Assignment 2 submission

Anya Wongreantong, Willem Barnard, Dhvani Vyas

2024-10-16

Perform exploratory analysis of the measured features and see if we can design machine learning tools to separate the benign (B) and malignant (M) cells.

```
#load the data

ovarian.dataset <- read.delim("ovarian.data", sep=",", header = FALSE)
features <- c("perimeter", "area", "smoothness", "symmetry", "concavity", paste("protein", seq(1, 25), s
names(ovarian.dataset) <- c("cell_id", "diagnosis", features)

head(ovarian.dataset)
```

```
##    cell_id diagnosis perimeter  area smoothness symmetry concavity protein1
## 1 85382601         M     17.02 23.98     112.80    899.3   0.11970  0.14960
## 2 88147102         B     15.00 15.51      97.45    684.5   0.08371  0.10960
## 3   889403         M     15.61 19.38     100.00    758.6   0.07840  0.05616
## 4  9112085         B     13.38 30.72      86.34    557.2   0.09245  0.07426
## 5   853612         M     11.84 18.70      77.93    440.6   0.11090  0.15160
## 6   909411         B     10.97 17.20      71.73    371.5   0.08915  0.11130
##   protein2 protein3 protein4 protein5 protein6 protein7 protein8 protein9
## 1  0.24170  0.12030   0.2248  0.06382   0.6009   1.3980    3.999    67.78
## 2  0.06505  0.03780   0.1881  0.05907   0.2318   0.4966    2.276    19.88
## 3  0.04209  0.02847   0.1547  0.05443   0.2298   0.9988    1.534    22.18
## 4  0.02819  0.03264   0.1375  0.06016   0.3408   1.9240    2.287    28.93
## 5  0.12180  0.05182   0.2301  0.07799   0.4825   1.0300    3.475    41.00
## 6  0.09457  0.03613   0.1489  0.06640   0.2574   1.3760    2.806    18.15
##   protein10 protein11 protein12 protein13 protein14 protein15 protein16
## 1  0.008268  0.030820  0.050420  0.011120   0.02102  0.003854     20.88
## 2  0.004119  0.032070  0.036440  0.011550   0.01391  0.003204     16.41
## 3  0.002826  0.009105  0.013110  0.005174   0.01013  0.001345     17.91
## 4  0.005841  0.012460  0.007936  0.009128   0.01564  0.002985     15.05
## 5  0.005551  0.034140  0.042050  0.010440   0.02273  0.005667     16.82
## 6  0.008565  0.046380  0.064300  0.017680   0.01516  0.004976     12.36
##   protein17 protein18 protein19 protein20 protein21 protein22 protein23
## 1     32.09    136.10    1344.0    0.1634    0.3559   0.55880   0.18470
## 2     19.31    114.20     808.2    0.1136    0.3627   0.34020   0.13790
## 3     31.67    115.90     988.6    0.1084    0.1807   0.22600   0.08568
## 4     41.61     96.69     705.6    0.1172    0.1421   0.07003   0.07763
## 5     28.12    119.40     888.7    0.1637    0.5775   0.69560   0.15460
## 6     26.87     90.14     476.4    0.1391    0.4082   0.47790   0.15550
##   protein24 protein25
## 1    0.3530   0.08482
## 2    0.2954   0.08362
```

```
## 3    0.2683   0.06829
## 4    0.2196   0.07675
## 5    0.4761   0.14020
## 6    0.2540   0.09532
```

Question 1 dimensionality reduction

Q1.1. Perform PCA on the features of the dataset. How much of the variation in the data is associated with PC1?

```
ovarian.pca <- prcomp(ovarian.dataset[features], center = TRUE, scale. = TRUE)
summary(ovarian.pca)$importance[2,1]
```

```
## [1] 0.42768
```

Q1.2. You want to represent 90% of the variance in the data by dimensionality reduction. How many PCs do you need to achieve this? In other word, what would be the dimensionality of the reduced feature space so that you preserve 90% of the variability in the data?

```
variance <- summary(ovarian.pca)$importance[2,]
which(cumsum(variance) >= 0.90)[1]
```

```
## PC9
##   9
```

Q1.3. As you should know by now, PCA transforms the data into a new space. In a 2-D plot, can you plot the observations corresponding to the first two important PCs? Note, use two different colors to represent the two classes of cells.

```
# install.packages("devtools")
library(devtools)
```

```
## Loading required package: usethis
```

```
## Warning: package 'usethis' was built under R version 4.3.3
```

```
# install.packages("remotes")
remotes::install_github("vqv/ggbiplot")
```

```
## Skipping install of 'ggbiplot' from a github remote, the SHA1 (f7ea76da) has not changed since last i
##   Use `force = TRUE` to force installation
```

```
library(ggbiplot)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: plyr
```

```
## Loading required package: scales
```

```
## Loading required package: grid
```

```
ggbiplot(ovarian.pca, choices = c(1,2), obs.scale = 1, var.scale = 1,
         groups = ovarian.dataset$diagnosis, ellipse = TRUE, circle = TRUE) +
  scale_color_manual(values = c("royalblue", "plum"))
```

Q1.4. Can you plot the "area" and "concavity" features associated with the cells?

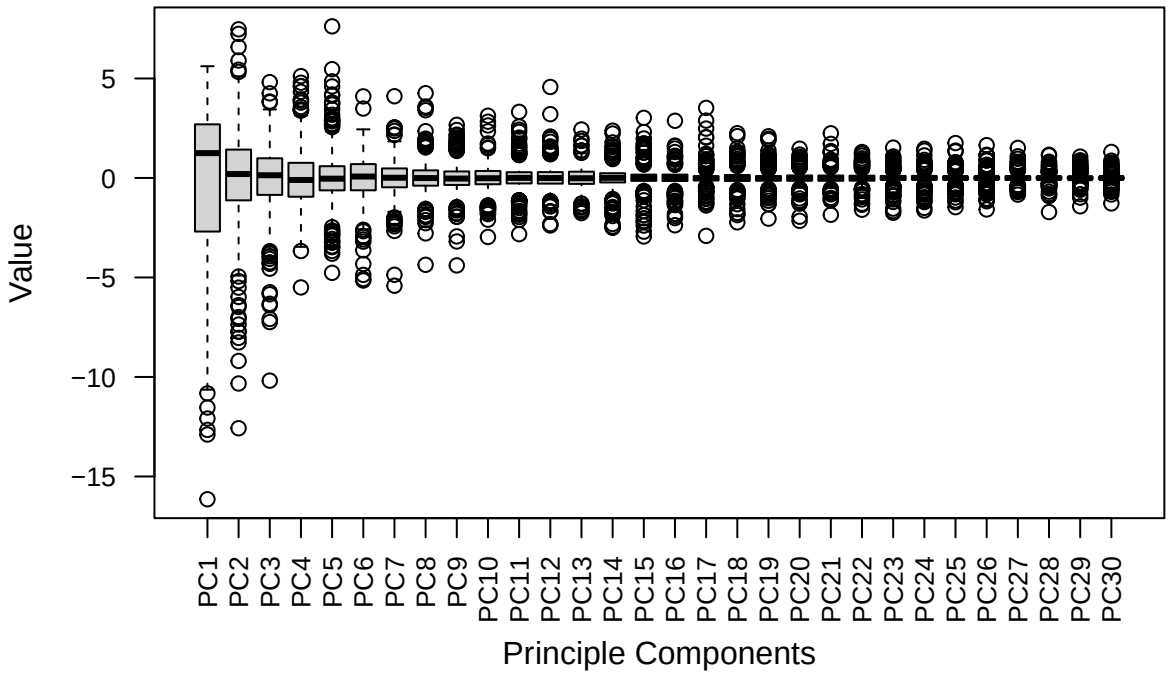
```
library(ggplot2)

ggplot(ovarian.dataset, aes(x = area, y = concavity,
                            color = as.factor(diagnosis))) +
  geom_point(size = 1) +
  labs(x = "Area", y = "Concavity", color = "Diagnosis") +
  scale_color_manual(values = c("springgreen", "hotpink"))
```

Q1.5. What is the difference between the two plots? Which one gives you better separation between the classes and why?

PCA plot provides better separation between classes because it maximizes variance across multiple features. While area vs concavity plot is limited to only 2 original features. PCA is more effective for visualizing class differences for multiple features in class separation.

Q1.6. (bonus): Plot the distribution of the PCs. Hint: you can use boxplot on the transformed dataset.

```
boxplot(ovarian.pca$x[,c(1:30)], xlab = "Principle Components", ylab = "Value",
        las = 2, cex.axis = 0.8)
```



Question 2 Clustering

Q2.1. Apply kmeans clustering on the data and identify two clusters within your dataset. What is the concordance between the clusters that you have identified and the true labels of the cells (Benign vs Malignant).

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
#scale the data
ovarian_scaled <- scale(ovarian.dataset[, c(1,3:32)])
```

```r
# Apply k-means
set.seed(786)
kmeans_result <- kmeans(ovarian_scaled, centers = 2, nstart = 25)

# Map clusters to diagnosis labels
predicted_labels <- ifelse(kmeans_result$cluster == 1, "M", "B")

table(Cluster = predicted_labels, ovarian.dataset$diagnosis)
```

```
##
## Cluster   B   M
##       B 371  35
##       M  14 205
```

Q2.2. Repeat the kmeans analysis 10 times and report the mean accuracy across the 10 runs. Why are the results different in each run?

mean accuracy across 10 runs: 0.9216 K-means clustering starts with randomly chosen centroids, which can lead to different final clusters due to varying initial points. This randomness causes variations in clustering outcomes and thus different accuracies for each run.

```r
accuracies <- numeric(10)

for (i in 1:10) {
  # Run k-means with 2 clusters
  kmeans_result <- kmeans(ovarian_scaled, centers = 2, nstart = 25)
  predicted_labels <- ifelse(kmeans_result$cluster == 1, "M", "B")

  # Calculate accuracy by comparing predicted labels to actual labels
  correct_labels <- ifelse(predicted_labels == ovarian.dataset$diagnosis, 1, 0)

  # Calculate accuracy
  accuracy <- sum(correct_labels) / length(correct_labels)

  reverse_accuracy <- sum(1 - correct_labels) / length(correct_labels)
  accuracies[i] <- max(accuracy, reverse_accuracy)
  print(reverse_accuracy)
}
```

```
## [1] 0.0784
## [1] 0.0784
## [1] 0.9216
## [1] 0.9216
## [1] 0.0784
## [1] 0.0784
## [1] 0.0784
## [1] 0.0784
## [1] 0.0784
## [1] 0.9216
```

```r
# Calculate mean accuracy across 10 runs
mean(accuracies)
```

```
## [1] 0.9216
```

Q2.3. Repeat the same analysis but with the top 5 PCs.

mean accuracy across 10 runs with the top 5 PCs: 0.9184

```
accuracies_5PCs <- numeric(10)

for (i in 1:10) {
  # Run k-means with 2 clusters
  set.seed(123)
  kmeans_result <- kmeans(ovarian.pca$x[, 1:5], centers = 2, nstart = 25)
  predicted_labels <- ifelse(kmeans_result$cluster == 1, "M", "B")

  # Calculate accuracy by comparing predicted labels to actual labels
  correct_labels <- ifelse(predicted_labels == ovarian.dataset$diagnosis, 1, 0)

  # Calculate accuracy
  accuracy <- sum(correct_labels) / length(correct_labels)

  reverse_accuracy <- sum(1 - correct_labels) / length(correct_labels)
  accuracies_5PCs[i] <- max(accuracy, reverse_accuracy)
}

# Calculate mean accuracy across 10 runs
mean(accuracies_5PCs)
```

```
## [1] 0.9184
```

Q2.4. Compare the results between Q2.2. and Q2.3.

The mean accuracy is slightly better in Q2.2, this suggests that the addtional principle conponents beyond first 5 contain some information that helps in distinguishing between the clusters.

Question 3 Classification

Q3.1. Design a logistic regression classifier to identify (differentiate) benign and malignant cells. Report the performance of the classification technique on the training and test sets. You can report accuracy, precision and recall. Compare the performance of the classifier on the training and test set and provide a reason as to why one is better than the other.

```
# Convert diagnosis to binary values (1 for Malignant, 0 for Benign)
ovarian.dataset$diagnosis_binary <- ifelse(ovarian.dataset$diagnosis == "M", 1, 0)

# Split the dataset into training and test sets
ovarian.dataset.train <- ovarian.dataset[sample(nrow(ovarian.dataset))[1:(nrow(ovarian.dataset)/2)],]
ovarian.dataset.test <- ovarian.dataset[sample(nrow(ovarian.dataset))[(nrow(ovarian.dataset)/2):(nrow(ov

# Train the logistic regression model on the training set using the binary diagnosis variable
logistic_model <- glm(diagnosis_binary ~ perimeter + area + smoothness + symmetry + concavity +
                      protein1 + protein2 + protein3 + protein4 + protein5,
                      data = ovarian.dataset.train, family = binomial)
```

```r
# Make predictions on the training set
train_pred <- predict(logistic_model, ovarian.dataset.train, type = "response")
train_class <- ifelse(train_pred >= 0.5, "M", "B")

# Make predictions on the test set
test_pred <- predict(logistic_model, ovarian.dataset.test, type = "response")
test_class <- ifelse(test_pred >= 0.5, "M", "B")

# Function to calculate accuracy, precision, and recall
evaluate_performance <- function(true_labels, predictions) {
  tp <- sum(predictions == "M" & true_labels == "M")
  tn <- sum(predictions == "B" & true_labels == "B")
  fp <- sum(predictions == "M" & true_labels == "B")
  fn <- sum(predictions == "B" & true_labels == "M")

  accuracy <- (tp + tn) / length(true_labels)
  precision <- tp / (tp + fp)
  recall <- tp / (tp + fn)

  return(c(accuracy = accuracy, precision = precision, recall = recall))
}

# Evaluate model performance as before
train_performance <- evaluate_performance(ovarian.dataset.train$diagnosis, train_class)
print("Training Performance")
```

```
## [1] "Training Performance"
```

```r
print(train_performance)
```

```
##  accuracy precision    recall
## 0.9551282 0.9545455 0.9210526
```

```r
test_performance <- evaluate_performance(ovarian.dataset.test$diagnosis, test_class)
print("Test Performance")
```

```
## [1] "Test Performance"
```

```r
print(test_performance)
```

```
##  accuracy precision    recall
## 0.9424920 0.9478261 0.9008264
```

Test sets shows a better value in accuracy, precision, and recall. This may be due to logistic regression captures generalizable patterns that align more closely with the broader data distribution seen in the test set.

Q3.2. Repeat the same task as Q3.1. with the top 5 PCs.

```r
# Prepare the data by using the first 5 PCs
ovarian.dataset$PC1 <- ovarian.pca$x[, 1]
ovarian.dataset$PC2 <- ovarian.pca$x[, 2]
ovarian.dataset$PC3 <- ovarian.pca$x[, 3]
ovarian.dataset$PC4 <- ovarian.pca$x[, 4]
ovarian.dataset$PC5 <- ovarian.pca$x[, 5]

# Convert diagnosis to binary values (1 for Malignant, 0 for Benign)
ovarian.dataset$diagnosis_binary <- ifelse(ovarian.dataset$diagnosis == "M", 1, 0)

# Split the dataset into training and test sets
ovarian.dataset.train <- ovarian.dataset[sample(nrow(ovarian.dataset))[1:(nrow(ovarian.dataset)/2)],]
ovarian.dataset.test <- ovarian.dataset[sample(nrow(ovarian.dataset))[(nrow(ovarian.dataset)/2):(nrow(ov

# Train the logistic regression model using the top 5 PCs
logistic_model_pcs <- glm(diagnosis_binary ~ PC1 + PC2 + PC3 + PC4 + PC5,
                          data = ovarian.dataset.train, family = binomial)
```

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

```r
# Make predictions on the training set
train_pred_pcs <- predict(logistic_model_pcs, ovarian.dataset.train, type = "response")
train_class_pcs <- ifelse(train_pred_pcs >= 0.5, "M", "B")

# Make predictions on the test set
test_pred_pcs <- predict(logistic_model_pcs, ovarian.dataset.test, type = "response")
test_class_pcs <- ifelse(test_pred_pcs >= 0.5, "M", "B")

# Evaluate model performance as before
train_performance_pcs <- evaluate_performance(ovarian.dataset.train$diagnosis, train_class_pcs)
print("Training Performance with Top 5 PCs")
```

```
## [1] "Training Performance with Top 5 PCs"
```

```r
print(train_performance_pcs)
```

```
##  accuracy precision    recall
## 0.9839744 0.9830508 0.9747899
```

```r
test_performance_pcs <- evaluate_performance(ovarian.dataset.test$diagnosis, test_class_pcs)
print("Test Performance with Top 5 PCs")
```

```
## [1] "Test Performance with Top 5 PCs"
```

```r
print(test_performance_pcs)
```

```
##  accuracy precision    recall
## 0.9744409 0.9739130 0.9572650
```

Q3.3. Compare the results between Q3.1. and Q3.2. Do the results get better or worse? Why?

Between 3.1 and 3.2, Q3.2 gives a higher amount of accuracy and precision along with a higher recall rate.

The formula of accuracy is as follows: (True + False Positives)/Total Observations The formula of precision is as follows: (True Positives)/(True Positives + False Positives) The formula of recall is as follows: (True Positives)/(True Positives + False Negatives)

As Q3.2 is higher in all three cases, it implies that it has a high level of true positives and a low degree of false positives and false negatives compared to Q3.1.

When PCA testing is done, it reduces the dimensions of the data, thereby removing all the redundant noise and "extra" data in the set, only visualising the most important variability. As this extra data is removed, it allows only the most important of information to remain in the dataset, and therefore increasing both the accuracy and precision as redundant data is not there to "stain" the dataset.

By only including the top five PCAs, the five most important PCAs with the most variability, Q3.3 further only calculates with the most important data, which further leads to more accurate and precise results with high recall.

However, it is important to note that Q3.1's result is also quite high, so it is not a terrible representation of the data, it is just that Q3.2's method gives a more better answer.

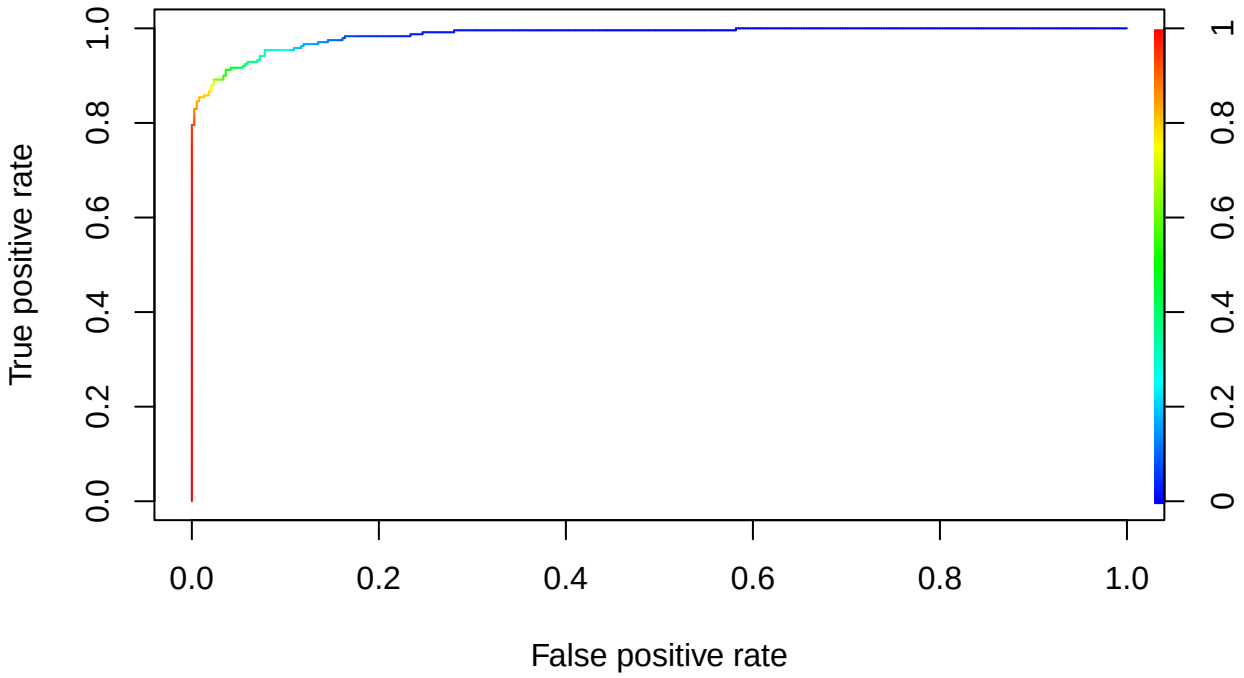
Q3.4. Compare the results of the clustering and classification methods. Which one gives you better result?

Clustering achieved a mean accuracy of around 0.9216 with 10PCs and 0.9184 with 5 PCs Classification had a test accuracy of 0.9585, and using the top 5 PCs improved accuracy further to 0.9776 Classification provides better results because it leverage labeled data to directly optimize the separation between classes.

Q3.5. Install the library "ROCR" and load it. Then, run the following lines using your trained logistic regression model:

The ROC curve indicates strong class separability, with minimal overlap between benign and malignant cells, as evidenced by its proximity to the top-left corner. This suggests that the model effectively distinguishes between classes. Unlike a single sensitivity or specificity measure, the ROC curve offers a comprehensive view of performance across thresholds, helping to visualize trade-offs between true positives and false positives.

```r
#install.packages("ROCR")
library(ROCR)

pred.prob <- predict(logistic_model, ovarian.dataset, type="response")
predict <- prediction(pred.prob, ovarian.dataset$diagnosis, label.ordering=c("B","M"))
perform <- performance(predict,"tpr","fpr")
plot(perform,colorize=TRUE)
```



Q3.6 . Design another classifier (using a different classification method) and repeat Q3.1-3.

```
#install.packages("randomForest")
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.3.3
```

```
## randomForest 4.7-1.2
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
# Convert diagnosis to binary values (1 for Malignant, 0 for Benign)
ovarian.dataset$diagnosis_binary <- ifelse(ovarian.dataset$diagnosis == "M", 1, 0)
# Split the dataset into training and test sets
ovarian.dataset.train <- ovarian.dataset[sample(nrow(ovarian.dataset))[1:(nrow(ovarian.dataset)/2)],]
ovarian.dataset.test <- ovarian.dataset[sample(nrow(ovarian.dataset))[(nrow(ovarian.dataset)/2):(nrow(ov
# Train the logistic regression model on the training set using the binary diagnosis variable
randomForest_model <- randomForest(diagnosis_binary ~ perimeter + area + smoothness + symmetry + concavi
protein1 + protein2 + protein3 + protein4 + protein5,
data = ovarian.dataset.train)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values.  Are you sure you want to do regression?
```

```r
7
```

```
## [1] 7
```

```r
# Make predictions on the training set
train_pred <- predict(logistic_model, ovarian.dataset.train, type = "response")
train_class <- ifelse(train_pred >= 0.5, "M", "B")
# Make predictions on the test set
test_pred <- predict(logistic_model, ovarian.dataset.test, type = "response")
test_class <- ifelse(test_pred >= 0.5, "M", "B")
# Function to calculate accuracy, precision, and recall
evaluate_performance <- function(true_labels, predictions) {
tp <- sum(predictions == "M" & true_labels == "M")
tn <- sum(predictions == "B" & true_labels == "B")
fp <- sum(predictions == "M" & true_labels == "B")
fn <- sum(predictions == "B" & true_labels == "M")
accuracy <- (tp + tn) / length(true_labels)
precision <- tp / (tp + fp)
recall <- tp / (tp + fn)
return(c(accuracy = accuracy, precision = precision, recall = recall))
}
# Evaluate model performance as before
train_performance <- evaluate_performance(ovarian.dataset.train$diagnosis, train_class)
print("Training Performance")
```

```
## [1] "Training Performance"
```

```r
print(train_performance)
```

```
##  accuracy precision    recall
## 0.9262821 0.9354839 0.8854962
```

```r
test_performance <- evaluate_performance(ovarian.dataset.test$diagnosis, test_class)
print("Test Performance")
```

```
## [1] "Test Performance"
```

```r
print(test_performance)
```

```
##  accuracy precision    recall
## 0.9424920 0.9406780 0.9098361
```

```r
# Prepare the data by using the first 5 PCs
ovarian.dataset$PC1 <- ovarian.pca$x[, 1]
ovarian.dataset$PC2 <- ovarian.pca$x[, 2]
ovarian.dataset$PC3 <- ovarian.pca$x[, 3]
ovarian.dataset$PC4 <- ovarian.pca$x[, 4]
ovarian.dataset$PC5 <- ovarian.pca$x[, 5]
# Convert diagnosis to binary values (1 for Malignant, 0 for Benign)
ovarian.dataset$diagnosis_binary <- ifelse(ovarian.dataset$diagnosis == "M", 1, 0)
# Split the dataset into training and test sets
ovarian.dataset.train <- ovarian.dataset[sample(nrow(ovarian.dataset))[1:(nrow(ovarian.dataset)/2)],]
ovarian.dataset.test <- ovarian.dataset[sample(nrow(ovarian.dataset))[(nrow(ovarian.dataset)/2):(nrow(ov
# Train the logistic regression model using the top 5 PCs
randomForest_model_pcs <- randomForest(diagnosis_binary ~ PC1 + PC2 + PC3 + PC4 + PC5,
data = ovarian.dataset.train)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values.  Are you sure you want to do regression?
```

```r
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
# Make predictions on the training set
train_pred_pcs <- predict(logistic_model_pcs, ovarian.dataset.train, type = "response")
train_class_pcs <- ifelse(train_pred_pcs >= 0.5, "M", "B")
# Make predictions on the test set
test_pred_pcs <- predict(logistic_model_pcs, ovarian.dataset.test, type = "response")
test_class_pcs <- ifelse(test_pred_pcs >= 0.5, "M", "B")
# Evaluate model performance as before
train_performance_pcs <- evaluate_performance(ovarian.dataset.train$diagnosis, train_class_pcs)
print("Training Performance with Top 5 PCs")
```

```
## [1] "Training Performance with Top 5 PCs"
```

```r
print(train_performance_pcs)
```

```
##  accuracy precision    recall
## 0.9775641 0.9618321 0.9843750
```

```r
test_performance_pcs <- evaluate_performance(ovarian.dataset.test$diagnosis, test_class_pcs)
print("Test Performance with Top 5 PCs")
```

```
## [1] "Test Performance with Top 5 PCs"
```

```r
print(test_performance_pcs)
```

```
##  accuracy precision    recall
## 0.9776358 0.9541985 0.9920635
```

In this case we used the randomForest classification method which we found when running the training data to have a percision of 100% which means that the number of false positives is equal to zero. This means the training set of data never included any false positive or that is could not find any false positives. We also found the outcome of the two methods to be very similar but the accuracy using the randomForest method was higher by0.3 percent and the precision was lower by 0.8 percent. When comparing this method of classification to the linear regression model we used previously we find that both give us similar outputs this means the difference between the the to methods is with the algorithm its self and but those differences don't affect greatly change the outcome. Some of these differences can include amount of processing power used, number of iterations, efficiency of the algorithm. Either way both of these are viable options for classifying whether we get true/false positives and true/false negatives since they both yield high 90 percent on accuracy, precision and recall. When comparing the test cases of the randomForset model when we use 5 PCs versus when we use all the data we again remove all the redundant noise and are left with higher values in accuracy, precision, and recall. This is nearly identical to the change we see when we remove the noise from the linear regression model except for in recall we see a 7 percent increase using randomForest and only a 3 percent increase using linear regression.

Collaboration Statement All team menbers contribute to the code and profread.