Breastcancer data set impuation and regression analysis

In this analysis we will explore mean/mode imputation to impute for missing values, and then use regression on those imputed values for the missing data.

```
cancer<-read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breas

head(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
```

```
str(cancer)
```

```
## 'data.frame':    699 obs. of  11 variables:
##  $ V1 : int  1000025 1002945 1015425 1016277 1017023 1017122 1018099 1018561 1033078 1033078 ...
##  $ V2 : int  5 5 3 6 4 8 1 2 2 4 ...
##  $ V3 : int  1 4 1 8 1 10 1 1 1 2 ...
##  $ V4 : int  1 4 1 8 1 10 1 2 1 1 ...
##  $ V5 : int  1 5 1 1 3 8 1 1 1 1 ...
##  $ V6 : int  2 7 2 3 2 7 2 2 2 2 ...
##  $ V7 : chr  "1" "10" "2" "4" ...
##  $ V8 : int  3 3 3 3 3 9 3 3 1 2 ...
##  $ V9 : int  1 2 1 7 1 7 1 1 1 1 ...
##  $ V10: int  1 1 1 1 1 1 1 1 5 1 ...
##  $ V11: int  2 2 2 2 2 4 2 2 2 2 ...
```

You can see no data for V7 or "Bare Nuclei". That is missing from the dataset.

```
#This is all values with a ? in v7 column
missing<-which(cancer$V7=='?')
missing
```

```
##  [1]  24  41 140 146 159 165 236 250 276 293 295 298 316 322 412 618
```

```
#This has all numerical values from all vars except values = ?
cancer[-missing,7]
```

```
##   [1] "1"  "10" "2"  "4"  "1"  "10" "10" "1"  "1"  "1"  "1"  "1"  "3"  "3"  "9"
##  [16] "1"  "1"  "1"  "10" "1"  "10" "7"  "1"  "1"  "7"  "1"  "1"  "1"  "1"  "1"
```

```
##  [31] "1"  "5"  "1"  "1"  "1"  "1"  "1"  "10" "7"  "3"  "10" "1"  "1"  "1"  "9"
##  [46] "1"  "1"  "8"  "3"  "4"  "5"  "8"  "8"  "5"  "6"  "1"  "10" "2"  "3"  "2"
##  [61] "8"  "2"  "1"  "2"  "1"  "10" "9"  "1"  "1"  "2"  "1"  "10" "4"  "2"  "1"
##  [76] "1"  "3"  "1"  "1"  "1"  "1"  "2"  "9"  "4"  "8"  "10" "1"  "1"  "1"  "1"
##  [91] "1"  "1"  "1"  "1"  "1"  "1"  "6"  "10" "5"  "5"  "1"  "3"  "1"  "3"  "10"
## [106] "10" "1"  "9"  "2"  "9"  "10" "8"  "3"  "5"  "2"  "10" "3"  "2"  "1"  "2"
## [121] "10" "10" "7"  "1"  "10" "1"  "10" "1"  "1"  "1"  "10" "1"  "1"  "2"  "1"
## [136] "1"  "1"  "1"  "1"  "5"  "5"  "1"  "8"  "2"  "1"  "10" "1"  "10" "5"  "3"
## [151] "1"  "10" "1"  "1"  "10" "10" "1"  "1"  "3"  "2"  "10" "1"  "1"  "1"  "1"
## [166] "1"  "1"  "10" "10" "10" "1"  "1"  "1"  "10" "1"  "1"  "1"  "10" "10" "1"
## [181] "8"  "10" "8"  "1"  "8"  "10" "1"  "1"  "1"  "1"  "7"  "1"  "1"  "1"  "10"
## [196] "10" "1"  "1"  "1"  "10" "5"  "1"  "1"  "1"  "10" "8"  "1"  "10" "10" "5"
## [211] "1"  "1"  "4"  "1"  "1"  "10" "5"  "8"  "10" "1"  "10" "5"  "1"  "10" "7"
## [226] "8"  "1"  "10" "1"  "10" "2"  "9"  "10" "2"  "1"  "1"  "5"  "1"  "2"  "10"
## [241] "9"  "1"  "1"  "10" "10" "10" "8"  "10" "1"  "1"  "1"  "8"  "10" "10" "10"
## [256] "10" "3"  "1"  "10" "10" "4"  "1"  "10" "1"  "10" "4"  "1"  "1"  "1"  "1"
## [271] "7"  "1"  "1"  "10" "10" "10" "10" "10" "1"  "5"  "10" "1"  "1"  "10" "10"
## [286] "5"  "1"  "10" "4"  "1"  "10" "1"  "10" "10" "1"  "1"  "3"  "5"  "1"  "1"
## [301] "1"  "1"  "1"  "10" "8"  "1"  "5"  "10" "1"  "10" "1"  "1"  "10" "1"  "4"
## [316] "10" "8"  "1"  "1"  "10" "10" "1"  "10" "1"  "1"  "10" "10" "1"  "1"  "1"
## [331] "10" "1"  "1"  "1"  "1"  "8"  "1"  "1"  "3"  "10" "1"  "1"  "3"  "10" "4"
## [346] "7"  "10" "10" "3"  "3"  "1"  "1"  "10" "10" "1"  "1"  "1"  "1"  "1"  "1"
## [361] "1"  "1"  "1"  "1"  "1"  "1"  "1"  "10" "1"  "1"  "1"  "1"  "10" "1"  "1"
## [376] "2"  "1"  "10" "1"  "1"  "1"  "1"  "1"  "1"  "1"  "1"  "9"  "1"  "1"  "4"
## [391] "1"  "1"  "1"  "1"  "2"  "1"  "1"  "4"  "1"  "10" "3"  "10" "1"  "2"  "1"
## [406] "3"  "10" "1"  "1"  "1"  "10" "1"  "2"  "1"  "1"  "1"  "1"  "1"  "1"  "8"
## [421] "10" "1"  "1"  "1"  "1"  "10" "4"  "3"  "2"  "1"  "1"  "1"  "1"  "1"  "10"
## [436] "1"  "1"  "1"  "10" "1"  "6"  "10" "3"  "1"  "1"  "1"  "5"  "1"  "1"  "1"
## [451] "4"  "10" "10" "1"  "1"  "1"  "1"  "1"  "1"  "1"  "1"  "1"  "1"  "1"  "10"
## [466] "1"  "1"  "5"  "10" "1"  "3"  "1"  "10" "3"  "4"  "1"  "10" "1"  "10" "5"
## [481] "1"  "1"  "1"  "1"  "1"  "1"  "1"  "1"  "1"  "1"  "1"  "5"  "4"  "1"  "1"
## [496] "1"  "1"  "1"  "1"  "10" "10" "1"  "1"  "1"  "10" "1"  "1"  "5"  "10" "1"
## [511] "1"  "1"  "1"  "1"  "1"  "10" "1"  "1"  "1"  "1"  "1"  "1"  "1"  "1"  "1"
## [526] "2"  "1"  "1"  "1"  "1"  "1"  "10" "1"  "1"  "5"  "1"  "1"  "1"  "5"  "1"
## [541] "1"  "1"  "1"  "1"  "1"  "1"  "1"  "1"  "1"  "1"  "10" "1"  "3"  "10" "5"
## [556] "10" "10" "1"  "1"  "2"  "1"  "1"  "1"  "1"  "1"  "1"  "10" "10" "1"  "1"
## [571] "1"  "10" "1"  "3"  "1"  "1"  "10" "10" "1"  "10" "1"  "1"  "1"  "1"  "1"
## [586] "1"  "1"  "1"  "1"  "10" "8"  "1"  "1"  "10" "1"  "10" "2"  "10" "1"  "1"
## [601] "1"  "1"  "1"  "1"  "1"  "2"  "1"  "1"  "1"  "4"  "6"  "5"  "1"  "1"  "1"
## [616] "1"  "1"  "3"  "1"  "1"  "1"  "2"  "1"  "1"  "1"  "1"  "1"  "1"  "1"  "1"
## [631] "1"  "1"  "2"  "1"  "4"  "1"  "1"  "1"  "1"  "1"  "1"  "1"  "10" "1"  "1"
## [646] "1"  "1"  "1"  "1"  "1"  "1"  "1"  "1"  "5"  "8"  "1"  "1"  "1"  "1"  "1"
## [661] "1"  "1"  "1"  "1"  "10" "10" "1"  "1"  "1"  "1"  "1"  "1"  "1"  "1"  "1"
## [676] "5"  "1"  "1"  "2"  "1"  "3"  "4"  "5"
```

```
length(missing)/nrow(cancer)
```

```
## [1] 0.02288984
```

Missing values are only 2% of missing data so imputation is fine. Generally want missing values to be less than 5% of data in order to do imputation. We are using mean/mode imputation where we substitute missing values with mean and mode.

```
as.integer(cancer$V7)
```

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

```
##   [1]  1 10  2  4  1 10 10  1  1  1  1  1  3  3  9  1  1  1 10  1 10  7  1 NA  1
##  [26]  7  1  1  1  1  1  1  5  1  1  1  1  1 10  7 NA  3 10  1  1  1  9  1  1  8
##  [51]  3  4  5  8  8  5  6  1 10  2  3  2  8  2  1  2  1 10  9  1  1  2  1 10  4
##  [76]  2  1  1  3  1  1  1  1  2  9  4  8 10  1  1  1  1  1  1  1  1  1  1  6 10
## [101]  5  5  1  3  1  3 10 10  1  9  2  9 10  8  3  5  2 10  3  2  1  2 10 10  7
## [126]  1 10  1 10  1  1  1 10  1  1  2  1  1  1 NA  1  1  5  5  1 NA  8  2  1 10
## [151]  1 10  5  3  1 10  1  1 NA 10 10  1  1  3 NA  2 10  1  1  1  1  1  1 10 10
## [176] 10  1  1  1 10  1  1  1 10 10  1  8 10  8  1  8 10  1  1  1  1  7  1  1  1
## [201] 10 10  1  1  1 10  5  1  1  1 10  8  1 10 10  5  1  1  4  1  1 10  5  8 10
## [226]  1 10  5  1 10  7  8  1 10  1 NA 10  2  9 10  2  1  1  5  1  2 10  9  1 NA
## [251]  1 10 10 10  8 10  1  1  1  8 10 10 10 10  3  1 10 10  4  1 10  1 10  4  1
## [276] NA  1  1  1  7  1  1 10 10 10 10 10  1  5 10  1  1 NA 10 NA 10  5 NA  1 10
## [301]  4  1 10  1 10 10  1  1  3  5  1  1  1  1  1 NA 10  8  1  5 10 NA  1 10  1
## [326]  1 10  1  4 10  8  1  1 10 10  1 10  1  1 10 10  1  1  1 10  1  1  1  1  8
## [351]  1  1  3 10  1  1  3 10  4  7 10 10  3  3  1  1 10 10  1  1  1  1  1  1  1
## [376]  1  1  1  1  1  1 10  1  1  1  1 10  1  1  2  1 10  1  1  1  1  1  1  1  1
## [401]  9  1  1  4  1  1  1  1  2  1  1 NA  4  1 10  3 10  1  2  1  3 10  1  1  1
## [426] 10  1  2  1  1  1  1  1  1  8 10  1  1  1  1 10  4  3  2  1  1  1  1  1 10
## [451]  1  1  1 10  1  6 10  3  1  1  1  5  1  1  1  4 10 10  1  1  1  1  1  1  1
## [476]  1  1  1  1 10  1  1  5 10  1  3  1 10  3  4  1 10  1 10  5  1  1  1  1  1
## [501]  1  1  1  1  1  1  5  4  1  1  1  1  1  1 10 10  1  1  1 10  1  1  5 10  1
## [526]  1  1  1  1  1 10  1  1  1  1  1  1  1  1  1  2  1  1  1  1  1 10  1  1  5
## [551]  1  1  1  5  1  1  1  1  1  1  1  1  1  1  1  1 10  1  3 10  5 10 10  1  1  2
## [576]  1  1  1  1  1  1 10 10  1  1  1 10  1  3  1  1 10 10  1 10  1  1  1  1  1
## [601]  1  1  1  1 10  8  1  1 10  1 10  2 10  1  1  1  1 NA  1  1  1  2  1  1  1
## [626]  4  6  5  1  1  1  1  1  3  1  1  1  2  1  1  1  1  1  1  1  1  1  1  2  1
## [651]  4  1  1  1  1  1  1  1 10  1  1  1  1  1  1  1  1  1  1  5  8  1  1  1  1
## [676]  1  1  1  1  1 10 10  1  1  1  1  1  1  1  1  1  5  1  1  2  1  3  4  5
```

```
mean <- round(mean(as.integer(cancer[-missing,7]),na.rm = TRUE))
mean
```

```
## [1] 4
```

```
table(cancer[-missing,7])
```

```
##
##   1  10   2   3   4   5   6   7   8   9
## 402 132  30  28  19  30   4   8  21   9
```

```
mode <- which.max(as.numeric(table(cancer[-missing,7])))
mode
```

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

So mean is rounded to 4 and mode is 1.

```
cancer[missing,7] = mean
cancer[missing,7]
```

```
##  [1] "4" "4" "4" "4" "4" "4" "4" "4" "4" "4" "4" "4" "4" "4" "4" "4"
```

putting mean into the data set. All values with initially ? are now 4.

Now using Regression including the imputed values.

```
data <- read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/brea

#changing v7 to numeric column from character column
data[, 7] <- sapply(data[, 7], as.numeric)
```

```
## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion

## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion

## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion

## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion

## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion

## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion

## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion

## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion

## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion

## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion

## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion

## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion

## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion

## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion

## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion

## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion
```

```
#all variables are predictors except for the missing values (imputed column) and response
index <- which(is.na(data$V7), arr.ind=TRUE)
newdata <- data[-index,2:10]


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

```
##
## Call:
## lm(formula = V7 ~ V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10, data = newdata)
##
## 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
```

Using stepwise regression below to fine tune the model to reduce the number of predictor variables so I can get a more accurate model. Drawbacks are that it is a greedy algorithm and for predictor variables that change in weight quickly this method won't produce the best solution. Testing out stepwise regression to see how accurate a model I can get.

```
#Using cross-validation and then using step wise regression to leave out
#insignificant factors

train <- trainControl(method = "cv", number = 10)

model1 <- train(V7 ~., data = newdata ,
                    method = "leapBackward",
                    tuneGrid = data.frame(nvmax = 1:4),
                    trControl = train
                    )

model1$results
```

```
##   nvmax     RMSE  Rsquared      MAE    RMSESD RsquaredSD     MAESD
## 1     1 2.644829 0.4710389 1.851783 0.2889885  0.1250305 0.2449767
## 2     2 2.509632 0.5244554 1.714347 0.2551182  0.1061307 0.1990182
## 3     3 2.387951 0.5701944 1.621977 0.2925457  0.1222840 0.2139419
## 4     4 2.271978 0.6121952 1.535337 0.3007289  0.1213962 0.1964813
```

We can see from this that the model with 4 predictors has the best model based on the highest R squared value at 0.61.

```
#Predicting values based on the stepwise regression model
predicted <- predict(model1, newdata=cancer[index,])

final <- data

final[index,]$V7 <- as.integer(predicted)

head(final)
```

```
##        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
```

Values imputed with regression has been done. Now we do the regression with perturbation.

```
#creating 16 random positive numbers for perturbation

n <- rnorm(16, mean = predicted, sd = sd(predicted))

abs(n)
```

```
##  [1] 6.7914113 8.9163660 2.7944864 1.8953335 0.4244491 3.9665215 3.1145754
##  [8] 0.3873431 0.3577999 4.5880031 0.9811989 4.3071569 7.1080988 3.8975010
## [15] 4.1701382 2.0965996
```

```
perturbed <- data
perturbed[index,]$V7 <- as.integer(abs(n))

head(perturbed)
```

```
##        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
```

```
#This is final data with perturbed values
```

This is the final data set we get with regression plus the imputation method. We have filled out the values using imputation. Using step wise regression we were able to eliminate insignificant predictors and use cross validation to reduce bias in the data by making sure the data is resampled and that significant data is not just seen in training or test data.