Breast Cancer Diagnosis

Tamia van Geloven

Introduction

The goal of this project is to create a machine learning algorithm that predicts whether a breast tumor is malignant or benign. A malignant tumor is cancerous, a benign tumor is not cancerous. Benign tumors can become malignant over time. Doctors and medical health professionals use a variety of tests and measures to determine is a tumor is malignant. For the purposes of this project I am using numeric variables that include the size, texture, compactness, concavity, and intricacy of the tissue of the tumor.

I began by importing the data from Kaggle. The data set is the Breast Cancer Wisconsin (Diagnostic) Data Set. I then inspected and cleaned the data and assigned the cleaned data set to breast_cancer. Using the data set breast_cancer, I created a boxplot for each variable to determine which variables would best predict whether the tumor is benign or malignant. Next, I split the data into a train_set and a test_set. Using the train_set I began the modeling process. As a baseline, I used only the variable radius_mean, which I had determined had consistently different values for malignant and benign tumors. I improved the algorithm by trying several glm models (Generalized Linear Model)s using different variables. I also tried several knn models (K Nearest Neighbors), and an rpart model (Regression Tree).

Methods/Analysis

I began by installing and loading the required packages.

Next I imported the data set and assigned it to breast_cancer. I removed the last column because it contained only NAs and I turned the tibble into a data set.

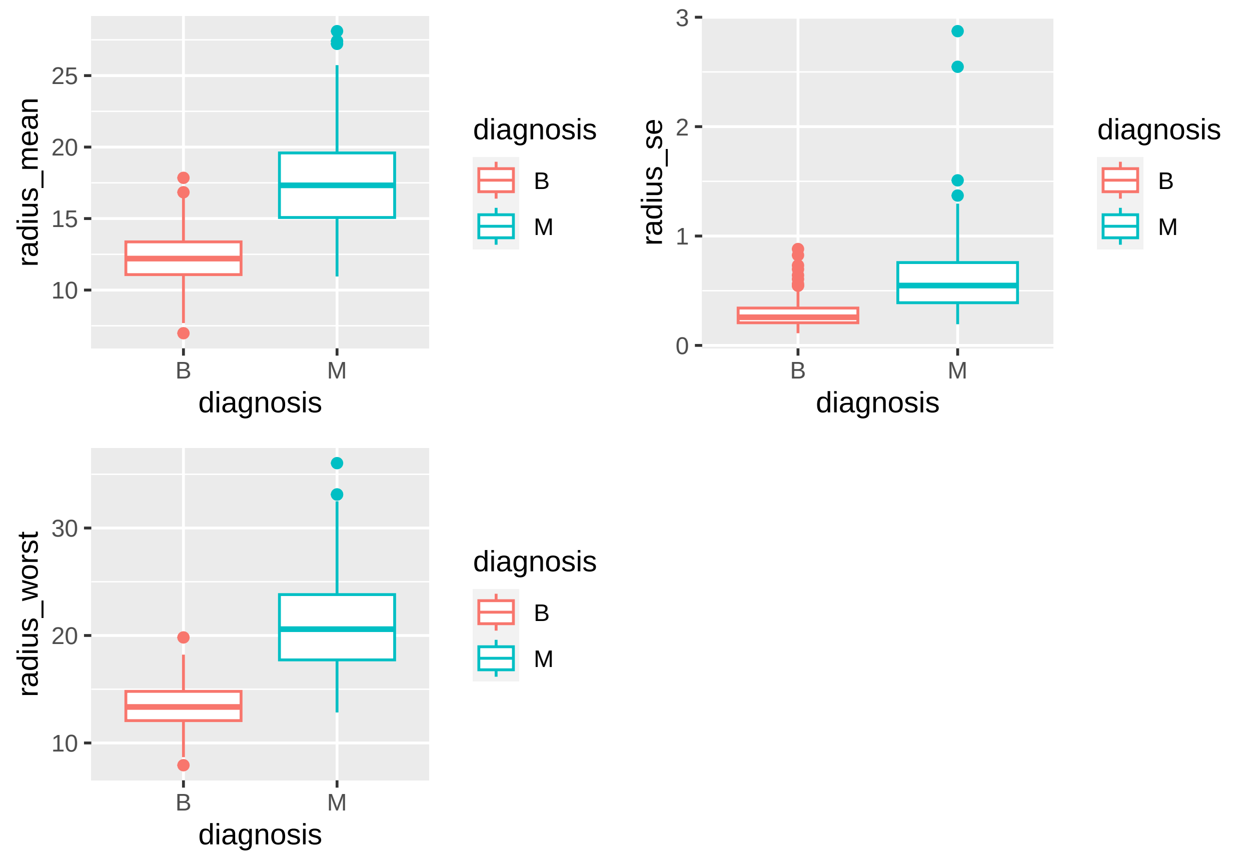
```r
#View the cleaned and organized data set
head(breast_cancer)
```

```
##         id diagnosis radius_mean texture_mean perimeter_mean area_mean
## 1   842302         M       17.99        10.38         122.80    1001.0
## 2   842517         M       20.57        17.77         132.90    1326.0
## 3 84300903         M       19.69        21.25         130.00    1203.0
## 4 84348301         M       11.42        20.38          77.58     386.1
## 5 84358402         M       20.29        14.34         135.10    1297.0
## 6   843786         M       12.45        15.70          82.57     477.1
##   smoothness_mean compactness_mean concavity_mean concave_points_mean
## 1         0.11840          0.27760         0.3001             0.14710
## 2         0.08474          0.07864         0.0869             0.07017
## 3         0.10960          0.15990         0.1974             0.12790
## 4         0.14250          0.28390         0.2414             0.10520
## 5         0.10030          0.13280         0.1980             0.10430
## 6         0.12780          0.17000         0.1578             0.08089
##   symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
## 1        0.2419                0.07871    1.0950     0.9053        8.589
## 2        0.1812                0.05667    0.5435     0.7339        3.398
## 3        0.2069                0.05999    0.7456     0.7869        4.585
## 4        0.2597                0.09744    0.4956     1.1560        3.445
## 5        0.1809                0.05883    0.7572     0.7813        5.438
## 6        0.2087                0.07613    0.3345     0.8902        2.217
##   area_se smoothness_se compactness_se concavity_se concave_points_se
## 1  153.40      0.006399        0.04904      0.05373           0.01587
## 2   74.08      0.005225        0.01308      0.01860           0.01340
## 3   94.03      0.006150        0.04006      0.03832           0.02058
## 4   27.23      0.009110        0.07458      0.05661           0.01867
## 5   94.44      0.011490        0.02461      0.05688           0.01885
## 6   27.19      0.007510        0.03345      0.03672           0.01137
##   symmetry_se fractal_dimension_se radius_worst texture_worst perimeter_worst
## 1     0.03003             0.006193        25.38         17.33          184.60
## 2     0.01389             0.003532        24.99         23.41          158.80
## 3     0.02250             0.004571        23.57         25.53          152.50
## 4     0.05963             0.009208        14.91         26.50           98.87
## 5     0.01756             0.005115        22.54         16.67          152.20
## 6     0.02165             0.005082        15.47         23.75          103.40
##   area_worst smoothness_worst compactness_worst concavity_worst
## 1     2019.0           0.1622            0.6656          0.7119
## 2     1956.0           0.1238            0.1866          0.2416
## 3     1709.0           0.1444            0.4245          0.4504
## 4      567.7           0.2098            0.8663          0.6869
## 5     1575.0           0.1374            0.2050          0.4000
## 6      741.6           0.1791            0.5249          0.5355
##   concave_points_worst symmetry_worst fractal_dimension_worst
## 1               0.2654         0.4601                 0.11890
## 2               0.1860         0.2750                 0.08902
## 3               0.2430         0.3613                 0.08758
## 4               0.2575         0.6638                 0.17300
## 5               0.1625         0.2364                 0.07678
## 6               0.1741         0.3985                 0.12440
```

Before creating a model, I created box plots for each variable mean. If the variable mean showed significant difference between the malignant and benign tumors, I created box plots of that variables se (standard error) and worst (the mean of the highest 3 values in that variable). I found that the most predictive variables were the radius mean, se, and worst, the perimeter mean, se, and worst, the area mean, se, and worst, and the concave points mean, se, and worst.

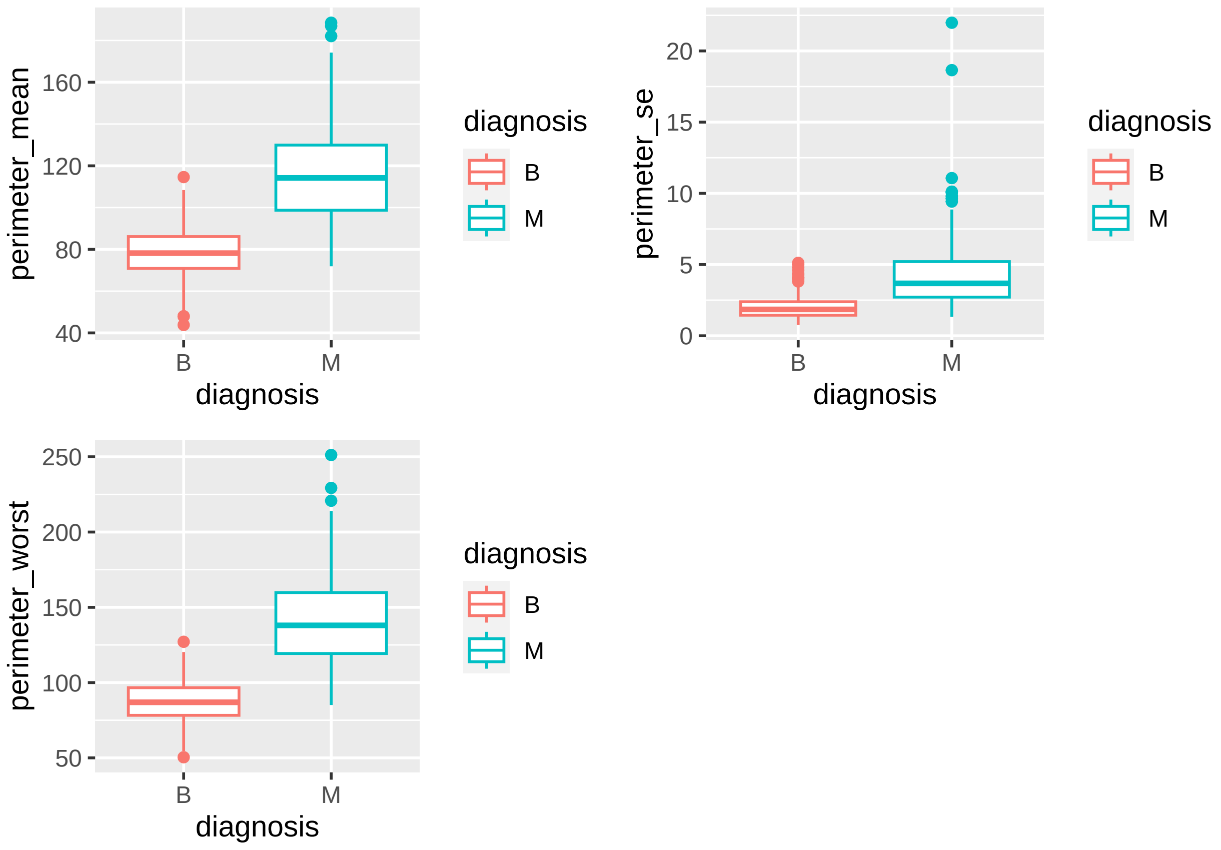
The following box plots show that the radius mean, se, and worst all have significantly different values for malignant and benign tumors.

```r
#Boxplot of radius mean (good), radius se (good), radius worst (good)
box_radius_mean <- breast_cancer %>%
    ggplot(aes(diagnosis, radius_mean, color = diagnosis)) +
    geom_boxplot()

box_radius_se <- breast_cancer %>%
    ggplot(aes(diagnosis, radius_se, color = diagnosis)) +
    geom_boxplot()

box_radius_worst <- breast_cancer %>%
    ggplot(aes(diagnosis, radius_worst, color = diagnosis)) +
    geom_boxplot()

ggarrange(box_radius_mean, box_radius_se, box_radius_worst,
          ncol = 2, nrow = 2)
```



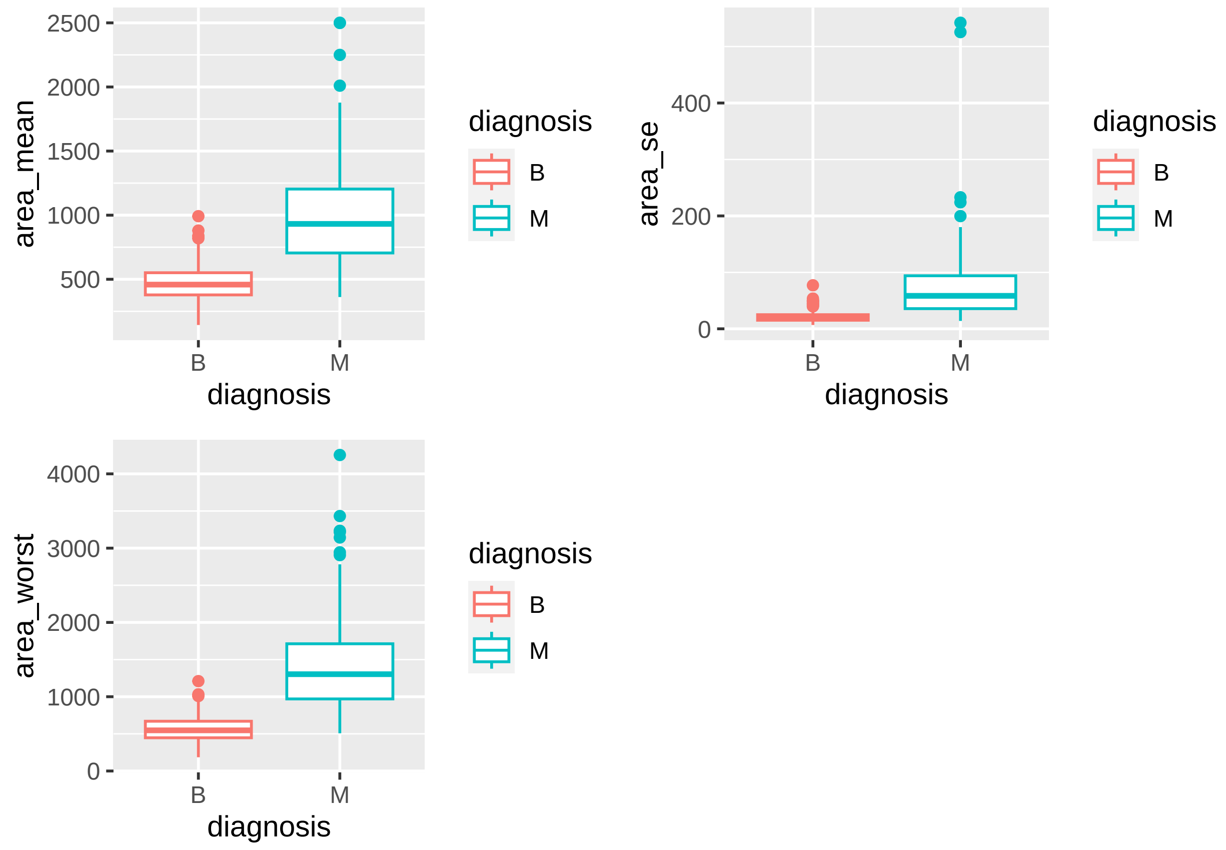
The following box plots show that the perimeter mean, se, and worst all have significantly different values for malignant and benign tumors.

```
#Boxplot of perimeter mean (good) perimeter se (could be good) perimeter worst (good)
box_perimeter_mean <- breast_cancer %>%
    ggplot(aes(diagnosis, perimeter_mean, color = diagnosis)) +
    geom_boxplot()

box_perimeter_se <- breast_cancer %>%
    ggplot(aes(diagnosis, perimeter_se, color = diagnosis)) +
    geom_boxplot()

box_perimeter_worst <- breast_cancer %>%
    ggplot(aes(diagnosis, perimeter_worst, color = diagnosis)) +
    geom_boxplot()

ggarrange(box_perimeter_mean, box_perimeter_se, box_perimeter_worst,
          ncol = 2, nrow = 2)
```



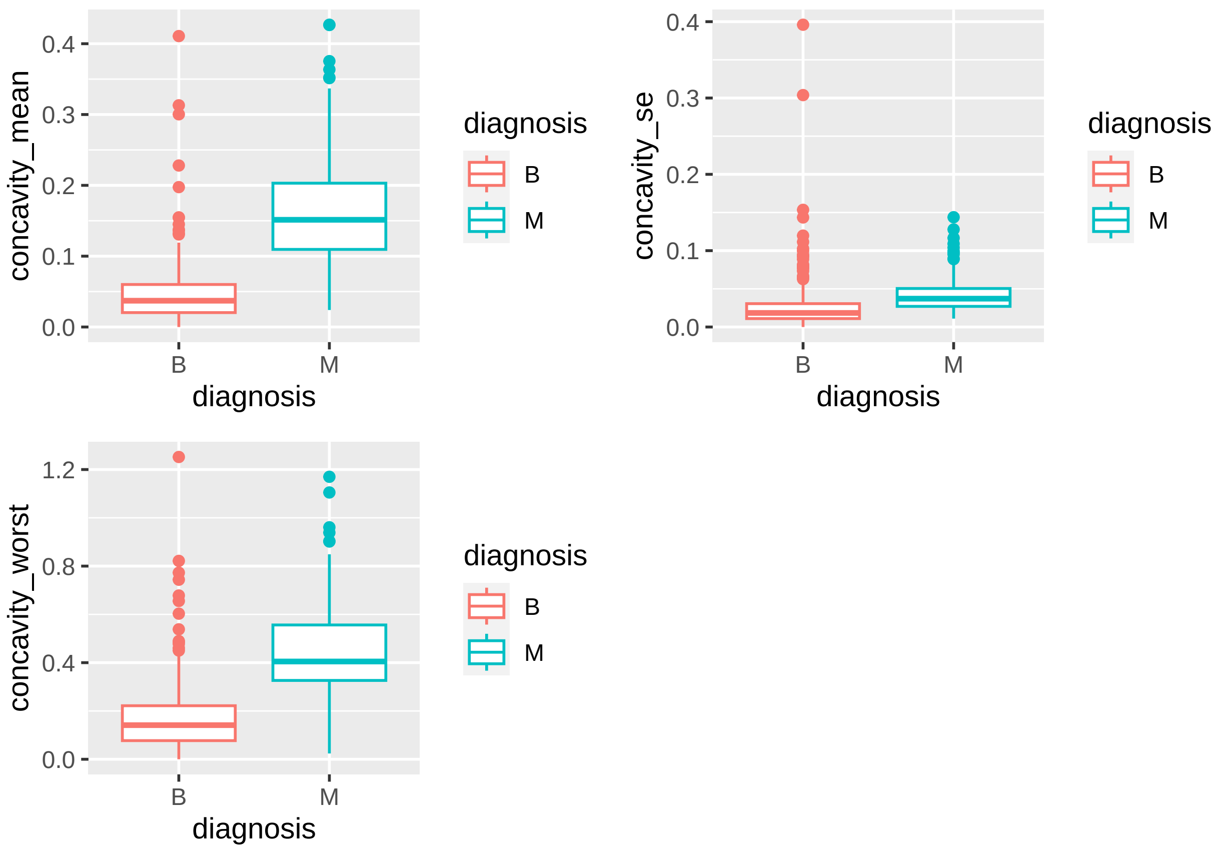
The following box plots show that the area mean, se, and worst all have significantly different values for malignant and benign tumors.

```r
#Boxplot of area mean (good) area se (good) and area worst (good)
box_area_mean <- breast_cancer %>%
    ggplot(aes(diagnosis, area_mean, color = diagnosis)) +
    geom_boxplot()

box_area_se <- breast_cancer %>%
    ggplot(aes(diagnosis, area_se, color = diagnosis)) +
    geom_boxplot()

box_area_worst <- breast_cancer %>%
    ggplot(aes(diagnosis, area_worst, color = diagnosis)) +
    geom_boxplot()

ggarrange(box_area_mean, box_area_se, box_area_worst,
          ncol = 2, nrow = 2)
```



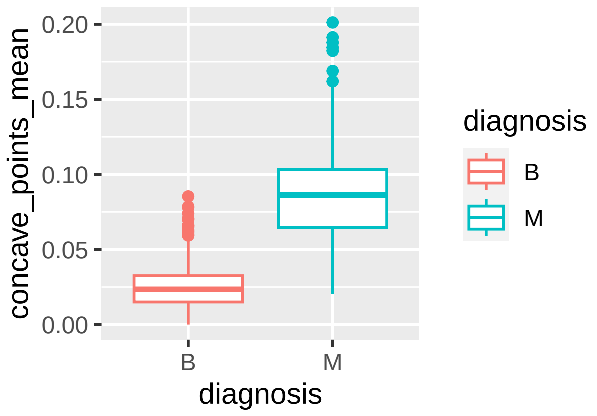
The following box plots show that the concavity mean significantly different values for malignant and benign tumors. However, the concavity se, and worst values seem to overlap a lot.

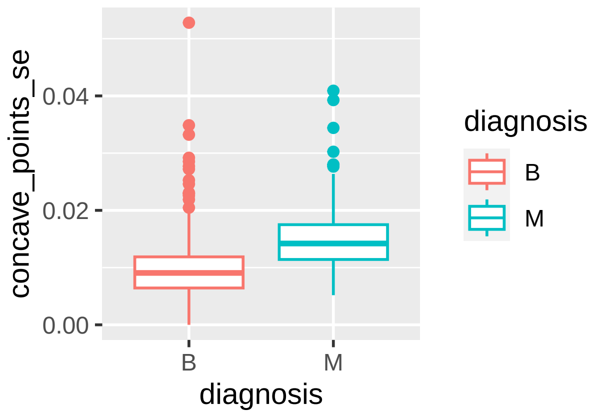
```
#Boxplots of cancavity mean (could be good) se (not good) and worst (not good)
box_concavity_mean <- breast_cancer %>%
    ggplot(aes(diagnosis, concavity_mean, color = diagnosis)) +
    geom_boxplot()

box_concavity_se <- breast_cancer %>%
    ggplot(aes(diagnosis, concavity_se, color = diagnosis)) +
    geom_boxplot()

box_concavity_worst <- breast_cancer %>%
    ggplot(aes(diagnosis, concavity_worst, color = diagnosis)) +
    geom_boxplot()

ggarrange(box_concavity_mean, box_concavity_se, box_concavity_worst,
          ncol = 2, nrow = 2)
```

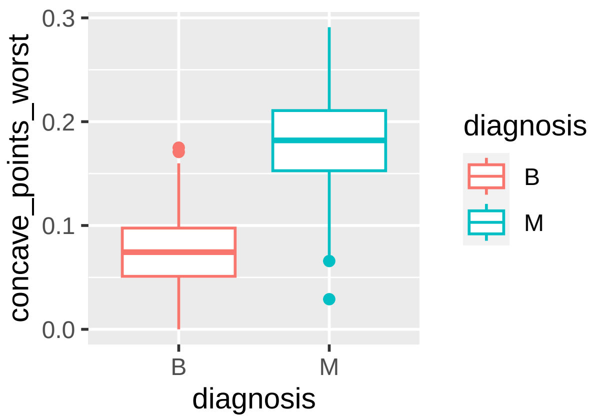


The following box plots show that the concave points mean, se, and worst all have significantly different values for malignant and benign tumors.

```r
#Boxplot of concave points mean (good) se (not so good) and worst (good)
box_concave_mean <- breast_cancer %>%
    ggplot(aes(diagnosis, concave_points_mean, color = diagnosis)) +
    geom_boxplot()

box_concave_se <- breast_cancer %>%
    ggplot(aes(diagnosis, concave_points_se, color = diagnosis)) +
    geom_boxplot()

box_concave_worst <- breast_cancer %>%
    ggplot(aes(diagnosis, concave_points_worst, color = diagnosis)) +
    geom_boxplot()

ggarrange(box_concave_mean, box_concave_se, box_concave_worst,
          ncol = 2, nrow = 2)
```

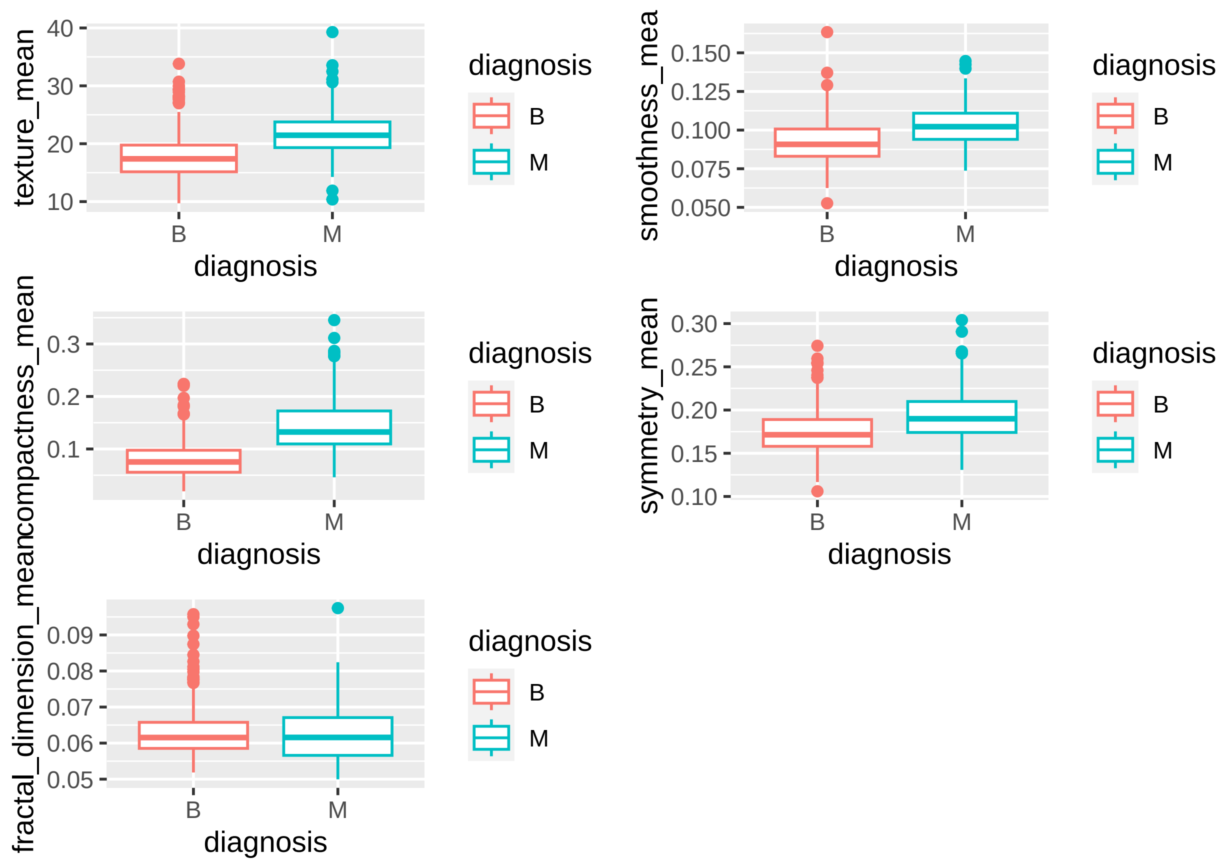






The following box plots show that the texture, smoothness, compactness, symmetry, and fractal dimension mean all have overlap between the values for malignant and benign tumors.

```r
#Boxplot of texture mean
box_texture_mean <- breast_cancer %>%
    ggplot(aes(diagnosis, texture_mean, color = diagnosis)) +
    geom_boxplot()

#Boxplot of smoothness mean
box_smooth_mean <- breast_cancer %>%
    ggplot(aes(diagnosis, smoothness_mean, color = diagnosis)) +
    geom_boxplot()

#Boxplot of compactness mean
box_comp_mean <- breast_cancer %>%
    ggplot(aes(diagnosis, compactness_mean, color = diagnosis)) +
    geom_boxplot()

#Boxplot of symmetry mean
box_symmetry_mean <- breast_cancer %>%
    ggplot(aes(diagnosis, symmetry_mean, color = diagnosis)) +
    geom_boxplot()

#Boxplot of fractal dimension mean
box_fractal_mean <- breast_cancer %>%
    ggplot(aes(diagnosis, fractal_dimension_mean, color = diagnosis)) +
    geom_boxplot()

#Page of less useful parameters
ggarrange(box_texture_mean, box_smooth_mean, box_comp_mean,
          box_symmetry_mean, box_fractal_mean,
          ncol = 2, nrow = 3)
```

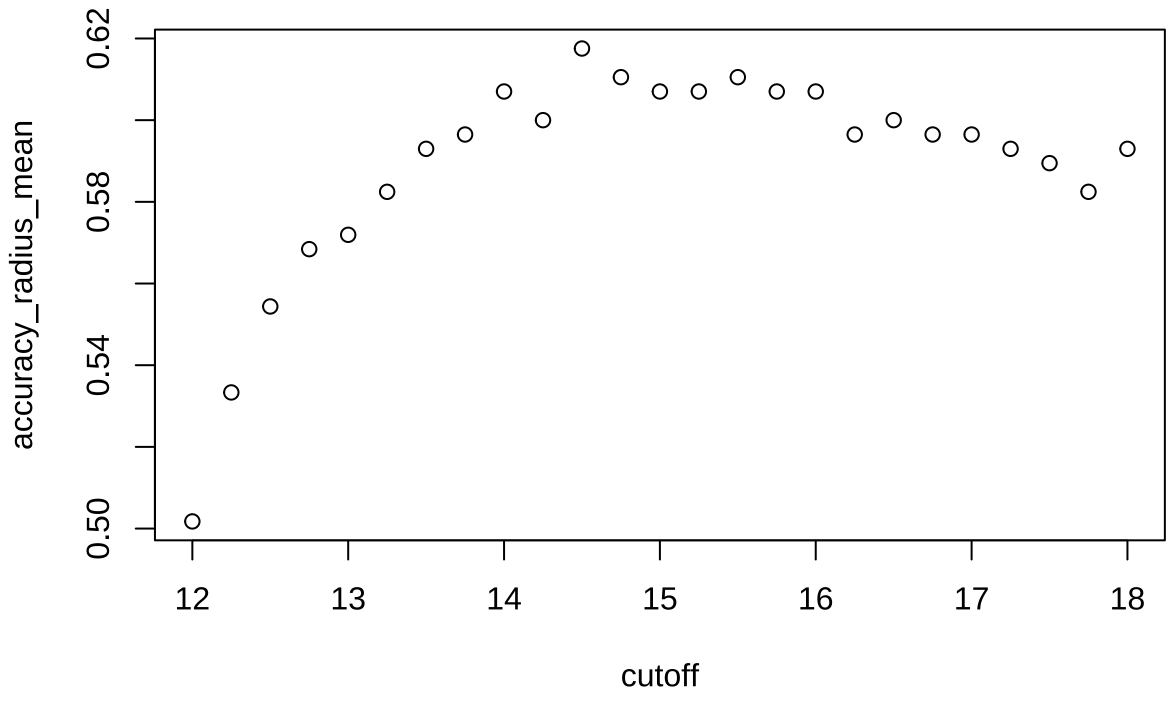


After having discovered which variables have the most predictive power I separated the breast_cancer data set into a test set and a train set.

```
#Set seed to 1
set.seed(1, sample.kind = "Rounding")

#Separate into train and test set
y <- breast_cancer$diagnosis

test_index <- createDataPartition(y, times = 1, p = 0.5, list = FALSE)

test_set <- breast_cancer[test_index, ]
train_set <- breast_cancer[-test_index, ]
```

Using the radius mean I created a primitive model to establish a baseline accuracy. I created a function that tests multiple cutoff points and uses the one that achieved the highest accuracy. The accuracy for this model is consistently between 0.58 and 0.62.

```
#Try predicting using the radius mean and choosing a cutoff
cutoff <- seq(12, 18, 0.25)
accuracy_radius_mean <- map_dbl(cutoff, function(x){
    y_hat <- ifelse(train_set$radius_mean > x, "M", "B")
    mean(y_hat == test_set$diagnosis)
})

#Plot the cutoffs against their accuracy at predicting B or M
plot(cutoff, accuracy_radius_mean)
```



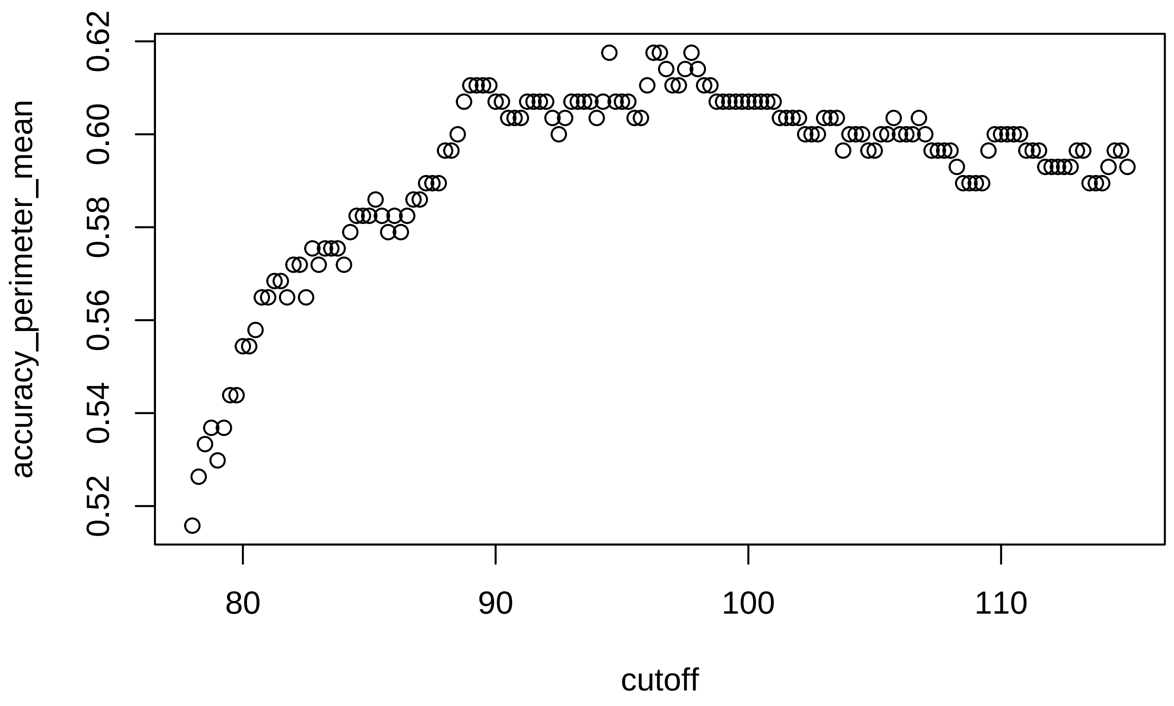
```
accuracy_mean <- max(accuracy_radius_mean)
accuracy_mean
```

```
## [1] 0.6175439
```

Next I tried the perimeter mean to see if there was an improvement. There was not.

```
#Try predicting using the perimeter mean and choosing a cutoff
cutoff <- seq(78, 115, 0.25)

accuracy_perimeter_mean <- map_dbl(cutoff, function(x){
    y_hat <- ifelse(train_set$perimeter_mean > x, "M", "B")
    mean(y_hat == test_set$diagnosis)
})
```

```
#Plot the cutoffs against their accuracy
plot(cutoff, accuracy_perimeter_mean)
```



```
accuracy_perimeter <- max(accuracy_perimeter_mean)
accuracy_perimeter
```

```
## [1] 0.6175439
```

To improve the accuracy I tried a glm model using only the radius mean. This model significantly improved the accuracy.

```
#Try a glm model for radius mean
train_glm_radius_mean <- train(diagnosis ~ radius_mean, method = "glm",
                               data = train_set)

y_hat_glm_radius_mean <- predict(train_glm_radius_mean, test_set)

confusionMatrix(y_hat_glm_radius_mean, as.factor(test_set$diagnosis))$overall[["Accuracy"]]
```

```
## [1] 0.8736842
```

While this is a significant improvement, it could be better. I tried the radius mean, se, and worst on a glm model.

```r
#Try a glm model for radius mean, se, and worst
train_glm_radius <- train(diagnosis ~ radius_mean + radius_se + radius_worst,
                          method = "glm", data = train_set)

y_hat_glm_radius <- predict(train_glm_radius, test_set)

accuracy_glm_radius <- confusionMatrix(y_hat_glm_radius,
                                       as.factor(test_set$diagnosis))$overall[["Accuracy"]]
accuracy_glm_radius
```

```
## [1] 0.9263158
```

I tried the perimeter mean, se, and worst on a glm model.

```r
#Try a glm model for perimeter mean, se, and worst
train_glm_perimeter <- train(diagnosis ~ perimeter_mean + perimeter_se + perimeter_worst, method = "glm

y_hat_glm_perimeter <- predict(train_glm_perimeter, test_set)

accuracy_glm_perimeter <- confusionMatrix(y_hat_glm_perimeter, as.factor(test_set$diagnosis))$overall[[
accuracy_glm_perimeter
```

```
## [1] 0.9333333
```

I tried the area mean, se, and worst on a glm model.

```r
#Try glm model for area mean, se, and worst
train_glm_area <- train(diagnosis ~ area_mean + area_se + area_worst,
                        method = "glm", data = train_set)

y_hat_glm_area <- predict(train_glm_area, test_set)

accuracy_glm_area <- confusionMatrix(y_hat_glm_area, as.factor(test_set$diagnosis))$overall[["Accuracy"
accuracy_glm_area
```

```
## [1] 0.9298246
```

I tried the concave points mean, se, and worst on a glm model.

```r
#Try glm model for concave points mean, se, and worst
train_glm_concave <- train(diagnosis ~ concave_points_mean + concave_points_se + concave_points_worst,

y_hat_glm_concave <- predict(train_glm_concave, test_set)

accuracy_glm_concave <- confusionMatrix(y_hat_glm_concave, as.factor(test_set$diagnosis))$overall[["Acc
accuracy_glm_concave
```

```
## [1] 0.9368421
```

I tried all the predictors on a glm model.

```r
#Try glm model for all predictors
train_glm_all <- train(diagnosis ~ .,
                       method = "glm",
                       data = train_set)

y_hat_glm_all <- predict(train_glm_all, test_set)

accuracy_glm <- confusionMatrix(y_hat_glm_all, as.factor(test_set$diagnosis))$overall[["Accuracy"]]
accuracy_glm
```

```
## [1] 0.9298246
```

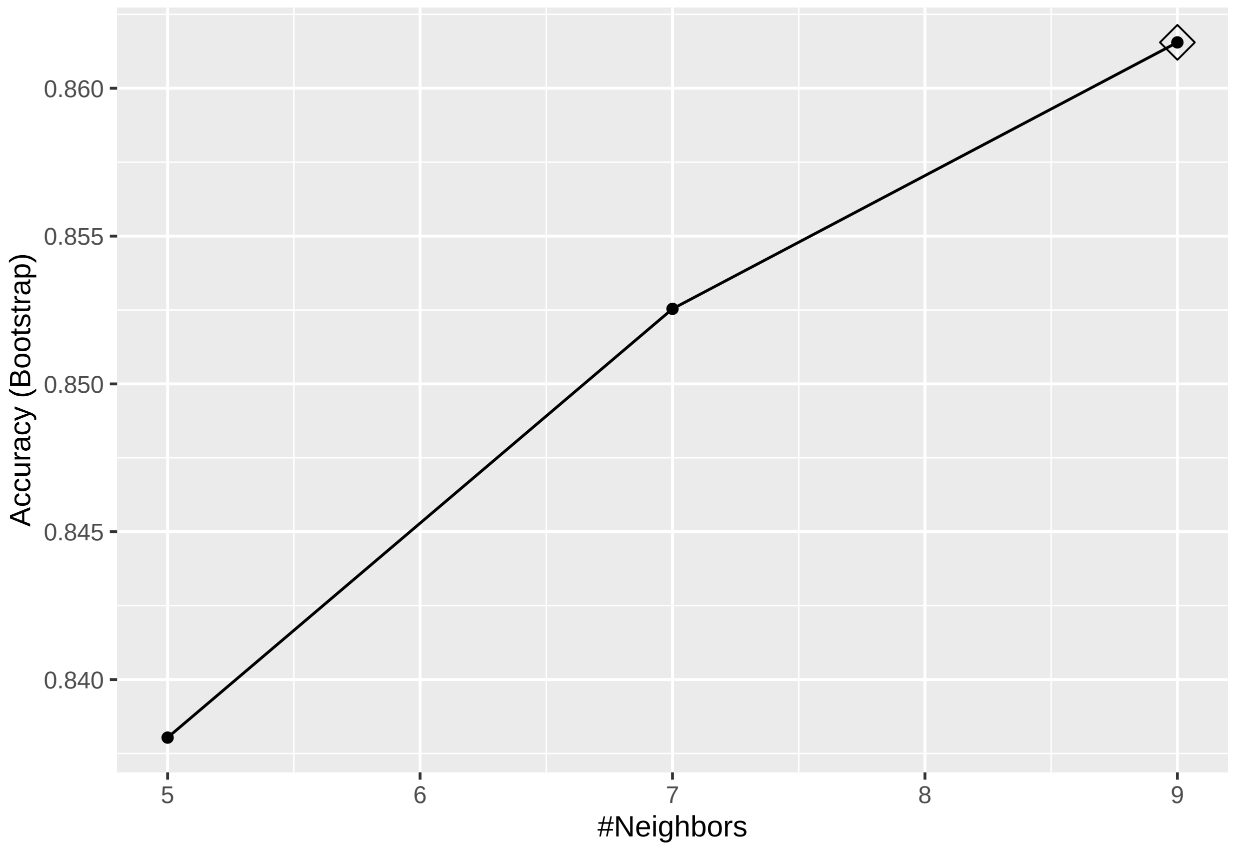
I tried a knn model and an rpart model to see if either of them would improve the accuracy.

First I tried the knn method using the default tune grid and only the radius mean. This model achieved a lower accuracy than the previous glm models.

```r
#Try knn model for radius mean (worse than glm)
train_knn_radius_mean <- train(diagnosis ~ radius_mean,
                               method = "knn",
                               data = train_set)

y_hat_knn_radius_mean <- predict(train_knn_radius_mean, test_set)

confusionMatrix(y_hat_knn_radius_mean, as.factor(test_set$diagnosis))$overall[["Accuracy"]]
```

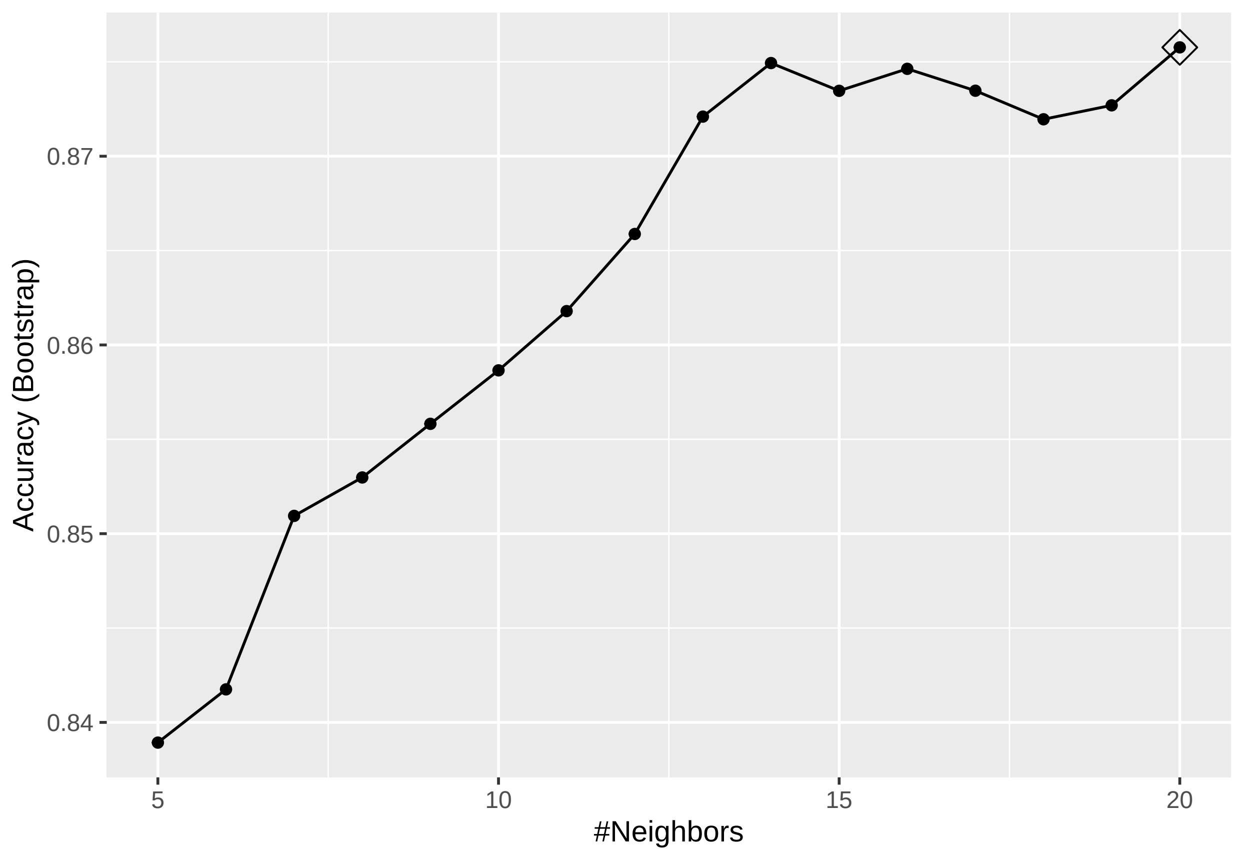
```
## [1] 0.8701754
```

```r
ggplot(train_knn_radius_mean, highlight = TRUE)
```



It is clear from the plot, the default tuning parameters do not include the most accurate tuning parameter. To address this I chose tuning parameters and tested the model again.

```r
#Try tuning knn for just radius mean
train_knn_radius_tune <- train(diagnosis ~ radius_mean,
                               method = "knn",
                               data = train_set,
                               tuneGrid = data.frame(k = seq(5, 20)))

ggplot(train_knn_radius_tune, highlight = TRUE)
```

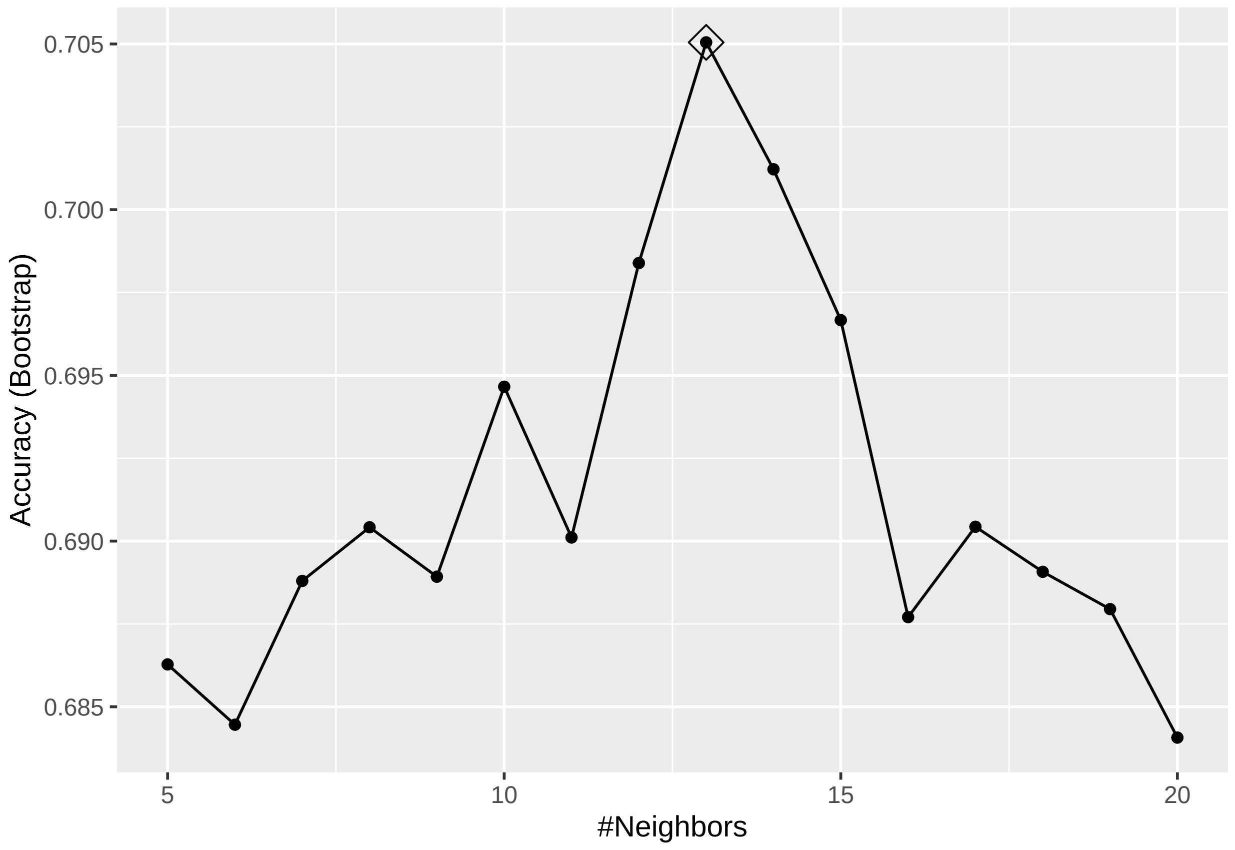


```
y_hat_knn_radius_tune <- predict(train_knn_radius_tune, test_set)

confusionMatrix(y_hat_knn_radius_tune, as.factor(test_set$diagnosis))$overall[["Accuracy"]]
```

```
## [1] 0.8736842
```

Using different tuning parameters increased the accuracy but only by a little. Next I tried the knn method with the chosen tuning parameters and all predictors. This improved the accuracy but was still lower than the glm models.

```
#Try tuning knn for all variables
train_knn_all <- train(diagnosis ~ .,
                       method = "knn",
                       data = train_set,
                       tuneGrid = data.frame(k = seq(5, 20)))

ggplot(train_knn_all, highlight = TRUE)
```

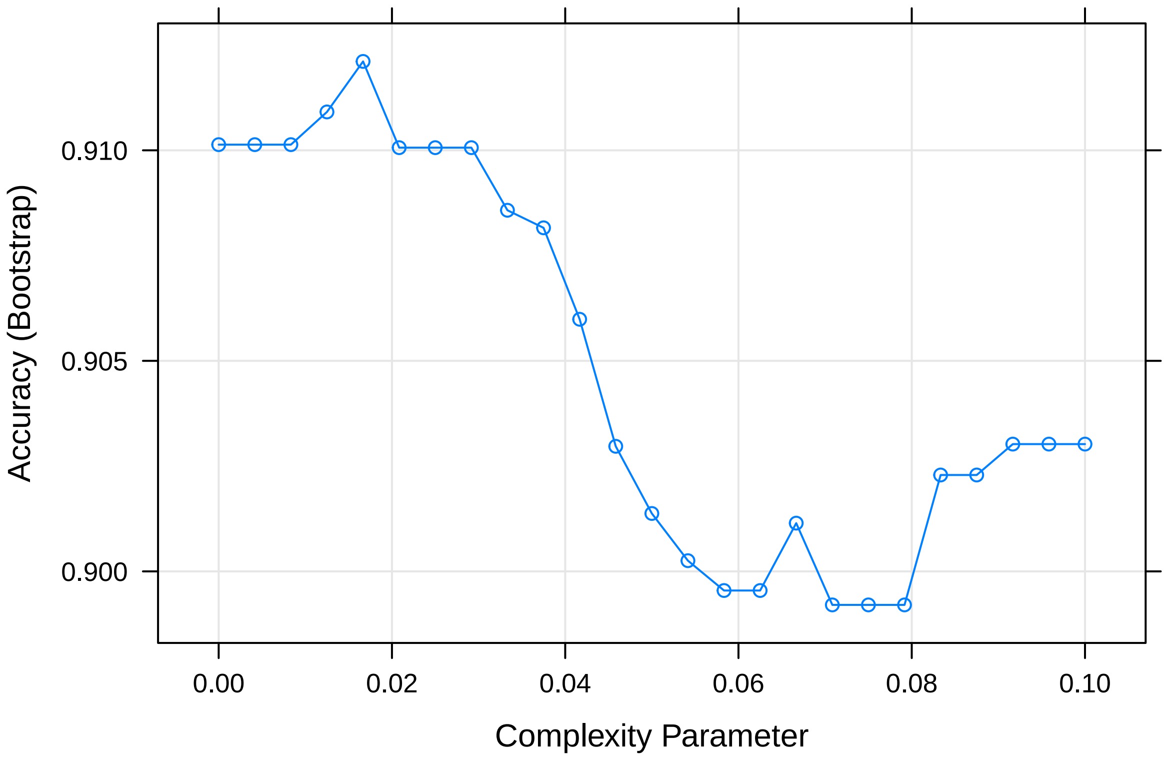


```
accuracy_knn_all <- confusionMatrix(predict(train_knn_all, test_set), as.factor(test_set$diagnosis))$ove
accuracy_knn_all
```

```
##  Accuracy
## 0.6947368
```

Using all the predictors did not improve the accuracy. The accuracy actually became significantly worse. I tried an rpart model using all the predictors. The accuracy was significantly better than the knn model and most of the glm models.

```
#Try an rpart model using all the predictors
train_rpart_all <- train(diagnosis ~ .,
                         method = "rpart",
                         tuneGrid = data.frame(cp = seq(0.0, 0.1, len = 25)),
                         data = train_set)

plot(train_rpart_all)
```



```
y_hat_rpart <- predict(train_rpart_all, test_set)

accuracy_rpart_all <- confusionMatrix(data = y_hat_rpart, reference = as.factor(test_set$diagnosis))$ove
accuracy_rpart_all
```

```
##  Accuracy
## 0.9368421
```

Results

In order to determine the most accurate model I created a table of accuracy. Overall, the glm models that used multiple predictors and the rpart model were the most accurate.

```
table_accuracies <- data.frame(accuracy_mean = mean(accuracy_mean),
                               accuracy_perimeter = mean(accuracy_perimeter),
                               accuracy_glm_area = mean(accuracy_glm_area),
                               accuracy_glm_concave = mean(accuracy_glm_concave),
                               accuracy_glm = mean(accuracy_glm),
                               accuracy_knn_all = mean(accuracy_knn_all),
                               accuracy_rpart_all = mean(accuracy_rpart_all),
                               row.names = "Accuracy")

t(table_accuracies)
```

```
##                         Accuracy
## accuracy_mean        0.6175439
## accuracy_perimeter   0.6175439
## accuracy_glm_area    0.9298246
## accuracy_glm_concave 0.9368421
## accuracy_glm         0.9298246
## accuracy_knn_all     0.6947368
## accuracy_rpart_all   0.9368421
```

The most accurate models were the rpart model and the glm concave points model. I inspected a confusion matrix for the rpart model to check the specificity and sensitivity. The models are random so the accuracy could change a little each time it is run.

```r
#Confusion matrix for the rpart model
confusionMatrix(data = predict(train_rpart_all, test_set),
                reference = as.factor(test_set$diagnosis))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   B   M
##          B 172  11
##          M   7  95
##
##                Accuracy : 0.9368
##                  95% CI : (0.902, 0.9621)
##     No Information Rate : 0.6281
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8638
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 0.9609
##             Specificity : 0.8962
##          Pos Pred Value : 0.9399
##          Neg Pred Value : 0.9314
##              Prevalence : 0.6281
##          Detection Rate : 0.6035
##    Detection Prevalence : 0.6421
##       Balanced Accuracy : 0.9286
##
##        'Positive' Class : B
##
```

The sensitivity is significantly higher than the specificity. This means that themodel more accurately predicts a benign tumor than it does a malignant one.

Conclusion

After importing, cleaning, and organizing the data set I looked at the relationships between the characteristics of the tumor and whether or not the tumor was malignant. Using box plots I determined that the radius mean, se, and worst, the perimeter mean, se, and worst, the area mean, se, and worst, and the concave points mean, se, and worst were the predictors for whether a tumor was malignant or not. I began by using only the radius mean for my first model. The accuracy from the radius mean model was consistently between

0.58 and 0.62. To improve this accuracy I tried the glm model, the knn model, and the rpart model and used multiple predictors. The most accurate models were the concave points model and the rparts model. There are models that I did not try. In the future, the model could be improved using a gam loess model or a random forest model. There are likely more variables that could help predict whether a tumor is malignant or not.

References

https://www.kaggle.com/datasets/priyanka841/breast-cancer-wisconsin?resource=download