# Intro to Data Science - Final Project

Ryan Maciej

---

## Background

A group of physicians have become interested in using machine learning techniques to assist in identifying malignant tumors. They have asked your group to **demonstrate machine learning techniques**.

## The Data

The physicians have identified a data set that consists of over 500 measurements from Fine Needle Aspiration (FNA) of breast tissue masses. In an FNA, a small needle is used to extract a sample of cells from a tissue mass. The cells are then photographed under a microscope. The resulting photographs are entered into graphical imaging software. A trained technician uses a mouse pointer to draw the boundary of the nuclei. The software then calculates each of ten characteristics for the nuclei.

## The Task

You have been asked by the physicians to conduct an analysis of the data using three of the classification methods we have seen in this class, and provide a video presentation that describes those results.

- Perform basic **Exploratory Data Analysis**
- Split the data into test and training data
- Build a classification algorithm using **Decision Trees** (prune your tree appropriately)
- Build a classification algorithm using **Random Forests** / bagging (adjust the parameters of the forest appropriately)
- Build a classification algorithm using **KNN** (Kth Nearest Neighbors - tune the value of K appropriately).

---

## Set Up

```
knitr::opts_chunk$set(tidy.opts = list(width.cutoff = 80), tidy = TRUE, out.width = "70%",
    warning = F)

# install.packages(c('tidyverse','knitr','readr','caret','randomForest','rpart','class','rattle','rpart

library(formatR)
library(tidyverse)
```

```
library(readr)
library(caret)
library(randomForest)
library(rpart)
library(class)
library(rattle)
library(rpart.plot)
library(parallel)
library(ROCR)
library(randomForest)
```

---

## Read In & Organize Data

```
FNA_cancer <- read_csv("Notre Dame/Classes/Intro to Data Science/FNA_cancer.csv")
```

```
## New names:
## Rows: 568 Columns: 33
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): diagnosis dbl (31): id, radius_mean, texture_mean, perimeter_mean,
## area_mean, smoothne... lgl (1): ...33
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...33`
```

```
glimpse(FNA_cancer)
```

```
## Rows: 568
## Columns: 33
## $ id                      <dbl> 842302, 842517, 84300903, 84348301, 84358402, ~
## $ diagnosis               <chr> "M", "M", "M", "M", "M", "M", "M", "M", "M", "~
## $ radius_mean             <dbl> 17.990, 20.570, 19.690, 11.420, 20.290, 12.450~
## $ texture_mean            <dbl> 10.38, 17.77, 21.25, 20.38, 14.34, 15.70, 19.9~
## $ perimeter_mean          <dbl> 122.80, 132.90, 130.00, 77.58, 135.10, 82.57, ~
## $ area_mean               <dbl> 1001.0, 1326.0, 1203.0, 386.1, 1297.0, 477.1, ~
## $ smoothness_mean         <dbl> 0.11840, 0.08474, 0.10960, 0.14250, 0.10030, 0~
## $ compactness_mean        <dbl> 0.27760, 0.07864, 0.15990, 0.28390, 0.13280, 0~
## $ concavity_mean          <dbl> 0.30010, 0.08690, 0.19740, 0.24140, 0.19800, 0~
## $ `concave points_mean`   <dbl> 0.14710, 0.07017, 0.12790, 0.10520, 0.10430, 0~
## $ symmetry_mean           <dbl> 0.2419, 0.1812, 0.2069, 0.2597, 0.1809, 0.2087~
## $ fractal_dimension_mean  <dbl> 0.07871, 0.05667, 0.05999, 0.09744, 0.05883, 0~
## $ radius_se               <dbl> 1.0950, 0.5435, 0.7456, 0.4956, 0.7572, 0.3345~
## $ texture_se              <dbl> 0.9053, 0.7339, 0.7869, 1.1560, 0.7813, 0.8902~
## $ perimeter_se            <dbl> 8.589, 3.398, 4.585, 3.445, 5.438, 2.217, 3.18~
## $ area_se                 <dbl> 153.40, 74.08, 94.03, 27.23, 94.44, 27.19, 53.~
## $ smoothness_se           <dbl> 0.006399, 0.005225, 0.006150, 0.009110, 0.0114~
## $ compactness_se          <dbl> 0.049040, 0.013080, 0.040060, 0.074580, 0.0246~
## $ concavity_se            <dbl> 0.05373, 0.01860, 0.03832, 0.05661, 0.05688, 0~
```

```
## $ `concave points_se`     <dbl> 0.015870, 0.013400, 0.020580, 0.018670, 0.0188~
## $ symmetry_se             <dbl> 0.03003, 0.01389, 0.02250, 0.05963, 0.01756, 0~
## $ fractal_dimension_se    <dbl> 0.006193, 0.003532, 0.004571, 0.009208, 0.0051~
## $ radius_worst            <dbl> 25.38, 24.99, 23.57, 14.91, 22.54, 15.47, 22.8~
## $ texture_worst           <dbl> 17.33, 23.41, 25.53, 26.50, 16.67, 23.75, 27.6~
## $ perimeter_worst         <dbl> 184.60, 158.80, 152.50, 98.87, 152.20, 103.40,~
## $ area_worst              <dbl> 2019.0, 1956.0, 1709.0, 567.7, 1575.0, 741.6, ~
## $ smoothness_worst        <dbl> 0.1622, 0.1238, 0.1444, 0.2098, 0.1374, 0.1791~
## $ compactness_worst       <dbl> 0.6656, 0.1866, 0.4245, 0.8663, 0.2050, 0.5249~
## $ concavity_worst         <dbl> 0.71190, 0.24160, 0.45040, 0.68690, 0.40000, 0~
## $ `concave points_worst`  <dbl> 0.26540, 0.18600, 0.24300, 0.25750, 0.16250, 0~
## $ symmetry_worst          <dbl> 0.4601, 0.2750, 0.3613, 0.6638, 0.2364, 0.3985~
## $ fractal_dimension_worst <dbl> 0.11890, 0.08902, 0.08758, 0.17300, 0.07678, 0~
## $ ...33                   <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
```

```
# Rename Variables with Spaces
FNA_cancer <- FNA_cancer %>%
    rename(concave_points_mean = `concave points_mean`, concave_points_se = `concave points_se`,
        concave_points_worst = `concave points_worst`)

# Remove '...33' Column from DataFrame - Contains only NAs
FNA_cancer <- FNA_cancer[-33]

# Check for NA's in any other column (there are none)
FNA_cancer %>%
    summarise_all(~sum(is.na(.)))
```

```
## # A tibble: 1 x 32
##      id diagnosis radius_mean texture_mean perimeter_mean area_mean
##   <int>     <int>       <int>        <int>          <int>     <int>
## 1     0         0           0            0              0         0
## # ... with 26 more variables: smoothness_mean <int>, compactness_mean <int>,
## #   concavity_mean <int>, concave_points_mean <int>, symmetry_mean <int>,
## #   fractal_dimension_mean <int>, radius_se <int>, texture_se <int>,
## #   perimeter_se <int>, area_se <int>, smoothness_se <int>,
## #   compactness_se <int>, concavity_se <int>, concave_points_se <int>,
## #   symmetry_se <int>, fractal_dimension_se <int>, radius_worst <int>,
## #   texture_worst <int>, perimeter_worst <int>, area_worst <int>, ...
```

```
# View table of response variable (diagnosis) and its factors(benign(B) &
# malignant(M))
table(FNA_cancer$diagnosis)
```

```
##
##   B   M
## 356 212
```

```
# Duplicate Response Variable into Binary Numeric
FNA_cancer$diagnosis01 <- ifelse(FNA_cancer$diagnosis == "M", 1, 0)

# Move Diagnosis01 variable to 3rd Column so all predictor variables are at the
# end of DF
FNA_cancer <- FNA_cancer[, c(1, 2, 33, 3:32)]
```

### Rescale Data

```
# Create Rescale Function
rescale_x <- function(x) {
    (x - min(x))/(max(x) - min(x))
}

# Rescale on [0,1]
FNA_cancer1 <- data.frame(lapply(FNA_cancer[3:33], rescale_x))
FNA_cancer1 <- cbind(FNA_cancer[1:2], FNA_cancer1)

# Relevel
FNA_cancer1$diagnosis01 <- relevel(as.factor(FNA_cancer1$diagnosis01), ref = 1)
FNA_cancer1$diagnosis <- relevel(as.factor(FNA_cancer1$diagnosis), ref = "M")
```

---

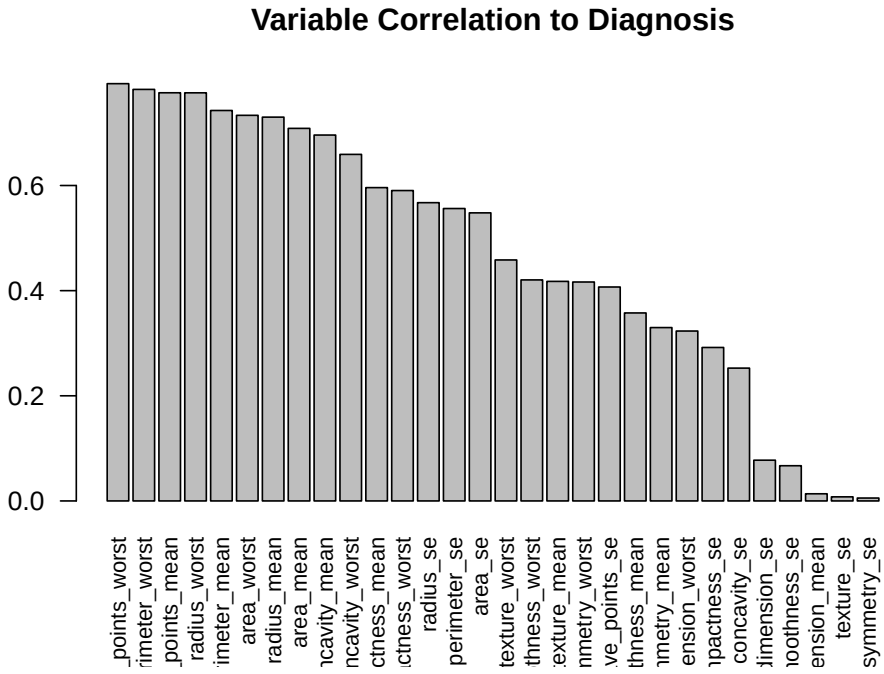
## EDA

```
# Correlation between all variables and Diagnosis01
FNAcor <- cor(FNA_cancer[-c(1, 2)])
FNAcor <- data.frame(correlation = abs(FNAcor[1, ]))
FNAcor <- FNAcor %>%
    arrange(by = -correlation)
FNAcor <- FNAcor %>%
    mutate(name = row.names(FNAcor))
FNAcor <- FNAcor[2:31, ]
FNAcor <- as.data.frame(FNAcor)
FNAcorRank <- FNAcor %>%
    mutate(Correlation.Rank = c(1:30)) %>%
    rename(variable_name = name)
FNAcorRank
```

```
##                         correlation          variable_name Correlation.Rank
## concave_points_worst    0.793743025    concave_points_worst               1
## perimeter_worst         0.782794681         perimeter_worst               2
## concave_points_mean     0.776396257     concave_points_mean               3
## radius_worst            0.776306336            radius_worst               4
## perimeter_mean          0.742716605          perimeter_mean               5
## area_worst              0.733496292              area_worst               6
## radius_mean             0.730060409             radius_mean               7
## area_mean               0.708659334               area_mean               8
## concavity_mean          0.695972573          concavity_mean               9
## concavity_worst         0.659172930         concavity_worst              10
## compactness_mean        0.595978844        compactness_mean              11
## compactness_worst       0.590426637       compactness_worst              12
## radius_se               0.567337589                radius_se              13
## perimeter_se            0.556230066             perimeter_se              14
## area_se                 0.547994823                  area_se              15
```

```
## texture_worst           0.458414893          texture_worst              16
## smoothness_worst        0.420442920       smoothness_worst              17
## texture_mean            0.417610112           texture_mean              18
## symmetry_worst          0.416447603         symmetry_worst              19
## concave_points_se       0.406972235      concave_points_se              20
## smoothness_mean         0.357584887        smoothness_mean              21
## symmetry_mean           0.329753554          symmetry_mean              22
## fractal_dimension_worst 0.323182507 fractal_dimension_worst              23
## compactness_se          0.291921244         compactness_se              24
## concavity_se            0.252675902           concavity_se              25
## fractal_dimension_se    0.077503649   fractal_dimension_se              26
## smoothness_se           0.066984236         smoothness_se               27
## fractal_dimension_mean  0.013609644 fractal_dimension_mean              28
## texture_se              0.007788668             texture_se              29
## symmetry_se             0.005506013            symmetry_se              30
```

```
# graph variables based on correlation
barplot(FNAcor$correlation, main = "Variable Correlation to Diagnosis", names.arg = FNAcor$name,
    cex.names = 0.8, las = 2)
```

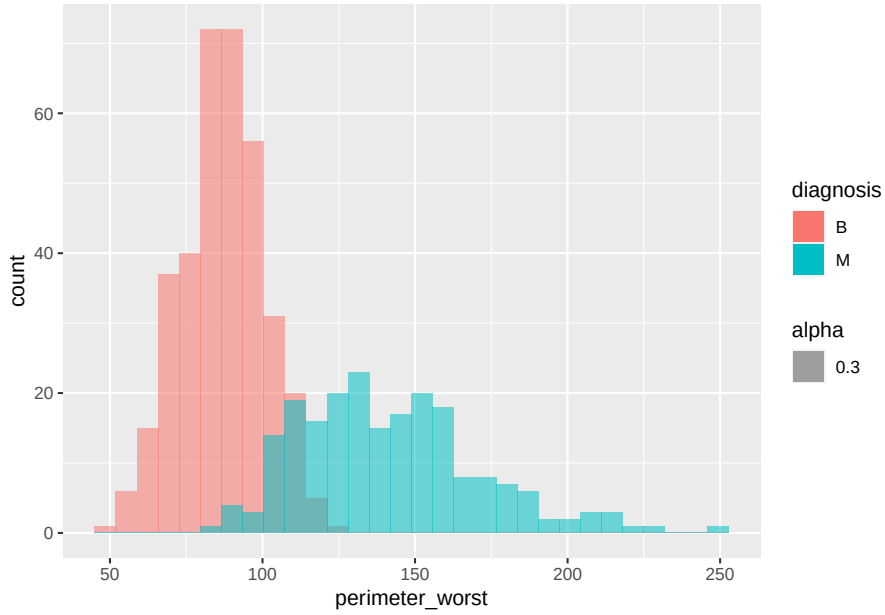



The variables with the 3 highest correlations are *concave points worst*, *perimeter worst*, and *concave points mean*

```
# Separate Benign and Malignant diagnosis into two dataframes.
malignant <- FNA_cancer1 %>%
    filter(diagnosis == "M") %>%
    dplyr::select(-c(1:3))
benign <- FNA_cancer1 %>%
    filter(diagnosis == "B") %>%
    dplyr::select(-c(1:3))

# Histogram of Perimeter Worst
```

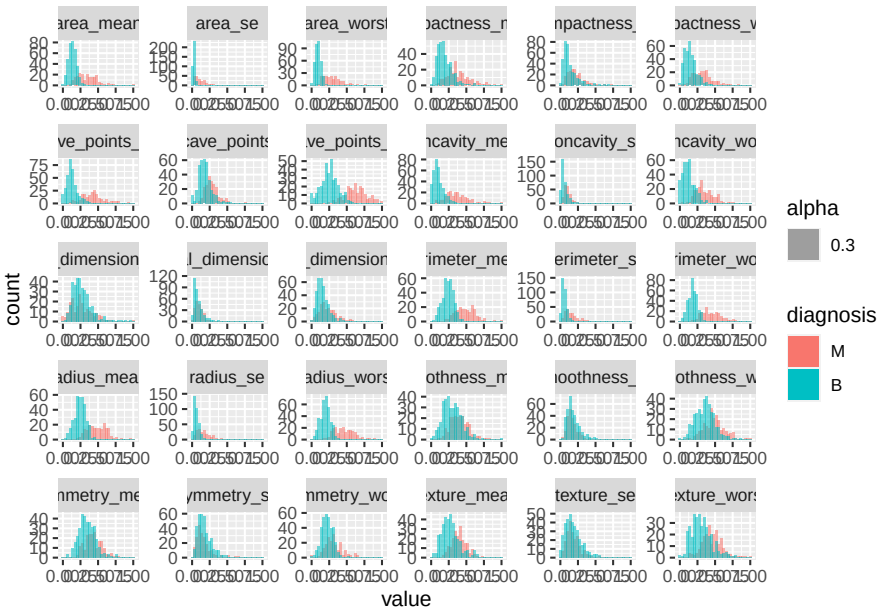
```
FNA_cancer %>%
    ggplot(aes(x = perimeter_worst, fill = diagnosis, alpha = 0.3)) + geom_histogram(position = "identi
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Histogram of All Variables to Visualize Which have a difference in
# distribution between Malignant and Benign
FNA_cancer_histograms <- FNA_cancer1 %>%
    dplyr::select(c(2, 4:33)) %>%
    pivot_longer(c(2:31))
FNA_cancer_histograms %>%
    ggplot(aes(x = value, fill = diagnosis, alpha = 0.3)) + geom_histogram(position = "identity") +
    facet_wrap(~name, scales = "free")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




The variables with the most distinct groups might be the most useful in our modeling.

```
# Calculate Means and SD of each Variable for Malig & Benign Groups
malignant_means <- data.frame(lapply(malignant, mean)) %>%
    pivot_longer(cols = c(1:30)) %>%
    rename(Malignant.Mean = value)
malignant_sd <- data.frame(apply(malignant, 2, sd)) %>%
    rename(Malignant.SD = apply.malignant..2..sd.)
malignant_sd <- malignant_sd %>%
    mutate(name = row.names(malignant_sd))

benign_means <- data.frame(lapply(benign, mean)) %>%
    pivot_longer(cols = c(1:30)) %>%
    rename(Benign.Mean = value)
benign_sd <- data.frame(apply(benign, 2, sd)) %>%
    rename(Benign.SD = apply.benign..2..sd.)
benign_sd <- benign_sd %>%
    mutate(name = row.names(benign_sd))

# Join into single DataFrame & add Difference of Means, and Number of Obs for
# Each Group
variable.stats <- left_join(malignant_means, malignant_sd, by = "name") %>%
    left_join(., benign_means, by = "name") %>%
    left_join(., benign_sd, by = "name") %>%
    mutate(Mean.Diff = abs(Benign.Mean - Malignant.Mean)) %>%
    mutate(m.N = nrow(malignant), b.N = nrow(benign))

# Calculate Pooled Variance
variable.stats2 <- variable.stats %>%
    mutate(pooled.var = ((m.N - 1) * Malignant.SD + (b.N - 1) * Benign.SD)/(m.N +
        b.N - 2))

# Calculate Confidence Intervals of Diff of Sample Means to see if the groups
# differ. (Based on 99% Confidence Interval)
t.stat <- qt(0.975, 566)
```

```
variable.stats3 <- variable.stats2 %>%
    mutate(CI.L = (Malignant.Mean - Benign.Mean) - t.stat * sqrt((Malignant.SD/m.N) +
        (Benign.SD/b.N)), CI.U = (Malignant.Mean - Benign.Mean) + t.stat * sqrt((Malignant.SD/m.N) +
        (Benign.SD/b.N)))

# Variables that might be Removed, no Statistical Difference Between Means
variables.no.diff <- variable.stats3 %>%
    mutate(IsDiff = ifelse(CI.U > 0 & CI.L < 0, 0, 1)) %>%
    filter(IsDiff == 0) %>%
    dplyr::select(name)
variables.no.diff
```

```
## # A tibble: 6 x 1
##   name
##   <chr>
## 1 fractal_dimension_mean
## 2 texture_se
## 3 smoothness_se
## 4 concavity_se
## 5 symmetry_se
## 6 fractal_dimension_se
```

---

## KNN Model

### KNN - Split Into Test/ Training Groups.

```
set.seed(1981)

n = nrow(FNA_cancer1)
test_index <- sample.int(n, size = round(0.2 * n))
train_FNA_cancer <- FNA_cancer1[-test_index, ]
test_FNA_cancer <- FNA_cancer1[test_index, ]
glimpse(train_FNA_cancer)
```

```
## Rows: 454
## Columns: 33
## $ id                    <dbl> 842302, 842517, 84300903, 84348301, 84358402, ~
## $ diagnosis             <fct> M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, B~
## $ diagnosis01           <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0~
## $ radius_mean           <dbl> 0.5210374, 0.6431445, 0.6014956, 0.2100904, 0.~
## $ texture_mean          <dbl> 0.0226581, 0.2725736, 0.3902604, 0.3608387, 0.~
## $ perimeter_mean        <dbl> 0.5459885, 0.6157833, 0.5957432, 0.2335015, 0.~
## $ area_mean             <dbl> 0.3637328, 0.5015907, 0.4494168, 0.1029056, 0.~
## $ smoothness_mean       <dbl> 0.5539697, 0.2203390, 0.4667460, 0.7928437, 0.~
## $ compactness_mean      <dbl> 0.7920373, 0.1817680, 0.4310165, 0.8113613, 0.~
## $ concavity_mean        <dbl> 0.70313964, 0.20360825, 0.46251172, 0.56560450~
## $ concave_points_mean   <dbl> 0.7311133, 0.3487575, 0.6356859, 0.5228628, 0.~
```

```
## $ symmetry_mean           <dbl> 0.6863636, 0.3797980, 0.5095960, 0.7762626, 0.~
## $ fractal_dimension_mean  <dbl> 0.60551811, 0.14132266, 0.21124684, 1.00000000~
## $ radius_se               <dbl> 0.35614702, 0.15643672, 0.22962158, 0.13909107~
## $ texture_se              <dbl> 0.12046941, 0.08258929, 0.09430251, 0.17587518~
## $ perimeter_se            <dbl> 0.36903360, 0.12444047, 0.18037035, 0.12665504~
## $ area_se                 <dbl> 0.27381126, 0.12565979, 0.16292179, 0.03815479~
## $ smoothness_se           <dbl> 0.15929565, 0.11938675, 0.15083115, 0.25145324~
## $ compactness_se          <dbl> 0.35139844, 0.08132304, 0.28395470, 0.54321507~
## $ concavity_se            <dbl> 0.13568182, 0.04696970, 0.09676768, 0.14295455~
## $ concave_points_se       <dbl> 0.3006251, 0.2538360, 0.3898466, 0.3536655, 0.~
## $ symmetry_se             <dbl> 0.31164518, 0.08453875, 0.20569032, 0.72814769~
## $ fractal_dimension_se    <dbl> 0.18304244, 0.09111010, 0.12700551, 0.28720479~
## $ radius_worst            <dbl> 0.6207755, 0.6069015, 0.5563856, 0.2483102, 0.~
## $ texture_worst           <dbl> 0.1415245, 0.3035714, 0.3600746, 0.3859275, 0.~
## $ perimeter_worst         <dbl> 0.6683102, 0.5398177, 0.5084417, 0.2413467, 0.~
## $ area_worst              <dbl> 0.45069799, 0.43521431, 0.37450845, 0.09400806~
## $ smoothness_worst        <dbl> 0.6011358, 0.3475533, 0.4835898, 0.9154725, 0.~
## $ compactness_worst       <dbl> 0.6192916, 0.1545634, 0.3853751, 0.8140117, 0.~
## $ concavity_worst         <dbl> 0.5686102, 0.1929712, 0.3597444, 0.5486422, 0.~
## $ concave_points_worst    <dbl> 0.9120275, 0.6391753, 0.8350515, 0.8848797, 0.~
## $ symmetry_worst          <dbl> 0.5984624, 0.2335896, 0.4037059, 1.0000000, 0.~
## $ fractal_dimension_worst <dbl> 0.41886396, 0.22287813, 0.21343303, 0.77371114~
```

**KNN - Iterate K Values.**

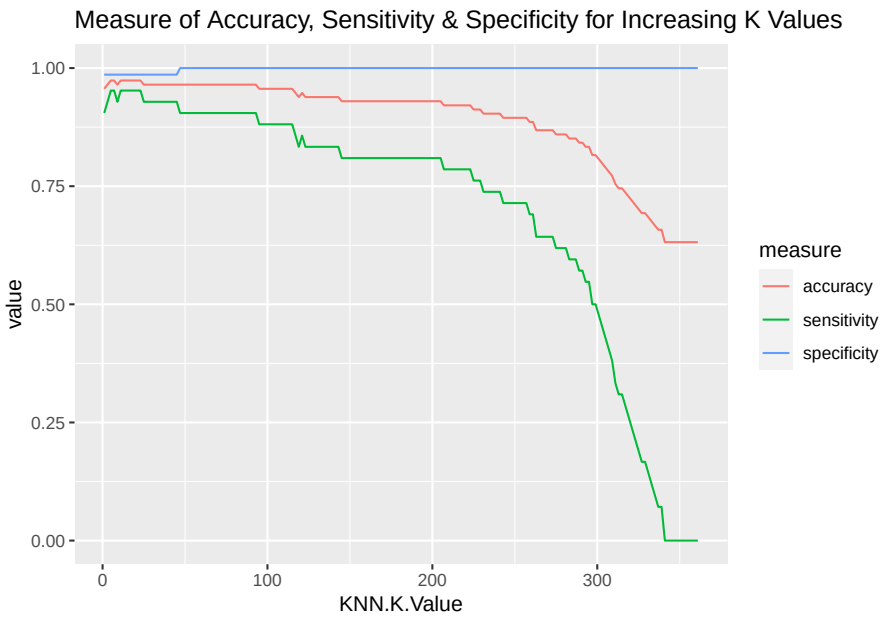
```
# Create vector with many values to test K
knn.values <- seq(1, 361, by = 2)

# Create vectors to store measurements for each K value in iteration
mod <- c(rep(NA, length(knn.values)))
acc <- c(rep(NA, length(knn.values)))
sen <- c(rep(NA, length(knn.values)))
spec <- c(rep(NA, length(knn.values)))

# iterate over all k values
for (i in 1:length(knn.values)) {
    n <- knn.values[i]
    knn.pred <- knn(train_FNA_cancer[-c(1:3)], test = test_FNA_cancer[-c(1:3)], cl = train_FNA_cancer$di
        k = n)
    FNA.knn.table <- table(knn.pred, test_FNA_cancer$diagnosis01, dnn = c("Prediction",
        "Diagnosis"))
    mod[i] <- n
    acc[i] <- sum(diag(FNA.knn.table))/sum(FNA.knn.table)
    sen[i] <- sensitivity(FNA.knn.table, positive = "1")
    spec[i] <- specificity(FNA.knn.table, negative = "0")
}

# Create table to look at measures
df <- data.frame(cbind(`KNN K Value` = mod, accuracy = round(acc, 4), sensitivity = round(sen,
    4), specificity = round(spec, 4)))
head(df, 20)
```

```
##    KNN.K.Value accuracy sensitivity specificity
```

```
## 1             1   0.9561      0.9048      0.9861
## 2             3   0.9649      0.9286      0.9861
## 3             5   0.9737      0.9524      0.9861
## 4             7   0.9737      0.9524      0.9861
## 5             9   0.9649      0.9286      0.9861
## 6            11   0.9737      0.9524      0.9861
## 7            13   0.9737      0.9524      0.9861
## 8            15   0.9737      0.9524      0.9861
## 9            17   0.9737      0.9524      0.9861
## 10           19   0.9737      0.9524      0.9861
## 11           21   0.9737      0.9524      0.9861
## 12           23   0.9737      0.9524      0.9861
## 13           25   0.9649      0.9286      0.9861
## 14           27   0.9649      0.9286      0.9861
## 15           29   0.9649      0.9286      0.9861
## 16           31   0.9649      0.9286      0.9861
## 17           33   0.9649      0.9286      0.9861
## 18           35   0.9649      0.9286      0.9861
## 19           37   0.9649      0.9286      0.9861
## 20           39   0.9649      0.9286      0.9861
```

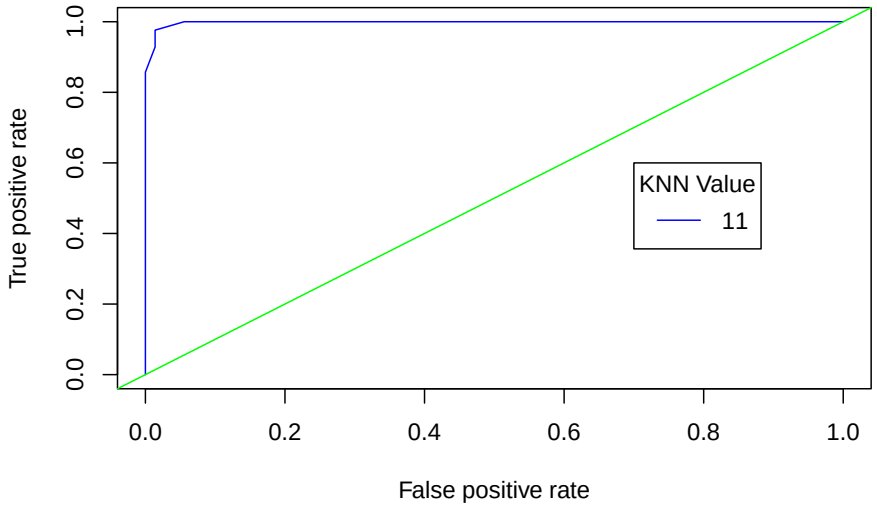
**KNN - Plot various measure over K values.**

```
df2 <- pivot_longer(df, cols = c(accuracy, sensitivity, specificity))
df2 <- df2 %>%
    rename(measure = name)
df2 %>%
    ggplot(aes(x = KNN.K.Value, y = value, color = measure)) + geom_line() + ggtitle("Measure of Accura
```




**KNN - ROC Curve**

```
# Predictions to create ROC Curve
knn.pred <- knn(train_FNA_cancer[-c(1:3)], test = test_FNA_cancer[-c(1:3)], cl = train_FNA_cancer$diagn
    k = 11, prob = T)
prob1 <- attr(knn.pred, "prob")
prob2 <- ifelse(knn.pred == "0", 1 - prob1, prob1)
knn.pred.pred <- ROCR::prediction(prob2, test_FNA_cancer$diagnosis01)
knn.pred11.perf <- ROCR::performance(knn.pred.pred, "tpr", "fpr")

plot(knn.pred11.perf, col = "blue")
abline(a = 0, b = 1, col = "green")
legend(x = 0.7, y = 0.6, legend = c("11"), col = c("blue", "red", "orange", "yellow",
    "purple", "gray"), lty = 1, lwd = 1, title = "KNN Value")
```




## Random Forests

### Random Forest - Model Creation

In this section we test various hyper parameters (number of variables and number of trees) for the Random Forest. The goal is to understand what hyper parameters create the best models.

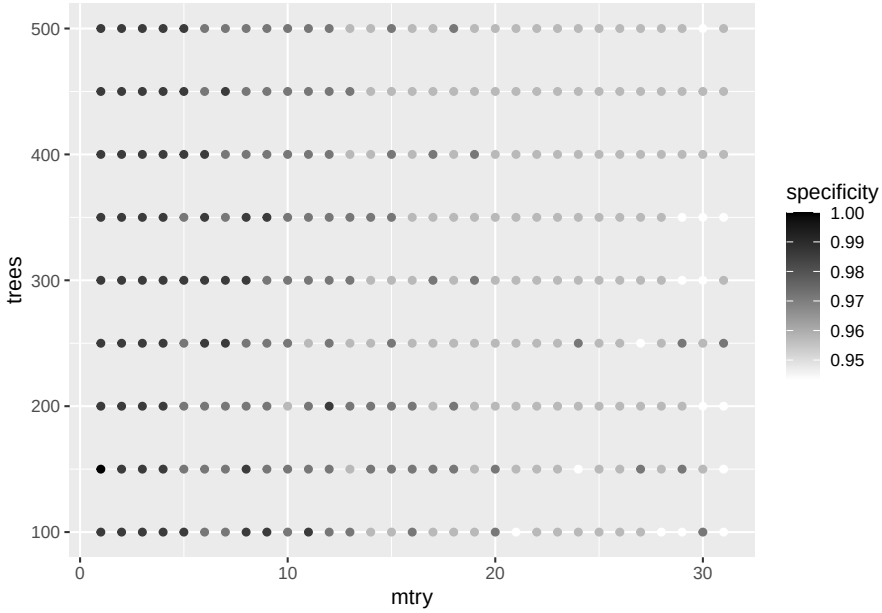
```
set.seed(1982)

# Prep the data for random forest analysis
random_forest_FNA_cancer <- drop_na(dplyr::select(FNA_cancer, -diagnosis01))
random_forest_FNA_cancer$diagnosis <- as.factor(random_forest_FNA_cancer$diagnosis)

# Setup Data Frame to store analysis
num_vars <- dim(random_forest_FNA_cancer)[2] - 1
sensitivity_specificity <- data.frame(matrix(ncol = 5))
colnames(sensitivity_specificity) <- c("mtry", "trees", "sensitivity", "specificity",
    "accuracy")

important_variables <- data.frame(matrix(ncol = 2))
```

```
colnames(important_variables) <- c("gini", "variable_name")

# Split the data frame for test and training
n = nrow(FNA_cancer)
test_index <- sample.int(n, size = round(0.2 * n))
random_forest_train_FNA_cancer <- random_forest_FNA_cancer[-test_index, ]
random_forest_test_FNA_cancer <- random_forest_FNA_cancer[test_index, ]

# Formula with all predictors
random_forest_formula <- as.formula(diagnosis ~ .)

# In this loop we test the number of variables and a range of a number of trees
# The goal is to determine a good value for the number of variables and trees
# to be used for prediction.

# The outer loop is a variable counter, the inner loop is a loop for the number
# of forests
for (i in 1:num_vars) {
    for (k in seq(100, 500, 50)) {

        # build a random forest model
        cancer_forest <- randomForest(random_forest_formula, data = random_forest_train_FNA_cancer,
            mtry = i, ntree = k, na.action = na.roughfix)

        # Run a prediction using the test data
        predicted <- predict(cancer_forest, random_forest_test_FNA_cancer)

        # Build a confusion matrix from the prediction
        tmp_confusion <- confusionMatrix(table(predicted, random_forest_test_FNA_cancer$diagnosis),
            positive = "M")

        # Store the values from the confusion matrix in a temporary dataframe
        tmp_df <- data.frame(mtry = i, trees = k, sensitivity = tmp_confusion$byClass["Sensitivity"],
            specificity = tmp_confusion$byClass["Specificity"], accuracy = tmp_confusion$overall["Accura
        sensitivity_specificity <- rbind(sensitivity_specificity, tmp_df)

        tmp_important_variables <- data.frame(randomForest::importance(cancer_forest))
        tmp_important_variables$variable_name <- row.names(tmp_important_variables)

        # Extract and store the variables and their gini values
        colnames(tmp_important_variables) <- c("gini", "variable_name")
        important_variables <- rbind(important_variables, tmp_important_variables)
    }
}

# Calculate the mean Gini for each variable this will give us the most
# important variables based on the Gini values from all the random forest
# models
top_important_variables <- important_variables %>%
    group_by(variable_name) %>%
    summarise(gini_mean = mean(gini)) %>%
    arrange(-gini_mean)
```

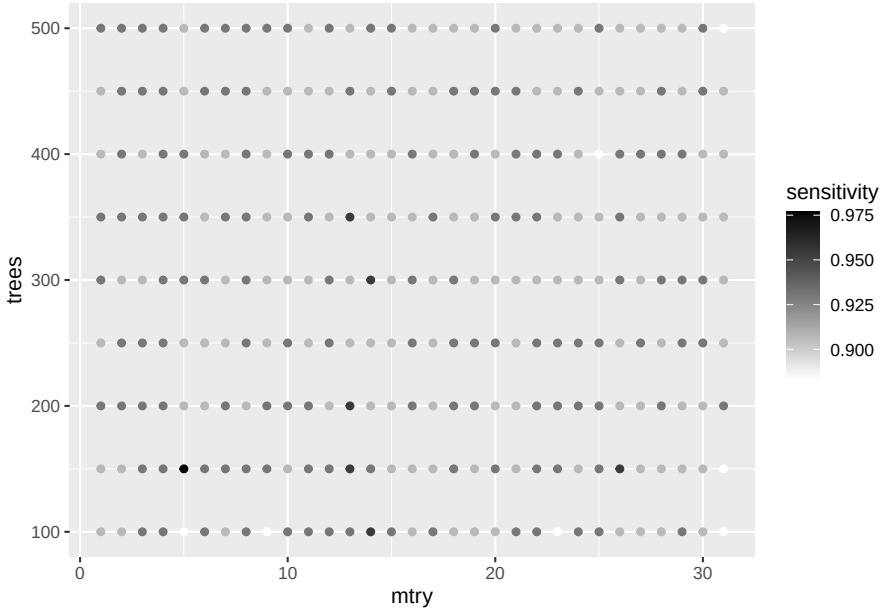
### Random Forest - Plot Hyper Variables

Here we plot sensitivity, specificity and accuracy. the goal is to get a better understanding of the relationships between the hyper parameters and sensitivity, specificity and accuracy.

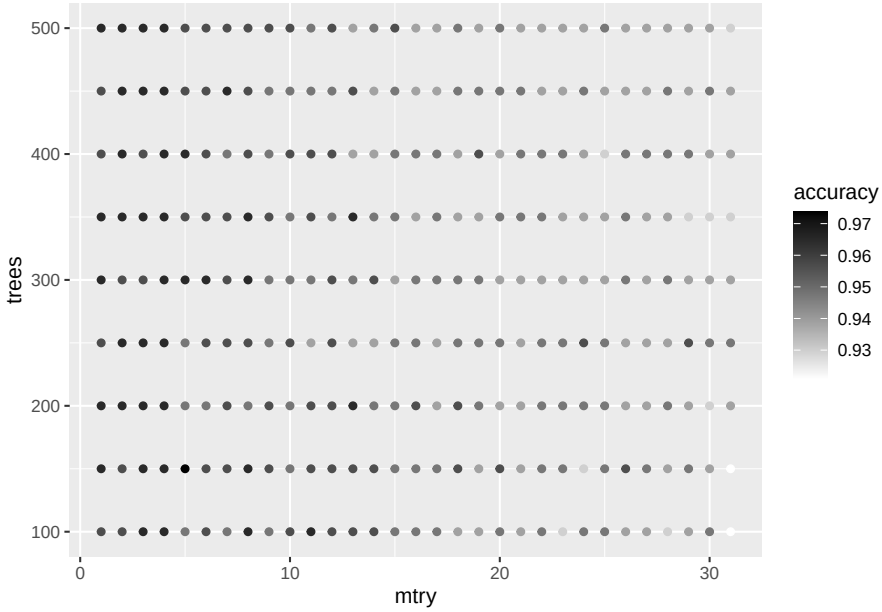
```
sensitivity_specificity <- drop_na(sensitivity_specificity)
sensitivity_specificity %>%
    ggplot(aes(x = mtry, y = trees, color = specificity)) + geom_point() + scale_color_gradient(low = "
    high = "black")
```




```
sensitivity_specificity %>%
    ggplot(aes(x = mtry, y = trees, color = sensitivity)) + geom_point() + scale_color_gradient(low = "
    high = "black")
```



```
sensitivity_specificity %>%
    ggplot(aes(x = mtry, y = trees, color = accuracy)) + geom_point() + scale_color_gradient(low = "whi
    high = "black")
```




**Random Forest - Model Selection**

Based on the graphs lower values of *mtry* and *trees* provide better results for sensitivity,specificity and accuracy. When we sort the results we find:

- A model built with an mtry of 5 and about 150 trees provides a model that is good sensitivity
- A model built with an mtry of 1 and about 100 trees provides a model with good specificity
- A model built with an mtry of 5 and about 150 trees provides a model with good specificity

```
# Order by specificity and sensitivity
specificity_ordered <- sensitivity_specificity %>%
    arrange(-specificity)
sensitivity_ordered <- sensitivity_specificity %>%
    arrange(-sensitivity)
accuracy_ordered <- sensitivity_specificity %>%
    arrange(-accuracy)

# Display the hyper-parameters that provide the best values for each of our
# model metrics
specificity_ordered[1, ]
```

```
##              mtry trees sensitivity specificity  accuracy
## Sensitivity1    1   150   0.9069767           1 0.9649123
```

```
sensitivity_ordered[1, ]
```

```
##               mtry trees sensitivity specificity  accuracy
## Sensitivity37    5   150   0.9767442    0.971831 0.9736842
```

```
accuracy_ordered[1, ]
```

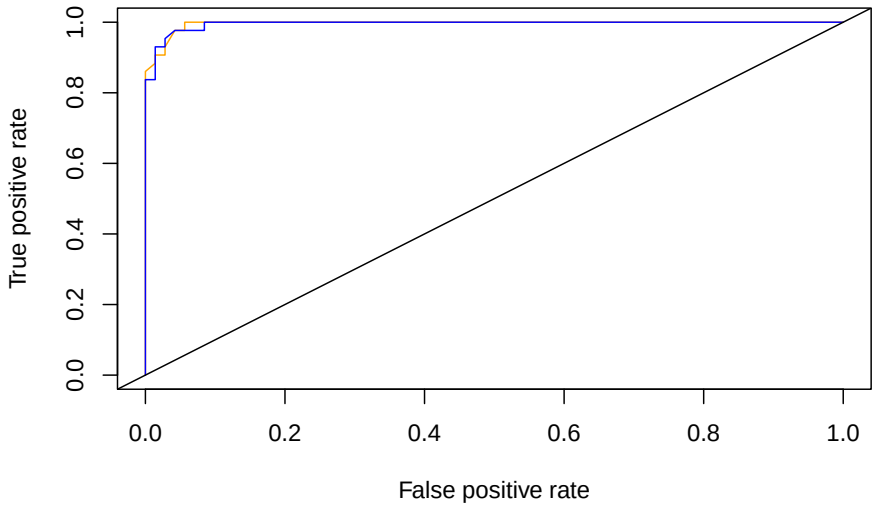
```
##                mtry trees sensitivity specificity  accuracy
## Sensitivity37     5   150   0.9767442    0.971831 0.9736842
```

```
# Here we build two models.  One for minimizing False Negative Rates
random_forest_sensitivity_model <- randomForest(random_forest_formula, data = random_forest_train_FNA_ca
    mtry = 5, ntree = 150, na.action = na.roughfix)
# This model minimizes False Positive Rate
random_forest_specificity_model <- randomForest(random_forest_formula, data = random_forest_train_FNA_ca
    mtry = 1, ntree = 100, na.action = na.roughfix)
```

### Random Forest - Create ROC Curve

In the followoing section we pring ROC Curves for specificity and sensitivity.

```
# Step 1 - create a predict object for the ROC curve
pred_specificity <- predict(random_forest_specificity_model, random_forest_test_FNA_cancer,
    type = "prob")
pred_sensitivity <- predict(random_forest_sensitivity_model, random_forest_test_FNA_cancer,
    type = "prob")

# Step 2 - Create a precition for the ROC curve
roc_pred_specificity <- ROCR::prediction(pred_specificity[, 2], random_forest_test_FNA_cancer$diagnosis
roc_pred_sensitivity <- ROCR::prediction(pred_sensitivity[, 2], random_forest_test_FNA_cancer$diagnosis

# Step 3 - Create a performance variable for the ROC Cuve
roc_perf_specificity <- ROCR::performance(roc_pred_specificity, "tpr", "fpr")
roc_perf_sensitivity <- ROCR::performance(roc_pred_sensitivity, "tpr", "fpr")

# Step 4 - Plot it
plot(roc_perf_specificity, col = "orange")
plot(roc_perf_sensitivity, col = "blue", add = T)
abline(a = 0, b = 1)  # this is the pure chance model
```



## Trees

### Prepare and Split The Data for Tree Models

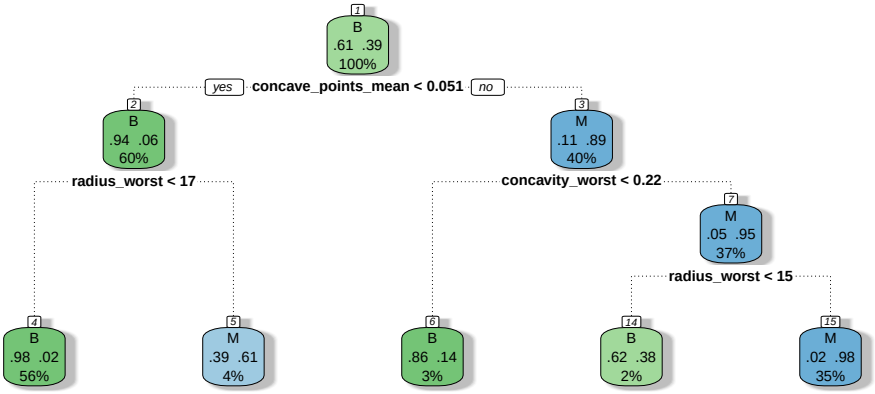
```
n = nrow(FNA_cancer)
tree_FNA_cancer <- drop_na(dplyr::select(FNA_cancer, -diagnosis01))
tree_FNA_cancer$diagnosis <- as.factor(tree_FNA_cancer$diagnosis)
test_index <- sample.int(n, size = round(0.2 * n))
tree_train_FNA_cancer <- tree_FNA_cancer[-test_index, ]
tree_test_FNA_cancer <- tree_FNA_cancer[test_index, ]

# Formula with all predictors
tree_formula <- as.formula(diagnosis ~ .)
```

### Tree Model With All Variables

```
set.seed(1982)
rtree_model <- rpart(tree_formula, data = tree_train_FNA_cancer)
rtree_model
```

```
## n= 454
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 454 178 B (0.60792952 0.39207048)
##    2) concave_points_mean< 0.05128 272  16 B (0.94117647 0.05882353)
##      4) radius_worst< 16.825 254   5 B (0.98031496 0.01968504) *
##      5) radius_worst>=16.825 18   7 M (0.38888889 0.61111111) *
##    3) concave_points_mean>=0.05128 182  20 M (0.10989011 0.89010989)
##      6) concavity_worst< 0.2248 14   2 B (0.85714286 0.14285714) *
##      7) concavity_worst>=0.2248 168   8 M (0.04761905 0.95238095)
##       14) radius_worst< 14.915 8   3 B (0.62500000 0.37500000) *
##       15) radius_worst>=14.915 160   3 M (0.01875000 0.98125000) *
```

```
fancyRpartPlot(rtree_model)
```




Rattle 2022−Jul−02 12:32:10 Ryan

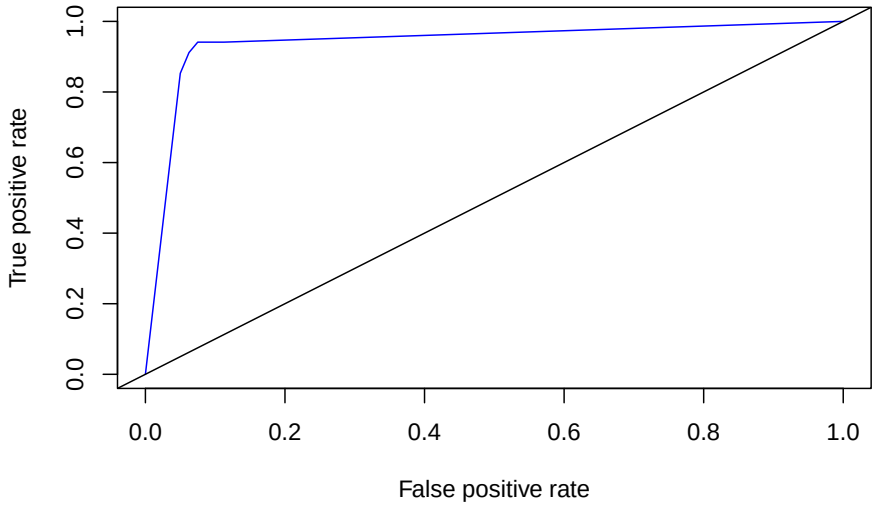
### Confusion Matrix for Simple Tree

```
tree_pred = predict(rtree_model, newdata = tree_test_FNA_cancer, "class")
tree_con_mat1 = confusionMatrix(data = tree_pred, reference = tree_test_FNA_cancer$diagnosis,
    positive = "M")
tree_con_mat1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 75  3
##          M  5 31
##
##                Accuracy : 0.9298
##                  95% CI : (0.8664, 0.9692)
##     No Information Rate : 0.7018
##     P-Value [Acc > NIR] : 2.063e-09
##
##                   Kappa : 0.8351
##
##  Mcnemar's Test P-Value : 0.7237
##
##             Sensitivity : 0.9118
##             Specificity : 0.9375
##          Pos Pred Value : 0.8611
##          Neg Pred Value : 0.9615
##              Prevalence : 0.2982
##          Detection Rate : 0.2719
##    Detection Prevalence : 0.3158
##       Balanced Accuracy : 0.9246
##
##        'Positive' Class : M
##
```

### ROC Curve for Malignant Detection

```
base_p1 = predict(rtree_model, newdata = tree_test_FNA_cancer, "prob")
base_p2 = prediction(base_p1[, 2], tree_test_FNA_cancer$diagnosis)
base_p3 = performance(base_p2, "tpr", "fpr")
plot(base_p3, col = "blue")
abline(a = 0, b = 1)
```



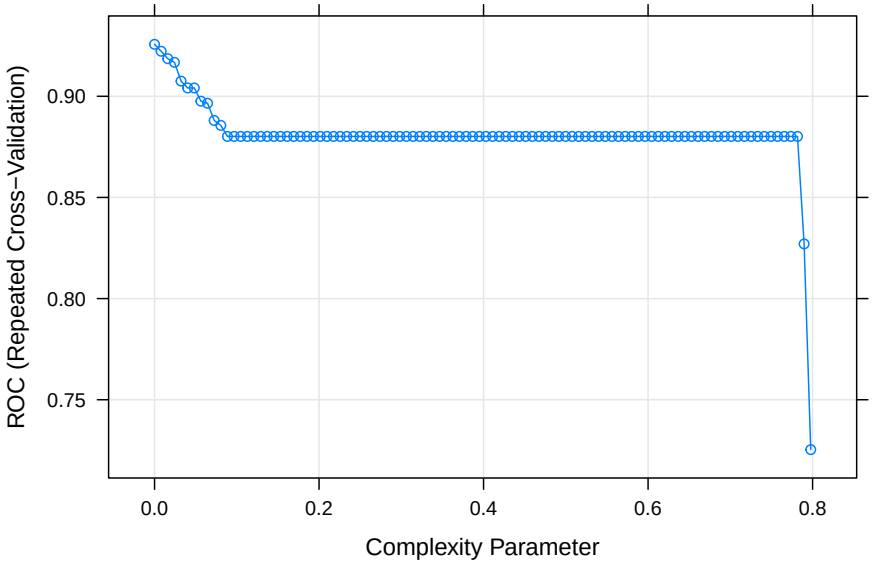

### Multiple Tree Tests

Here we are using `train` to test different hyper parameters for tree analysis.

```
set.seed(1982)
train.control <- trainControl(
                            method = "repeatedcv",
                            number = 5, ## 5-fold CV
                            repeats = 3,## repeated three times
                            # USE AUC
                            summaryFunction = twoClassSummary,
                            classProbs = TRUE
                            )

rpartFit1 <- train(tree_formula, data=tree_train_FNA_cancer,
                   method = "rpart",
                   tuneLength = 100,
                   trControl = train.control,
                   metric = "ROC"
                  )
```

### Repeated Cross Validation against CP

```
plot(rpartFit1)
```

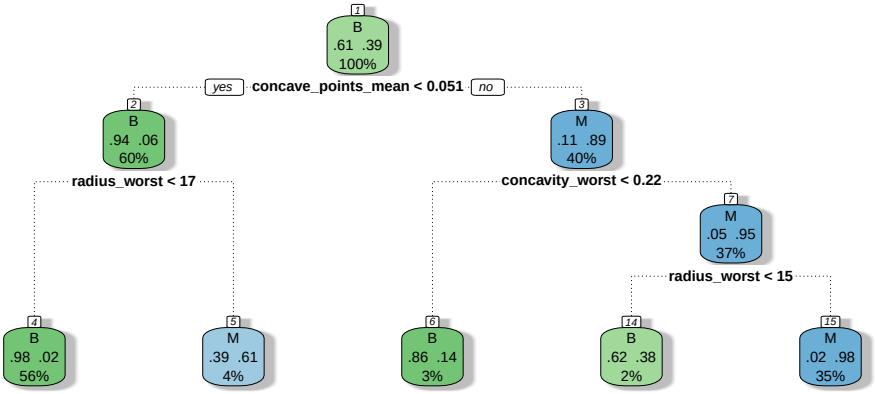



### Plot the Best Model From the Train

```
rpartFit1$finalModel
```

```
## n= 454
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 454 178 B (0.60792952 0.39207048)
##    2) concave_points_mean< 0.05128 272  16 B (0.94117647 0.05882353)
##      4) radius_worst< 16.825 254   5 B (0.98031496 0.01968504) *
##      5) radius_worst>=16.825 18   7 M (0.38888889 0.61111111) *
##    3) concave_points_mean>=0.05128 182  20 M (0.10989011 0.89010989)
##      6) concavity_worst< 0.2248 14   2 B (0.85714286 0.14285714) *
##      7) concavity_worst>=0.2248 168   8 M (0.04761905 0.95238095)
##       14) radius_worst< 14.915 8   3 B (0.62500000 0.37500000) *
##       15) radius_worst>=14.915 160   3 M (0.01875000 0.98125000) *
```

```
fancyRpartPlot(rpartFit1$finalModel)
```




Rattle 2022−Jul−02 12:32:21 Ryan

### Final Prediction Using the Best Model from Train()

```
tree_pred2 = predict(rpartFit1$finalModel, newdata = tree_test_FNA_cancer, "class")
tree_con_mat2 = confusionMatrix(tree_pred2, ref = tree_test_FNA_cancer$diagnosis,
    positive = "M")
tree_con_mat2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 75  3
##          M  5 31
##
##                Accuracy : 0.9298
##                  95% CI : (0.8664, 0.9692)
##     No Information Rate : 0.7018
##     P-Value [Acc > NIR] : 2.063e-09
##
##                   Kappa : 0.8351
##
##  Mcnemar's Test P-Value : 0.7237
##
##             Sensitivity : 0.9118
##             Specificity : 0.9375
##          Pos Pred Value : 0.8611
##          Neg Pred Value : 0.9615
##              Prevalence : 0.2982
##          Detection Rate : 0.2719
##    Detection Prevalence : 0.3158
##       Balanced Accuracy : 0.9246
##
##        'Positive' Class : M
##
```

### Original Confusion Matrix

```
tree_con_mat1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 75  3
##          M  5 31
##
##                Accuracy : 0.9298
##                  95% CI : (0.8664, 0.9692)
##     No Information Rate : 0.7018
##     P-Value [Acc > NIR] : 2.063e-09
##
##                   Kappa : 0.8351
##
##  Mcnemar's Test P-Value : 0.7237
##
##             Sensitivity : 0.9118
##             Specificity : 0.9375
##          Pos Pred Value : 0.8611
##          Neg Pred Value : 0.9615
##              Prevalence : 0.2982
##          Detection Rate : 0.2719
##    Detection Prevalence : 0.3158
##       Balanced Accuracy : 0.9246
##
##        'Positive' Class : M
##
```

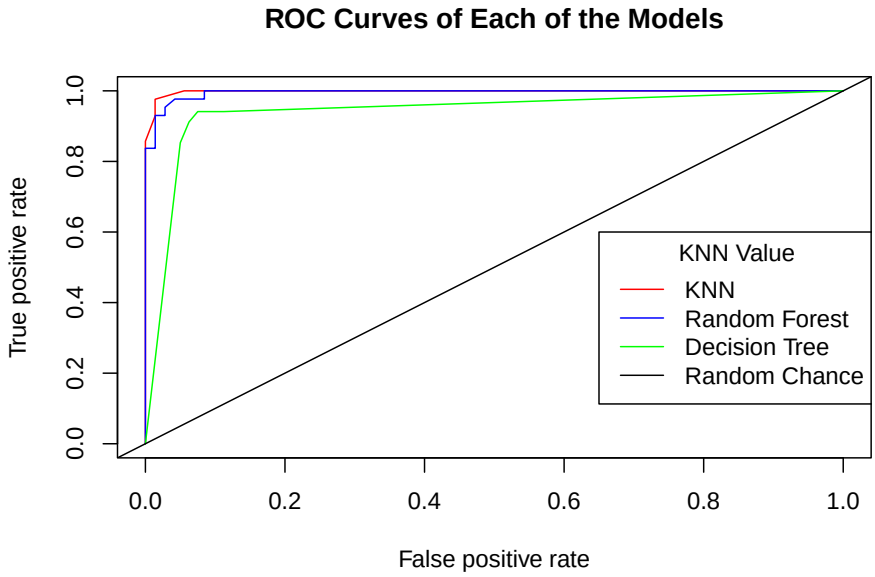
## Summary

Based on the results of the model analysis the best overall model is the *Random Forest*. The best model for accuracy and sensitivity is the *Random Forest*. The best model for specificity is the KNN model.

### ROC_

Below are ROC curves, one from each model. These are the best results from each model.

```
plot(knn.pred11.perf, col = "red")
plot(roc_perf_sensitivity, col = "blue", add = T)
plot(base_p3, col = "green", add = T)
abline(a = 0, b = 1)  # this is the pure chance model
legend(x = 0.65, y = 0.6, legend = c("KNN", "Random Forest", "Decision Tree", "Random Chance"),
    col = c("red", "blue", "green", "black"), lty = 1, lwd = 1, title = "KNN Value")
title(main = "ROC Curves of Each of the Models")
```




### Important Variables

```
RF.Rank <- top_important_variables %>%
    mutate(RF.Rank = c(1:32)) %>%
    drop_na() %>%
    filter(variable_name != "id")
rank1 <- left_join(RF.Rank, FNAcorRank, by = "variable_name") %>%
    dplyr::select(c(1, 3, 5))
ttest.rank <- variables.no.diff %>%
    mutate(TTest.Rank = "Worst 6") %>%
    rename(variable_name = name)
rank2 <- left_join(rank1, ttest.rank, by = "variable_name")
vars_in_tree <- data.frame(variable_name = c("concave_points_mean", "concavity_worst",
    "radius_worst"), Tree.Rank = "1-3")
left_join(rank2, vars_in_tree, by = "variable_name")
```

```
## # A tibble: 30 x 5
##    variable_name        RF.Rank Correlation.Rank TTest.Rank Tree.Rank
##    <chr>                  <int>            <int> <chr>      <chr>
##  1 concave_points_worst       1                1 <NA>       <NA>
##  2 perimeter_worst            2                2 <NA>       <NA>
##  3 concave_points_mean        3                3 <NA>       1-3
##  4 radius_worst               4                4 <NA>       1-3
##  5 area_worst                 5                6 <NA>       <NA>
##  6 concavity_worst            6               10 <NA>       1-3
##  7 texture_worst              7               16 <NA>       <NA>
##  8 concavity_mean             8                9 <NA>       <NA>
##  9 area_se                    9               15 <NA>       <NA>
## 10 area_mean                 10                8 <NA>       <NA>
## # ... with 20 more rows
```