Case study 5 – Analytics in Medicine

Vishwa Koppisetti

February 25, 2019

I. Executive Summary

Breast cancer is the most common cancer among women and one of the major causes of death among women worldwide. When detected in its early stages, there is a 30% chance that the cancer can be treated effectively, but the late detection of advanced-stage tumors makes the treatment more difficult.
Currently, FNA (Fine Needle Aspiration) with visual interpretation (65% to 98% correctness) is one of the most used techniques to detect breast cancer in early stages.
This case study discusses a diagnosis technique that uses the FNA with computational interpretation via machine learning and aims to create a classifier that provides a high level of accuracy, with a low rate of false-negatives.

The dataset is provided with several physical metrics regarding breast cancer tumors with the goal of predicting malignancy. Features are computed from a digitized image of FNA of a breast mass for breast cancer patients. They describe characteristics of the cell nuclei present in the image.
Attribute Information:
1) Patient ID number
2) Diagnosis (M = malignant, B = benign)
3-32) Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)

Response Class distribution: 357 benign, 212 malignant.

Summary of Data: Appendix B

Objectives:

- To identify the best suited model for predicting the class (B or M) of the patient
- To find important features from the data with respect to explainability and predictive power
- To discuss the tradeoffs of different methods

The response is converted to 1 if M(Malignant) else 0 if B(Benign).
We split the data into 398 cases of train set (70%) and 171 cases of test set (30%).

The correlation plot between different predictors is shown in Appendix A. We observe multicollinearity between some of the variables. For example, radius_mean is highly correlated with perimeter_mean and area_mean which makes sense because area is square of radius and perimeter is twice of radius multiplied by pi. Radius_mean is also correlated with radius_worst, perimeter_worst and area_worst.

Though the correlation plots help in locating highly correlated predictors, with the help of boxplots between each predictor and the response in Appendix A we can detect one more issue as to which variables are perfectly separating the 2 cases of the response variable.
We eliminate the variables that perfectly separate the IQR ranges of the 2 cases of the response variable since the logistic regression model can not handle such a scenario on its own.
Logisitc Regression: We model the response with the remaining 15 variables that result from filtering the variables through boxplot. From the logistic model we remove insignificant terms by backward elimination manually or through stepAIC and the final model has 8 variables.
Predicting on the test set gives us an error rate of 10.5%(accuracy of 89.4%). The misclassification rate for class M is 17% which is more and undesired for such sensitive study.
If we predict on the model with 15 variables without removing the insignificant ones, we get a slightly better accuracy and an error rate of 9.35%, misclassification rate of class M is down to 15.4%.
Multicollinearity: If we check multicollinearity (VIF values) for the model with 8 significant variables, we see that all are <=10 VIF so this meets the assumption of multicollinearity for logistic model.
The interpretation is simpler, in terms of log odds but the model accuracy is not satisfactory, and we need to search for better models

Decision Tree: The model with just 8 significant variables (from stepAIC) gave an error rate of 15.78% (accuracy of 84%) when pruned with best tree size = 6 and misclassification rate of class M is 19.4%. The most important variables seem to be concave.points_se and texture_worst which are the nodes of first split and second split respectively.

Interpretation: The interpretation is pretty straightforward and easy. The patient id with concave.points_se above 0.0092 units, texture_worst above 23.55, texture_se below 1.32 could be diagnosed as having malignant tumor whereas ones with texture_se above 1.32, whose concave.points_se are above 0.014 and smoothness_mean above 0.086 are predicted to have malignant tumor. Appendix D

With 15 variables (that we get after filtering out ones that separated the response class perfectly) we get an error rate of 11% (better accuracy of 88.9%) when pruned with best size = 19 and misclassification rate of class M is down to 15%.

Random Forest: The model with 15 variables have an OOB error rate of 11.8%, misclassification rate for class M is 21% The prediction gives a test error rate of 7% (accuracy of 92.9%), which is interesting, misclassification rate for class M is 0% which means it perfectly predicts which patients have malignant tumor.
The most important variables as shown in Appendix E are concave.points_se and texture_worst as observed in decision tree model.
We further model this with optimal hyper parameters ntree=800 and mtry=4. The OOB error rate has slightly reduced to 11.31%, Class M error rate is down to 19%, Test error rate is down to 5.8% (accuracy of 94%) Class M error rate for test set is 0% which means it perfectly predicts which patients have malignant tumor.
Random Forest model could be one of the best models, not only is the accuracy good, the class M error rate is 0% hence it perfectly predicts which patients have malignant tumor.

SVM: Linear SVM with 15 variables gives an error rate of 9.94% (accuracy of 90.06%) but Class M error rate of 17%.
The radial SVM however gives an error rate of 8% but the class M error rate is only 6.7% hence radial is better than linear SVM.

When SVM is run with all the 30 variables we get an error rate of only 0.6% (accuracy of 99.4%) and class M error rate of 0% which means the model perfectly predicts the patients with malignant tumor. Among all the models used so far radial SVM model with all variables predicts the best because of high prediction accuracy and also 0% class M error rate. Random Forest also has 0% class M error rate but has slightly lower accuracy compared to radial SVM.

II. The Problem

A. Introduction/Background

Breast cancer is the most common cancer among women and one of the major causes of death among women worldwide. When detected in its early stages, there is a 30% chance that the cancer can be treated effectively, but the late detection of advanced-stage tumors makes the treatment more difficult.
Currently, FNA (Fine Needle Aspiration) with visual interpretation (65% to 98% correctness) is one of the most used techniques to detect breast cancer in early stages
This case study discusses a diagnosis technique that uses the FNA with computational interpretation via machine learning and aims to create a classifier that provides a high level of accuracy, with a low rate of false-negatives.

The dataset is provided with several physical metrics regarding breast cancer tumors with the goal of predicting malignancy. Features are computed from a digitized image of FNA of a breast mass for breast cancer patients. They describe characteristics of the cell nuclei present in the image.

B. Purpose of study/importance of study/statement of problem

- To identify the best suited model for predicting the class (B or M) of the patient
- To find important features from the data with respect to explainability and predictive power
- To discuss the tradeoffs of different methods

C. Questions to be answered/conceptual statement of hypotheses

- Which is the best suited model for predicting class(B or M) of the patient
- What are the important features from the data with respect to explainability and predictive power
- Tradeoffs of different methods

D. Outline of remainder of report (brief)

- Procedure followed to model the response variable on train set using for the different models.
- Assessing the performance of the model using test set
- Comparing prediction accuracies of different models

III. Review of Related Literature

A. Acquaint reader with existing methodologies used in this area.

Several papers were published during the last 20 years trying to achieve the best performance for the computational interpretation of FNA samples, including two well-known machine learning techniques: Bayesian Networks and J48

Kharya et al., (2014) used Naive Bayes (NB) algorithm for breast cancer detection and demonstrated the accuracy results as 93%. However evaluation and improvement measures for NB algorithm has not been proposed by the researchers. Chaurasia et al., (2018) compared algorithms like NB, Radial Basis Function Network and J48 for breast cancer prediction and proved the performance of NB algorithm. Stojadinovic et al., (2010) applied NB algorithm for breast cancer risk stratification. Mandal et al., (2017) analyzed the performance of NB, Logistic Regression and Decision Tree for breast cancer detection and proved the performance of NB classifier. Huang et al., (2017) compared Support Vector Machines (SVM) and SVM based ensemble method. The researchers proved the performance of SVM based ensemble and suggested the usage of boosting method with machine learning techniques for better performance and accuracy. Jing et al., (2008) proposed Boosted Bayesian Network classifier for breast cancer classification. Most of the researchers suggested the usage of NB based classifiers for breast cancer prediction (Asian Pac J Cancer Prev. 2018; 19(10): 2917–2920 n.d.)

IV. Methodology

A. Identification, classification and operationalization of variables.

- The dataset is provided with several physical metrics (32) regarding breast cancer tumors with the goal of predicting malignancy.
- We initially split the data into 70% train set and 30% test set.
- Summary of Data: Appendix B

B. Statements of hypotheses being tested and/or models being developed.

- To identify the best suited model for predicting the class (B or M) of the patient
- To find important features from the data with respect to explainability and predictive power
- To discuss the tradeoffs of different methods

C. Sampling techniques, if full data is not being used.

We split the data into 398 cases of train set (70%) and 171 cases of test set (30%). Appendix B

D. Data collection process, including data sources, data size, etc. Primary/secondary?

The dataset is secondary source of data publicly available and was created by Dr. William H. Wolberg, physician at the University of Wisconsin Hospital at Madison, Wisconsin, USA. The dataset has 598 cases of breast cancer patients with benign and malignant tumor.

E. Modeling analysis/techniques used

Different modelling techniques such as Decision Trees, Random Forests and support vector machines are used to test for prediction accuracy, appropriateness of each models. Appendix C - Appendix F

F. Methodological assumptions and limitations.

The decision Tree, Random forest and SVM are non-parametric method which do not require any assumptions. Decision Tree is easy to interpret but random forest and SVM have better accuracy.

V. Data

A. Data cleaning Appendix A

There are no missing values in the datasets. The response is converted to 1 if M(Malignant) else 0 if B(Benign).

B. Data preprocessing Appendix B

The correlation plot between different predictors is shown in Appendix A. Though the correlation plots help in locating highly correlated predictors, with the help of boxplots between each predictor and the response in Appendix A we can detect one more issue as to which variables are perfectly separating the 2 cases of the response variable.
We split the data into 398 cases of train set (70%) and 171 cases of test set (30%).

C. Data Limitations

There are only around 600 observations which makes the model built on such few records less reliable. Also, there are many variables that are highly correlated so the information that these variables provide about the response is redundant in the presence of the other variables.

VI. Findings (Results)

A. Results presented in tables or charts when appropriate

Qualitative analysis of relationship between explanatory variables analysed through correlation plots Appendix A. With the help of boxplots between each predictor and the response in we can detect one more issue as to which variables are perfectly separating the 2 cases of the response variable.

B. Results reported with respect to hypotheses/models.

Logisitc Regression: With the help of boxplot we filter the variables that perfectly separate the IQR ranges of the 2 cases of the response variables. We model the response with the remaining 15 variables. From the logistic model we remove insignificant terms by backward elimination manually or through stepAIC and the final model has 8 variables. Appendix C
Predicting on the test set gives us an error rate of 10.5% (accuracy of 89.4%). The misclassification rate for class M is 17% which is more and undesired for such sensitive study.
If we predict on the model with 15 variables without removing the insignificant ones, we get a slightly better accuracy and an error rate of 9.35%, misclassification rate of class M is down to 15.4%.
Multicollinearity: If we check multicollinearity (VIF values) for the model with 8 significant variables, we see that all are <=10 so this meets the assumption of multicollinearity for logistic model.
The interpretation is simpler, in terms of log odds but the model accuracy is not satisfactory, and we need to search for better models

Decision Tree: The model with just 8 significant variables (from stepAIC) gave an error rate of 15.78% (accuracy of 84%) when pruned with best tree size = 6 and

misclassification rate of class M is 19.4%. The most important variables seem to be concave.points_se and texture_worst which are the nodes of first split and second split respectively.

Interpretation: The interpretation is pretty straightforward and easy. The patient id with concave.points_se above 0.0092 units, texture_worst above 23.55, texture_se below 1.32 could be diagnosed as having malignant tumor whereas ones with texture_se above 1.32, whose concave.points_se are above 0.014 and smoothness_mean above 0.086 are predicted to have malignant tumor. Appendix D

With 15 variables (that we get after filtering out ones that separated the response class perfectly) we get an error rate of 11% (better accuracy of 88.9%) when pruned with best size = 19 and misclassification rate of class M is down to 15%.

Random Forest: The model with 15 variables have an OOB error rate of 11.8%, misclassification rate for class M is 21% The prediction gives a test error rate of 7% (accuracy of 92.9%), which is interesting, misclassification rate for class M is 0% which means it perfectly predicts which patients have malignant tumor.
The most important variables as shown in Appendix E are concave.points_se and texture_worst as observed in decision tree model.
We further model this with optimal hyper parameters ntree=800 and mtry=4. The OOB error rate has slightly reduced to 11.31%, Class M error rate is down to 19%, Test error rate is down to 5.8% (accuracy of 94%) Class M error rate for test set is 0% which means it perfectly predicts which patients have malignant tumor.
Random Forest model could be one of the best models, not only is the accuracy good, the class M error rate is 0% hence it perfectly predicts which patients have malignant tumor.

SVM: Linear SVM with 15 variables gives an error rate of 9.94% (accuracy of 90.06%) but Class M error rate of 17%. Appendix F
The radial SVM however gives an error rate of 8% but the class M error rate is only 6.7% hence radial is better than linear SVM.
When SVM is run with all the 32 variables we get an error rate of only 0.6% (accuracy of 99.4%) and class M error rate of 0% which means the model perfectly predicts the patients with malignant tumor. Among all the models used so far radial SVM model with all variables predicts the best because of high prediction accuracy and also 0% class M error rate. Random Forest also has 0% class M error rate, but has slightly lower accuracy compared to radial SVM.

C. Factual information kept separate from interpretation, inference and evaluation.

During 2009-2013, the age-adjusted incidence rate for invasive female breast cancer was 127.2 per 100,000 population, reporting approx.4300 newly diagnosed breast cancers annually. The rate among black women increased by 23% per 100,000in 1995 to 133.6 per 100,000 in 2013. Whereas the breast cancer mortality rate among white women decreased by 33%. (Wisconsin cancer reporting system, office of

health informatics, division of public health, department of health sciences and national center for health sciences n.d.)

VII. Conclusions and Recommendations

We conclude that though logistic model is better at interpreting the estimates the accuracy can be improved with other non-parametric models. Also logistic regression cannot be run with all the variables provided; few variables were eliminated from the model to avoid warnings of clearly separating the cases of response variable

Among all the models used so far radial SVM model with all variables predicts the best because of high prediction accuracy and also 0% class M error rate. Random Forest also has 0% class M error rate, but has slightly lower accuracy compared to radial SVM.

SVM predicts better than all others tested so far but theirs a trade off in terms interpretability. I recommend SVM model if accuracy is the main goal.

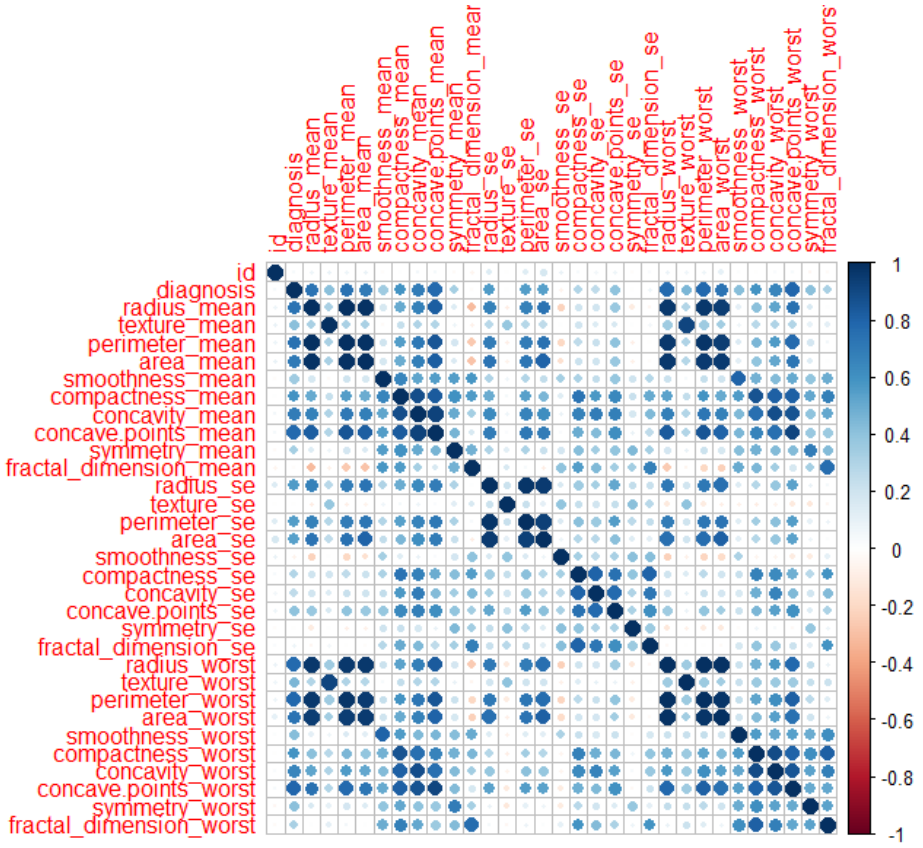
Also, there are many other classifiers to test, such as Tree Augmented Naive Bayes (TAN), Boosted Augmented Naive Bayes (BAN) and Bayes Belief Network (BBN), KNN, etc.

Appendix A

```
cancerData = read.csv("C:/Users/prady_000/Documents/Vishwa/MSDA/DAA/Analytics
in medicine case study/CancerData.csv", sep= ',',header=TRUE)

CD$diagnosis = ifelse(CD$diagnosis=="M",1,0)
```

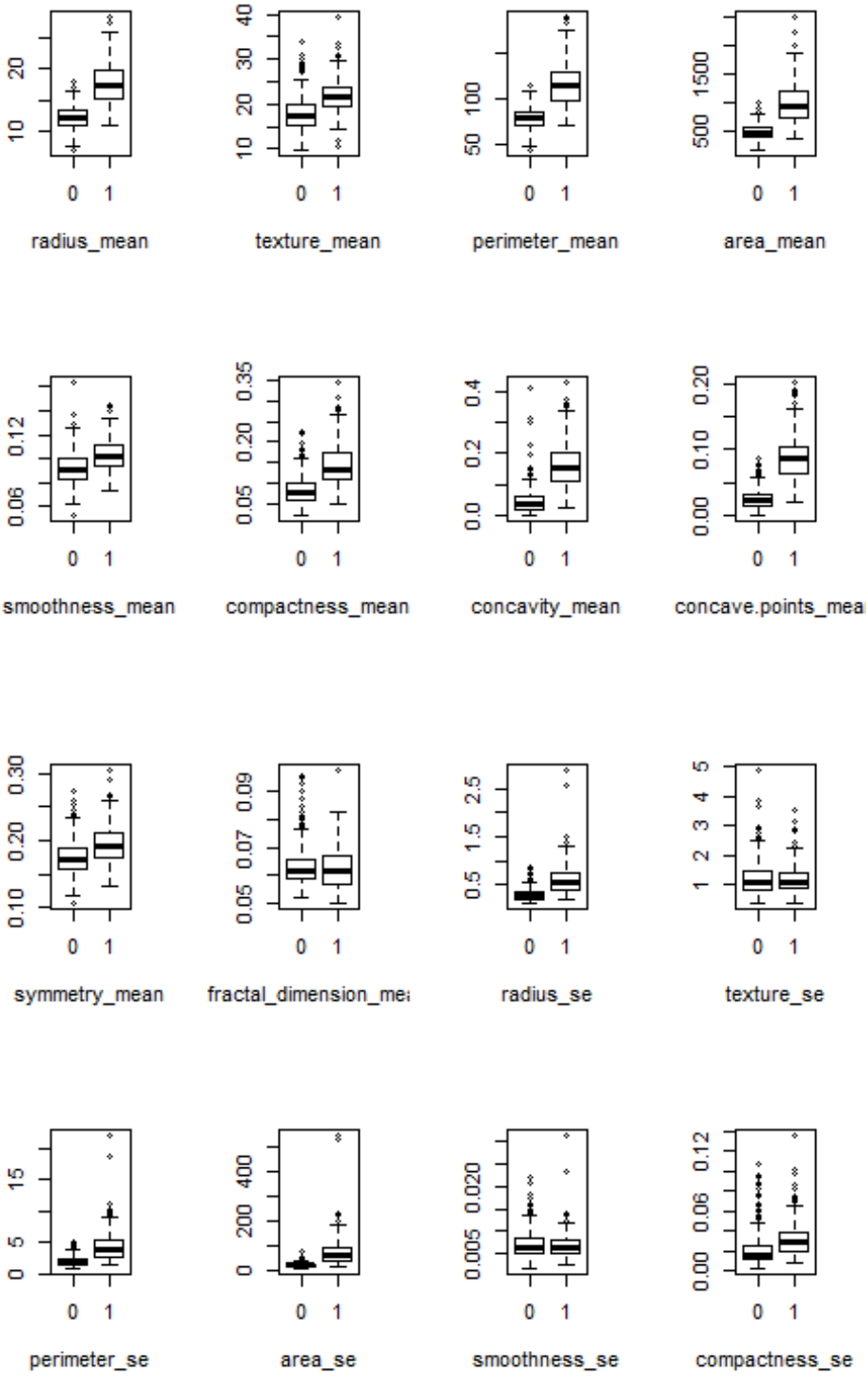
Correlation plot to check for multicollinearity

```
co = cor(CD)
corrplot(co)
```



Boxplots between the response and the explanatory variables

```
#to check if any of the variables are perfectly seperating the cases of respo
nse
par(mfrow=c(2,4))
for (i in 3:32){
 boxplot(CD[,i]~CD$diagnosis,data=CD,xlab=names(CD[i]))
}
```


radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave.points_mea
symmetry_mean
fractal_dimension_mea
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se

Appendix B

```
summary(CD)
```

```
##        id              diagnosis       radius_mean      texture_mean
##  Min.   :     8670   Min.   :0.0000   Min.   : 6.981   Min.   : 9.71
##  1st Qu.:   869218   1st Qu.:0.0000   1st Qu.:11.700   1st Qu.:16.17
##  Median :   906024   Median :0.0000   Median :13.370   Median :18.84
##  Mean   : 30371831   Mean   :0.3726   Mean   :14.127   Mean   :19.29
##  3rd Qu.:  8813129   3rd Qu.:1.0000   3rd Qu.:15.780   3rd Qu.:21.80
##  Max.   :911320502   Max.   :1.0000   Max.   :28.110   Max.   :39.28
##  perimeter_mean     area_mean      smoothness_mean   compactness_mean
##  Min.   : 43.79   Min.   : 143.5   Min.   :0.05263   Min.   :0.01938
##  1st Qu.: 75.17   1st Qu.: 420.3   1st Qu.:0.08637   1st Qu.:0.06492
##  Median : 86.24   Median : 551.1   Median :0.09587   Median :0.09263
##  Mean   : 91.97   Mean   : 654.9   Mean   :0.09636   Mean   :0.10434
##  3rd Qu.:104.10   3rd Qu.: 782.7   3rd Qu.:0.10530   3rd Qu.:0.13040
##  Max.   :188.50   Max.   :2501.0   Max.   :0.16340   Max.   :0.34540
##  concavity_mean    concave.points_mean symmetry_mean
##  Min.   :0.00000   Min.   :0.00000     Min.   :0.1060
##  1st Qu.:0.02956   1st Qu.:0.02031     1st Qu.:0.1619
##  Median :0.06154   Median :0.03350     Median :0.1792
##  Mean   :0.08880   Mean   :0.04892     Mean   :0.1812
##  3rd Qu.:0.13070   3rd Qu.:0.07400     3rd Qu.:0.1957
##  Max.   :0.42680   Max.   :0.20120     Max.   :0.3040
##  fractal_dimension_mean   radius_se        texture_se      perimeter_se
##  Min.   :0.04996        Min.   :0.1115   Min.   :0.3602   Min.   : 0.757
##  1st Qu.:0.05770        1st Qu.:0.2324   1st Qu.:0.8339   1st Qu.: 1.606
##  Median :0.06154        Median :0.3242   Median :1.1080   Median : 2.287
##  Mean   :0.06280        Mean   :0.4052   Mean   :1.2169   Mean   : 2.866
##  3rd Qu.:0.06612        3rd Qu.:0.4789   3rd Qu.:1.4740   3rd Qu.: 3.357
##  Max.   :0.09744        Max.   :2.8730   Max.   :4.8850   Max.   :21.980
##     area_se        smoothness_se      compactness_se      concavity_se
##  Min.   :  6.802   Min.   :0.001713   Min.   :0.002252   Min.   :0.00000
##  1st Qu.: 17.850   1st Qu.:0.005169   1st Qu.:0.013080   1st Qu.:0.01509
##  Median : 24.530   Median :0.006380   Median :0.020450   Median :0.02589
##  Mean   : 40.337   Mean   :0.007041   Mean   :0.025478   Mean   :0.03189
##  3rd Qu.: 45.190   3rd Qu.:0.008146   3rd Qu.:0.032450   3rd Qu.:0.04205
##  Max.   :542.200   Max.   :0.031130   Max.   :0.135400   Max.   :0.39600
##  concave.points_se   symmetry_se       fractal_dimension_se
##  Min.   :0.000000   Min.   :0.007882   Min.   :0.0008948
##  1st Qu.:0.007638   1st Qu.:0.015160   1st Qu.:0.0022480
##  Median :0.010930   Median :0.018730   Median :0.0031870
##  Mean   :0.011796   Mean   :0.020542   Mean   :0.0037949
##  3rd Qu.:0.014710   3rd Qu.:0.023480   3rd Qu.:0.0045580
##  Max.   :0.052790   Max.   :0.078950   Max.   :0.0298400
##   radius_worst   texture_worst   perimeter_worst    area_worst
##  Min.   : 7.93   Min.   :12.02   Min.   : 50.41   Min.   : 185.2
##  1st Qu.:13.01   1st Qu.:21.08   1st Qu.: 84.11   1st Qu.: 515.3
##  Median :14.97   Median :25.41   Median : 97.66   Median : 686.5
##  Mean   :16.27   Mean   :25.68   Mean   :107.26   Mean   : 880.6
##  3rd Qu.:18.79   3rd Qu.:29.72   3rd Qu.:125.40   3rd Qu.:1084.0
##  Max.   :36.04   Max.   :49.54   Max.   :251.20   Max.   :4254.0
##  smoothness_worst  compactness_worst concavity_worst  concave.points_worst
```

```
##  Min.   :0.07117   Min.   :0.02729   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.11660   1st Qu.:0.14720   1st Qu.:0.1145   1st Qu.:0.06493
##  Median :0.13130   Median :0.21190   Median :0.2267   Median :0.09993
##  Mean   :0.13237   Mean   :0.25427   Mean   :0.2722   Mean   :0.11461
##  3rd Qu.:0.14600   3rd Qu.:0.33910   3rd Qu.:0.3829   3rd Qu.:0.16140
##  Max.   :0.22260   Max.   :1.05800   Max.   :1.2520   Max.   :0.29100
##  symmetry_worst   fractal_dimension_worst
##  Min.   :0.1565   Min.   :0.05504
##  1st Qu.:0.2504   1st Qu.:0.07146
##  Median :0.2822   Median :0.08004
##  Mean   :0.2901   Mean   :0.08395
##  3rd Qu.:0.3179   3rd Qu.:0.09208
##  Max.   :0.6638   Max.   :0.20750
```

```
#splitting into train and test sets
set.seed(123)
train = sample(1:nrow(CD),nrow(CD)*0.7,rep=F)
test=-train

library(tree)
library(randomForest)

library(corrplot)

library(MASS)

library(dplyr)

library(caret)

library(car)

library(e1071
```

Appendix C

Logit Model

```
log.model1=glm(diagnosis~texture_mean+smoothness_mean+symmetry_mean+fractal_d
imension_mean+texture_se+smoothness_se+compactness_se+
                 concavity_se+concave.points_se+symmetry_se+fractal_dimension
_se+
                 texture_worst+smoothness_worst+symmetry_worst+fractal_dimens
ion_worst,data=CD[train,],family="binomial")
summary(log.model1)

##
## Call:
## glm(formula = diagnosis ~ texture_mean + smoothness_mean + symmetry_mean +
##     fractal_dimension_mean + texture_se + smoothness_se + compactness_se +
##     concavity_se + concave.points_se + symmetry_se + fractal_dimension_se
+
```

```
##     texture_worst + smoothness_worst + symmetry_worst + fractal_dimension_
worst,
##     family = "binomial", data = CD[train, ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2320  -0.2069  -0.0264   0.0972   3.5190
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -1.936e+00  3.379e+00  -0.573  0.56676
## texture_mean              6.646e-03  1.753e-01   0.038  0.96977
## smoothness_mean           2.140e+02  5.237e+01   4.087 4.37e-05 ***
## symmetry_mean            -7.886e+00  1.855e+01  -0.425  0.67068
## fractal_dimension_mean   -7.567e+02  1.143e+02  -6.621 3.57e-11 ***
## texture_se               -2.015e+00  1.174e+00  -1.717  0.08602 .
## smoothness_se            -2.551e+02  2.073e+02  -1.230  0.21851
## compactness_se            2.366e+01  3.332e+01   0.710  0.47757
## concavity_se             -1.120e+01  1.134e+01  -0.988  0.32300
## concave.points_se         4.042e+02  8.125e+01   4.975 6.54e-07 ***
## symmetry_se               3.650e+01  7.009e+01   0.521  0.60254
## fractal_dimension_se      3.450e+01  3.394e+02   0.102  0.91905
## texture_worst             3.225e-01  1.704e-01   1.893  0.05837 .
## smoothness_worst          1.989e+01  3.890e+01   0.511  0.60914
## symmetry_worst            1.031e+01  1.283e+01   0.804  0.42130
## fractal_dimension_worst   1.506e+02  4.836e+01   3.113  0.00185 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 524.25  on 397  degrees of freedom
## Residual deviance: 141.00  on 382  degrees of freedom
## AIC: 173
##
## Number of Fisher Scoring iterations: 7
```

```
log.model2=glm(diagnosis~smoothness_mean+fractal_dimension_mean+texture_se+
               concave.points_se+smoothness_se+
                 texture_worst+symmetry_worst+fractal_dimension_worst,data=CD
[train,],family="binomial")
summary(log.model2)
```

```
##
## Call:
## glm(formula = diagnosis ~ smoothness_mean + fractal_dimension_mean +
##     texture_se + concave.points_se + smoothness_se + texture_worst +
##     symmetry_worst + fractal_dimension_worst, family = "binomial",
##     data = CD[train, ])
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1940  -0.2138  -0.0290   0.1100   3.4013
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -2.90888    3.00766  -0.967   0.3335
## smoothness_mean           228.36425   36.02466   6.339 2.31e-10 ***
## fractal_dimension_mean   -756.44199  110.99578  -6.815 9.42e-12 ***
## texture_se                 -1.83978    0.80272  -2.292   0.0219 *
## concave.points_se         394.47424   62.58148   6.303 2.91e-10 ***
## smoothness_se            -189.43481  125.10089  -1.514   0.1300
## texture_worst               0.32948    0.06373   5.170 2.35e-07 ***
## symmetry_worst             10.51704    4.64507   2.264   0.0236 *
## fractal_dimension_worst   165.13017   32.96731   5.009 5.47e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 524.25  on 397  degrees of freedom
## Residual deviance: 142.92  on 389  degrees of freedom
## AIC: 160.92
##
## Number of Fisher Scoring iterations: 7

stepAIC(log.model1,direction="backward",trace=F)

##
## Call:  glm(formula = diagnosis ~ smoothness_mean + fractal_dimension_mean
+
##     texture_se + smoothness_se + concave.points_se + texture_worst +
##     symmetry_worst + fractal_dimension_worst, family = "binomial",
##     data = CD[train, ])
##
## Coefficients:
##             (Intercept)          smoothness_mean   fractal_dimension_mean
##                 -2.9089                 228.3642                -756.4420
##              texture_se            smoothness_se        concave.points_se
##                 -1.8398                -189.4348                 394.4742
##           texture_worst          symmetry_worst  fractal_dimension_worst
##                  0.3295                 10.5170                 165.1302
##
## Degrees of Freedom: 397 Total (i.e. Null);  389 Residual
## Null Deviance:       524.3
## Residual Deviance: 142.9     AIC: 160.9

pred.log= predict.glm(log.model2,newdata=CD[test,],type="response")
PredDiagnosis = ifelse(pred.log >= 0.5,1,0)
```

```
caret::confusionMatrix(as.factor(CD[test,]$diagnosis),as.factor(PredDiagnosis
))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 99  7
##          1 11 54
##
##                Accuracy : 0.8947
##                  95% CI : (0.8387, 0.9364)
##     No Information Rate : 0.6433
##     P-Value [Acc > NIR] : 5.16e-14
##
##                   Kappa : 0.7739
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 0.9000
##             Specificity : 0.8852
##          Pos Pred Value : 0.9340
##          Neg Pred Value : 0.8308
##              Prevalence : 0.6433
##          Detection Rate : 0.5789
##    Detection Prevalence : 0.6199
##       Balanced Accuracy : 0.8926
##
##        'Positive' Class : 0
##
```

```
log.acc = mean(PredDiagnosis==CD[test,]$diagnosis)*100;log.acc
```

```
## [1] 89.47368
```

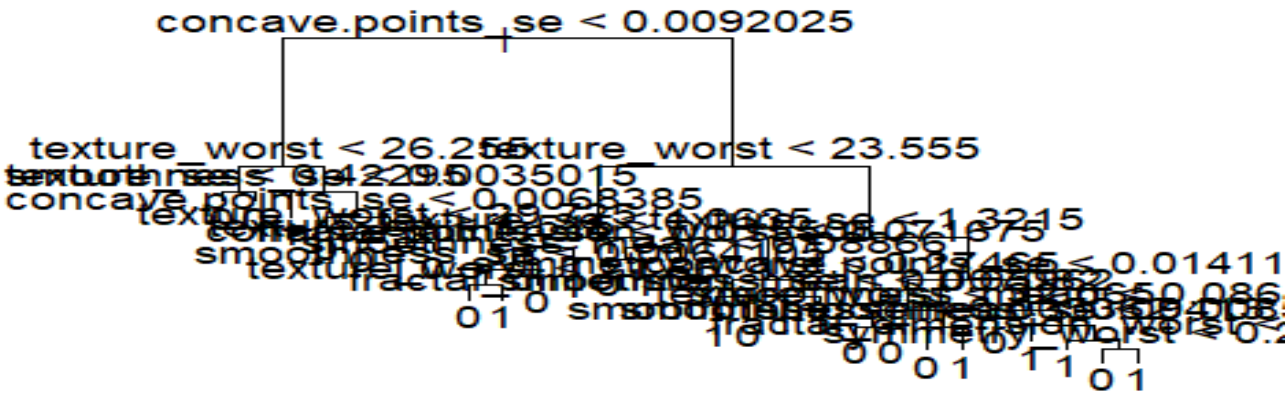
```
mulcol = car::vif(log.model2);mulcol
```

```
##         smoothness_mean  fractal_dimension_mean              texture_se
##                4.543869               10.655500                3.709895
##       concave.points_se           smoothness_se           texture_worst
##                2.204709                3.003860                2.810422
##         symmetry_worst fractal_dimension_worst
##                1.341281                5.601450
```

Appendix D

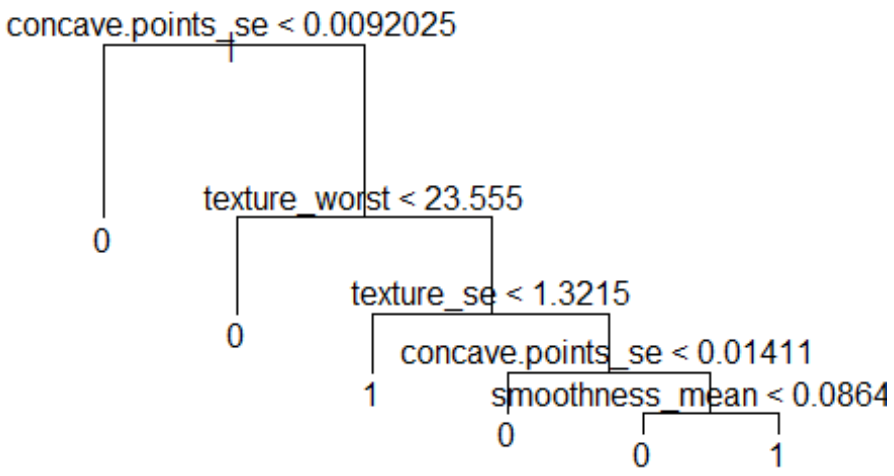
Decision Tree

```
tree.model <- tree(as.factor(diagnosis)~smoothness_mean+fractal_dimension_mea
n+texture_se+
                concave.points_se+smoothness_se+ texture_worst+symmetry_worst+
fractal_dimension_worst, data= CD[train,])


plot(tree.model)

text(tree.model, pretty = 0, cex= 1.1)
```




```
set.seed(123)
cv.CD = cv.tree(tree.model, FUN = prune.misclass)
cv.CD

## $size
##  [1] 27 24 22 19 16 10  6  5  3  1
##
## $dev
##  [1]  85  85  86  81  79  75  72  79 103 155
##
## $k
##  [1]      -Inf  0.000000  0.500000  1.333333  1.666667  2.500000  3.500000
##  [8]  4.000000 10.500000 32.000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
prune.CD = prune.misclass(tree.model, best = 6)
plot(prune.CD)
text(prune.CD, pretty = 0)
```




```
tree.pred1 = predict(prune.CD, CD[test,], type = "class")
table(tree.pred1, CD[test,]$diagnosis)
```

```
##
## tree.pred1  0  1
##          0 94 15
##          1 12 50
```

```
tree.acc1 = mean(tree.pred1==CD[test,]$diagnosis)*100;tree.acc1
```

```
## [1] 84.21053
```

```
tree.model1 <- tree(as.factor(diagnosis)~texture_mean+smoothness_mean+symmetr
y_mean+fractal_dimension_mean+texture_se+smoothness_se+compactness_se+
                  concavity_se+concave.points_se+symmetry_se+fractal_dimension
_se+
                  texture_worst+smoothness_worst+symmetry_worst+fractal_dimens
ion_worst, data= CD[train,])

set.seed(123)
cv.CD1 = cv.tree(tree.model1, FUN = prune.misclass)
cv.CD1
```

```
## $size
## [1] 22 19 17 12  8  6  4  3  1
##
## $dev
## [1]  83  82  86  91  90  90  96 108 161
##
## $k
## [1] -Inf  0.0  1.0  2.0  2.5  3.0  7.5 16.0 32.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
prune.CD1 = prune.misclass(tree.model1, best = 19)


tree.pred2 = predict(prune.CD1, CD[test,], type = "class")
table(tree.pred2, CD[test,]$diagnosis)
```

```
##
## tree.pred2  0  1
##          0 96  9
##          1 10 56
```

```
tree.acc2 = mean(tree.pred2==CD[test,]$diagnosis)*100;tree.acc2
```

```
## [1] 88.88889
```

Appendix E

Random Forest

```
set.seed(123)
forest1 <- randomForest(as.factor(diagnosis)~texture_mean+smoothness_mean+sym
metry_mean+fractal_dimension_mean+texture_se+smoothness_se+compactness_se+
                  concavity_se+concave.points_se+symmetry_se+fractal_dimension
_se+
                  texture_worst+smoothness_worst+symmetry_worst+fractal_dimens
ion_worst, data = CD[train,],importance = T)
forest1
```

```
##
## Call:
##  randomForest(formula = as.factor(diagnosis) ~ texture_mean +      smoothn
ess_mean + symmetry_mean + fractal_dimension_mean +      texture_se + smoothn
ess_se + compactness_se + concavity_se +      concave.points_se + symmetry_se
+ fractal_dimension_se +      texture_worst + smoothness_worst + symmetry_wor
st + fractal_dimension_worst,      data = CD[train, ], importance = T)
```

```
##                 Type of random forest: classification
##                       Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 11.81%
## Confusion matrix:
##     0   1 class.error
## 0 235  16  0.06374502
## 1  31 116  0.21088435
```

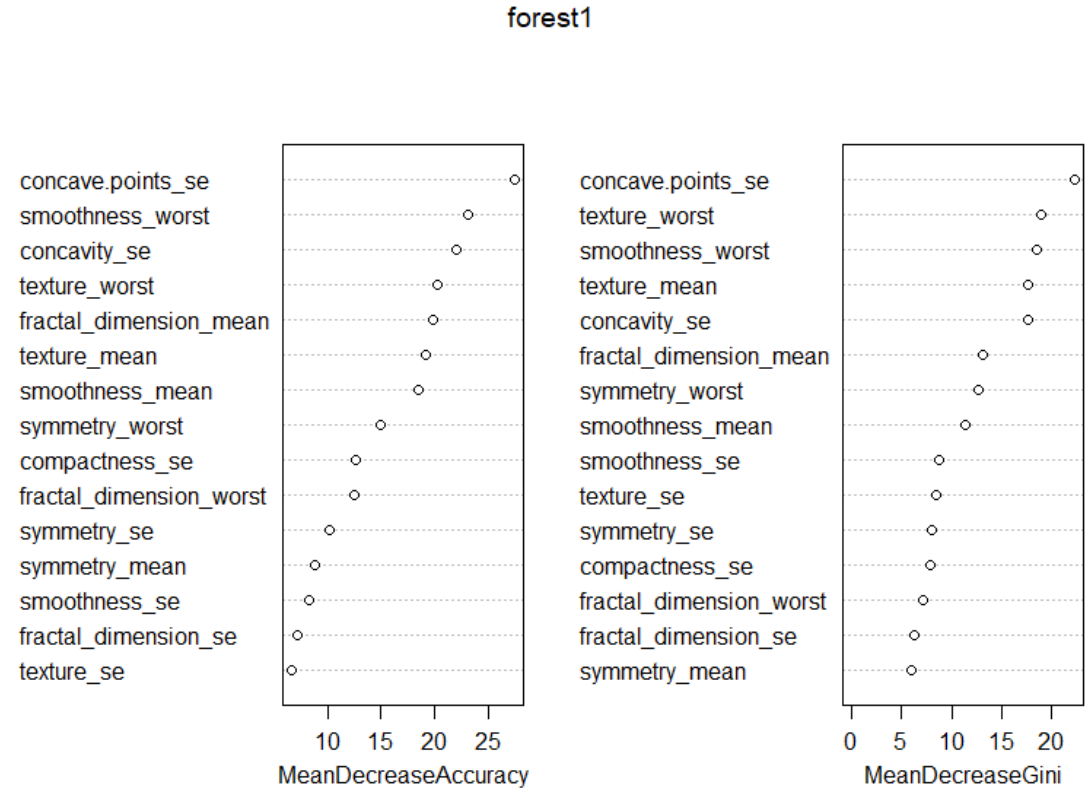
```
  predRFun = predict(forest1, newdata = CD[test,], type="class")
  table(predRFun, CD[test,]$diagnosis)
```

```
##
## predRFun   0   1
##        0 106  12
##        1   0  53
```

```
  accRFun = mean(predRFun == CD[test,]$diagnosis)*100;accRFun
```

```
## [1] 92.98246
```

```
varImpPlot(forest1)
```




```
set.seed(123)
  # Grid search for opt mtry and ntree
```

```r
  mtry.values <- seq(2,14,2)
  ntree.values <- seq(500,1000,100)

  # Create a data frame containing all combinations
  hyper_grid <- expand.grid(mtry = mtry.values, ntree = ntree.values)

  # Create an empty vector to store OOB error values
  oob_err <- c()

  # Write a loop over the rows of hyper_grid to train the grid of models
  for (i in 1:nrow(hyper_grid)) {

    # Train a Random Forest model
    set.seed(123)
    model <- randomForest(as.factor(diagnosis)~texture_mean+smoothness_mean+s
ymmetry_mean+fractal_dimension_mean+texture_se+smoothness_se+compactness_se+
                  concavity_se+concave.points_se+symmetry_se+fractal_dimension
_se+
                  texture_worst+smoothness_worst+symmetry_worst+fractal_dimens
ion_worst, data = CD[train,], importance = T, mtry = hyper_grid$mtry[i],
                           ntree = hyper_grid$ntree[i])

    # Store OOB error for the model
    oob_err[i] <- model$err.rate[model$ntree]

  }
  # Find location of opt index in grid search
  opt_i <- which.min(oob_err)
  hyper_grid[opt_i,]
```

```
##    mtry ntree
## 23    4   800
```

```r
  # Run random forest with opt mtry and ntree, predict on test set, and evalu
ate MSE.
  set.seed(123)
  resultsRF = randomForest(as.factor(diagnosis)~texture_mean+smoothness_mean+
symmetry_mean+fractal_dimension_mean+texture_se+smoothness_se+compactness_se+
                  concavity_se+concave.points_se+symmetry_se+fractal_dimension
_se+
                  texture_worst+smoothness_worst+symmetry_worst+fractal_dimens
ion_worst, data = CD[train,], mtry = hyper_grid$mtry[opt_i],
                           ntree = hyper_grid$ntree[opt_i],importance = T)
  resultsRF
```

```
##
## Call:
##  randomForest(formula = as.factor(diagnosis) ~ texture_mean +      smoothn
ess_mean + symmetry_mean + fractal_dimension_mean +      texture_se + smoothn
ess_se + compactness_se + concavity_se +      concave.points_se + symmetry_se
```

```
+ fractal_dimension_se +      texture_worst + smoothness_worst + symmetry_wor
st + fractal_dimension_worst,      data = CD[train, ], mtry = hyper_grid$mtry
[opt_i], ntree = hyper_grid$ntree[opt_i],      importance = T)
##                Type of random forest: classification
##                      Number of trees: 800
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 11.31%
## Confusion matrix:
##     0   1 class.error
## 0 235  16  0.06374502
## 1  29 118  0.19727891
```

```
  predRF = predict(resultsRF, newdata = CD[test,])
  table(predRF,CD[test,]$diagnosis)
```

```
##
## predRF   0   1
##      0 106  10
##      1   0  55
```

```
  hyper.acc = mean(predRF==CD[test,]$diagnosis)*100;hyper.acc
```

```
## [1] 94.15205
```

Appendix F

SVM

```
set.seed(123)
model_svm = as.factor(diagnosis)~texture_mean+smoothness_mean+symmetry_mean+f
ractal_dimension_mean+texture_se+smoothness_se+compactness_se+
                concavity_se+concave.points_se+symmetry_se+fractal_dimension
_se+
                texture_worst+smoothness_worst+symmetry_worst+fractal_dimens
ion_worst
CD.tune=tune.svm(model_svm,data=CD[train,],cost=seq(0.1, 1, by = 0.1),kernel=
"linear") # checking best cost by 10 fold cv
summary(CD.tune$best.model)
```

```
##
## Call:
## best.svm(x = model_svm, data = CD[train, ], cost = seq(0.1, 1,
##     by = 0.1), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.4
```

```
##       gamma:  0.06666667
##
## Number of Support Vectors:  103
##
##  ( 53 50 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

Linear SVM

```
CD_svm = svm(formula = model_svm, data = CD[train,],kernel="linear",cost  = C
D.tune$best.parameters$cost)

# accessing the performance of model with test set
svm.test = predict(CD_svm, CD[test,])
caret::confusionMatrix(as.factor(CD[test,]$diagnosis),svm.test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 100   6
##          1  11  54
##
##                Accuracy : 0.9006
##                  95% CI : (0.8456, 0.941)
##     No Information Rate : 0.6491
##     P-Value [Acc > NIR] : 3.277e-14
##
##                   Kappa : 0.7859
##  Mcnemar's Test P-Value : 0.332
##
##             Sensitivity : 0.9009
##             Specificity : 0.9000
##          Pos Pred Value : 0.9434
##          Neg Pred Value : 0.8308
##              Prevalence : 0.6491
##          Detection Rate : 0.5848
##    Detection Prevalence : 0.6199
##       Balanced Accuracy : 0.9005
##
##        'Positive' Class : 0
##
```

```
#linear.acc = mean(svm.test==CD[test,]$diagnosis)*100;linear.acc
```

Radial SVM

```
set.seed(123)
radial_svm = tune.svm(model_svm,data = CD[train,], gamma = seq(.01,0.1, by =
0.01), cost = seq(0.1, 1, by = 0.1))
radial_CD = svm(formula = model_svm, data = CD[train,], gamma  = radial_svm$b
est.parameters$gamma, cost=
                    radial_svm$best.parameters$cost)

summary(radial_CD)

##
## Call:
## svm(formula = model_svm, data = CD[train, ], gamma = radial_svm$best.param
eters$gamma,
##     cost = radial_svm$best.parameters$cost)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.03
##
## Number of Support Vectors:  167
##
##  ( 85 82 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1

# accessing the performance of model with test set
svm.radial.test = predict(radial_CD, CD[test,])
table(svm.radial.test,CD[test,]$diagnosis)

##
## svm.radial.test   0   1
##               0 102  10
##               1   4  55

radial.acc = mean(svm.radial.test==CD[test,]$diagnosis)*100;radial.acc

## [1] 91.81287
```

SVM with all the variables including ones that separate the two response class

```r
set.seed(123)
radial_svm_all = tune.svm(as.factor(diagnosis)~.,data = CD[train,], gamma = s
eq(.01,0.1, by = 0.01), cost = seq(0.1, 1, by = 0.1))
radial_CD_all = svm(formula = as.factor(diagnosis)~., data = CD[train,], gamm
a  = radial_svm_all$best.parameters$gamma, cost=
                    radial_svm_all$best.parameters$cost)



# accessing the performance of model with test set
svm.radial.test_all = predict(radial_CD_all, CD[test,])
table(svm.radial.test_all,CD[test,]$diagnosis)
```

```
##
## svm.radial.test_all   0   1
##                   0 106   1
##                   1   0  64
```

```r
radial.acc_all = mean(svm.radial.test_all==CD[test,]$diagnosis)*100;radial.ac
c_all
```

```
## [1] 99.4152
```