STA6703 SML Take-Home Prelim, Fall 2022

Directions

Please submit **one PDF** file including all your reports (answer + code + figures + comments; must be easily readable and have file size under a few megabytes) and **one R code script**. The R script is supplementary material to ensure that your code runs correctly. If you are using RMarkdown, please also include your `.Rmd` file.

Place these two (or three) files in a folder, make a zip or rar archive, and submit the archive electronically via Dropbox file request at `tinyurl.com/nbliznyuk-submit-files` (on the landing page, enter your name so that we know it is you and email so that you get a confirmation).

This exam is "open books/notes/class videos/Dropbox course folder", "open R", but "closed internet". Absolutely no collaboration or discussion with anyone (except the course staff). Using or trying to obtain solutions of others is prohibited.

Deadline: 72 hours after the exam is made available; i.e., the start time is Monday 17-Oct (or Sunday 16-Oct) 10:00 AM EST.

There are 6 problems. Unless marked otherwise, each is worth 20 points. There are 110 points total, but the grading will be out of 100 points; i.e., up to 10 bonus points if you solve everything correctly!

To avoid unnecessary penalties:

1. Read and follow directions and problem-specific guidelines (next page) in their entirety.
2. Use R for all problems (your acquired knowledge of R is a part of this assessment). If you can ONLY solve a problem using Python, do that for partial credit (since this is better than not doing the problem at all).
3. Your code must run and produce the results you report (test this!)
4. Answer the questions asked. Show all your relevant work; do not show irrelevant work.
5. Unless explicitly asked to do otherwise, report numerical results up to three digits after the decimal place, without the scientific "E" notation. Do NOT directly include the output that R "spits out". To this end, you can use the following function, `myRound()`:

   ```
   myRound <- function(x, acc=3) {mult = 10^acc; round(x*mult)/mult}
   ```

6. Do NOT copy and paste the code from the prelim pdf; instead, use the source `.Rmd` file.
7. Proof-read what you submit; continue proof-reading until you stop findings typos and other "bugs."
8. This assignment will take a well-prepared (and well-rested) student under one day (8 hours) of focused work to solve. Do not wait till the last day; start asap!

Please inform the instructor of suspected typos asap at nbliznyuk@ufl.edu

Required Problems (for submission)

Guidelines to specific problems:

Problem 1: Your observed data for players B and C should be generated as

```
set.seed(0); n = 100; origData = rnorm(n);       # case 1
set.seed(0); n = 100; origData = rt(n,df=3);     # case 2
set.seed(0); n = 100; origData = rt(n,df=25);    # case 3
# comment out the irrelevant lines depending on the case you are doing
```

For all players (A, B and C), all $m = 1000$ datasets must be generated at once as an $m \times n$ matrix (i.e., m is the number of replicated datasets, and the ith replicated dataset is in the ith row of this replications matrix). To this end, `set.seed(0)` (for each player) and generate a vector z (of length $m \cdot n$) of iid replicates from the appropriate distribution, then reshape it into a matrix as `matrix(z,nrow=m)`. For nonparametric bootstrap, generate z as `z=sample(x=origData, size=n*m, replace=TRUE)`.

Problems 2-4: Use the following code (with `seed=0`) to create your validation and training folds:

```
myCVids <- function(n, K, seed=0) {
  # balanced subsets generation (subset sizes differ by at most 1)
  # n is the number of observations/rows in the training set
  # K is the desired number of folds (e.g., 5 or 10)
  set.seed(seed);
  t = floor(n/K); r = n-t*K;
  id0 = rep((1:K),times=t)
  ids = sample(id0,t*K)
  if (r > 0) {ids = c(ids, sample(K,r))}
  ids
}
# Example
inds.part = myCVids(101, 5); # partition of the indices into K=5 sets
k = 1; isk = (inds.part == k); # k varies from 1 to K
valid.k = which(isk); train.k = which(!isk); # indices of kth valid and train folds
```

Problem 6: To generate data, set `set.seed(0)` and generate all your data at once as an $m \times n$ matrix following the strategy of player C in problem 1.

Problem 1 [20 pts + 5 pts bonus]:

Simulation-based inference vs parametric bootstrap vs nonparametric bootstrap.

Consider three setups (A, B and C) that will be used in this problem:

A. Simulation-based (Monte Carlo) inference: we know the exact/true data-generating mechanism. Instead of taking integrals analytically which is typically impossible for nonlinear functions of the sample, we replace those with Monte Carlo integrals (expressed as appropriate sample means).

B. Parametric bootstrap (discussed in class): We do not know the exact data-generating mechanism, but we make distributional assumptions, then estimate distribution-specific parameters. Then we use this approximate mechanism in place of the true mechanism (A) and make inference as in A.

C. Nonparametric bootstrap (discussed in class): We do not make distributional assumptions about our data. Instead, we (implicitly) estimate the distribution of the data using the empirical (cumulative) distribution function (ECDF), and generate samples using the ECDF (which is equivalent to resampling our original sample with replacement.)

Consider players A, B and C that will do inference as discussed in A, B and C above, respectively. Specifically,

- Player A always knows the exact distribution from which the data came from, and can simulate new independent samples/datasets from it.
- Player B guesses the parametric family for the distribution of the data (always as the univariate normal), estimates the parameters using the original sample (here, using the sample mean $\bar{X}$ and sample variance S^2), and then simulates new bootstrap samples from $Normal(\mu = \bar{X}, \sigma^2 = S^2)$ distribution.
- Player C does not make parametric assumptions, and instead resamples the original dataset as discussed in class; (equivalently, uses the empirical cdf to estimate the distribution of the data).

Suppose the original sample comes from distribution G. We shall estimate the variance and sampling distribution (via a histogram) of the sample variance estimator under different distributional assumptions on the original data.

For each of the three cases below (after this list of steps), complete the following steps:

(a) Simulate 1000 datasets (with n=100 for each dataset) from G. For ith dataset (in the ith row of your matrix of dataset replications), compute the sample variance and save those in a vector. Then compute the sample mean and variance of your vector of sample variances (report these). Also report the histogram based on your vector of sample variances. This is our "golden standard": the best estimates of "truth"/baseline.

(b) Estimate mean and variance using the sample mean $\bar{X}$ and sample variance S^2 from the "observed" dataset (specified below). Simulate 1000 datasets (with n=100 for each dataset) from $Normal(\mu = \bar{X}, \sigma^2 = S^2)$. For each dataset, compute the sample variance and save those in a vector. Then compute the sample mean and variance of your vector of sample variances (report these). Also report the histogram based on your vector of sample variances.

(c) Use resampling to generate 1000 datasets (with n=100 for each dataset) from the observed sample generated from G. For each dataset, compute the sample variance and save those in a vector. Then compute the sample mean and variance of your vector of sample variances (report these). Also report the histogram based on your vector of sample variances.

(d) Carefully discuss your findings; specifically, accuracy (bias and variance) and merits/drawbacks of parametric vs nonparametric bootstrap. You can treat the results of the Monte Carlo-based inference (under simulation from the true distribution) as the golden standard, against which the parametric and nonparametric bootstrap results are compared.

(e) (Optional/Bonus [5 pts]:) Investigate the accuracy of the Monte Carlo approach (under exact simulation), parametric and nonparametric bootstrap as n increases (i.e., redo Cases 1-3 for n=10, 25, 50, 100, 200, 400); discuss your findings. A suggestion: to make your answers less noisy, set random seed and generate a single dataset of size $n = 400$; e.g., call it data400. Then use `origData = data400[1:n]` as your original dataset for a given value of n.

Case 1: G is normal with mean 0 and variance 1. (I.e., B's guess about the family is correct)

Case 2: G is Student's t with 3 degrees of freedom. (I.e., B's guess about the family is very wrong.)

Case 3: G is Student's t with 25 degrees of freedom. (I.e., B's guess about the family is "somewhat" wrong but not entirely unreasonable.)

In each case, use `set.seed(0)` (once) to generate $n = 100$ independent observations from the specified distribution; this is your observed sample/dataset that you will use to estimate μ and σ^2 in (b) and resample in (c). Refer to "guidelines to specific problems' ' in the beginning of the prelim (right before the problems) for additional details.

Problem 2: exploring the wrong way to do cross-validation for estimating out-of-sample misclassification error rate (ch.05 slides, pp.26-32).

Investigate the effect of the high dimensionality of features (p) on the out-of-sample misclassification rate when variable prescreening is allowed to "see" the entire data. Perform the experiment below and briefly discuss your results.

Let $n = 25$ be the number of observations per group in a two group classification problem (e.g., cases (1) and controls (0)); let `Y = c(rep(1,25),rep(0,25))`. For $p = 200 \cdot 2^i$ ($i = 0, 1, ..., 5$), simulate features as iid $Normal(0, 1)$, so that there is no "group effect" (marginal distributions of each feature are the same for cases and controls). For each feature, perform a two-sample t-test of the equality of the means ($H_0 : \mu_0 = \mu_1$); keep at most p^* features that have the lowest p-values under 0.05 (use $p^* = 5$, 10, 20 and 40). I.e., if n_t tests reject the null hypothesis (that the means are equal) at the level of significance 0.05, use $p* \leq n_t$ features with the lowest p-values. If $p^* > n_t$, then use all n_t features with p-values under 0.05.

Report in a table the average ten-fold CV misclassification rate for a logistic regression classifier (with the intercept and linear terms only); the table columns should correspond to the values of p^* and rows - to the values of i (equivalently, p). Briefly discuss your findings.

A couple of suggestions:

1. To obtain folds for the K-fold cross-validation, reuse the function `myCVids` provided and illustrated above.

2. When simulating the data, in order to reduce the amount of variability in the results due to different random seeds and to ensure that the predictors for smaller p are nested within those for larger p, generate all predictors at once as

```
set.seed(0); nr = 50; nc = 200*2^5; # nc = 6400
M = matrix(rnorm(nr*nc),nrow=50);
Y = c(rep(1,25),rep(0,25)); # same across n
```

and use `X = M[,1:p]` as the design matrix for the problem with p covariates.

A remark: here, the choice of p^* serves as a means to variable selection after prescreening. Variable selection is a subject of Ch.06. We will likely revisit this problem after Ch.06 has been covered. E.g., for each p, retain p^* candidate predictors, and use those as input for more formal variable selection using tools from Ch.06.

Problem 3: Exploring the right way to do cross-validation for estimating out-of-sample misclassification error rate (ch.05 slides, pp.26-32).

Revise your solution to Problem 2 in order to correctly apply cross-validation in order to estimate the misclassification error rate. Specifically, carry out prescreening for every training fold separately.

The rest of the setup (10-fold CV) and the statistical test for prescreening should be the same.

Report your results as a table (with the same layout as in Problem 2) for the same logistic regression classifier and briefly discuss your findings.

Problem 4: Estimating the tuning parameters (degree of the polynomial) and test MSE by K-fold cross-validation.

Suppose Nature generates training data ($n = 200$) and test data ($m = 400$) as follows:

```
genData <- function(n, seed=0) {
  set.seed(seed)
  x = seq(-1,1,length.out=n)
  y = x - x^2 + 2*rnorm(n) # true sigma = 2;
  out.df = data.frame(x=x, y=y)
  out.df
}
train.df = genData(n=200,seed=100)
test.df  = genData(400)
```

Consider polynomial regression models $M(d) : y_i = \sum_{j=0}^{d} \beta_j x_i^j + \epsilon_i$ for $d = 0, \ldots, 4$. The goals of this problem are to use CV (i) to select a suitable model $M(d)$ (equivalently, the degree d) and (ii) to estimate the out-of-sample MSE.

Automate your code for subproblems 1-3 as much as possible. Ideally, given the training set, test set, and the CV subsets, your code should run without human intervention. (This will lead to a major reduction of the amount of work for subproblem 4.)

1. Use K-fold cross-validation (with $K = 5$) on the original training data to determine the optimal value of the tuning parameter d. To this end,

   (a) Partition the set of the training data into K disjoint groups using the above function as `inds.part = myCVids(n=200,K=5)`. The indices for the kth validation and training folds can be extracted as in the example above following the `myCVids()` definition.

   (b) For $k = 1, \ldots, K$, use the kth training fold to fit models $M(d)$ for $d = 0, \ldots, 4$. Specifically, for each k and d, fit model $M(d)$, use it to predict the response y on the left out validation fold and compute the MSE. Store the MSE values in a $(d + 1) \times K$ matrix, call it M; rows correspond to the values of d and columns correspond to the validation folds.

   (c) For each $d = 0, \ldots, 4$, aggregate the MSE's across the K validation folds and select the "best" value of d.

Report: your matrix M of MSE values by validation fold, the tally (the aggregated MSE as the last column) and the best choice of d.

2. Use the LOOCV on the training data to estimate the out-of-sample MSE using $M(d)$ (for every d) and to choose the "best" value of d. Report: the vector of LOOCV estimates of test MSE and the best value of d.

3. On the same plot, visualize your MSE estimates (from the training data) based on the K-fold CV and the LOOCV and best choices of d with respect to each approach.

   Add to the plot the values of the training and test MSEs when the entire training data set is used to fit models $M(d)$, for $d = 0, \ldots, 4$. Discuss your findings.

4. Suppose a bigger training dataset from the same true model is available, generated as `train.df = genData(n=400,seed=100)`, but the test set is the same. Rerun your code for subproblems 1-3 and discuss your findings.

Problem 5: [16 pts]

Choosing the optimal classifier under group disbalance and different error costs in binary classification.

In this problem, the goal is to determine the optimal classifier by minimizing the objective function for the overall cost of misclassification: $G(x) = c_{FN} \cdot FN(x) + c_{FP} \cdot FP(x)$, where x is the FPR. (Please refer to Tables 4.6 and 4.7 of the ISLR book for notation and acronyms.) For convenience, you can assume $n = 100$, $P = n \cdot q$, $N = n \cdot (1 - q)$, where q is the proportion of class 1 in the population (so that $(1 - q)$ is the proportion of class 0). Allow non-integer values of FP and FN.

Consider the following three "design variables" for our experiment:

A - Populations:

A1: A balanced population, $q = 0.5$ (proportion of class 1)

A2: Unbalanced population, $q = 0.2$ (proportion of class 1).

C - Relative costs of FP and FN:

C1: $c_{FN} = c_{FP}$

C2: $c_{FN} = 10 \cdot c_{FP}$ (e.g., a serious disease)

R - ROC curves for the classifiers:

R1: A classifier with a ROC curve $TPR(x) = x$, where x is FPR.

R2: A classifier with a ROC curve $TPR(x) = \sqrt{x}$, where x is FPR.

Here, a setup is a combination of the form (A,C,R), where each design variable can take one of the two values. For each of the 8 setups, determine the optimal classifier (state FPR and TPR) that minimizes $G(\cdot)$ with respect to x. Justify your choices by showing all relevant details. Some of the setups may require writing a simple program/"calculator".

Problem 6 [4 pts + 5 bonus pts]: Distribution of the number of unique samples in bootstrap.

Suppose the original dataset has n rows. Use sampling with replacement to obtain the sampling distribution (a histogram) for the number of unique samples.

Specifically, let $m = 1000$ be the number of replications, each replication being sampling n times independently with replacement from the set $\mathcal{I} = \{1, 2, \ldots, n\}$ (use function `sample(...,replace=TRUE)`) and counting the number of unique values (use function `unique` to get unique values); for the ith replication, call this rv U_i.

For $n = 25, 50, 100, 200, 400, 800$, run `set.seed(0)`, sample $\mathcal{I}$ and create a histogram based on your sequence $U_1, U_2, \ldots, U_m$. Arrange the plots in a 3x2 panel (or 2x3 depending on your layout). Additionally, report in a table the values of n (row 1), the corresponding means (row 2) and standard deviations (row 3) of the U_is. Report your results with the accuracy of 2 digits after the decimal place.

Discuss your findings in view of the results from class (about the average numbers of unique resamples in each bootstrap dataset of size n for the nonparametric bootstrap).