Assignment 2

Abdullah Ejaz

November 16, 2017

Imputation Methods

Predictive Mean Matching: This methods can be used by installing MICE package. This method is commonly used for the numerical missing values. It imputes data on a variable by variable basis. By default, linear regression is used to predict continuous missing values.Once the cycle of imputation is complete, multiple data sets are generated. We can access each and every dataset separately or either combined.

k-NN Imputation Method: k-NN is an abbreviation for the k- Nearest Neighbors. This method takes into consideration k number of neighbors of the element and then by those number of neighbors it classifies the class of the missing variable or we can say it predicts the value of the missing element.

Regression Imputation method: This method creates a regression model which predicts a value for the missing data. Regression works in a way that it predicts the value of an observed variable by analyzing other variables. Regressions generally predicts the value of a dependent variable based on some independent variables. That same idea is used in the imputation process.

Non-Parametric imputation Method: missForest is a non parametric imputation method. It can be applied to various variable types. It basically builds a random forest model for the variable to predict the missing value. It do not provide the user with the exact missing values. It tries to estimate the values using the random forest model as close as possible to the data point that it do not look impractical.

1 Predictive Mean Matching

1.1 Definition

PMM is an abbreviation for the method Predictive Mean Matching. This method is used by MICE package to impute missing values. This method is most commonly used for the numeric variables. The MICE package imputed data on a variable by variable basis by specifying an imputation model on them.

1.2 Code Snippet for PMM()

```
1
2 install.packages("missForest")
3 install.packages("hydroGOF")
```

```r
install.packages("mice")
install.packages("VIM")

library(missForest)
library(hydroGOF)
library(mice)
library(VIM)

dataset<- (iris.StringsAsFactor=FALSE)
summary(iris)

#producing 10% missing values
iris.mis<- prodNA(iris, noNA =0.1)

#check for the missing values are included in the dataset
summary(iris.mis)

#tabular form of the missing values present in each variable
md.pattern(iris.mis)

#for visualizing the dataset VIM package

mice_plot <- aggr(iris.mis, col=c('navyblue','yellow'),
                  numbers=TRUE, sortVars=TRUE,
                  labels=names(iris.mis), cex.axis=.7,
                  gap=3, ylab=c("Missing data","Pattern"))


#imputing the missing values.
imputed<- mice(iris.mis, m=5, maxit =50, method= 'pmm', seed=500)
summary(imputed)

imputed$imp$Sepal.Width
completeData<- complete(imputed,1)

#Calculating the RMSE between the imputed values and the original one.
rmse1<- rmse(completeData[,-5], iris[,-5], na.rm = TRUE)
rmse1

#Normalizing the numerical features

normalize<- function(x) {
  return((x-min(x)) / (max(x)- min(x)))
}

iris_normal<- as.data.frame(lapply(iris[,c(1,2,3,4)], normalize))
str(iris_normal)
summary(iris_normal)

completeData_normal<- as.data.frame(lapply(completeData[,c(1,2,3,4)], normalize))

str(completeData_normal)
summary(completeData_normal)

iris_train <- iris_normal[1:150,]
iris_test <- completeData_normal[1:150,]
iris_train_target <- iris[1:150, 5]
iris_test_target <- iris[1:150, 5]

```

```
require(class)

#building the model using knn
model1<- knn(train=iris_train, test= iris_test, cl=iris_train_target, k=13)

table(iris_test_target, model1)
```

1.3 Output of RMSE

1.3.1 For 2 percent missing values

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
 0.072571804  0.008164966  0.033665016  0.008164966
```

1.3.2 For 5 percent missing values

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
  0.02943920   0.06879922   0.08602325   0.05656854
```

1.3.3 For 10 percent missing values

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
  0.12409674   0.10360180   0.08793937   0.06000000
```

1.3.4 For 15 percent missing values

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
  0.19339080   0.15620499   0.14422205   0.08124038
```

1.3.5 For 20 percent missing values

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
   0.1902630    0.1559915    0.3034249    0.1751190
```

1.3.6 For 25 percent missing values

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
   0.2289105    0.2031420    0.2083267    0.1439907
```

1.4 Output for Supervised classification Error

1.4.1 For 2 percent missing values

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	49	1
virginica	0	3	47

1.4.2 For 5 percent missing values

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	49	1
virginica	0	3	47

1.4.3 For 10 percent missing values

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	49	1
virginica	0	3	47

1.4.4 For 15 percent missing values

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	50	0
virginica	0	4	46

1.4.5 For 20 percent missing values

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	47	3
virginica	0	2	48

1.4.6 For 25 percent missing values

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	49	1
virginica	0	2	48

2 Amelia II

2.1 Definition

This package was named after a female aviator Amelia Earhart. She was the first female aviator. This package performs multiple imputations to impute missing values. It works with an algorithm named EMB which makes it faster and robust to impute many variables. The imputation process works in a way that it first makes bootstrap samples from the data and then applies EMB algorithm to each sample. And then the set of estimates obtained from the processing of samples are being used to impute the missing values.

2.2 Code snippet for Amelia

```
install.packages("missForest")
install.packages("mice")
install.packages("VIM")
install.packages("Amelia")
library(mice)
library(missForest)
library(VIM)
library(Amelia)

dataset<- (iris)
summary(iris)

#producing 10% missing values for demo
iris.mis<- prodNA(iris, noNA =0.1)

#check for the missing values are included in the dataset
summary(iris.mis)

#tabular form of the missing values present in each variable
md.pattern(iris.mis)

#for visualizing the dataset
mice_plot <- aggr(iris.mis, col=c('navyblue','yellow'),
                  numbers=TRUE, sortVars=TRUE,
                  labels=names(iris.mis), cex.axis=.7,
                  gap=3, ylab=c("Missing data","Pattern"))

#imputing missingn values
amelia <- amelia(iris.mis, m=1, parallel = "multicore", noms = "Species")

#printing imputed values
completeData2<- amelia$imputations[[1]]

#If want to print individual imputed columns
amelia$imputations[[5]]$Sepal.Length

rmse2<- rmse(completeData2[,-5], iris[,-5], na.rm = TRUE)
```

```
rmse2

normalize<- function(x) {
  return((x-min(x)) / (max(x)- min(x)))
}

iris_normal<- as.data.frame(lapply(iris[,c(1,2,3,4)], normalize))
str(iris_normal)
summary(iris_normal)

completeData2_normal<- as.data.frame(lapply(completeData2[,c(1,2,3,4)], normalize))
str(completeData2_normal)
summary(completeData2_normal)

iris_train2 <- iris_normal[1:150,]
iris_test2 <- completeData2_normal[1:150,]
iris_train_target2 <- iris[1:150, 5]
iris_test_target2 <- iris[1:150, 5]

require(class)

#building the model using knn
model2<- knn(train=iris_train2, test= iris_test2, cl=iris_train_target2, k=13)

table(iris_test_target2, model2)
model2
```

2.3 Output of RMSE

2.3.1 For 2 percent missing values

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
  0.06042024   0.01646770   0.02937913   0.02277872
```

2.3.2 For 5 percent missing values

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
  0.03133159   0.04104625   0.05864277   0.05889102
```

2.3.3 For 10 percent missing values

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
  0.11800738   0.12069842   0.12499180   0.08432627
```

2.3.4 For 15 percent missing values

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
  0.16968321   0.10347346   0.12861433   0.09310484
```

2.3.5 For 20 percent missing values

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
  0.17401235   0.20193756   0.19650357   0.08856154
```

2.3.6 For 25 percent missing values

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
   0.2835795    0.2438688    0.3731690    0.1431871
```

2.4 Output for Supervised classification Error

2.4.1 For 2 percent missing values

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	48	2
virginica	0	3	47

2.4.2 For 5 percent missing values

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	48	2
virginica	0	3	47

2.4.3 For 10 percent missing values

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	47	3
virginica	0	4	46

2.4.4 For 15 percent missing values

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	44	6
virginica	0	2	48

2.4.5 For 20 percent missing values

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	44	6
virginica	0	2	48

2.4.6 For 25 percent missing values

	setosa	versicolor	virginica
setosa	49	1	0
versicolor	0	48	2
virginica	0	4	46

3 missForest

3.1 Definition

missForest is used to impute missing values particularly in the case of mixed-type data. It is a non-parametric imputation method. The algorithm is based in the random forest. missForest works in a way that it fits the random forest for each variable on the observed part and then predicts the missing values. missForest runs in iteration.

3.2 Code Snippet for missForest

```
install.packages("missForest")
library(missForest)
install.packages("hydroGOF")
library(hydroGOF)
install.packages("mice")
library(mice)
install.packages("VIM")
library(VIM)
install.packages("mi")
library(mi)

dataset<- iris

#producing 10% missing values for demo
iris.mis<- prodNA(iris, noNA =0.1)

#tabular form of the missing values present in each variable
md.pattern(iris.mis)
```

```
#impute missing values, using all parameters as default values
imputedSet <- missForest(iris.mis)
completeData3<- imputedSet$ximp

#calculating RMSE
rmse3<- rmse(completeData3[,-5], iris[,-5], na.rm = TRUE)
rmse3

#For k-NN
normalize<- function(x) {
  return((x-min(x)) / (max(x)- min(x)))
}

iris_normal<- as.data.frame(lapply(iris[,c(1,2,3,4)], normalize))
summary(iris_normal)

completeData3_normal<- as.data.frame(lapply(completeData3[,c(1,2,3,4)], normalize))
summary(completeData3_normal)

iris_train3 <- iris_normal[1:150,]
iris_test3 <- completeData3_normal[1:150,]
iris_train_target3 <- iris[1:150, 5]
iris_test_target3 <- iris[1:150, 5]

require(class)

#building the model using knn
model3<- knn(train=iris_train3, test= iris_test3, cl=iris_train_target3, k=13)

tab3 <- table(iris_test_target3, model3)
tab3
```

3.3 Output of RMSE

3.3.1 For 2 percent missing values

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
  0.00000000   0.04107446   0.01015989   0.04052156
```

3.3.2 For 5 percent missing values

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
  0.08527183   0.02810309   0.02364685   0.03691599
```

3.3.3 For 10 percent missing values

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
  0.12281496   0.11137945   0.07891013   0.07024052
```

3.3.4 For 15 percent missing values

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
  0.10135996   0.10325158   0.10648023   0.06203929
```

3.3.5 For 20 percent missing values

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
   0.2136009    0.1526757    0.1858022    0.1012917
```

3.3.6 For 25 percent missing values

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
   0.2248725    0.1417708    0.1730437    0.1036213
```

3.4 Output for Supervised Classification Error

3.4.1 For 2 percent missing values

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	49	1
virginica	0	3	47

3.4.2 For 5 percent missing values

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	48	2
virginica	0	3	47

3.4.3 For 10 percent missing values

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	48	2
virginica	0	3	47

3.4.4 For 15 percent missing value

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	49	1
virginica	0	3	47

3.4.5 For 20 percent missing values

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	48	2
virginica	0	0	50

3.4.6 For 25 percent missing values

	setosa	versicolor	virginica
setosa	49	1	0
versicolor	0	47	3
virginica	0	3	47

4 Conclusion

- As we can see from all the results of the RMSE and k-NN as the number of missing value is increasing, the percentage of error is also increasing.
- According to the RMSE results predictive Mean Matching(pmm) is the least performing among all of the above methods. It gives the most errors among all.

5 SCHEME

The problem statement was to find a scheme for producing NA values in the iris Dataset but not at random. So, the scheme which I created basically applies on the values greater than 4.5 in the Sepal length and Petal length column. Because only those two columns have values greater than 4.5cm. Suppose we have a measurement tool which can measure a maximum length only till 4.5cm. So, any petal length or sepal length above 4.5 will be unknown. So, the Dataset will have NA values in those places.

5.0.1 Code Snippet for the Scheme

```
#Scheme for producing NA values in the iris Dataset
dataset<- (iris)
summary(iris)

```

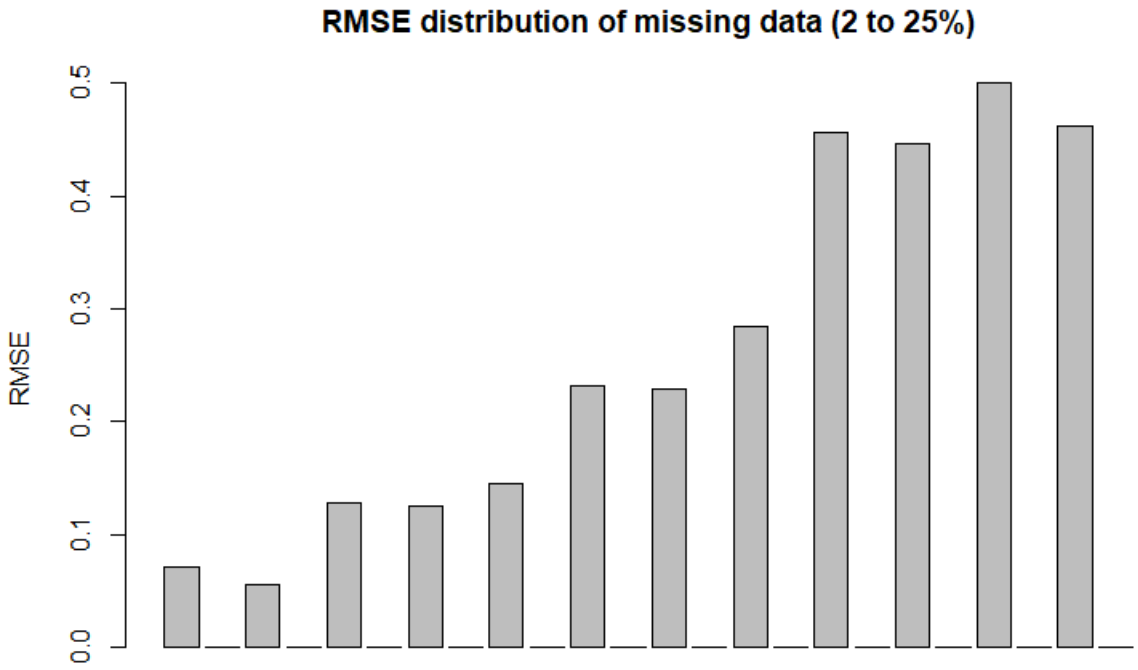
```
main <- function(x){
missing<- function(df, percentage, totalDataCount){
  allowedNAVal <- (totalDataCount * percentage)/100
  cnt <- 1
  for(i in 1:length(df$Sepal.Length)){
    if(df$Sepal.Length[i] > 4.5 &&  cnt<= allowedNAVal)
    {
      df$Sepal.Length[i]<- NA
      print(df$Sepal.Length[i])
      cnt <- cnt+1
    }
  }
  return(df)
}
mm<- missing(iris, 1, 600)

missing1<- function(df1, percentage1, totalDataCount1){
  allowedNAVal1 <- (totalDataCount1 * percentage1)/100
  cnt1 <- 1
  for(i in 1:length(df1$Petal.Length)){
    if(df1$Petal.Length[i] > 4.5 &&  cnt1<= allowedNAVal1)
    {
      df1$Petal.Length[i]<- NA
      print(df1$Petal.Length[i])
      cnt1 <- cnt1+1
    }
  }
  return(df1)
}
nn<- missing1(mm, 1, 600)
}
missing_data<- main(x)
missing_data
summary(missing_data)
```

5.1 Code snippet for scheme with PMM

```
#Scheme for producing NA values in the iris Dataset
dataset<- (iris)
summary(iris)

main <- function(x){
  missing<- function(df, percentage, totalDataCount){
    allowedNAVal <- (totalDataCount * percentage)/100
    cnt <- 1
    for(i in 1:length(df$Sepal.Length)){
      if(df$Sepal.Length[i] > 5 &&  cnt<= allowedNAVal)
      {
        df$Sepal.Length[i]<- NA
        print(df$Sepal.Length[i])
        cnt <- cnt+1
      }#if
    }#for
    return(df)
  }#fun

mm<- missing(iris, 1, 600)

missing1<- function(df1, percentage1, totalDataCount1){
```

```
    allowedNAVal1 <- (totalDataCount1 * percentage1)/100
    cnt1 <- 1
    for(i in 1:length(df1$Petal.Length)){
      if(df1$Petal.Length[i] > 4 &&  cnt1<= allowedNAVal1)
      {
        df1$Petal.Length[i]<- NA
        print(df1$Petal.Length[i])
        cnt1 <- cnt1+1
      }#if
    }#for
    return(df1)
  }

  nn<- missing1(mm, 1, 600)
  #str(nn)
  #summary(nn)
  #nn
}
missing_data<- main(x)
missing_data
summary(missing_data)

imputed_set<- mice(missing_data, m=5, maxit =10, method= 'pmm', seed=500)
summary(imputed_set)

imputedd$imp$Sepal.Width
completeData4<- complete(imputed,1)
completeData4

rmse4<- rmse(completeData4[,-5], iris[,-5], na.rm = TRUE)
rmse4

#For k-NN
normalize<- function(x) {
  return((x-min(x)) / (max(x)- min(x)))
}

iris_normal<- as.data.frame(lapply(iris[,c(1,2,3,4)], normalize))
str(iris_normal)
summary(iris_normal)

completeData4_normal<- as.data.frame(lapply(completeData4[,c(1,2,3,4)], normalize))
str(completeData4_normal)
summary(completeData4_normal)

iris_train4 <- iris_normal[1:150,]
iris_test4 <- completeData4_normal[1:150,]
iris_train_target4 <- iris[1:150, 5]
iris_test_target4 <- iris[1:150, 5]

require(class)
#building the model using knn
model4<- knn(train=iris_train4, test= iris_test4, cl=iris_train_target4, k=13)

tab4<- table(iris_test_target4, model4)
tab4
```

5.2 Output of RMSE



5.3 Output for Supervised Classification Error

Accuracy for 2 percent missing values : 0.9733
Accuracy for 5 percent missing values : 0.9667
Accuracy for 10 percent missing values : 0.9733
Accuracy for 15 percent missing values : 0.9733
Accuracy for 20 percent missing values : 1
Accuracy for 25 percent missing values : 0.9867

5.4 Code sinppet for scheme with Amelia

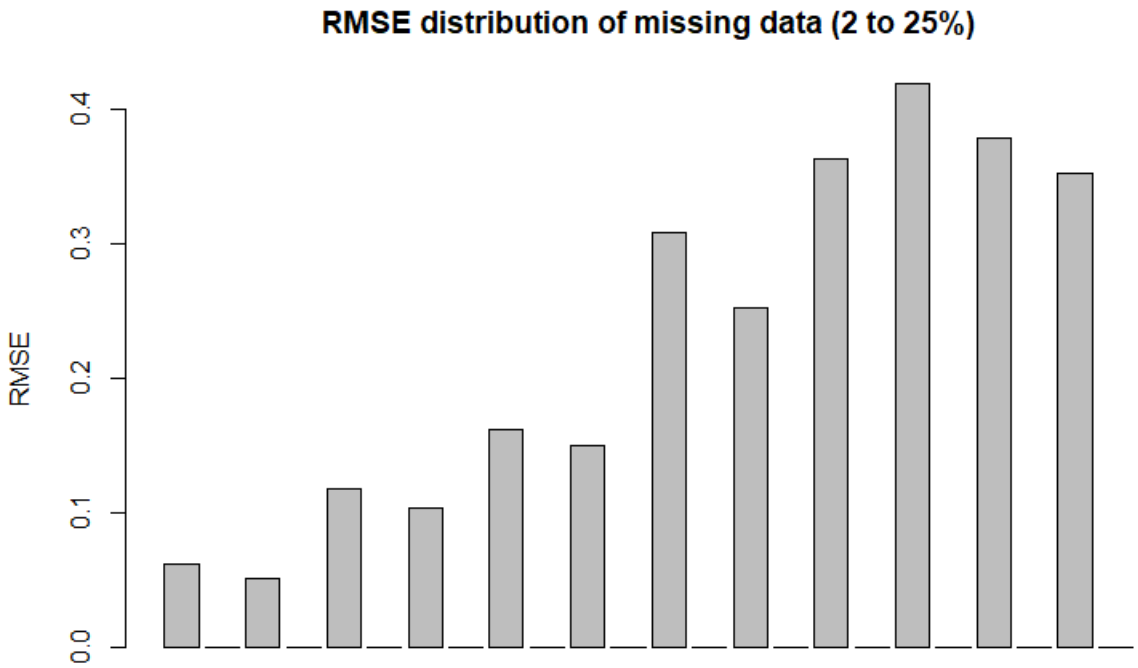
```
1
2 #Scheme for producing NA values in the iris Dataset
3 dataset<- (iris)
4 summary(iris)
5
6 missing<- function(df, percentage, totalDataCount){
7   allowedNAVal <- (totalDataCount * percentage)/100
8   cnt <- 1
9   for(i in 1:length(df$Sepal.Length)){
10    if(df$Sepal.Length[i] > 4.5 &&  cnt<= allowedNAVal)
11    {
12      df$Sepal.Length[i]<- NA
13      print(df$Sepal.Length[i])
```

```
      cnt <- cnt+1
    }#if
  }#for
  return(df)
}#fun

mm1<- missing(iris, 1, 600)
mm2<- missing(iris, 2.5, 600)
mm3<- missing(iris, 5, 600)
mm4<- missing(iris, 7.5, 600)
mm5<- missing(iris, 10, 600)
mm6<- missing(iris, 12.5, 600)

missing1<- function(df1, percentage1, totalDataCount1){
  allowedNAVal1 <- (totalDataCount1 * percentage1)/100
  cnt1 <- 1
  for(i in 1:length(df1$Petal.Length)){
    if(df1$Petal.Length[i] > 4 &&  cnt1<= allowedNAVal1)
    {
      df1$Petal.Length[i]<- NA
      print(df1$Petal.Length[i])
      cnt1 <- cnt1+1
    }#if
  }#for
  return(df1)
}
# For 2%, 5%, 10%, 15%, 20%, 25%
nn1<- missing1(mm1, 1, 600)
nn2<- missing1(mm2, 2.5, 600)
nn3<- missing1(mm3, 5, 600)
nn4<- missing1(mm4, 7.5, 600)
nn5<- missing1(mm5, 10, 600)
nn6<- missing1(mm6, 12.5, 600)

amelia_sch1 <- amelia(nn1, m=1, parallel = "multicore", noms = "Species")
amelia_sch2 <- amelia(nn2, m=1, parallel = "multicore", noms = "Species")
amelia_sch3 <- amelia(nn3, m=1, parallel = "multicore", noms = "Species")
amelia_sch4 <- amelia(nn4, m=1, parallel = "multicore", noms = "Species")
amelia_sch5 <- amelia(nn5, m=1, parallel = "multicore", noms = "Species")
amelia_sch6 <- amelia(nn6, m=1, parallel = "multicore", noms = "Species")

completeData_sch_am1<- amelia_sch1$imputations[[1]]
completeData_sch_am2<- amelia_sch2$imputations[[1]]
completeData_sch_am3<- amelia_sch3$imputations[[1]]
completeData_sch_am4<- amelia_sch4$imputations[[1]]
completeData_sch_am5<- amelia_sch5$imputations[[1]]
completeData_sch_am6<- amelia_sch6$imputations[[1]]

rmse_sch_am1<- rmse(completeData_sch_am1[,-5], iris[,-5], na.rm = TRUE)
rmse_sch_am2<- rmse(completeData_sch_am2[,-5], iris[,-5], na.rm = TRUE)
rmse_sch_am3<- rmse(completeData_sch_am3[,-5], iris[,-5], na.rm = TRUE)
rmse_sch_am4<- rmse(completeData_sch_am4[,-5], iris[,-5], na.rm = TRUE)
rmse_sch_am5<- rmse(completeData_sch_am5[,-5], iris[,-5], na.rm = TRUE)
rmse_sch_am6<- rmse(completeData_sch_am6[,-5], iris[,-5], na.rm = TRUE)

rmse_sch_am1
rmse_sch_am2
rmse_sch_am3
rmse_sch_am4
```

```
rmse_sch_am5
rmse_sch_am6

tot_rmse <- c(rmse_sch_am1, rmse_sch_am2, rmse_sch_am3, rmse_sch_am4, rmse_sch_am5, rmse_sch_am6)
per_col <- c(2,5,10,15,20,25)
rmse_df <- data.frame(percentage = per_col,error = tot_rmse);
barplot(rmse_df$error,  ylab = "RMSE", main="RMSE distribution of missing data (2 to 25%)")


#For k-NN
normalize<- function(x) {
  return((x-min(x)) / (max(x)- min(x)))
}

iris_normal<- as.data.frame(lapply(iris[,c(1,2,3,4)], normalize))

completeData_sch_am1_normal<- as.data.frame(lapply(completeData_sch_am1[,c(1,2,3,4)], normalize))
completeData_sch_am2_normal<- as.data.frame(lapply(completeData_sch_am2[,c(1,2,3,4)], normalize))
completeData_sch_am3_normal<- as.data.frame(lapply(completeData_sch_am3[,c(1,2,3,4)], normalize))
completeData_sch_am4_normal<- as.data.frame(lapply(completeData_sch_am4[,c(1,2,3,4)], normalize))
completeData_sch_am5_normal<- as.data.frame(lapply(completeData_sch_am5[,c(1,2,3,4)], normalize))
completeData_sch_am6_normal<- as.data.frame(lapply(completeData_sch_am6[,c(1,2,3,4)], normalize))


iris_train_sch_am <- iris_normal[1:150,]
iris_train_target_sch_am <- iris[1:150, 5]
iris_test_target_sch_am <- iris[1:150, 5]

iris_test_sch_am1 <- completeData_sch_am1_normal[1:150,]
iris_test_sch_am2 <- completeData_sch_am2_normal[1:150,]
iris_test_sch_am3 <- completeData_sch_am3_normal[1:150,]
iris_test_sch_am4 <- completeData_sch_am4_normal[1:150,]
iris_test_sch_am5 <- completeData_sch_am5_normal[1:150,]
iris_test_sch_am6 <- completeData_sch_am6_normal[1:150,]

require(class)

#building the model using knn
model_sch_am1<- knn(train=iris_train_sch_am, test= iris_test_sch_am1, cl=iris_train_target_sch_am, k=13)
model_sch_am2<- knn(train=iris_train_sch_am, test= iris_test_sch_am2, cl=iris_train_target_sch_am, k=13)
model_sch_am3<- knn(train=iris_train_sch_am, test= iris_test_sch_am3, cl=iris_train_target_sch_am, k=13)
model_sch_am4<- knn(train=iris_train_sch_am, test= iris_test_sch_am4, cl=iris_train_target_sch_am, k=13)
model_sch_am5<- knn(train=iris_train_sch_am, test= iris_test_sch_am5, cl=iris_train_target_sch_am, k=13)
model_sch_am6<- knn(train=iris_train_sch_am, test= iris_test_sch_am6, cl=iris_train_target_sch_am, k=13)

```

```
tab_sch_am1<- table(iris_test_target_sch_am, model_sch_am1)
tab_sch_am2<- table(iris_test_target_sch_am, model_sch_am2)
tab_sch_am3<- table(iris_test_target_sch_am, model_sch_am3)
tab_sch_am4<- table(iris_test_target_sch_am, model_sch_am4)
tab_sch_am5<- table(iris_test_target_sch_am, model_sch_am5)
tab_sch_am6<- table(iris_test_target_sch_am, model_sch_am6)

tab_sch_am1
tab_sch_am2
tab_sch_am3
tab_sch_am4
tab_sch_am5
tab_sch_am6

confusionMatrix(tab_sch_am1)
confusionMatrix(tab_sch_am2)
confusionMatrix(tab_sch_am3)
confusionMatrix(tab_sch_am4)
confusionMatrix(tab_sch_am5)
confusionMatrix(tab_sch_am6)
```

5.5 Output of RMSE



5.6 Output for Supervised Classification Error

Accuracy for 2 percent missing values : 0.9733
Accuracy for 5 percent missing values : 0.9733
Accuracy for 10 percent missing values : 0.98

Accuracy for 15 percent missing values : 0.9867
Accuracy for 20 percent missing values : 1
Accuracy for 25 percent missing values : 0.9733

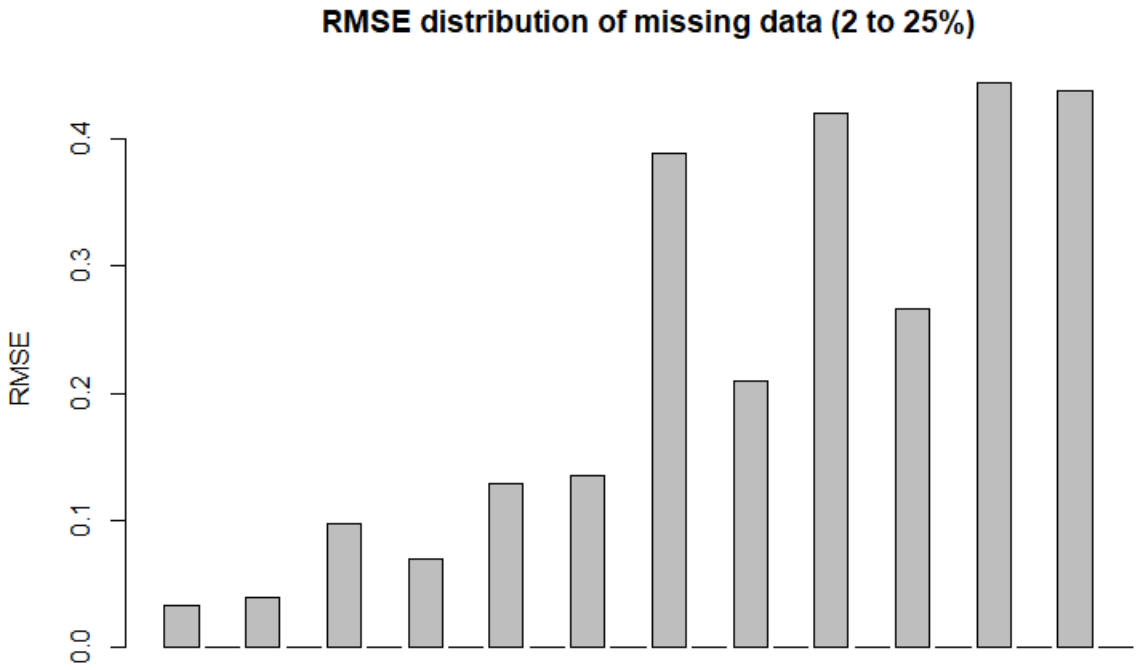
5.7 Code Snippet for Scheme with missForest

```
#Scheme for producing NA values in the iris Dataset
dataset<- (iris)
summary(iris)

missing<- function(df, percentage, totalDataCount){
  allowedNAVal <- (totalDataCount * percentage)/100
  cnt <- 1
  for(i in 1:length(df$Sepal.Length)){
    if(df$Sepal.Length[i] > 4.5 &&  cnt<= allowedNAVal)
    {
      df$Sepal.Length[i]<- NA
      print(df$Sepal.Length[i])
      cnt <- cnt+1
    }#if
  }#for
  return(df)
}#fun

mm1<- missing(iris, 1, 600)
mm2<- missing(iris, 2.5, 600)
mm3<- missing(iris, 5, 600)
mm4<- missing(iris, 7.5, 600)
mm5<- missing(iris, 10, 600)
mm6<- missing(iris, 12.5, 600)

missing1<- function(df1, percentage1, totalDataCount1){
  allowedNAVal1 <- (totalDataCount1 * percentage1)/100
  cnt1 <- 1
  for(i in 1:length(df1$Petal.Length)){
    if(df1$Petal.Length[i] > 4 &&  cnt1<= allowedNAVal1)
    {
      df1$Petal.Length[i]<- NA
      print(df1$Petal.Length[i])
      cnt1 <- cnt1+1
    }#if
  }#for
  return(df1)
}

nn1<- missing1(mm1, 1, 600)
nn2<- missing1(mm2, 2.5, 600)
nn3<- missing1(mm3, 5, 600)
nn4<- missing1(mm4, 7.5, 600)
nn5<- missing1(mm5, 10, 600)
nn6<- missing1(mm6, 12.5, 600)

#imputing missing values using missForest

imputed_sch_mf1 <- missForest(nn1)
imputed_sch_mf2 <- missForest(nn2)
imputed_sch_mf3 <- missForest(nn3)
```

```
imputed_sch_mf4 <- missForest(nn4)
imputed_sch_mf5 <- missForest(nn5)
imputed_sch_mf6 <- missForest(nn6)

completeData_sch_mf1<- imputed_sch_mf1$ximp
completeData_sch_mf2<- imputed_sch_mf2$ximp
completeData_sch_mf3<- imputed_sch_mf3$ximp
completeData_sch_mf4<- imputed_sch_mf4$ximp
completeData_sch_mf5<- imputed_sch_mf5$ximp
completeData_sch_mf6<- imputed_sch_mf6$ximp

rmse_sch_mf1<- rmse(completeData_sch_mf1[,-5], iris[,-5], na.rm = TRUE)
rmse_sch_mf2<- rmse(completeData_sch_mf2[,-5], iris[,-5], na.rm = TRUE)
rmse_sch_mf3<- rmse(completeData_sch_mf3[,-5], iris[,-5], na.rm = TRUE)
rmse_sch_mf4<- rmse(completeData_sch_mf4[,-5], iris[,-5], na.rm = TRUE)
rmse_sch_mf5<- rmse(completeData_sch_mf5[,-5], iris[,-5], na.rm = TRUE)
rmse_sch_mf6<- rmse(completeData_sch_mf6[,-5], iris[,-5], na.rm = TRUE)

rmse_sch_mf1
rmse_sch_mf2
rmse_sch_mf3
rmse_sch_mf4
rmse_sch_mf5
rmse_sch_mf6

tot_rmse <- c(rmse_sch_mf1, rmse_sch_mf2, rmse_sch_mf3, rmse_sch_mf4, rmse_sch_mf5,
    rmse_sch_mf6)
per_col <- c(2,5,10,15,20,25)
rmse_df1 <- data.frame(percentage = per_col,error = tot_rmse);
barplot(rmse_df1$error,  ylab = "RMSE", main="RMSE distribution of missing data (2
    to 25%)")

#For k-NN
normalize<- function(x) {
  return((x-min(x)) / (max(x)- min(x)))
}

iris_normal<- as.data.frame(lapply(iris[,c(1,2,3,4)], normalize))

completeData_sch_mf1_normal<- as.data.frame(lapply(completeData_sch_mf1[,c(1,2,3,4)
    ], normalize))
completeData_sch_mf2_normal<- as.data.frame(lapply(completeData_sch_mf2[,c(1,2,3,4)
    ], normalize))
completeData_sch_mf3_normal<- as.data.frame(lapply(completeData_sch_mf3[,c(1,2,3,4)
    ], normalize))
completeData_sch_mf4_normal<- as.data.frame(lapply(completeData_sch_mf4[,c(1,2,3,4)
    ], normalize))
completeData_sch_mf5_normal<- as.data.frame(lapply(completeData_sch_mf5[,c(1,2,3,4)
    ], normalize))
completeData_sch_mf6_normal<- as.data.frame(lapply(completeData_sch_mf6[,c(1,2,3,4)
    ], normalize))

iris_train_sch_mf <- iris_normal[1:150,]
iris_train_target_sch_mf <- iris[1:150, 5]
iris_test_target_sch_mf <- iris[1:150, 5]

iris_test_sch_mf1 <- completeData_sch_mf1_normal[1:150,]
iris_test_sch_mf2 <- completeData_sch_mf2_normal[1:150,]
iris_test_sch_mf3 <- completeData_sch_mf3_normal[1:150,]
```

```
iris_test_sch_mf4 <- completeData_sch_mf4_normal[1:150,]
iris_test_sch_mf5 <- completeData_sch_mf5_normal[1:150,]
iris_test_sch_mf6 <- completeData_sch_mf6_normal[1:150,]

require(class)

#building the model using knn
model_sch_mf1<- knn(train=iris_train_sch_mf, test= iris_test_sch_mf1, cl=iris_train_target_sch_mf, k=13)
model_sch_mf2<- knn(train=iris_train_sch_mf, test= iris_test_sch_mf2, cl=iris_train_target_sch_mf, k=13)
model_sch_mf3<- knn(train=iris_train_sch_mf, test= iris_test_sch_mf3, cl=iris_train_target_sch_mf, k=13)
model_sch_mf4<- knn(train=iris_train_sch_mf, test= iris_test_sch_mf4, cl=iris_train_target_sch_mf, k=13)
model_sch_mf5<- knn(train=iris_train_sch_mf, test= iris_test_sch_mf5, cl=iris_train_target_sch_mf, k=13)
model_sch_mf6<- knn(train=iris_train_sch_mf, test= iris_test_sch_mf6, cl=iris_train_target_sch_mf, k=13)

tab_sch_mf1<- table(iris_test_target_sch_mf, model_sch_mf1)
tab_sch_mf2<- table(iris_test_target_sch_mf, model_sch_mf2)
tab_sch_mf3<- table(iris_test_target_sch_mf, model_sch_mf3)
tab_sch_mf4<- table(iris_test_target_sch_mf, model_sch_mf4)
tab_sch_mf5<- table(iris_test_target_sch_mf, model_sch_mf5)
tab_sch_mf6<- table(iris_test_target_sch_mf, model_sch_mf6)

tab_sch_mf1
tab_sch_mf2
tab_sch_mf3
tab_sch_mf4
tab_sch_mf5
tab_sch_mf6

confusionMatrix(tab_sch_mf1)
confusionMatrix(tab_sch_mf2)
confusionMatrix(tab_sch_mf3)
confusionMatrix(tab_sch_mf4)
confusionMatrix(tab_sch_mf5)
confusionMatrix(tab_sch_mf6)
```

5.8 Output of RMSE



5.9 Output for Supervised Classification Error

Accuracy for 2 percent missing values : 0.9733
Accuracy for 5 percent missing values : 0.9733
Accuracy for 10 percent missing values : 0.98
Accuracy for 15 percent missing values : 0.98
Accuracy for 20 percent missing values : 0.9933
Accuracy for 25 percent missing values : 0.9667

6 GitHub Link

`https://github.com/Abdullahejaz/Data_Imputation_R`