Breast Cancer Detection: Model Comparisons

Corey Moon and Asma Mohammed Asiri

4/18/2022

# Motivation and Overview

Cancer is the second leading cause of death globally and is responsible for an estimated 9.6 million deaths in 2018. Globally, about 1 in 6 deaths are due to cancer. It is the leading cause of death in developed and developing countries, with projected annual deaths rising to 13.1 million by 2030. However, some forms of cancer, like breast cancer, have a higher chance of total remission if they are detected at an early stage and adequately treated.

# Background Work

Model comparison is a hot topic in the machine learning domain, as multiple models are often used on the same dataset to see how they differ in performance. Generally, there isn’t one model that dominates for any given type of data, some there are differences in interpretability and presentability between types of models. Some of the nuances of this topic are discussed [here](https://medium.com/@taniyaghosh29/machine-learning-algorithms-what-are-the-differences-9b71df4f248f) and [here](https://medium.com/@nischitasadananda/the-battle-between-logistic-regression-random-forest-classifier-xg-boost-and-support-vector-46d773c70f41), by authors Taniya and Nischitha Sadananda respectively.

# Data

## Data sorce

The data used for our project sourced from the UC Irvine Machine Learning Repository, created by Dr. Mangasarian, Dr. Wolberg, and Dr. Street, which can be found [here](https://archive-beta.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+diagnostic), and a description of their research can be found [here](https://pages.cs.wisc.edu/~olvi/uwmp/cancer.html#diag).

## Dataset Features

The covariates included are:

* Cell radius - mean distance from cell center to points on the perimeter
* Cell texture - standard deviation of grey-scale image values
* Perimeter
* Area
* Smoothness - Local variation in radius lengths
* Compactness - Defined as perimeter2 / Area - 1
* Concavity - Severity of concave portions of the cell contour
* Symmetry
* Fractal dimension - Defined as the coastline approximation - 1

# Exploratory Data Analysis

The first step was to validate the dataset to ensure there were no missing values and remove any redundant information. The standard error columns were removed from the analysis since estimations would not likely provide any more meaningful insight that couldn’t be found from mean and worst-case columns.

##### Loading in packages

library(tidyverse)  
#install.packages("ggcorrplot")  
library(ggcorrplot)  
library(grid)  
library(gridExtra)

##### Formatting dataset for analysis

#Loading dataset  
breast\_cancer <- readxl::read\_xlsx('Breast Cancer data - CS 5610.xlsx')  
  
#Removing id and standard error columns  
b\_cancer <- breast\_cancer[, -1]  
b\_cancer <- b\_cancer[, !grepl('\_se', colnames(b\_cancer))]  
colnames(b\_cancer)

## [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\_worst"   
## [13] "texture\_worst" "perimeter\_worst"   
## [15] "area\_worst" "smoothness\_worst"   
## [17] "compactness\_worst" "concavity\_worst"   
## [19] "concave points\_worst" "symmetry\_worst"   
## [21] "fractal\_dimension\_worst"

colnames(b\_cancer)[c(9, 19)] <- c("concave\_points\_mean", "concave\_points\_worst")  
  
#converting diagnosis to factor  
b\_cancer$diagnosis <- b\_cancer$diagnosis %>% as.factor()

##### Overview of formatted dataset

dim(b\_cancer)

## [1] 569 21

summary(b\_cancer$diagnosis)

## B M   
## 357 212

any(is.na(b\_cancer)) #No missing data

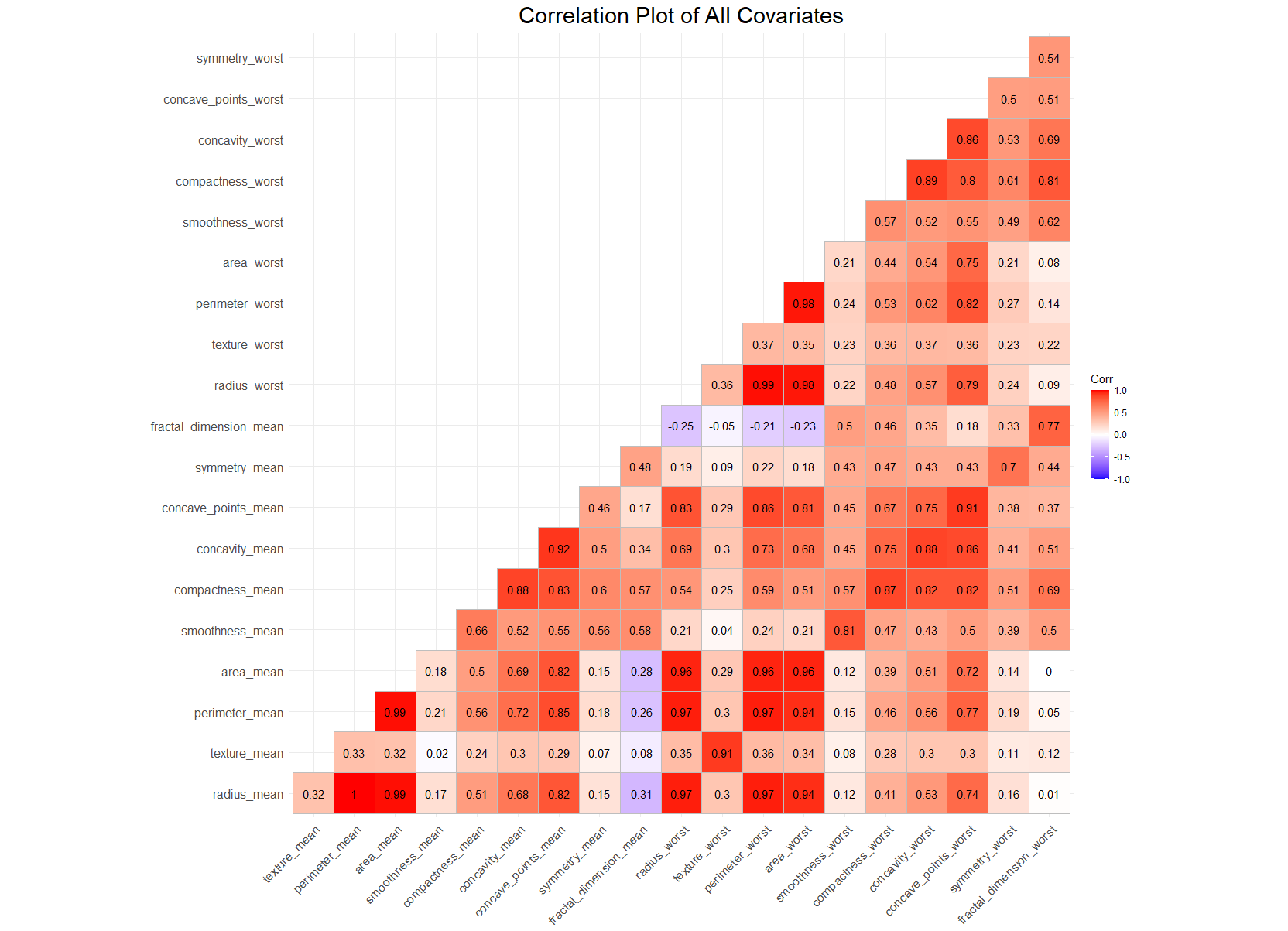
## [1] FALSE

### Correlation and multicolinearity

Next the covariate matrix was checked for potential multicollinearity issues. There did appear to be several highly correlated covariates in the dataset that could impact model performance. Most of the highly correlated variables were related to radiuses, perimeters, and areas.

ggcorrplot(cor(b\_cancer[-1]), type = 'lower', lab = TRUE) +  
 ggtitle("Correlation Plot of All Covariates") +   
 theme(plot.title = element\_text(hjust = 0.5, size = 22))

knitr::include\_graphics(paste0(getwd(),"/Cancer\_data\_plots/correlation\_plot.png"))

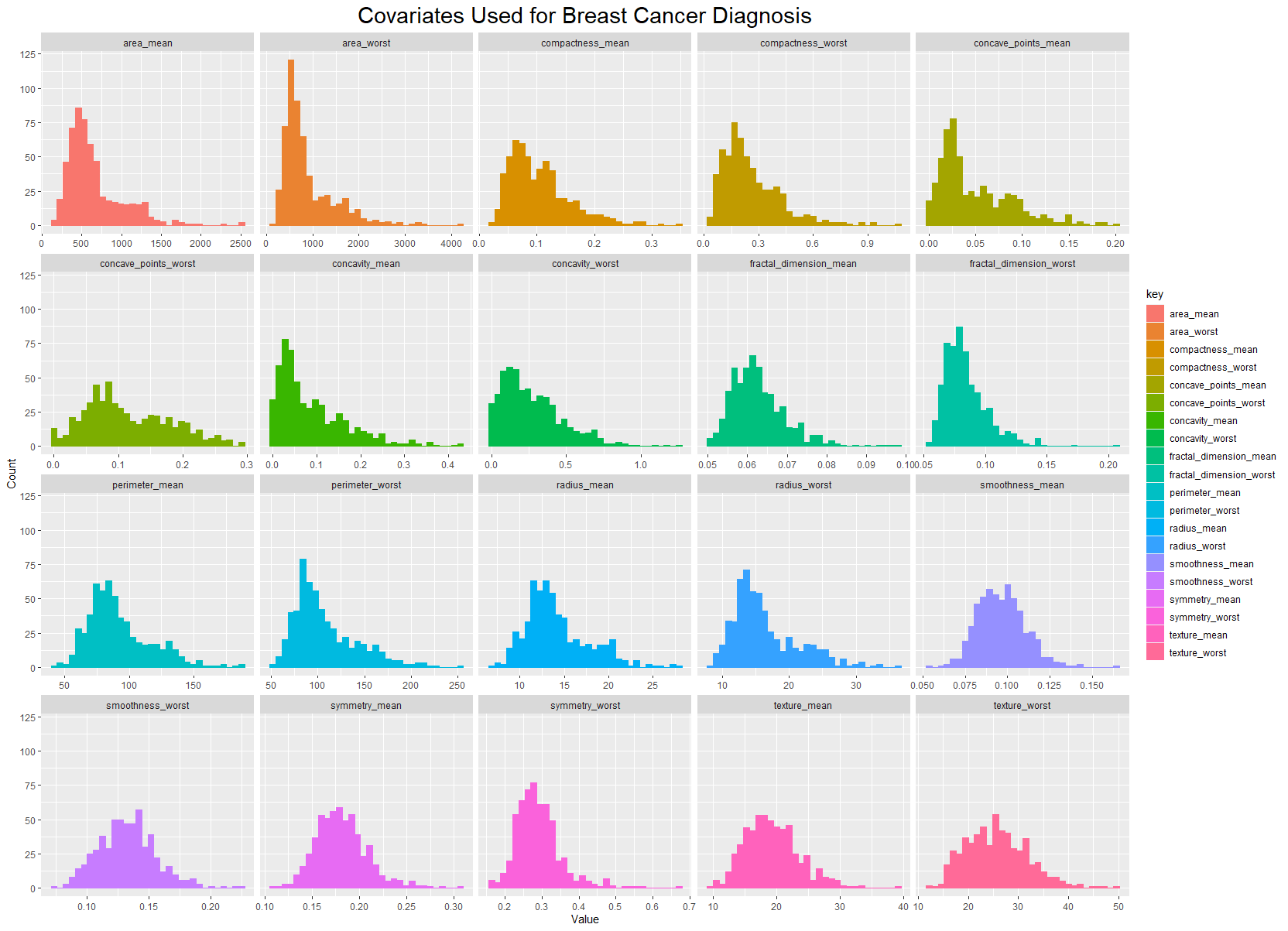


### Covariate Histograms

It also appears that there’s a variety of spread for each covariate, ranging from extremely right skewed too moderately skewed, to normally distributed. Additionally, not all of the covariates share similar scales. Dissimilar data can sometimes impact model performance, and some models are more susceptible to performance loss than others.

ggplot(gather(b\_cancer[,-1]), aes(x = value, color = key, fill = key)) +  
 geom\_histogram(bins = 32) +  
 ggtitle("Covariates Used for Breast Cancer Diagnosis") +  
 xlab("Value") + ylab("Count") +  
 theme(plot.title = element\_text(hjust = 0.5, size = 22)) +  
 facet\_wrap(~key, scales = 'free\_x')

knitr::include\_graphics(paste0(getwd(), "/Cancer\_data\_plots/histograms\_all\_covariates.png"))

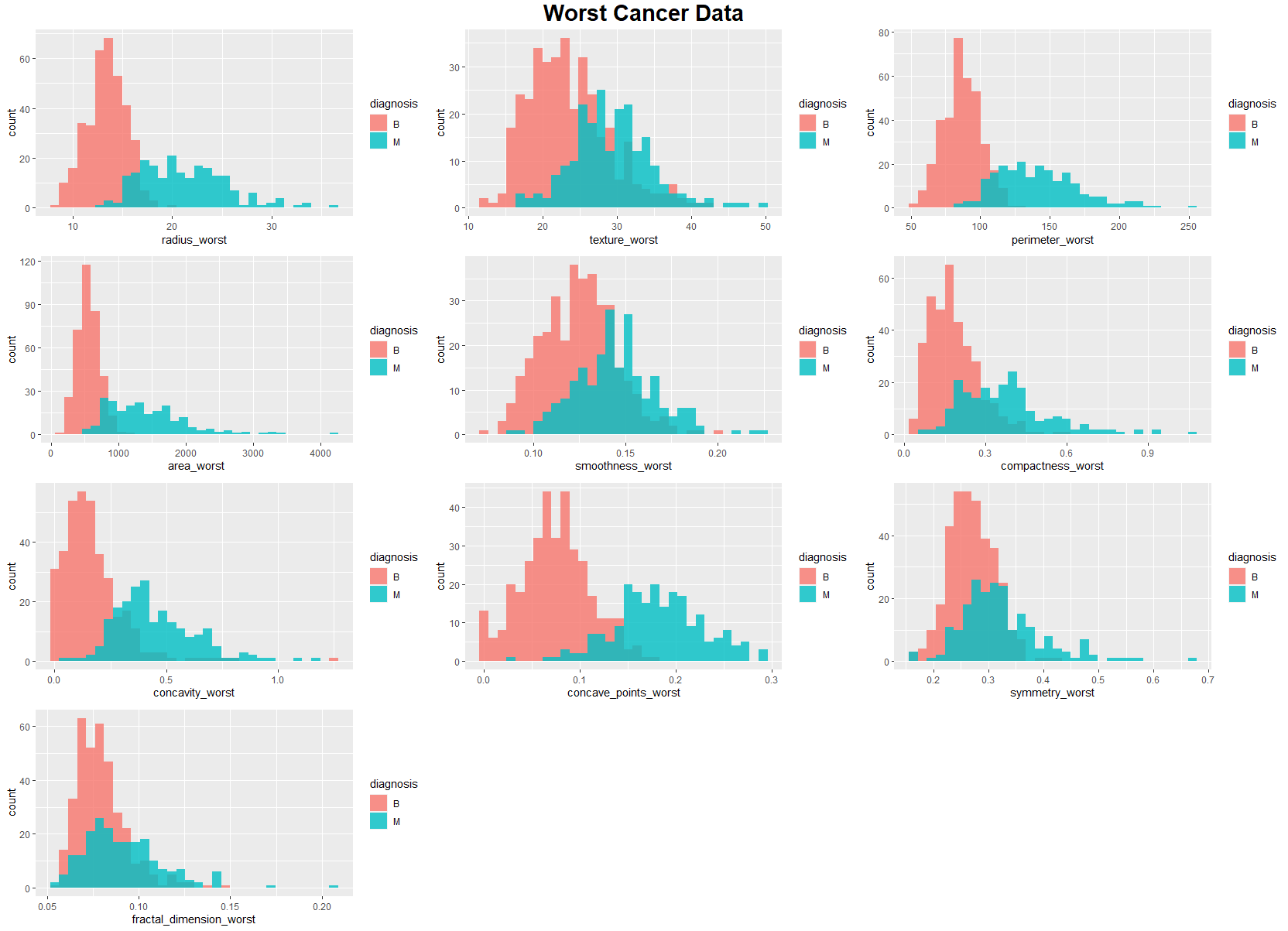
 Lastly, the dataset was split into mean and worst cases, and histograms were plotted by diagnosis (Benign or Malignant). This was done to identify any potential differences in frequencies by diagnosis.

#Plotting histograms of covariates grouped by diagnosis, for mean/worst  
hist <- list()  
for(i in names(b\_cancer[,-1])){  
 hist[[i]] <- ggplot(data = b\_cancer, aes\_string(x = i,  
 fill = "diagnosis")) +  
 geom\_histogram(position = 'identity', alpha = 0.8, bins = 32)   
}

##### Plotting the worst cancer data

#Worst count covariates  
grep('worst', names(b\_cancer[,-1]))  
grid.arrange(hist[[11]], hist[[12]], hist[[13]], hist[[14]], hist[[15]],  
 hist[[16]], hist[[17]], hist[[18]], hist[[19]], hist[[20]],  
 nrow = 4,  
 top = textGrob("Worst Cancer Data",  
 gp = gpar(fontsize = 22, font = 2)))

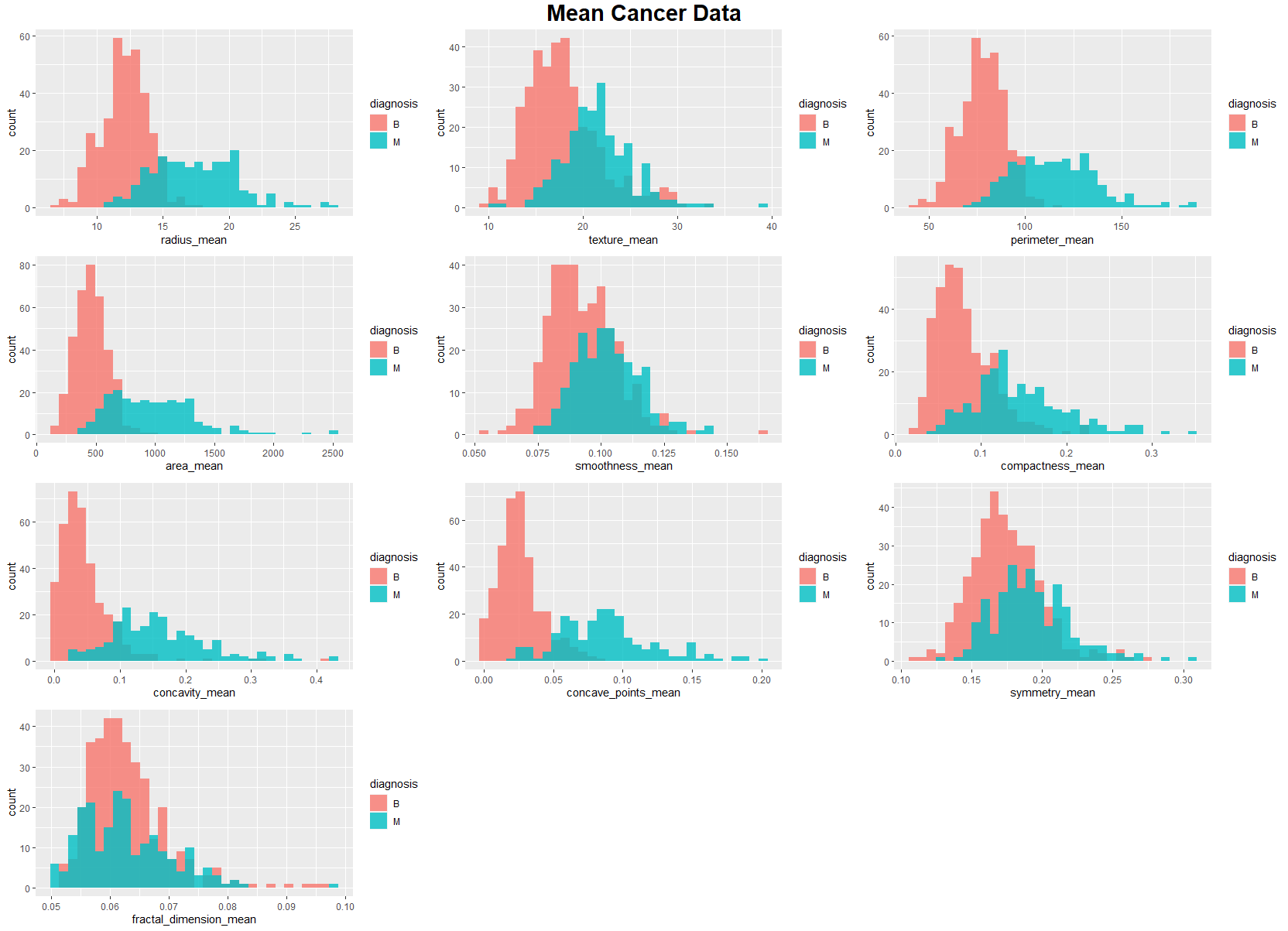
knitr::include\_graphics(paste0(getwd(), "/Cancer\_data\_plots/worst\_histogram\_by\_diag.png"))



##### Plotting the mean cancer data

#Mean count covariates  
grep('mean', names(b\_cancer[,-1]))  
grid.arrange(hist[[1]], hist[[2]], hist[[3]], hist[[4]], hist[[5]],  
 hist[[6]], hist[[7]], hist[[8]], hist[[9]], hist[[10]],  
 nrow = 4,  
 top = textGrob("Mean Cancer Data",  
 gp = gpar(fontsize =22, font = 2)))

knitr::include\_graphics(paste0(getwd(), "/Cancer\_data\_plots/mean\_histogram\_by diag.png"))



These plots clearly show differences in the values for most of the covariates by diagnosis. Separations like these will be beneficial for classification model training for both accuracy and misclassification rates. The variables that do have clear separations are likely to be highly predictive of cancer diagnosis, too.

# Model Implementation

## Implementation steps

Three different types of models were used for this analysis: logistic regression, random forests, and gradient boosting with XGBoost. Each were applied to the total dataset, just the mean cancer data, and just the worst cancer data. Each model generally followed these steps:

* Loaded datasets and divided into 75/25 train-test split
* Model specific data formatting
* Subset selection or hyperparameter tuning
* Final model summary
* Test data prediction and analysis
* Model diagnostics

## Logistic Regression

Logistic regression is a type of generalized linear model that works with binary response variables and can be easily interpreted. One of the benefits of the logistic regression is that it can identify the log-odds and odds ratios for covariates of interest. More information about logistic regression can be found [here](https://courses.lumenlearning.com/introstats1/chapter/introduction-to-logistic-regression/).

### Loading packages

### Installing necessary packages  
#install.packages("tidyverse")  
#install.packages("caTools") # For Logistic regression  
#install.packages("ROCR") # For ROC curve to evaluate model  
#install.packages("pscl") # Model evaluation  
  
### Loading package  
library(plyr)  
library(tidyverse)  
library(caTools)  
library(ROCR)  
library(carData)  
library(caret)  
library(car)  
library(pscl)

### All Cancer Data

##### Formatting data

###load dataset  
data\_all <- readxl::read\_xlsx("Breast Cancer data - CS 5610.xlsx")  
#remove ids and standard errors, setting diagnosis to factor variable  
# factor set 1 == "M", 0 == "B"  
data\_all <- data\_all[,-1]  
data\_all <- data\_all[, !grepl('\_se', colnames(data\_all))]  
colnames(data\_all)[c(9, 19)] <- c("concave\_points\_mean", "concave\_points\_worst")  
data\_all$diagnosis <- as.factor(data\_all$diagnosis)  
data\_all$diagnosis <- as.integer(data\_all$diagnosis)-1

### Correlation plot for whole dataset   
findCorrelation(cor(data\_all[-1]), cutoff = 0.75, names = TRUE)

## [1] "concave\_points\_worst" "concave\_points\_mean"   
## [3] "concavity\_mean" "compactness\_mean"   
## [5] "concavity\_worst" "perimeter\_worst"   
## [7] "compactness\_worst" "radius\_worst"   
## [9] "perimeter\_mean" "area\_worst"   
## [11] "area\_mean" "smoothness\_worst"   
## [13] "fractal\_dimension\_worst" "texture\_mean"

# 14 variables that have at least one correlation above 0.75

##### Train-test split

### Splitting dataset dividing data 75/25 split  
set.seed(99)  
split <- sample.split(data\_all$diagnosis, SplitRatio = 0.75)  
  
train\_all <- subset(data\_all, split == "TRUE")  
test\_all <- subset(data\_all, split == "FALSE")

##### Full model

### Training model full model and summary output  
logistic\_full <- glm(diagnosis ~ ., data=train\_all, family="binomial")

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

pR2(logistic\_full)["McFadden"]

## fitting null model for pseudo-r2

## McFadden   
## 0.9084415

vif(logistic\_full)

## radius\_mean texture\_mean perimeter\_mean   
## 1814.660498 13.252368 1077.329499   
## area\_mean smoothness\_mean compactness\_mean   
## 603.890780 17.151417 80.098313   
## concavity\_mean concave\_points\_mean symmetry\_mean   
## 41.137726 16.287288 5.390728   
## fractal\_dimension\_mean radius\_worst texture\_worst   
## 31.875074 391.940065 9.281670   
## perimeter\_worst area\_worst smoothness\_worst   
## 74.294101 279.561432 5.543660   
## compactness\_worst concavity\_worst concave\_points\_worst   
## 58.542447 36.312709 8.271853   
## symmetry\_worst fractal\_dimension\_worst   
## 4.359011 32.192332

The full model appeared to fit the data well with a high McFadden R2 = 0.9084415, but it only had one significant coefficient, and coefficients had high variable inflation factors.

#we can assume that multicollinearity is an issue in our model. So, we have   
#values above 5 indicate severe multicollinearity such that radius\_worst and perimeter\_worst.  
# Set a VIF threshold. All the variables having higher VIF than threshold  
#are dropped from the model  
threshold=4.99  
  
### Sequentially drop the variable with the largest VIF until  
# all variables have VIF less than threshold  
logistic\_all <- logistic\_full  
drop=TRUE  
  
aftervif=data.frame()  
while(drop==TRUE) {  
 vmodel=vif(logistic\_all)  
 aftervif=rbind.fill(aftervif,as.data.frame(t(vmodel)))  
 if(max(vmodel)>threshold) {  
 logistic\_all=update(logistic\_all,as.formula(paste(".","~",".","-",names(which.max(vmodel))))) }  
 else { drop=FALSE }}

#Model after removing correlated Variables with their VIFs  
print(as.data.frame(vmodel))

## vmodel  
## texture\_mean 1.554087  
## concavity\_mean 3.728442  
## concave\_points\_mean 4.798353  
## symmetry\_mean 2.777392  
## fractal\_dimension\_mean 3.269113  
## perimeter\_worst 1.545516  
## smoothness\_worst 2.385307  
## concave\_points\_worst 3.296249  
## symmetry\_worst 2.866333

##### Creating predictions

### Use the Model to Make Predictions on test data  
# Predict test data, converting to 0 or 1 based on 0.5 cutoff value  
predict\_reg <- predict(logistic\_all, test\_all, type = "response")  
predict\_reg <- ifelse(predict\_reg > 0.5, 1, 0)  
predict\_reg <- as.vector(predict\_reg)

##### Model diagnostics and evaluation

# Diagnostics plots  
par(mfrow = c(2,2))  
plot(logistic\_all, which = 1:4, main = "All Cancer Data")  
  
### ROC-AUC Curve  
ROCPred <- prediction(predict\_reg, test\_all$diagnosis)  
ROCPer <- performance(ROCPred, measure = "tpr",  
 x.measure = "fpr")  
  
auc <- performance(ROCPred, measure = "auc")  
auc <- auc@y.values[[1]]  
auc  
  
### Plotting curve  
par(mfrow = c(1,1))  
plot(ROCPer, main = "ROC Curve for All Cancer Data")  
abline(a = 0, b = 1)  
auc <- round(auc, 4)  
legend(.6, .4, auc, title = "AUC", cex = 1)

### Evaluating model accuracy  
predict\_reg <- factor(ifelse(predict\_reg > 0.5, 1, 0),  
 labels = c("B", "M"))  
test\_all$diagnosis <- factor(ifelse(test\_all$diagnosis > 0.5, 1, 0),  
 labels = c("B", "M"))  
all\_confusion <- caret::confusionMatrix(test\_all$diagnosis, predict\_reg,  
 mode = 'everything',  
 positive = 'M')  
all\_r2 <- pR2(logistic\_all)["McFadden"]

### Mean Cancer Data

Here is the same setup, but for just the data that includes mean cancer data.

###load dataset  
data\_mean <- read.csv("breast\_cancer\_mean.csv")  
  
#r Setting diagnosis to factor variable: factor set 1 == "M", 0 == "B"  
data\_mean$diagnosis <- as.factor(data\_mean$diagnosis)  
data\_mean$diagnosis <- as.integer(data\_mean$diagnosis)-1  
  
### Summary of dataset in package  
summary(data\_mean)  
nrow(data\_mean)  
  
### Correlation plot for whole dataset   
#pairs(data\_mean[-1])  
findCorrelation(cor(data\_mean[-1]), cutoff = 0.7, names = TRUE)  
  
### Splitting dataset dividing data 75/25 split  
set.seed(99)  
split <- sample.split(data\_mean$diagnosis, SplitRatio = 0.75)  
head(split)  
  
train\_mean <- subset(data\_mean, split == "TRUE")  
test\_mean <- subset(data\_mean, split == "FALSE")  
  
### Training model full model and summary output  
logistic\_mean <- glm(diagnosis ~ ., data=train\_mean, family="binomial")  
logistic\_mean  
summary(logistic\_mean)  
  
#Assessing Model Fit  
#We can compute McFadden's R2 for our model using the pR2 function from the pscl package.  
  
pR2(logistic\_mean)["McFadden"]  
#A value of 0.9084415 is quite high for McFadden's R2,   
#which indicates that our model fits the data very well and has high predictive power.  
  
#Variable Importance  
varImp(logistic\_mean, sort = TRUE)  
  
#calculate VIF values for each predictor variable in our model  
vif(logistic\_mean)  
  
#we can assume that multicollinearity is an issue in our model. So, we have   
#values above 5 indicate severe multicollinearity such that radius\_worst and perimeter\_worst.  
# Set a VIF threshold. All the variables having higher VIF than threshold  
#are dropped from the model  
threshold=4.99  
  
### Sequentially drop the variable with the largest VIF until  
# all variables have VIF less than threshold  
drop=TRUE  
  
aftervif=data.frame()  
while(drop==TRUE) {  
 vmodel=vif(logistic\_mean)  
 aftervif=rbind.fill(aftervif,as.data.frame(t(vmodel)))  
 if(max(vmodel)>threshold) {  
 logistic\_mean=update(logistic\_mean,as.formula(paste(".","~",".","-",names(which.max(vmodel))))) }  
 else { drop=FALSE }}  
  
### How variables removed sequentially  
t\_aftervif= as.data.frame(t(aftervif))

# Final (uncorrelated) variables with their VIFs  
print(as.data.frame(vmodel))

## vmodel  
## texture\_mean 1.778788  
## area\_mean 1.983117  
## smoothness\_mean 2.954579  
## compactness\_mean 3.592998  
## concavity\_mean 2.585441  
## symmetry\_mean 1.817450

### Use the Model to Make Predictions on test data  
# Predict test data, converting to 0 or 1 based on 0.5 cutoff value  
predict\_reg <- predict(logistic\_mean, test\_mean, type = "response")  
predict\_reg <- ifelse(predict\_reg > 0.5, 1, 0)  
predict\_reg <- as.vector(predict\_reg)

### Model Diagnostics  
# Diagnostic plots  
par(mfrow = c(2,2))  
plot(logistic\_mean, which = 1:4, main = "Mean Cancer Data")  
  
# ROC-AUC Curve  
ROCPred <- prediction(predict\_reg, test\_mean$diagnosis)  
ROCPer <- performance(ROCPred, measure = "tpr",  
 x.measure = "fpr")  
  
auc <- performance(ROCPred, measure = "auc")  
auc <- auc@y.values[[1]]  
auc  
  
### Plotting curve  
par(mfrow = c(1,1))  
plot(ROCPer, main = "ROC Curve for Mean Cancer Data")  
abline(a = 0, b = 1)  
auc <- round(auc, 4)  
legend(.6, .4, auc, title = "AUC", cex = 1)

### Evaluating model accuracy  
predict\_reg <- factor(ifelse(predict\_reg > 0.5, 1, 0),  
 labels = c("B", "M"))  
test\_mean$diagnosis <- factor(ifelse(test\_mean$diagnosis > 0.5, 1, 0),  
 labels = c("B", "M"))  
mean\_confusion<- caret::confusionMatrix(test\_mean$diagnosis, predict\_reg,  
 mode = 'everything',  
 positive = 'M')  
mean\_r2 <- pR2(logistic\_mean)["McFadden"]

## fitting null model for pseudo-r2

### Worst Cancer Data

Here is the same setup, but for just the worst cancer data.

###load dataset  
data\_worst <- read.csv("breast\_cancer\_worst.csv")  
  
#r Setting diagnosis to factor variable: factor set 1 == "M", 0 == "B"  
data\_worst$diagnosis <- as.factor(data\_worst$diagnosis)  
data\_worst$diagnosis <- as.integer(data\_worst$diagnosis)-1  
  
### Summary of dataset in package  
summary(data\_worst)  
nrow(data\_worst)  
  
### Correlation plot for whole dataset   
#pairs(data\_worst[-1])  
findCorrelation(cor(data\_worst[-1]), cutoff = 0.7, names = TRUE)  
  
### Splitting dataset dividing data 75/25 split  
set.seed(99)  
split <- sample.split(data\_worst$diagnosis, SplitRatio = 0.75)  
head(split)  
  
train\_worst <- subset(data\_worst, split == "TRUE")  
test\_worst <- subset(data\_worst, split == "FALSE")  
  
### Training model full model and summary output  
logistic\_worst <- glm(diagnosis ~ ., data=train\_worst, family="binomial")  
logistic\_worst  
  
summary(logistic\_worst)  
  
#Assessing Model Fit  
#We can compute McFadden's R2 for our model using the pR2 function from the pscl package.  
  
pR2(logistic\_worst)["McFadden"]  
#A value of 0.9084415 is quite high for McFadden's R2,   
#which indicates that our model fits the data very well and has high predictive power.  
  
#Variable Importance  
varImp(logistic\_worst, sort = TRUE)  
  
#calculate VIF values for each predictor variable in our model  
vif(logistic\_worst)  
  
#we can assume that multicollinearity is an issue in our model. So, we have   
#values above 5 indicate severe multicollinearity such that radius\_worst and perimeter\_worst.  
# Set a VIF threshold. All the variables having higher VIF than threshold  
#are dropped from the model  
threshold=4.99  
  
  
### Sequentially drop the variable with the largest VIF until  
# all variables have VIF less than threshold  
drop=TRUE  
  
aftervif=data.frame()  
while(drop==TRUE) {  
 vmodel=vif(logistic\_worst)  
 aftervif=rbind.fill(aftervif,as.data.frame(t(vmodel)))  
 if(max(vmodel)>threshold) {  
 logistic\_worst=update(logistic\_worst,as.formula(paste(".","~",".","-",names(which.max(vmodel))))) }  
 else { drop=FALSE }}  
  
#Model after removing correlated Variables  
summary(logistic\_worst)  
vif(logistic\_worst)  
  
  
### How variables removed sequentially  
t\_aftervif= as.data.frame(t(aftervif))

# Final (uncorrelated) variables with their VIFs  
print(as.data.frame(vmodel))

## vmodel  
## texture\_worst 1.429116  
## area\_worst 1.496540  
## smoothness\_worst 2.555601  
## concavity\_worst 3.040412  
## concave\_points\_worst 2.707778  
## symmetry\_worst 1.368137  
## fractal\_dimension\_worst 3.729195

### Use the Model to Make Predictions on test data  
# Predict test data, converting to 0 or 1 based on 0.5 cutoff value  
predict\_reg <- predict(logistic\_worst, test\_worst, type = "response")  
predict\_reg <- ifelse(predict\_reg > 0.5, 1, 0)  
predict\_reg <- as.vector(predict\_reg)

### Model Diagnostics  
# Diagnostic plots  
par(mfrow = c(2,2))  
plot(logistic\_worst, which = 1:4, main = "Worst Cancer Data")  
  
# ROC-AUC Curve  
ROCPred <- prediction(predict\_reg, test\_worst$diagnosis)  
ROCPer <- performance(ROCPred, measure = "tpr",  
 x.measure = "fpr")  
  
auc <- performance(ROCPred, measure = "auc")  
auc <- auc@y.values[[1]]  
auc  
  
### Plotting curve  
par(mfrow = c(1,1))  
plot(ROCPer, main = "ROC Curve for Worst Cancer Data")  
abline(a = 0, b = 1)  
auc <- round(auc, 4)  
legend(.6, .4, auc, title = "AUC", cex = 1)

### Evaluating model accuracy  
predict\_reg <- factor(ifelse(predict\_reg > 0.5, 1, 0),  
 labels = c("B", "M"))  
test\_worst$diagnosis <- factor(ifelse(test\_worst$diagnosis > 0.5, 1, 0),  
 labels = c("B", "M"))  
worst\_confusion<- caret::confusionMatrix(test\_worst$diagnosis, predict\_reg,  
 mode = 'everything',  
 positive = 'M')  
worst\_r2 <- pR2(logistic\_worst)["McFadden"]

## Random forest

A random forest algorithm that uses ensemble learning for either regression or classification problems. A random forest establishes predictions based on the aggerate outcomes of multiple decision trees. Decision trees are intuitive, as they are similar to a heuristics like flow charts. Random forests rely on a majority-voting system, in which the final prediction is determined by the outcome that is consistent with the majority of the decision trees. That is, if a majority of the ensemble decision trees predicted “Yes”, the final random forests prediction would be the same. More information about random forests can be found [here](https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/).

### Preping Data for Analysis

#Loading required libraries  
library(tidyverse)  
library(randomForest)  
library(caTools)  
library(caret)  
  
#Loading datasets  
breast\_cancer <- readxl::read\_xlsx('Breast Cancer data - CS 5610.xlsx')  
cancer\_mean <- read.csv('breast\_cancer\_mean.csv')  
cancer\_worst <- read.csv('breast\_cancer\_worst.csv')  
  
#Removing SE columns and renaming two columns for cancer\_all  
cancer\_all <- breast\_cancer[, -1]  
cancer\_all <- cancer\_all[, !grepl('\_se', colnames(cancer\_all))]  
colnames(cancer\_all)[c(9, 19)] <- c("concave\_points\_mean", "concave\_points\_worst")  
  
#Setting 'diagnosis' to factor variable  
cancer\_all$diagnosis <- as.factor(cancer\_all$diagnosis)  
cancer\_mean$diagnosis <- as.factor(cancer\_mean$diagnosis)  
cancer\_worst$diagnosis <- as.factor(cancer\_worst$diagnosis)

### All Cancer Data

##### Splitting dataset

set.seed(99)  
sampl\_all <- sample.split(cancer\_all$diagnosis, SplitRatio = 0.75)  
train\_all <- subset(cancer\_all, sampl\_all == TRUE)  
test\_all <- subset(cancer\_all, sampl\_all != TRUE)

##### Hyperparameter tuning and model training

control <- trainControl(method = 'cv', number = 5, search = 'grid')  
tunegrid <- expand.grid(mtry = c(1:ncol(train\_all)))  
set.seed(99)  
test\_rf <- train(diagnosis ~., data = train\_all, method = 'rf',  
 metric = 'Accuracy',  
 tuneGrid = tunegrid,  
 trControl = control)  
  
test\_rf$bestTune

## mtry  
## 20 20

## building training model  
set.seed(99)  
rf\_all <- randomForest(diagnosis ~., data = train\_all,   
 ntree = 500,  
 mtry = 20,  
 importance = TRUE)

##### Model testing and visualizations

cancer\_all\_rf.pred <- predict(rf\_all, newdata = test\_all)  
confusionMatrix(cancer\_all\_rf.pred, test\_all$diagnosis,  
 mode = 'everything',  
 positive = 'M')  
  
plot(test\_rf, main = "CV Accuracy per Number of Included Predictors",  
 sub = "All Cancer Data")

varImpPlot(rf\_all, main = "Variable Importance: All Cancer Data")

### Mean Cancer Data

##### Splitting dataset

set.seed(99)  
sampl\_mean <- sample.split(cancer\_mean$diagnosis, SplitRatio = 0.75)  
train\_mean <- subset(cancer\_mean, sampl\_mean == TRUE)  
test\_mean <- subset(cancer\_mean, sampl\_mean != TRUE)

##### Hyperparameter tuning and model training

control <- trainControl(method = 'cv', number = 5, search = 'grid')  
tunegrid <- expand.grid(mtry = c(1:ncol(train\_mean)))  
set.seed(99)  
test\_rf <- train(diagnosis ~., data = train\_mean, method = 'rf',  
 metric = 'Accuracy',  
 tuneGrid = tunegrid,  
 trControl = control)  
  
test\_rf$bestTune

## mtry  
## 8 8

## building training model  
set.seed(99)  
rf\_mean <- randomForest(diagnosis ~., data = train\_mean,   
 ntree = 500,  
 mtry = 8,  
 importance = TRUE)

##### Model testing and visualizations

cancer\_mean\_rf.pred <- predict(rf\_mean, newdata = test\_mean)  
confusionMatrix(cancer\_mean\_rf.pred, test\_mean$diagnosis,  
 mode = 'everything',  
 positive = 'M')  
  
plot(test\_rf, main = "CV Accuracy per Number of Included Predictors",  
 sub = "Mean Cancer Data")

varImpPlot(rf\_mean, main = "Variable Importance: Mean Cancer Data")

### Worst Cancer Data

##### Splitting dataset

set.seed(99)  
sampl\_worst <- sample.split(cancer\_worst$diagnosis, SplitRatio = 0.75)  
train\_worst <- subset(cancer\_worst, sampl\_worst == TRUE)  
test\_worst <- subset(cancer\_worst, sampl\_worst != TRUE)

##### Hyperparameter tuning and model training

control <- trainControl(method = 'cv', number = 5, search = 'grid')  
tunegrid <- expand.grid(mtry = c(1:ncol(train\_worst)))  
set.seed(99)  
test\_rf <- train(diagnosis ~., data = train\_worst, method = 'rf',  
 metric = 'Accuracy',  
 tuneGrid = tunegrid,  
 trControl = control)  
  
test\_rf$bestTune

## mtry  
## 10 10

## building training model  
set.seed(99)  
rf\_worst <- randomForest(diagnosis ~., data = train\_worst,  
 ntree = 500,   
 mtry = 10,  
 importance = TRUE)

##### Model testing and visualizations

cancer\_worst\_rf.pred <- predict(rf\_worst, newdata = test\_worst)  
confusionMatrix(cancer\_worst\_rf.pred, test\_worst$diagnosis,  
 mode = 'everything',  
 positive = 'M')  
  
plot(test\_rf, main = "CV Accuracy per Number of Included Predictors",  
 sub = "Worst Cancer Data")

varImpPlot(rf\_worst, main = "Variable Importance: Worst Cancer Data")

## XGBoost

XGBoost stands for “extreme gradient boosting”; it is an extension of gradient boosted decision trees that is optimized to improve speed and performance. In general, gradient boosting refers to iteratively fitting residuals of a loss function from the previous fitted model to the next fitted model. XGBoost is a recent library that uses a gradient boosted decision trees for classification problems, and gradient boosted generalized linear models for regression problems. More information about the XGBoost library can be found [here](https://medium.com/analytics-vidhya/introduction-to-xgboost-algorithm-d2e7fad76b04).

### Preping Data for Analysis

#Loading packages  
#install.packages('xgboost')  
library(xgboost)  
library(caret)  
library(caTools)  
library(tidyverse)  
library(pROC)  
  
#Loading datasets  
breast\_cancer <- readxl::read\_xlsx('Breast Cancer data - CS 5610.xlsx')  
cancer\_mean <- read.csv('breast\_cancer\_mean.csv')  
cancer\_worst <- read.csv('breast\_cancer\_worst.csv')  
  
#Removing SE columns and renaming two columns for cancer\_all  
cancer\_all <- breast\_cancer[, -1]  
cancer\_all <- cancer\_all[, !grepl('\_se', colnames(cancer\_all))]  
colnames(cancer\_all)[c(9, 19)] <- c("concave\_points\_mean", "concave\_points\_worst")  
  
#Setting 'diagnosis' to factor variable  
cancer\_all$diagnosis <- as.factor(cancer\_all$diagnosis)  
cancer\_mean$diagnosis <- as.factor(cancer\_mean$diagnosis)  
cancer\_worst$diagnosis <- as.factor(cancer\_worst$diagnosis)

### All Cancer Data

##### Splitting into train/test data

set.seed(99)  
sampl\_all <- sample.split(cancer\_all$diagnosis, SplitRatio = 0.75)  
train\_all <- subset(cancer\_all, sampl\_all == TRUE)  
test\_all <- subset(cancer\_all, sampl\_all != TRUE)

##### Formatting data for XGBoost modeling

## Creating the independent variable and label matricies of train/test data  
train\_all\_data <- as.matrix(train\_all[-1])  
train\_all\_label <- train\_all$diagnosis  
## Converting labels to 0,1 where "M" is coded at 1  
train\_all\_label <- as.integer(train\_all\_label)-1  
train\_all$diagnosis[1:5]; train\_all\_label[1:5]

## [1] M M M M M  
## Levels: B M

## [1] 1 1 1 1 1

## Repeat for test dataset  
test\_all\_data <- as.matrix(test\_all[-1])  
test\_all\_label <- test\_all$diagnosis  
test\_all\_label <- as.integer(test\_all\_label)-1  
test\_all$diagnosis[1:5]; test\_all\_label[1:5]

## [1] M M B M M  
## Levels: B M

## [1] 1 1 0 1 1

## Formatting data for XGBoost matricies  
all\_dtrain <- xgb.DMatrix(data = train\_all\_data, label = train\_all\_label)  
all\_dtest <- xgb.DMatrix(data = test\_all\_data, label = test\_all\_label)

##### Hyperparameter tuning using random search

### parameters: max\_depth, eta, subsample, colsample\_bytree, and min\_child\_weight  
all\_low\_err\_list <- list()  
all\_parameters\_list <- list()  
set.seed(99)  
for(i in 1:3000){  
 params <- list(booster = "gbtree",  
 objective = "binary:logistic",  
 max\_depth = sample(3:25, 1),  
 eta = runif(1, 0.01, 0.3),  
 subsample = runif(1, 0.5, 1),  
 colsample\_bytree = runif(1, 0.5, 1),  
 min\_child\_weight = sample(0:10, 1)  
 )  
   
 parameters <- as.data.frame(params)  
 all\_parameters\_list[[i]] <- parameters  
}  
  
all\_parameters\_df <- do.call(rbind, all\_parameters\_list) #df containing random search params  
  
### Fitting xgboost models based on search parameters  
for (row in 1:nrow(all\_parameters\_df)){  
 set.seed(99)  
 all\_tmp\_mdl <- xgb.cv(data = all\_dtrain,  
 booster = "gbtree",  
 objective = "binary:logistic",  
 nfold = 5,  
 prediction = TRUE,  
 max\_depth = all\_parameters\_df$max\_depth[row],  
 eta = all\_parameters\_df$eta[row],  
 subsample = all\_parameters\_df$subsample[row],  
 colsample\_bytree = all\_parameters\_df$colsample\_bytree[row],  
 min\_child\_weight = all\_parameters\_df$min\_child\_weight[row],  
 nrounds = 200,  
 eval\_metric = "error",  
 early\_stopping\_rounds = 20,  
 print\_every\_n = 500,  
 verbose = 0  
 )  
   
 #this is the lowest error for the iteration  
 all\_low\_err <- as.data.frame(1 - min(all\_tmp\_mdl$evaluation\_log$test\_error\_mean))  
 all\_low\_err\_list[[row]] <- all\_low\_err  
}  
  
all\_low\_err\_df <- do.call(rbind, all\_low\_err\_list) #accuracies   
all\_randsearch <- cbind(all\_low\_err\_df, all\_parameters\_df) #data frame with everything  
  
###Reformatting the dataframe  
all\_randsearch <- all\_randsearch %>%  
 dplyr::rename(val\_acc = '1 - min(all\_tmp\_mdl$evaluation\_log$test\_error\_mean)') %>%  
 dplyr::arrange(-val\_acc)  
  
###Grabbing just the top model  
all\_randsearch\_best <- all\_randsearch[1,]  
  
###Storing best parameters in list  
all\_best\_params <- list(booster = all\_randsearch\_best$booster,  
 objective = all\_randsearch\_best$objective,  
 max\_depth = all\_randsearch\_best$max\_depth,  
 eta = all\_randsearch\_best$eta,  
 subsample = all\_randsearch\_best$subsample,  
 colsample\_bytree = all\_randsearch\_best$colsample\_bytree,  
 min\_child\_weight = all\_randsearch\_best$min\_child\_weight)

##### Hyperparameter tuning nround with 5-fold cross validation

### Finding the best nround parameter for the model using 5-fold cross validation  
set.seed(99)  
all\_xgbcv <- xgb.cv(params = all\_best\_params,  
 data = all\_dtrain,  
 nrounds = 500,  
 nfold = 5,  
 prediction = TRUE,  
 print\_every\_n = 50,  
 early\_stopping\_rounds = 25,  
 eval\_metric = "error",  
 verbose = 0  
 )  
all\_xgbcv$best\_iteration

##### Model training using best hyperparameters

set.seed(99)  
all\_best\_xgb <- xgb.train(params = all\_best\_params,  
 data = all\_dtrain,  
 nrounds = all\_xgbcv$best\_iteration,  
 eval\_metric = "error",  
 )  
  
xgb.save(all\_best\_xgb, 'final\_xgb\_cancerall')

## [1] TRUE

##### Model testing and visualizations

cancer\_all.pred <- predict(all\_best\_xgb, all\_dtest)  
cancer\_all.pred <- factor(ifelse(cancer\_all.pred > 0.5, 1, 0),  
 labels = c("B", "M"))  
confusionMatrix(cancer\_all.pred, test\_all$diagnosis,  
 mode = 'everything',  
 positive = 'M')  
  
## Visualizations  
all\_impt\_mtx <- xgb.importance(feature\_names = colnames(test\_all\_data), model = all\_best\_xgb)  
xgb.plot.importance(importance\_matrix = all\_impt\_mtx,  
 xlab = "Variable Importance")

### ROC curve for 5-fold CV random parameter search  
all\_randsearch\_roc <- roc(response = train\_all\_label,  
 predictor = all\_tmp\_mdl$pred,  
 print.auc = TRUE,  
 plot = TRUE)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

### ROC curve for 5-fold CV nround parameter search  
all\_nround\_roc <- roc(response = train\_all\_label,  
 predictor = all\_xgbcv$pred,  
 print.auc = TRUE,  
 plot = TRUE)

## Setting levels: control = 0, case = 1  
## Setting direction: controls < cases

### Mean Cancer Data

##### Splitting into train/test data

set.seed(99)  
sampl\_mean <- sample.split(cancer\_mean$diagnosis, SplitRatio = 0.75)  
train\_mean <- subset(cancer\_mean, sampl\_mean == TRUE)  
test\_mean <- subset(cancer\_mean, sampl\_mean != TRUE)

##### Formatting data for XGBoost modeling

## Creating the independent variable and label matricies of train/test data  
train\_mean\_data <- as.matrix(train\_mean[-1])  
train\_mean\_label <- train\_mean$diagnosis  
## Converting labels to 0,1 where "M" is coded at 1  
train\_mean\_label <- as.integer(train\_mean\_label)-1  
train\_mean$diagnosis[1:5]; train\_mean\_label[1:5]

## [1] M M M M M  
## Levels: B M

## [1] 1 1 1 1 1

## Repeat for test dataset  
test\_mean\_data <- as.matrix(test\_mean[-1])  
test\_mean\_label <- test\_mean$diagnosis  
test\_mean\_label <- as.integer(test\_mean\_label)-1  
test\_mean$diagnosis[1:5]; test\_mean\_label[1:5]

## [1] M M B M M  
## Levels: B M

## [1] 1 1 0 1 1

## Formatting data for XGBoost matricies  
mean\_dtrain <- xgb.DMatrix(data = train\_mean\_data, label = train\_mean\_label)  
mean\_dtest <- xgb.DMatrix(data = test\_mean\_data, label = test\_mean\_label)

##### Hyperparameter tuning using random search

### parameters: max\_depth, eta, subsample, colsample\_bytree, and min\_child\_weight  
mean\_low\_err\_list <- list()  
mean\_parameters\_list <- list()  
set.seed(99)  
for(i in 1:3000){  
 params <- list(booster = "gbtree",  
 objective = "binary:logistic",  
 max\_depth = sample(3:25, 1),  
 eta = runif(1, 0.01, 0.3),  
 subsample = runif(1, 0.5, 1),  
 colsample\_bytree = runif(1, 0.5, 1),  
 min\_child\_weight = sample(0:10, 1)  
 )  
 parameters <- as.data.frame(params)  
 mean\_parameters\_list[[i]] <- parameters  
}  
mean\_parameters\_df <- do.call(rbind, mean\_parameters\_list) #df containing random search params  
  
### Fitting xgboost models based on search parameters  
for (row in 1:nrow(mean\_parameters\_df)){  
 set.seed(99)  
 mean\_tmp\_mdl <- xgb.cv(data = mean\_dtrain,  
 booster = "gbtree",  
 objective = "binary:logistic",  
 nfold = 5,  
 prediction = TRUE,  
 max\_depth = mean\_parameters\_df$max\_depth[row],  
 eta = mean\_parameters\_df$eta[row],  
 subsample = mean\_parameters\_df$subsample[row],  
 colsample\_bytree = mean\_parameters\_df$colsample\_bytree[row],  
 min\_child\_weight = mean\_parameters\_df$min\_child\_weight[row],  
 nrounds = 200,  
 eval\_metric = "error",  
 early\_stopping\_rounds = 20,  
 print\_every\_n = 500,  
 verbose = 0  
 )   
   
 #this is the lowest error for the iteration  
 mean\_low\_err <- as.data.frame(1 - min(mean\_tmp\_mdl$evaluation\_log$test\_error\_mean))  
 mean\_low\_err\_list[[row]] <- mean\_low\_err  
}  
  
mean\_low\_err\_df <- do.call(rbind, mean\_low\_err\_list) #accuracies   
mean\_randsearch <- cbind(mean\_low\_err\_df, mean\_parameters\_df) #data frame with everything  
  
###Reformatting the dataframe  
mean\_randsearch <- mean\_randsearch %>%  
 dplyr::rename(val\_acc = '1 - min(mean\_tmp\_mdl$evaluation\_log$test\_error\_mean)') %>%  
 dplyr::arrange(-val\_acc)  
  
###Grabbing just the top model  
mean\_randsearch\_best <- mean\_randsearch[1,]  
  
### Storing best parameters in list  
mean\_best\_params <- list(booster = mean\_randsearch\_best$booster,  
 objective = mean\_randsearch\_best$objective,  
 max\_depth = mean\_randsearch\_best$max\_depth,  
 eta = mean\_randsearch\_best$eta,  
 subsample = mean\_randsearch\_best$subsample,  
 colsample\_bytree = mean\_randsearch\_best$colsample\_bytree,  
 min\_child\_weight = mean\_randsearch\_best$min\_child\_weight)

##### Hyperparameter tuning nround with 5-fold cross validation

set.seed(99)  
mean\_xgbcv <- xgb.cv(params = mean\_best\_params,  
 data = mean\_dtrain,  
 nrounds = 500,  
 nfold = 5,  
 prediction = TRUE,  
 print\_every\_n = 50,  
 early\_stopping\_rounds = 25,  
 eval\_metric = "error",  
 verbose = 0  
 )  
mean\_xgbcv$best\_iteration

##### Model training using best hyperparameters

set.seed(99)  
mean\_best\_xgb <- xgb.train(params = mean\_best\_params,  
 data = mean\_dtrain,  
 nrounds = mean\_xgbcv$best\_iteration,  
 eval\_metric = "error",  
 )  
  
xgb.save(mean\_best\_xgb, 'final\_xgb\_cancermean')

## [1] TRUE

##### Model testing and visualizations

cancer\_mean.pred <- predict(mean\_best\_xgb, mean\_dtest)  
cancer\_mean.pred <- factor(ifelse(cancer\_mean.pred > 0.5, 1, 0),  
 labels = c("B", "M"))  
  
## Visualizations  
mean\_impt\_mtx <- xgb.importance(feature\_names = colnames(test\_mean\_data), model = mean\_best\_xgb)  
xgb.plot.importance(importance\_matrix = mean\_impt\_mtx,  
 xlab = "Variable Importance")

### ROC curve for 5-fold CV random parameter search  
mean\_randsearch\_roc <- roc(response = train\_mean\_label,  
 predictor = mean\_tmp\_mdl$pred,  
 print.auc = TRUE,  
 plot = TRUE)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

### ROC curve for 5-fold CV nround parameter search  
mean\_nround\_roc <- roc(response = train\_mean\_label,  
 predictor = mean\_xgbcv$pred,  
 print.auc = TRUE,  
 plot = TRUE)

## Setting levels: control = 0, case = 1  
## Setting direction: controls < cases

### Worst Cancer Data

##### Splitting into train/test data

set.seed(99)  
sampl\_worst <- sample.split(cancer\_worst$diagnosis, SplitRatio = 0.75)  
train\_worst <- subset(cancer\_worst, sampl\_worst == TRUE)  
test\_worst <- subset(cancer\_worst, sampl\_worst != TRUE)

##### Formatting data for XGBoost modeling

## Creating the independent variable and label matricies of train/test data  
train\_worst\_data <- as.matrix(train\_worst[-1])  
train\_worst\_label <- train\_worst$diagnosis  
## Converting labels to 0,1 where "M" is coded at 1  
train\_worst\_label <- as.integer(train\_worst\_label)-1  
train\_worst$diagnosis[1:5]; train\_worst\_label[1:5]

## [1] M M M M M  
## Levels: B M

## [1] 1 1 1 1 1

## Repeat for test dataset  
test\_worst\_data <- as.matrix(test\_worst[-1])  
test\_worst\_label <- test\_worst$diagnosis  
test\_worst\_label <- as.integer(test\_worst\_label)-1  
test\_worst$diagnosis[1:5]; test\_worst\_label[1:5]

## [1] M M B M M  
## Levels: B M

## [1] 1 1 0 1 1

## Formatting data for XGBoost matricies  
worst\_dtrain <- xgb.DMatrix(data = train\_worst\_data, label = train\_worst\_label)  
worst\_dtest <- xgb.DMatrix(data = test\_worst\_data, label = test\_worst\_label)

##### Hyperparameter tuning using random search

### parameters: max\_depth, eta, subsample, colsample\_bytree, and min\_child\_weight  
worst\_low\_err\_list <- list()  
worst\_parameters\_list <- list()  
set.seed(99)  
for(i in 1:3000){  
 params <- list(booster = "gbtree",  
 objective = "binary:logistic",  
 max\_depth = sample(3:25, 1),  
 eta = runif(1, 0.01, 0.3),  
 subsample = runif(1, 0.5, 1),  
 colsample\_bytree = runif(1, 0.5, 1),  
 min\_child\_weight = sample(0:10, 1)  
 )  
 parameters <- as.data.frame(params)  
 worst\_parameters\_list[[i]] <- parameters  
}  
worst\_parameters\_df <- do.call(rbind, worst\_parameters\_list) #df containing random search params  
  
### Fitting 5-fold CV xgboost models based on search parameters   
for (row in 1:nrow(worst\_parameters\_df)){  
 set.seed(99)  
 worst\_tmp\_mdl <- xgb.cv(data = worst\_dtrain,  
 booster = "gbtree",  
 objective = "binary:logistic",  
 nfold = 5,  
 prediction = TRUE,  
 max\_depth = worst\_parameters\_df$max\_depth[row],  
 eta = worst\_parameters\_df$eta[row],  
 subsample = worst\_parameters\_df$subsample[row],  
 colsample\_bytree = worst\_parameters\_df$colsample\_bytree[row],  
 min\_child\_weight = worst\_parameters\_df$min\_child\_weight[row],  
 nrounds = 200,  
 eval\_metric = "error",  
 early\_stopping\_rounds = 20,  
 print\_every\_n = 500,  
 verbose = 0  
 )  
   
   
 #this is the lowest error for the iteration  
 worst\_low\_err <- as.data.frame(1 - min(worst\_tmp\_mdl$evaluation\_log$test\_error\_mean))  
 worst\_low\_err\_list[[row]] <- worst\_low\_err  
}  
  
worst\_low\_err\_df <- do.call(rbind, worst\_low\_err\_list) #accuracies   
worst\_randsearch <- cbind(worst\_low\_err\_df, worst\_parameters\_df) #data frame with everything  
  
###Reformatting the dataframe  
worst\_randsearch <- worst\_randsearch %>%  
 dplyr::rename(val\_acc = '1 - min(worst\_tmp\_mdl$evaluation\_log$test\_error\_mean)') %>%  
 dplyr::arrange(-val\_acc)  
  
###Grabbing just the top model  
worst\_randsearch\_best <- worst\_randsearch[1,]  
  
### Storing best parameters in list  
worst\_best\_params <- list(booster = worst\_randsearch\_best$booster,  
 objective = worst\_randsearch\_best$objective,  
 max\_depth = worst\_randsearch\_best$max\_depth,  
 eta = worst\_randsearch\_best$eta,  
 subsample = worst\_randsearch\_best$subsample,  
 colsample\_bytree = worst\_randsearch\_best$colsample\_bytree,  
 min\_child\_weight = worst\_randsearch\_best$min\_child\_weight)

##### Hyperparameter tuning nround with 5-fold cross validation

set.seed(99)  
worst\_xgbcv <- xgb.cv(params = worst\_best\_params,  
 data = worst\_dtrain,  
 nrounds = 500,  
 nfold = 5,  
 prediction = TRUE,   
 print\_every\_n = 50,  
 early\_stopping\_rounds = 25,  
 eval\_metric = "error",  
 verbose = 0  
 )  
worst\_xgbcv$best\_iteration

##### Model training using best hyperparameters

set.seed(99)  
worst\_best\_xgb <- xgb.train(params = worst\_best\_params,  
 data = worst\_dtrain,  
 nrounds = worst\_xgbcv$best\_iteration,  
 eval\_metric = "error"  
 )  
xgb.save(worst\_best\_xgb, 'final\_xgb\_cancerworst')

## [1] TRUE

cancer\_worst.pred <- predict(worst\_best\_xgb, worst\_dtest)  
cancer\_worst.pred <- factor(ifelse(cancer\_worst.pred> 0.5, 1, 0),  
 labels = c("B", "M"))

##### Model testing and visualizations

### variable importance plot  
worst\_impt\_mtx <- xgb.importance(feature\_names = colnames(test\_worst\_data), model = worst\_best\_xgb)  
xgb.plot.importance(importance\_matrix = worst\_impt\_mtx,  
 xlab = "Variable Importance")

### ROC curve for 5-fold CV random parameter search  
worst\_randsearch\_roc <- roc(response = train\_worst\_label,  
 predictor = worst\_tmp\_mdl$pred,  
 print.auc = TRUE,  
 plot = TRUE)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

### ROC curve for 5-fold CV nround parameter search  
worst\_nround\_roc <- roc(response = train\_worst\_label,  
 predictor = worst\_xgbcv$pred,  
 print.auc = TRUE,  
 plot = TRUE)

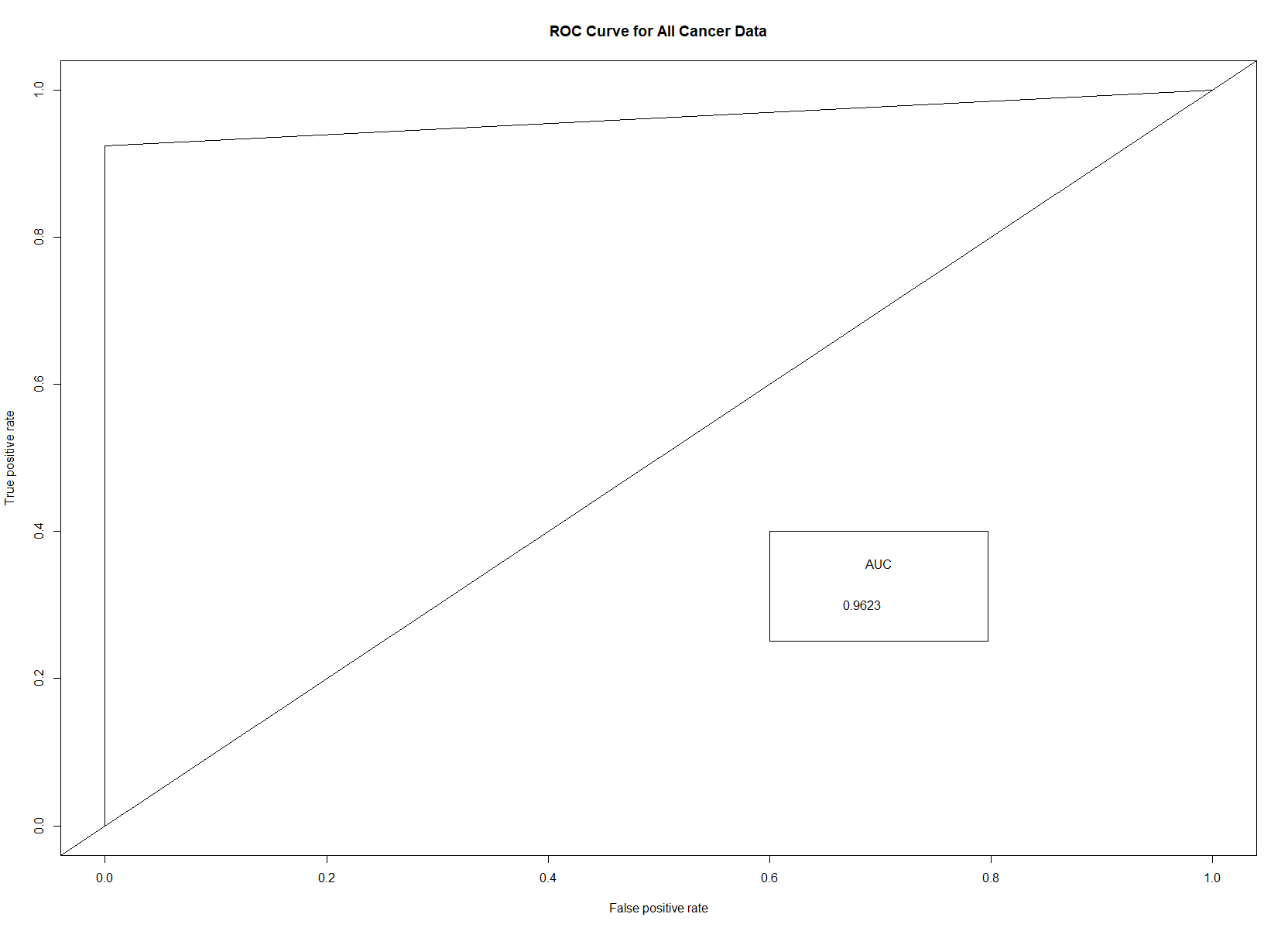
## Setting levels: control = 0, case = 1  
## Setting direction: controls < cases

# Results

## Logistic Regression

##### All cancer data

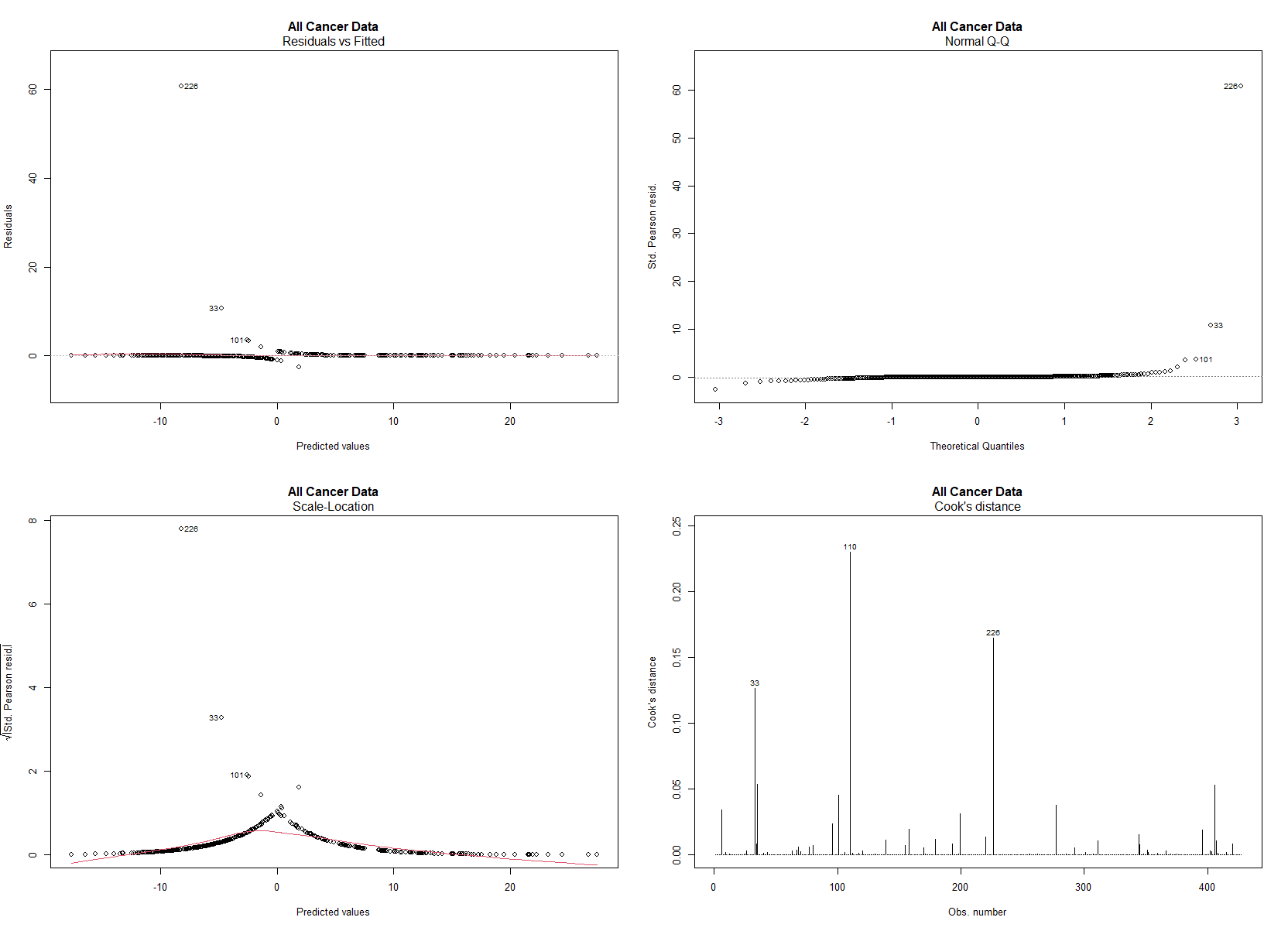
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction B M  
## B 89 0  
## M 4 49  
##   
## Accuracy : 0.9718   
## 95% CI : (0.9294, 0.9923)  
## No Information Rate : 0.6549   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9389   
##   
## Mcnemar's Test P-Value : 0.1336   
##   
## Sensitivity : 1.0000   
## Specificity : 0.9570   
## Pos Pred Value : 0.9245   
## Neg Pred Value : 1.0000   
## Precision : 0.9245   
## Recall : 1.0000   
## F1 : 0.9608   
## Prevalence : 0.3451   
## Detection Rate : 0.3451   
## Detection Prevalence : 0.3732   
## Balanced Accuracy : 0.9785   
##   
## 'Positive' Class : M   
##



##   
## Call:  
## glm(formula = diagnosis ~ texture\_mean + concavity\_mean + concave\_points\_mean +   
## symmetry\_mean + fractal\_dimension\_mean + perimeter\_worst +   
## smoothness\_worst + concave\_points\_worst + symmetry\_worst,   
## family = "binomial", data = train\_all)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.0045 -0.0902 -0.0124 0.0045 4.0534   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -24.83421 7.50130 -3.311 0.000931 \*\*\*  
## texture\_mean 0.38508 0.08764 4.394 1.11e-05 \*\*\*  
## concavity\_mean 5.62729 14.58252 0.386 0.699576   
## concave\_points\_mean 58.21992 36.38346 1.600 0.109560   
## symmetry\_mean 0.79034 20.70848 0.038 0.969556   
## fractal\_dimension\_mean -215.73218 93.21534 -2.314 0.020649 \*   
## perimeter\_worst 0.13176 0.03570 3.690 0.000224 \*\*\*  
## smoothness\_worst 56.25842 24.20205 2.325 0.020097 \*   
## concave\_points\_worst 4.89131 15.50494 0.315 0.752406   
## symmetry\_worst 14.72512 8.02238 1.836 0.066431 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 563.813 on 426 degrees of freedom  
## Residual deviance: 75.096 on 417 degrees of freedom  
## AIC: 95.096  
##   
## Number of Fisher Scoring iterations: 9

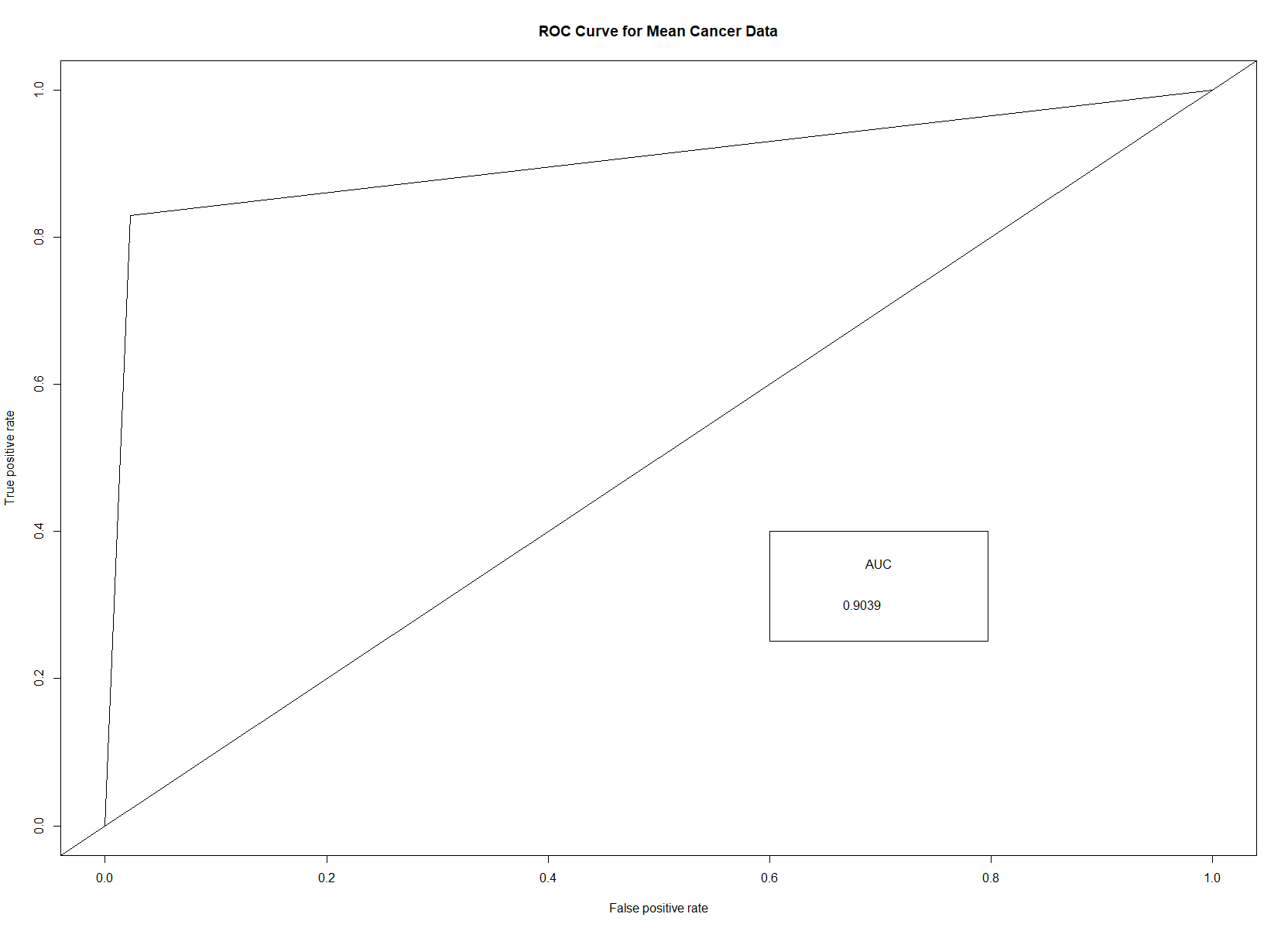
## fitting null model for pseudo-r2

## McFadden   
## 0.8668077



##### Mean cancer data

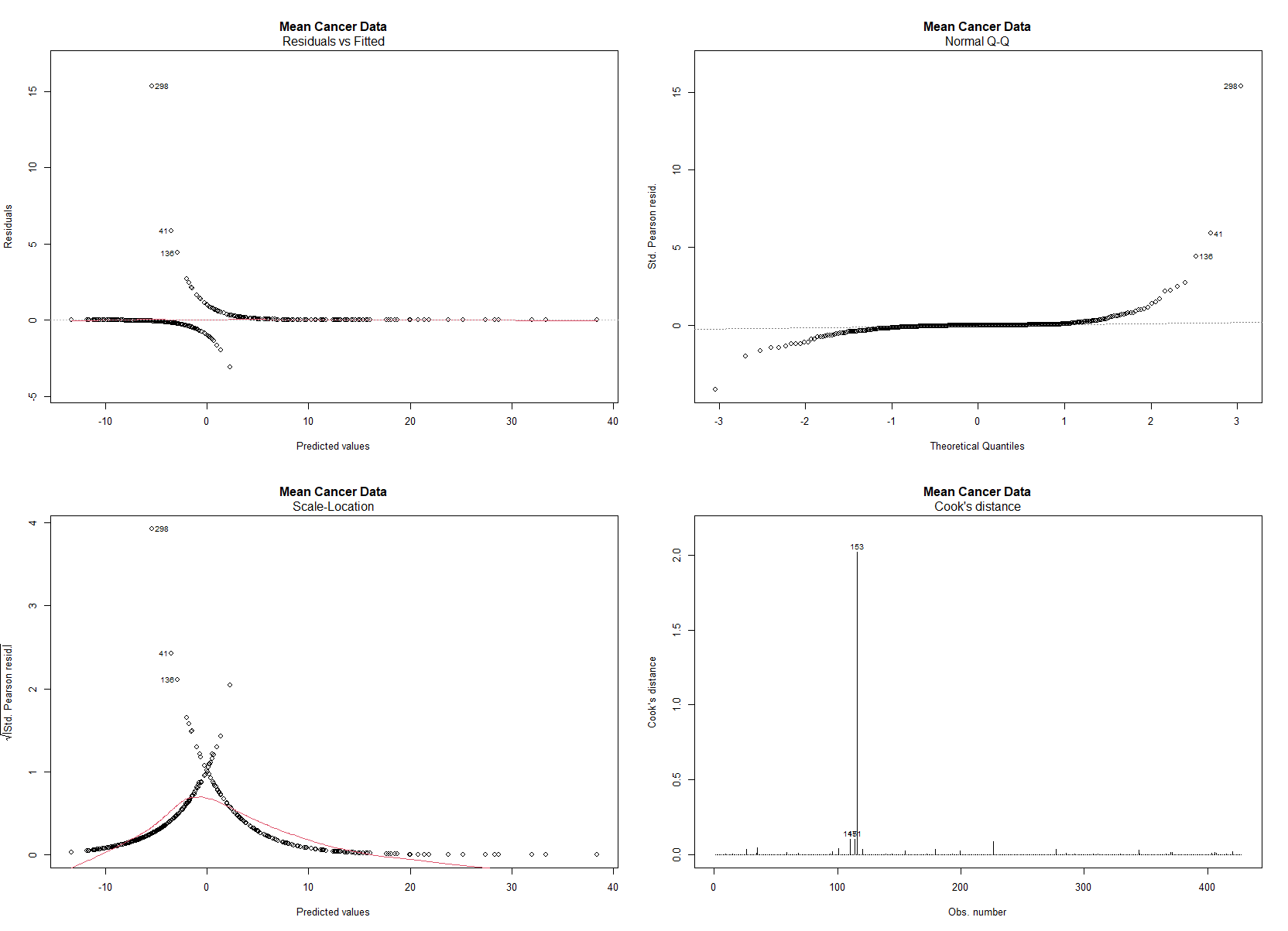
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction B M  
## B 87 2  
## M 9 44  
##   
## Accuracy : 0.9225   
## 95% CI : (0.8656, 0.9607)  
## No Information Rate : 0.6761   
## P-Value [Acc > NIR] : 2.116e-12   
##   
## Kappa : 0.8299   
##   
## Mcnemar's Test P-Value : 0.07044   
##   
## Sensitivity : 0.9565   
## Specificity : 0.9062   
## Pos Pred Value : 0.8302   
## Neg Pred Value : 0.9775   
## Precision : 0.8302   
## Recall : 0.9565   
## F1 : 0.8889   
## Prevalence : 0.3239   
## Detection Rate : 0.3099   
## Detection Prevalence : 0.3732   
## Balanced Accuracy : 0.9314   
##   
## 'Positive' Class : M   
##



##   
## Call:  
## glm(formula = diagnosis ~ texture\_mean + area\_mean + smoothness\_mean +   
## compactness\_mean + concavity\_mean + symmetry\_mean, family = "binomial",   
## data = train\_mean)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.1678 -0.1344 -0.0271 0.0051 3.3066   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -35.33487 5.47241 -6.457 1.07e-10 \*\*\*  
## texture\_mean 0.41225 0.07396 5.574 2.49e-08 \*\*\*  
## area\_mean 0.01736 0.00264 6.576 4.83e-11 \*\*\*  
## smoothness\_mean 122.34322 32.54862 3.759 0.000171 \*\*\*  
## compactness\_mean -22.80941 11.84340 -1.926 0.054115 .   
## concavity\_mean 27.63625 8.02356 3.444 0.000572 \*\*\*  
## symmetry\_mean 20.51731 12.36067 1.660 0.096937 .   
## ---  
## 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: 109.94 on 420 degrees of freedom  
## AIC: 123.94  
##   
## Number of Fisher Scoring iterations: 8

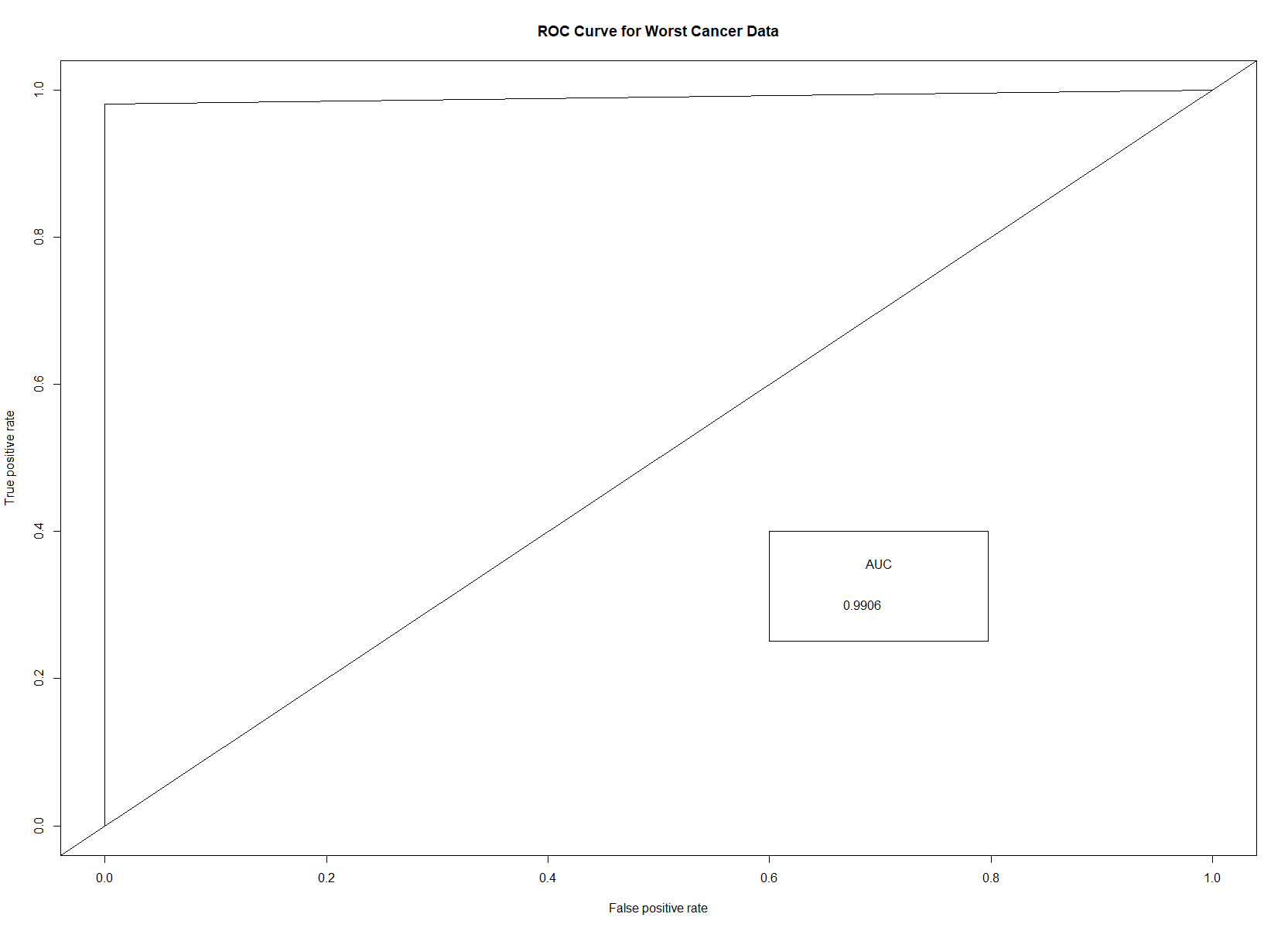
## fitting null model for pseudo-r2

## McFadden   
## 0.8050058



##### Worst cacner data

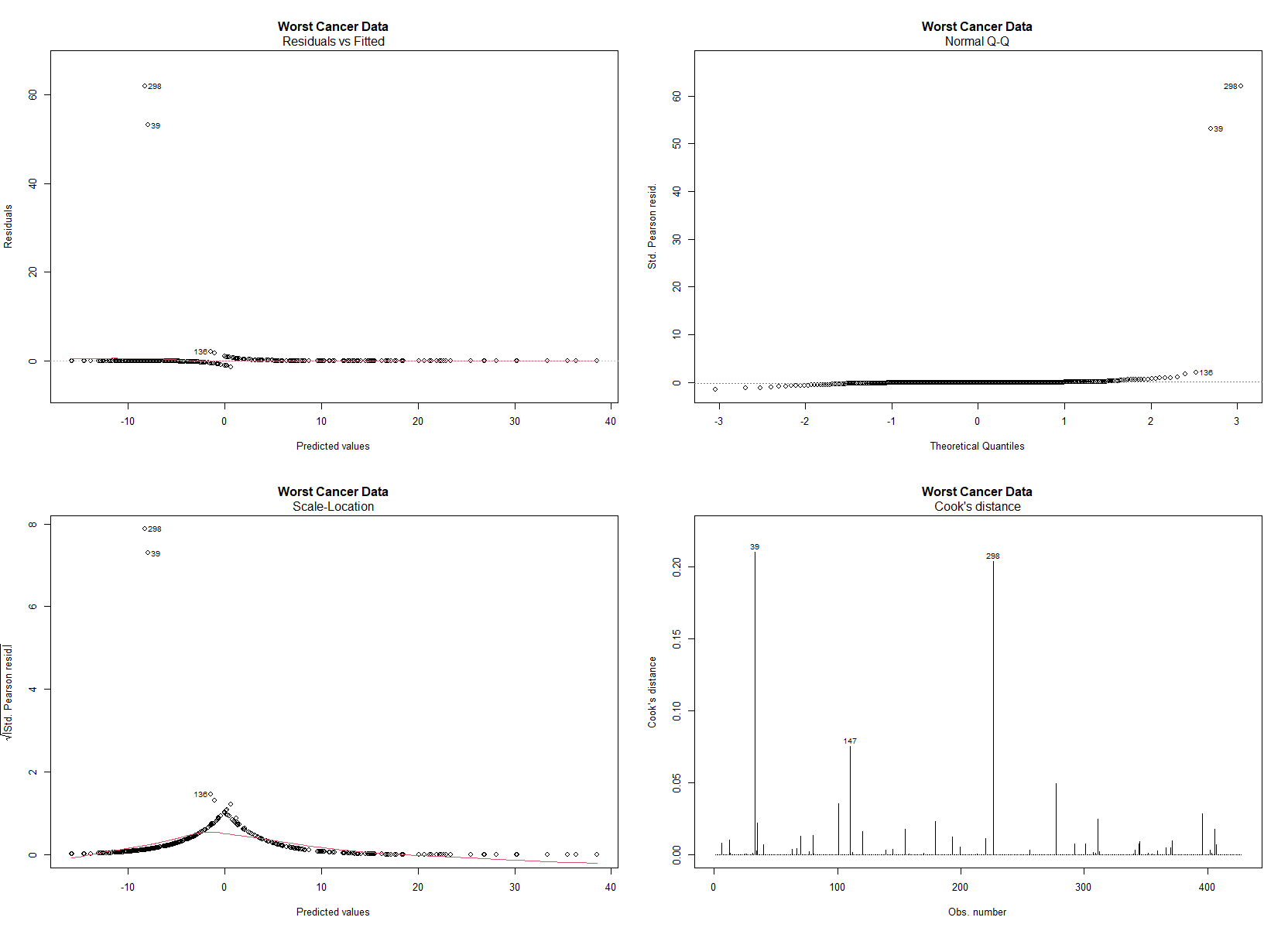
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction B M  
## B 89 0  
## M 1 52  
##   
## Accuracy : 0.993   
## 95% CI : (0.9614, 0.9998)  
## No Information Rate : 0.6338   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9849   
##   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 1.0000   
## Specificity : 0.9889   
## Pos Pred Value : 0.9811   
## Neg Pred Value : 1.0000   
## Precision : 0.9811   
## Recall : 1.0000   
## F1 : 0.9905   
## Prevalence : 0.3662   
## Detection Rate : 0.3662   
## Detection Prevalence : 0.3732   
## Balanced Accuracy : 0.9944   
##   
## 'Positive' Class : M   
##



##   
## Call:  
## glm(formula = diagnosis ~ texture\_worst + area\_worst + smoothness\_worst +   
## concavity\_worst + concave\_points\_worst + symmetry\_worst +   
## fractal\_dimension\_worst, family = "binomial", data = train\_worst)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.4564 -0.0837 -0.0116 0.0018 4.0632   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -26.942363 4.919300 -5.477 4.33e-08 \*\*\*  
## texture\_worst 0.267815 0.063886 4.192 2.76e-05 \*\*\*  
## area\_worst 0.012137 0.002427 5.001 5.71e-07 \*\*\*  
## smoothness\_worst 51.235210 23.666386 2.165 0.0304 \*   
## concavity\_worst 4.444385 4.073509 1.091 0.2753   
## concave\_points\_worst 29.575980 14.443473 2.048 0.0406 \*   
## symmetry\_worst 10.518961 5.986965 1.757 0.0789 .   
## fractal\_dimension\_worst -66.389888 33.621091 -1.975 0.0483 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 563.813 on 426 degrees of freedom  
## Residual deviance: 72.707 on 419 degrees of freedom  
## AIC: 88.707  
##   
## Number of Fisher Scoring iterations: 9

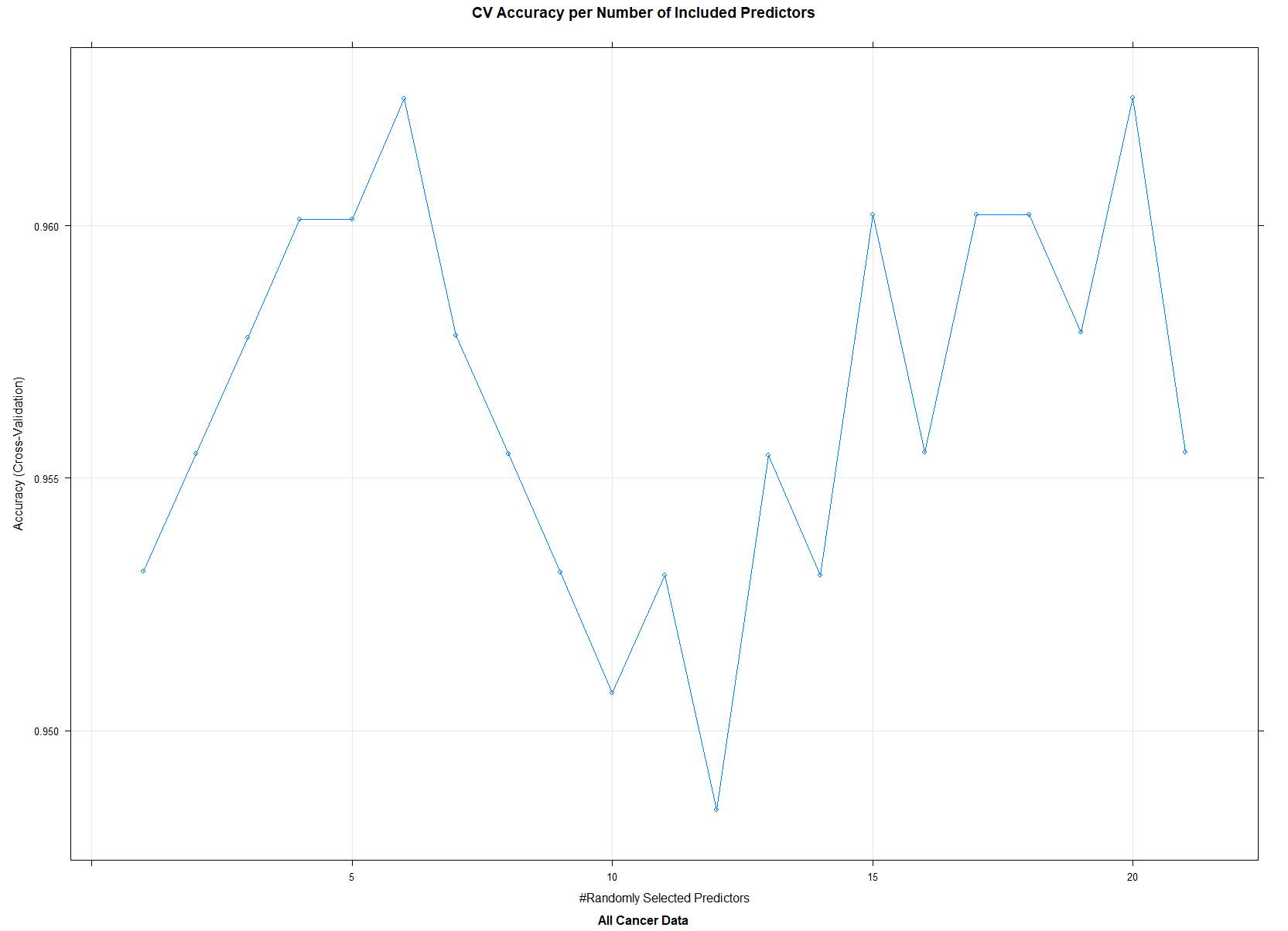
## fitting null model for pseudo-r2

## McFadden   
## 0.8710438

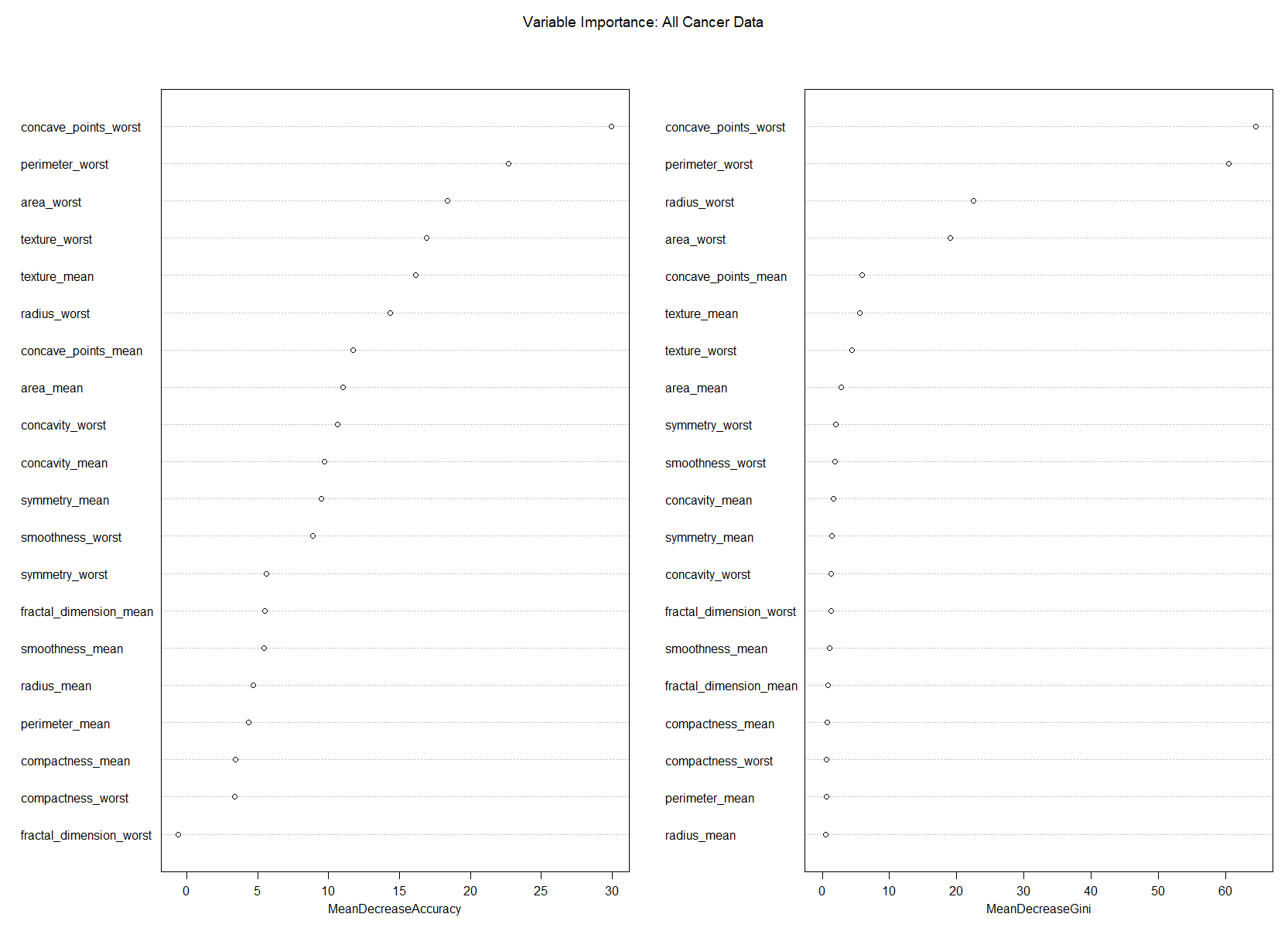


## Random Forest

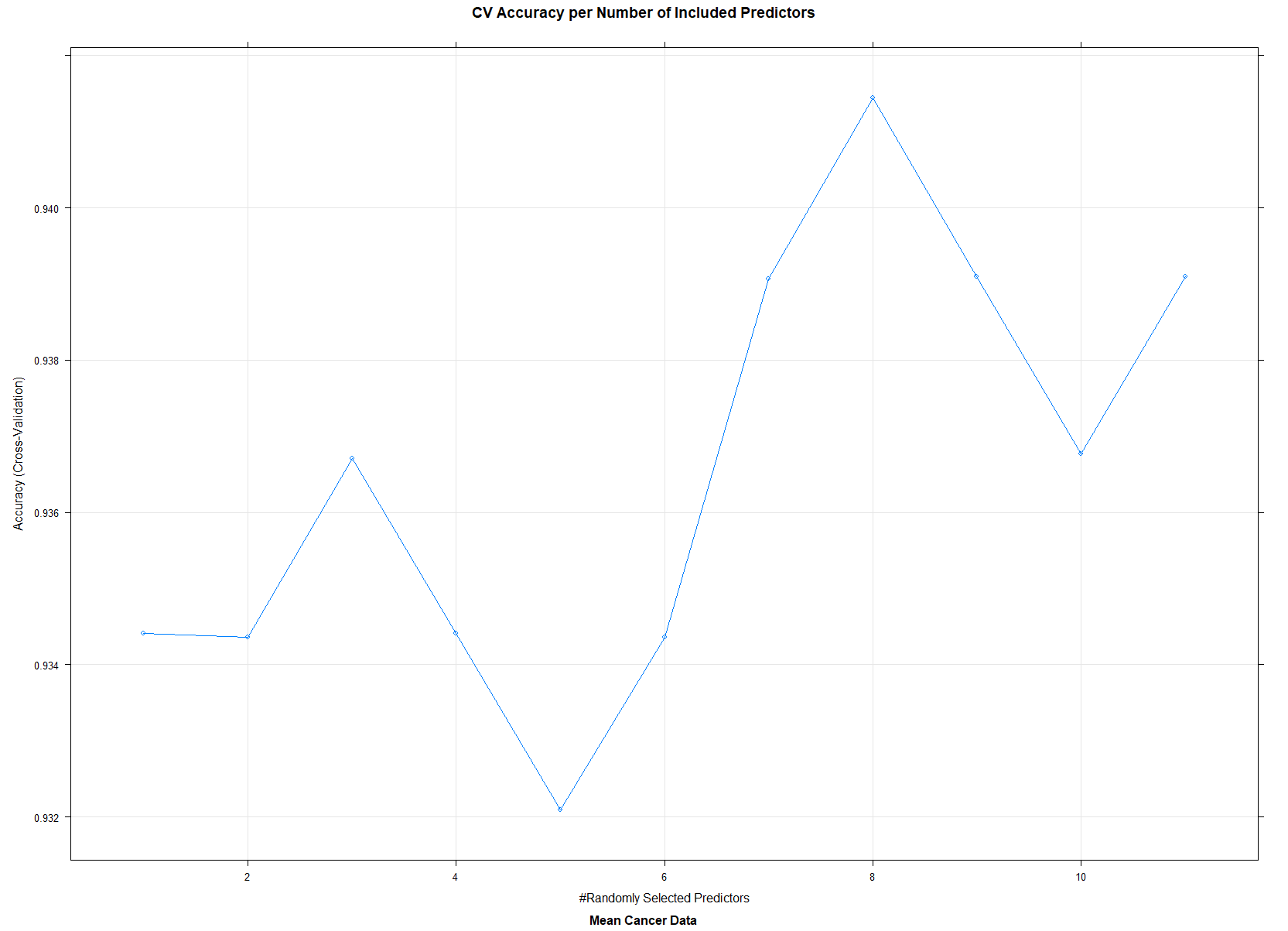
##### All cancer data



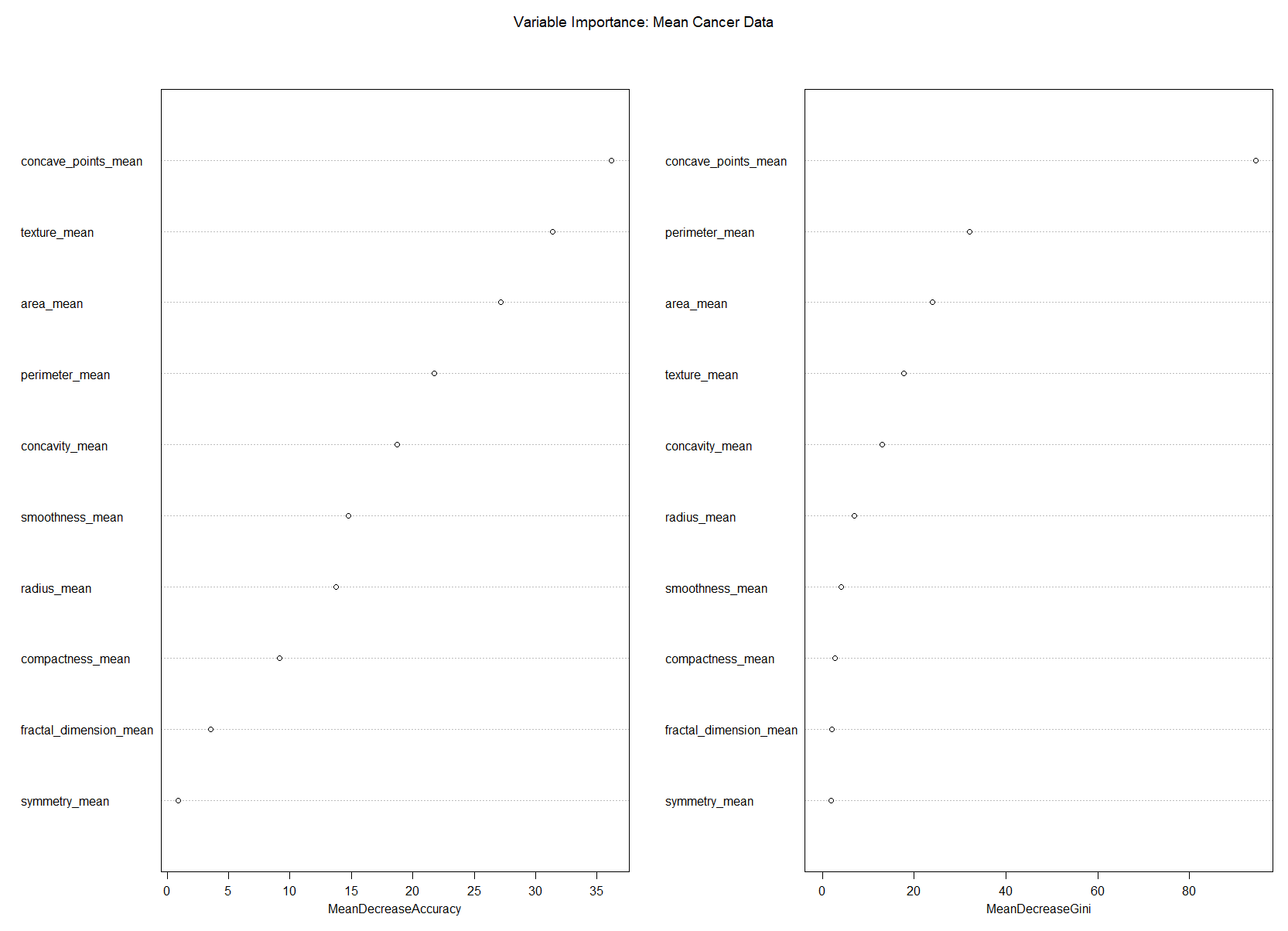
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction B M  
## B 88 2  
## M 1 51  
##   
## Accuracy : 0.9789   
## 95% CI : (0.9395, 0.9956)  
## No Information Rate : 0.6268   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9547   
##   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 0.9623   
## Specificity : 0.9888   
## Pos Pred Value : 0.9808   
## Neg Pred Value : 0.9778   
## Precision : 0.9808   
## Recall : 0.9623   
## F1 : 0.9714   
## Prevalence : 0.3732   
## Detection Rate : 0.3592   
## Detection Prevalence : 0.3662   
## Balanced Accuracy : 0.9755   
##   
## 'Positive' Class : M   
##



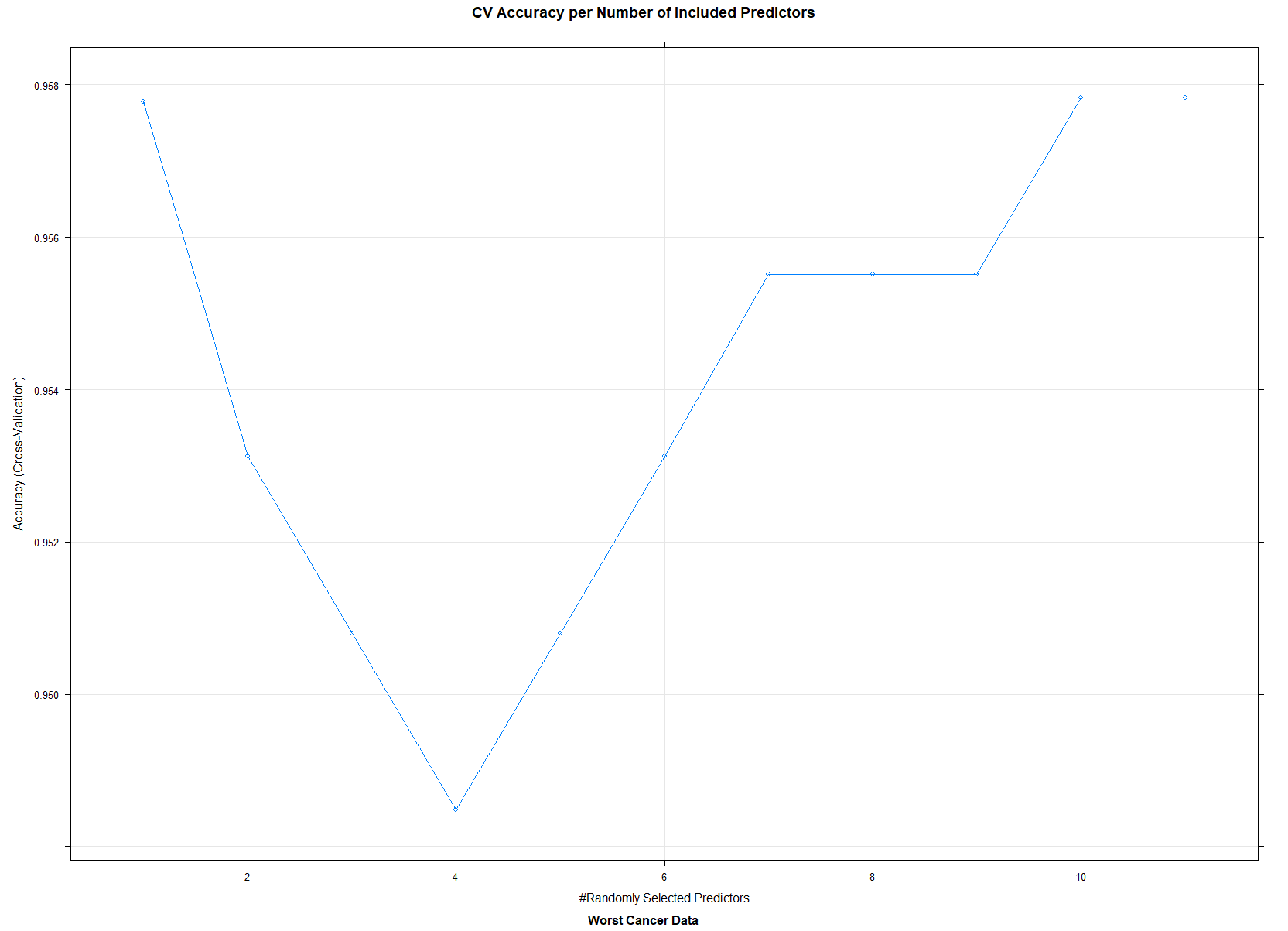
##### Mean cancer data



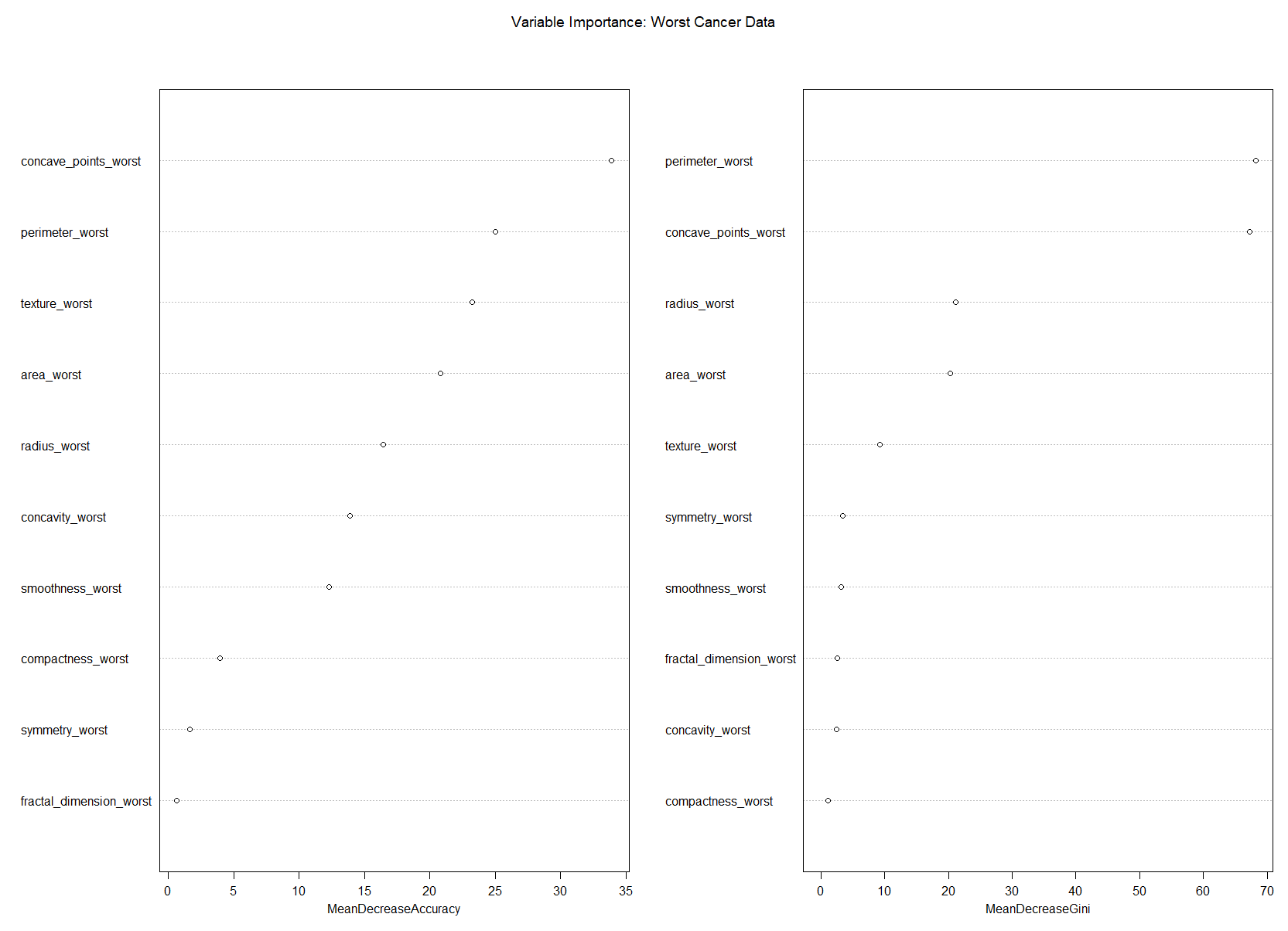
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction B M  
## B 88 7  
## M 1 46  
##   
## Accuracy : 0.9437   
## 95% CI : (0.892, 0.9754)  
## No Information Rate : 0.6268   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.8768   
##   
## Mcnemar's Test P-Value : 0.0771   
##   
## Sensitivity : 0.8679   
## Specificity : 0.9888   
## Pos Pred Value : 0.9787   
## Neg Pred Value : 0.9263   
## Precision : 0.9787   
## Recall : 0.8679   
## F1 : 0.9200   
## Prevalence : 0.3732   
## Detection Rate : 0.3239   
## Detection Prevalence : 0.3310   
## Balanced Accuracy : 0.9283   
##   
## 'Positive' Class : M   
##



##### Worst cacner data



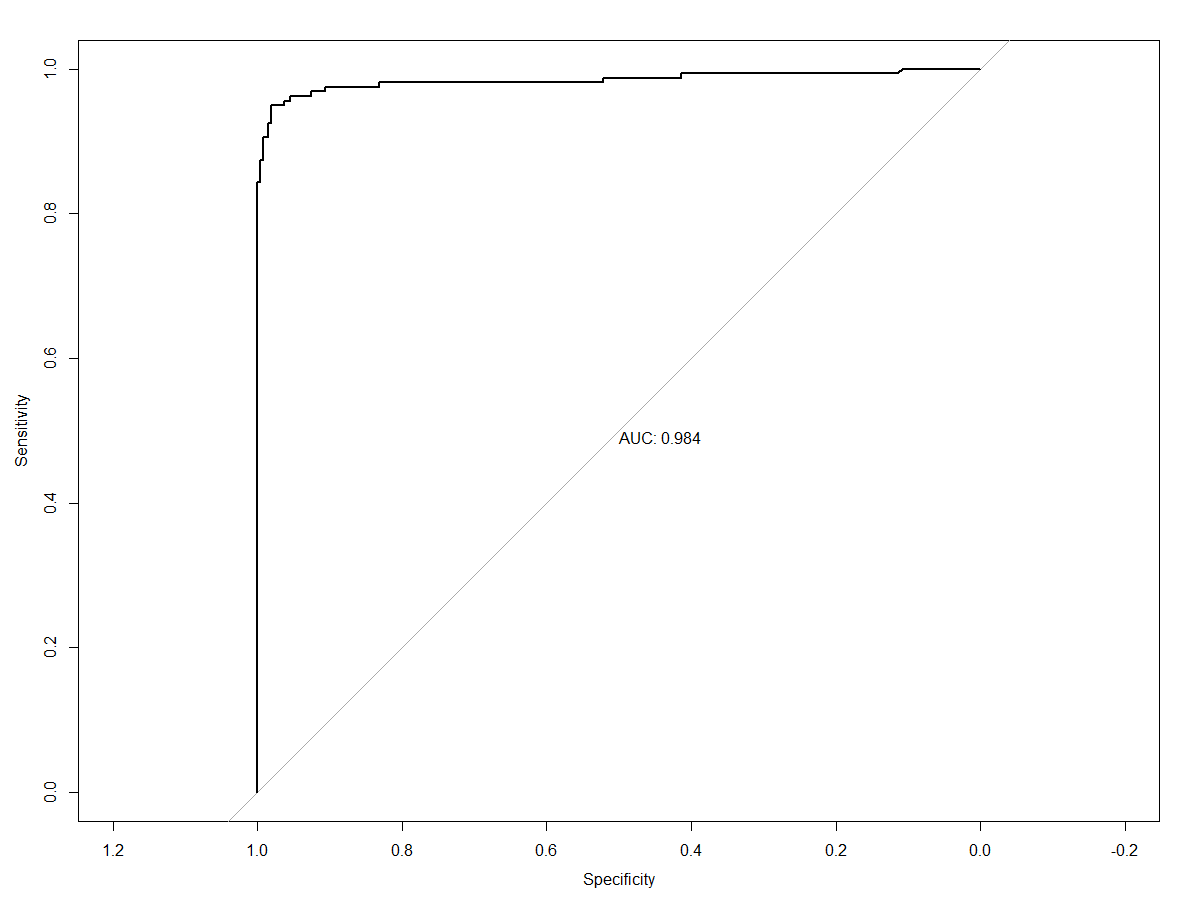
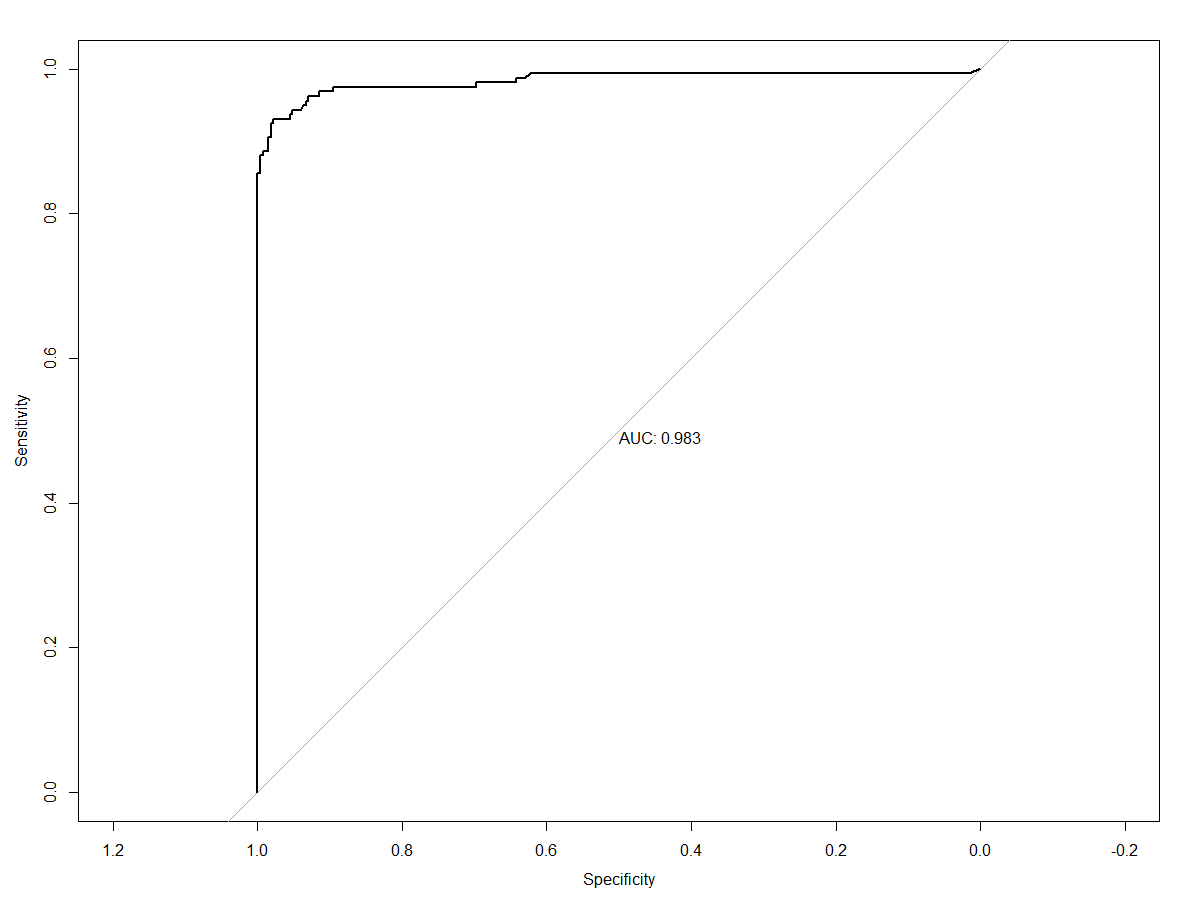
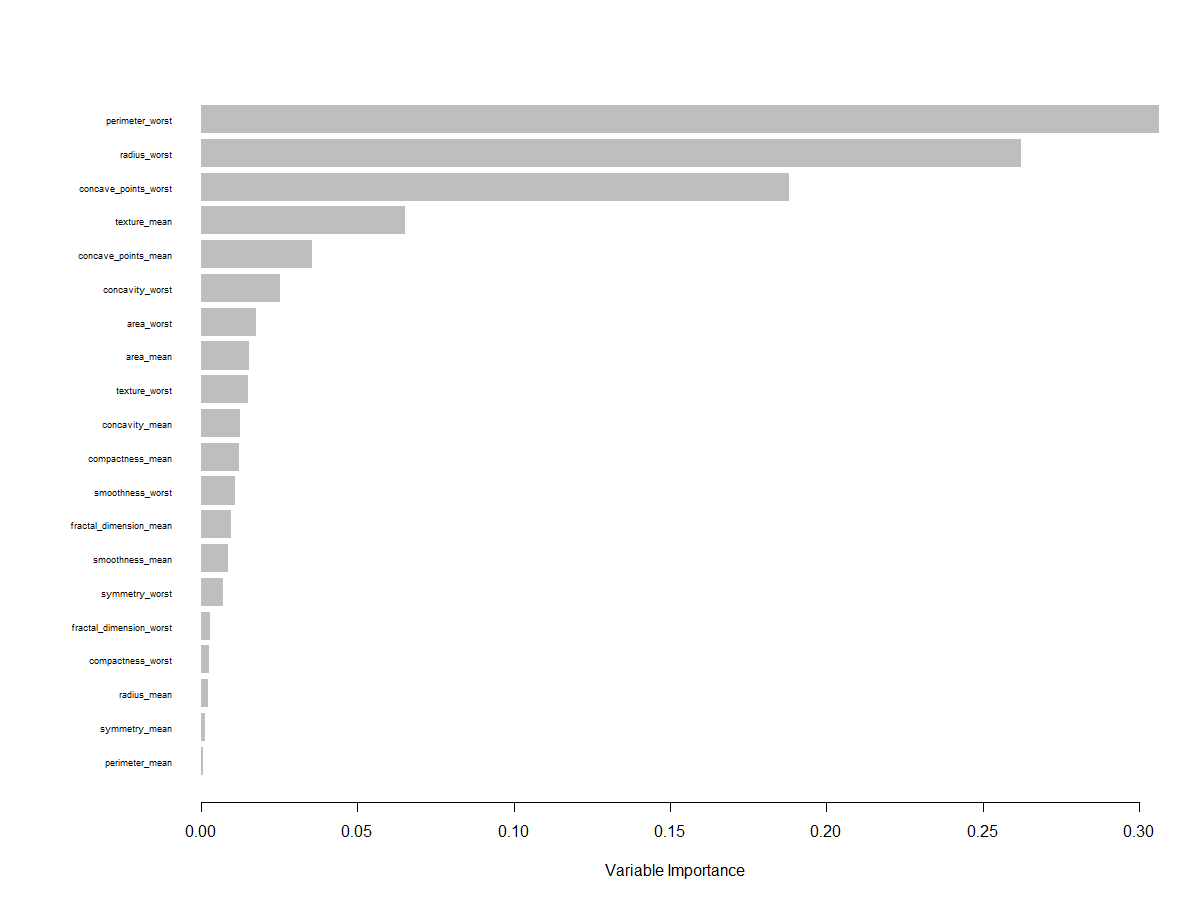
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction B M  
## B 89 2  
## M 0 51  
##   
## Accuracy : 0.9859   
## 95% CI : (0.95, 0.9983)  
## No Information Rate : 0.6268   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9697   
##   
## Mcnemar's Test P-Value : 0.4795   
##   
## Sensitivity : 0.9623   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 0.9780   
## Precision : 1.0000   
## Recall : 0.9623   
## F1 : 0.9808   
## Prevalence : 0.3732   
## Detection Rate : 0.3592   
## Detection Prevalence : 0.3592   
## Balanced Accuracy : 0.9811   
##   
## 'Positive' Class : M   
##



## XGBoost

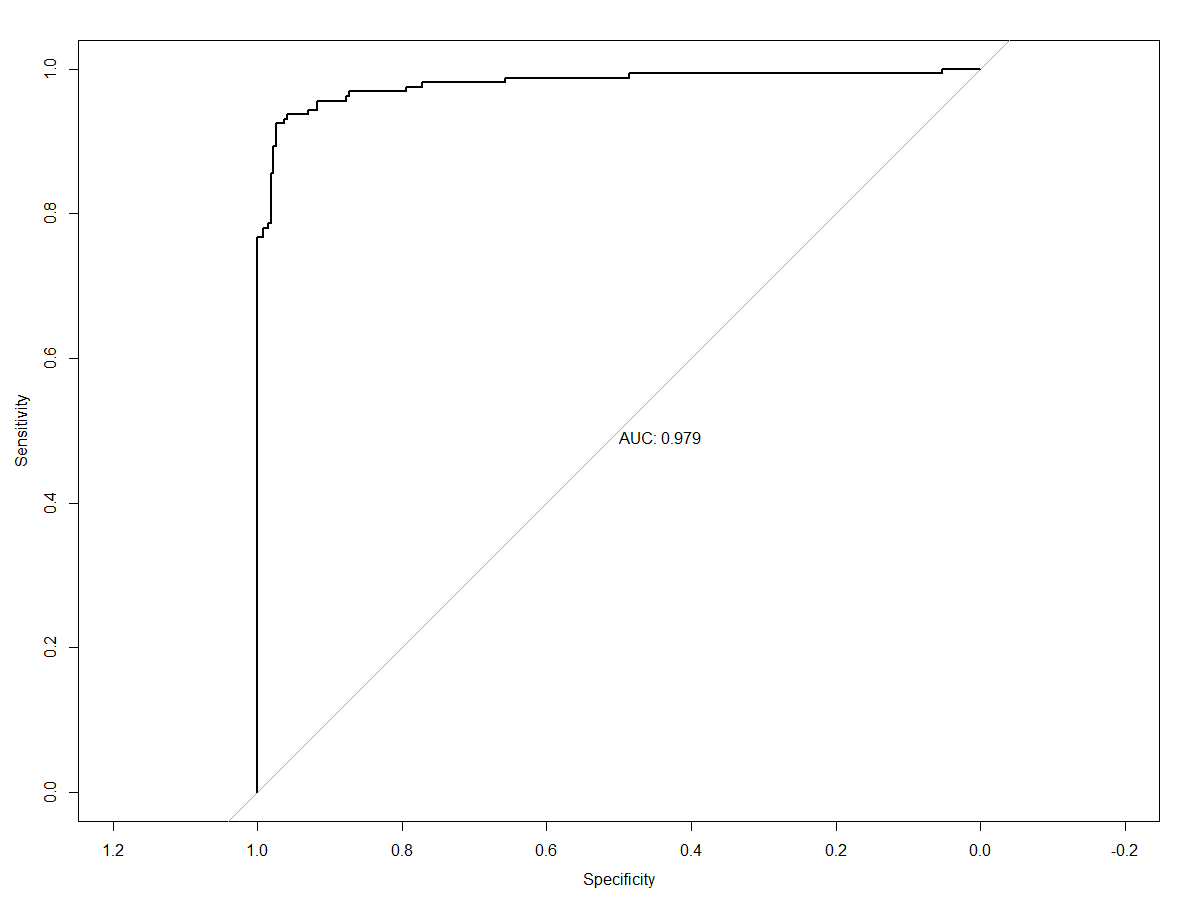
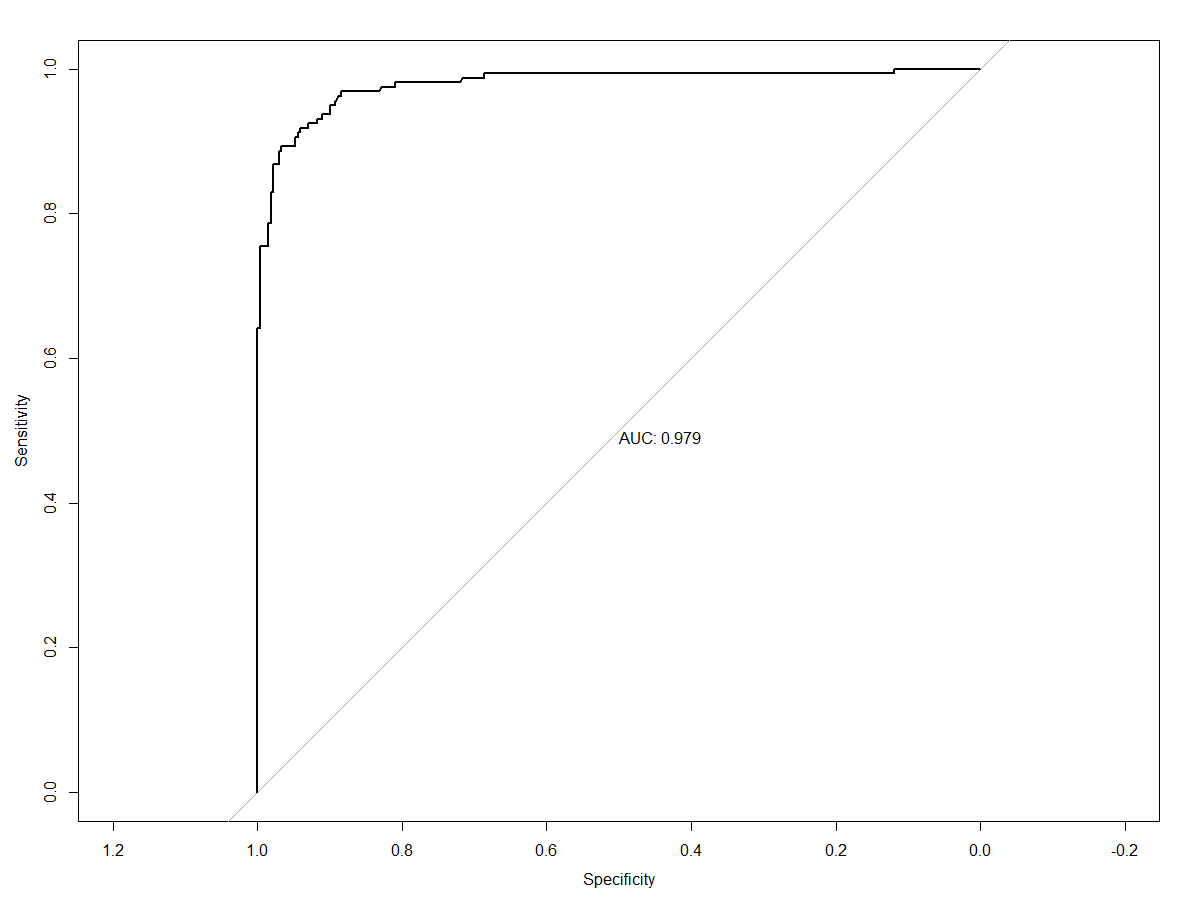
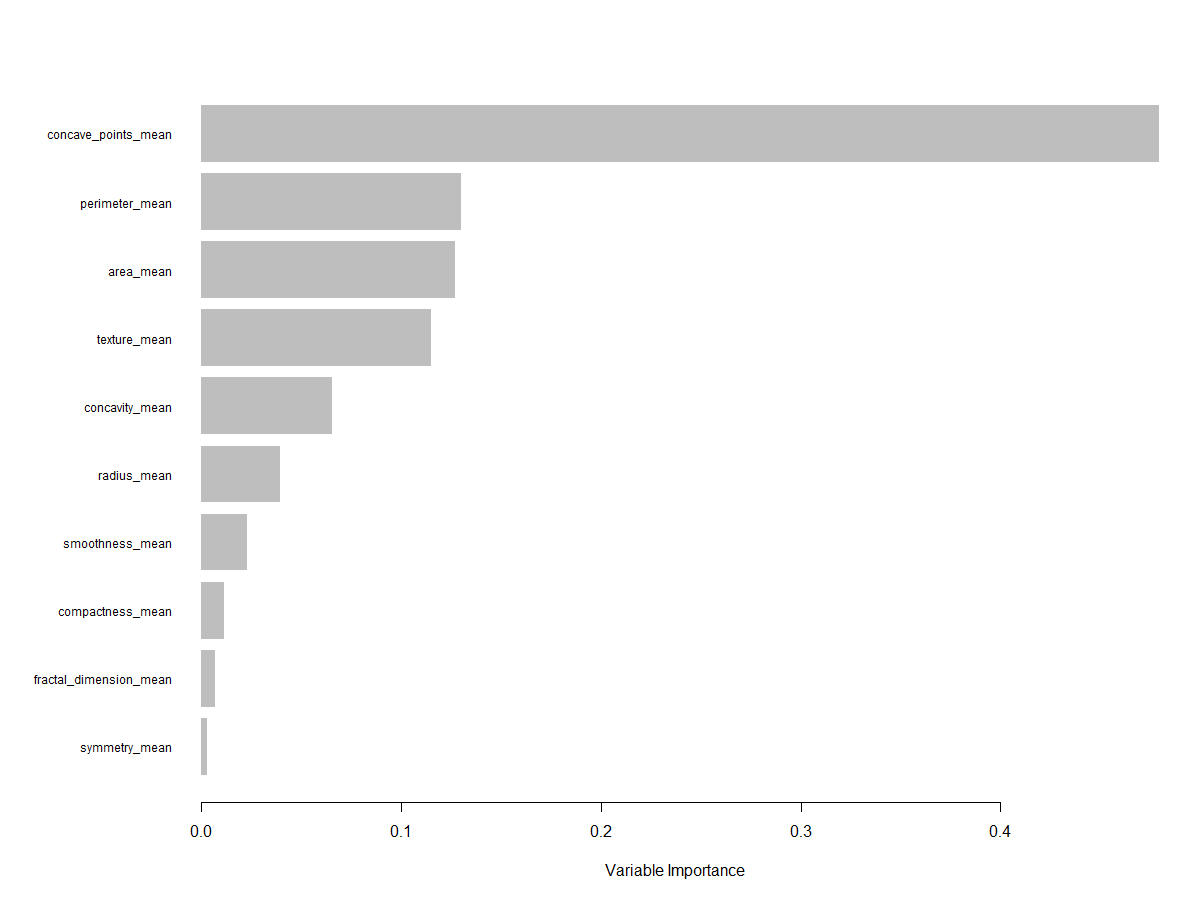
##### All cancer data

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction B M  
## B 89 3  
## M 0 50  
##   
## Accuracy : 0.9789   
## 95% CI : (0.9395, 0.9956)  
## No Information Rate : 0.6268   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9543   
##   
## Mcnemar's Test P-Value : 0.2482   
##   
## Sensitivity : 0.9434   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 0.9674   
## Precision : 1.0000   
## Recall : 0.9434   
## F1 : 0.9709   
## Prevalence : 0.3732   
## Detection Rate : 0.3521   
## Detection Prevalence : 0.3521   
## Balanced Accuracy : 0.9717   
##   
## 'Positive' Class : M   
##



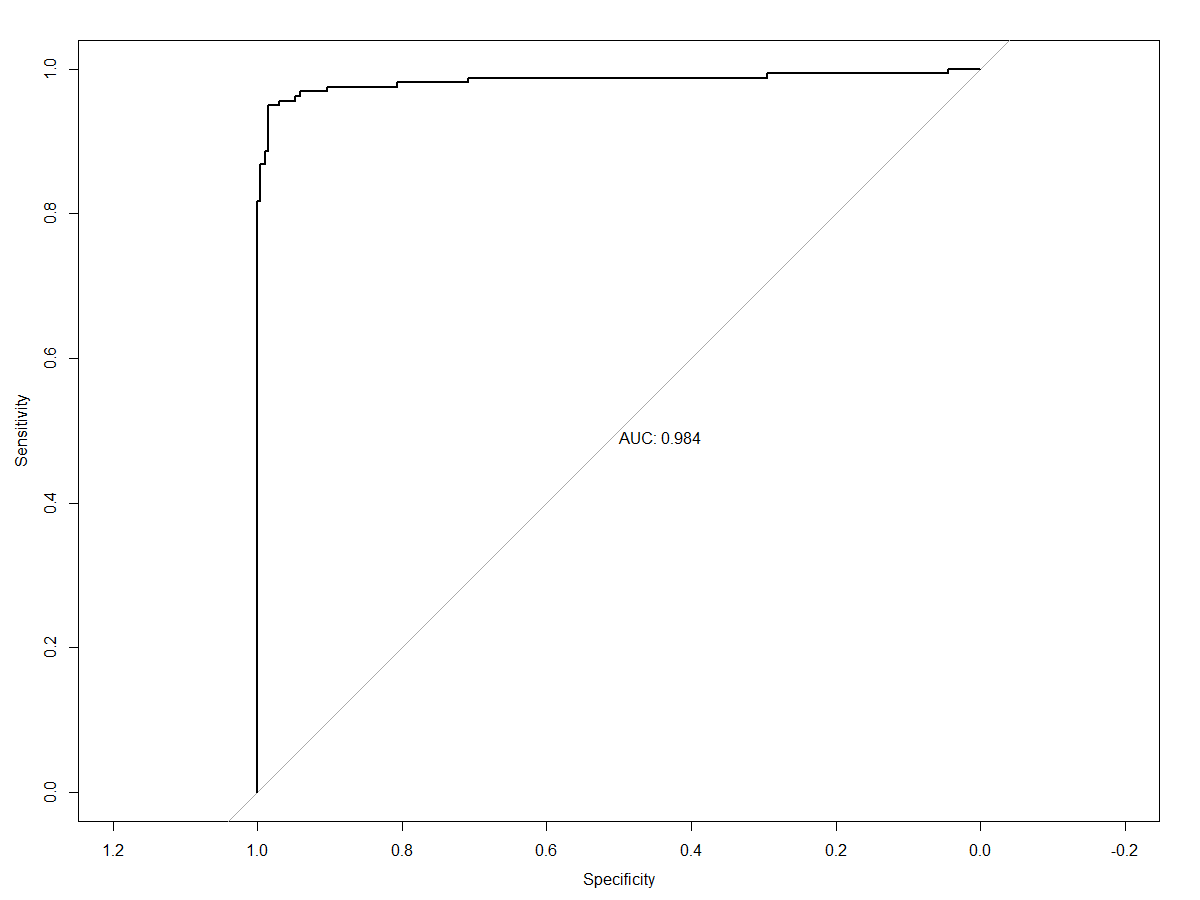
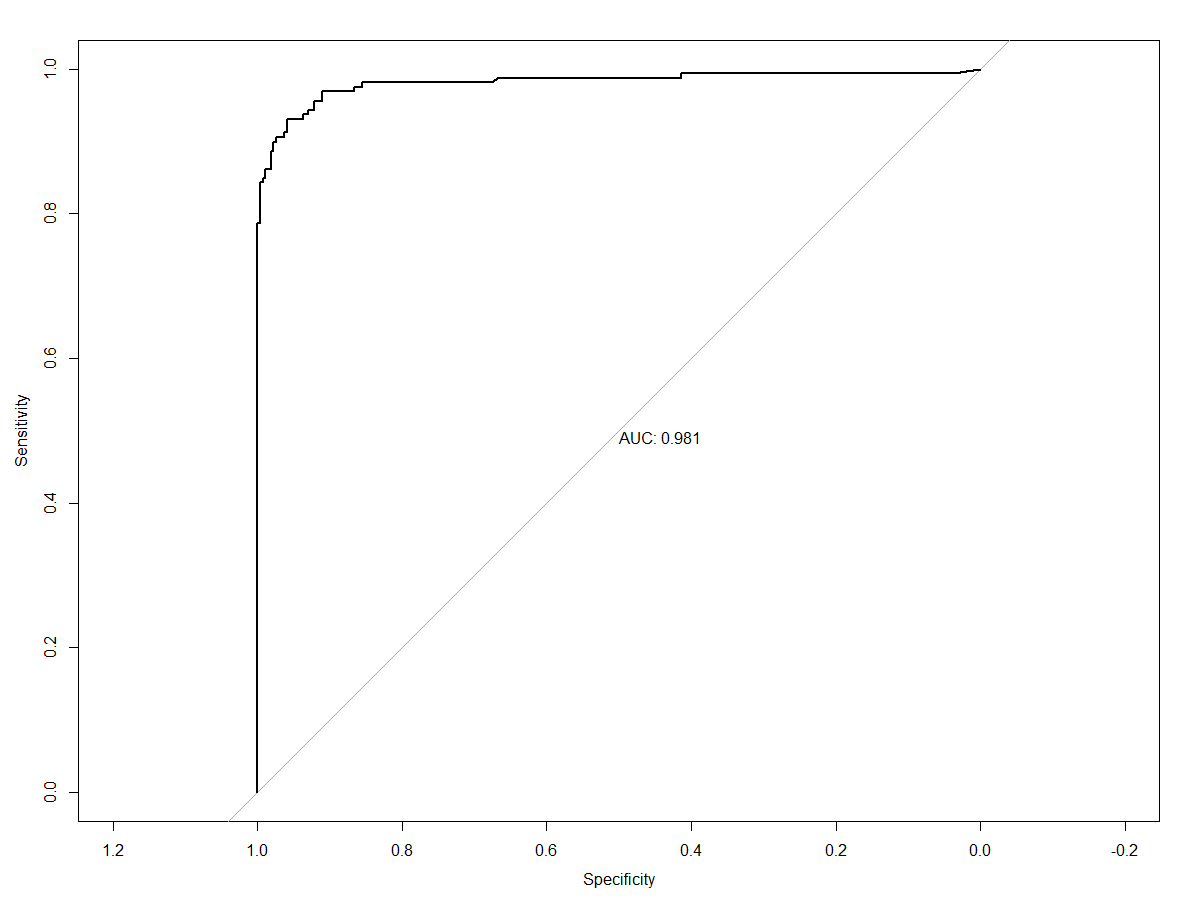
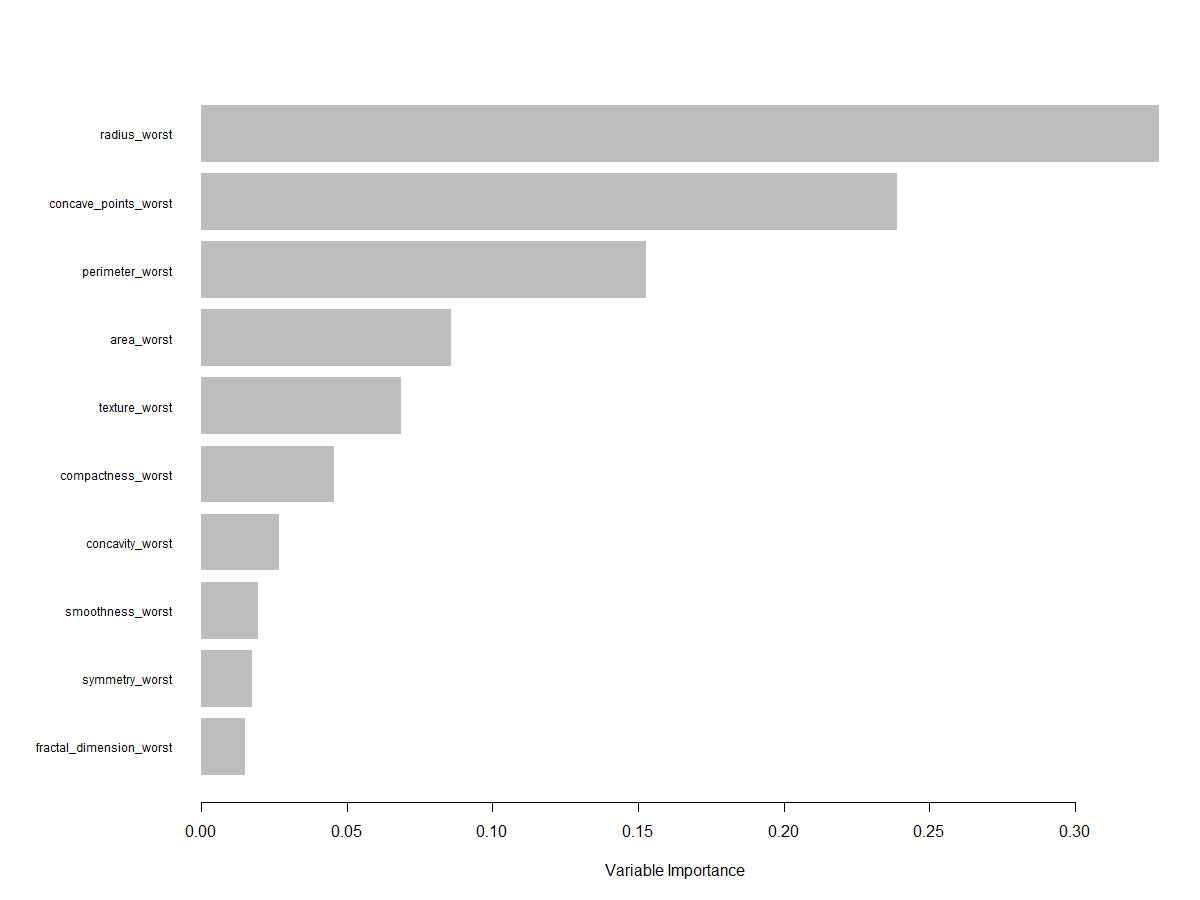
##### Mean cacner data

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction B M  
## B 88 6  
## M 1 47  
##   
## Accuracy : 0.9507   
## 95% CI : (0.9011, 0.98)  
## No Information Rate : 0.6268   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.8926   
##   
## Mcnemar's Test P-Value : 0.1306   
##   
## Sensitivity : 0.8868   
## Specificity : 0.9888   
## Pos Pred Value : 0.9792   
## Neg Pred Value : 0.9362   
## Precision : 0.9792   
## Recall : 0.8868   
## F1 : 0.9307   
## Prevalence : 0.3732   
## Detection Rate : 0.3310   
## Detection Prevalence : 0.3380   
## Balanced Accuracy : 0.9378   
##   
## 'Positive' Class : M   
##



##### Worst cancer data

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction B M  
## B 89 2  
## M 0 51  
##   
## Accuracy : 0.9859   
## 95% CI : (0.95, 0.9983)  
## No Information Rate : 0.6268   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9697   
##   
## Mcnemar's Test P-Value : 0.4795   
##   
## Sensitivity : 0.9623   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 0.9780   
## Precision : 1.0000   
## Recall : 0.9623   
## F1 : 0.9808   
## Prevalence : 0.3732   
## Detection Rate : 0.3592   
## Detection Prevalence : 0.3592   
## Balanced Accuracy : 0.9811   
##   
## 'Positive' Class : M   
##



# Discussion

Since the main focus of this project was to identify the best model that could identify malignant breast cancer tumors, accuracy scores and false negative rates were emphasized. False negatives are a type of misclassification when the predicted value is negative even though the true value is positive. Although overall model accuracy is important for cancer diagnoses, incorrectly identifying a cancerous cell as benign can lead to more harm than incorrectly identifying a benign cell as cancerous.

## Logistic Regression

For logistic regression the model with the highest accuracy score *(0.993)* and lowest false negative rate *(Sensitivity = 1.000)* was found when only looking at worst mean cancer data. The confusion matrix showed zero false negatives and one false positive. Combined with an *AUC = 0.991*, the model performed very well at classification. The model summary showed that all covariates except worst mean cell concavity and and worst mean cell symmetry were significant at an α = 0.05 level. Odd ratios for all covariates in the model could be calculated by exponentiating the coefficient estimates; for example, the odd ratio for the worst mean cell texture is 1.307105. Meaning that holding all other covariates constant, each additional increase in the standard deviation of grey-scale image values of the worst mean cells corresponded to a 30.7% increase in the odds of classifying the sample as malignant. A McFadden R2 of 0.867 indicated a good overall measure of model fit. The model diagnostics indicated that there were some outlier issues that could’ve influenced model performance.

## Random Forest

For random forests, the model with the highest accuracy score *(0.986)* was found when only looking at worst mean cancer data. This model had two false negatives *(Sensitivity = 0.962)* and zero false positives *(Specificity = 1)*. When hyperparameter tuning the model, it was found that including all ten covariates in each spit in the decision trees led to the highest training model accuracy. A variable importance plot indicated that the top three covariates that influenced model performance were the number of worst mean concave points, worst mean perimeter, and worst mean cell texture in that order.

## XGBoost

For the XGBoost algorithm, the model with the highest accuracy score *(0.986)* and false negative rate *(Sensitivity = 0.962)* was found when only looking at the worst mean cancer data. This model had 2 false negatives and 0 false positives *(Specificity = 1.00)*. A variance importance plot showed that the top three covariates that influenced the training model performance were cell radii, the number of concave points on the cell perimeters, and the perimeters themselves. ROC curves of the 5-fold cross validated hyperparameter searches had high AUC values *(AUC = 0.981, AUC = 0.984)* indicated that the training model performed well at classification.

## Cross Model Comparison

Overall, the model that best classified cancer cells as benign or malignant was a logistic regression using different characteristics of worst average cell data. The XGBoost algorithm performed competitively, but had a lower sensitivity score than logistic regression. The results of logistic regression were easier to interpret compared to the results of XGBoost. An additional benefit of the logistic regression was that inferences on the covariates could also be made along side with the overall accuracy of the model.

# Limitations and Future Work

These models didn’t include a validation set during the splitting process. A validation set would provide another subset of data to test the trained models. Although subset selection processes were used, feature engineering was not. All models used started with their full sets of covariates during model construction. In the future better domain knowledge could be used to assess which features would be most appropriate for the project problem. Model diagnostics of the logistic regression models indicated an outlier problem. There were at least two influential datum that impacted the assumptions of linear models. It should be determined if these points could be removed in the future, and see if their removal changes the results of logistic regression models. Although random forests and XGBoost should be more robust to outliers, all other models should be ran again with the new dataset to test for improvements.

# References

<https://medium.com/@taniyaghosh29/machine-learning-algorithms-what-are-the-differences-9b71df4f248f>

<https://medium.com/@nischitasadananda/the-battle-between-logistic-regression-random-forest-classifier-xg-boost-and-support-vector-46d773c70f41>

<https://archive-beta.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+diagnostic>

<https://pages.cs.wisc.edu/~olvi/uwmp/cancer.html#diag>

<https://courses.lumenlearning.com/introstats1/chapter/introduction-to-logistic-regression/>

<https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/>

<https://xgboost.readthedocs.io/en/stable/tutorials/model.html>

<https://cran.r-project.org/web/packages/xgboost/xgboost.pdf>

<https://towardsdatascience.com/getting-to-an-hyperparameter-tuned-xgboost-model-in-no-time-a9560f8eb54b>