Project 2: Supervised and Unsupervised Learning

SDS322E

Dangerous Dingos: Doan Nguyen, Ethan Chang, Natleigh Burns

Introduction

Supervised and unsupervised learning are two of the most popular approaches to machine learning, a field of study that allows computers to learn without being explicitly programmed to do so. While they have many applications in the real world, it is still an emerging technology that has not been fully utilized in many professional fields. As such, we seek to analyze how effective and useful these tools can be in one of the most important fields in our lives: the medical field.

The data set we chose to analyze was the Breast Cancer Wisconsin (Diagnostic) data set found on Kaggle. We were interested in seeing how different machine learning approaches would do at predicting the diagnosis of cancer based off cell nuclei data. As there were 32 variables initially present in the data set, we decided to cut it down to 11 main variables:

Binary Variables:

1. Diagnosis - determines whether the breast tissue is M = malignant or B = benign.

Numeric Variables (mean value for each cell nucleus):

2. Radius - the mean distance from the center of the cell nuclei to the perimeter
3. Texture - standard deviation of gray-scale values on the cell nuclei
4. Perimeter - perimeter of the cell nuclei
5. Area - area of the cell nuclei
6. Smoothness - local variation in radius lengths
7. Compactness - calculated as perimeter^2 / area - 1.0

8. Concavity - severity of concave portions of the contour
9. Concave points - number of concave portions of the contour
10. Symmetry - how symmetrical the cell nuclei is
11. Fractal Dimension - coastline approximation - 1, measures the complexity of the cell nuclei

In our data set, there was a total of 569 observations, 357 of which were diagnosed as benign and 212 diagnosed as malignant.

```
library(tidyverse)
library(ggplot2)
library(ROCR)
library(caret)
library(pROC)
library(janitor)
library(ggthemes)
library(ggfortify)
library(e1071)
library(caret)
library(rpart)
library(randomForest)
library(rpart.plot)
```

```
data <- read.csv("data.csv")
head(data)
```

```
##         id diagnosis radius_mean texture_mean perimeter_mean area_mean
## 1   842302         M       17.99        10.38          122.8      1001
## 2   842517         M       20.57        17.77          132.9      1326
## 3 84300903         M       19.69        21.25          130.0      1203
##   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
##   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
##   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
##   symmetry_se fractal_dimension_se radius_worst texture_worst perimeter_worst
## 1     0.03003             0.006193        25.38         17.33           184.6
## 2     0.01389             0.003532        24.99         23.41           158.8
## 3     0.02250             0.004571        23.57         25.53           152.5
##   area_worst smoothness_worst compactness_worst concavity_worst
## 1       2019           0.1622            0.6656          0.7119
## 2       1956           0.1238            0.1866          0.2416
## 3       1709           0.1444            0.4245          0.4504
##   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
##  [ reached 'max' / getOption("max.print") -- omitted 3 rows ]
```

```
# if your dataset needs tidying, do so here
cancerdata <- data %>%
    select(diagnosis, radius_mean, texture_mean, perimeter_mean,
        area_mean, smoothness_mean, compactness_mean, concavity_mean,
        concave.points_mean, symmetry_mean, fractal_dimension_mean)
head(cancerdata)
```

```
##   diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean
## 1         M       17.99        10.38         122.80    1001.0         0.11840
## 2         M       20.57        17.77         132.90    1326.0         0.08474
## 3         M       19.69        21.25         130.00    1203.0         0.10960
## 4         M       11.42        20.38          77.58     386.1         0.14250
## 5         M       20.29        14.34         135.10    1297.0         0.10030
## 6         M       12.45        15.70          82.57     477.1         0.12780
##   compactness_mean concavity_mean concave.points_mean symmetry_mean
## 1          0.27760         0.3001             0.14710        0.2419
## 2          0.07864         0.0869             0.07017        0.1812
## 3          0.15990         0.1974             0.12790        0.2069
## 4          0.28390         0.2414             0.10520        0.2597
## 5          0.13280         0.1980             0.10430        0.1809
## 6          0.17000         0.1578             0.08089        0.2087
##   fractal_dimension_mean
## 1                0.07871
## 2                0.05667
## 3                0.05999
## 4                0.09744
## 5                0.05883
## 6                0.07613
```

```
# any other code here
cancerdata$diagnosis <- as.factor(cancerdata$diagnosis)
```

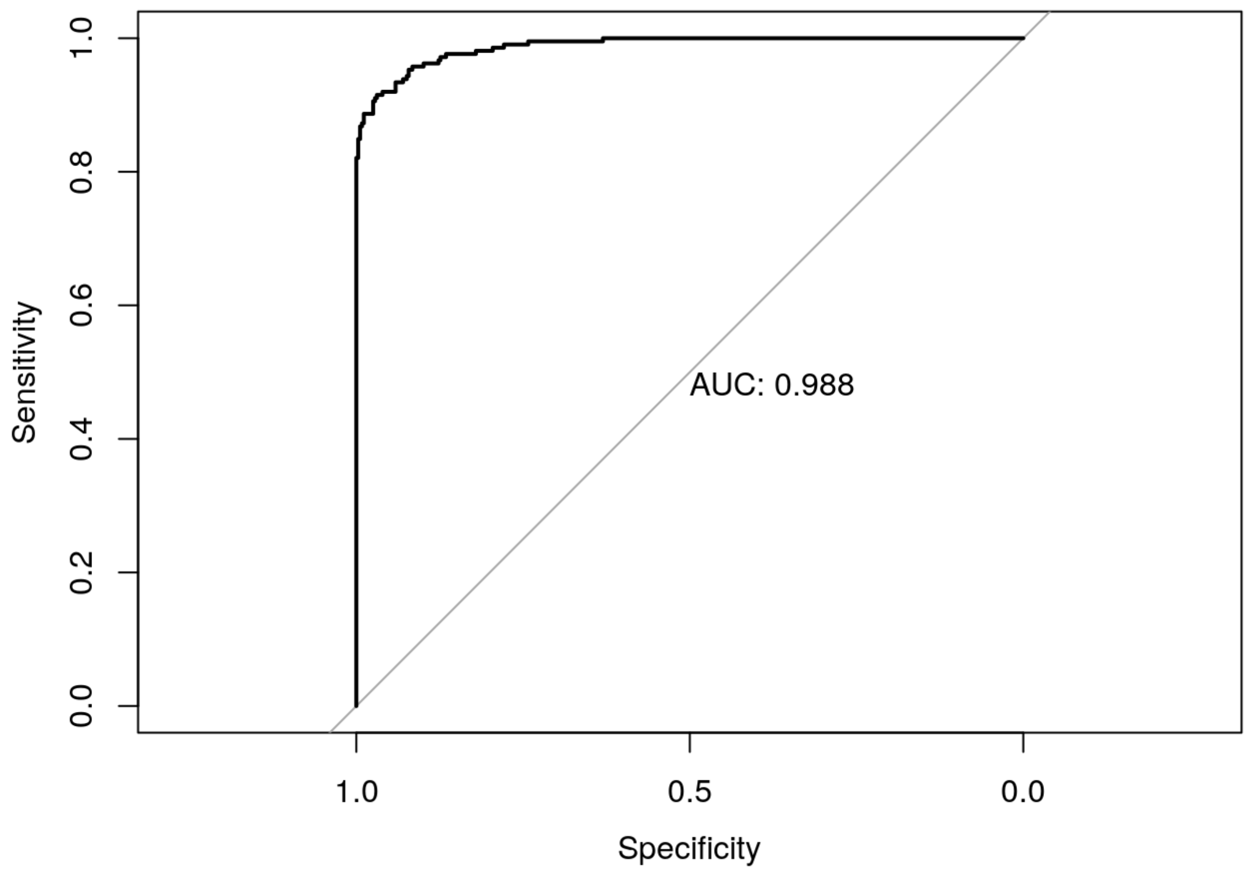
Supervised Learning

The first machine learning approach we take is that of supervised learning. In this, we provide our machine learning models with data consisting of both input and output data, allowing our model to learn an algorithm to accurately predict the output if similar input data were entered. This allows us to train our model to make accurate predictions if the appropriate data and training methods are used.

Logistic Regression

To start, we will first apply logistic regression to our entire data set to predict the diagnosis (our binary variable) from all of the means/numerical values.

```
logistic_model <- glm(diagnosis ~ ., family = binomial, data = cancerdata)
log_prediction <- predict(logistic_model, cancerdata, type = "response")
prediction <- ifelse(log_prediction > 0.5, "M", "B")
CM <- confusionMatrix(as.factor(prediction), cancerdata$diagnosis)
test_roc <- roc(cancerdata$diagnosis ~ log_prediction, plot = TRUE,
    print.auc = TRUE)
```



```
Value <- c(CM$overall[1], CM$byClass[1], CM$byClass[2], AUC = test_roc$auc)
data.frame(Value)
```

```
##                 Value
## Accuracy    0.9490334
## Sensitivity 0.9719888
## Specificity 0.9103774
## AUC         0.9879235
```

When analyzing the the entire data set using a logistic regression model, the classifier actually seems to be performing relatively well in terms of prediction. It has a high accuracy, sensitivity, specificity, and AUC (all exceeding 90%) which exceed our expectations going in. It only predicted 29 out of 569 cases incorrectly, which we thought to be quite low.

Using Training/Test Sets

Although our initial logistic regression model on the entire data set predicted the data pretty well, there is always a chance that it was overfitting the data. In order to account for this, we perform logistic regression again, but this time after splitting the initial data set into training and testing data that account for 80% and 20% of the entire data set, respectively. This way, we perform our logistic regression on the training data, and predict the results of the testing data to compare with the actual testing data, allowing us to check how accurately our model can predict data similar to that of the training data. We perform this 5 times using different seeds to record a more accurate representation of the average accuracy, sensitivity, specificity, and AUC values.

```
seeds <- c(122223, 283838, 26263, 19982, 376543)
sensitivity_logreg <- c()
specificity_logreg <- c()
accuracy_logreg <- c()
AUC_logreg <- c()

cancerdata$diagnosis <- ifelse(cancerdata$diagnosis == "M", 1,
    0)

for (i in 1:length(seeds)) {
    set.seed(seeds[i])
    idx_train <- createDataPartition(cancerdata$diagnosis, p = 0.8)[[1]]
    data_train <- cancerdata[idx_train, ]
    data_test <- cancerdata[-idx_train, ]

    glm.cancer <- glm(diagnosis ~ ., family = binomial, data = data_train)
    predicted <- predict(glm.cancer, data_test, type = "response")
    prediction <- ifelse(predicted > 0.5, 1, 0)

    result <- confusionMatrix(as.factor(prediction), as.factor(data_test$diagnosis))

    accuracy_logreg[i] <- result$overall["Accuracy"]
    sensitivity_logreg[i] <- sensitivity(as.factor(data_test$diagnosis),
        as.factor(prediction))
    specificity_logreg[i] <- specificity(as.factor(data_test$diagnosis),
        as.factor(prediction))
    AUC_logreg[i] <- auc(data_test$diagnosis, prediction)
}

log_a <- mean(accuracy_logreg)
log_b <- mean(sensitivity_logreg)
log_c <- mean(specificity_logreg)
log_d <- mean(AUC_logreg)

Name <- c("Accuracy", "Sensitivity", "Specificity", "AUC")
Value <- c(log_a, log_b, log_c, log_d)
data.frame(Name, Value)
```

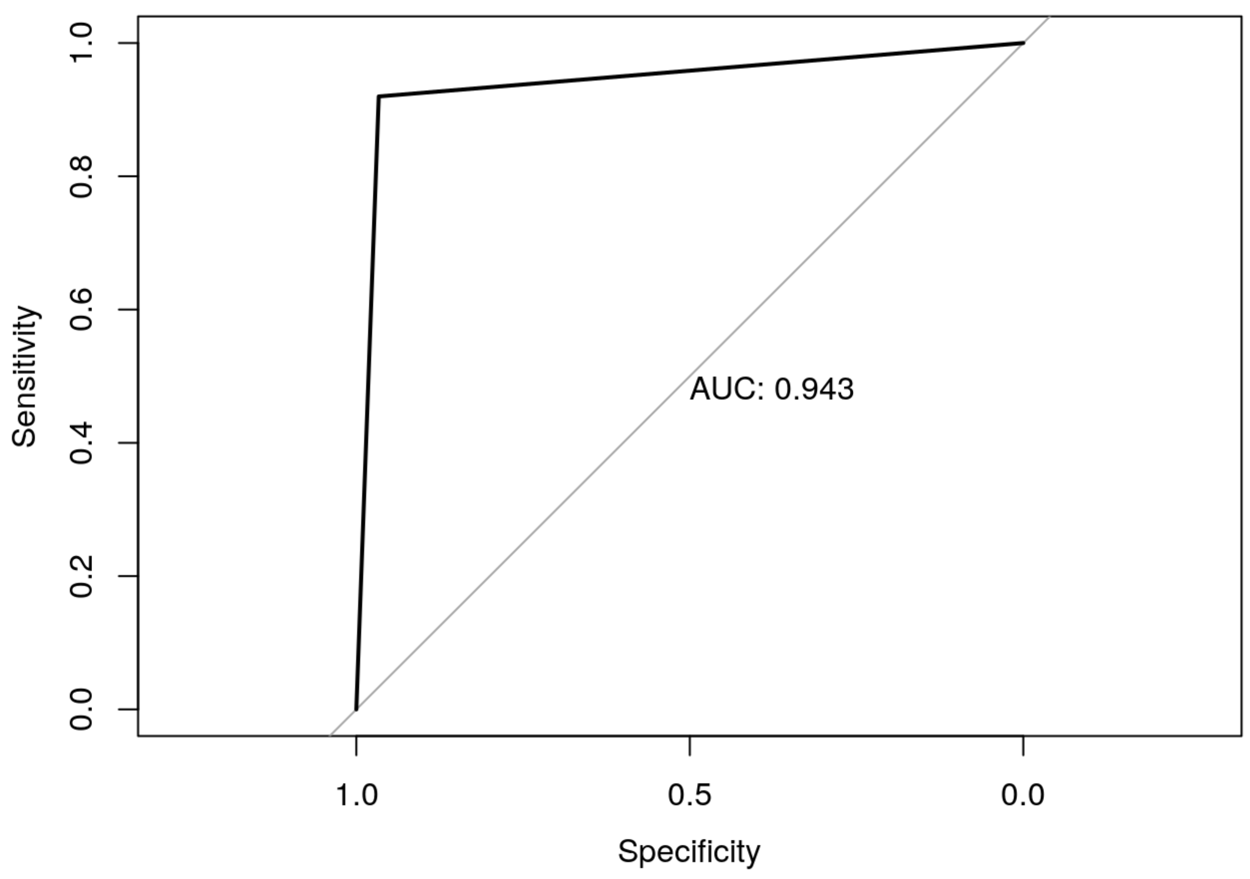
```
##          Name     Value
## 1    Accuracy 0.9238938
## 2 Sensitivity 0.9305236
## 3 Specificity 0.9120715
## 4         AUC 0.9159990
```

When looking at the results, it is clear that they are not as good as performing logistic regression on the entire data set, which is expected. Despite this, it still does a good job at predicting new observations on the testing data, with accuracy, sensitivity, specificity, and AUC values all still exceeding 90%. With this we would say that our model is predicting new observations per train/test splitting pretty well still. As such we do not really see any obvious signs of overfitting in our logistic regression model.

Tree-based Classifiers

Next, we fit a CART and a random forest, both tree-based classifiers, on our same data set to predict the same diagnosis variable from the same numerical variables.

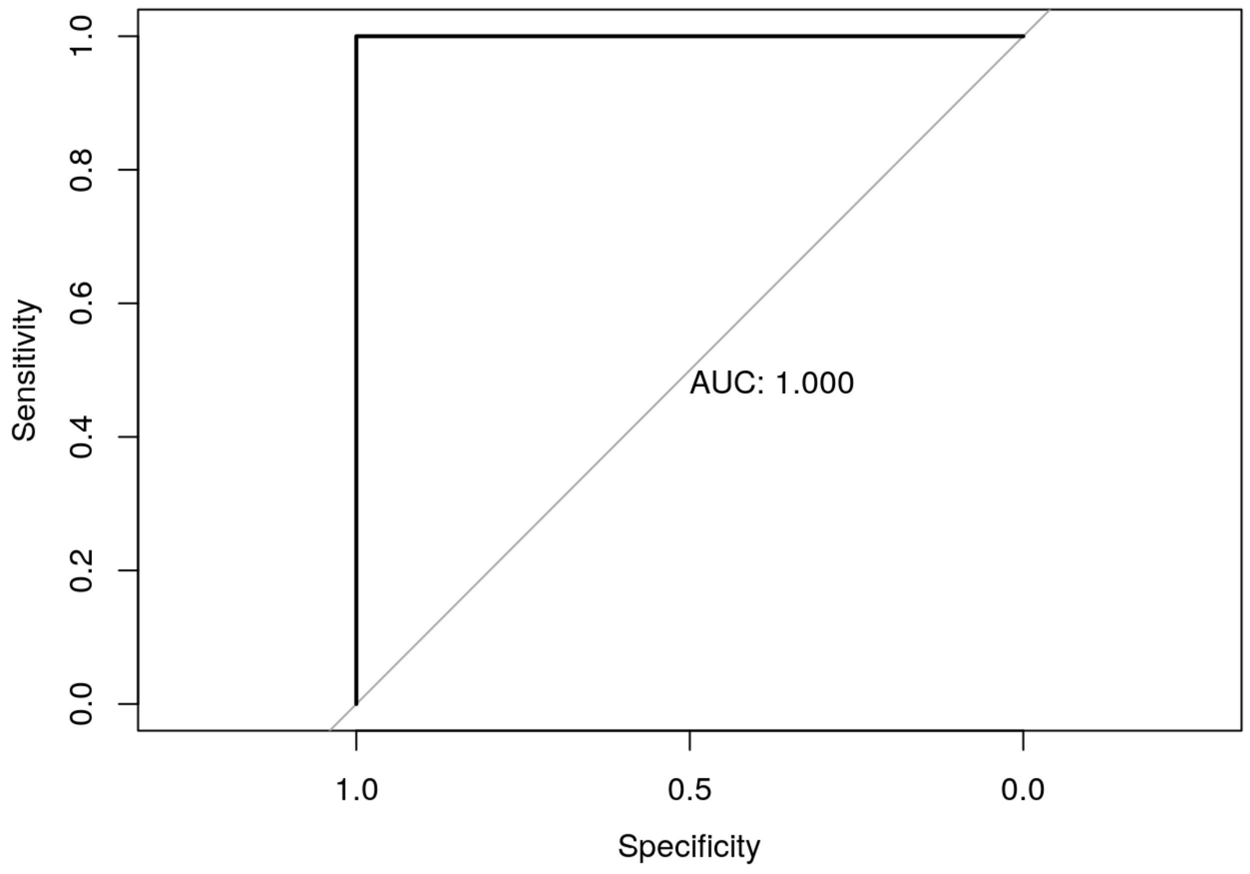
```
# fit a CART using rpart
cancer_rpart <- rpart(diagnosis ~ ., cancerdata)
cancer_predict_rp <- predict(cancer_rpart, cancerdata)
cancer_predict_rp <- ifelse(cancer_predict_rp > 0.5, 1, 0)

CM2 <- confusionMatrix(as.factor(cancer_predict_rp), as.factor(cancerdata$diagnosis))
rp_roc <- roc(cancerdata$diagnosis ~ cancer_predict_rp, plot = TRUE,
    print.auc = TRUE)
```



```
Value <- c(CM2$overall[1], CM2$byClass[1], CM2$byClass[2], AUC = rp_roc$auc)
data.frame(Value)
```

```
##                 Value
## Accuracy    0.9490334
## Sensitivity 0.9663866
## Specificity 0.9198113
## AUC         0.9430989
```

```
# fit a random forest using randomForest
cancer_rf <- randomForest(diagnosis ~ ., cancerdata)
cancer_predict_rf <- predict(cancer_rf, cancerdata)
cancer_predict_rf <- ifelse(cancer_predict_rf > 0.5, 1, 0)
CM3 <- confusionMatrix(as.factor(cancer_predict_rf), as.factor(cancerdata$diagnosis))
rf_roc <- roc(cancerdata$diagnosis ~ cancer_predict_rf, plot = TRUE,
    print.auc = TRUE)
```



```
Value <- c(CM3$overall[1], CM3$byClass[1], CM3$byClass[2], AUC = rf_roc$auc)
data.frame(Value)
```

```
##             Value
## Accuracy        1
## Sensitivity     1
## Specificity     1
## AUC             1
```

When looking at the results of both the CART and random forest fits, it can be seen that both methods result in relatively high accuracy, sensitivity, specificity, and AUC values (above 90%) with random forest having 100% for everything. This implies that these models do a good job in terms of prediction, but just like with logistic regression, there is a chance that the model may be overfitting the data.

Using Training/Test Sets

As such, we will perform the same train/test splits as before to check for that.

```
sensitivity_cart <- c()
specificity_cart <- c()
accuracy_cart <- c()
AUC_cart <- c()

sensitivity_rf <- c()
specificity_rf <- c()
accuracy_rf <- c()
AUC_rf <- c()

for (i in 1:length(seeds)) {
    set.seed(seeds[i])
    idx_train1 <- createDataPartition(cancerdata$diagnosis, p = 0.8)[[1]]
    data_train1 <- cancerdata[idx_train1, ]
    data_test1 <- cancerdata[-idx_train1, ]

    # CART
    cancer_rpart <- rpart(diagnosis ~ ., data_train1)
    cancer_predict_rp <- predict(cancer_rpart, data_test1)
    cancer_predict_rp <- ifelse(cancer_predict_rp > 0.5, 1, 0)

    result <- confusionMatrix(as.factor(cancer_predict_rp), as.factor(data_test1$diagnosi
s))
    accuracy_cart[i] <- result$overall["Accuracy"]
    sensitivity_cart[i] <- sensitivity(as.factor(data_test1$diagnosis),
        as.factor(cancer_predict_rp))
    specificity_cart[i] <- specificity(as.factor(data_test1$diagnosis),
        as.factor(cancer_predict_rp))
    AUC_cart[i] <- auc(data_test1$diagnosis, cancer_predict_rp)

    # Random Forest
    cancer_rf <- randomForest(diagnosis ~ ., data_train1)
    cancer_predict_rf <- predict(cancer_rf, data_test1)
    cancer_predict_rf <- ifelse(cancer_predict_rf > 0.5, 1, 0)

    result <- confusionMatrix(as.factor(cancer_predict_rf), as.factor(data_test1$diagnosi
s))
    accuracy_rf[i] <- result$overall["Accuracy"]
    sensitivity_rf[i] <- sensitivity(as.factor(data_test1$diagnosis),
        as.factor(cancer_predict_rf))
    specificity_rf[i] <- specificity(as.factor(data_test1$diagnosis),
        as.factor(cancer_predict_rf))
    AUC_rf[i] <- auc(data_test1$diagnosis, cancer_predict_rf)
}

# CART
cart_a <- mean(accuracy_cart)
cart_b <- mean(sensitivity_cart)
cart_c <- mean(specificity_cart)
cart_d <- mean(AUC_cart)

Name <- c("Accuracy", "Sensitivity", "Specificity", "AUC")
```

```
Value <- c(cart_a, cart_b, cart_c, cart_d)
data.frame(Name, Value)
```

```
##          Name     Value
## 1    Accuracy 0.9362832
## 2 Sensitivity 0.9498095
## 3 Specificity 0.9171446
## 4         AUC 0.9318733
```

```
# Random Forest
rf1 <- mean(accuracy_rf)
rf2 <- mean(sensitivity_rf)
rf3 <- mean(specificity_rf)
rf4 <- mean(AUC_rf)

Name <- c("Accuracy", "Sensitivity", "Specificity", "AUC")
Value <- c(rf1, rf2, rf3, rf4)
data.frame(Name, Value)
```

```
##          Name     Value
## 1    Accuracy 0.9469027
## 2 Sensitivity 0.9498664
## 3 Specificity 0.9419980
## 4         AUC 0.9411824
```

Just like with the logistic regression model, performing a train/split test results in lower overall performance. Despite this, our models still do a good job of predicting new observations as they all have values above 90% still, just not as high as before. As such, we do not really see any obvious signs of overfitting here either. When comparing the performance of these tree-based methods with those of the logistic regression, it can be seen that all of these methods have relatively similar performance, all above 90% (around low 90s). Though if we look at it more precisely, the tree-based methods seem to perform slightly better than the logistic regression by about 1-2%.

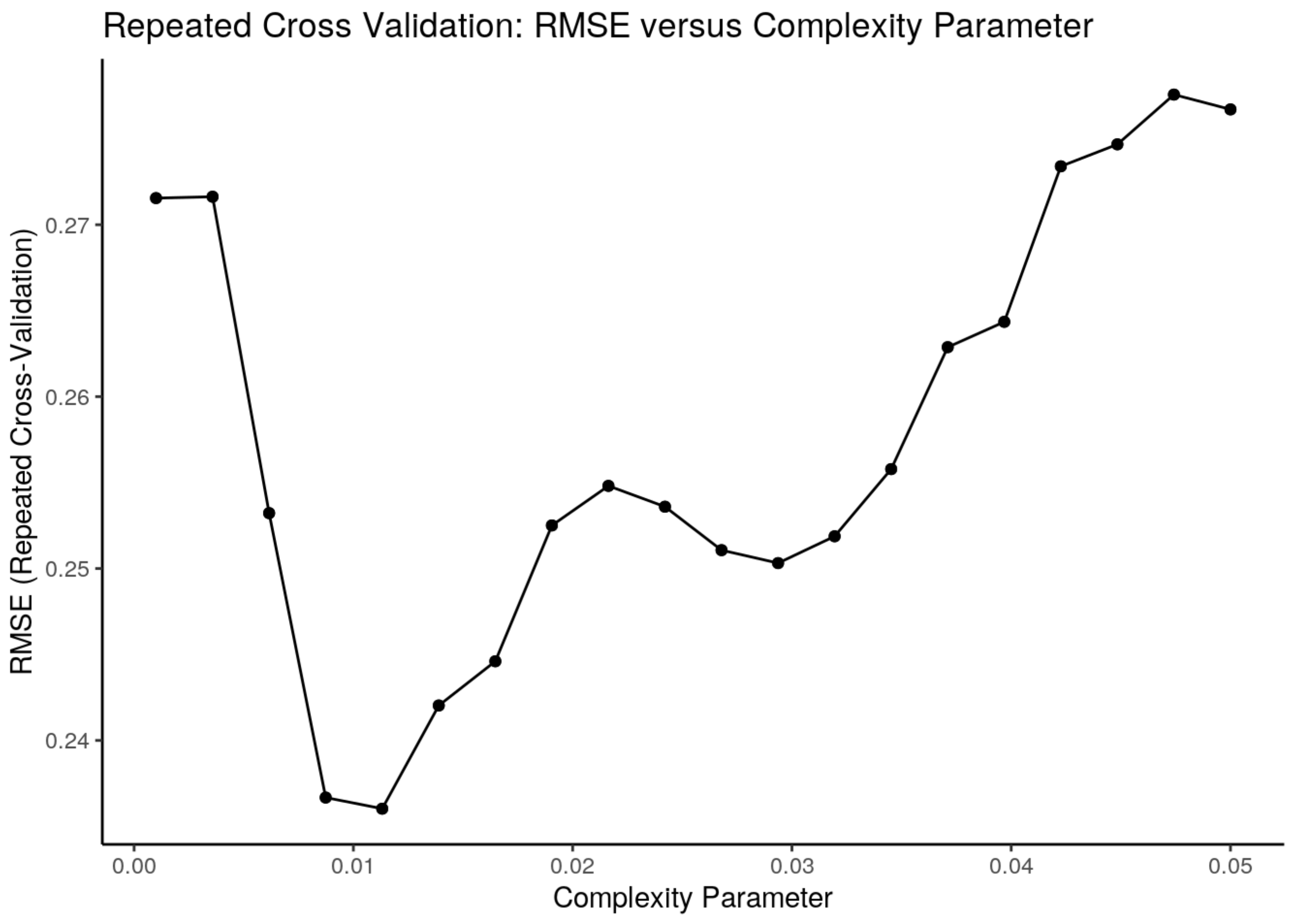
Choose Complexity Parameter

In order to improve the accuracy of our CART fit, we want to choose an appropriate complexity parameter (cp) that achieves the lowest root-mean-square error (RMSE). To do so, we utilize a repeated k-fold cross-validation method to repeatedly test over multiple cp values and determine which cp had the best performance.

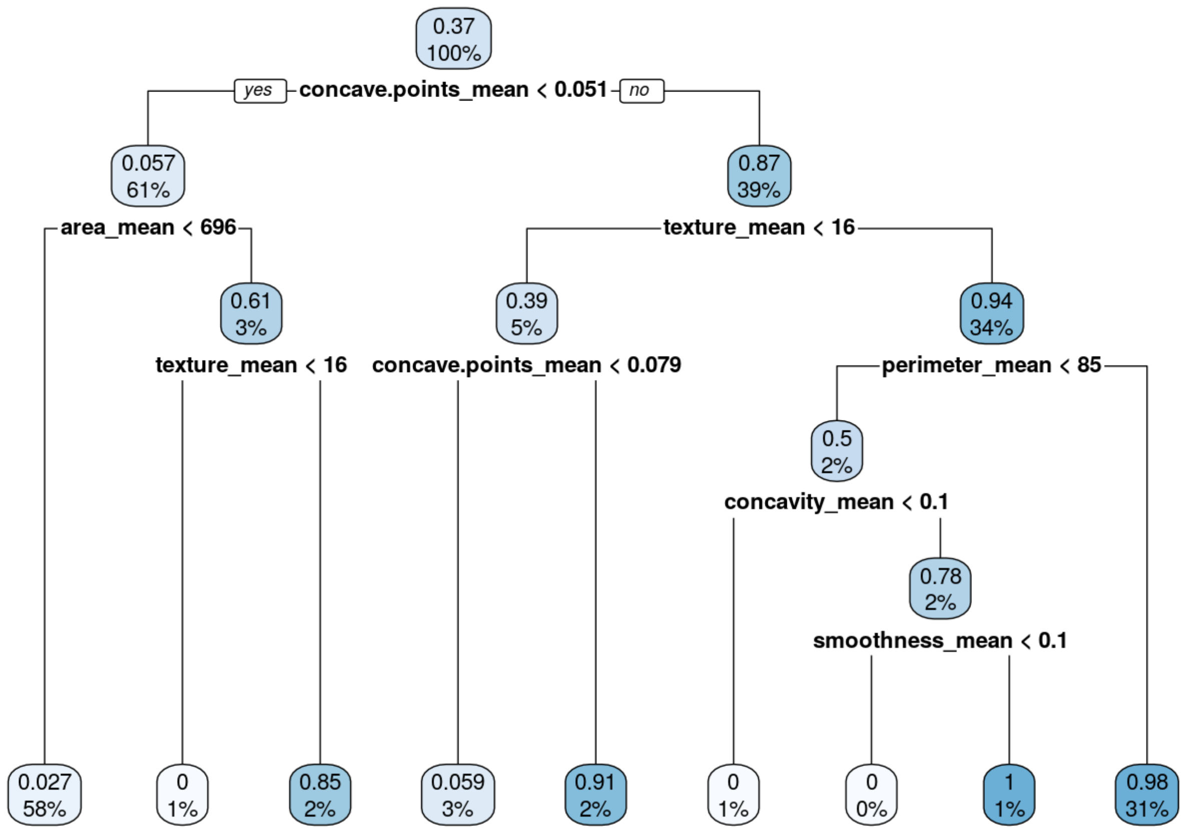
```
possible_cps <- data.frame(cp = seq(from = 0.001, to = 0.05,
    length = 20))
control <- rpart.control(minsplit = 2)
train_control <- trainControl(method = "repeatedcv", number = 20,
    repeats = 10)
tuned_rpart <- train(diagnosis ~ ., method = "rpart", data = cancerdata,
    trControl = train_control, control = control, tuneGrid = expand.grid(cp = possible_cp
s))
tuned_rpart$results %>%
    select(cp, RMSE) %>%
    slice_min(RMSE)
```

```
##          cp      RMSE
## 1 0.01131579 0.2360304
```

```
ggplot(tuned_rpart, aes(x = cp, y = RMSE)) + geom_point() + ggtitle("Repeated Cross Valida
tion: RMSE versus Complexity Parameter") +
    theme_classic()
```



```
rpart.plot(tuned_rpart$finalModel)
```



After running everything, it can be seen that the best cp with the lowest RMSE was 0.008736842, producing the tree shown above. When analyzing the tree, we see that it chooses to highlight the variables concave.points_mean, area_mean, texture_mean, perimeter_mean, concavity_mean, and smoothness_mean. This suggests that these variables are the most important for predicting the diagnosis of breast cancer. Going through the tree, it can be seen that whenever any one of these values is greater than the value shown, there is a high probability that the tumor is malignant. This trend continues as one goes down the tree until the very end. Similarly, if one of the values for these variables is less than the value indicated, there is a higher probability of the tumor being benign instead. This all demonstrates that the tree does its classifications through a numerical value basis, classifying the tumors as malignant when the values are high and benign when they are low.

Regression/Numeric Prediction

While the supervised machine learning algorithms we have used thus far have been focused on classification, some of these methods can be used for numeric prediction as well. In this section, we aim to utilize regression methods to predict the numerical variable: area_mean. By doing so, we aim to see whether a relationship between this variable exists with the other variables in the data set. We start by first performing a simple linear regression analysis on our data and calculating the RMSE to gauge the reliability and accuracy of it.

```
cancerdata2 <- cancerdata %>%
    select(-diagnosis)

model <- lm(area_mean ~ ., cancerdata2)
predict_value <- predict(model, cancerdata2)
lmRMSE <- sqrt(mean((cancerdata2$area_mean - predict_value)^2))
lmRMSE
```

```
## [1] 46.86648
```

After running everything, the calculated RMSE was 46.86648, which is rather high. This implies that our linear model is rather unreliable at predicting area_mean. As such, we sought to perform a k-fold cross validation on the model in an attempt to produce better results.

```
set.seed(12390)
tr_control <- trainControl(method = "cv", number = 10)

my_fit <- train(area_mean ~ ., method = "lm", data = cancerdata2,
    trControl = tr_control)
my_fit$results[2]
```

```
##       RMSE
## 1 48.07036
```

However, after running the code and calculating the new RMSE value, we see that it is actually higher than before (47.98723). This implies that no matter what we do, a linear regression is just not complex enough of a model to properly predict area_mean with our current variables. As such, we opt to use random forest instead in an attempt to increase the complexity of our model.

```
set.seed(12390)
my_fit2 <- train(area_mean ~ ., method = "rf", data = cancerdata2,
    trControl = tr_control)



name <- c("Linear Regression", "Linear Regression with k-fold CV",
    "Random Forest with k-fold CV")
RMSE <- c(lmRMSE, my_fit$results[, 2], min(my_fit2$results[2]))
data.frame(name, RMSE)
```

```
##                               name     RMSE
## 1                Linear Regression 46.86648
## 2 Linear Regression with k-fold CV 48.07036
## 3     Random Forest with k-fold CV 22.66567
```

As shown above, while the RMSE for the random forest is not the best (22.66567), it is significantly better than that of the linear regression as it is about half the amount when using k-fold cross-validation. As such we can conclude that between these two methods, the random forest seems to perform best at predicting the numerical variable area_mean.

Unsupervised Learning

All of the methods we have worked with thus far have utilized known data to make relatively accurate predictive models for classifying breast cancer diagnosis, but what if there is something more to our data that isn’t that obvious? We have had a lot of control over what variables we want to use and what to predict, but to really find something interesting about the structure of our data, we want to utilize unsupervised learning methods instead.

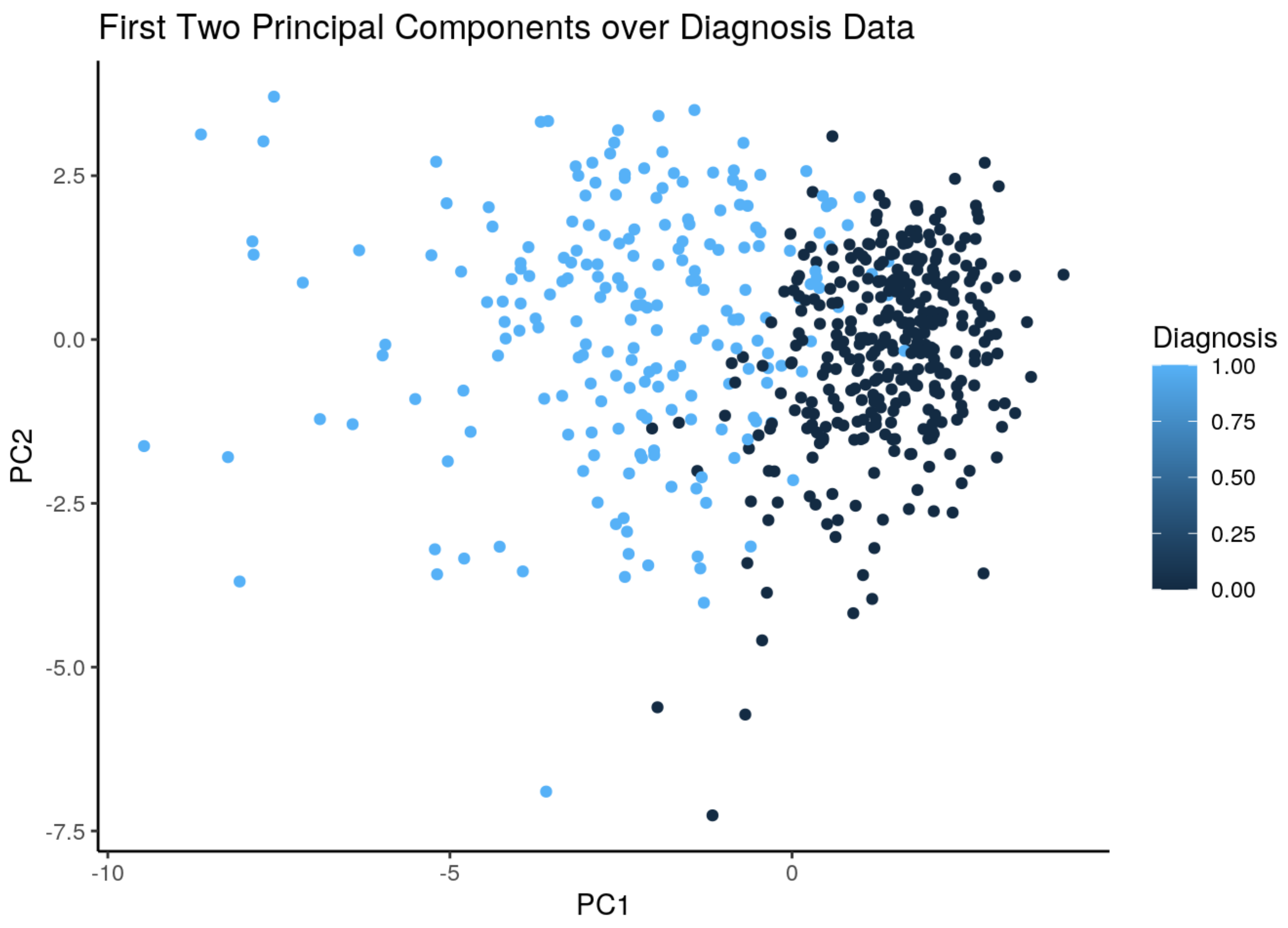
Principal Component Analysis

The unsupervised learning method we opted to use was principal component analysis (PCA), a common dimensionality reduction method. This method goes through all variables in the data set to find which variables are most closely correlated and pairs them up, reducing the total number of variables, or dimensions, necessary to predict and represent the data. To start, we first performed PCA on all of the numeric variables in our data set.

```
## take out the factor variable
cancer_data <- cancerdata %>%
    select(-diagnosis)

## perform PCA for the rest of the variable
cancer_pca <- cancer_data %>%
    scale() %>%
    prcomp()
cancer_pca$rotation[, 1:2] %>%
    data.frame()
```

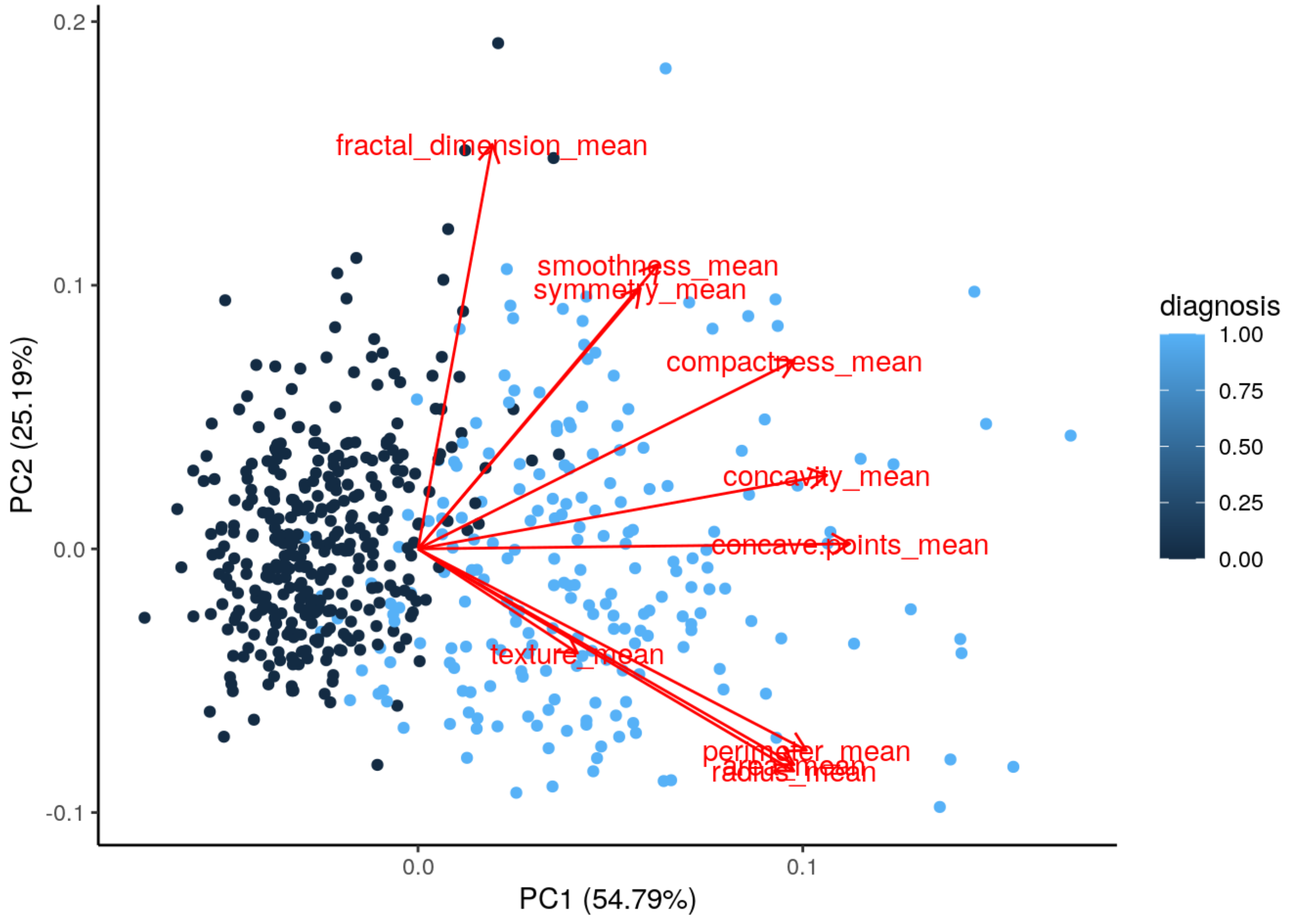
```
##                                PC1          PC2
## radius_mean            -0.36393793  0.313929073
## texture_mean           -0.15445113  0.147180909
## perimeter_mean         -0.37604434  0.284657885
## area_mean              -0.36408585  0.304841714
## smoothness_mean        -0.23248053 -0.401962324
## compactness_mean       -0.36444206 -0.266013147
## concavity_mean         -0.39574849 -0.104285968
## concave.points_mean    -0.41803840 -0.007183605
## symmetry_mean          -0.21523797 -0.368300910
## fractal_dimension_mean -0.07183744 -0.571767700
```

```
## we then can visualize the first two PCs and the data
pca_data <- data.frame(cancer_pca$x, Diagnosis = cancerdata$diagnosis)

ggplot(pca_data, aes(x = PC1, y = PC2, color = Diagnosis)) +
    geom_point(alpha = 2) + ggtitle("First Two Principal Components over Diagnosis Data")
 +
    theme_classic()
```



Here we can see that there are ten distinct principal components. This is to be expected because there are in general min(n - 1,p) informative principal component in a data set with n observations and p variables. Diagnosis is clearly associated with the principal components.

```
## Here we decided to flip the rotation matrix so that we
## can easily see how the PCs change as the variables are
## increased.
cancer_pca$rotation <- -cancer_pca$rotation
cancer_pca$x <- -cancer_pca$x

autoplot(cancer_pca, data = cancerdata, colour = "diagnosis",
    loadings = TRUE, loadings.label = TRUE, cex = 0.1) + theme_classic()
```



First, we can clearly see that radius, perimeter, area and texture are highly correlated; symmetry and smoothness are also highly correlated. Also, diagnosis is associated with PC1.

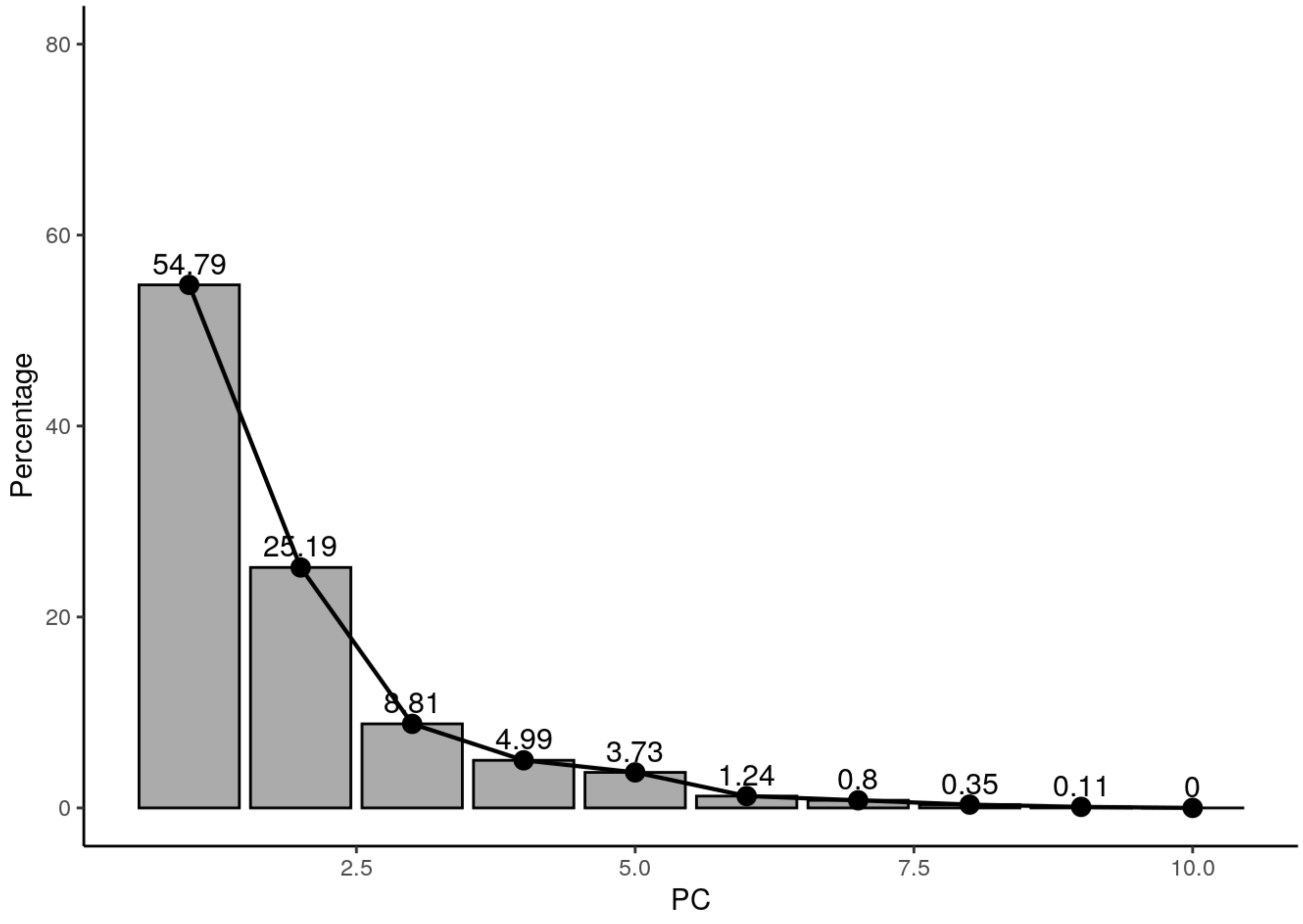
Next, we can see that variable concave point has, by far, the largest loading on the first principal component because they have the highest variance among the ten variable, while the second principal component loading vector places most of its weight on fractal dimension, smoothness, and symmetry. This makes PC1 a principal component that seems to focus more on the shape/physical attributes of the cell nuclei, while PC2 focuses more on the complexity of it.

As we observed, we can see that:

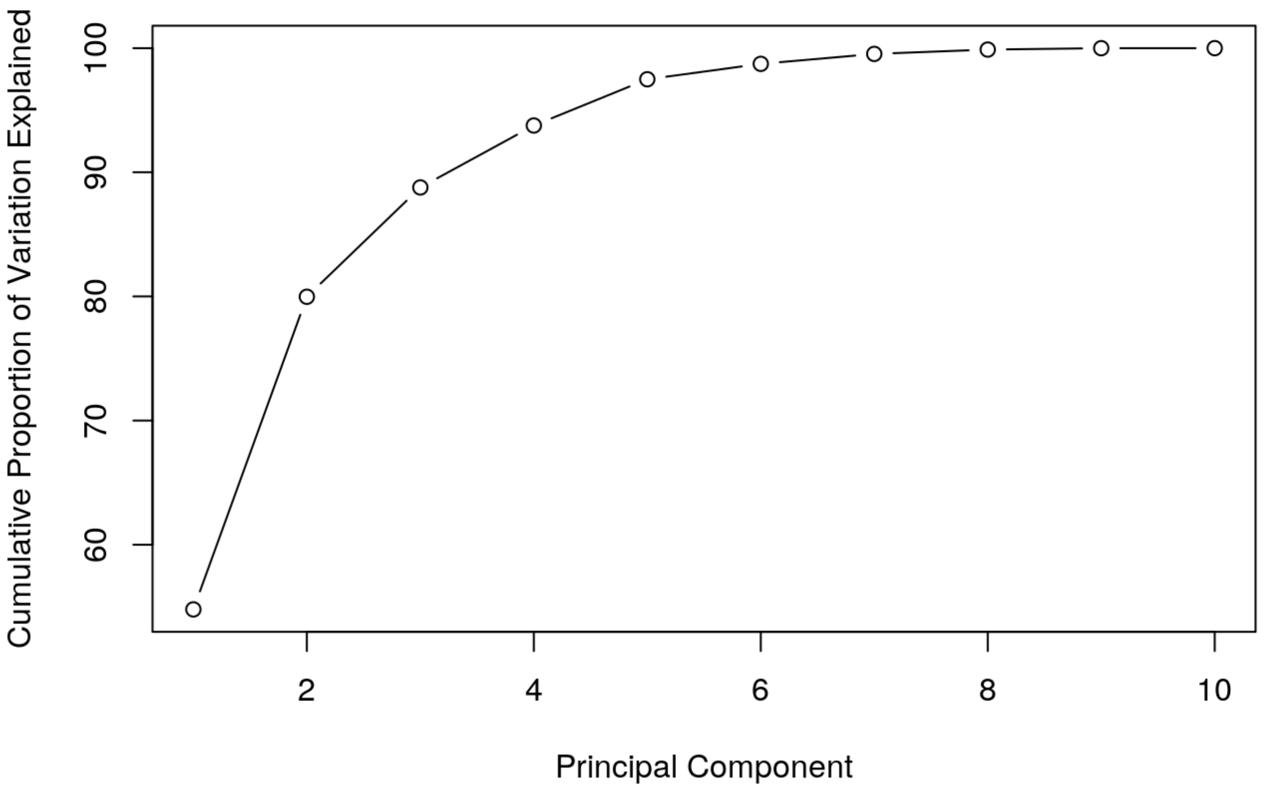
- When we increase the concavity and concave point, it increases the first principal component and has almost no effect on the second principal component at all.
- When we increase the fractal dimension, smoothness, symmetry, and compactness, it increases both the first and second principal components. However, increasing the fractal dimension on its own only marginally increases the first principal component.
- When we increase the texture, perimeter, radius, and area, it increases the first principal component and the second principal component as well.

Hence, we conclude that the first PC is associated with the overall “physical attribution,” those with a larger PC1 tend to have a larger concavity and concave point. On the other hand, the second PC is associated with the overall “complexity,” those with a larger PC2 tend to be more complicated.

```
## Here, we want to look at the percent variance explained.
## The `prcomp()` function gives us standard deviations. To
## convert them into percent variance explained, we square
## them and then divide by the sum over all squared
## standard deviations:
percent <- 100 * cancer_pca$sdev^2/sum(cancer_pca$sdev^2)

perc_data <- data.frame(Percent = percent, PC = 1:length(percent))
ggplot(perc_data, aes(x = PC, y = percent)) + geom_col(alpha = 0.5,
    color = "black") + geom_line(size = 0.75) + geom_point(size = 3) +
    geom_text(aes(label = round(percent, 2)), size = 4, vjust = -0.5) +
    ylim(0, 80) + theme_classic() + ylab("Percentage")
```



```
## We can plot the cumulative PVE as well:
plot(cumsum(percent), xlab = "Principal Component", ylab = "Cumulative Proportion of Varia
tion Explained",
    type = "b", theme = "classic")
```



Here, we see that the first component accounts for 54.79% of the variance in the data, the second principal component accounts for 25.19% of the variance, the third 9%, the fourth 5%, the fifth 4%, the sixth 1%, and the rest make up the remaining 1% of the variance in total. By retaining the first 5 PCs, we account for at least 97% of the variance in the data.

Hence, we can transforms high-dimensions (with 10 PCs) data into lower-dimensions (5 PCs) while retaining as much information as possible.

Concluding Remarks

With all of our analysis completed, it can be seen that the effectiveness of these machine learning models in the field of medical diagnosis can prove to be extremely useful if the appropriate data and training is provided. Both the logistic regression and tree-based classifiers have shown to be effective methods of classifying data accurately, and PCA has shown how we could have simplified our models by reducing the dimensions and variables while still accounting for a majority of the variance in the data. These small analyses demonstrate that these methods are effective at classifying breast cancer data, a small subset of the medical field as a whole. If we take these same machine learning methods and expand their use to apply them to even more complex medical data, we could discover trends never seen before or even create a predictive model capable of accurately diagnosing complicated diseases. The applications of this technology in such a useful field are endless, and this report demonstrates just one simple application of it in such a specific portion of the field.