WK_10_HW_10

Anon Skywalker

10/28/2020

Question 14.1 Intro

The breast cancer data set breast-cancer-wisconsin.data.txt from http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/ (http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/) (description at http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29 (http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29)) has missing values.

1. Use the mean/mode imputation method to impute values for the missing data.
2. Use regression to impute values for the missing data.
3. Use regression with perturbation to impute values for the missing data.
4. *(Optional)* Compare the results and quality of classification models (e.g., SVM, KNN) build using

1. the data sets from questions 1,2,3;
2. the data that remains after data points with missing values are removed; and (3) the data set when a binary variable is introduced to indicate missing values

To begin with question one, we need to establish a better understanding of the data and where missing data is occurring. To do this, we can view the data set and see that the "?" symbol is used to define where missing data is instead of NA. We can search across all columns for question marks and find that only column V7 is missing values. By searching for the question marks in column V7, we can discover that there are 16 total missing values out of 699 rows in the dataset. Next, we must determine if imputation is necessary to perform given the knowledge that 16 data points are missing. As 16/699 is about *2.29%* (less than 5% standard), we can confirm that it is safe to implement mean/mode imputation. However, one other category we should check is the bias that can exist within the missing values, as sometimes there is a pattern that can be represented by missing data. An example of this is when wealthy clients prefer not to provide their income data whenever possible more so than clients with lower incomes. In the example situation, we would not want to lose that data and inserting a mean/mode would essentially throw out the fact that these were wealthier clients that could have provided a useful sample for that wealth class. To check this, I used table() to verify the distribution of 2s and 4s (the predictions) for the result column V11 for the complete data set, the missing data set (only missing data rows), and the clean data set (missing values removed). As the code shows below, the distribution is swayed for the missing data set, as 87.5% of the data seems to predict a value of 2. Despite this known factor, we will continue with the analysis because it is assigned as HW and because the missing rows only add up to 2.29% of the data.

```
#find which columns have the missing values:
bcancer[which(bcancer$V7 == "?"), ]
```

```
##          V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 24  1057013  8  4  5  1  2  ?  7  3   1   4
## 41  1096800  6  6  6  9  6  ?  7  8   1   2
## 140 1183246  1  1  1  1  1  ?  2  1   1   2
## 146 1184840  1  1  3  1  2  ?  2  1   1   2
## 159 1193683  1  1  2  1  3  ?  1  1   1   2
## 165 1197510  5  1  1  1  2  ?  3  1   1   2
## 236 1241232  3  1  4  1  2  ?  3  1   1   2
## 250  169356  3  1  1  1  2  ?  3  1   1   2
## 276  432809  3  1  3  1  2  ?  2  1   1   2
## 293  563649  8  8  8  1  2  ?  6 10   1   4
## 295  606140  1  1  1  1  2  ?  2  1   1   2
## 298   61634  5  4  3  1  2  ?  2  3   1   2
## 316  704168  4  6  5  6  7  ?  4  9   1   2
## 322  733639  3  1  1  1  2  ?  3  1   1   2
## 412 1238464  1  1  1  1  1  ?  2  1   1   2
## 618 1057067  1  1  1  1  1  ?  1  1   1   2
```

```
missing <- which(bcancer$V7 == "?")
#less than 5%! imputation time!
percent.missing <- paste( round(16/ nrow(bcancer), 4) * 100, "%" )
percent.missing
```

```
## [1] "2.29 %"
```

```
#Use colsums to find number of ? (missing values) per column
colSums(bcancer == "?")
```

```
##  V1  V2  V3  V4  V5  V6  V7  V8  V9 V10 V11
##   0   0   0   0   0   0  16   0   0   0   0
```

```
bcancer[bcancer$V7== "?",]
```

```
##          V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 24  1057013  8  4  5  1  2  ?  7  3   1   4
## 41  1096800  6  6  6  9  6  ?  7  8   1   2
## 140 1183246  1  1  1  1  1  ?  2  1   1   2
## 146 1184840  1  1  3  1  2  ?  2  1   1   2
## 159 1193683  1  1  2  1  3  ?  1  1   1   2
## 165 1197510  5  1  1  1  2  ?  3  1   1   2
## 236 1241232  3  1  4  1  2  ?  3  1   1   2
## 250  169356  3  1  1  1  2  ?  3  1   1   2
## 276  432809  3  1  3  1  2  ?  2  1   1   2
## 293  563649  8  8  8  1  2  ?  6 10   1   4
## 295  606140  1  1  1  1  2  ?  2  1   1   2
## 298   61634  5  4  3  1  2  ?  2  3   1   2
## 316  704168  4  6  5  6  7  ?  4  9   1   2
## 322  733639  3  1  1  1  2  ?  3  1   1   2
## 412 1238464  1  1  1  1  1  ?  2  1   1   2
## 618 1057067  1  1  1  1  1  ?  1  1   1   2
```

```
nrow(bcancer[which(bcancer$V7 == "?"),])
```

```
## [1] 16
```

```
missing.rows <- c(24 , 41, 140 ,146 ,159, 165, 236, 250,
                  276, 293, 295, 298, 316, 322, 412, 618)

clean.bcancer <- bcancer[-missing, ]

miss.bcancer <- bcancer[missing,]

#see the distribution of result types in V11 result column
table(bcancer$V11)
```

```
##
##   2   4
## 458 241
```

```
#check for distrubtion for clean data
table(clean.bcancer$V11)
```

```
##
##   2   4
## 444 239
```

```
#check for distrubtion for missing data
table(miss.bcancer$V11)
```

```
##
##  2  4
## 14  2
```

```
#percent of 2s in complete data
sum(bcancer$V11 == 2)/ nrow(bcancer)
```

```
## [1] 0.6552217
```

```
#percent of 2s in clean data
sum(clean.bcancer$V11 == 2)/ nrow(clean.bcancer)
```

```
## [1] 0.6500732
```

```
#percent of 2s in missing data
sum(miss.bcancer$V11 == 2)/ nrow(miss.bcancer)
```

```
## [1] 0.875
```

Question 14.1 Mean/Mode Imputation

Now that we have a better understanding of the data and where the missing values are within it, we can continue with the mean/mode imputation. For this step, I used the rows without missing values within column V7 to calculate the mean. Then I utilized a for loop to replace the missing values wherever they occurred with the mean value. See the code segment below for a detailed run-down on how this process was done and refer to the avg.cancer data set for the final version of the data with imputed mean values. For the mode imputation, R does not have a method as simple as mode() unlike the mean problem. To get around this, I utilized a simple function called getmode that can be seen below to automatically find the most common number within column V7. After finding that this number was 1, I verified by using the table() function again to find that there was a total of 402 values of 1 in the column with missing values. As this accounts for about 57% of the data, we can say with some certainty that it is safe to use the mode for this situation. After imputing the mode through the same for loop method as in the mean problem, the final dataset with mode imputations: “mode.cancer” can be seen below.

```
#create factor of the non-missing rows for col 7
nonmissing.rows <- bcancer[-missing.rows , 7 ]

#avg for col 7
avg <- round(mean(as.numeric(as.character(nonmissing.rows))), 0)

#takes a vector "data" and a value x to replace missing/NA values with.
replacer <- function(data, x)
   {

   for(i in 1:699){

     if(is.na(data[i]) == TRUE)
     {

       data[i] = x
     }

     i = i + 1

   }
  return(data)
}

#fix the col to not be a factor
fix.V7 <- as.numeric(as.character(bcancer[ , 7 ]))
```

```
## Warning: NAs introduced by coercion
```

```
#protect original data by using a new name for full data set
avg.cancer <- bcancer #for avg impute

mode.cancer <- bcancer #for mode impute
#correct the data set to use avg number for col v7
avg.cancer$V7 <- replacer(fix.V7, avg)

head(avg.cancer)
```

```
##        V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 1 1000025  5  1  1  1  2  1  3  1   1   2
## 2 1002945  5  4  4  5  7 10  3  2   1   2
## 3 1015425  3  1  1  1  2  2  3  1   1   2
## 4 1016277  6  8  8  1  3  4  3  7   1   2
## 5 1017023  4  1  1  3  2  1  3  1   1   2
## 6 1017122  8 10 10  8  7 10  9  7   1   4
```

```
#verify that all NAs have been replaced with mean value.
#integer(0) indicates that all have been replaced!
which(is.na(avg.cancer$V7)==TRUE)
```

```
## integer(0)
```

```
#get the mode!

getmode <- function(V) {
  unique.v <- unique(V)
  unique.v[which.max(tabulate(match(V, unique.v)))]


}

mode <- getmode(fix.V7)

#verify if 1 is the most common. 402 to 281 is a clear indicator that 1 is the most commonly occ
uring number.
table(fix.V7 == 1)
```

```
##
## FALSE  TRUE
##   281   402
```

```
mode.cancer$V7 <- replacer(fix.V7, 1)
head(mode.cancer)
```

```
##        V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 1 1000025  5  1  1  1  2  1  3  1   1   2
## 2 1002945  5  4  4  5  7 10  3  2   1   2
## 3 1015425  3  1  1  1  2  2  3  1   1   2
## 4 1016277  6  8  8  1  3  4  3  7   1   2
## 5 1017023  4  1  1  3  2  1  3  1   1   2
## 6 1017122  8 10 10  8  7 10  9  7   1   4
```

Question 14.1 Part 2: Regression Imputation

Another way of handling missing data values is to use linear regression as a method of predicting what the expected value of a missing data point would be based on other columns of factors. This is helpful for the example situation I described above as it would likely be a better estimate of income if it could use surrounding columns to find that a person's income may be higher based on zip code, insurance type, etc. To solve this problem, I began with a basic lm() linear regression model using all predictors and then checked out the model quality. From the summary of the default model, not all predictors correlate well when attempting to determine the missing values of V7. To drill down on this, I utilized the train() function from caret package and using stepwise regression "leapseq" from the leaps package to build a better model. The train() function also allows for CV to be performed simultaneously, which is the train.control method that can be seen within the function below. In the end, this provided a r-squared of about 61.8%. The stepwise model also decided on using factors V2-V5 for use in the final model. I went ahead and used this stepwise model for prediction and arrived at a list of the predicted values (after rounding to integers to match the column norm) that I could use for imputation. I then used a similar for loop method as I did previously in the mean/mode imputation to replace the missing values of column V7 with the corresponding regression imputation list and arrived at a final data set called “reg.bcancer” that can be seen in the code segment below.

```
mod.bcancer <- bcancer[-missing, -c(1,11)]



mod.bcancer$V7 <- as.integer(mod.bcancer$V7)
#create lm model

V7.model <- lm(V7~V2+V3+V4+V5+V6+V8+V9+V10, data = mod.bcancer)
summary(V7.model)
```

```
##
## Call:
## lm(formula = V7 ~ V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10, data = mod.bcancer)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.7316 -0.9426 -0.3002  0.6725  8.6998
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.616652   0.194975  -3.163  0.00163 **
## V2           0.230156   0.041691   5.521 4.83e-08 ***
## V3          -0.067980   0.076170  -0.892  0.37246
## V4           0.340442   0.073420   4.637 4.25e-06 ***
## V5           0.339705   0.045919   7.398 4.13e-13 ***
## V6           0.090392   0.062541   1.445  0.14883
## V8           0.320577   0.059047   5.429 7.91e-08 ***
## V9           0.007293   0.044486   0.164  0.86983
## V10         -0.075230   0.059331  -1.268  0.20524
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.274 on 674 degrees of freedom
## Multiple R-squared:  0.615,  Adjusted R-squared:  0.6104
## F-statistic: 134.6 on 8 and 674 DF,  p-value: < 2.2e-16
```

```
#TEST DEFAULT MODEL
AIC(V7.model)
```

```
## [1] 3071.697
```

```
BIC(V7.model)
```

```
## [1] 3116.962
```

```r
#eliminate insignificant factors based on p-values
train.control <- trainControl(method = "cv", number =10)

step.model <- train(V7~., data = mod.bcancer, method = "leapSeq",
                    tuneGrid = data.frame(nvmax = 1:8), trControl = train.control)



#r-Squared of 0.618 for 4 vars!
step.model$results
```

```
##   nvmax     RMSE  Rsquared      MAE    RMSESD RsquaredSD     MAESD
## 1     1 2.550838 0.5253576 1.767985 0.2472277 0.09882892 0.1768299
## 2     2 2.382032 0.5779084 1.605291 0.2496723 0.09630852 0.1486806
## 3     3 2.424824 0.5641615 1.652467 0.2562888 0.09846286 0.2126874
## 4     4 2.281619 0.6109026 1.535624 0.2467243 0.09337998 0.1665893
## 5     5 2.298428 0.6067909 1.543729 0.2230623 0.08643028 0.1649843
## 6     6 2.282123 0.6112039 1.534321 0.2457943 0.09220504 0.1682379
## 7     7 2.283699 0.6110433 1.530753 0.2505822 0.09351143 0.1749190
## 8     8 2.284647 0.6107471 1.537842 0.2475400 0.09308246 0.1706002
```

```r
#best # of variables = 4
step.model$bestTune
```

```
##   nvmax
## 4     4
```

```r
#best model appears to be with 6 predictors!
step.final <- step.model$finalModel
#See significant vars used: model chose  V2-V5
summary(step.model)
```

```
## Subset selection object
## 8 Variables  (and intercept)
##     Forced in Forced out
## V2      FALSE      FALSE
## V3      FALSE      FALSE
## V4      FALSE      FALSE
## V5      FALSE      FALSE
## V6      FALSE      FALSE
## V8      FALSE      FALSE
## V9      FALSE      FALSE
## V10     FALSE      FALSE
## 1 subsets of each size up to 4
## Selection Algorithm: 'sequential replacement'
##          V2  V3  V4  V5  V6  V8  V9  V10
## 1  ( 1 ) " " " " "*" " " " " " " " " " "
## 2  ( 1 ) " " " " "*" "*" " " " " " " " "
## 3  ( 1 ) " " " " "*" "*" " " "*" " " " "
## 4  ( 1 ) "*" "*" "*" "*" " " " " " " " "
```

```
#preserve orig dataset
reg.bcancer <- bcancer
#regression impute
impute.list <- round(predict(step.model, newdata = miss.bcancer), 0)


to.impute <- c(4 ,  8  , 1 ,  2  , 1  , 2  ,
               3  , 1  , 2 ,  6 ,  1 ,  3 ,
               6 ,  1 ,  1  , 1)


V7.impute <- fix.V7
x <- 1

for(i in 1:699){

  if(is.na(V7.impute[i]) == TRUE)
  {

    V7.impute[i] = to.impute[x]
    x = x + 1
  }

  i = i + 1

}
#replace unclean column with imputed values column
reg.bcancer$V7 <- V7.impute
#final data set with regression imputation
head(reg.bcancer)
```

```
##        V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 1 1000025  5  1  1  1  2  1  3  1   1   2
## 2 1002945  5  4  4  5  7 10  3  2   1   2
## 3 1015425  3  1  1  1  2  2  3  1   1   2
## 4 1016277  6  8  8  1  3  4  3  7   1   2
## 5 1017023  4  1  1  3  2  1  3  1   1   2
## 6 1017122  8 10 10  8  7 10  9  7   1   4
```

Question 14.1 part 3: Regression Imputation with Perturbation

For part 3 of the assignment, we are to use regression with perturbation as a method for replacing the missing values of column V7. To do this, I have used the same step.model as I did above for the purpose of implementing perturbation to our predicted values. By using rnorm() as a way of generating deviation, we can insert the predicted values we found previously as a substitute for the mean value and set the standard deviation to be the SD of our to.impute (the predicted values). Then all I had to do was round this to be integer values and insert them into the original data set. I remembered how to replace the values more easily by this point in the assignment so no for loop was used at this stage! To be sure that this the replacements were accurate, we can take the range() of the original clean data version of V7 and find that the normal distribution of the data resides between 1 and 9. With this knowledge, we can simply replace any values over 9 and under 1 with a value of 1 or 10 to prevent the missing numbers from being too high of an outlier.

```
set.seed(1)
#pertubate and round to integer values
V7.pert <- round(rnorm(nrow(bcancer[missing,]),
                 mean = to.impute, sd(to.impute) ), 0)

V7.pert
```

```
##  [1]  3  8 -1  6  2  0  4  3  3  5  4  4  5 -4  3  1
```

```
#preserve orig dataset
pert.bcancer <- bcancer

pert.bcancer[missing, ]$V7 <- as.integer(V7.pert)

#verify the range of the orignal data
range(clean.bcancer$V7)
```

```
## [1] "1" "9"
```

```
#implement a replacer for if the value is bigger or smaller than the
#normal range of the data
pert.bcancer$V7 <- as.integer(pert.bcancer$V7)
pert.bcancer$V7[pert.bcancer$V7 < 1] <- 1
pert.bcancer$V7[pert.bcancer$V7 > 10] <- 10

pert.bcancer[missing, ]$V7
```

```
##  [1] 3 8 1 6 2 1 4 3 3 5 4 4 5 1 3 1
```

```
head(pert.bcancer)
```

```
##        V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 1 1000025  5  1  1  1  2  1  3  1   1   2
## 2 1002945  5  4  4  5  7 10  3  2   1   2
## 3 1015425  3  1  1  1  2  2  3  1   1   2
## 4 1016277  6  8  8  1  3  4  3  7   1   2
## 5 1017023  4  1  1  3  2  1  3  1   1   2
## 6 1017122  8 10 10  8  7 10  9  7   1   4
```

14.1 Conclusion

After using the 3 different methodologies for dealing with missing variables, I feel that this data set benefited most from the regression with imputation version. The alternative mean/mode are good for some scenarios, but when dealing with cancer data, it can be a fatal mistake to assume that a missing value should be represented by the most common or the most average value. However, as this column is not the response column, it is likely fine to use the average/mode to replace due to the small number of missing values though after some research, it is a predictor that is statistically significant when determining whether or not a clump is malignant. Here is a article to refer to for that claim: Narasimha, A., Vasavi, B., & Kumar, H. M. (2013). Significance of nuclear morphometry in benign and malignant breast aspirates. International journal of applied & basic medical research, 3(1), 22–26. https://doi.org/10.4103/2229-516X.112237 (https://doi.org/10.4103/2229-516X.112237)

For the regression with perturbation, it was clear that the values that were predicted by the model were often outliers in terms of the normal range of the data. Some came out to be negative - far below the lowest values in the clean V7 original data, some values that were predicted ended up higher than the rest. Even after using the replacement code for amending this with values of 1 and 10, it is likely that these data points are not as helpful due to the number of values imputed that ended up being replaced with values of 1 and 10.

Coming back to regression imputation without perturbation, I feel that the predicted values fit well within the normal distribution of the data and would be a good representation for what the value should be given other factors. Again, as this is cancer data, this would be the approach I would use so as not to leave out any possible columns that could give a reasonable estimate for V7 and help in the future prediction of whether or not a tumor is malignant or benign.

Question 15.1 Describe a situation…

For my current job as a clinical data analyst, being able to create an optimization model for staffing requirements is something that would greatly improve patient wait times and staff overload problems. *I would need data that shows staff to patient breakdowns on a weekly staffing basis, feedback about patient wait times from the automated surveys we send out, staff surveys on work life balance/overload feedback on a quarterly basis, and estimated patient volumes*.