Module 10 Worksheet

Prepared by: Katherine Daignault and the STA302 Teaching Team

Worksheet Information

Goal of the worksheet:

The Module 10 worksheet is an opportunity to practice various validation techniques. By completing this worksheet, each student will be developing the skills to achieve the following weekly learning objectives:

- Perform appropriate model validation on a given model
- Defend the decision of whether a model has been validated using supporting evidence

This worksheet, in addition to the remainder of the class time, is **important practice for completing questions on the final exam, and for your final project**.
Preparation assumed:

For hybrid sections: As part of the flipped design of the course, it is assumed that each student is attending this lecture having completed the following pre-class preparation:

- Watched this week's module videos, attempted the Pre-Class Quiz, and accessed the code provided in the Guided Practice

For in person sections:

- Please complete this worksheet after attending your in-person lectures for each week. If you did not attend class, please review the annotated slides posted on Quercus which will be posted on each week's Quercus Page.
- Additional R/Coding resources are linked here.

How to complete this worksheet:

- Students may work in groups of 2-3 if desired. However **each student** must submit their worksheet to MarkUs to receive their completion credit. It is recommended that each student work on their **own copy** of the assignment.
- All the code and course knowledge needed to complete this worksheet has been provided in the pre-class materials. It may help to have these open while working on this document.
- Follow the instructions provided in each question to complete the code.
- DO NOT change the names of the variables that store your final answers.
- When in doubt about a question in the worksheet or your code, ask a TA or the instructor during office hours or on the discussion board.

Steps for submitting to MarkUs:

1. Go to MarkUs and log in using your UofT credentials.
2. Select this worksheet from the assignment list.

3. Under Submissions, upload your Rmd file and select the file name from the list.
4. Go to Automated Testing and select Run Tests to check your worksheet answers.
5. You can submit as many times as you want, but only your latest submission before the deadline will be counted.
6. It is recommended you submit your file to MarkUs after you complete each activity to check your answers before moving on. You can submit multiple times to check your work, as your autograding tokens regenerate over time.

What to do if a test fails on MarkUs

1. Don't panic. Your work won't really be graded until the deadline, so start early to make sure you have lots of time to resolve issues before the deadline.
2. Read the message to get hints about what the problem is. For example "variable X not present" means that you may have a typo in your variable name.
3. Double check the instructions for each question to ensure you are entering an answer in the correct format.
4. Search on the discussion board to see if other classmates have encountered a similar error (and if not, consider posting a screenshot of the error message).
5. Come to TA or instructor office hours with your issue.

The due date for MarkUs Worksheet 10 is Tuesday, November 25, at 11:59pm

This week, we will be working with a dataset that comes with lots of interesting problems. The data `pgatour2006.csv` has been loaded to JupyterHub with your RMD worksheet. It contains the variables:

- Y, PrizeMoney = average prize money per tournament
- X_1, Driving Accuracy is the percent of time a player is able to hit the fairway with his tee shot.
- X_2, GIR, Greens in Regulation, is the percent of time a player was able to hit the green in regulation. A green is considered hit in regulation if any part of the ball is touching the putting surface and the number of strokes taken is two or less than par.
- X_3, Putting Average measures putting performance on those holes where the gree is hit in regulation (GIR).
- X_4, Birdie Conversion % is the percent of time a player makes a birdie or better after hitting the green in regulation
- X_5, SandSaves % is the percent of time a player was able to get "up and down" once in a greenside sand bunker.
- X_6, Scrambling % is the percent of time that a player misses the green in regulation, but still makes par or better.
- X_7, PuttsPerRound is the average total number of putts per round.

We will be using this dataset to practice validating our model using multiple techniques.

1) Validation using training and test datasets

Let's validate the 3-predictor model we found using all possible subsets in Module 9 Worksheet. First, load the `pga_train.csv` and `pga_test.csv` datasets. These have already been split for you from the original `pgatour2006.csv` dataset.

```r
# load the data
train1 <- read.csv(file="pga_train.csv", header=T)
test1 <- read.csv(file="pga_test.csv", header=T)
```

Now that you have your training and test sets, build your 3-predictor model in the training dataset. This model uses `lnPrize` as the response and `GIR`, `BirdieConversion`, and `Scrambling` as predictors. Store the model from the training dataset in `mod_train1`. Then fit the same model in the test dataset and store it in `mod_test1`.

```r
# fit model in training dataset
mod_train1 <- NULL


# fit model in test dataset
mod_test1 <- NULL
```

Now compute the mean square error for the model fit in the training and test datasets. To do this, use the training model `mod_train1` to predict responses for data in each of `train1` and `test1`. Store these in `pred_train1` and `pred_test1` respectively. Then compute the MSE using responses from `train1` and then `test1` to get the `mse_train1` and `mse_test1` respectively.

```r
# compute fitted values for the training data
pred_train1 <- NULL

# compute fitted values for the test data
pred_test1 <- NULL

# compute mse in training data
mse_train1 <- NULL

# compute mse in test data
mse_test1 <- NULL
```

Question: Do the MSE indicate the model is validated? Why or why not? Which dataset yields better predictions and why?
TYPE YOUR ANSWER BELOW:

2) Compare the datasets

We can try to understand our results by looking at the descriptive statistics of both the training and test datasets. We can do this most easily using the `summary()` function which simply requires the dataset you want summarized as input.

Produce descriptive statistics for `train1` and `test1` using `describe()`.

```r
# produce descriptive statistics for train1


# produce descriptive statistics for test1
```

Question: Are there any large differences between the two dataset? If so, what are they and would this explain the results of your validation earlier?
TYPE YOUR ANSWER BELOW:

We can also check if problematic points contribute to our conclusions. Run the code below to check for the number of problematic points of each type (except the betas) in both datasets.

```r
# number of problematic points in the training set
cbind(length(which(hatvalues(mod_train1)>2*4/nrow(train1))),
      length(which(abs(rstandard(mod_train1))>4)),
      length(which(cooks.distance(mod_train1)>qf(0.5, 3, 94))),
      length(which(abs(dffits(mod_train1))>2*sqrt(4/nrow(train1)))))

# number of problematic points in test set
cbind(length(which(hatvalues(mod_test1)>2*4/nrow(test1))),
      length(which(abs(rstandard(mod_test1))>4)),
      length(which(cooks.distance(mod_test1)>qf(0.5, 3, 94))),
      length(which(abs(dffits(mod_test1))>2*sqrt(4/nrow(test1)))))
```

Question: Are there any large differences between the number of problematic observations?
TYPE YOUR ANSWER BELOW:

3) Leave-one-out cross validation

We can try to validate the model using leave-one-out cross validation. This requires us to run a for-loop that will remove one observation at a time from the model fit and use this model to predict the deleted observation. By doing this for each observation, we get another MSE that we can use to decide if the model is validated that is less reliant on the split we created earlier.

To do this, you will need to fill in the code below where each `NULL` is located (ignore the one for `se <- NULL` as this is needed to run the code). First fit the same model as above on the training set `LOOtrain`. Next, predict from the model `model` using the test dataset `LOOtest`. Finally compute the squared difference between `LOOtest$lnPrize` and `fitted`.

```r
# load the original dataset
pga <- read.csv(file="pgatour2006.csv", header=T)

# create lnPrize
pga$lnPrize <- log(pga$PrizeMoney)

# create storage for prediction errors
se <- NULL

# look through all n=nrow(data) observations
for(i in 1:nrow(pga)){
  #create training set by removing observation i
  LOOtrain <- pga[-i,]
  #create testing set containing only observation i
  LOOtest <- pga[i, ]
  #fit chosen model
  model <- NULL
  #make prediction for observation i
  fitted <- NULL

  #compute prediction error
  se_i <- NULL
  #store this with others
  se <- c(se, se_i)
}

mse <- mean(se)
```

```
## Warning in mean.default(se): argument is not numeric or logical: returning NA
```

```r
mse
```

```
## [1] NA
```

END OF WORKSHEET - BE SURE TO SUBMIT YOUR WORKSHEET ON MARKUS TO RECEIVE COMPLETION CREDIT