CSML1000 Winter 2020, Group 8, Assignment1: Breast Cancer Diagnosis Prediction Model

Steven Wang, Tarun Bagga, Paul Doucet, Jerry Khidaroo,

2/15/2020

Load Libraries

```
# Load packages
library('ggplot2') # visualization
library('ggthemes') # visualization
library('scales') # visualization
library('dplyr') # data manipulation
library('mice') # imputation
library('randomForest') # classification algorithm
library('caret')
library('e1071')
library('shiny')
library('shinythemes')
library('readr')
# load the package "nnet" which provide the ANN modeling functionality
library("nnet")
# load the libary "purrr" for the ANN function map
library("purrr")
library('corrplot')
library('data.table')
library('caTools')
# library(klaR)
```

1. Business Understanding

- Business Problem: When a patient has a Radiologic Study done to image a tumor there can be a lengthy wait time between the capture of the study and recording of the measurement by the technologist to the time when the tumor is identified as ‘Benign’ or ‘Maligant’ either through a Radiologist Observation Result or a Biopsy Result. A machine Learning Model that could predict the diagnosis as soon as the measurements are entered would give Clinicians an early warning tool that could speed up the time for the start of treatment of a patient.
- Project Plan:
    - Load and get an understanding of the dataset, its target variable and its features.
    - Make any modifications to the dataset needed to enable learning algorithms to be run on the data.
    - Identify the features of the dataset that are important in predicting the target variable (dianosis in this case).

    - Build and evaluate 3 to 4 Models from the dataset by appling various machine learning algoritms as appropiate and testing them.
    - Identify the best model to use for the project.
    - Build a shiny app that deploys the selected model with a user interface for end users to imput measurement values from a study and obtain a pridiction result.
    - Identify any ethical considerations that should be addressed at each stage of the process.
- Business Success Criteria: A successful project outcome would be achieved if a model is created that can predict the ‘B’ or ‘M’ outcome with a high degree of accuracy and sensitivity when given an unlabeled set of measurements from a tumor study.

- Ethical Framework Questions:

    - How could your system negatively impact individuals? The greatest negative impact would occur for a false positive diagnosis since this could delay treatment and in a life threatening scenario. A false negative would also be negatively impactful but to a lessor degree.
    - Who is most vulnerable and why? The most vulnerable would be patients with a ‘M’ dianosis not detectable by the model.
    - How much error in predictions can your business accept for this use case? False positives need to be minimised as much as posible. Minimsing False negatives are second in priority.
    - Will you need to explain which input factors had the greatest influence on outputs? Yes. Being able to explain which features have the most influence on outcome is very desirable.
    - Do you need PII or can you provide group-level data? The analysis requires patient level data however any PII can be anonymised

2. Data Understanding

- Ethical Framework Questions:
    - Have you de-identified your data and taken measures to reduce the probability of reidentification? The data is de-identified.
    - Will socially sensitive features like gender or ethnic background influence outputs? No demographic data is present.
    - Are seemingly harmless features like location hiding proxies for socially sensitive features? No demographic data is present.

Get Data File

- The Dataset used is obtained from: https://www.kaggle.com/uciml/breast-cancer-wisconsin-data (https://www.kaggle.com/uciml/breast-cancer-wisconsin-data)

```
# Concatenate data file subdir and name.
full_file_name <- paste("./input/diagnosis_data.csv",sep="")
show(full_file_name)
```

```
## [1] "./input/diagnosis_data.csv"
```

Load and check data

```
# Load full data
diagnosis_data_full = read.csv(full_file_name)
breastcancer_data <- read.csv(full_file_name)

# check data
str(diagnosis_data_full)
```

```
## 'data.frame':    569 obs. of  33 variables:
##  $ id                      : int  842302 842517 84300903 84348301 84358402 843786 844359 84458
202 844981 84501001 ...
##  $ diagnosis               : Factor w/ 2 levels "B","M": 2 2 2 2 2 2 2 2 2 2 ...
##  $ radius_mean             : num  18 20.6 19.7 11.4 20.3 ...
##  $ texture_mean            : num  10.4 17.8 21.2 20.4 14.3 ...
##  $ perimeter_mean          : num  122.8 132.9 130 77.6 135.1 ...
##  $ area_mean               : num  1001 1326 1203 386 1297 ...
##  $ smoothness_mean         : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
##  $ compactness_mean        : num  0.2776 0.0786 0.1599 0.2839 0.1328 ...
##  $ concavity_mean          : num  0.3001 0.0869 0.1974 0.2414 0.198 ...
##  $ concave.points_mean     : num  0.1471 0.0702 0.1279 0.1052 0.1043 ...
##  $ symmetry_mean           : num  0.242 0.181 0.207 0.26 0.181 ...
##  $ fractal_dimension_mean : num  0.0787 0.0567 0.06 0.0974 0.0588 ...
##  $ radius_se               : num  1.095 0.543 0.746 0.496 0.757 ...
##  $ texture_se              : num  0.905 0.734 0.787 1.156 0.781 ...
##  $ perimeter_se            : num  8.59 3.4 4.58 3.44 5.44 ...
##  $ area_se                 : num  153.4 74.1 94 27.2 94.4 ...
##  $ smoothness_se           : num  0.0064 0.00522 0.00615 0.00911 0.01149 ...
##  $ compactness_se          : num  0.049 0.0131 0.0401 0.0746 0.0246 ...
##  $ concavity_se            : num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
##  $ concave.points_se       : num  0.0159 0.0134 0.0206 0.0187 0.0188 ...
##  $ symmetry_se             : num  0.03 0.0139 0.0225 0.0596 0.0176 ...
##  $ fractal_dimension_se    : num  0.00619 0.00353 0.00457 0.00921 0.00511 ...
##  $ radius_worst            : num  25.4 25 23.6 14.9 22.5 ...
##  $ texture_worst           : num  17.3 23.4 25.5 26.5 16.7 ...
##  $ perimeter_worst         : num  184.6 158.8 152.5 98.9 152.2 ...
##  $ area_worst              : num  2019 1956 1709 568 1575 ...
##  $ smoothness_worst        : num  0.162 0.124 0.144 0.21 0.137 ...
##  $ compactness_worst       : num  0.666 0.187 0.424 0.866 0.205 ...
##  $ concavity_worst         : num  0.712 0.242 0.45 0.687 0.4 ...
##  $ concave.points_worst    : num  0.265 0.186 0.243 0.258 0.163 ...
##  $ symmetry_worst          : num  0.46 0.275 0.361 0.664 0.236 ...
##  $ fractal_dimension_worst: num  0.1189 0.089 0.0876 0.173 0.0768 ...
##  $ X                       : logi  NA NA NA NA NA NA ...
```

```
summary(diagnosis_data_full)
```

```
##        id            diagnosis  radius_mean      texture_mean
##  Min.   :     8670   B:357     Min.   : 6.981   Min.   : 9.71
##  1st Qu.:   869218   M:212     1st Qu.:11.700   1st Qu.:16.17
##  Median :   906024             Median :13.370   Median :18.84
##  Mean   : 30371831             Mean   :14.127   Mean   :19.29
##  3rd Qu.:  8813129             3rd Qu.:15.780   3rd Qu.:21.80
##  Max.   :911320502             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    fractal_dimension_mean
##  Min.   :0.00000   Min.   :0.00000     Min.   :0.1060   Min.   :0.04996
##  1st Qu.:0.02956   1st Qu.:0.02031     1st Qu.:0.1619   1st Qu.:0.05770
##  Median :0.06154   Median :0.03350     Median :0.1792   Median :0.06154
##  Mean   :0.08880   Mean   :0.04892     Mean   :0.1812   Mean   :0.06280
##  3rd Qu.:0.13070   3rd Qu.:0.07400     3rd Qu.:0.1957   3rd Qu.:0.06612
##  Max.   :0.42680   Max.   :0.20120     Max.   :0.3040   Max.   :0.09744
##    radius_se        texture_se      perimeter_se       area_se
##  Min.   :0.1115   Min.   :0.3602   Min.   : 0.757   Min.   :  6.802
##  1st Qu.:0.2324   1st Qu.:0.8339   1st Qu.: 1.606   1st Qu.: 17.850
##  Median :0.3242   Median :1.1080   Median : 2.287   Median : 24.530
##  Mean   :0.4052   Mean   :1.2169   Mean   : 2.866   Mean   : 40.337
##  3rd Qu.:0.4789   3rd Qu.:1.4740   3rd Qu.: 3.357   3rd Qu.: 45.190
##  Max.   :2.8730   Max.   :4.8850   Max.   :21.980   Max.   :542.200
##  smoothness_se      compactness_se      concavity_se     concave.points_se
##  Min.   :0.001713   Min.   :0.002252   Min.   :0.00000   Min.   :0.000000
##  1st Qu.:0.005169   1st Qu.:0.013080   1st Qu.:0.01509   1st Qu.:0.007638
##  Median :0.006380   Median :0.020450   Median :0.02589   Median :0.010930
##  Mean   :0.007041   Mean   :0.025478   Mean   :0.03189   Mean   :0.011796
##  3rd Qu.:0.008146   3rd Qu.:0.032450   3rd Qu.:0.04205   3rd Qu.:0.014710
##  Max.   :0.031130   Max.   :0.135400   Max.   :0.39600   Max.   :0.052790
##   symmetry_se       fractal_dimension_se  radius_worst   texture_worst
##  Min.   :0.007882   Min.   :0.0008948    Min.   : 7.93   Min.   :12.02
##  1st Qu.:0.015160   1st Qu.:0.0022480    1st Qu.:13.01   1st Qu.:21.08
##  Median :0.018730   Median :0.0031870    Median :14.97   Median :25.41
##  Mean   :0.020542   Mean   :0.0037949    Mean   :16.27   Mean   :25.68
##  3rd Qu.:0.023480   3rd Qu.:0.0045580    3rd Qu.:18.79   3rd Qu.:29.72
##  Max.   :0.078950   Max.   :0.0298400    Max.   :36.04   Max.   :49.54
##  perimeter_worst    area_worst     smoothness_worst  compactness_worst
##  Min.   : 50.41   Min.   : 185.2   Min.   :0.07117   Min.   :0.02729
##  1st Qu.: 84.11   1st Qu.: 515.3   1st Qu.:0.11660   1st Qu.:0.14720
##  Median : 97.66   Median : 686.5   Median :0.13130   Median :0.21190
##  Mean   :107.26   Mean   : 880.6   Mean   :0.13237   Mean   :0.25427
##  3rd Qu.:125.40   3rd Qu.:1084.0   3rd Qu.:0.14600   3rd Qu.:0.33910
##  Max.   :251.20   Max.   :4254.0   Max.   :0.22260   Max.   :1.05800
##  concavity_worst  concave.points_worst symmetry_worst   fractal_dimension_worst
##  Min.   :0.0000   Min.   :0.00000      Min.   :0.1565   Min.   :0.05504
##  1st Qu.:0.1145   1st Qu.:0.06493      1st Qu.:0.2504   1st Qu.:0.07146
##  Median :0.2267   Median :0.09993      Median :0.2822   Median :0.08004
```

```
##  Mean   :0.2722   Mean   :0.11461      Mean   :0.2901   Mean   :0.08395
##  3rd Qu.:0.3829   3rd Qu.:0.16140      3rd Qu.:0.3179   3rd Qu.:0.09208
##  Max.   :1.2520   Max.   :0.29100      Max.   :0.6638   Max.   :0.20750
##     X
##  Mode:logical
##  NA's:569
##
##
##
##
```

```
# head(diagnosis_data_full)
```

- From an intial examination of the dataset we have a target variable: diagnosis, with values of ‘B’ for benign and ‘M’ for malignant. There are no missing entries in the dataset,
- The last column, ‘X’ has no information and the ‘ID’ column is not useful for analysis.

3. Data Preparation

a) Data Modification

Column Removals

```
# Remove Lines for the ID and Null X final Columns
diagnosis_data_full <- diagnosis_data_full[2:32]
```

Scale Data Set

```
# Encoding the target feature as factor
# diagnosis_data_full$diagnosis = factor(diagnosis_data_full$diagnosis,
#                                        levels = c('B', 'M'),
#                                        labels = c(0,1))

# Scaling the dataset for models that require it
diagnosis_data_scaled <- diagnosis_data_full
diagnosis_data_scaled[,2:31] <- scale(diagnosis_data_scaled[,2:31])
str(diagnosis_data_scaled)
```

```
## 'data.frame':    569 obs. of  31 variables:
##  $ diagnosis              : Factor w/ 2 levels "B","M": 2 2 2 2 2 2 2 2 2 2 ...
##  $ radius_mean            : num  1.096 1.828 1.578 -0.768 1.749 ...
##  $ texture_mean           : num  -2.072 -0.353 0.456 0.254 -1.151 ...
##  $ perimeter_mean         : num  1.269 1.684 1.565 -0.592 1.775 ...
##  $ area_mean              : num  0.984 1.907 1.558 -0.764 1.825 ...
##  $ smoothness_mean        : num  1.567 -0.826 0.941 3.281 0.28 ...
##  $ compactness_mean       : num  3.281 -0.487 1.052 3.4 0.539 ...
##  $ concavity_mean         : num  2.6505 -0.0238 1.3623 1.9142 1.3698 ...
##  $ concave.points_mean    : num  2.53 0.548 2.035 1.45 1.427 ...
##  $ symmetry_mean          : num  2.21557 0.00139 0.93886 2.86486 -0.00955 ...
##  $ fractal_dimension_mean : num  2.254 -0.868 -0.398 4.907 -0.562 ...
##  $ radius_se              : num  2.488 0.499 1.228 0.326 1.269 ...
##  $ texture_se             : num  -0.565 -0.875 -0.779 -0.11 -0.79 ...
##  $ perimeter_se           : num  2.831 0.263 0.85 0.286 1.272 ...
##  $ area_se                : num  2.485 0.742 1.18 -0.288 1.189 ...
##  $ smoothness_se          : num  -0.214 -0.605 -0.297 0.689 1.482 ...
##  $ compactness_se         : num  1.3157 -0.6923 0.8143 2.7419 -0.0485 ...
##  $ concavity_se           : num  0.723 -0.44 0.213 0.819 0.828 ...
##  $ concave.points_se      : num  0.66 0.26 1.42 1.11 1.14 ...
##  $ symmetry_se            : num  1.148 -0.805 0.237 4.729 -0.361 ...
##  $ fractal_dimension_se   : num  0.9063 -0.0994 0.2933 2.0457 0.4989 ...
##  $ radius_worst           : num  1.885 1.804 1.511 -0.281 1.297 ...
##  $ texture_worst          : num  -1.358 -0.369 -0.024 0.134 -1.465 ...
##  $ perimeter_worst        : num  2.3 1.53 1.35 -0.25 1.34 ...
##  $ area_worst             : num  2 1.89 1.46 -0.55 1.22 ...
##  $ smoothness_worst       : num  1.307 -0.375 0.527 3.391 0.22 ...
##  $ compactness_worst      : num  2.614 -0.43 1.082 3.89 -0.313 ...
##  $ concavity_worst        : num  2.108 -0.147 0.854 1.988 0.613 ...
##  $ concave.points_worst   : num  2.294 1.086 1.953 2.174 0.729 ...
##  $ symmetry_worst         : num  2.748 -0.244 1.151 6.041 -0.868 ...
##  $ fractal_dimension_worst: num  1.935 0.281 0.201 4.931 -0.397 ...
```

```
names(diagnosis_data_full)
```

```
##  [1] "diagnosis"               "radius_mean"
##  [3] "texture_mean"            "perimeter_mean"
##  [5] "area_mean"               "smoothness_mean"
##  [7] "compactness_mean"        "concavity_mean"
##  [9] "concave.points_mean"     "symmetry_mean"
## [11] "fractal_dimension_mean"  "radius_se"
## [13] "texture_se"              "perimeter_se"
## [15] "area_se"                 "smoothness_se"
## [17] "compactness_se"          "concavity_se"
## [19] "concave.points_se"       "symmetry_se"
## [21] "fractal_dimension_se"    "radius_worst"
## [23] "texture_worst"           "perimeter_worst"
## [25] "area_worst"              "smoothness_worst"
## [27] "compactness_worst"       "concavity_worst"
## [29] "concave.points_worst"    "symmetry_worst"
## [31] "fractal_dimension_worst"
```

Split Data into Train and Test Sets

- Here we are splitting the datatsets in test and train datasets which alows for running the models.

```
# Split the data into a train set and a test set
diagnosis_data_train <- diagnosis_data_full[1:426,]
diagnosis_data_test <- diagnosis_data_full[427:569,]
# Split the scaled version as well
diag_data_scaled_train <- diagnosis_data_scaled[1:426,]
diag_data_scaled_test <- diagnosis_data_scaled[427:569,]
```

The Dependent Variable

- The dependant variable in our analysis ‘Diagnosis’ of cancer was of interest during data exploration. It has slight imbalance data and binary classification of benign and malignant. We can check it by looking at the breakdown on the dependent variable.

```
prop.table(table(diagnosis_data_train$diagnosis))*100
```

```
##
##       B       M
## 58.4507 41.5493
```

```
prop.table(table(diagnosis_data_test$diagnosis))*100
```

```
##
##        B        M
## 75.52448 24.47552
```

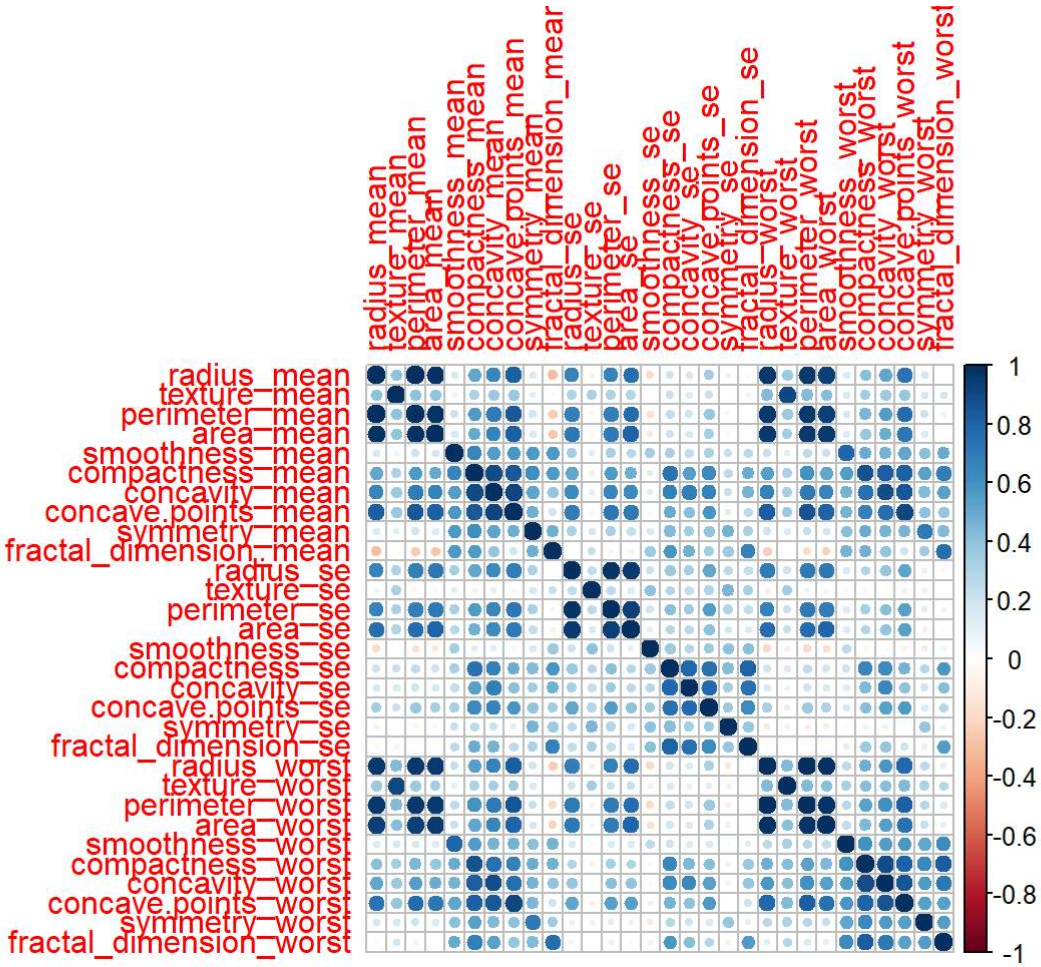
- This simple split is fine as it gives greater importance to ‘M’ outcomes for training which is desirable for the high sensitivity needed.
- The target variable distribution also confirms that class imbalance is not an issue.

b) Feature Engineering

We needed to do feature engineering as there were lot of correlated features, for this particular use case, we decided to ensure that any and all possible relationships between input and output variables can be explored (non-linear or linear), and since random forest in general gives feature importance we used that feature from random forest liabrary, we decided to keep 4 importantfeatures out of 32 to give us better prediction.

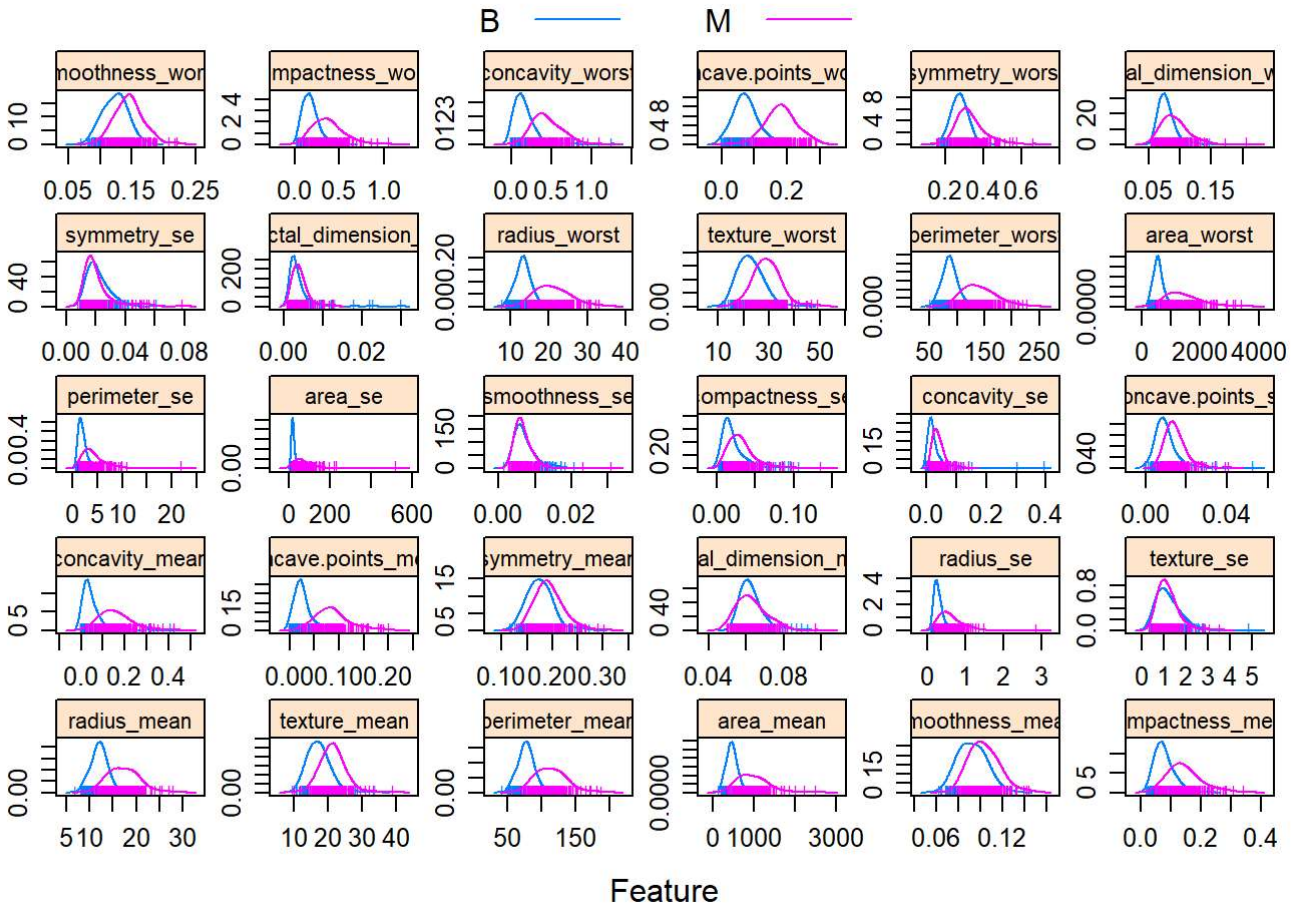
View Correlation Matrix to explore highly correlated features

```
M <- cor(diagnosis_data_train[,2:31])
# 24) perimeter_worst, 19) concave.points_worst, 22) radius_worst, 9) concave.points_mean, 8) co
ncavity_mean
# M <- cor(diagnosis_data_full[24, 19, 22, 9, 8])
corrplot(M, method="circle", type="full")
```



Check Feature variables distribution vs Target

```
featurePlot(x = diagnosis_data_train[, 2:31],
            y = diagnosis_data_train$diagnosis,
            plot = "density",
            strip=strip.custom(par.strip.text=list(cex=.7)),
            scales = list(x = list(relation="free"),
                          y = list(relation="free")), layout = c(6, 5), adjust = 1.5, pch = "|",
auto.key=list(columns=2))
```



- Looking at this output we can see that the following features show higher density dirvergence w.r.t. the 2 target classes:
    - “compactness_worst”
    - “concavity_worst”
    - “concave.points_worst”
    - “radius_worst”
    - “texture_worst”
    - “perimeter_worst”
    - “area_worst”
    - “concavity_mean”
    - “concave.points_mean”
    - “radius_mean”
    - “texture_mean”
    - “perimeter_mean”
    - “area_mean”
    - “compactness_mean”

Try a recursive feature elimination check - Feature Selection Method 1

We tried multiple methods to decide which variables to use for feature importance. First one is recursive feature elimination technique as below.

```
set.seed(100)
options(warn=-1)

subsets <- c(1:5, 15, 20, 25, 31)

ctrl <- rfeControl(functions = rfFuncs,
                   method = "repeatedcv",
                   repeats = 5,
                   verbose = FALSE)

lmProfile <- rfe(x=diagnosis_data_train[, 2:31], y=diagnosis_data_train$diagnosis,
                 sizes = subsets,
                 rfeControl = ctrl)

lmProfile
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (10 fold, repeated 5 times)
##
## Resampling performance over subset size:
##
##  Variables Accuracy  Kappa AccuracySD KappaSD Selected
##          1   0.8657 0.7236    0.04637 0.09417
##          2   0.8991 0.7915    0.04208 0.08706
##          3   0.9103 0.8149    0.03806 0.07821
##          4   0.9253 0.8462    0.03676 0.07571
##          5   0.9426 0.8818    0.03656 0.07480
##         15   0.9545 0.9063    0.02566 0.05263
##         20   0.9563 0.9100    0.02465 0.05074
##         25   0.9573 0.9120    0.02582 0.05318        *
##         30   0.9569 0.9112    0.02605 0.05339
##
## The top 5 variables (out of 25):
##    perimeter_worst, area_worst, radius_worst, concave.points_worst, texture_worst
```

- Key Output: The top 5 variables (out of 25): perimeter_worst, area_worst, radius_worst, concave.points_worst, texture_worst
- These 5 were also identified in the Feature variables distribution check. It is possible that other suitable variables were left out due to strong correlations.

4. Data Modeling

- Ethical Framework Questions:
    - Does your use case require a more interpretable algorithm? Accuracy is more impotant than interpretability for this study.
    - Should you be optimizing for a different outcome than accuracy to make your outcomes fairer? Since there is no demographic data, fairness would be hard to determine.
    - Is it possible that a malicious actor has compromised training data and created misleading results? No. The data is from a reputable source.

4. A) Data Modeling - Random Forest

Build a Random Forest Model based on all values to start.

This gives us a model that is resistant to overfitting.

```
diag_rf_model <- randomForest(factor(diagnosis) ~ radius_mean + texture_mean + perimeter_mean +
                              smoothness_mean + compactness_mean + concavity_mean + concave.points
_mean +
                              symmetry_mean + fractal_dimension_mean +
                              radius_se + texture_se + perimeter_se +
                              smoothness_se + compactness_se + concavity_se + concave.points_se +
                              symmetry_se + fractal_dimension_se +
                              radius_worst + texture_worst + perimeter_worst +
                              smoothness_worst + compactness_worst + concavity_worst + concave.poi
nts_worst +
                              symmetry_worst + fractal_dimension_worst,
                            data = diagnosis_data_train)
diag_rf_model
```

```
##
## Call:
##  randomForest(formula = factor(diagnosis) ~ radius_mean + texture_mean +      perimeter_mean
+ smoothness_mean + compactness_mean + concavity_mean +      concave.points_mean + symmetry_mean
+ fractal_dimension_mean +      radius_se + texture_se + perimeter_se + smoothness_se + compactn
ess_se +      concavity_se + concave.points_se + symmetry_se + fractal_dimension_se +      radiu
s_worst + texture_worst + perimeter_worst + smoothness_worst +      compactness_worst + concavit
y_worst + concave.points_worst +      symmetry_worst + fractal_dimension_worst, data = diagnosis
_data_train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 4.69%
## Confusion matrix:
##     B   M class.error
## B 239  10  0.04016064
## M  10 167  0.05649718
```

5. A) Data Evaluation - Random Forest

Test the Random Forest Model

```
pre_rf <- predict(diag_rf_model, diagnosis_data_test[,-c(1,1)])
cm_rf <- confusionMatrix(pre_rf, diagnosis_data_test$diagnosis)
cm_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   B   M
##          B 105   0
##          M   3  35
##
##                Accuracy : 0.979
##                  95% CI : (0.9399, 0.9957)
##     No Information Rate : 0.7552
##     P-Value [Acc > NIR] : 6.397e-14
##
##                   Kappa : 0.9449
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 0.9722
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9211
##              Prevalence : 0.7552
##          Detection Rate : 0.7343
##    Detection Prevalence : 0.7343
##       Balanced Accuracy : 0.9861
##
##        'Positive' Class : B
##
```

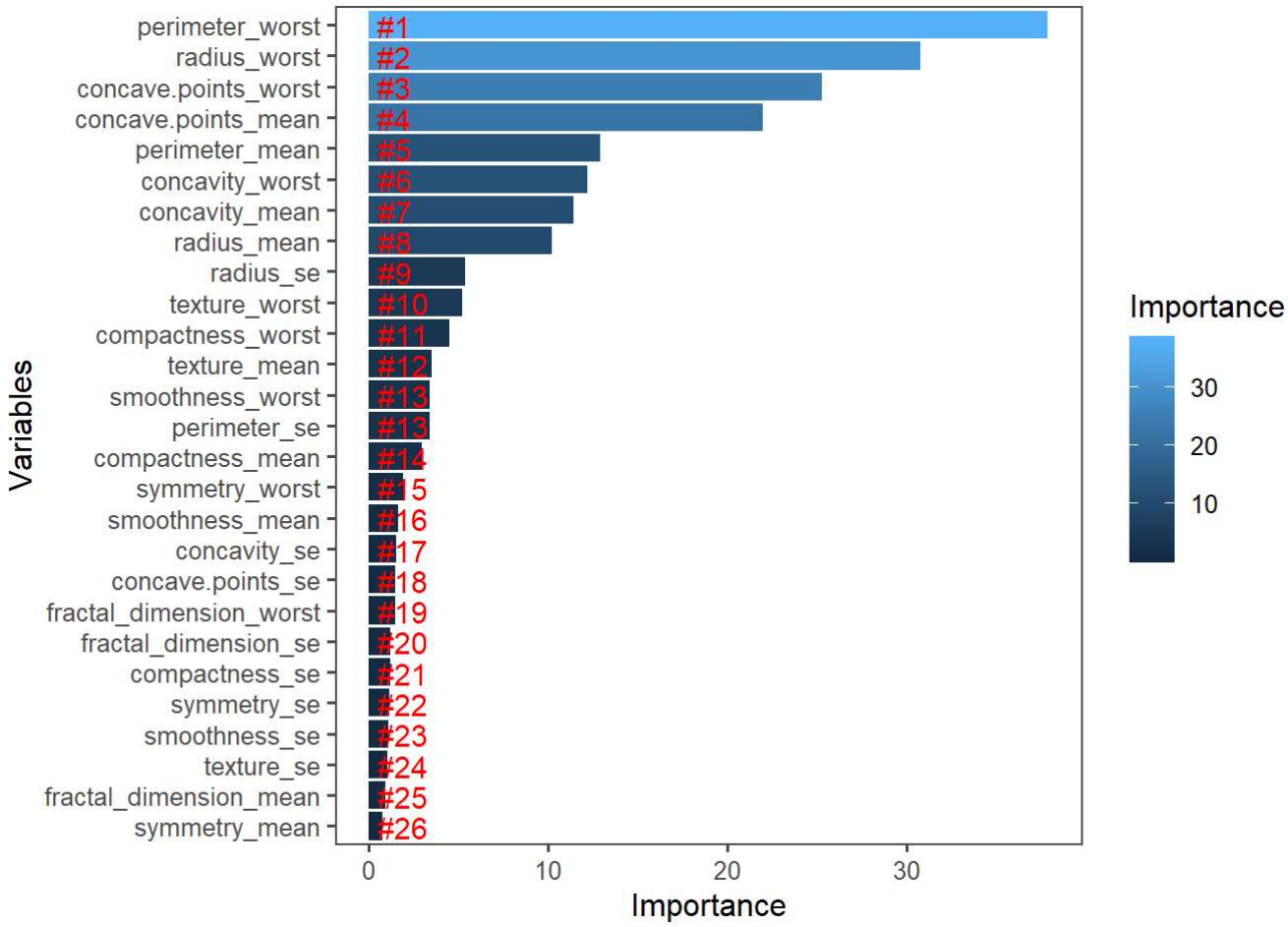
Feature Engineering Continued - Feature Selection Method 2

Examine the Random Forest Model for feature importance

```
# Get importance
importance    <- importance(diag_rf_model)
varImportance <- data.frame(Variables = row.names(importance),
                            Importance = round(importance[ ,'MeanDecreaseGini'],2))

# Create a rank variable based on importance
rankImportance <- varImportance %>%
  mutate(Rank = paste0('#',dense_rank(desc(Importance))))

# Use ggplot2 to visualize the relative importance of variables
ggplot(rankImportance, aes(x = reorder(Variables, Importance),  y = Importance, fill = Importanc
e)) +
    geom_bar(stat='identity') +
    geom_text(aes(x = Variables, y = 0.5, label = Rank), hjust=0, vjust=0.55, size = 4, colour =
'red') +
    labs(x = 'Variables') + coord_flip() +  theme_few()
```



- Shows the following 5 features as most important: perimeter_worst, concave.points_worst,radius_worst, concave.points_mean, concavity_mean
    - Again, these 5 were also identified in the Feature variables distribution check.
    - Since it appears that this method considered a wider range of variables we will proceed with this outcome.

Features Selected: 24) perimeter_worst, 19) concave.points_worst, 22) radius_worst, 9) concave.points_mean, 8) concavity_mean

- Now lets try a Logistic Regression Model using those top 5 features

4. B) Data Modeling - Logistic Regression

```
library('ranger')
dim(breastcancer_data)
```

```
## [1] 569  33
```

```
# head(breastcancer_data,6)
# summary(breastcancer_data)
names(breastcancer_data)
```

```
##  [1] "id"                      "diagnosis"
##  [3] "radius_mean"             "texture_mean"
##  [5] "perimeter_mean"          "area_mean"
##  [7] "smoothness_mean"         "compactness_mean"
##  [9] "concavity_mean"          "concave.points_mean"
## [11] "symmetry_mean"           "fractal_dimension_mean"
## [13] "radius_se"               "texture_se"
## [15] "perimeter_se"            "area_se"
## [17] "smoothness_se"           "compactness_se"
## [19] "concavity_se"            "concave.points_se"
## [21] "symmetry_se"             "fractal_dimension_se"
## [23] "radius_worst"            "texture_worst"
## [25] "perimeter_worst"         "area_worst"
## [27] "smoothness_worst"        "compactness_worst"
## [29] "concavity_worst"         "concave.points_worst"
## [31] "symmetry_worst"          "fractal_dimension_worst"
## [33] "X"
```

```
# summarize the class distribution
percentage <- prop.table(table(breastcancer_data$diagnosis)) * 100
cbind(freq=table(breastcancer_data$diagnosis), percentage=percentage)
```

```
##   freq percentage
## B  357   62.74165
## M  212   37.25835
```

```
#remove id and x
target <- ifelse(breastcancer_data$diagnosis=="B", 1, 0)
#head(target)
model_1 = select (breastcancer_data,-c(X,id,diagnosis))
nobs <- nrow(model_1)
nobs
```

```
## [1] 569
```

```
# head(model_1)
model_2=scale(model_1)
model_3<-data.frame(cbind(model_2,target))
# summary(model_3)

#model start
library(caTools)
set.seed(123)
split = sample.split(model_3$target, SplitRatio = 0.75)
train_data = subset(model_3, split == TRUE)
test_data = subset(model_3, split == FALSE)

dim(train_data)
```

```
## [1] 427  31
```

```
# Logistic_Model <- glm(target ~ perimeter_mean +
#                         + smoothness_mean + compactness_mean + concavity_mean + concave.points
_mean +
#                         symmetry_mean ,data=train_data, family = binomial)
Logistic_Model <- glm(target ~ perimeter_worst + concave.points_worst + radius_worst + concave.p
oints_mean + concavity_mean,
                      data=train_data, family = binomial)

summary(Logistic_Model)
```

```
##
## Call:
## glm(formula = target ~ perimeter_worst + concave.points_worst +
##     radius_worst + concave.points_mean + concavity_mean, family = binomial,
##     data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3039  -0.0167   0.0306   0.1339   1.7468
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)             0.8400     0.2837   2.961 0.003062 **
## perimeter_worst         3.8812     3.0649   1.266 0.205398
## concave.points_worst   -2.4012     0.7159  -3.354 0.000796 ***
## radius_worst           -8.1366     3.0367  -2.679 0.007375 **
## concave.points_mean    -0.2579     0.9038  -0.285 0.775328
## concavity_mean         -0.6809     0.5014  -1.358 0.174423
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 563.81  on 426  degrees of freedom
## Residual deviance: 109.50  on 421  degrees of freedom
## AIC: 121.5
##
## Number of Fisher Scoring iterations: 8
```

5. B) Data Evaluation - Logistic Regression

Test Logistic Regression Model

```
predictTrain = predict(Logistic_Model, type="response")
summary(predictTrain)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.01884 0.96378 0.62763 0.99707 1.00000
```

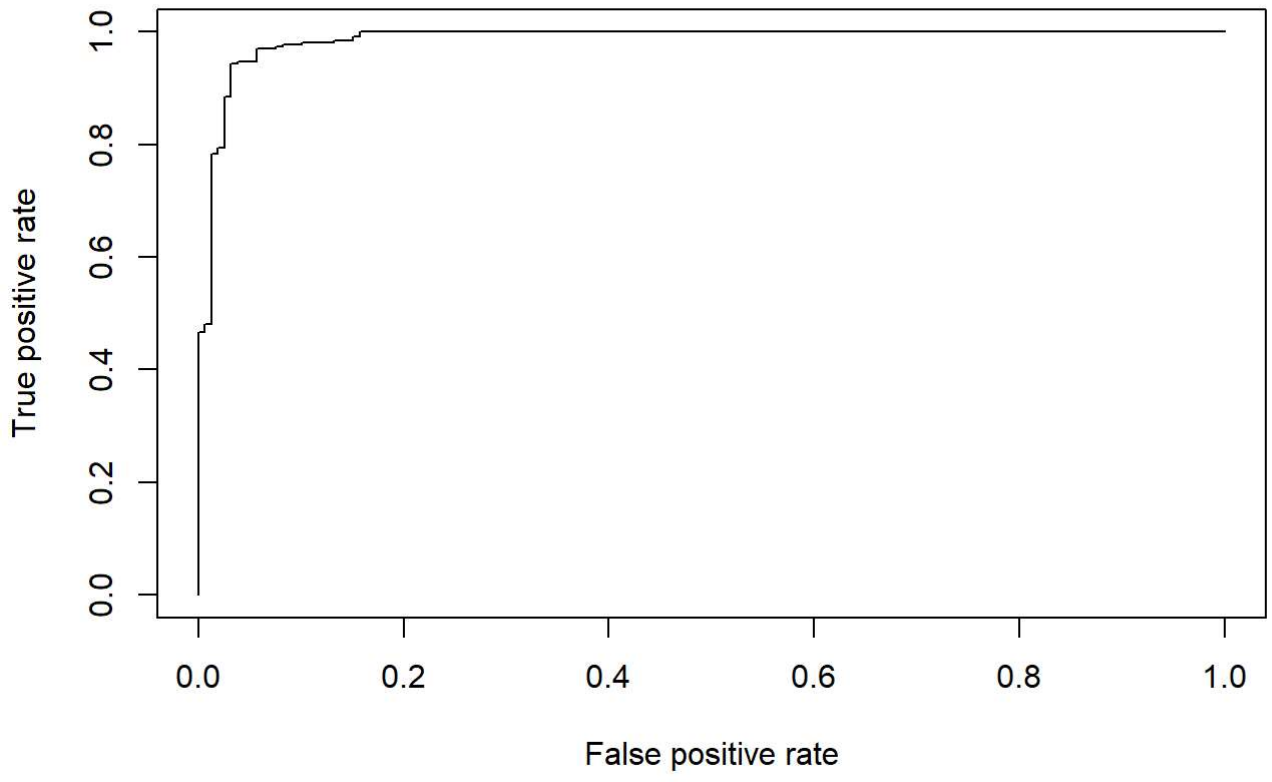
```
tapply(predictTrain, train_data$target, mean)
```

```
##         0         1
## 0.0984821 0.9415722
```

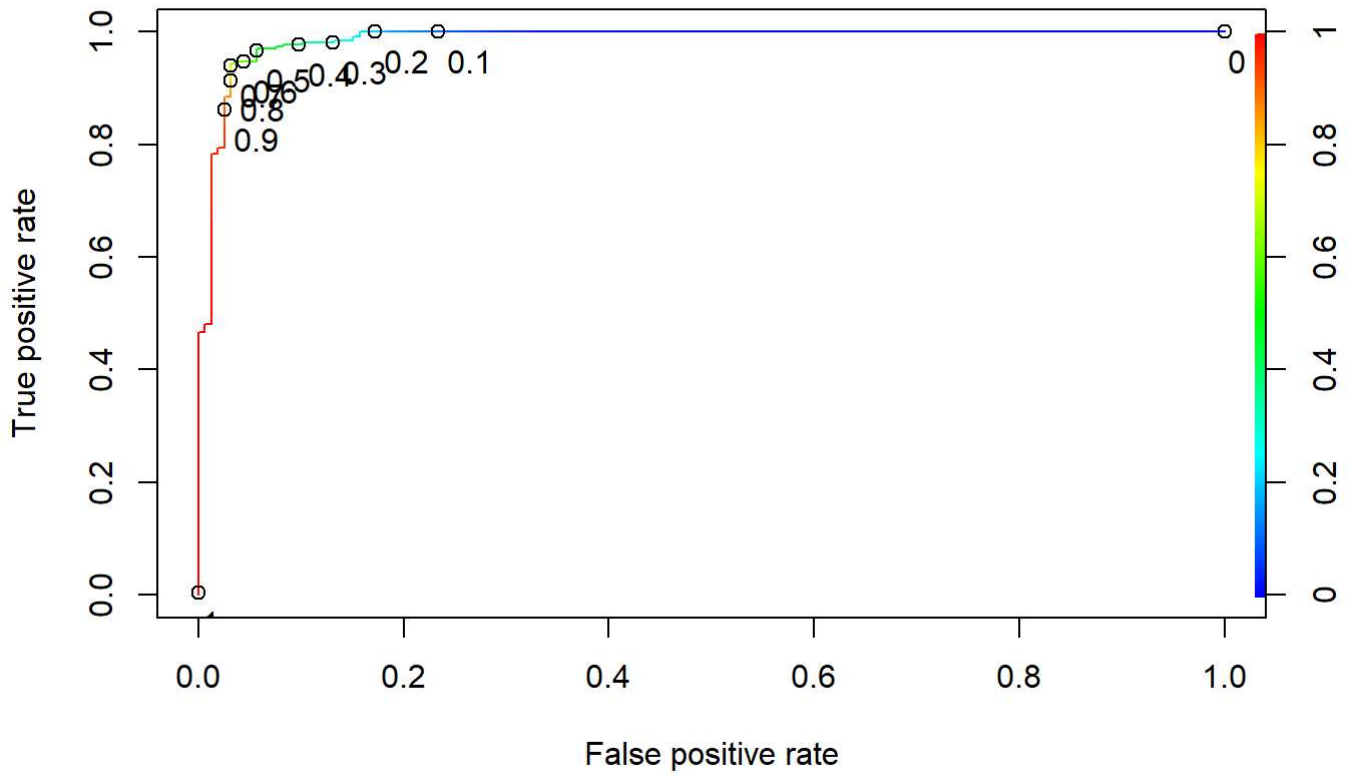
```
table(train_data$target, predictTrain > 0.5)
```

```
##
##     FALSE TRUE
##   0   150    9
##   1     9  259
```

```
# lg_cm <- confusionMatrix(table(train_data$target, predictTrain > 0.5))
# lg_cm
# install.packages("ROCR")
library(ROCR)
ROCRpred = prediction(predictTrain, train_data$target)
ROCRperf = performance(ROCRpred, "tpr", "fpr")
plot(ROCRperf)
```



```
plot(ROCRperf, colorize=TRUE)
plot(ROCRperf, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-0.2,1.7))
```




```
predictTest = predict(Logistic_Model, type = "response", newdata = test_data)
summary(predictTest)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.01663 0.95183 0.62793 0.99790 1.00000
```

```
library('MLmetrics')
pred <- ifelse(Logistic_Model$fitted.values<0.5, 0, 1)
cm_lg <- ConfusionMatrix(y_pred = pred, y_true = train_data$target)
cm_lg2 <- confusionMatrix(cm_lg)
cm_lg2
```

```
## Confusion Matrix and Statistics
##
##       y_pred
## y_true   0   1
##      0 150   9
##      1   9 259
##
##                Accuracy : 0.9578
##                  95% CI : (0.9342, 0.9748)
##     No Information Rate : 0.6276
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9098
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9434
##             Specificity : 0.9664
##          Pos Pred Value : 0.9434
##          Neg Pred Value : 0.9664
##              Prevalence : 0.3724
##          Detection Rate : 0.3513
##    Detection Prevalence : 0.3724
##       Balanced Accuracy : 0.9549
##
##        'Positive' Class : 0
##
```

- Next lets try a Logistic Neural Network using those top 5 features

4. C) Data Modeling - Neural Network

Universal approximation theorem states that simple ariticial neural networks with only one hidden layer has the potentail to represent almost any contineous functions with nicely assigned parameters and a proper non-polymonial activation function such as signoid or rectified linear unit.

R has a package nnet, which can be used for Classification and Regression. We decided to try a simple ANN model.

```
# load the package "nnet" which provide the ANN modeling functionality
library("nnet")
```

preprocess the data for modeling

```
# load the data into a data frame
library('readr')
diagnosis_data_full = read.csv("./input/diagnosis_data.csv")

# set up the function for data normalization
normalize <- function(x) {return ((x - min(x)) / (max(x) - min(x)))}

# normalize the numeric data from column 3 to column 32
maxmindf <- as.data.frame(lapply(diagnosis_data_full[3:32], normalize))

# normalize the factor column, transform M -> -1 and B -> 1

# set up the mapping function to use
myMapper <- function(x) { if(x=="M") {temp <- -1} else {temp <- 1}; return (temp) }

# load the libary "purrr" for the function map
library("purrr")

# pull out column #2 which is the factor column for prediction and transform/normalize it.
factors <- diagnosis_data_full[2:2]
factors$c2 <- unlist(map(diagnosis_data_full$diagnosis, myMapper))
nfactors <- as.data.frame(lapply(factors[2:2], normalize))

# combine the normalized input and output columns into one data frame
cleanData <- cbind(nfactors, maxmindf)

# split the data into trainset and testset
trainset <- cleanData[1:426,]
testset <- cleanData[427:569,]
```

Data modeling with the cleaned and normalized data set

```
# based on the importance analysis result, we rebuild the ANN with less input variables from 30
 to 10
# we found that this will preserve the accuracy as told by the confusionMatrix
# while we reduced the ANN parameters from 161 to 61.
# model becomes more resilent and able to catch the most essential information

# set the seed with nice prime number, so make the training re-producible
set.seed(887)
# we use 5 neurons in the hidden layer
#
fit_net_10<-nnet(c2~perimeter_worst + concave.points_worst + radius_worst + concave.points_mean
 + concavity_mean,
                 data=trainset,size=5, decay=5e-4, maxit=2000)
```

```
## # weights:  36
## initial  value 120.319089
## iter  10 value 19.731074
## iter  20 value 16.877115
## iter  30 value 15.905767
## iter  40 value 15.243169
## iter  50 value 14.685755
## iter  60 value 13.516136
## iter  70 value 12.714979
## iter  80 value 12.215396
## iter  90 value 12.043982
## iter 100 value 11.703608
## iter 110 value 11.422878
## iter 120 value 11.311326
## iter 130 value 11.220984
## iter 140 value 11.076634
## iter 150 value 11.022681
## iter 160 value 10.975679
## iter 170 value 10.951949
## iter 180 value 10.933711
## iter 190 value 10.886404
## iter 200 value 10.849149
## iter 210 value 10.789595
## iter 220 value 10.747267
## iter 230 value 10.642256
## iter 240 value 10.491664
## iter 250 value 10.066835
## iter 260 value 9.427088
## iter 270 value 9.161631
## iter 280 value 8.962918
## iter 290 value 8.758192
## iter 300 value 8.624465
## iter 310 value 8.596763
## iter 320 value 8.586525
## iter 330 value 8.578700
## iter 340 value 8.575303
## iter 350 value 8.574417
## iter 360 value 8.573425
## iter 370 value 8.572592
## iter 380 value 8.572084
## iter 390 value 8.571788
## iter 400 value 8.571564
## iter 410 value 8.571490
## final  value 8.571489
## converged
```

```
fit_net_10
```

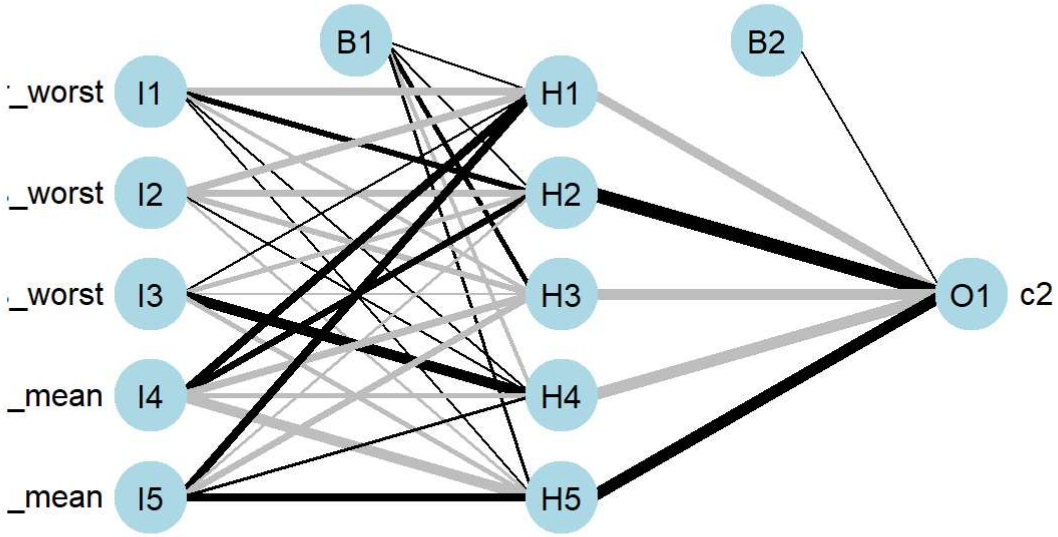
```
## a 5-5-1 network with 36 weights
## inputs: perimeter_worst concave.points_worst radius_worst concave.points_mean concavity_mean
## output(s): c2
## options were - decay=5e-04
```

5. C) Model evaluation - Neural Network

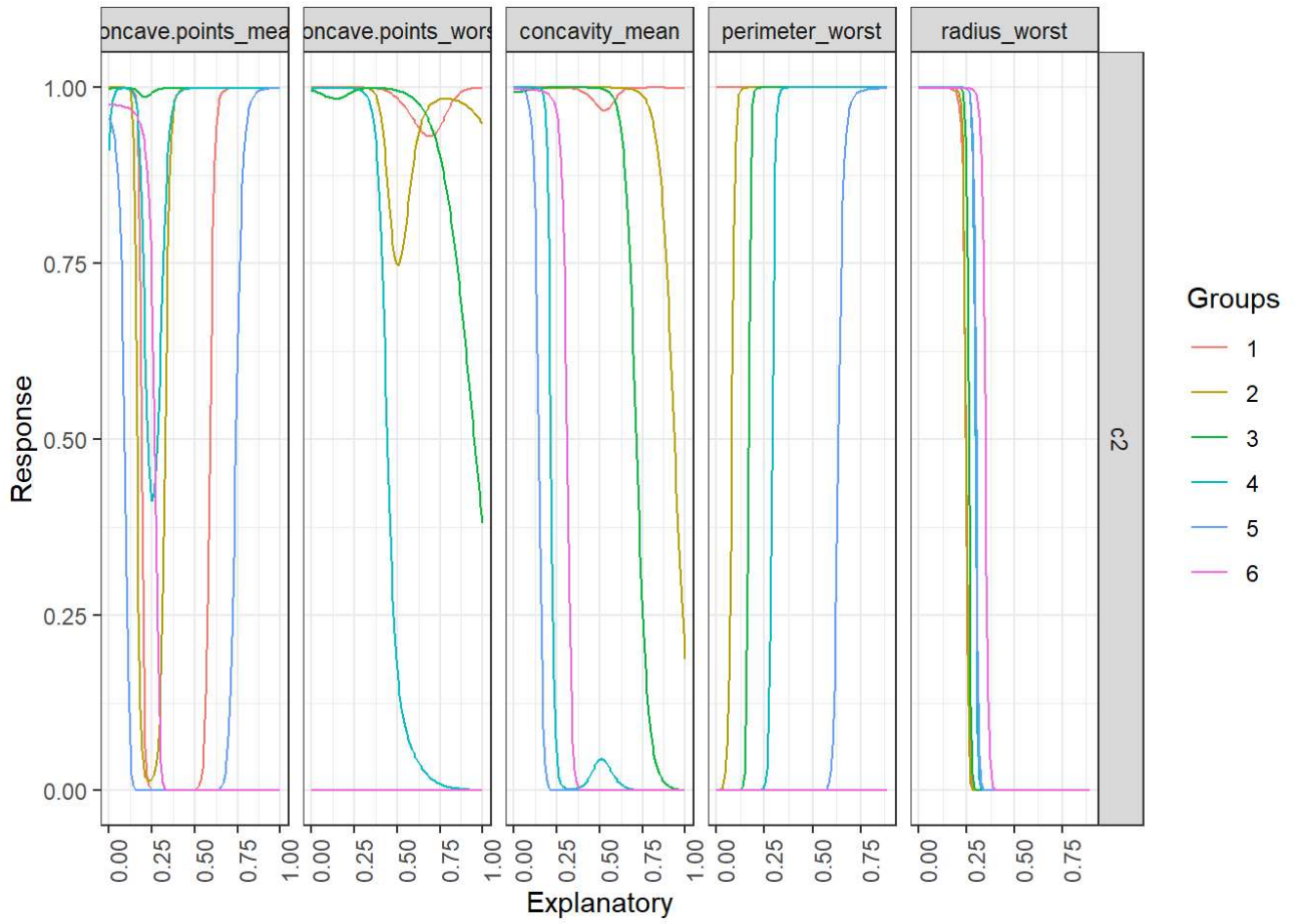
```
# put the predicted values and original values into one single data frame
library('NeuralNetTools')
predictions_10 <- data.frame(cbind(round(predict(fit_net_10, testset),digits=0), testset$c2))
cm_nn <- table(predictions_10$X1,predictions_10$X2)
confusionMatrix(cm_nn)
```

```
## Confusion Matrix and Statistics
##
##
##       0   1
##   0  35   3
##   1   0 105
##
##                Accuracy : 0.979
##                  95% CI : (0.9399, 0.9957)
##     No Information Rate : 0.7552
##     P-Value [Acc > NIR] : 6.397e-14
##
##                   Kappa : 0.9449
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 1.0000
##             Specificity : 0.9722
##          Pos Pred Value : 0.9211
##          Neg Pred Value : 1.0000
##              Prevalence : 0.2448
##          Detection Rate : 0.2448
##    Detection Prevalence : 0.2657
##       Balanced Accuracy : 0.9861
##
##        'Positive' Class : 0
##
```

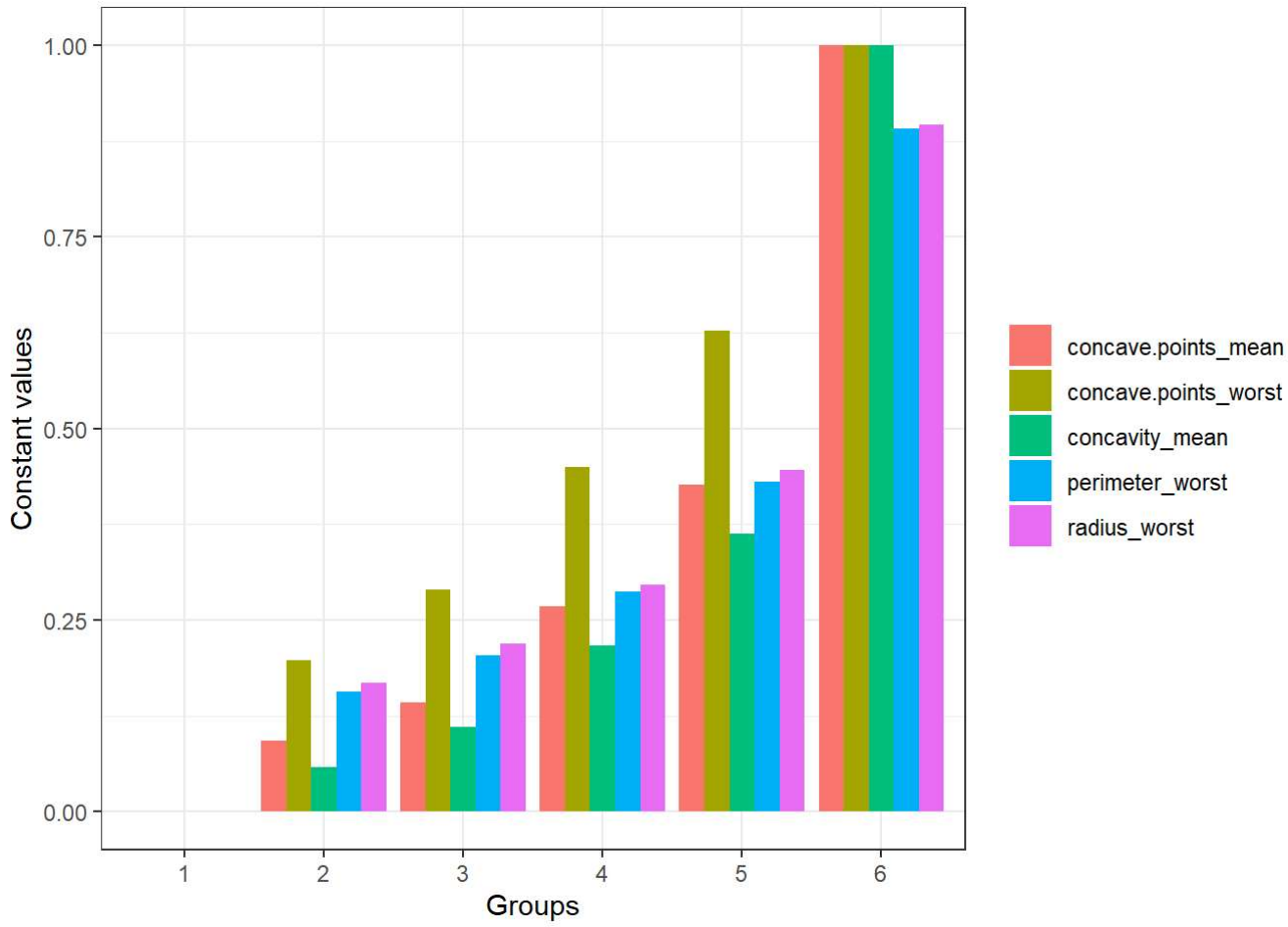
```
NID_10 <- NeuralNetTools::plotnet(fit_net_10)
```



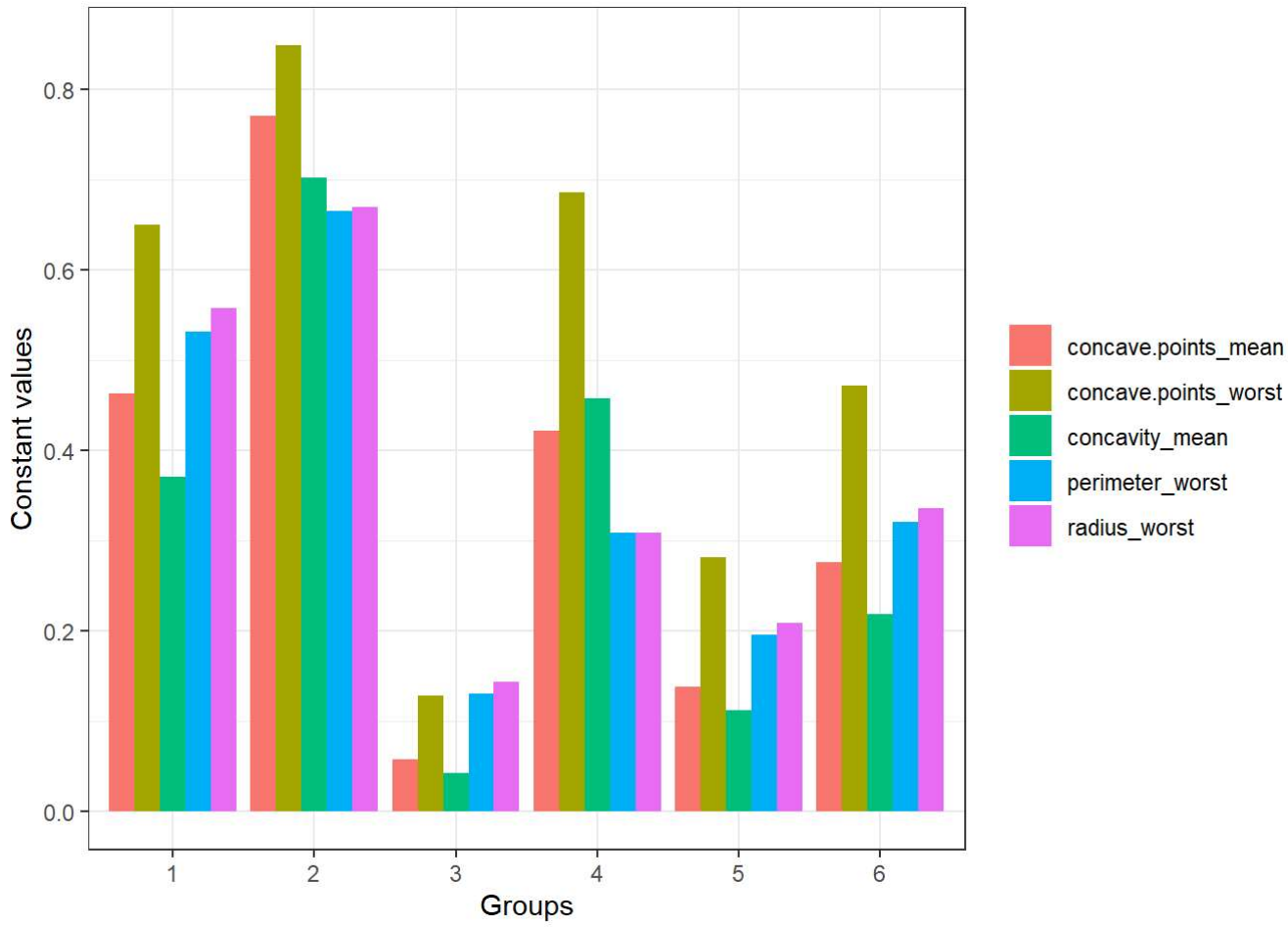

```
sensitivity_lekprofile_10 <- lekprofile(fit_net_10) + theme(axis.text.x = element_text(angle = 9
0))
sensitivity_lekprofile_10
```




```
sensitivity_lekprofile_Group_10 <- lekprofile(fit_net_10, group_show = TRUE)
sensitivity_lekprofile_Group_10
```




```
sensitivity_lekprofile_vals_10 <- lekprofile(fit_net_10, group_vals=6, group_show = TRUE)
sensitivity_lekprofile_vals_10
```



- Finally lets try a kernelSVM Model using those top 5 features and using the scaled dataset

4. D) Data Modeling - kernelSVM

```
dt <- diag_data_scaled_train[,-c(1)]
diag_kSVM_model <- svm(formula = diag_data_scaled_train$diagnosis ~.,
                 data = diag_data_scaled_train[,-c(1)],
                 type = 'C-classification',
                 kernel = 'radial')
```

5. D) Data Evaluation - kernelSVM

Test the kernel SVM model.

```
diag_kSVM_pre <- predict(diag_kSVM_model, diag_data_scaled_test[,-c(1)])
cm_kSVM <- confusionMatrix(diag_kSVM_pre, diag_data_scaled_test$diagnosis)
cm_kSVM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   B   M
##          B 105   0
##          M   3  35
##
##                Accuracy : 0.979
##                  95% CI : (0.9399, 0.9957)
##     No Information Rate : 0.7552
##     P-Value [Acc > NIR] : 6.397e-14
##
##                   Kappa : 0.9449
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 0.9722
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9211
##              Prevalence : 0.7552
##          Detection Rate : 0.7343
##    Detection Prevalence : 0.7343
##       Balanced Accuracy : 0.9861
##
##        'Positive' Class : B
##
```

6. Final Model Analysis and Selection

Cost Analysis

This is a situation where we want to predict cancer cells in a patient based on the MRI done in the Hospital. Previously this kind of prediction was done using the historical data of previous cancer patients over the years and compare them with current data points. By using feature selection to filter out irrelevant features and using model pipeline to go through numerous features we select the best machine learning models based on measures such as specifity and sensitivity. Two models we focus on primarily are and Decision Tree abd SVM as they give best predictive results. The predictions we are focused on when working with cancer use cases. Before latest machine learning techniques this was done with around 33 variables based on historical data for previous patients. Now the number of features has exponentially increased due to various kinds of data being used such as clinical data, genomic data of patients and tumor related data. The issue here again same as the fraud problem of imbalanced Dataset. This is a common problem with these use cases which is an imbalance of predictive events with parameters (too few events, too many parameters), overtraining, and a lack of external validation or testing. What is important in this case is the sample per ratio of training set. Size and variety if training set is important. Since we want to predict a minority class of values, we need a Representative sample of the training and test data. It is possible that we do not fully utilize the representative sample and model uses bias or a selected set of training data and keeps training on it. What ends up happening is that Training too many times on too few examples with too little variety leads to the phenomenon of over-training or simply training on noise. When it is asked to predict new target then the model fails spectacularly. Just using accuracy as a measure would not yield good results as we have imbalanced dataset. Generally, this problem deals with the trade-off between recall (percent of truly positive instances that were classified as such) and precision (percent of positive classifications that are truly

positive). Since we always want to detect instances of a minority class in our case cancer cells out of set of good cells in our body. It is usually more expensive to miss a positive cancer cells than to falsely label a cell which is not malcontent to be tumor cells. It is best to avoid Accuracy and use measures such as Precision and AUROC.

Model Comparison

- The following Machine Learning Algorithms were used in this analysis: Accuracy Sesitivity Specificity
    - Random Forest 0,9790 0.9722 1.0000
    - Logistic Regression 0.9578 0.9434 0.9664
    - Neural Net 0.9790 1.0000 0.9722
    - kernel SVM 0.9790 0.9722 1.0000

Size of Training Data Size of data is the key in our problem set. If our data is small with a smaller number of features then our training set is small as well, it results in high bias/low variance classifiers such as Naive Bayes. These classifiers have an edge of overfitting over the set of low bias/high variance classifiers which would be K-NN. Hence over time these classifiers start to perform better as training set size increases meaning the error decreases in the cost function. These high bias classifiers end up being quite meek for a good case of accurate predictive modeling. In the following paragraphs I would be doing comparison between different algorithms to show the strength and accuracy of each classification algorithm in term of performance efficiency and time complexity.

Features of Random Forest: Random Forest is a classification and regression algorithm. Here, we train several decision trees. The original learning dataset is randomly divided into several subsets of equal size. A decision tree is trained for each subset.

Advantages: - Robust to overfitting (thus solving one of the biggest disadvantages of decision trees) - Parameterization remains quite simple and intuitive - Performs very well when the number of features is big and for large quantity of learning data

Disadvantages: - Models generated with Random Forest may take a lot of memory - Learning may be slow (depending on the parameterization) - Not possible to iteratively improve the generated models

Logistic Regression: Logistic Regression Model is a generalized form of Linear Regression Model. It is a very good Discrimination Tool. Following are the advantages and disadvantage of Logistic Regression:

Advantages: - Logistic Regression performs well when the dataset is linearly separable. - Logistic regression is less prone to over-fitting but it can overfit in high dimensional datasets. You should consider Regularization (L1 and L2) techniques to avoid over-fitting in these scenarios. - Logistic Regression not only gives a measure of how relevant a predictor (coefficient size) is, but also its direction of association (positive or negative). - Logistic regression is easier to implement, interpret and very efficient to train.

Disadvantages: - Main limitation of Logistic Regression is the assumption of linearity between the dependent variable and the independent variables. In the real world, the data is rarely linearly separable. Most of the time data would be a jumbled mess. - If the number of observations are lesser than the number of features, Logistic Regression should not be used, otherwise it may lead to overfit. - Logistic Regression can only be used to predict discrete functions. Therefore, the dependent variable of Logistic Regression is restricted to the discrete number set. This restriction itself is problematic, as it is prohibitive to the prediction of continuous data.

Neural Network Advantages: - Complicated functions and non-linear problems can be solved easily by Neural network. - Can use ensembling with other techniques to get good solution. Disadvantage - Much Slower for training and classification - Hard to interpret, - Data comes in streams - Not usable with small datasets.

Kernel SVM Features of SVMs: Support Vector machine is a classification algorithm used primarily with text classification problems. It uses hyperplane to separate out different cluster of data. You can cut the universe in different classes using the hyperplane which can be molded in any direction. This can be done both linearly and non-linearly. The identified hyperplane can be thought as a decision boundary between the two clusters. This allows classification of vectors multi dimensionally. This can be used with text classification by encoding on text data. This results in every item in the dataset being represented as a vector with large value dimensions, everyone representing the frequency one of the words of the text.

Advantages: - High accuracy with small data, - Not susceptible to overfitting - Works with linear and non-linear data.

Disadvantage: - Memory-intensive operationally - Hard to understand and implement certain.

Selected Model: We selected the Kernel SVM Model due to high accuracy and sensitivity as well as its charateristic of being accurate with small datasets of which this one.

7. Deployment

Shiny App Url: https://paul-doucet.shinyapps.io/Group8Assignment1/ (https://paul-doucet.shinyapps.io/Group8Assignment1/)

Summary Explanation

- Limitations of our analysis:
    - Our model is trained on a relatively small dataset provided by a single institution. Its ability to predict dianoses could be biased depending on the demographic population that make use of this institution's facilities. This demographic data was not provided with the dataset.
    - Our analysis considered 4 Machine learning algorithms or the many that are available.
- Further steps we could take:
    - A more comprehensive study could be completed is similiar data could be obtain from other diverse sources around the world.
    - Given time and resources, additional algorithms and optimization techniques could be explored to improve the performance.
- Explanation of Model:
    - Our model was arrived at by analysing the 30 columns of data included with the dataset. The different features included as columns were derived from measurements taken on tumors of actual patients who underwent a radiologic study. The features included size measurements such as radius, area, perimeter, etc. as well as shape and density measurements. The model analysed the features to determine which ones were strongly corelated to the outcome of the dianosis.
    - We identified 5 of the 31 columns as the most important determining factors.
    - The Factors that contributed to malignant vs benign tumor identification were the following columns:
        - concavity_mean: mean of severity of concave portions of the contour
        - concave points_mean: mean for number of concave portions of the contour
        - radius_worst: "worst" or largest mean value for mean of distances from center to points on the perimeter
        - concavity_points_worst: "worst" or largest mean value for number of concave portions of the contour
        - perimeter_worst: (Discription not given)
    - Using these 5 columns as our feature metrics we investigated 4 different machine learning algorihtms to determine which one provides the best solution for predicting a diagnostic outcome outcome, i.e. Whether a tumor is Benign or Malignant.

    - The model chosen was able to predict with an accucuracy of 97.9 %.
- Ethical Framework Questions:
    - Can a malicious actor infer information about individuals from your system? No. There is no PII present.
    - Are you able to identify anomalous activity on your system that might indicate a security breach? This would need to be considered for each specific deployment.
    - Do you have a plan to monitor for poor performance on individuals or subgroups? N/A since No demographic data is present.
    - Do you have a plan to log and store historical predictions if a consumer requests access in the future? N/A since No demographic data is present.
    - Have you documented model retraining cycles and can you confirm that a subject's data has been removed from models? N/A since No demographic data is present.

References

Yihui Xie, J. J. Allaire, Garrett Grolemund, 2019, R Markdown: The Definitive Guide https://bookdown.org/yihui/rmarkdown/markdown-syntax.html (https://bookdown.org/yihui/rmarkdown/markdown-syntax.html)

Jonathan McPherson, 2016, R Notebooks https://blog.rstudio.com/2016/10/05/r-notebooks (https://blog.rstudio.com/2016/10/05/r-notebooks)

Adam Kardash, Patricia Kosseim, 2018, Responsible AI in Consumer Enterprise, integrate.ai

Mercedes Ovejero Bruna, 2019, https://www.kaggle.com/mercheovejero/breast-cancer-analysis-real-machine-learning (https://www.kaggle.com/mercheovejero/breast-cancer-analysis-real-machine-learning)

J Marcus W. Beck, 2018, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6262849/ (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6262849/)