Breast Cancer Diagnostic - Machine Learning

Renz Asprec

2024-11-16

Contents

Executive Summary

This project aims to produce an algorithm that will diagnose breast cancer via R. This project used the breast cancer Wisconsin diagnostic data set from UCI machine learning repository.

This project explored 8 models in total: k-means clustering, logistic regression, lda, qda, loess, k-nearest neighbors, k-nearest neighbors (with dimension reduction), random forest, and ensemble.

The model which produced the best accuracy is the ensemble. The ensemble model was then used to predict the outcomes using the validation set.

Introduction

This project aims to produce an algorithm that will diagnose breast cancer via R. This project used the breast cancer Wisconsin diagnostic data set from UCI machine learning repository.

Dataset

The data used was the breast cancer Wisconsin diagnostic data set (brca) from UCI machine learning repository. It consists of 569 observations with 30 features. Each observation has a corresponding outcome. The outcome is a factor with two levels denoting whether a mass is malignant (M) or benign (B). The features correspond to properties of cell nuclei, such as size, shape and regularity. The mean, standard error, and worst value of each of 10 nuclear parameters is reported for a total of 30 features. The features are shown below:

features
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave_pts_mean
symmetry_mean
fractal_dim_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave_pts_se
symmetry_se
fractal_dim_se
radius_worst
texture_worst
perimeter_worst
area_worst
smoothness_worst
compactness_worst
concavity_worst
concave_pts_worst
symmetry_worst
fractal_dim_worst

Evaluation Metrics

The diagnostic algorithms were evaluated by calculating the accuracy compared to a validation set.

Methodology

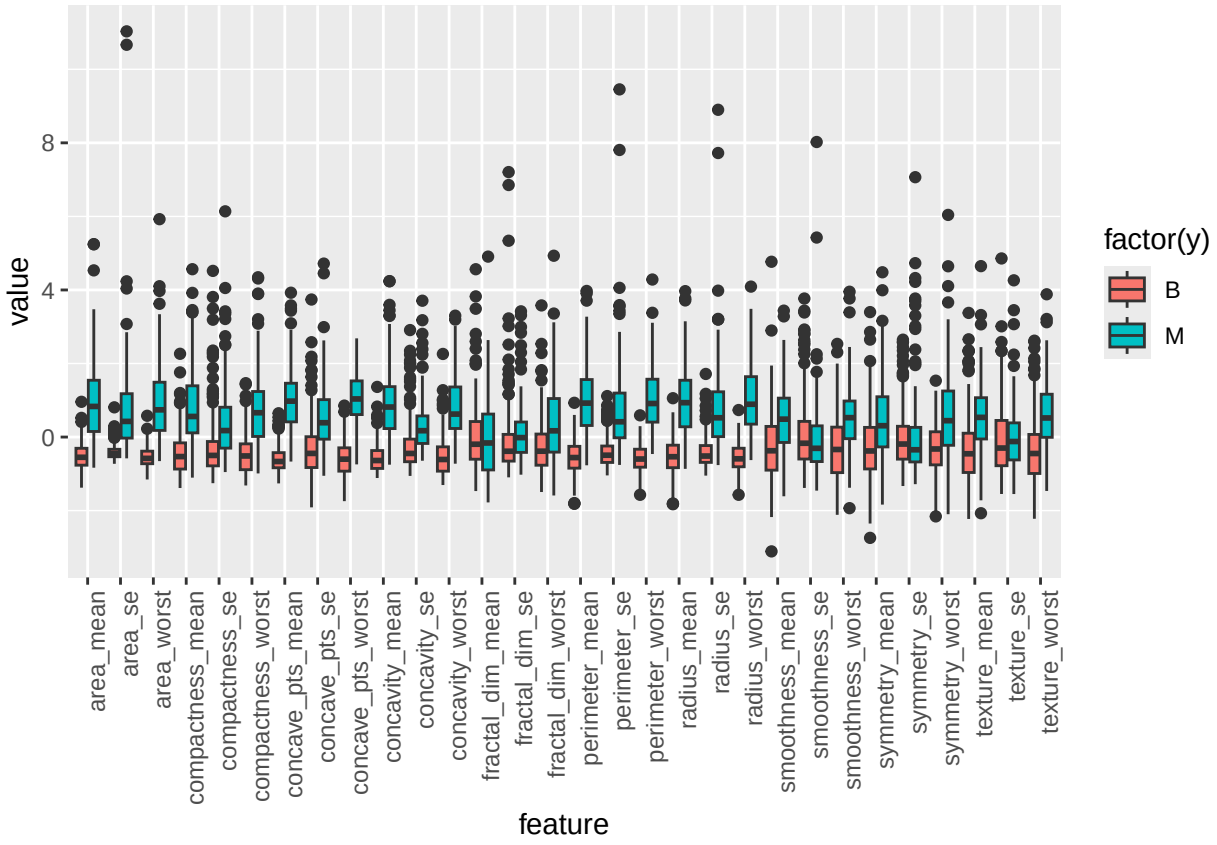
Data Preparation

The breast cancer data (brca) were partitioned into a 20% test, and 80% training data. The data were centered and scaled prior to partitioning.

A 20% validation set was subsequently obtained from the training set.

Data Exploration

A box plot of the values of each feature for each outcome was plotted. The plot is shown below:



From this plot, it can be seen that there are several features without overlapping IQRs. From this, it can be expected that the models produced would generate high accuracies.

Models

The following models were used to generate diagnostic algorithms on the training data:

models
kmeans
glm
lda

models
qda
loess
knn
knn with pca
rf
ensemble

The model that yielded the best accuracy on the test set would be used to evaluate the validation set.

Results

Model Training

The models in this project used 25 bootstrap samples to train.

K-means Clustering

Use k-means to extract 2 centers from the 25 bootstrap samples. The centers correspond to the 2 outcomes, benign (B) and malignant (M). The distances of each observation from the centers were used to predict the outcome.

The model resulted to an accuracy of 0.904.

Logistic Regression

Trained a model using *glm* using all features.

The model resulted to an accuracy of 0.957.

LDA and QDA models

Trained a model using *lda* and *qda* using all features.

The model resulted to accuracies of 0.939 and 0.922, for lda and qda, respectively.

Loess Model

Trained a model using *gamLoess* using all features. The model requires the tuning parameters span, and degree. Default parameters were used.

The best tuning parameters are shown below:

```
##   span degree
## 1  0.5      1
```

The model resulted to an accuracy of 0.974.

K-nearest Neighbors

Trained a model using *knn* using all the features. The model requires the tuning parameter, k, the number of neighbors. The model was trained using the odd values of k between 3 and 21.

The best k is :

```
##    k
## 7 15
```

This model resulted to an accuracy of 0.939.

K-nearest Neighbors Model with Dimension Reduction (PCA)

Applied dimension reduction to the training data and trained a model using *knn*. The training data was reduced using a principal component analysis (pca) pre-processing. The model was also trained using the odd values of k between 3 and 21.

7 principal components accounted for 90% of the variance.

This model resulted to an accuracy of 0.939.

Random Forest Model

Trained a random forest model using *rf* using all features. This model requires a tuning parameter mtry, the number of randomly selected predictors.

```
## Accuracy
##    0.965
```

The number of selected predictors which resulted to the best accuracy is:

```
##   mtry
## 2    5
```

The 5 most important variables are shown below:

```
##                        B     M
## perimeter_worst   100.0 100.0
## concave_pts_worst  96.6  96.6
## area_worst         94.5  94.5
## radius_worst       85.3  85.3
## concave_pts_mean   72.7  72.7
```

The least important variable is:

```
##             B M
## symmetry_se 0 0
```

This model resulted to an accuracy of 0.965.

Ensemble Model

The prediction for the ensemble model is the majority prediction across all the models.

The ensemble of all the models resulted to an accuracy of 0.957.

Validation

The summary of the accuracies for each model is shown below:

model	accuracy
kmeans	0.904
glm	0.957
lda	0.939
qda	0.922
knn	0.939
knn with pca	0.939
rf	0.965
ensemble	0.957

The model which resulted to the best accuracy on the test data is the rf model. This would be the final model used for evaluating the validation set.

The accuracy obtained using the ensemble model on the validation set is 0.945.

Conclusion

This project explored 8 models in total: k-means clustering, logistic regression, lda, qda, loess, k-nearest neighbors, k-nearest neighbors (with dimension reduction), random forest, and ensemble. The model which produced the best accuracy is the ensemble. The ensemble model was then used to predict the outcomes using the validation set. This achieved an accuracy of 0.945.

Recommendations

The diagnostic models generated here only used specific algorithms. It is recommended to explore other algorithms. Moreover, for the ensemble model, it is suggested to explore removing low accuracy models.

Appendix

```r
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(matrixStats)) install.packages("matrixStats", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(dslabs)) install.packages("dslabs", repos = "http://cran.us.r-project.org")
if(!require(knitr)) install.packages("knitr", repos = "http://cran.us.r-project.org")
if(!require(doParallel)) install.packages("doParallel", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(matrixStats)
library(caret)
library(dslabs)
library(knitr)
library(doParallel)
options(digits = 3)
data(brca)

# for parallel computation
library(doParallel)
nc <- detectCores() - 1
cl <- makeCluster(nc)
registerDoParallel(cl)

# number of observations in the dataset
length(brca$y)

# number of features
ncol(brca$x)

# list of features
kable(colnames(brca$x), col.names = "features")

# models to be used for the breast cancer diagnostic algorithm
models<- c("kmeans","glm","lda","qda","loess", "knn","knn with pca", "rf", "ensemble")
kable(models,col.names = "models")

#### Preprocessing ####
# pre-processing of data by centering and scaling
pp<- preProcess(x = brca$x,method = c("center","scale"))
x_scaled<- predict(object = pp,newdata = brca$x)

#### Data Preparation ####
# partition data into a train and test set
set.seed(1996)
test_index <- createDataPartition(brca$y, times = 1, p = 0.2, list = FALSE)
test_x <- x_scaled[test_index,]
test_y <- brca$y[test_index]
train_x <- x_scaled[-test_index,]
train_y <- brca$y[-test_index]

# obtain validation set from the train set. The validation set would be used to calculate the final acc
set.seed(1996)
```

```r
validation_index <- createDataPartition(train_y, times = 1, p = 0.2, list = FALSE)
validation_x<- train_x[validation_index,]
validation_y<- train_y[validation_index]
train_x<- train_x[-validation_index,]
train_y <- train_y[-validation_index]

#### Data Exploration ####
# boxplot of each feature for each outcome
train_x |>
  data.frame() |>
  mutate(y=train_y) |>
  pivot_longer(-y,names_to = "feature",values_to = "value") |>
  ggplot() +
  geom_boxplot(aes(feature,value,fill=factor(y))) +
  theme(axis.text.x = element_text(angle = 90,hjust = 1))


#### Model Training ####

##### Training a k-means model #####

# define a function which takes two arguments - a matrix of observations x and a k-means object k - and

predict_kmeans <- function(x, k) {
  centers <- k$centers    # extract cluster centers
  # calculate distance to cluster centers
  distances <- sapply(1:nrow(x), function(i){
    apply(centers, 1, function(y) dist(rbind(x[i,], y)))
  })
  max.col(-t(distances))  # select cluster with min distance to center
}

# Create 25 bootstrap samples
set.seed(1996)
# a single vector of bootstrap indexes
bootstrap_indexes<- createResample(y = train_y,times = 25,list = FALSE) |> as.vector()

bootstrap_x<- train_x[bootstrap_indexes,]

# Use k-means to extract 2 centers from the 25 bootstrap samples. The centers correspond to the 2 outco
set.seed(1996)
k<- kmeans(bootstrap_x,centers = 2)

# predict using the predict_kmeans(function)
y_hat_kmeans<- predict_kmeans(x = test_x,k = k)
y_hat_kmeans<- ifelse(y_hat_kmeans==1,                        "M","B") |>
  factor(levels=levels(test_y))

# accuracy of k-means model
accuracy_kmeans<- confusionMatrix(data = y_hat_kmeans,reference = test_y)$overall["Accuracy"]

##### Logistic regression model #####
# Training of a logistic regression model using all predictors
```

```r
set.seed(1996)
fit_glm<- train(x = train_x,y = train_y,method = "glm")

# prediction with glm model
y_hat_glm<- predict(object = fit_glm,newdata = test_x) |>
  factor(levels=levels(test_y))

# accuracy of glm
accuracy_glm<- confusionMatrix(data = y_hat_glm,reference = test_y)$overall["Accuracy"]

##### LDA and QDA models #####
# Train of LDA and QDA models
set.seed(1996)
fit_lda<- train(x = train_x,y = train_y,method = "lda")
fit_qda<- train(x = train_x,y = train_y,method = "qda")

y_hat_lda<- predict(object = fit_lda,newdata = test_x) |>
  factor(levels=levels(test_y))
y_hat_qda<- predict(object = fit_qda,newdata = test_x) |>
  factor(levels=levels(test_y))

# accuracy of lda
accuracy_lda<- confusionMatrix(data = y_hat_lda,reference = test_y)$overall["Accuracy"]

# accuracy of qda
accuracy_qda<- confusionMatrix(data = y_hat_qda,reference = test_y)$overall["Accuracy"]

##### Loess Model #####

# Training of gamLoess with default tuning parameters
set.seed(1996)
if(!require(gam)) install.packages("gam", repos = "http://cran.us.r-project.org")
library(gam)

fit_loess<- train(x = train_x,y = train_y,method = "gamLoess")

# prediction with loess model
y_hat_loess<- predict(object = fit_loess,newdata=test_x) |>
  factor(levels=levels(test_y))

# Best tuning parameters
fit_loess$bestTune

# accuracy of gamLoess
accuracy_loess<- confusionMatrix(data = y_hat_loess,reference = test_y)$overall["Accuracy"]

##### K- nearest Nieghbors #####

# Training of a knn model
set.seed(1996)
fit_knn<- train(x = train_x,y = train_y,method = "knn",tuneGrid=data.frame(k=seq(3,21,2)))

# prediction with knn model
```

```r
y_hat_knn<- predict(object = fit_knn,newdata=test_x) |>
  factor(levels=levels(test_y))

# Best tuning parameters
fit_knn$bestTune

# accuracy of knn
accuracy_knn<- confusionMatrix(data = y_hat_knn,reference = test_y)$overall["Accuracy"]


##### K-nearest Neighbors with PCA #####
# Training of a knn model with pca
pca<- prcomp(x = train_x)

# Obtain the minimum number of principal components required to explain at least 90% of the variance
p<- first(which((cumsum(pca$sdev^2)/sum(pca$sdev^2))>0.9))

# using the number of PCAs obtained, predict using a knn model
set.seed(1996)
fit_knn_pca<- train(x = train_x,y = train_y,method = "knn",tuneGrid=data.frame(k=seq(3,21,2)), preProces

# prediction with knn model
y_hat_knn_pca<- predict(object = fit_knn_pca,newdata = test_x) |>
  factor(levels=levels(test_y))

# accuracy of knn model with pca
accuracy_knn_pca<- confusionMatrix(data = y_hat_knn_pca,reference = test_y)$overall["Accuracy"]

##### Random Forest #####
# Training of a random forest model
set.seed(1996)
fit_rf<- train(x = train_x,y = train_y,method = "rf",tuneGrid=data.frame(mtry=c(3,5,7,9)),importance=TRU

# Best tuning parameters
fit_rf$bestTune

# prediction with random forest model
y_hat_rf<- predict(object = fit_rf,newdata = test_x) |>
  factor(levels=levels(test_y))

# accuracy of random forest
accuracy_rf<- confusionMatrix(data = y_hat_rf,reference = test_y)$overall["Accuracy"]
accuracy_rf

# 5 Most important variables
varImp(object = fit_rf)$importance |>
  slice_max(order_by = B,n = 5)

# least important variable
varImp(object = fit_rf)$importance |>
  slice_min(order_by = B,n = 1)
```

```r
##### Ensemble model #####
# Create an ensemble model
ensemble<- data.frame(y_hat_kmeans,y_hat_glm,y_hat_lda,y_hat_qda,y_hat_loess,y_hat_knn,y_hat_knn_pca, y
  as.matrix()

# prediction with ensemble
y_hat_ensemble<- apply(X = ensemble,MARGIN = 1,FUN = function(x){
  ifelse(mean(x=="B")>0.5,"B","M") |>
    factor(levels=levels(test_y))
})

# accuracy of ensemble
accuracy_ensemble<- confusionMatrix(data = y_hat_ensemble,reference = test_y)$overall["Accuracy"]

# summary of models with corresponding accuracies
model_summary<- data.frame(model=c("kmeans","glm","lda","qda","knn","knn with pca", "rf","ensemble"), a

kable(model_summary)

# model with the best accuracy
model_summary$model[which.max(model_summary$accuracy)]

# Use the ensemble model to evaluate the validation set

# k-means prediction
y_hat_kmeans_v<- predict_kmeans(x = validation_x,k = k)
y_hat_kmeans_v<- ifelse(y_hat_kmeans_v==1,                          "M","B") |>
  factor(levels=levels(test_y))

# glm prediction
y_hat_glm_v<- predict(object = fit_glm,newdata = validation_x) |>
  factor(levels=levels(test_y))

# qda and lda prediction
y_hat_lda_v<- predict(object = fit_lda,newdata = validation_x) |>
  factor(levels=levels(test_y))
y_hat_qda_v<- predict(object = fit_qda,newdata = validation_x) |>
  factor(levels=levels(test_y))

# loess prediction
y_hat_loess_v<- predict(object = fit_loess,newdata=validation_x) |>
  factor(levels=levels(test_y))

# knn prediction
y_hat_knn_v<- predict(object = fit_knn,newdata=validation_x) |>
  factor(levels=levels(test_y))

# knn with pca prediction
y_hat_knn_pca_v<- predict(object = fit_knn_pca,newdata = validation_x) |>
  factor(levels=levels(test_y))

# rf prediction
y_hat_rf_v<- predict(object = fit_rf,newdata = validation_x) |>
```

```
  factor(levels=levels(test_y))

# ensemble prediction
ensemble_v<- data.frame(y_hat_kmeans_v,y_hat_glm_v,y_hat_lda_v,y_hat_qda_v,y_hat_loess_v,y_hat_knn_v,y_h
  as.matrix()
y_hat_ensemble_v<- apply(X = ensemble_v,MARGIN = 1,FUN = function(x){
  ifelse(mean(x=="B")>0.5,"B","M") |>
    factor(levels=levels(test_y))
})

# final accuracy on the validation set
accuracy_validation<- confusionMatrix(data = y_hat_ensemble_v, reference = validation_y)$overall["Accura

# stop parallelization
stopCluster(cl)
stopImplicitCluster()
```