In [1]:
```
url <- "http://archive.ics.uci.edu/ml/machine-learning-databases/breas
t-cancer-wisconsin/breast-cancer-wisconsin.data"
data <- read.table(url, stringsAsFactors = FALSE, header = FALSE, sep
= ",")
```

In [2]:
```
head(data)
```

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

In [3]:
```
### checking the value of missing values in each column. Missing value
s are represented by '?'
colSums(data == '?')
```

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

In [4]:
```
data[data$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

In [5]:
```
num_missing = 100*nrow(data[data$V7=='?',])/nrow(data)
num_missing
```

2.28898426323319

It can be seen there are missing values only in V7. We have 16 missing values which is less than 5%. There doesn't seem to be any bias.

In [6]:
```
mode_function <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}
```

In [7]:
```
### getting the indices for missing data

missing_indices <- which(data$V7 == '?', arr.ind = T)
```

In [8]:
```
### finding the mode for column V7 using data that is not missing

mode_V7 <- as.numeric(mode_function(data[-missing_indices,'V7']))
mode_V7
```

1

In [9]:
```
###imputation using mode

data_impute_mode <- data
data_impute_mode[missing_indices, 'V7'] <- mode_V7
```

In [10]:
```
### no missing values after mode imputation

colSums(data_impute_mode == '?')
```

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

In [11]:
```
### finding the mean of column V7 using data that is not missing

mean_V7 <- mean(as.integer(data[-missing_indices, 'V7']))
mean_V7
```

3.54465592972182

In [12]:
```
#### imputation using mean value

data_impute_mean <- data
data_impute_mean[missing_indices, 'V7'] <- as.integer(mean_V7)
```

In [13]:
```
colSums(data_impute_mean == '?')
```

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

In [14]:
```
#### imputation using regression


data_regress <- data[-missing_indices, 2:10]
```

In [15]:
```
head(data_regress)
```

V2	V3	V4	V5	V6	V7	V8	V9	V10
5	1	1	1	2	1	3	1	1
5	4	4	5	7	10	3	2	1
3	1	1	1	2	2	3	1	1
6	8	8	1	3	4	3	7	1
4	1	1	3	2	1	3	1	1
8	10	10	8	7	10	9	7	1

In [16]:
```
model <- lm(V7 ~., data = data_regress)
summary(model)
```

```
Call:
lm(formula = V7 ~ ., data = data_regress)

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
```

In [17]:
```
### it can be seen that not all variables are significant. Hence perfo
rming a stepwise regression to
### select siginificant features
```

In [18]:
```
step(model)
```

```
Start:  AIC=1131.43
V7 ~ V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10

       Df Sum of Sq    RSS    AIC
- V9    1     0.139 3486.8 1129.5
- V3    1     4.120 3490.8 1130.2
- V10   1     8.317 3495.0 1131.0
<none>              3486.6 1131.4
- V6    1    10.806 3497.5 1131.5
- V4    1   111.227 3597.9 1150.9
- V8    1   152.482 3639.1 1158.7
- V2    1   157.657 3644.3 1159.6
- V5    1   283.119 3769.8 1182.8

Step:  AIC=1129.45
```

```
V7 ~ V2 + V3 + V4 + V5 + V6 + V8 + V10

       Df Sum of Sq    RSS    AIC
- V3    1     4.028 3490.8 1128.2
- V10   1     8.179 3495.0 1129.0
<none>              3486.8 1129.5
- V6    1    11.211 3498.0 1129.7
- V4    1   114.768 3601.6 1149.6
- V2    1   158.696 3645.5 1157.8
- V8    1   160.776 3647.6 1158.2
- V5    1   285.902 3772.7 1181.3

Step:  AIC=1128.24
V7 ~ V2 + V4 + V5 + V6 + V8 + V10

       Df Sum of Sq    RSS    AIC
- V6    1     8.606 3499.4 1127.9
- V10   1     8.889 3499.7 1128.0
<none>              3490.8 1128.2
- V4    1   153.078 3643.9 1155.6
- V2    1   155.308 3646.1 1156.0
- V8    1   157.123 3647.9 1156.3
- V5    1   282.133 3772.9 1179.3

Step:  AIC=1127.92
V7 ~ V2 + V4 + V5 + V8 + V10

       Df Sum of Sq    RSS    AIC
- V10   1     5.562 3505.0 1127.0
<none>              3499.4 1127.9
- V2    1   159.594 3659.0 1156.4
- V8    1   169.954 3669.4 1158.3
- V4    1   206.785 3706.2 1165.1
- V5    1   295.807 3795.2 1181.3

Step:  AIC=1127.01
V7 ~ V2 + V4 + V5 + V8

       Df Sum of Sq    RSS    AIC
<none>              3505.0 1127.0
- V2    1    155.70 3660.7 1154.7
- V8    1    172.42 3677.4 1157.8
- V4    1    201.22 3706.2 1163.1
- V5    1    290.68 3795.7 1179.4
```

```
Call:
lm(formula = V7 ~ V2 + V4 + V5 + V8, data = data_regress)

Coefficients:
(Intercept)           V2           V4           V5           V8
    -0.5360       0.2262       0.3173       0.3323       0.3238
```

In [19]:

```
###
model_modified <- lm(V7 ~ V2 + V4 + V5 + V8, data = data_regress)
summary(model_modified)
```

```
Call:
lm(formula = V7 ~ V2 + V4 + V5 + V8, data = data_regress)

Residuals:
    Min      1Q  Median      3Q     Max
-9.8115 -0.9531 -0.3111  0.6678  8.6889

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.53601    0.17514  -3.060   0.0023 **
V2           0.22617    0.04121   5.488 5.75e-08 ***
V4           0.31729    0.05086   6.239 7.76e-10 ***
V5           0.33227    0.04431   7.499 2.03e-13 ***
V8           0.32378    0.05606   5.775 1.17e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.274 on 678 degrees of freedom
Multiple R-squared:  0.6129,    Adjusted R-squared:  0.6107
F-statistic: 268.4 on 4 and 678 DF,  p-value: < 2.2e-16
```

In [20]:
```
### performing cross validation to check the performance of above mode
l.

library(caret)
data_regress$V7 <- as.integer(data_regress$V7)
train.control <- trainControl(method = 'repeatedcv', repeats = 5,
number = 5)
model_cv <- train(V7 ~ V2 + V4 + V5 + V8, data = data_regress, method
= 'lm', trControl = train.control)
print(model_cv)
```

```
Warning message:
“package ‘caret’ was built under R version 3.4.4”Loading required pa
ckage: lattice
Loading required package: ggplot2
Warning message:
“package ‘ggplot2’ was built under R version 3.4.4”Warning message i
n as.POSIXlt.POSIXct(Sys.time()):
“unknown timezone 'zone/tz/2018f.1.0/zoneinfo/America/New_York'”
```

```
Linear Regression

683 samples
  4 predictor

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 546, 547, 546, 547, 546, 546, ...
Resampling results:

  RMSE      Rsquared   MAE
  2.294518  0.6136416  1.53733

Tuning parameter 'intercept' was held constant at a value of TRUE
```

In [21]:
```
V7_regress_impute <- predict(model_modified, data[missing_indices,])
```

In [22]:
```
### printing values to check if they are within 1-10
V7_regress_impute
```

24	5.45853515759216
41	7.98161057649732
140	0.987283186665087
146	1.62185603726251
159	0.980785074712774
165	2.21574405733159
236	2.71526516651999
250	1.76340589062385
276	2.07419420397025
293	6.08660989623727
295	0.987283186665087
298	2.52653237067799
316	5.24383468761997
322	1.76340589062385
412	0.987283186665087
618	0.663498649414061

In [23]:
```
data_impute_regress <- data
data_impute_regress[missing_indices, 'V7'] <- V7_regress_impute
data_impute_regress$V7 <- as.integer(data_impute_regress$V7)

###ensuring all values are within the range

data_impute_regress$V7[data_impute_regress$V7 > 10] <- 10
data_impute_regress$V7[data_impute_regress$V7 < 1] <- 1
```

In [24]:
```
#### imputation using regression pertubation


set.seed(42)

data_impute_pert <- data
data_impute_pert[missing_indices, 'V7'] <- rnorm(length(V7_regress_imp
ute),
                                                 V7_regress_impute,
                                                 sd(V7_regress_impute)
)
data_impute_pert$V7 <- as.integer(data_impute_pert$V7)


### ensuring the values are between 1 and 10
data_impute_pert$V7[data_impute_pert$V7 > 10] <- 10
data_impute_pert$V7[data_impute_pert$V7 < 1] <- 1
```

In [25]:
```
####classification using KNN

library(kknn)

set.seed(42)

training <- sample(nrow(data), size = floor(nrow(data) * 0.75))
test <- setdiff(1:nrow(data), training)
```

```
Attaching package: ‘kknn’

The following object is masked from ‘package:caret’:

    contr.dummy
```

In [26]:
```
### mode imputation

###if we don't include the following command we get a subscript out of bond error
data_impute_mode$V7 <- as.integer(data_impute_mode$V7)

for (k in 1:5) {

  knn_model <- kknn(V11~V2+V3+V4+V5+V6+V7+V8+V9+V10, data_impute_mode[training,], data_impute_mode[test,], k=k)
  pred <- as.integer(fitted(knn_model)+0.5) # round off to 2 or 4
  acc_knn = sum(pred == data_impute_mode[test,]$V11) / nrow(data_impute_mode[test,])
  print(acc_knn)
}
```

```
[1] 0.9542857
[1] 0.9542857
[1] 0.9371429
[1] 0.9371429
[1] 0.9371429
```

In [27]:
```
### mean imputation

data_impute_mean$V7 <- as.integer(data_impute_mean$V7)

for (k in 1:5) {

  knn_model <- kknn(V11~V2+V3+V4+V5+V6+V7+V8+V9+V10, data_impute_mean[training,], data_impute_mean[test,], k=k)
  pred <- as.integer(fitted(knn_model)+0.5) # rounding off
  acc_knn = sum(pred == data_impute_mean[test,]$V11) / nrow(data_impute_mean[test,])
  print(acc_knn)
}
```

```
[1] 0.9542857
[1] 0.9542857
[1] 0.9371429
[1] 0.9371429
[1] 0.9371429
```

In [28]:
```
### regression imputation

for (k in 1:5) {

  knn_model <- kknn(V11~V2+V3+V4+V5+V6+V7+V8+V9+V10,
data_impute_regress[training,],
                    data_impute_regress[test,], k=k)
  pred <- as.integer(fitted(knn_model)+0.5) # rounding off
  acc_knn = sum(pred == data_impute_regress[test,]$V11) / nrow(data_im
pute_regress[test,])
  print(acc_knn)
}
```

```
[1] 0.9485714
[1] 0.9485714
[1] 0.9314286
[1] 0.9314286
[1] 0.9314286
```

In [29]:
```
### regression pertubation imputation

for (k in 1:5) {

  knn_model <- kknn(V11~V2+V3+V4+V5+V6+V7+V8+V9+V10, data_impute_pert[
training,],
                    data_impute_pert[test,], k=k)
  pred <- as.integer(fitted(knn_model)+0.5) # rounding off
  acc_knn = sum(pred == data_impute_pert[test,]$V11) / nrow(data_imput
e_pert[test,])
  print(acc_knn)
}
```

```
[1] 0.9485714
[1] 0.9485714
[1] 0.9257143
[1] 0.9257143
[1] 0.9257143
```

In [30]:
```
###dropping missing values
data_drop <- data[-missing_indices,]

training_drop <- sample(nrow(data_drop), size = floor(nrow(data_drop)
* 0.75))
test_drop <- setdiff(1:nrow(data_drop), training)
for (k in 1:5) {

  knn_model <- kknn(V11~V2+V3+V4+V5+V6+V7+V8+V9+V10, data_drop[trainin
g_drop,],
                    data_drop[test_drop,], k=k)
  pred <- as.integer(fitted(knn_model)+0.5) # rounding off
  acc_knn = sum(pred == data_drop[test_drop,]$V11) / nrow(data_drop[te
st_drop,])
  print(acc_knn)
}
```

```
[1] 0.994152
[1] 0.994152
[1] 0.9532164
[1] 0.9532164
[1] 0.9532164
```

In [31]:
```
### using interaction

data_binary <- data
data_binary$V12[data$V7 == "?"] <- 0
data_binary$V12[data$V7 != "?"] <- 1

# Create interaction factor for V7 and V12.

data_binary$V13[data$V7 == "?"] <- 0
data_binary$V13[data$V7 != "?"] <- as.integer(data[-missing_indices,]$
V7)

for (k in 1:5) {

  knn_model <- kknn(V11~V2+V3+V4+V5+V6+V8+V9+V10+V13, data_binary[trai
ning,], data_binary[test,], k=k)
  pred <- as.integer(fitted(knn_model)+0.5) # rounding off
  acc_knn = sum(pred == data_binary[test,]$V11) / nrow(data_binary[tes
t,])
  print(acc_knn)
}
```

```
[1] 0.9542857
[1] 0.9542857
[1] 0.9371429
[1] 0.9371429
[1] 0.9371429
```

It can be seen that there's not much of a difference between accuracies obtained using different datasets (data using different imputations). The highest accuracy obtained was when we dropped the missing values. However, the difference was not so significant. One more interesting thing to note --- performance on mean, mode and interaction dataset is similar i.e. same accuracy values are obtained. I believe this nature is due to very less missing values. If we had more data missing, then performance would have been different.

15.1

I work as a data scientist at a small business lending firm. We have targets for each month i.e. number of loans to be booked. At the same time, amount of capital to lend is limited. We have to ensure that we optimize and book loans with good quality to ensure efficient utilization of the capital.

n <- number of loans to book amt <- capital available to book loans

loss on each loan <- loss due to default of individual loans

min (total loss) while ensuring the constraint on loans and amount.

data required would be performance of loans booked in the past -- losses, characteristics of loans, etc.