R Notebook

Virly Ananda

Question 1

Download the data set for the tutorial.

```
url<-"https://da5030.weebly.com/uploads/8/6/5/9/8659576/prostate_cancer.csv"
destfile<-"/Users/virlyananda/Desktop/DA5030/DA5030.PR3.Virly.Ananda/Prostate_Cancer.
csv"
download.file(url,destfile)
```

```
# Store our dataset in a variable called prc
prc<-read.csv("Prostate_Cancer.csv",stringsAsFactors = F) # Customize our dataset to
not convert character data into factors
str(prc)
```

```
## 'data.frame':    100 obs. of  10 variables:
##  $ id               : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ diagnosis_result : chr  "M" "B" "M" "M" ...
##  $ radius           : int  23 9 21 14 9 25 16 15 19 25 ...
##  $ texture          : int  12 13 27 16 19 25 26 18 24 11 ...
##  $ perimeter        : int  151 133 130 78 135 83 120 90 88 84 ...
##  $ area             : int  954 1326 1203 386 1297 477 1040 578 520 476 ...
##  $ smoothness       : num  0.143 0.143 0.125 0.07 0.141 0.128 0.095 0.119 0.127 0.
119 ...
##  $ compactness      : num  0.278 0.079 0.16 0.284 0.133 0.17 0.109 0.165 0.193 0.2
4 ...
##  $ symmetry         : num  0.242 0.181 0.207 0.26 0.181 0.209 0.179 0.22 0.235 0.2
03 ...
##  $ fractal_dimension: num  0.079 0.057 0.06 0.097 0.059 0.076 0.057 0.075 0.074 0.
082 ...
```

Question 2

Follow this tutorial on applying kNN to prostate cancer detection and implement all of the steps in an R Notebook. Make sure to explain each step and what it does. (Note: The data set provided as part of this assignment has been slightly modified from the one used in the tutorial, so small deviations in the result can be expected.)

```
# Remove column ID since we won't be using the column
prc<-prc[-1]
head(prc) # The ID column has been removed, thus leaving us with diagnosis_result as
the 1st column
```

	diagnosis_result <chr>	radius <int>	texture <int>	perimeter <int>	a... <int>	smoothn... <dbl>	compactn... <dbl>	symm... <dbl>
1	M	23	12	151	954	0.143	0.278	0.242
2	B	9	13	133	1326	0.143	0.079	0.181
3	M	21	27	130	1203	0.125	0.160	0.207
4	M	14	16	78	386	0.070	0.284	0.260
5	M	9	19	135	1297	0.141	0.133	0.181
6	B	25	25	83	477	0.128	0.170	0.209

6 rows

```
# Check the number of patients through table()
table(prc$diagnosis_result)
```

```
##
##  B  M
## 38 62
```

```
# B stands for Benign Cancer while M stands for Malignant Cancer. Create new column w
here we store the percentage
prc$diagnosis<-factor(prc$diagnosis_result,levels = c("B","M"), labels = c("Benign","
Malignant"))
round(prop.table(table(prc$diagnosis))*100, digits = 1)
```

```
##
##    Benign Malignant
##        38        62
```

```
# To prepare our data for machine learning application, we change columns containing
numeric values to a common scale:
normalize<-function(x){
  return((x-min(x))/(max(x)-min(x)))
}

prc_n<-as.data.frame(lapply(prc[2:9],normalize))

summary(prc_n$radius) # Here we can check whether the data has been normalized
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1875  0.5000  0.4906  0.7500  1.0000
```

```
# A glimpse of what is contained in prc_n dataframe
head(prc_n)
```

	radius	texture	perimeter	area	smoothn...	compactn...	symmetry	fractal_dimen
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<(
1	0.8750	0.0625	0.8250000	0.4486874	1.0000000	0.7817590	0.6331361	0.59090
2	0.0000	0.1250	0.6750000	0.6706444	1.0000000	0.1335505	0.2721893	0.09090
3	0.7500	1.0000	0.6500000	0.5972554	0.7534247	0.3973941	0.4260355	0.15909
4	0.3125	0.3125	0.2166667	0.1097852	0.0000000	0.8013029	0.7396450	1.00000
5	0.0000	0.5000	0.6916667	0.6533413	0.9726027	0.3094463	0.2721893	0.13636
6	1.0000	0.8750	0.2583333	0.1640811	0.7945205	0.4299674	0.4378698	0.52272

6 rows

```
# Create Training and Test Dataset: Ratio of 65:35
prc_train<-prc_n[1:65,] # Training Set
prc_test<-prc_n[66:100,] # The rest of the 35 dataset are stored as test set
```

```
# Include target variable(diagnosis_result) from 1st column of prc into the training
and testing set
prc_train_labels<-prc[1:65,1]
prc_test_labels<-prc[66:100,1]
```

```
# Check the results: Notice here both labels are considered to be characters
head(prc_train_labels)
```

```
## [1] "M" "B" "M" "M" "M" "B"
```

```
head(prc_test_labels)
```

```
## [1] "M" "B" "B" "B" "B" "M"
```

Once we have prepared our data into training and testing dataset, we can apply KNN to train our model data:

```
# To apply KNN, install packages class
# install.packages("class")
library("class") # Load package
```

```
# Classify our test data
# K=sqrt(original observation which is 100)

prc_test_pred<-knn(train = prc_train, test = prc_test, cl = prc_train_labels, k = 7)
```

Evaluate model performance to evaluate prc_test_pred whether they match with prc_test_labels:

```
# Install and apply gmodels
# install.packages("gmodels")
library(gmodels)
```

```
# Apply CrossTable() function
CrossTable(x = prc_test_labels, y = prc_test_pred,prop.chisq = F)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  35
##
##
##                 | prc_test_pred
## prc_test_labels |         B |         M | Row Total |
## ----------------|-----------|-----------|-----------|
##               B |        10 |         9 |        19 |
##                 |     0.526 |     0.474 |     0.543 |
##                 |     0.909 |     0.375 |           |
##                 |     0.286 |     0.257 |           |
## ----------------|-----------|-----------|-----------|
##               M |         1 |        15 |        16 |
##                 |     0.062 |     0.938 |     0.457 |
##                 |     0.091 |     0.625 |           |
##                 |     0.029 |     0.429 |           |
## ----------------|-----------|-----------|-----------|
##    Column Total |        11 |        24 |        35 |
##                 |     0.314 |     0.686 |           |
## ----------------|-----------|-----------|-----------|
##
##
```

Based on the result, the total data has 35 observations. Out of the total, in Benign, there are 10 cases prediction are accurate(28.6%), and 15 cases prediction are for Malignant are accurate(43%). However, we also detect False Negative(FN) in both Benign(9 cases as Malignant = 25.7%) and Malignant(1 case as Benign = 3%).

To conclude(based on original ones with k=10: TP = 17% FP = 3% TN = 43% FN = 37%

Total Accuracy: (TN + TP)/35 = 60%

1st initiation: 60% Accuracy with k=10 2nd initiation: 71.5% Accuracy with k=7 (This is initiated after the implementation of caret package below)

Question 3

Once you've complete the tutorial, try another kNN implementation from another package, such as the caret package. Compare the accuracy of the two implementations.

```
# Apply Caret package
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
# Re-read the data to avoid cofusion
prc<-read.csv("Prostate_Cancer.csv",stringsAsFactors = F) # Customize our dataset to
not convert character
```

```
# Create a copy of prc dataset
prc2<-prc[-1]
prc2$diagnosis <- factor(prc2$diagnosis_result, levels = c("B", "M"), labels = c("Ben
ign", "Malignant"))
```

```
# Create Training and Testing Dataset
set.seed(123)
dataPartition<-createDataPartition(y = prc2$diagnosis, p=0.65,list = FALSE)
traindp<-prc2[dataPartition,]
testdp<-prc2[-dataPartition,]

# Obtain the result from train dataset
round(prop.table(table(traindp$diagnosis))*100,digits = 1)
```

```
##
##    Benign Malignant
##      37.9      62.1
```

```
# Obtain the result from test dataset
round(prop.table(table(testdp$diagnosis)) * 100, digits = 1)
```

```
##
##    Benign Malignant
##      38.2      61.8
```

```
# Obtain the result from the original data
round(prop.table(table(prc2$diagnosis)) * 100, digits = 1)
```

```
##
##    Benign Malignant
##        38        62
```

To preprocess the data, we should normalized the data once again for KNN application:

```
ppTrain<-traindp[,names(traindp) != "diagnosis"]
ppNormalize<-preProcess(x=ppTrain, method = c("scale","center","YeoJohnson"))
ppNormalize
```

```
## Created from 66 samples and 9 variables
##
## Pre-processing:
##   - centered (8)
##   - ignored (1)
##   - scaled (8)
##   - Yeo-Johnson transformation (4)
##
## Lambda estimates for Yeo-Johnson transformation:
## 0.84, -0.33, 0.46, 0.25
```

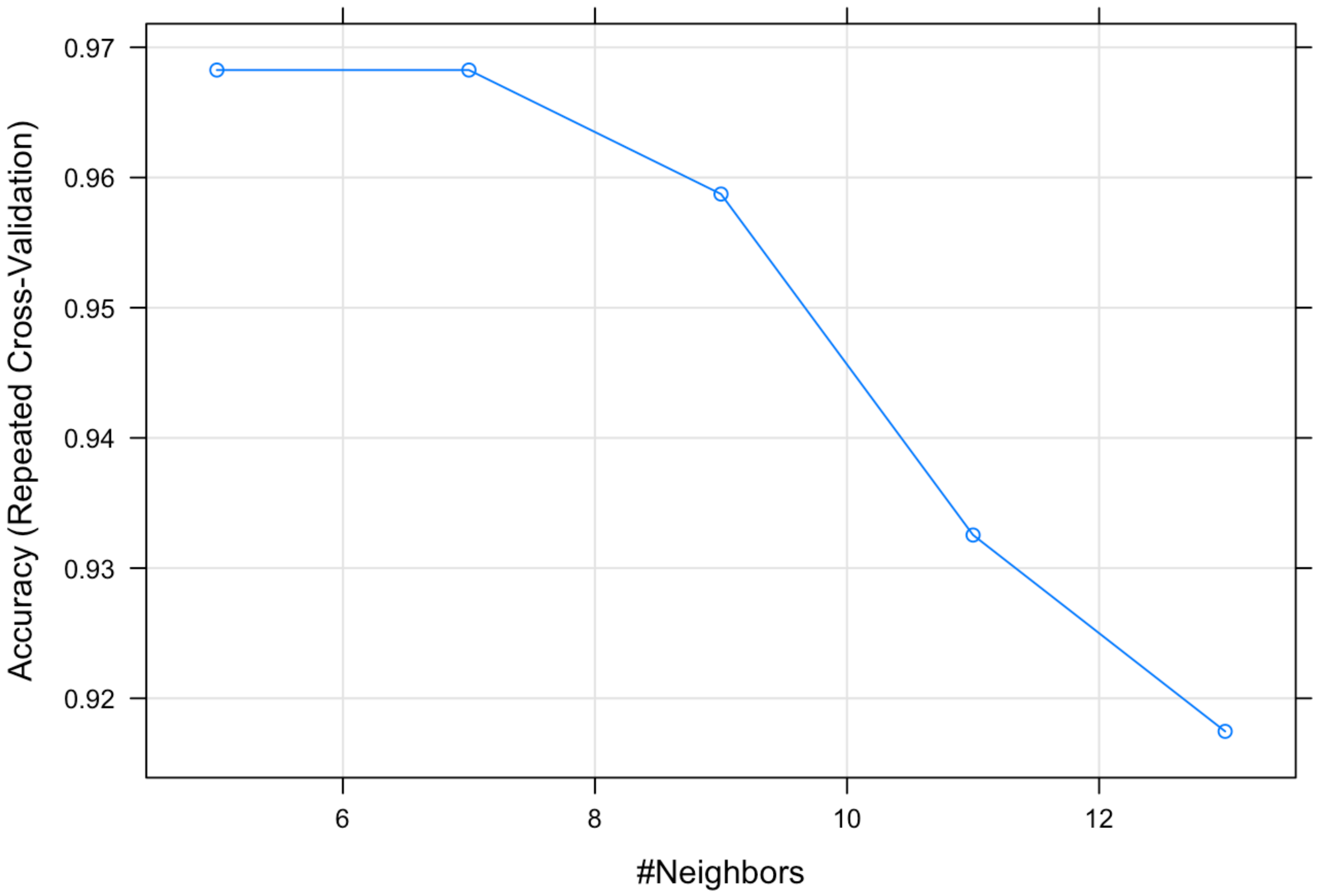
Train the data

```
set.seed(123)
# Apply repeated cross validation method
ctrlData<-trainControl(method = "repeatedcv", number = 10, repeats = 3) # 10 Folds
fitKNN<-train(diagnosis ~ ., data = traindp, method = "knn", trControl = ctrlData, pr
eProcess=c("scale"), tuneLength = 5)

fitKNN
```

```
## k-Nearest Neighbors
##
## 66 samples
##  9 predictor
##  2 classes: 'Benign', 'Malignant'
##
## Pre-processing: scaled (9)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 59, 59, 60, 60, 59, 60, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.9682540  0.9225673
##    7  0.9682540  0.9225673
##    9  0.9587302  0.9039006
##   11  0.9325397  0.8359917
##   13  0.9174603  0.8046246
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 7.
```

```
# Visualize accuracy(RCV: Repeated Cross Validation)
plot(fitKNN)
```



Based on the visualization shown above, the best accuracy falls on k=5 to k=7. Whereas the highest number throughout k-folds decreases the accuracy. Compared to the previously applied Class package, Caret package provides a more detailed information on which k-fold should be used to obtain accuracy. As we applied k=7 to Class package to execute KNN, the accuracy increases to 71.5%(previously 60% on the original k=10).

In conclusion, Caret package provides more accuracy(97%) in terms of KNN application compared to Class(71.5%).

Question 4

Try the confusionMatrix function from the caret package to determine the accuracy of both algorithms.

Below, we display Confusion Matrix to check the total accuracy:

```
# Apply confusion matrix to check the accuracy of the predicted KNN from Caret Packag
e
Predict.KNN<-predict(fitKNN,newdata = testdp)
confusionMatrix(Predict.KNN, testdp$diagnosis)
```

```
## Confusion Matrix and Statistics
## 
##             Reference
## Prediction  Benign Malignant
##   Benign        11         0
##   Malignant      2        21
##                                           
##                Accuracy : 0.9412          
##                  95% CI : (0.8032, 0.9928)
##     No Information Rate : 0.6176          
##     P-Value [Acc > NIR] : 1.82e-05        
##                                           
##                   Kappa : 0.8717          
##                                           
##  Mcnemar's Test P-Value : 0.4795          
##                                           
##             Sensitivity : 0.8462          
##             Specificity : 1.0000          
##          Pos Pred Value : 1.0000          
##          Neg Pred Value : 0.9130          
##              Prevalence : 0.3824          
##          Detection Rate : 0.3235          
##    Detection Prevalence : 0.3235          
##       Balanced Accuracy : 0.9231          
##                                           
##        'Positive' Class : Benign          
## 
```

```
# Apply confusion matrix to check the accuracy of the predicted KNN from Class Packag
e

confusionMatrix(prc_test_pred, as.factor(prc_test_labels))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction  B  M
##          B 10  1
##          M  9 15
##                                          
##                Accuracy : 0.7143         
##                  95% CI : (0.537, 0.8536)
##     No Information Rate : 0.5429         
##     P-Value [Acc > NIR] : 0.02934        
##                                          
##                   Kappa : 0.4462         
##                                          
##  Mcnemar's Test P-Value : 0.02686        
##                                          
##             Sensitivity : 0.5263         
##             Specificity : 0.9375         
##          Pos Pred Value : 0.9091         
##          Neg Pred Value : 0.6250         
##              Prevalence : 0.5429         
##          Detection Rate : 0.2857         
##    Detection Prevalence : 0.3143         
##       Balanced Accuracy : 0.7319         
##                                          
##        'Positive' Class : B              
## 
```

CONFUSION MATRIX ACCURACY Caret Package: 94% Class Package: 71%