Fall 2017

Assignment 2: Classification

The purpose of this exercise is to compare the Naive Bayes (NB) model to Logistic Regression (LR) on data with categorical attributes. We will first train/fit an NB and an LR model on some data set and then compare their performance by looking how well they predict in terms of 0/1-loss (misclassification rate, error percentage). We will do this on a real-world data set, the *Car Evaluation* data. The data set can be downloaded from the UCI Machine Learning repository, https://archive.ics.uci.edu/ml/datasets/Car+Evaluation.

Data. The goal is to predict (categorical) car acceptability based on a total of six price, technical and comfort characteristics. The number of observations is 1728 and all attributes are categorical. Note there are no missing values in this data set. The data can be imported to R (for instance by copying the web page data into a .txt file and then using read.table as is done in example code below). For educational purposes, we recode the outcome variable from four to two levels as follows:

R:

```
raw <- read.table("cardata.txt", sep = ",")
mydata <- data.frame(raw[, 1:6], eval = ifelse(raw[, 7] == "unacc", "Negative",
"Positive"))
```

Python:

```
myfile = open("cardata.txt", "r")
mydata = []
for line in myfile:
    row = line.strip().split(',')
    if row[6] == 'unacc':
        row[6] = 'Negative'
    else:
        row[6] = 'Positive'
    mydata.append(row)
```

As a result, you should now have a data frame with the outcome in the last column.

Methods. For R, Naive Bayes maximum likelihood estimation has been implemented in the *e1071* package of the TU Vienna. For Python, Naive Bayes maximum likelihood estimation has been implemented in the *scikit learn* (sklearn) package. See the manual pages of the packages for examples of the uses of the packages. Our course book (Hastie, Tibshirani, Friedman) is rather short on Naive Bayes; consult wikipedia and/or other web sources if you need more information. Logistic regression conditional likelihood maximization can be performed in R via the *glm* (generalized linear model)-command. For an example of its use, check out the mini-tutorial *Logistic Regression in R* by Christopher Manning available from `http://nlp.stanford.edu/manning/courses/ling289/logistic.pdf`. An implementation of Logistic regression conditional likelihood maximization in Python can be found in the *scikit learn* (sklearn) package.

General Remarks

- It is allowed to compute results with any programming language you like (e.g. R or Python), but we advise R or Python given the packages we suggest throughout the assignment.
- Although code is a necessity for this report and its results a significant contributor to the grade, it is not sufficient. The majority of points is earned by interpreting and explaining your findings in a separate report.
- We encourage students learning from one another, but make sure to write your code and your report individually.
- Note that the corresponding *predict*-functions in R work differently for NB and LR. Check out page 6 of Manning's tutorial to get an idea about how to compute the 0/1-performance and how to use *predict* with LR. Check out the *e1071* documentation to find out how to use *predict* with NB.
- Finally, recall that the outcome should not be treated as a feature.

Assignment. Please do all the following experiments and summarize the results in a written report that includes the required graph and answers to all of the questions below. *It is advisable to first read the entire assignment before computing anything!*

Experiments. Randomly split the data in two equal parts; the first part constitutes the training data, the second part is the test data. Suppose that the training data consists of say N vectors. Now, do the following for each $n = 1 \dots N$:

Naive Bayes

1. Train an NB model on the first n data vectors in the training data according to the following specifications:
   (a) with Laplace smoothing, assuming $\alpha = 1$, i.e. one "fake" observation per each class. This means that, for each class y (e.g. $y \in \{\text{Negative}, \text{Positive}\}$) and each feature (attribute) X_j , the probability that X_j has value k given that the class is y is set to $(n_{k,y} + \alpha)/(n_y + \alpha K_j))$, where $n_{k,y}$ is the number of data vectors where the feature has value k and the class is y, n_y is the number of data vectors with class y, and K_j is the number of values that feature X_j can take.
   (b) with Laplace smoothing for a value of α close to 0, say 0.1 (so that the estimated parameters are very close to maximum likelihood).
   (c) with $\alpha = 10$.
2. Now use the three inferred models to predict the test set and measure loss using the 0/1-loss function.

Logistic Regression

3. Train an LR model on the first n samples in the training set. Use the inferred model to do prediction on test set with the 0/1-loss. Hint: you may experience some problems with the LR procedure for low n, in that case please proceed to the next n. (The reason is that for low n, the ML parameters in the LR model may not be uniquely identified.)
4. Train an LR model on the first n samples in the training set, regressing only on the first 2 attributes (and the intercept). Use the inferred model to evaluate prediction on the test set with 0/1-loss.

Comparison

As a result of this procedure, you should have obtained $N \times 5$ 0/1-errors: for each sample size of the training set, you learned 5 predictors (3 NB, 2 LR) and tested them on the full test set.

5. Now make a plot of these five 0/1-errors as a function of $n = 1 \ldots N$. Include the plot in your report and analyse its meaning. Please ensure the methods (i.e. the lines) in your plot are identified clearly, e.g. with a legend. Furthermore, plotting the horizontal axis on a log scale may help interpretation. To get a smooth plot, you may want to repeat the whole procedure many (say $M = 20$) times and average the results; each time, you start with a random permutation of all the samples, putting the first half into training and second half into test set. Then you learn and predict for n from 1 to N, ending up with a new $N \times 5$ matrix. Therefore, you will obtain M of these matrices. Make a new final $N \times 5$ matrix with, for all i, j, entry (i, j) equal to the average of all the (i, j)-entries of the M matrices you constructed. If this takes too much computation time, you may skip some values of n and interpolate in the graph.

6. Is one of the five methods the overall winner? Are some of the methods generally desirable or undesirable? Is what you see in accordance with what you would have expected from the theoretical insights you have learned in the course? If so, explain. If not, what is different from what you expected?

7. How extreme (close to 0 or 1) are the predictive probabilities based on NB and LR? Can you conclude something about the predictive ability for each outcome class of both methods?

Bonus

For **bonus points**, you may want to add a final curve in the graph based on using penalized rather than standard logistic regression. This cannot be implemented with the standard *glm*-function though and you will need to install additional package(s). For example, in R one can use the *glmnet* package. In Python one can use the *scikit learn* (sklearn) package.

.

Please submit your code (e.g .R or .py) and report (in .pdf format) including your name and student number to

`d.van.der.hoeven@math.leidenuniv.nl`