

Assignment 2

Abdullah Ejaz

November 16, 2017

Imputation Methods

Predictive Mean Matching: This method 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")
```

```

4 install.packages("mice")
5 install.packages("VIM")
6
7 library(missForest)
8 library(hydroGOF)
9 library(mice)
10 library(VIM)
11
12 dataset<- (iris.StringsAsFactor=FALSE)
13 summary(iris)
14
15 #producing 10% missing values
16 iris.mis<- prodNA(iris , noNA =0.1)
17
18 #check for the missing values are included in the dataset
19 summary(iris.mis)
20
21 #tabular form of the missing values present in each variable
22 md.pattern(iris.mis)
23
24 #for visualizing the dataset VIM package
25
26 mice_plot <- aggr(iris.mis, col=c('navyblue','yellow'),
27                        numbers=TRUE, sortVars=TRUE,
28                        labels=names(iris.mis), cex.axis=.7,
29                        gap=3, ylab=c("Missing data","Pattern"))
30
31
32 #imputing the missing values.
33 imputed<- mice(iris.mis, m=5, maxit =50, method= 'pmm', seed=500)
34 summary(imputed)
35
36 imputed$imp$Sepal.Width
37 completeData<- complete(imputed,1)
38
39 #Calculating the RMSE between the imputed values and the original one.
40 rmse1<- rmse(completeData[, -5], iris[, -5], na.rm = TRUE)
41 rmse1
42
43 #Normalizing the numerical features
44
45 normalize<- function(x) {
46   return((x-min(x)) / (max(x)- min(x)))
47 }
48
49 iris_normal<- as.data.frame(lapply(iris[,c(1,2,3,4)], normalize))
50 str(iris_normal)
51 summary(iris_normal)
52
53 completeData_normal<- as.data.frame(lapply(completeData[,c(1,2,3,4)], normalize))
54
55 str(completeData_normal)
56 summary(completeData_normal)
57
58 iris_train <- iris_normal[1:150,]
59 iris_test <- completeData_normal[1:150,]
60 iris_train_target <- iris[1:150, 5]
61 iris_test_target <- iris[1:150, 5]
62

```

```

63 require(class)
64
65 #building the model using knn
66 modell<- knn(train=iris_train, test= iris_test, cl=iris_train_target, k=13)
67
68 table(iris_test_target, modell)

```

1.3 Output of RMSE

1.3.1 For 2 percent missing values

```

1 Sepal.Length Sepal.Width Petal.Length Petal.Width
2 0.072571804 0.008164966 0.033665016 0.008164966

```

1.3.2 For 5 percent missing values

```

1 Sepal.Length Sepal.Width Petal.Length Petal.Width
2 0.02943920 0.06879922 0.08602325 0.05656854

```

1.3.3 For 10 percent missing values

```

1 Sepal.Length Sepal.Width Petal.Length Petal.Width
2 0.12409674 0.10360180 0.08793937 0.06000000

```

1.3.4 For 15 percent missing values

```

1 Sepal.Length Sepal.Width Petal.Length Petal.Width
2 0.19339080 0.15620499 0.14422205 0.08124038

```

1.3.5 For 20 percent missing values

```

1 Sepal.Length Sepal.Width Petal.Length Petal.Width
2 0.1902630 0.1559915 0.3034249 0.1751190

```

1.3.6 For 25 percent missing values

```

1 Sepal.Length Sepal.Width Petal.Length Petal.Width
2 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

```

1 install.packages("missForest")
2 install.packages("mice")
3 install.packages("VIM")
4 install.packages("Amelia")
5 library(mice)
6 library(missForest)
7 library(VIM)
8 library(Amelia)
9
10 dataset<- (iris)
11 summary(iris)
12
13 #producing 10% missing values for demo
14 iris.mis<- prodNA(iris , noNA =0.1)
15
16 #check for the missing values are included in the dataset
17 summary(iris.mis)
18
19 #tabular form of the missing values present in each variable
20 md.pattern(iris.mis)
21
22 #for visualizing the dataset
23 mice_plot <- aggr(iris.mis, col=c('navyblue','yellow'),
24                      numbers=TRUE, sortVars=TRUE,
25                      labels=names(iris.mis), cex.axis=.7,
26                      gap=3, ylab=c("Missing data","Pattern"))
27
28 #imputing missingn values
29 amelia <- amelia(iris.mis, m=1, parallel = "multicore", noms = "Species")
30
31 #printing imputed values
32 completeData2<- amelia$imputations[[1]]
33
34 #If want to print individual imputed columns
35 amelia$imputations[[5]]$Sepal.Length
36
37 rmse2<- rmse(completeData2[,-5], iris[,-5], na.rm = TRUE)

```

```

38 rmse2
39
40 normalize<- function(x) {
41   return((x-min(x)) / (max(x)- min(x)))
42 }
43
44 iris_normal<- as.data.frame(lapply(iris[,c(1,2,3,4)], normalize))
45 str(iris_normal)
46 summary(iris_normal)
47
48 completeData2_normal<- as.data.frame(lapply(completeData2[,c(1,2,3,4)], normalize))
49 str(completeData2_normal)
50 summary(completeData2_normal)
51
52 iris_train2 <- iris_normal[1:150,]
53 iris_test2 <- completeData2_normal[1:150,]
54 iris_train_target2 <- iris[1:150, 5]
55 iris_test_target2 <- iris[1:150, 5]
56
57 require(class)
58
59 #building the model using knn
60 model2<- knn(train=iris_train2, test= iris_test2, cl=iris_train_target2, k=13)
61
62 table(iris_test_target2, model2)
63 model2

```

2.3 Output of RMSE

2.3.1 For 2 percent missing values

1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
2	0.06042024	0.01646770	0.02937913	0.02277872

2.3.2 For 5 percent missing values

1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
2	0.03133159	0.04104625	0.05864277	0.05889102

2.3.3 For 10 percent missing values

1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
2	0.11800738	0.12069842	0.12499180	0.08432627

2.3.4 For 15 percent missing values

1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
2	0.16968321	0.10347346	0.12861433	0.09310484

2.3.5 For 20 percent missing values

1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
2	0.17401235	0.20193756	0.19650357	0.08856154

2.3.6 For 25 percent missing values

1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
2	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

```

1 install.packages("missForest")
2 library(missForest)
3 install.packages("hydroGOF")
4 library(hydroGOF)
5 install.packages("mice")
6 library(mice)
7 install.packages("VIM")
8 library(VIM)
9 install.packages("mi")
10 library(mi)
11
12 dataset<- iris
13
14 #producing 10% missing values for demo
15 iris.mis<- prodNA(iris, noNA =0.1)
16
17 #tabular form of the missing values present in each variable
18 md.pattern(iris.mis)

```



```

19
20 #impute missing values, using all parameters as default values
21 imputedSet <- missForest(iris.mis)
22 completeData3<- imputedSet$rimp
23
24 #calculating RMSE
25 rmse3<- rmse(completeData3[, -5], iris[, -5], na.rm = TRUE)
26 rmse3
27
28 #For k-NN
29 normalize<- function(x) {
30   return((x-min(x)) / (max(x)- min(x)))
31 }
32
33 iris_normal<- as.data.frame(lapply(iris[, c(1,2,3,4)], normalize))
34 summary(iris_normal)
35
36 completeData3_normal<- as.data.frame(lapply(completeData3[, c(1,2,3,4)], normalize))
37 summary(completeData3_normal)
38
39 iris_train3 <- iris_normal[1:150,]
40 iris_test3 <- completeData3_normal[1:150,]
41 iris_train_target3 <- iris[1:150, 5]
42 iris_test_target3 <- iris[1:150, 5]
43
44 require(class)
45
46 #building the model using knn
47 model3<- knn(train=iris_train3, test= iris_test3, cl=iris_train_target3, k=13)
48
49 tab3 <- table(iris_test_target3, model3)
50 tab3

```

3.3 Output of RMSE

3.3.1 For 2 percent missing values

1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
2	0.00000000	0.04107446	0.01015989	0.04052156

3.3.2 For 5 percent missing values

1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
2	0.08527183	0.02810309	0.02364685	0.03691599

3.3.3 For 10 percent missing values

1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
2	0.12281496	0.11137945	0.07891013	0.07024052

3.3.4 For 15 percent missing values

1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
2	0.10135996	0.10325158	0.10648023	0.06203929

3.3.5 For 20 percent missing values

1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
2	0.2136009	0.1526757	0.1858022	0.1012917

3.3.6 For 25 percent missing values

1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
2	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

```
1 #Scheme for producing NA values in the iris Dataset
2 dataset<- (iris)
3 summary(iris)
4
```

```

5 main <- function(x){
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])
14        cnt <- cnt+1
15      }
16    }
17    return(df)
18  }
19  mm<- missing(iris, 1, 600)
20
21  missing1<- function(df1, percentage1, totalDataCount1){
22    allowedNAVal1 <- (totalDataCount1 * percentage1)/100
23    cnt1 <- 1
24    for(i in 1:length(df1$Petal.Length)){
25      if(df1$Petal.Length[i] > 4.5 && cnt1<= allowedNAVal1)
26      {
27        df1$Petal.Length[i]<- NA
28        print(df1$Petal.Length[i])
29        cnt1 <- cnt1+1
30      }
31    }
32    return(df1)
33  }
34  nn<- missing1(mm, 1, 600)
35  }
36  missing_data<- main(x)
37  missing_data
38  summary(missing_data)

```

5.1 Code snippet for scheme with PMM

```

1 #Scheme for producing NA values in the iris Dataset
2 dataset<- (iris)
3 summary(iris)
4
5 main <- function(x){
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] > 5 && cnt<= allowedNAVal)
11      {
12        df$Sepal.Length[i]<- NA
13        print(df$Sepal.Length[i])
14        cnt <- cnt+1
15      }#if
16    }#for
17    return(df)
18  }#fun
19
20  mm<- missing(iris, 1, 600)
21
22  missing1<- function(df1, percentage1, totalDataCount1){

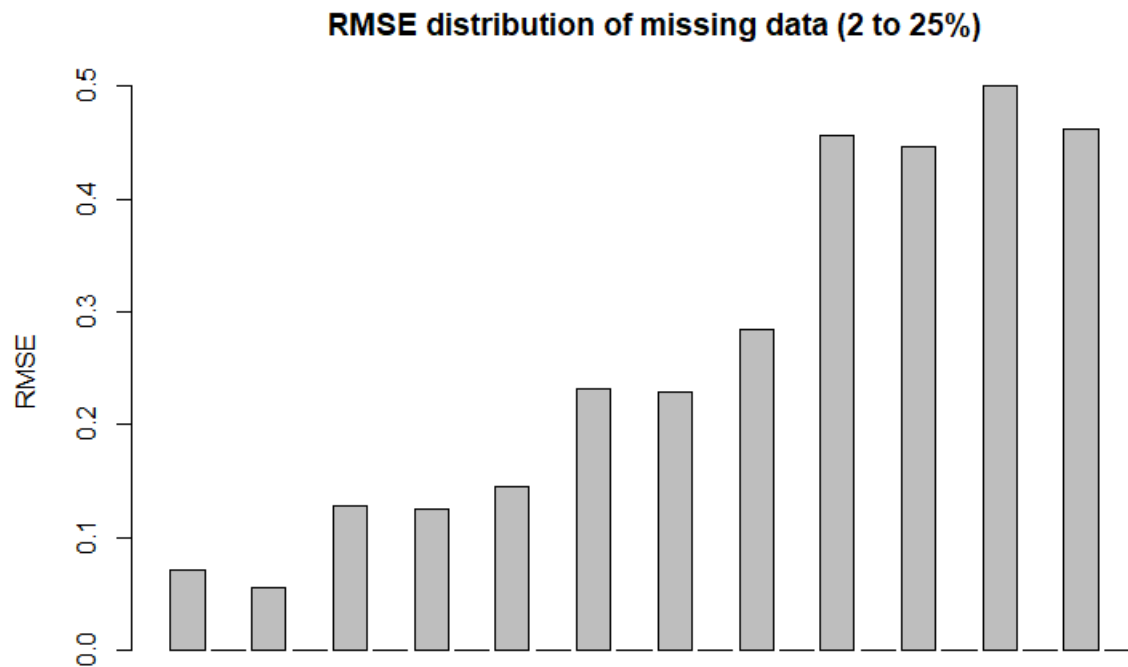
```

```

23     allowedNAVal1 <- (totalDataCount1 * percentage1)/100
24     cnt1 <- 1
25     for(i in 1:length(df1$Petal.Length)){
26         if(df1$Petal.Length[i] > 4 && cnt1<= allowedNAVal1)
27         {
28             df1$Petal.Length[i]<- NA
29             print(df1$Petal.Length[i])
30             cnt1 <- cnt1+1
31         }#if
32     }#for
33     return(df1)
34 }
35
36 nn<- missing1(mm, 1, 600)
37 #str(nn)
38 #summary(nn)
39 #nn
40 }
41 missing_data<- main(x)
42 missing_data
43 summary(missing_data)
44
45 imputed_set<- mice(missing_data, m=5, maxit =10, method= 'pmm', seed=500)
46 summary(imputed_set)
47
48 imputedd$imp$Sepal.Width
49 completeData4<- complete(imputed,1)
50 completeData4
51
52 rmse4<- rmse(completeData4[, -5], iris[, -5], na.rm = TRUE)
53 rmse4
54
55 #For k-NN
56 normalize<- function(x) {
57     return((x-min(x)) / (max(x)- min(x)))
58 }
59
60 iris_normal<- as.data.frame(lapply(iris[, c(1,2,3,4)], normalize))
61 str(iris_normal)
62 summary(iris_normal)
63
64 completeData4_normal<- as.data.frame(lapply(completeData4[, c(1,2,3,4)], normalize))
65 str(completeData4_normal)
66 summary(completeData4_normal)
67
68 iris_train4 <- iris_normal[1:150,]
69 iris_test4 <- completeData4_normal[1:150,]
70 iris_train_target4 <- iris[1:150, 5]
71 iris_test_target4 <- iris[1:150, 5]
72
73 require(class)
74 #building the model using knn
75 model4<- knn(train=iris_train4, test= iris_test4, cl=iris_train_target4, k=13)
76
77 tab4<- table(iris_test_target4, model4)
78 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])
14     }
15     cnt = cnt + 1
16   }
17 }
```

```

14     cnt <- cnt+1
15   }#if
16 }#for
17 return(df)
18 }#fun
19
20 mm1<- missing(iris , 1, 600)
21 mm2<- missing(iris , 2.5, 600)
22 mm3<- missing(iris , 5, 600)
23 mm4<- missing(iris , 7.5, 600)
24 mm5<- missing(iris , 10, 600)
25 mm6<- missing(iris , 12.5, 600)
26
27 missing1<- function(df1, percentage1, totalDataCount1){
28   allowedNAVal1 <- (totalDataCount1 * percentage1)/100
29   cnt1 <- 1
30   for(i in 1:length(df1$Petal.Length)){
31     if(df1$Petal.Length[i] > 4 && cnt1<= allowedNAVal1)
32     {
33       df1$Petal.Length[i]<- NA
34       print(df1$Petal.Length[i])
35       cnt1 <- cnt1+1
36     }#if
37   }#for
38   return(df1)
39 }
40 # For 2%, 5%, 10%, 15%, 20%, 25%
41 nn1<- missing1(mm1, 1, 600)
42 nn2<- missing1(mm2, 2.5, 600)
43 nn3<- missing1(mm3, 5, 600)
44 nn4<- missing1(mm4, 7.5, 600)
45 nn5<- missing1(mm5, 10, 600)
46 nn6<- missing1(mm6, 12.5, 600)
47
48 amelia_sch1 <- amelia(nn1, m=1, parallel = "multicore", noms = "Species")
49 amelia_sch2 <- amelia(nn2, m=1, parallel = "multicore", noms = "Species")
50 amelia_sch3 <- amelia(nn3, m=1, parallel = "multicore", noms = "Species")
51 amelia_sch4 <- amelia(nn4, m=1, parallel = "multicore", noms = "Species")
52 amelia_sch5 <- amelia(nn5, m=1, parallel = "multicore", noms = "Species")
53 amelia_sch6 <- amelia(nn6, m=1, parallel = "multicore", noms = "Species")
54
55 completeData_sch_am1<- amelia_sch1$imputations[[1]]
56 completeData_sch_am2<- amelia_sch2$imputations[[1]]
57 completeData_sch_am3<- amelia_sch3$imputations[[1]]
58 completeData_sch_am4<- amelia_sch4$imputations[[1]]
59 completeData_sch_am5<- amelia_sch5$imputations[[1]]
60 completeData_sch_am6<- amelia_sch6$imputations[[1]]
61
62 rmse_sch_am1<- rmse(completeData_sch_am1[, -5], iris[, -5], na.rm = TRUE)
63 rmse_sch_am2<- rmse(completeData_sch_am2[, -5], iris[, -5], na.rm = TRUE)
64 rmse_sch_am3<- rmse(completeData_sch_am3[, -5], iris[, -5], na.rm = TRUE)
65 rmse_sch_am4<- rmse(completeData_sch_am4[, -5], iris[, -5], na.rm = TRUE)
66 rmse_sch_am5<- rmse(completeData_sch_am5[, -5], iris[, -5], na.rm = TRUE)
67 rmse_sch_am6<- rmse(completeData_sch_am6[, -5], iris[, -5], na.rm = TRUE)
68
69 rmse_sch_am1
70 rmse_sch_am2
71 rmse_sch_am3
72 rmse_sch_am4

```

```

73 rmse_sch_am5
74 rmse_sch_am6
75
76 tot_rmse <- c(rmse_sch_am1, rmse_sch_am2, rmse_sch_am3, rmse_sch_am4, rmse_sch_am5,
  rmse_sch_am6)
77 per_col <- c(2,5,10,15,20,25)
78 rmse_df <- data.frame(percentage = per_col, error = tot_rmse);
79 barplot(rmse_df$error, ylab = "RMSE", main="RMSE distribution of missing data (2 to
  25%)")
80
81
82 #For k-NN
83 normalize<- function(x) {
84   return((x-min(x)) / (max(x)- min(x)))
85 }
86
87 iris_normal<- as.data.frame(lapply(iris[,c(1,2,3,4)], normalize))
88
89 completeData_sch_am1_normal<- as.data.frame(lapply(completeData_sch_am1[,c(1,2,3,4)
  ], normalize))
90 completeData_sch_am2_normal<- as.data.frame(lapply(completeData_sch_am2[,c(1,2,3,4)
  ], normalize))
91 completeData_sch_am3_normal<- as.data.frame(lapply(completeData_sch_am3[,c(1,2,3,4)
  ], normalize))
92 completeData_sch_am4_normal<- as.data.frame(lapply(completeData_sch_am4[,c(1,2,3,4)
  ], normalize))
93 completeData_sch_am5_normal<- as.data.frame(lapply(completeData_sch_am5[,c(1,2,3,4)
  ], normalize))
94 completeData_sch_am6_normal<- as.data.frame(lapply(completeData_sch_am6[,c(1,2,3,4)
  ], normalize))
95
96
97 iris_train_sch_am <- iris_normal[1:150,]
98 iris_train_target_sch_am <- iris[1:150, 5]
99 iris_test_target_sch_am <- iris[1:150, 5]
100
101 iris_test_sch_am1 <- completeData_sch_am1_normal[1:150,]
102 iris_test_sch_am2 <- completeData_sch_am2_normal[1:150,]
103 iris_test_sch_am3 <- completeData_sch_am3_normal[1:150,]
104 iris_test_sch_am4 <- completeData_sch_am4_normal[1:150,]
105 iris_test_sch_am5 <- completeData_sch_am5_normal[1:150,]
106 iris_test_sch_am6 <- completeData_sch_am6_normal[1:150,]
107
108 require(class)
109
110 #building the model using knn
111 model_sch_am1<- knn(train=iris_train_sch_am, test= iris_test_sch_am1, cl=iris_train_
  target_sch_am, k=13)
112 model_sch_am2<- knn(train=iris_train_sch_am, test= iris_test_sch_am2, cl=iris_train_
  target_sch_am, k=13)
113 model_sch_am3<- knn(train=iris_train_sch_am, test= iris_test_sch_am3, cl=iris_train_
  target_sch_am, k=13)
114 model_sch_am4<- knn(train=iris_train_sch_am, test= iris_test_sch_am4, cl=iris_train_
  target_sch_am, k=13)
115 model_sch_am5<- knn(train=iris_train_sch_am, test= iris_test_sch_am5, cl=iris_train_
  target_sch_am, k=13)
116 model_sch_am6<- knn(train=iris_train_sch_am, test= iris_test_sch_am6, cl=iris_train_
  target_sch_am, k=13)
117

```

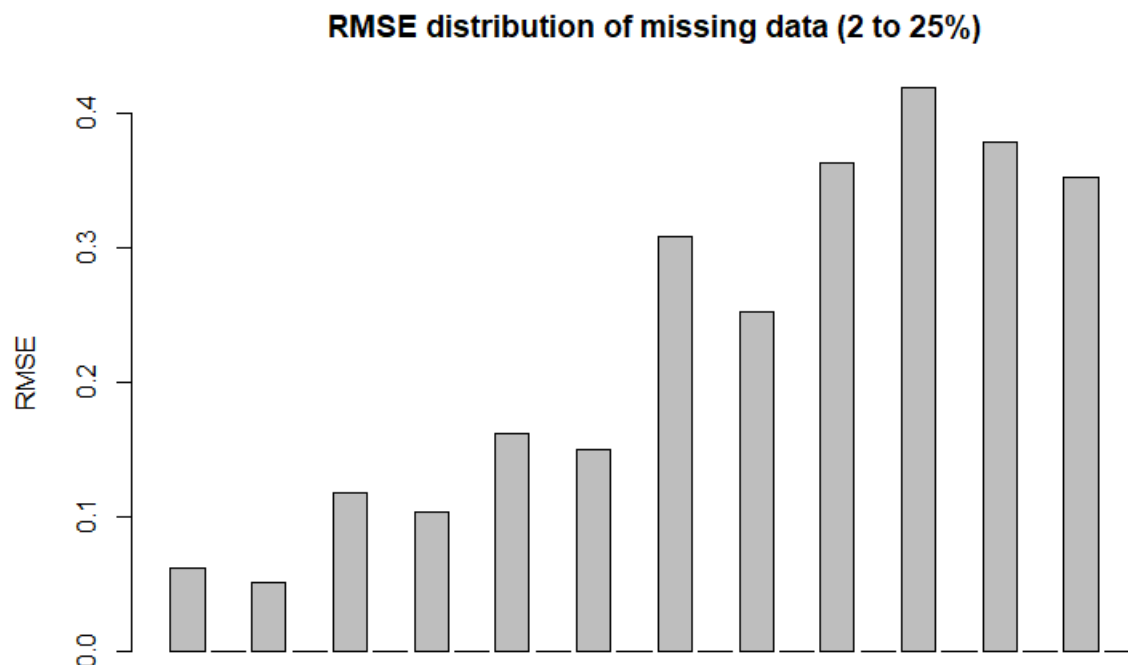


```

118 tab_sch_am1<- table(iris_test_target_sch_am, model_sch_am1)
119 tab_sch_am2<- table(iris_test_target_sch_am, model_sch_am2)
120 tab_sch_am3<- table(iris_test_target_sch_am, model_sch_am3)
121 tab_sch_am4<- table(iris_test_target_sch_am, model_sch_am4)
122 tab_sch_am5<- table(iris_test_target_sch_am, model_sch_am5)
123 tab_sch_am6<- table(iris_test_target_sch_am, model_sch_am6)
124
125 tab_sch_am1
126 tab_sch_am2
127 tab_sch_am3
128 tab_sch_am4
129 tab_sch_am5
130 tab_sch_am6
131
132 confusionMatrix(tab_sch_am1)
133 confusionMatrix(tab_sch_am2)
134 confusionMatrix(tab_sch_am3)
135 confusionMatrix(tab_sch_am4)
136 confusionMatrix(tab_sch_am5)
137 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

```

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])
14       cnt <- cnt+1
15     }#if
16   }#for
17   return(df)
18 }#fun
19
20 mm1<- missing(iris, 1, 600)
21 mm2<- missing(iris, 2.5, 600)
22 mm3<- missing(iris, 5, 600)
23 mm4<- missing(iris, 7.5, 600)
24 mm5<- missing(iris, 10, 600)
25 mm6<- missing(iris, 12.5, 600)
26
27 missing1<- function(df1, percentage1, totalDataCount1){
28   allowedNAVal1 <- (totalDataCount1 * percentage1)/100
29   cnt1 <- 1
30   for(i in 1:length(df1$Petal.Length)){
31     if(df1$Petal.Length[i] > 4 && cnt1<= allowedNAVal1)
32     {
33       df1$Petal.Length[i]<- NA
34       print(df1$Petal.Length[i])
35       cnt1 <- cnt1+1
36     }#if
37   }#for
38   return(df1)
39 }
40
41 nn1<- missing1(mm1, 1, 600)
42 nn2<- missing1(mm2, 2.5, 600)
43 nn3<- missing1(mm3, 5, 600)
44 nn4<- missing1(mm4, 7.5, 600)
45 nn5<- missing1(mm5, 10, 600)
46 nn6<- missing1(mm6, 12.5, 600)
47
48 #imputing missing values using missForest
49
50 imputed_sch_mf1 <- missForest(nn1)
51 imputed_sch_mf2 <- missForest(nn2)
52 imputed_sch_mf3 <- missForest(nn3)

```

```

53 imputed_sch_mf4 <- missForest(nn4)
54 imputed_sch_mf5 <- missForest(nn5)
55 imputed_sch_mf6 <- missForest(nn6)
56
57 completeData_sch_mf1<- imputed_sch_mf1$ximp
58 completeData_sch_mf2<- imputed_sch_mf2$ximp
59 completeData_sch_mf3<- imputed_sch_mf3$ximp
60 completeData_sch_mf4<- imputed_sch_mf4$ximp
61 completeData_sch_mf5<- imputed_sch_mf5$ximp
62 completeData_sch_mf6<- imputed_sch_mf6$ximp
63
64 rmse_sch_mf1<- rmse(completeData_sch_mf1[, -5], iris[, -5], na.rm = TRUE)
65 rmse_sch_mf2<- rmse(completeData_sch_mf2[, -5], iris[, -5], na.rm = TRUE)
66 rmse_sch_mf3<- rmse(completeData_sch_mf3[, -5], iris[, -5], na.rm = TRUE)
67 rmse_sch_mf4<- rmse(completeData_sch_mf4[, -5], iris[, -5], na.rm = TRUE)
68 rmse_sch_mf5<- rmse(completeData_sch_mf5[, -5], iris[, -5], na.rm = TRUE)
69 rmse_sch_mf6<- rmse(completeData_sch_mf6[, -5], iris[, -5], na.rm = TRUE)
70
71 rmse_sch_mf1
72 rmse_sch_mf2
73 rmse_sch_mf3
74 rmse_sch_mf4
75 rmse_sch_mf5
76 rmse_sch_mf6
77
78 tot_rmse <- c(rmse_sch_mf1, rmse_sch_mf2, rmse_sch_mf3, rmse_sch_mf4, rmse_sch_mf5,
  rmse_sch_mf6)
79 per_col <- c(2,5,10,15,20,25)
80 rmse_dfl <- data.frame(percentage = per_col,error = tot_rmse);
81 barplot(rmse_dfl$error, ylab = "RMSE", main="RMSE distribution of missing data (2
  to 25%)")
82
83 #For k-NN
84 normalize<- function(x) {
85   return((x-min(x)) / (max(x)- min(x)))
86 }
87
88 iris_normal<- as.data.frame(lapply(iris[, c(1,2,3,4)], normalize))
89
90 completeData_sch_mf1_normal<- as.data.frame(lapply(completeData_sch_mf1[, c(1,2,3,4)
  ], normalize))
91 completeData_sch_mf2_normal<- as.data.frame(lapply(completeData_sch_mf2[, c(1,2,3,4)
  ], normalize))
92 completeData_sch_mf3_normal<- as.data.frame(lapply(completeData_sch_mf3[, c(1,2,3,4)
  ], normalize))
93 completeData_sch_mf4_normal<- as.data.frame(lapply(completeData_sch_mf4[, c(1,2,3,4)
  ], normalize))
94 completeData_sch_mf5_normal<- as.data.frame(lapply(completeData_sch_mf5[, c(1,2,3,4)
  ], normalize))
95 completeData_sch_mf6_normal<- as.data.frame(lapply(completeData_sch_mf6[, c(1,2,3,4)
  ], normalize))
96
97 iris_train_sch_mf <- iris_normal[1:150,]
98 iris_train_target_sch_mf <- iris[1:150, 5]
99 iris_test_target_sch_mf <- iris[1:150, 5]
100
101 iris_test_sch_mf1 <- completeData_sch_mf1_normal[1:150,]
102 iris_test_sch_mf2 <- completeData_sch_mf2_normal[1:150,]
103 iris_test_sch_mf3 <- completeData_sch_mf3_normal[1:150,]

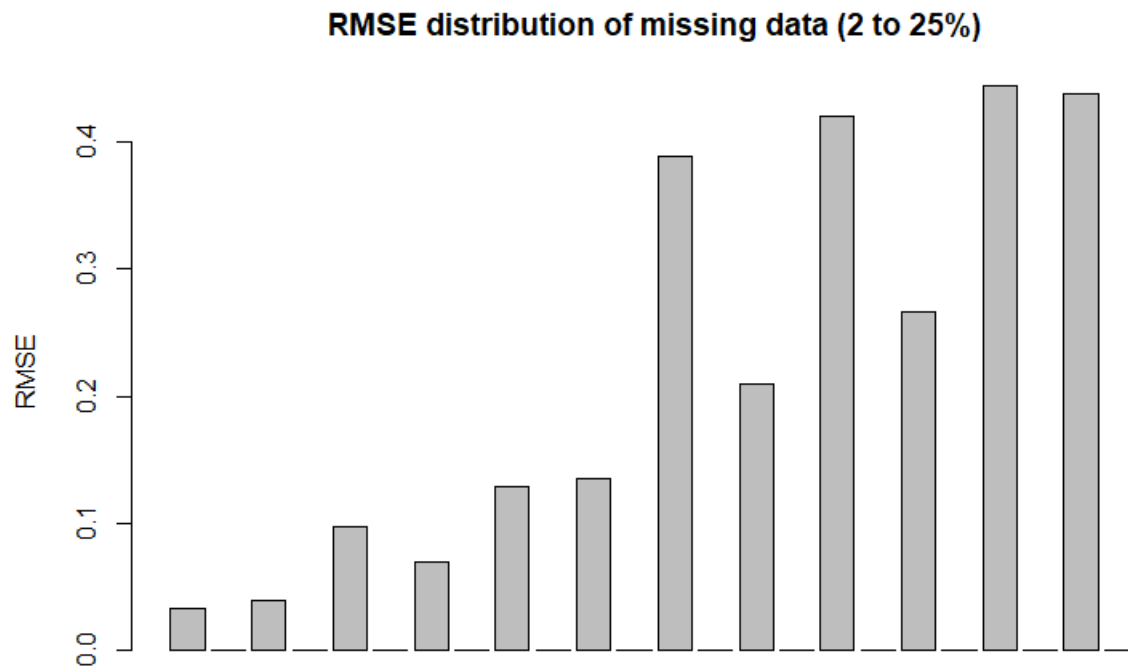
```

```

104 iris_test_sch_mf4 <- completeData_sch_mf4_normal[1:150,]
105 iris_test_sch_mf5 <- completeData_sch_mf5_normal[1:150,]
106 iris_test_sch_mf6 <- completeData_sch_mf6_normal[1:150,]
107
108 require(class)
109
110 #building the model using knn
111 model_sch_mf1<- knn(train=iris_train_sch_mf, test= iris_test_sch_mf1, cl=iris_train_
    target_sch_mf, k=13)
112 model_sch_mf2<- knn(train=iris_train_sch_mf, test= iris_test_sch_mf2, cl=iris_train_
    target_sch_mf, k=13)
113 model_sch_mf3<- knn(train=iris_train_sch_mf, test= iris_test_sch_mf3, cl=iris_train_
    target_sch_mf, k=13)
114 model_sch_mf4<- knn(train=iris_train_sch_mf, test= iris_test_sch_mf4, cl=iris_train_
    target_sch_mf, k=13)
115 model_sch_mf5<- knn(train=iris_train_sch_mf, test= iris_test_sch_mf5, cl=iris_train_
    target_sch_mf, k=13)
116 model_sch_mf6<- knn(train=iris_train_sch_mf, test= iris_test_sch_mf6, cl=iris_train_
    target_sch_mf, k=13)
117
118 tab_sch_mf1<- table(iris_test_target_sch_mf, model_sch_mf1)
119 tab_sch_mf2<- table(iris_test_target_sch_mf, model_sch_mf2)
120 tab_sch_mf3<- table(iris_test_target_sch_mf, model_sch_mf3)
121 tab_sch_mf4<- table(iris_test_target_sch_mf, model_sch_mf4)
122 tab_sch_mf5<- table(iris_test_target_sch_mf, model_sch_mf5)
123 tab_sch_mf6<- table(iris_test_target_sch_mf, model_sch_mf6)
124
125 tab_sch_mf1
126 tab_sch_mf2
127 tab_sch_mf3
128 tab_sch_mf4
129 tab_sch_mf5
130 tab_sch_mf6
131
132 confusionMatrix(tab_sch_mf1)
133 confusionMatrix(tab_sch_mf2)
134 confusionMatrix(tab_sch_mf3)
135 confusionMatrix(tab_sch_mf4)
136 confusionMatrix(tab_sch_mf5)
137 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