Predict Breast Cancer as Benign or Malignant

Jeet Bhavesh Thakur, Monisha Gowda, Karan Pankaj Makhija

7/21/2020

Description of the Problem

The following project is based on the dataset "Breast Cancer Wisconsin (Diagnostic) Data Set" available at UCI Machine Learning repository donated by " Dr. William H Wolberg at University of Wisconsin ".

The project is aimed at providing a tool to assist the ongoing scientific breakthroughs in the field of medicine using technology. The problem of correctly classifying a Breast Cancer tumor as Malignant or Benign has been an issue for the experts in the industry. The tool is aimed at providing a soft accurate prediction for the specimen in about no time once it has been discovered. The problem we are trying to solve reduces both cost and time that are valuable resources for the patient and helps them get the treatment they deserve.

The Prediction of the cancer is performed by analyzing the data recovered by Dr William from a digitized image of a fine needle aspirate (FNA) of a breast mass. Breast Cancer is known to be the most common of cancer and has a higher rate of successful suppression amongst other known cancerous cells in the human body. Thus, the ultimate goal in the making of this tool is to make sure successful identification of such cells is done at the earliest to provide more time for treatment.

Let us now see the data

What is exactly the function of your system? That is, what will it do? (Supervised or Unsupervised? Regression or classification?)

Coming to the functionality of our system, this project aims to predict cancer in a patient's body using different symptoms. Ten attributes are sufficient to classify cancer as harmless or harmful. Each attribute has a numerical range of 1-10 and the class column has the value either B (Benign/Harmless) or M (Malignant/Harmful). The data we have chosen is a diagnostic one in which we are classifying the common feature, the worst of the feature, and the spread of that feature.

So with this data, we can efficiently perform classification tasks and make sure the accuracy is improved to a great extent by comparing different algorithms. We also plan to create interesting visualizations that aim at showing vibrant clusters which classifies the data into different classes.

Why would we need such a tool and who would you expect to use it and benefit from it?

Breast cancer is the most common malignancy among women, accounting for nearly 1 in 3 cancers diagnosed among women in the United States, and it is the second leading cause of cancer death among women. Breast Cancer occurs as a result of abnormal growth of cells in the breast tissue, commonly referred to as a tumor. A tumor does not mean cancer - tumors can be benign (not cancerous) or malignant (cancerous). Breast lumps are common, and most often they're benign, particularly in younger women. Tests such as MRI,

mammogram, ultrasound and biopsy are commonly used to diagnose breast cancer. It is important to have any breast lump evaluated early by a doctor. The features in the dataset are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. Using this tool, a medical professional can make a fairly accurate prediction of whether the tumor is benign or malignant.

You should mention the data and the candidate core algorithms that you will use.

To create the dataset Dr. Wolberg used fluid samples, taken from patients with solid breast masses and an easy-to-use graphical computer program called Xcyt, which is capable of performing the analysis of cytological features based on a digital scan. The program uses a curve-fitting algorithm, to compute ten features from each one of the cells in the sample, then it calculates the mean value, extreme value and standard error of each feature for the image, returning a 30 real-valued vector. All feature values are re-coded with four significant digits.

Number of attributes: 32 (ID, diagnosis, 30 real-valued input features) Attribute information

1) 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)
```

The mean, standard error and worst/largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

For this project we will be using the following algorithms:

Support Vector Machines(SVM), K-Means, Random Forest, Decision Tree and Adaboost. SVM will be our core algorithm as it works relatively well when there is a clear margin of separation between classes such as in bivariate data and SVM is more effective in high dimensional spaces. We will use adaBoost for more analysis.

Checking for missing values

```
##              diagnosis            radius_mean           texture_mean
##                      0                      0                      0
##         perimeter_mean              area_mean        smoothness_mean
##                      0                      0                      0
##        compactness_mean         concavity_mean     concave.points_mean
##                      0                      0                      0
```

```
##           symmetry_mean  fractal_dimension_mean              radius_se
##                       0                       0                      0
##              texture_se            perimeter_se                 area_se
##                       0                       0                      0
##           smoothness_se          compactness_se           concavity_se
##                       0                       0                      0
##       concave.points_se             symmetry_se    fractal_dimension_se
##                       0                       0                      0
##            radius_worst           texture_worst         perimeter_worst
##                       0                       0                      0
##              area_worst        smoothness_worst       compactness_worst
##                       0                       0                      0
##         concavity_worst    concave.points_worst          symmetry_worst
##                       0                       0                      0
## fractal_dimension_worst
##                       0
```

We can clearly see that the dataset contains no missing values, the ones persisted were removed earlier in the cleaning section. Now let us visualize our data in depth. First, we start with our target class.

Visualization of the class variable - Data Visualization

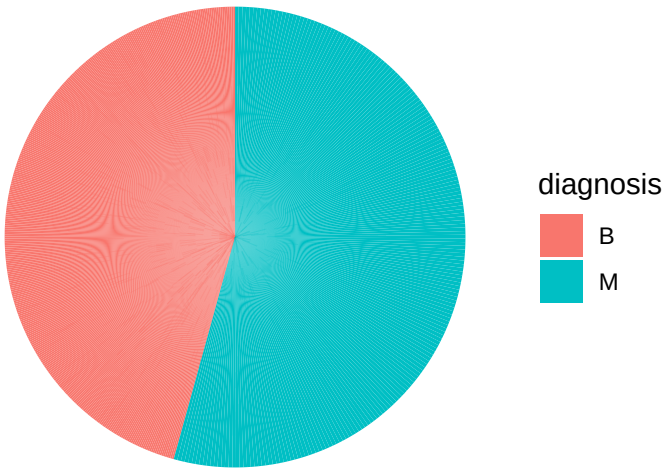


Figure 1: Distribution of Class variable

The dataset used for the analysis when visualized for the target class provided us with the above pie chart. This simple graphical representaiton of the target class provides us with some useful insight on the distribution of the target class ie. The value of the diagnosis as either Benign and Malignant. We see that the distribution is around 50-50 which means this dataset is perfect for this exercise.

Correlation Plot

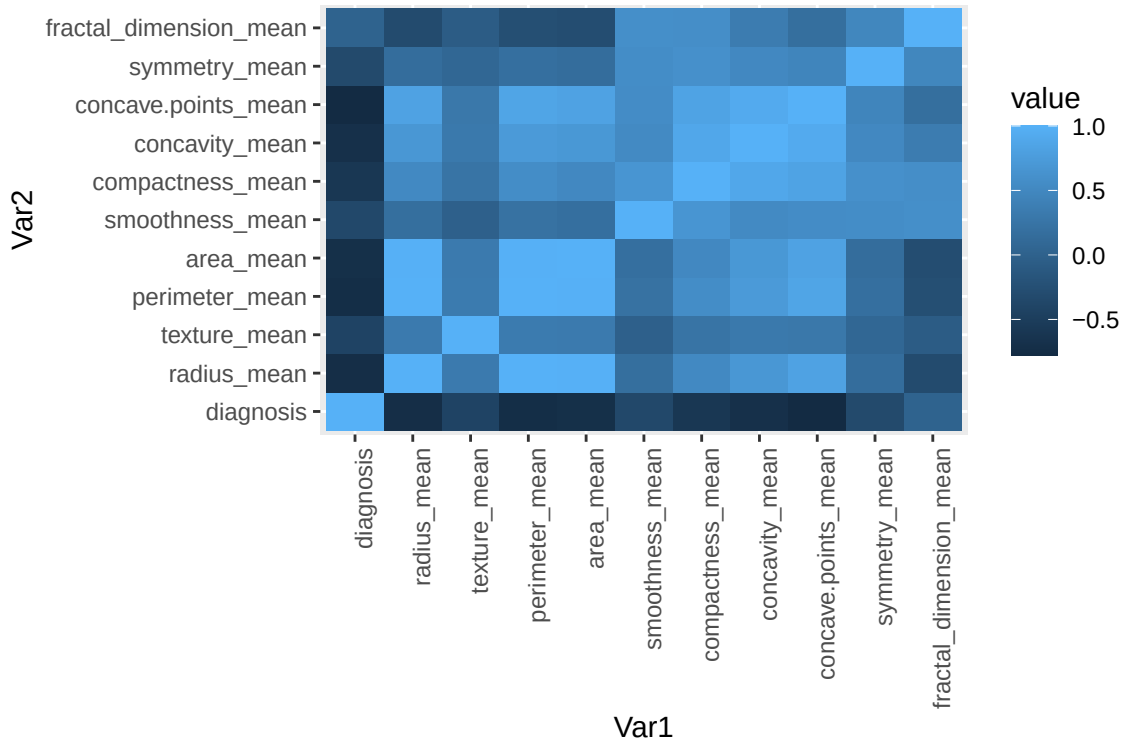


Figure 2: Mean attr corr plot

With the above correlation plot we can determine which features highly correlate to a diagnosis. A higher value (lighter color) corresponds to a high correlation. We can see that among mean values, concave points is the most highly correlated feature to diagnosis while fractal_dimension is the least.

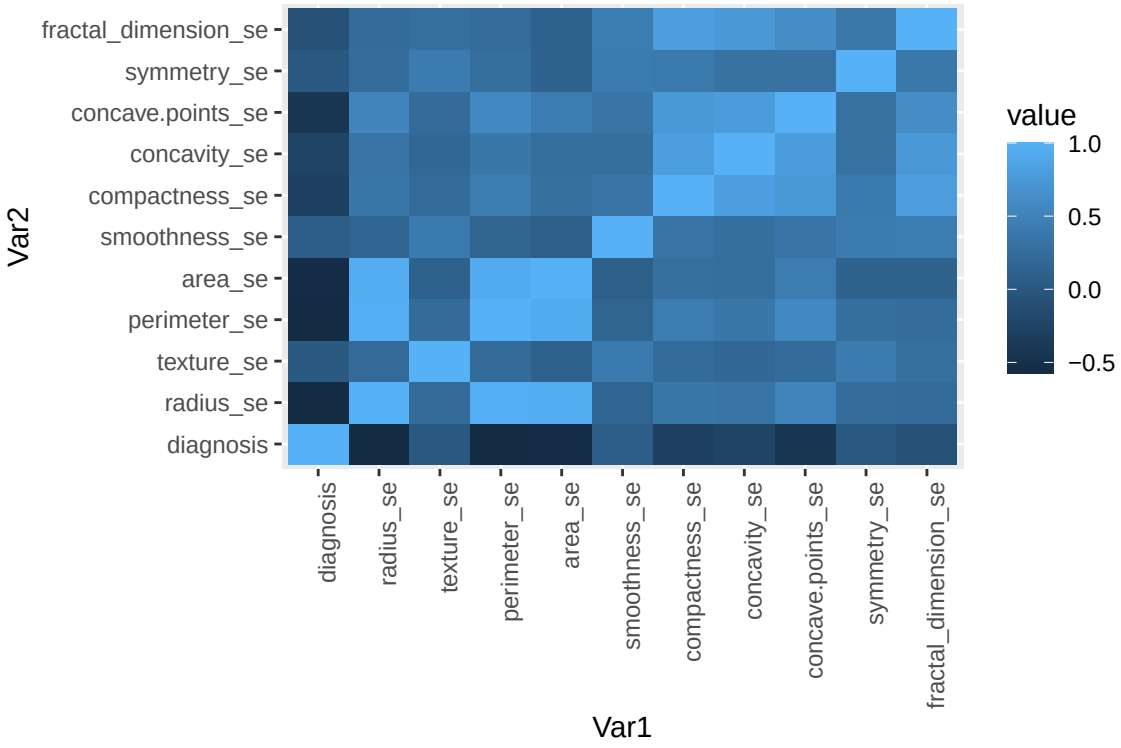


Figure 3: Se attr corr plot

For standard error values of all features, we can see that area is most highly correlated to diagnosis while texture and smoothness are the least.

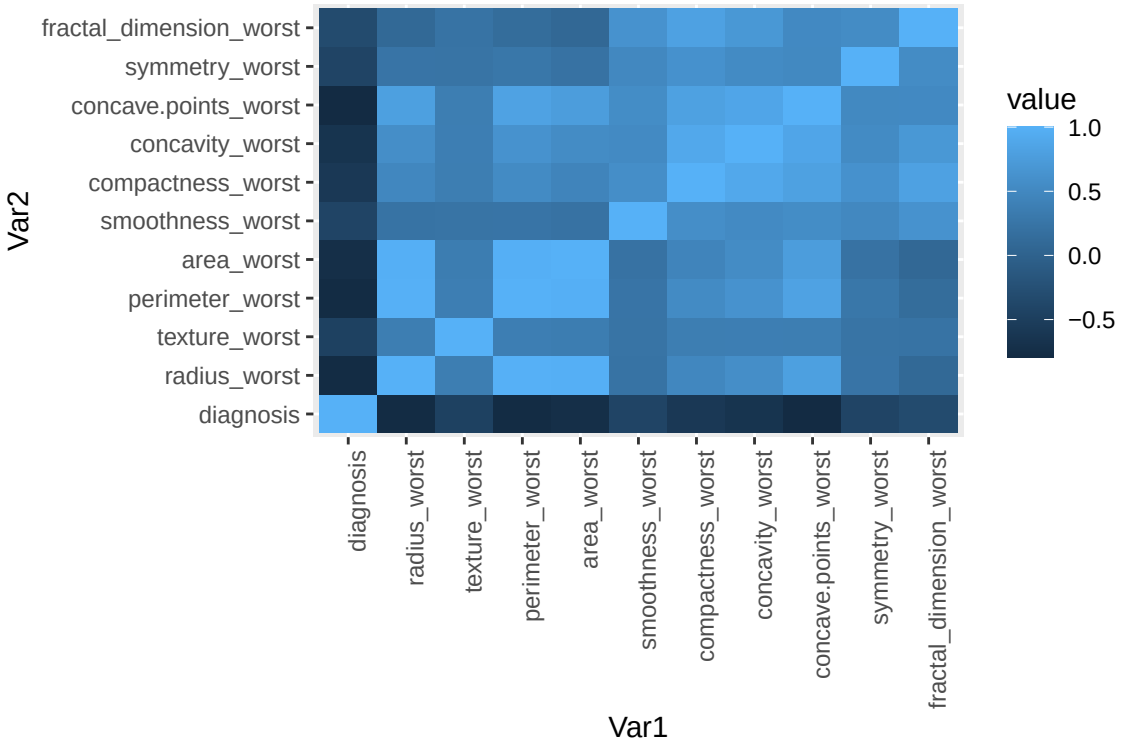


Figure 4: Worst attr corr plot

Among worst values of all the feature we can observe that concave points is the most highly correlated feature to diagnosis while fractal dimension seems to be the least.

Scatter Plot Visualization

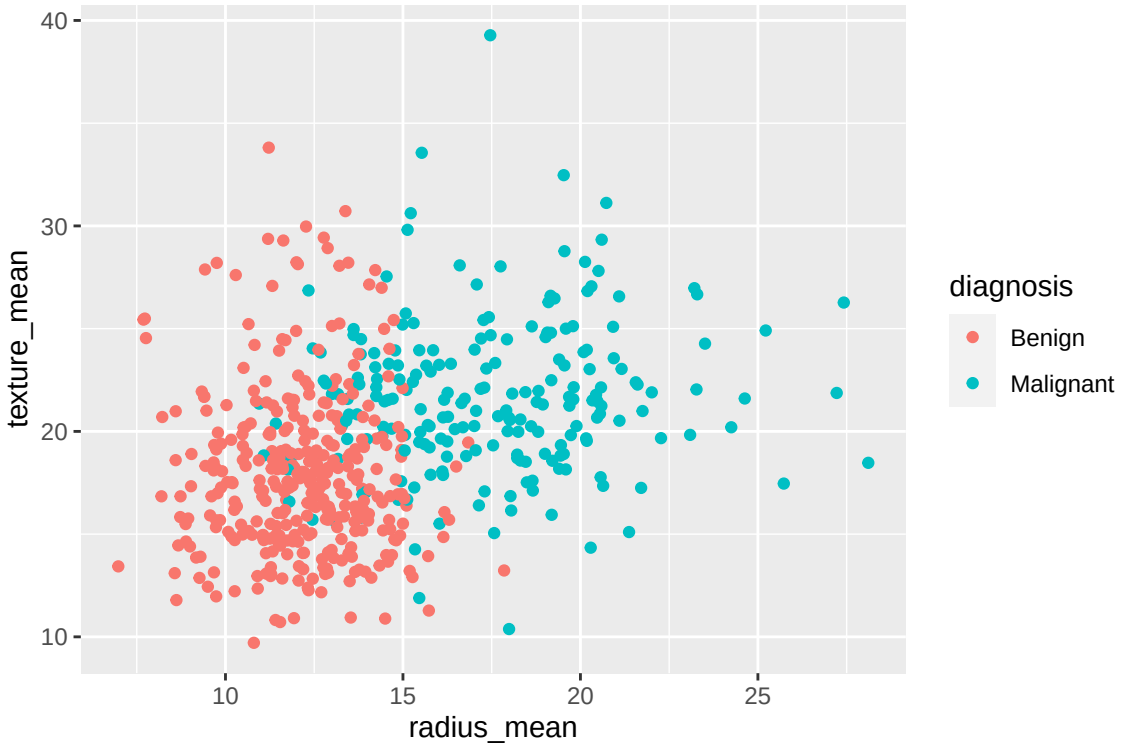


Figure 5: Scatter plot for the best variable by visual analysis

Using the above scatterplot it can be understood that if the value of radius_mean is above 18, i.e, by drawing a vertical line on the graph above at radius_mean = 18, we can say with 100% accuracy that the tumor is malignant. A value of radius_mean greater than 15 also gives a pretty good prediction of a malignant tumor. This indicates a high correlation of the radius_mean with the diagnosis.

Histogram Plot Visualization

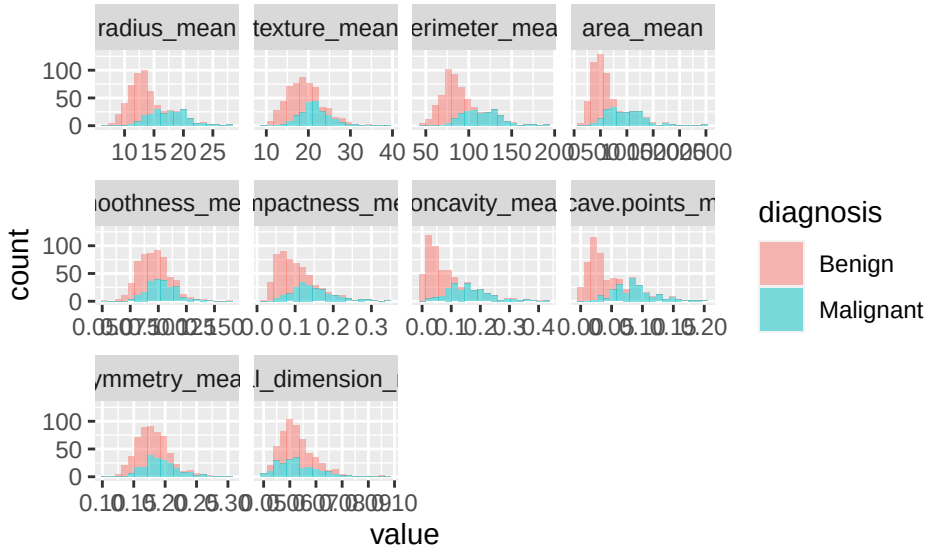


Figure 6: Class variable count by _mean valued attributes

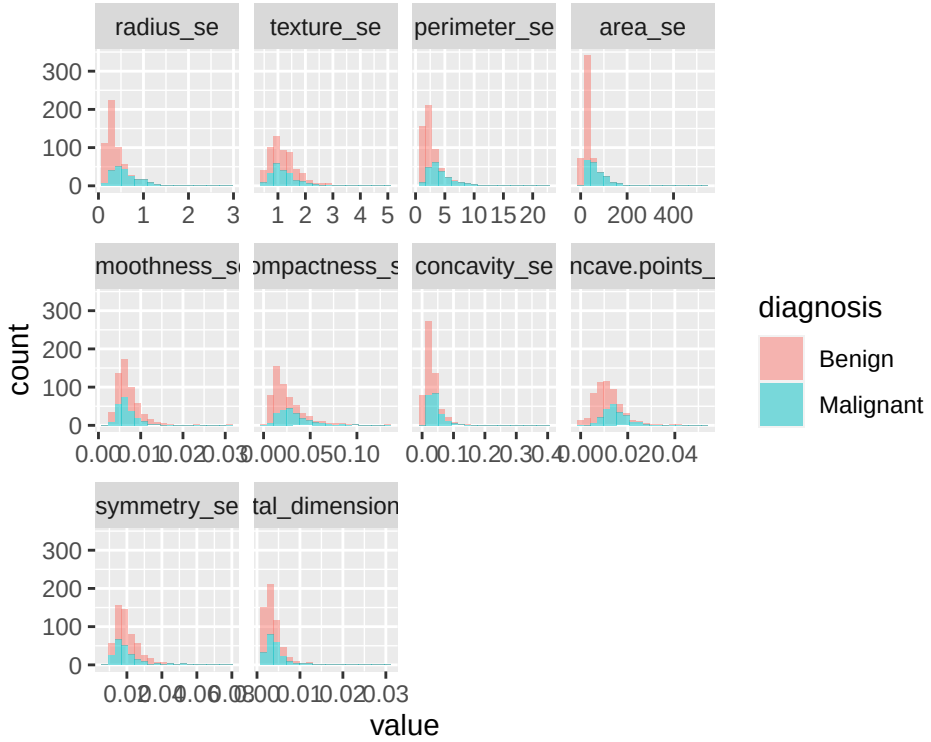


Figure 7: Class variable count by _se valued attributes

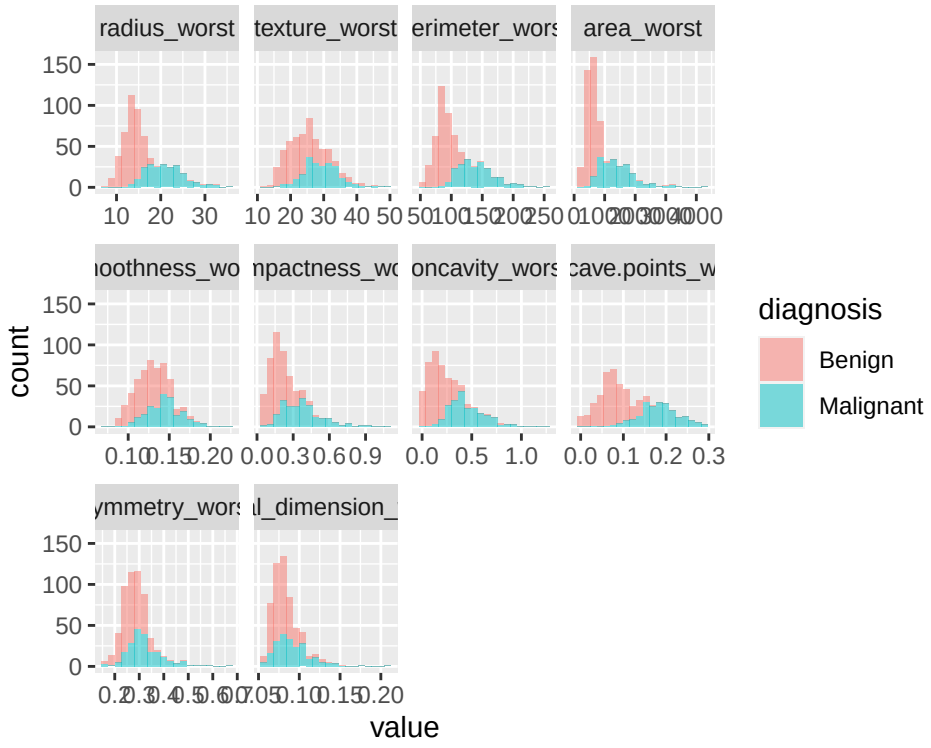
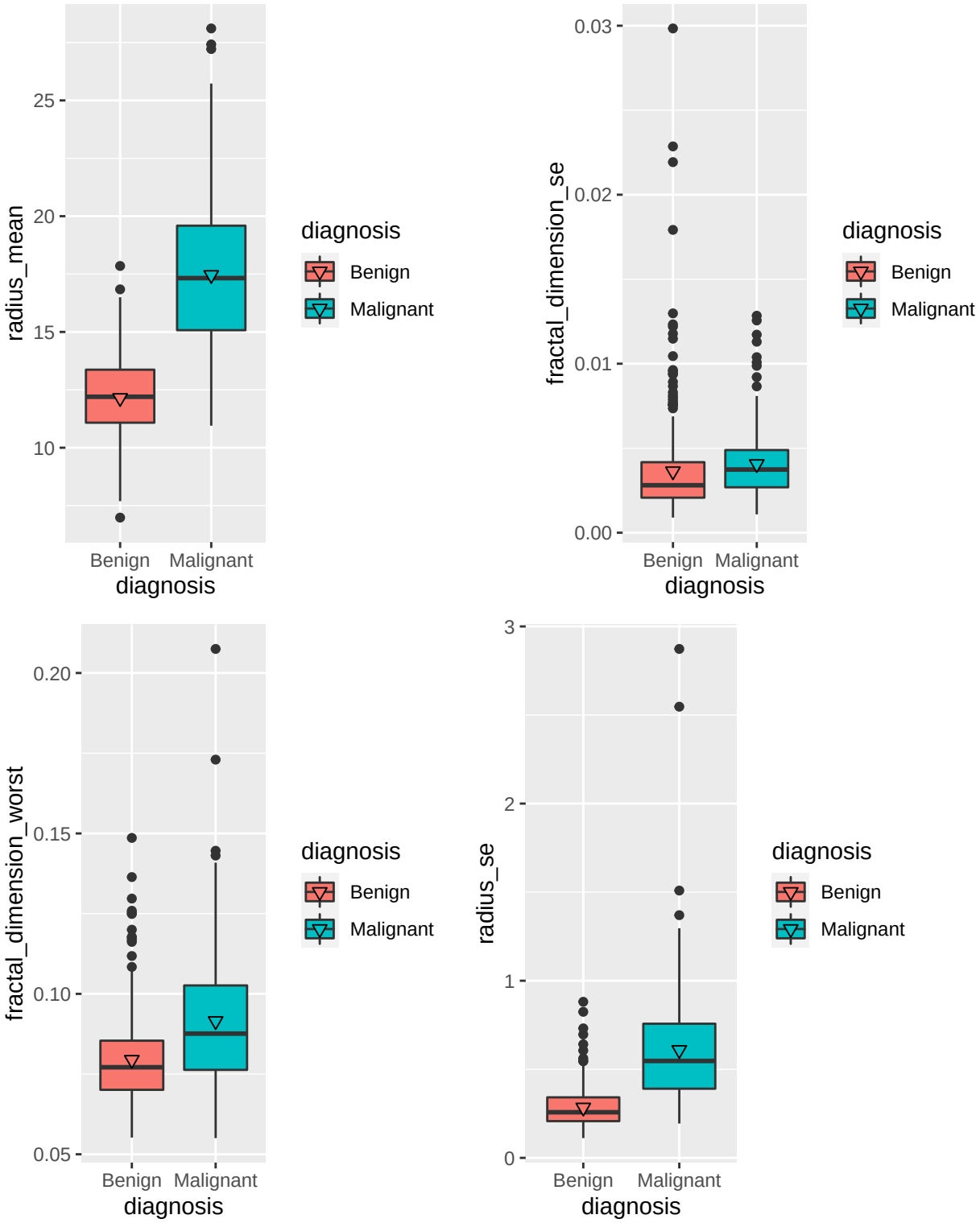


Figure 8: Class variable count by _worst valued attributes

Histograms for all mean, standard error and worst values of the 10 features are shown above. It is clear from the above plots that there are more benign cases than malignant ones. We can observe that a low mean value usually corresponds to a benign tumor. There are also certains values for which it would be hard to distinguish between benign and malignant cases as there are an equal number of both. We can also see that there is a certain mean value beyond which there are a very less number of cases. One of the most clear take aways from the above plots is that most benign cases have a lower value of area.

Box Plot Visualization



Here, Box plots plays an important role in description analysis. We have used those attributes which are closely related to the class attributes. We can see that there are a few outliers in each plot and that the spread of Benign is much more than Malignant. We can also clearly distinguish the means of different plots.

The SE attribute of fractal dimension has too many outliers which will take different elevations of values and hence has a smaller range.

Candidate Algorithm study

Before we begin any analysis on our data for the 3 candidate algorithms for our study, we will need to have a performance matrix set up. The basis of this matrix will be to identify how well our statistical learning algorithms fared on our data. First, let us split our data into two dataframes that we can use constantly throughout all the 3 tasks so that the data remains constant for our performance criteria. The splitting criteria is to have a 70-30 training-testing split. Let's dive deep into the 3 candidate algorithms on the cards.

Algorithm 1 - K-Means

The first candidate algorithm to study is K-Means. This algorithm is one of the very basic algorithms while learning statistical algorithms. It is one of the most easy and simple to learn clustering algorithms used to analyze the data sample. It is a vector quantization method directed to partition n observations into k partitions where k is the number of unique classes present. The prominent feature distinguishing K-means is that it reduces within-cluster variances.

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Benign Malignant
##   Benign       111        25
##   Malignant      0        35
##
##                Accuracy : 0.8538
##                  95% CI : (0.7918, 0.9031)
##     No Information Rate : 0.6491
##     P-Value [Acc > NIR] : 1.666e-09
##
##                   Kappa : 0.6451
##
##  Mcnemar's Test P-Value : 1.587e-06
##
##             Sensitivity : 1.0000
##             Specificity : 0.5833
##          Pos Pred Value : 0.8162
##          Neg Pred Value : 1.0000
##              Prevalence : 0.6491
##          Detection Rate : 0.6491
##    Detection Prevalence : 0.7953
##       Balanced Accuracy : 0.7917
##
##        'Positive' Class : Benign
##
```

Now let us study the various technical terms which have been given above in the output. The main features of these attributes are Accuracy and No Information Rate. The accuracy reported by the model is around 83%. The no information rate is the rate at which the model predicts the predominant class. This value should be lower than the accuracy reported by the model considerably to indicate that the model is better

than a literal coin toss / predict using the 1 rule. The next attributes present are sensitivity, which is high, the specificity being at 50% is also good as it resembles the no information rate. The balanced classifier is the best metric to suggest which model is the best based on all weighted attributes. The Balanced Accuracy for the model is around 77% currently and indicates that it is a good model but definitely not the best.

Algorithm 2 - Adaboost

One of the algorithms that we are using here is AdaBoost. AdaBoost is a boosting technique in which multiple small trees are recursively partitioned. It is a different name to CART. Here we use the control function to define the complexity parameter and maximum depth. Then we use library ada to test and predict the data. This algorithm is specifically being used to increase the accuracy by adding multiple CARTs which eventually helps the model to learn on its own with better accuracy as the data increases.

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Benign Malignant
##   Benign       110         2
##   Malignant      1        58
##
##                Accuracy : 0.9825
##                  95% CI : (0.9496, 0.9964)
##     No Information Rate : 0.6491
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9613
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9910
##             Specificity : 0.9667
##          Pos Pred Value : 0.9821
##          Neg Pred Value : 0.9831
##              Prevalence : 0.6491
##          Detection Rate : 0.6433
##    Detection Prevalence : 0.6550
##       Balanced Accuracy : 0.9788
##
##        'Positive' Class : Benign
##
```

As we can see the accuracy is far higher than the other two models because it is a combination of multiple models. We have used 100 boosting iterations. We are using gentle boosting. Also, we can mention the completeness during a tie in rpart because we can evidently say that this will lead to completeness and it is also useful to know not just which split was chosen, but which variable came in second, third, etc. The accuracy that we have here is more than 95% predicting the TP and TN almost perfectly. Type I and Type II errors are minimal. As there are two output class variables, we can form a Confidence interval. As the data is a bit biased we can say that the Benign class can be predicted without the model as well. This is no information rate. The P-value gives strong hypothesis that the data is not too biased. This is also the McNemar test. Hence, this is a good model to work with if extensive testing and different iterations are further applied in future work.

Algorithm 3 - Decision Tree

A decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Decision trees requires little data preparation and are able to handle both numerical and categorical data. However, decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting. They can also be unstable because small variations in the data might result in a completely different tree being generated. While there are numerous implementations of decision trees, one of the most well-known is the C5.0 algorithm. The C5.0 algorithm has become the industry standard for producing decision trees, because it does well for most types of problems directly out of the box.

```
##  Factor w/ 2 levels "Benign","Malignant": 1 1 1 1 1 1 2 2 2 1 ...
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Benign Malignant
##   Benign       108         4
##   Malignant      3        56
##
##                Accuracy : 0.9591
##                  95% CI : (0.9175, 0.9834)
##     No Information Rate : 0.6491
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9098
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9730
##             Specificity : 0.9333
##          Pos Pred Value : 0.9643
##          Neg Pred Value : 0.9492
##              Prevalence : 0.6491
##          Detection Rate : 0.6316
##    Detection Prevalence : 0.6550
##       Balanced Accuracy : 0.9532
##
##        'Positive' Class : Benign
##
```

From the results of the confusion matrix, the Reference is the real class and Prediction is the predicted class. Where we have 97 Reference Benign(B) and Prediction B, that is True Positive, because all of those were correctly identified as what they are. The other 6 values in Reference B column are False Negatives, because they are B's falsely predicted to be the other class. Similarly for the Malignant(M) class we have 62 True Positives and 6 False Negatives. The accuracy indicates overall how often is the algorithm correct thus the C50 algorithm gives a correct prediction for 92.98% of tested values. The Kappa value is basically a metric that measures how good the predictions are compared to random guessing / assignment. This reduces chance based predictions when most of the data belongs to one of the classes.The no-information rate indicates the largest proportion of the observed classes. Specificity measures true negative rate and specificity value is 91% means that 1 of every 10 data values are mis-labeled and 9 are correctly labeled. Sensitivity answers the question: "How many of the positive cases were detected?". Balance accuracy is the arithmetic mean of the two. Thus the decision tree gives a fairly accurate prediction of whether the tumor

is benign or malignant however there are algorithms that make better predictions with higher accuracy as we have seen with adaboost.

Algorithm 4 - Random Forest

Earlier in the paper we learnt about D-Tree's. The pro's and cons of D-Tree's are known to all, however, Random forest is an ensemble of multiple D-Tree's. This brings us to the point to assert that random forest helps us to mitigate the cons as when multiple of them run on the same dataset in a random fashion with a random start seed, the error is minimised and the best individual tree is saved from the collection for the modal. One of the unique features for Random forest is that the selection process happens by means of voting.

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Benign Malignant
##   Benign       110         2
##   Malignant      1        58
##
##                Accuracy : 0.9825
##                  95% CI : (0.9496, 0.9964)
##     No Information Rate : 0.6491
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9613
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9910
##             Specificity : 0.9667
##          Pos Pred Value : 0.9821
##          Neg Pred Value : 0.9831
##              Prevalence : 0.6491
##          Detection Rate : 0.6433
##    Detection Prevalence : 0.6550
##       Balanced Accuracy : 0.9788
##
##        'Positive' Class : Benign
##
```

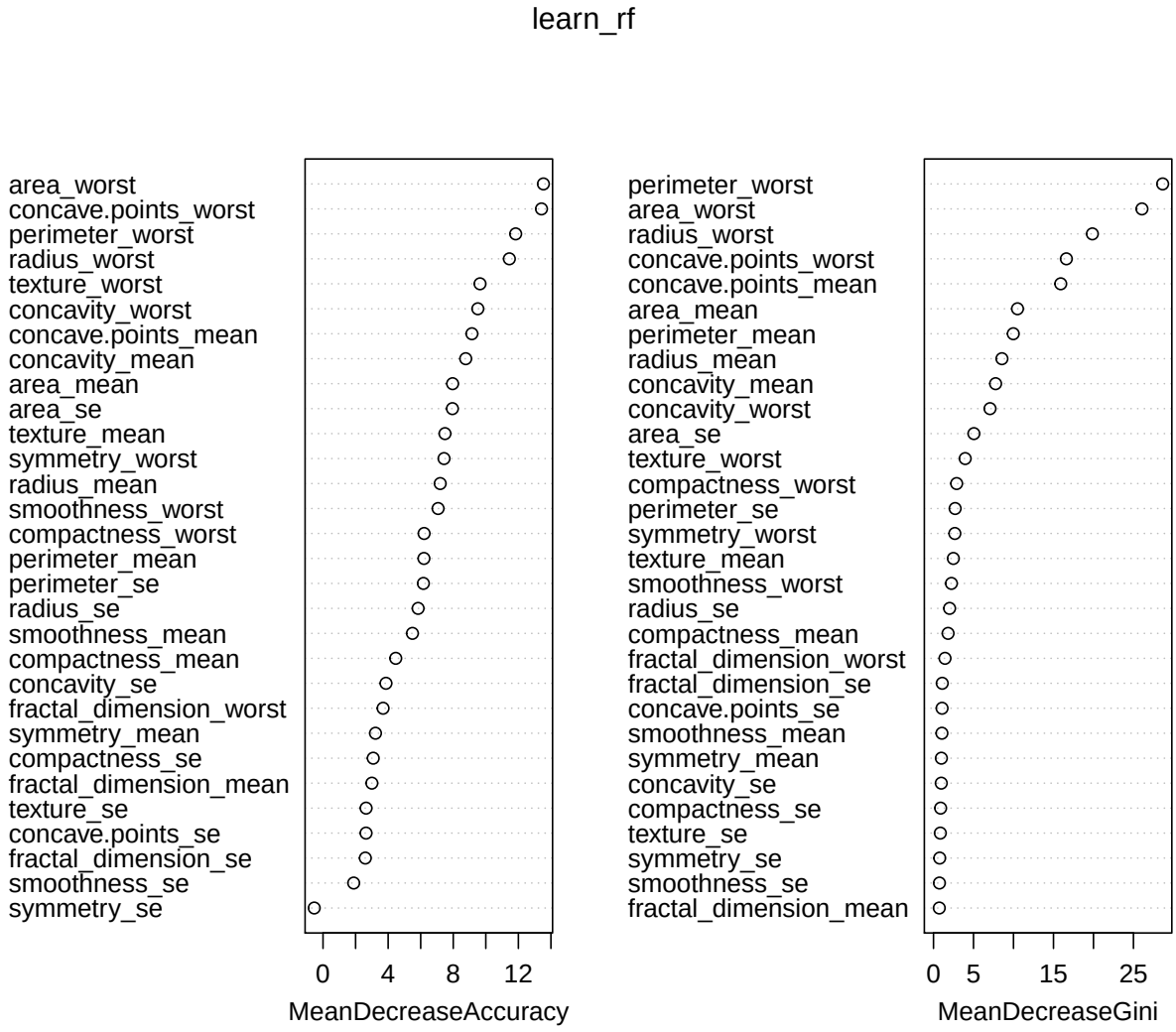


Figure 9: Variance Importance Plot

In the variance important plot the MeanDecreaseAccuracy can be used to determine important features for improving accuracy. As we can see, radius_mean is the best feature to improve accuracy followed by concave.points_worst, area_worst and then perimeter_worst. The MeanDecreaseGini is used to find the features for improving node impurities and we can see that perimeter_worst fared the best, followed by radius_worst, area_worst and then concave.points_worst.

Algorithm 5:- Core Algorithm: SVM

SVM transforms the raw variables by applying statistical learning theory, and make it to multiple dimensions and complex spaces. which helps in finding a linear decision surface. Therefore, this transformation is computed using a kernel function. The kernel function is one which maps it into linear space. It transforms the data into a new feature space and classifies data in that new space. The Gram matrix is used as a kernel function. It is said that we have a valid kernel matrix when values are positive semi-definite. Therefore, to find separating hyperplanes optimization problem has to be solved. The weight vector is rotated with respect

to the data points. Therefore, it is attractive approach because of its linear model, and also the dataset can be visualized by rotating the dataset. The rotation is done to the point classes are clearly separable.

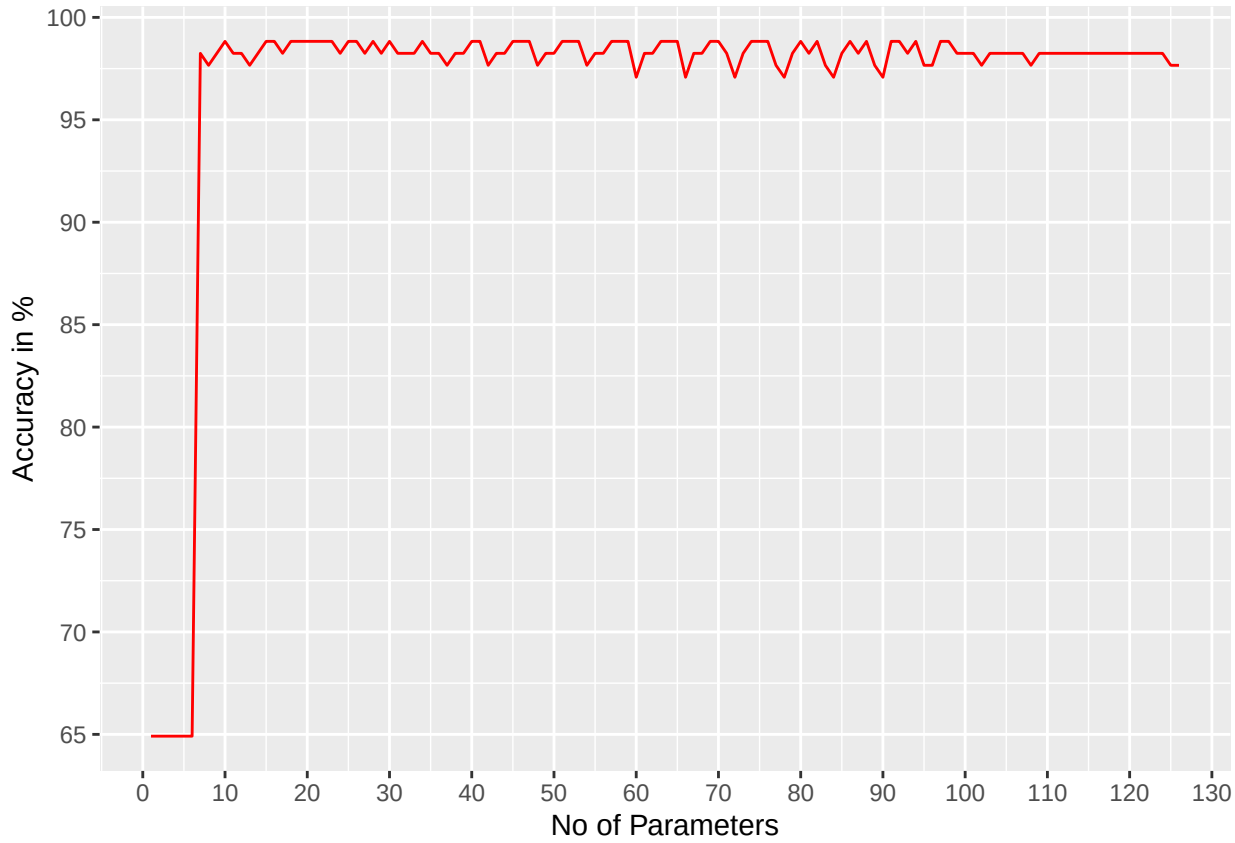
Drawbacks of SVM

Requires a lot of storage space and is slow. Having more than two classes, it does pairwise comparison. Need to try several kernels to know which is the best one.

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Benign Malignant
##   Benign       109         1
##   Malignant      2        59
##
##                Accuracy : 0.9825
##                  95% CI : (0.9496, 0.9964)
##     No Information Rate : 0.6491
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9616
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9820
##             Specificity : 0.9833
##          Pos Pred Value : 0.9909
##          Neg Pred Value : 0.9672
##              Prevalence : 0.6491
##          Detection Rate : 0.6374
##    Detection Prevalence : 0.6433
##       Balanced Accuracy : 0.9827
##
##        'Positive' Class : Benign
##
```

Now let us imporve the svm algorithm in a way that it boosts the performane for our dataset.

```
## [1] "Optimal number of parameter is 10 (accuracy : 0.988304093567251 ) in SVM"
```



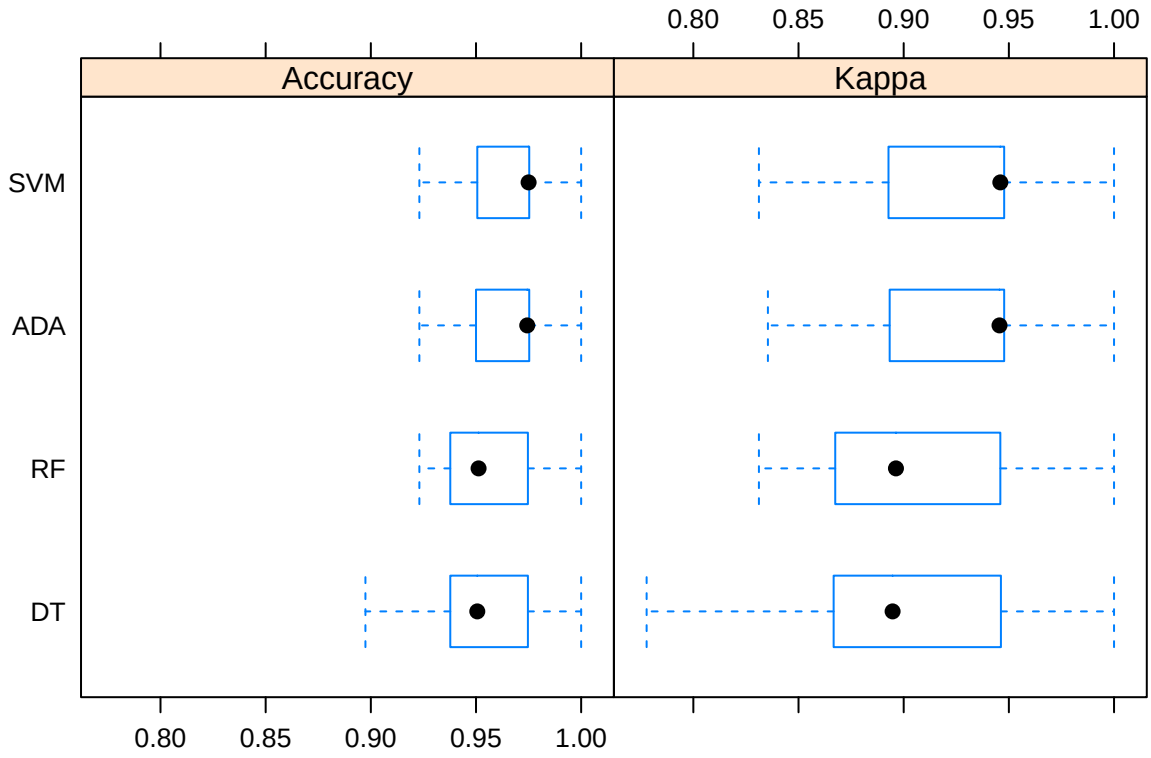
The SVM tune is used to get optimal values for cost and gamma. Therefore, this function tunes hyperparameters using a grid search on parameter ranges. Either can be used to tune the function or a character string naming. Here ranges are nothing but the ist of parameters vector spanning the sampling space. We plot the accuracy to see how many parameters are optimal.

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Benign Malignant
##   Benign       110         1
##   Malignant      1        59
##
##                Accuracy : 0.9883
##                  95% CI : (0.9584, 0.9986)
##     No Information Rate : 0.6491
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9743
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9910
##             Specificity : 0.9833
##          Pos Pred Value : 0.9910
##          Neg Pred Value : 0.9833
##              Prevalence : 0.6491
```

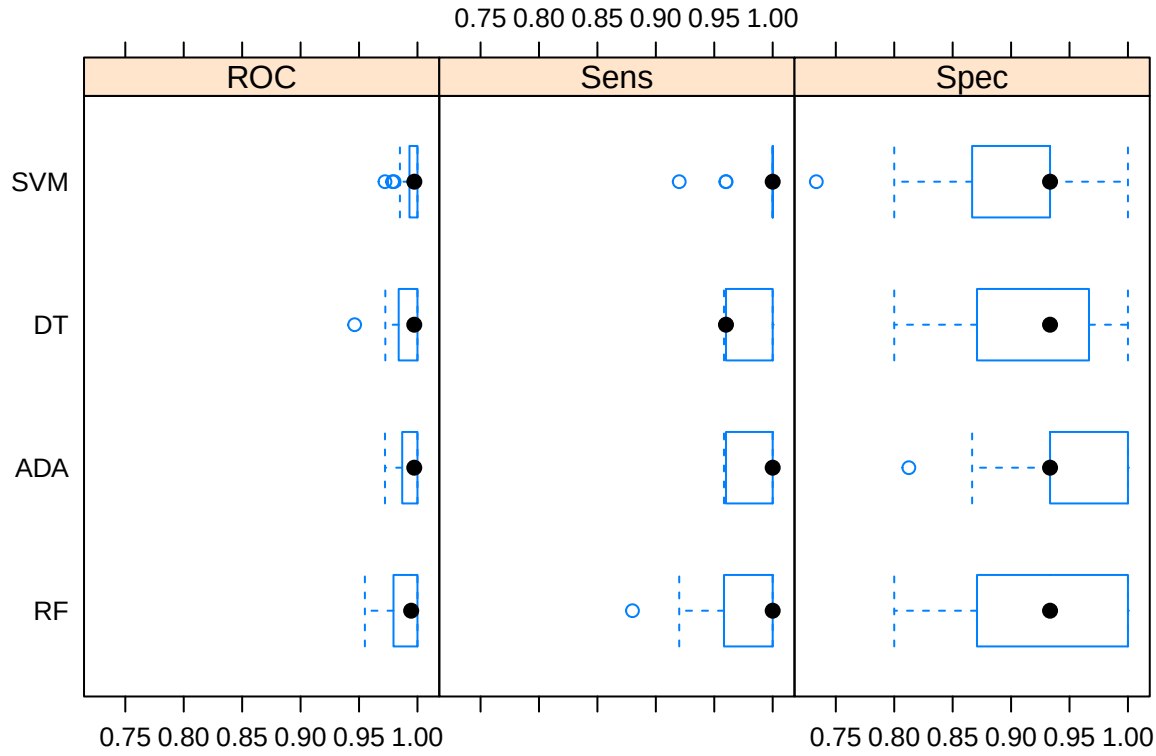
```
##          Detection Rate : 0.6433
##    Detection Prevalence : 0.6491
##       Balanced Accuracy : 0.9872
##
##        'Positive' Class : Benign
##
```

As we can see the effect is extensive and we successfully increased the accuracy for SVM. This is the best tuned model that we can present.

We use a train model here which has 10 different cross validation matrix and repeat this loop with 2. We have used all the models here and tried to compare using two metrics, Accuracy and ROC points.



This is a seperate section where we modify the comparisions according to ROC points.



Let's talk about this Model comparison overview. We have used Accuracy and ROC metrics here. Accuracy metrics involve Accuracy score and Kappa value whereas the ROC metric includes the ROC values, Sensitivity, and Specificity. We have compared our models according and displayed them in a table format below for clear visuals. We have also plotted some Box and whisker plots to understand the spread of each metric. Clearly, our SVM model did the best job and the rest of them followed the SVM value for Accuracy. Speaking about Spread, decision Tree was the distant model which proves how crucial is the parameters tuning in the Decision tree. Let's compare different metrics because we can obviously not state a firm result just on basis of Accuracy. Speaking about ROC metrics, We can see that the results on Accuracy ALMOST justifies what we want to prove. It becomes super clear when we combine and try to plot this on a ROC curve plot. We can do that as future work.

Conclusion

Thus in our project we have analysed the Breast Cancer Wisconsin (Diagnostic) Data Set, used various visualizations to extrapolate information and implemented five algorithms to compare which fares better in classifying the tumor data as either Benign or Malignant. We chose SVM as our core algorithm and applied fine tuning on it. We finally conclude that the fined tuned SVM algorithm gave us the best accuracy. As all three involved in this project are used to writing code in Python, this project has been a great way to get to learn R and understand its usefulness.

Algorithms Comparison

##Apendix–Code

Table 1: Accuracy values of Algorithms

Algorithms	Min	Mean	Median
Random Forest	92.3%	95.6%	95.1%
Adaboost	92.3%	96.61%	97.43%
Support Vector Machine	92.30%	96.74%	97.50%
Decision Tree	89.74%	90.36%	89.47%

Table 2: Kappa values of Algorithms

Algorithms	Min	Mean	Median
Random Forest	83.11%	90.61%	89.63%
Adaboost	83.54%	92.77%	94.55%
Support Vector Machine	83.11%	93.02%	94.59%
Decision Trees	77.78%	90.36%	89.47%

Table 3: ROC values of Algorithms

Algorithms	Min	Mean	Median
Random Forest	95.5%	98.9%	99.46%
Adaboost	97.22%	99.31%	99.72%
Support Vector Machine	97.22%	99.37%	99.73%
Decision Trees	94.62%	99.06%	99.72%

Table 4: Sensitivity values of Algorithms

Algorithms	Min	Mean	Median
Random Forest	88%	97.37%	100%
Adaboost	95.83%	98.17%	100%
Support Vector Machine	92%	99.2%	100%
Decision Trees	95.83%	97.56%	96%

Table 5: Specificity values of Algorithms

Algorithms	Min	Mean	Median
Random Forest	80%	92.52%	93.33%
Adaboost	81.25%	94.14%	93.33%
Support Vector Machine	73.33%	90.2%	93.33%
Decision Trees	80%	91.52%	93.33%

```
knitr::opts_chunk$set(echo=FALSE, warning=FALSE, message=FALSE, fig.pos = 'H')
library(ggplot2)
library(reshape2)

data <- read.csv("Data.csv", header = TRUE)

# Cleaning the Null column
data$X <- NULL
data$id <- NULL
summary(data)
# Missing value identification -- found none
sapply(data, function(x) sum(is.na(x)))
# Pie chart for the number of B / M count of the diagnosis
library(ggplot2)
ggplot(data, aes(x="", y=diagnosis, fill=diagnosis)) +
  geom_bar(stat="identity", width=1) +
  coord_polar("y", start=0)  +  theme_void()

#Plotting the correlation matrix

# One Hot encoding
data$diagnosis <- factor(ifelse(data$diagnosis =="B", 1, 0))

# Converting it to a numeric value
data$diagnosis <- as.numeric(data$diagnosis)

# Extracting _Mean value attributes
corr_cancer_mean <- data[,c(0:11)]

# Extracting _Se value attributes
corr_cancer_se <- data[,c(-2:-11)]
corr_cancer_se <- corr_cancer_se[,c(0:11)]

# Extracting _Worst value attributes
corr_cancer_worst <- data[,c(-2:-21)]


cor_cancer_mean <- round(cor(corr_cancer_mean),2)
melted_cancer_mean <- melt(cor_cancer_mean)
```

```
c1 <- ggplot(data = melted_cancer_mean, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() + theme(axis.text.x=element_text(angle=90,hjust=1))

cor_cancer_se <- round(cor(corr_cancer_se),2)
melted_cancer_se <- melt(cor_cancer_se)
c2 <- ggplot(data = melted_cancer_se, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() + theme(axis.text.x=element_text(angle=90,hjust=1))

cor_cancer_worst <- round(cor(corr_cancer_worst),2)
melted_cancer_worst <- melt(cor_cancer_worst)
c3 <- ggplot(data = melted_cancer_worst, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() + theme(axis.text.x=element_text(angle=90,hjust=1))

c1
c2
c3
# Ggplot scatter for the radius-mean feature
library("ggplot2")

# One Hot encoding
data$diagnosis <- factor(ifelse(data$diagnosis == 2, "Benign", "Malignant"))

# Converting it to a numeric value
data$diagnosis <- as.factor(data$diagnosis)

ggplot(data, aes(x = radius_mean, y = texture_mean)) +
  geom_point(aes(color = diagnosis))

# Extracting _Mean value attributes
corr_cancer_mean <- data[,c(0:11)]

# Extracting _Se value attributes
corr_cancer_se <- data[,c(-2:-11)]
corr_cancer_se <- corr_cancer_se[,c(0:11)]

# Extracting _Worst value attributes
corr_cancer_worst <- data[,c(-2:-21)]


# Plot for mean
ggplot(data = melt(corr_cancer_mean, id.var = "diagnosis"), mapping = aes(x = value)) +
    geom_histogram(bins = 20, aes(fill=diagnosis), alpha=0.5) + facet_wrap(~variable, scales ='free_x')

# Plot for SE
ggplot(data = melt(corr_cancer_se, id.var = "diagnosis"), mapping = aes(x = value)) +
    geom_histogram(bins = 20, aes(fill=diagnosis), alpha=0.5) + facet_wrap(~variable, scales = 'free_x')
# Plot for worst
ggplot(data = melt(corr_cancer_worst, id.var = "diagnosis"), mapping = aes(x = value)) +
    geom_histogram(bins = 20, aes(fill=diagnosis), alpha=0.5) + facet_wrap(~variable, scales = 'free_x')

data$diagnosis <- as.factor(data$diagnosis)
b1 <- ggplot(data, aes(x=diagnosis, y=radius_mean,fill=diagnosis)) +
  geom_boxplot()
```

```
b1 <- b1 + stat_summary(fun=mean, geom="point", shape=25, size=2)

b2 <- ggplot(data, aes(x=diagnosis, y=fractal_dimension_se,fill=diagnosis)) +
  geom_boxplot()
b2 <- b2 + stat_summary(fun=mean, geom="point", shape=25, size=2)

b3 <- ggplot(data, aes(x=diagnosis, y=fractal_dimension_worst,fill=diagnosis)) +
  geom_boxplot()
b3 <- b3 + stat_summary(fun=mean, geom="point", shape=25, size=2)

b4 <- ggplot(data, aes(x=diagnosis, y=radius_se,fill=diagnosis)) +
  geom_boxplot()
b4 <- b4 + stat_summary(fun=mean, geom="point", shape=25, size=2)
par(mfrow=c(2,2))
plot(b1)
plot(b2)
plot(b3)
plot(b4)

nrows <- NROW(data)
set.seed(218)                           ## fix random value
index <- sample(1:nrows, 0.7 * nrows)   ## shuffle and divide

train <- data[index,]                   ## 398 train data (70%)
test <- data[-index,]                   ## 171 test data (30%)

library(caret)

predictMy.kmeans <- function(newdata, object){
    centers <- object$centers
    n_centers <- nrow(centers)
    dist_mat <- as.matrix(dist(rbind(centers, newdata)))
    dist_mat <- dist_mat[-seq(n_centers), seq(n_centers)]
    max.col(-dist_mat)
}

learn_kmeans <- kmeans(train[,-1], centers=2)

pre_kmeans <- predictMy.kmeans(test[,-1],learn_kmeans)
pre_kmeans <- ifelse(pre_kmeans == 1,"Benign","Malignant")

pre_kmeans <- as.factor(pre_kmeans)
cm_kmeans <- confusionMatrix(pre_kmeans, test$diagnosis)
cm_kmeans

library(rpart)
library(ada)

control <- rpart.control(maxdepth = 10, xval = 0)
learn_ada <- ada(diagnosis~., data = train, test.x = train[,-1], test.y = train[,1], type = "gentle", c
predict_ada <- predict(learn_ada, test[,-1])
cm_ada <- confusionMatrix(predict_ada, test$diagnosis)
cm_ada
```

```
library(C50)

train$diagnosis<-as.factor(train$diagnosis)
str(train$diagnosis)
learn_c50 <- C5.0(train[,-1],train$diagnosis)
pre_c50 <- predict(learn_c50, test[,-1])
cm_c50 <- confusionMatrix(pre_c50, as.factor(test$diagnosis))
cm_c50

library(randomForest)
learn_rf <- randomForest(diagnosis~., data=train, ntree=300, proximity=T, importance=T)
pre_rf   <- predict(learn_rf, test[,-1])
cm_rf    <- confusionMatrix(pre_rf, test$diagnosis)
cm_rf
varImpPlot(learn_rf)
library(e1071)
learn_svm <- svm(diagnosis~., data=train)
pre_svm <- predict(learn_svm, test[,-1])
cm_svm <- confusionMatrix(pre_svm, test$diagnosis)
cm_svm

library(e1071)
gamma <- seq(0,0.1,0.005)
cost <- 2^(0:5)
parms <- expand.grid(cost=cost, gamma=gamma)    ## 231

acc_test <- numeric()
accuracy1 <- NULL; accuracy2 <- NULL

for(i in 1:NROW(parms)){
        learn_svm <- svm(diagnosis~., data=train, gamma=parms$gamma[i], cost=parms$cost[i])
        pre_svm <- predict(learn_svm, test[,-1])
        accuracy1 <- confusionMatrix(pre_svm, test$diagnosis)
        accuracy2[i] <- accuracy1$overall[1]
}

acc <- data.frame(p= seq(1,NROW(parms)), cnt = accuracy2)

opt_p <- subset(acc, cnt==max(cnt))[1,]
sub <- paste("Optimal number of parameter is", opt_p$p, "(accuracy :", opt_p$cnt,") in SVM")

sub

ggplot() + geom_line(aes(x=acc$p,y=acc$cnt*100),color='red') +
           scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
           scale_y_continuous(breaks = scales::pretty_breaks(n = 10)) +
           ylab('Accuracy in %')+xlab('No of Parameters')
learn_imp_svm <- svm(diagnosis~., data=train, cost=parms$cost[opt_p$p], gamma=parms$gamma[opt_p$p])
pre_imp_svm <- predict(learn_imp_svm, test[,-1])
cm_imp_svm <- confusionMatrix(pre_imp_svm, test$diagnosis)
cm_imp_svm
library(mlbench)
library(caret)
```

```
control <- trainControl(method="repeatedcv", number=10, repeats=2)
set.seed(7)
modelrf <- train(diagnosis~., data=train, method="rf", ntree=100, trControl=control)
set.seed(7)
modelada <- train(diagnosis~., data=train, method="ada", trControl=control, verbose=FALSE)
set.seed(7)
modelsvm <- train(diagnosis~., data=train, method="svmRadialWeights", trControl=control,verbose=FALSE)
set.seed(7)
modelTree <- train(diagnosis~., data=train, method="C5.0", trControl=control, verbose=FALSE)
results <- resamples(list(RF=modelrf, ADA=modelada, SVM=modelsvm,DT=modelTree))
bwplot(results)
control <- trainControl(method="repeatedcv", number=10, repeats=2,classProbs = TRUE,summaryFunction = tw
set.seed(7)
modelrf <- train(diagnosis~., data=train, method="rf", ntree=100, trControl=control,metric="ROC")
set.seed(7)
modelada <- train(diagnosis~., data=train, method="ada", trControl=control, verbose=FALSE,metric="ROC")
set.seed(7)
modelsvm <- train(diagnosis~., data=train, method="svmRadialWeights", trControl=control,verbose=FALSE
                  ,metric="ROC")
set.seed(7)
modelTree <- train(diagnosis~., data=train, method="C5.0", trControl=control, verbose=FALSE,metric="ROC"
results <- resamples(list(RF=modelrf, ADA=modelada, SVM=modelsvm,DT=modelTree))
bwplot(results, layout = c(3, 1))
```