MIS 382N: Advanced Predictive Modelling

Assignment #4

Due: Wed, Nov 05, 2014; Total points: 40

Upload instructions: Your homework should be a single pdf file containing all the answers, plots/pictures and code. You may however insert equations by hand if you wish. Name the pdf file as <your eid>.pdf The included code should be straightforward to run by just copying it into the interpretrator or the command line without any changes. So add any include statements for libraries etc. in the code as well. Homeworks are due at the beginning of class on the due date, submitted through Canvas. If the choice of a programming language is specified in the problem, please don't use any other language.

Note: All the data files needed for this hw are zipped and uploaded in canvas as `hw4files.zip`.

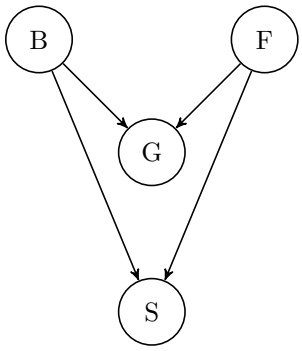
1. **(3+3+3=9 pts) Bayesian Networks**

   In the Bayesian network shown below, B stands for "Battery", F for "Fuel", G for "Gauge", and S for "Start". Compute the following probabilities:

   (a) P(B = bad, F = empty, G = empty, S = no)

   (b) Given that the fuel is not empty, compute the probability that the car will not start.

   (c) Given that the battery is bad and the car doesn't start, compute the probability that the gauge will show empty.



P(B = bad) = 0.1
P(F = empty) = 0.2

P(G = empty | B = good, F = not empty) = 0.1
P(G = empty | B = good, F = empty) = 0.8
P(G = empty | B = bad, F = not empty) = 0.2
P(G = empty | B = bad, F = empty) = 0.9

P(S = no | B = good, F = not empty) = 0.1
P(S = no | B = good, F = empty) = 0.8
P(S = no | B = bad, F = not empty) = 0.9
P(S = no | B = bad, F = empty) = 1.0

2. **(4+3+3=10 pts) Logistic Regression**

   In this problem you will use Logistic Regression to perform binary classification. The complete description of the dataset can be found at `https://archive.ics.uci.edu/ml/datasets/Ionosphere`. It contains various features, where the last column is the target variable. Drop the second column from the dataset and then standardize the features (make each column zero mean and unit variance). Split the dataset (randomly) into training and test set where the training data is 70% of the total data.

   Use the following R command to load the dataset:

```
read.table(
  "https://archive.ics.uci.edu/ml/machine-learning-databases/ionosphere/ionosphere.data",
   sep=","
);
```

(a) Fit a ridge logistic regression (i.e., logistic regression with a ridge penalty) model using `glmnet` package. Use 10-fold cross-validation to choose the strength of the regularizer. Report the mean error rate (fraction of incorrect labels) on both the training and test sets.

(b) Plot the receiver operating characteristic (ROC) curve on the test data. Use `ROCR`[1] to get the ROC curve. Report the area under the ROC curve (AUC). What information does the ROC curve show that is "masked" by the AUC?

(c) Generate and plot the lift curve for the logistic regression model.

3. **(2+2+1+2+2=9 pts) Regression Trees (RT)**. In this problem, you will use RT to predict Miles Per Gallon ("MPG") in a cars dataset adapted from

`http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data`, given all the other variables as predictors. The data has already been divided into `data.train` and `data.test` (available on canvas, both are csv text files). Use `R` package `rpart` for this question.

(a) Build an RT on the training data using the `rpart` package. Set the `control` variable to use 5-fold crossvalidation, and `cp` to be a reasonably low value (say, 0.0001). Use the method `anova`.

(b) Make use of `printcp()` and then `prune()` on the built model to prune the tree appropriately. Recall pruning is useful in avoiding overfitting. Justify your choice.

(c) Plot the tree corresponding to your pruned RT model. You can use the function `prp()` in the package `rpart.plot`.

(d) Report MSE for predicted values for both train and test data. Plot the predicted values (x-axis) against the target values (y-axis) in test data using your model, and comment if the model captures the relationships well.

(e) Repeat the experiment by using method as `poisson` when calling `rpart`, and prune using the same way. Does it give better train/test MSE than `anova` ? (You don't have to do other parts of this question using method `poisson`, just build, prune and predict to calculate the MSE).

4. **(2+3+2=7 pts) Spam classification using support vector machines. This problem should be solved using the *scikit_learn svm* package in Python.**[2]
Consider the email spam dataset `spam.csv`, which can be loaded using the *numpy* command `numpy.genfromtext()`. The complete dataset description can be found at `http://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection`. This consists of 4601 email messages, from which 57 features have been extracted. These are as follows:

- 48 continuous real [0,100] attributes of type word_freq_WORD = percentage of words in the e-mail that match WORD, i.e. 100 * (number of times the WORD appears in the e-mail) / total number of words in e-mail. A "word" in this case is any string of alphanumeric characters bounded by non-alphanumeric characters or end-of-string.
- 6 continuous real [0,100] attributes of type char_freq_CHAR = percentage of characters in the e-mail that match CHAR, i.e. 100 * (number of CHAR occurences) / total characters in e-mail
- 1 continuous real [1,...] attribute of type capital_run_length_average = average length of uninterrupted sequences of capital letters
- 1 continuous integer [1,...] attribute of type capital_run_length_longest = length of longest uninterrupted sequence of capital letters
- 1 continuous integer [1,...] attribute of type capital_run_length_total = sum of length of uninterrupted sequences of capital letters = total number of capital letters in the e-mail
- 1 nominal 0,1 class attribute of type spam = denotes whether the e-mail was considered spam (1) or not (0), i.e., unsolicited commercial e-mail.

[1] A basic tutorial for the package can be found at `http://rocr.bioinf.mpi-sb.mpg.de/ROCR_Talk_Tobias_Sing.ppt`.
[2] Tuning and training the SVM on this dataset might take some time

- 1 boolean attribute of type test = denotes whether the e-mail is part of the test set (TRUE) or not (FALSE).

(a) Apply a linear SVM, for the spam detection problem on the standardized pre-processed datasets. Specify how you chose the cost penalty for the model. Try different values of the penalty (0.1,1,2,5,10) and report only the best performing model's accuracy and the corresponding penalty value.

(b) Repeat (a) but using a Gaussian radial basis kernel. Mention any additional parameter(s) you needed to determine for this kernel function. Now try different values of this parameter $(2^{-12}, 2^{-10}, 2^{-8}, 2^{-6}, 2^{-5})$ and different penalty costs for each of these parameter values. Essentially you will have 25 models (using the same set of penalties that you used for linear SVM) with the RBF kernel. Report the accuracy for the best performing model and the corresponding penalty and kernel parameter value. Comment on which model you will prefer to use in practice (linear or RBF) and why? (be brief and answer in points.)

(c) Summarize the performance of all the classifiers that you applied to the spam filter problem in a single table. Comment briefly on the performance of the 2-class linear SVM vs 2-class RBF SVM model.

5. **(5 pts) Outlier detection via OneClassSVM** In this question, you will explore application of OneClassSVM for outlier detection. You will make use of a simulated dataset `oneclassdata.csv` (available on canvas), and the scikit library `sklearn.svm`. The first two columns of `oneclassdata.csv` are the features, while the last column is a flag'label saying whether the respective datapoint is a legitimate "inlier" (flag will be 1 here), or an outlier (=0).

(a) Extract out the feature matrix (will be 1000*2) from the provided dataset and fit a one class SVM model on it. You can use the following code snippet to initialize the oneclassSVM object (note the use of the parameters and the RBF kernel:

```
modl = svm.OneClassSVM(nu=0.95 * outliers_fraction + 0.05,
                                        kernel="rbf", gamma=0.1)
```

(b) Note that you did not use the true label information while training the model. Get the predictions made by the model on the datapoints. This can be done from by using `modl.decision_function()`. Lets call this `ypred`.

(c) Suppose we are expecting that 10% of the data are outliers. SVMs can do classification by thresholding the continuous predictions. Based on the predicted scores `ypred`, classify the top 90% as 1 (inliers), and the bottom 10% as 0 (outliers) Report the number of errors against the true label information made by using this method. Note that an error is made when (true label ! = predicted label).